

УДК 519.178

ВЕРОЯТНОСТНЫЙ ПОИСК С ЗАПРЕТАМИ ДЛЯ ЗАДАЧИ УПАКОВКИ КРУГОВ И ПРЯМОУГОЛЬНИКОВ В ПОЛОСУ *)

А. С. Руднев

Аннотация. Рассматривается задача двумерной упаковки кругов и прямоугольников различных размеров в полубесконечную полосу минимальной длины. Представлена математическая постановка задачи в терминах нелинейного целочисленного программирования. Предложена кодирующая схема для двухконтактных решений. На её основе разработан вероятностный алгоритм поиска с запретами для нахождения приближённого решения задачи. Численные эксперименты на случайно сгенерированных примерах, а также на известных тестовых примерах для частных случаев рассматриваемой задачи показали эффективность алгоритма. Для четырёх известных примеров упаковки кругов в полосу удалось найти новые рекордные значения целевой функции.

Ключевые слова: упаковка в полосу, кодирующие схемы, вероятностный поиск с запретами.

Введение

В статье рассматривается задача упаковки конечного множества кругов и прямоугольников в полубесконечную полосу заданной ширины. Каждый круг задаётся радиусом, а каждый прямоугольник задаётся длиной и шириной. Требуется разместить в полосе все предметы без пересечений (касание границ допускается) 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 NP-трудной задачи одномерной упаковки в контейнеры [2] и также является NP-трудной в сильном смысле.

Рассматриваемая задача относится к классу задач двумерного раскроя и упаковки. Известны различные постановки задач из этого класса

*) Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований (проект 08-07-00037) и АВЦП Рособразования (проект 2.1.1/3235).

[3, 4, 11, 27]. Для упаковки только прямоугольников, либо только кругов получено немало результатов. Однако задача упаковки одновременно кругов и прямоугольников остаётся малоизученной. Например, в [15] рассматривается задача упаковки в полосу кругов, прямоугольников и более сложных фигур различных размеров. Для её решения используется генетический алгоритм, основанный на эвристической процедуре преобразования перестановки предметов в упаковку. В [31] предлагается алгоритм для нахождения глобального оптимума задачи упаковки в полосу кругов и прямоугольников и её обобщений. Предложенный подход является комбинацией метода ветвей и границ и метода градиентов.

Наиболее известные постановки задач двумерной прямоугольной упаковки связаны с упаковкой различных прямоугольных предметов в полосу минимальной длины, либо в минимальное число одинаковых прямоугольных контейнеров. Самым изученным является ориентированный случай задачи, когда повороты предметов запрещены и стороны предметов параллельны сторонам контейнеров или полосы. Подробный обзор, посвящённый моделям, нижним оценкам, приближённым и точным алгоритмам, а также реализациям метаэвристик для ориентированного случая задачи прямоугольной упаковки в полосу, либо в контейнеры можно найти в [27]. Другие варианты задачи, когда допускаются повороты предметов на 90 градусов, а также ограничения на размещение предметов, такие как «гильотинный раскрой», подробно освещаются в [28, 29]. Из последних результатов особый интерес представляют следующие работы. В [8] рассматривается задача упаковки прямоугольников в выпуклую область. Данная задача формулируется в работе в терминах нелинейного программирования. Для её решения авторы используют численные методы. В [9] предлагается генетический алгоритм для решения как ориентированного, так и неориентированного случаев задачи прямоугольной упаковки в полосу. Для этой же задачи в [6] разработан гиперэвристический алгоритм, объединяющий в себе принципы нескольких эвристик. В [34] 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 [5] предлагается метод GRASP, а в [21] — алгоритм локального поиска, использующий процедуру вычисления целевой функции, основанную на динамическом программировании. В [18] для решения задачи прямоугольной упаковки в контейнер заданного размера предлагается жадный полиномиальный алгоритм, который авторам также удалось эффективно применить к за-

даче прямоугольной упаковки в полосу. Что касается точных методов, то в [25] 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 [32] рассматривается задача упаковки кругов различных размеров в полосу минимальной длины и предлагается алгоритм для её точного решения. Существует ряд статей, посвящённых задаче упаковки различных кругов в двумерный рюкзак, в качестве которого выступает прямоугольник, либо другая геометрическая фигура заданного размера. Цель задачи — так заполнить рюкзак предметами из списка, чтобы минимизировать неиспользуемое в рюкзаке место. Например, в [13] рассматривается задача упаковки труб разного радиуса в прямоугольный контейнер, что эквивалентно упаковке различных кругов в прямоугольный рюкзак. Для её 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 [19, 20] авторы предлагают жадные полиномиальные алгоритмы упаковки различных кругов в заданный контейнер прямоугольной или круглой формы. В [16] авторы рассматривают задачу упаковки в прямоугольный рюкзак круглых деталей нескольких типов с учётом максимального спроса на них. Для нахождения приближённого решения предлагается генетический алгоритм.

Также существуют постановки, где требуется минимизировать площадь, в которой размещаются предметы. Например, в [30] рассматривается задача об упаковке одинаковых кругов в прямоугольник минимальной площади. Для её решения в работе приводятся два алгоритма: один имитирует физическое взаимодействие кругов под действием силы, направленной на сжатие прямоугольника, другой алгоритм основан на переборе возможных шаблонов упаковки кругов. В [33] 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 её решения. Из последних работ отметим [23], в которой приводится математическая поста-

новка 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 настоящей статье 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 приближённого решения задачи. Работа организована следующим образом. В разд. 1 приводится математическая постановка задачи. В разд. 2 вводится понятие двухконтактной схемы кодирования, рассматриваются некоторые её свойства. В разд. 3 определяются окрестности решений. В разд. 4 описывается разработанный вероятностный алгоритм поиска с запретами. В разд. 5 приводятся результаты численных экспериментов.

1. Математическая постановка задачи

Обозначим исходные данные задачи следующими параметрами: W — ширина полосы, N^c — число кругов, N^r — число прямоугольников, $I = \{1, 2, \dots, N^c\}$ — множество кругов, $J = \{1, 2, \dots, N^r\}$ — 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.

Введём переменные задачи. Пусть x_i^c и y_i^c — координаты центра i -го круга, x_j^r и y_j^r — координаты левого нижнего угла j -го прямоугольника. Для учёта поворотов прямоугольников на 90 градусов введём бинарную переменную z_j , принимающую значение 1, если j -й прямоугольник повернут на 90 градусов, и значение 0 в противном случае. В качестве целевой функции будет выступать длина занятой части полосы L : найти $\min L$ при ограничениях

$$x_i^c + r_i \leq L, \quad i \in I, \quad (x_j^r + l_j)(1 - z_j) + (y_j^r + w_j)z_j \leq L, \quad j \in J.$$

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

$$x_i^c - r_i \geq 0, \quad i \in I, \quad y_i^c - r_i \geq 0, \quad i \in I, \quad y_i^c + r_i \leq W, \quad i \in I.$$

По аналогии выпишем соответствующие условия для прямоугольников:

$$x_j^r(1 - z_j) + y_j^r z_j \geq 0, \quad j \in J,$$

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

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

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

Следующие пять условий исключают пересечения прямоугольников между собой. Чтобы два прямоугольника не пересекались, необходимо и достаточно, чтобы один был левее, правее, выше или ниже другого, т. е. должно выполняться хотя бы одно из первых четырёх неравенств следующей группы. Дизъюнкция этих неравенств реализуется с помощью пятого условия, использующего бинарные переменные a_{jl}^m :

$$\begin{aligned} ((x_j^r + l_j)(1 - z_j) + (y_j^r + w_j)z_j)a_{jl}^1 &\leq x_l^r(1 - z_l) + y_l^r z_l, \quad l, j \in J, \\ ((x_l^r + l_l)(1 - z_l) + (y_l^r + w_l)z_l)a_{jl}^2 &\leq x_j^r(1 - z_j) + y_j^r z_j, \quad l, j \in J, \\ ((y_j^r + w_j)(1 - z_j) + (x_j^r + l_j)z_j)a_{jl}^3 &\leq y_l^r(1 - z_l) + x_l^r z_l, \quad l, j \in J, \\ ((y_l^r + w_l)(1 - z_l) + (x_l^r + l_l)z_l)a_{jl}^4 &\leq y_j^r(1 - z_j) + x_j^r z_j, \quad l, j \in J, \end{aligned}$$

$$\sum_{m=1}^4 a_{jl}^m = 1, \quad l, j \in J,$$

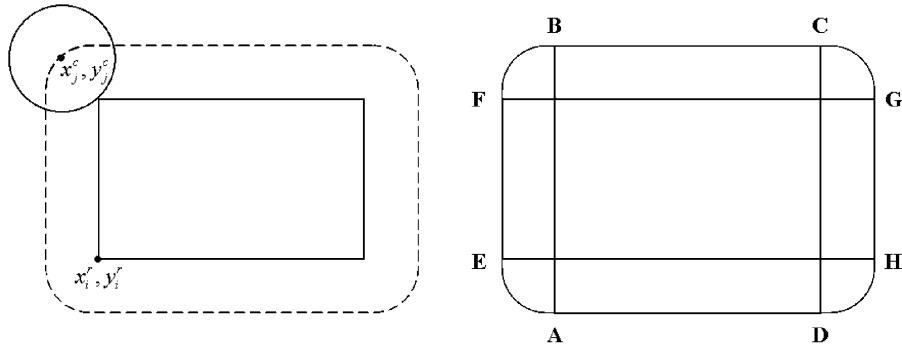


Рис. 1. Условие непересечения прямоугольника и круга

Остальные условия модели 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. 1 (слева). Если центр круга

лежит на границе, то фигуры соприкасаются. Следовательно, необходимо выписать условия, которые исключают нахождение точки x_i^c, y_i^c в 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 . Таким образом, точка x_i^c, y_i^c не должна лежать ни внутри одного из четырёх кругов, ни внутри прямоугольника ABCD, ни внутри прямоугольника EFGH.

Следующая группа неравенств гарантирует, что центр круга x_i^c, y_i^c лежит за пределами двух покрывающих прямоугольников для каждого j -го прямоугольника. Здесь также возникают дизъюнкции неравенств, реализующиеся с помощью бинарных переменных b_{ij}^m и c_{ij}^m :

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

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

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

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

Поставленная задача относится к классу нелинейных целочисленных 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 приближённого алгоритма, способного за небольшое время находить решения, близкие к оптимальным, для примеров достаточно большой размерности. Алгоритмы, называемые *метаэвристиками*, относятся к одному из эффективных с этой точки зрения классов алгоритмов [12, 14]. 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 неё также зависит трудоёмкость каждого шага алгоритма. Следующие два раздела посвящены определению этих понятий для рассматриваемой задачи.

2. Двухконтактные кодировки

Важную роль в задачах упаковки играет способ представления (кодирования) допустимых решений. Самая естественная 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 её использования и кодировать решения с помощью более сложных структур данных. Для задач двумерной прямоугольной упаковки разработано множество различных кодировок, отличающихся как трудоёмкостью 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 [17] рассматриваются два таких алгоритма: Bottom-Left с трудоёмкостью $O(n^2)$ и Bottom-Left-Fill с трудоёмкостью $O(n^3)$, где

n — число предметов. Также в [27–29] 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. Посвящённый этой теме обзор содержится в [26]. Для задачи упаковки кругов в [13] 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 [15, 16] похожая идея используется для задач упаковки кругов, прямоугольников и предметов более сложной формы. Порядок предметов задаётся 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 трудоёмкость декодирования. В данной статье предложена схема кодировки двухконтактных решений, основанная на перестановке предметов и использующая более простой и менее трудоёмкий алгоритм декодирования, чем в [15, 16]. Упаковка предметов при заданной перестановке осуществляется следующим образом.

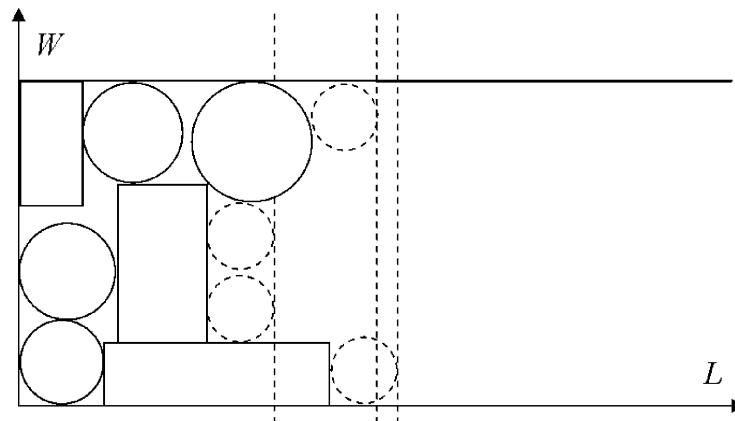


Рис. 2. Построение двухконтактного решения

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 поло-

сы. Пересечения предметов и их выход за границы полосы не допускаются. Если таких положений несколько для очередного предмета, то выбирается положение с минимальным увеличением длины полосы (рис. 2). Прямоугольные предметы при поиске допустимых положений рассматриваются как в их начальной ориентации, так и с поворотом на 90 градусов. Таким образом, ориентация прямоугольных предметов определяется в процессе декодирования. В случае, когда минимальному увеличению длины полосы соответствует несколько положений, для определённости 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 $n!$.

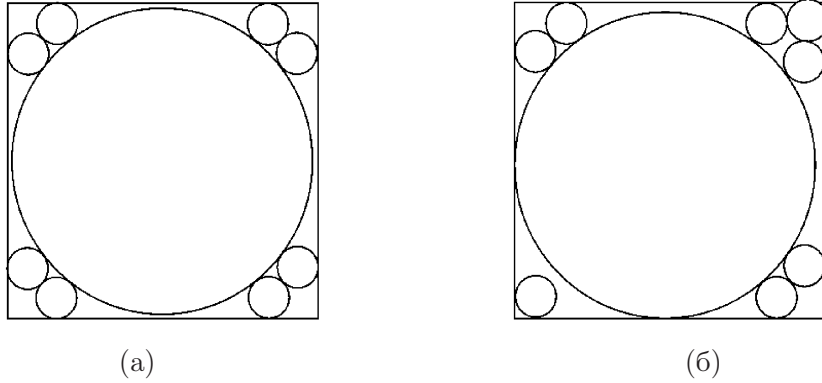


Рис. 3. Оптимальное решение (а) и двухконтактное решение (б)

Заметим, что множество двухконтактных упаковок может не содержать оптимального решения задачи. На рис. 3 показан квадрат со стороной $2(R + \varepsilon)$, в центре которого размещен круг радиуса R , а в углах — восемь кругов радиуса $r = (5 - \sqrt{2})R + (2 - \sqrt{2})\varepsilon - \sqrt{2(10 - 7\sqrt{2})(2R + \varepsilon)R}$. Параметры R и ε выбраны так, чтобы выполнялось условие

$$R + 2\varepsilon < 3r + 2\sqrt{(R + \varepsilon)(r - \varepsilon)},$$

что гарантирует единственность способа упаковки предметов в этом квадрате. Это решение не является двухконтактным, так как в левом нижнем углу нет предмета, касающегося двух сторон квадрата. Наилучшее двухконтактное решение показано на рис. 3 (б). Однако оно не вмещает в себя круг, расположенный в правом верхнем углу, и требует увеличения длины полосы. Таким образом, поиск в области двухконтактных упаковок

не гарантирует нахождения оптимального решения задачи. Тем не менее, эта кодировка даёт экспоненциальное число кодов. И дальнейшие усилия будут направлены на поиск наилучшего решения в этом множестве.

В [24] исследовался вопрос о сравнении описанной двухконтактной кодировки для упаковки кругов в полосу с другой, более мощной двухконтактной кодировкой из [13], мощность которой составляла

$$O((n!(n+1)!(n+2)!)/2^{n+1}).$$

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 только 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 остаётся открытым.

3. Окрестности

Введём операции над перестановками, с помощью которых строятся окрестности решений. Выбор окрестности играет важную роль при построении алгоритмов локального поиска. От него существенно зависит трудоёмкость одного шага алгоритма, общее число шагов и, в конечном счёте, 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 .

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 даёт худшие результаты, чем использование обеих. Операции оказывают разное влияние на упаковку предметов, соответствующую некоторой перестановке, и в совокупности порождают достаточно эффективную окрестность мощностью $O(n^2)$. Заметим, что для двух произвольных перестановок можно получить одну из другой, применив конечное

число операций СДВИГ и ЗАМЕНА. Другими словами, можно обойти все пространство кодов, перемещаясь от одного решения к соседнему.

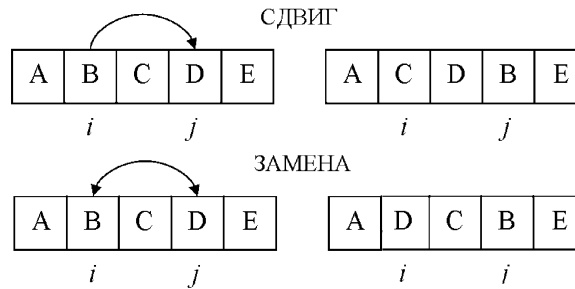


Рис. 4. Действие операций СДВИГ и ЗАМЕНА

4. Поиск с запретами

Алгоритм, разработанный для решения задачи упаковки кругов и прямоугольников в полосу, основан на принципах поиска с запретами, основоположником которого является Ф. Гловер [14]. Алгоритм поиска с запретами представляет собой вероятностную процедуру локального поиска и принадлежит классу так называемых метаэвристик, описанных в общих терминах и применимых ко многим задачам дискретной оптимизации. Все метаэвристики являются итерационными процедурами. Для многих из них установлена асимптотическая сходимость наилучшего найденного решения к глобальному оптимуму [1]. Кроме поиска с запретами к числу метаэвристик относятся генетические алгоритмы, алгоритм имитации отжига, нейронные сети, муравьиные колонии, вероятностные жадные процедуры и др. [12]. Для работы алгоритмов локального поиска требуется определить окрестность решения. В нашем случае окрестностью решения i_k будем называть множество решений $N(i_k)$, полученных путём 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 её, 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 $TABU_l(i_k)$. 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 $f(i_{k+1}) = \min\{f(j) \mid j \in N_P(i_k) \setminus TABU_l(i_k)\}$, где $f(i)$ — функционал, выступающий критерием качества решения. Окрестность текущего решения ограничивается теми решениями, которые не запрещены списком $TABU_l(i_k)$. 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. 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 заиклиться. При длинном списке мо-

жет оказаться так, что большая часть окрестности будет запрещена, что также не приведёт 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 $\text{TABU}_l(i_0) := \emptyset$, $i^* := i_0$, $L^* := L(i^*)$, $i^r := i_0$, $L^r := L(i^r)$, $k := 0$, $P := P_{\min}$, $\text{sgn} := +1$.
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 $L(i_{k+1}) = \min_{j \in N_P(i_k) \setminus \text{TABU}_l(i_k)} L(j)$.
 - 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 $\text{sgn} := +1$.
 - 3.1.5. Положить $k := k + 1$ и обновить $\text{TABU}_l(i_k)$.
 - 3.2. Если $\text{sgn} = +1$, то $i_k := i^r$.
 - 3.3. Положить $P := P + \text{sgn} \cdot \Delta P$.
 - 3.4. Если $P \geq P_{\max}$, то $\text{sgn} := -1$.
 - 3.5. Если $P \leq P_{\min}$, то $\text{sgn} := +1$, $L^r := L(i_k)$ 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, $\text{TABU}_l(i_k)$ — список запретов на k -й итерации, P — параметр рандомизации окрестности, $\text{sgn} \in \{-1, +1\}$ — 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 $N_P(i) \setminus \text{TABU}_l(i)$. Критерием качества решения выступает длина занятого участка полосы L . В процессе поиска параметр

рандомизации P меняется в пределах $[P_{\min}, \dots, P_{\max}]$ на величину ΔP в зависимости от того, как часто встречаются решения малой относительной погрешности. Таким способом осуществляется интенсификация и диверсификация поиска [14]. 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, с помощью увеличения значения P , а вместе с ним доли просматриваемой окрестности, и возвращения 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 удаётся 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 удаётся улучшить значение целевой функции, либо общее число итераций.

5. 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 трёх 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 путём 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.

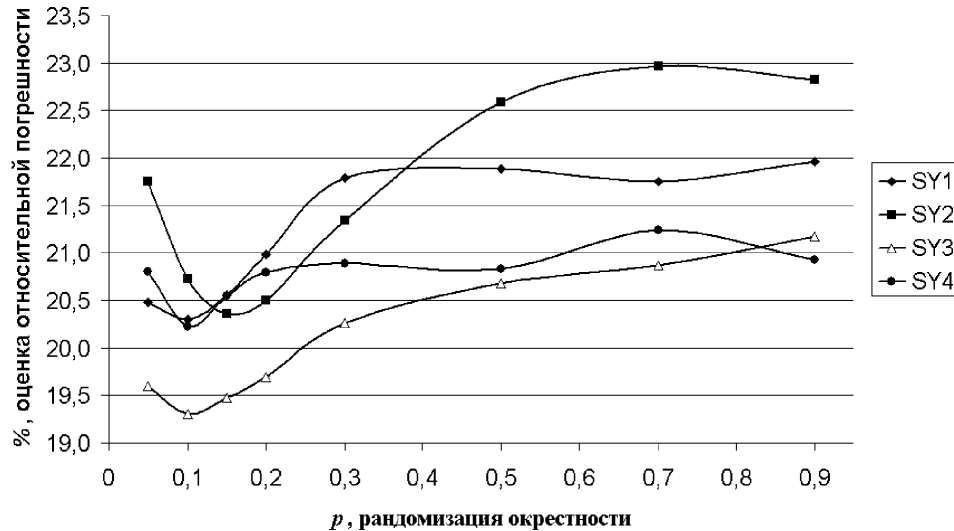


Рис. 5. Влияние рандомизации окрестности на относительную погрешность решений

5.1. Влияние рандомизации и длины списка запретов. В [1] 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 дальнейших вычислений. Для этого необходимо выяснить, как влияют размер доли рассматриваемой окрестности и длина списка запретов на качество получаемых решений. В результате численных расчётов 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.

В табл. 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, W — ширина полосы. Были взяты первые четыре примера из этой таблицы. Значение параметра рандомизации при каждом запуске алгоритма оставалось постоянным и в процессе поиска не менялось. Расчёты 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. 5 показана 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 трудоёмкость одной итерации. Следовательно, при большой рандомизации алгоритм вырождается в случайное блуждание, а при малой поведение алгоритма будет похожем на стандартный локальный спуск.

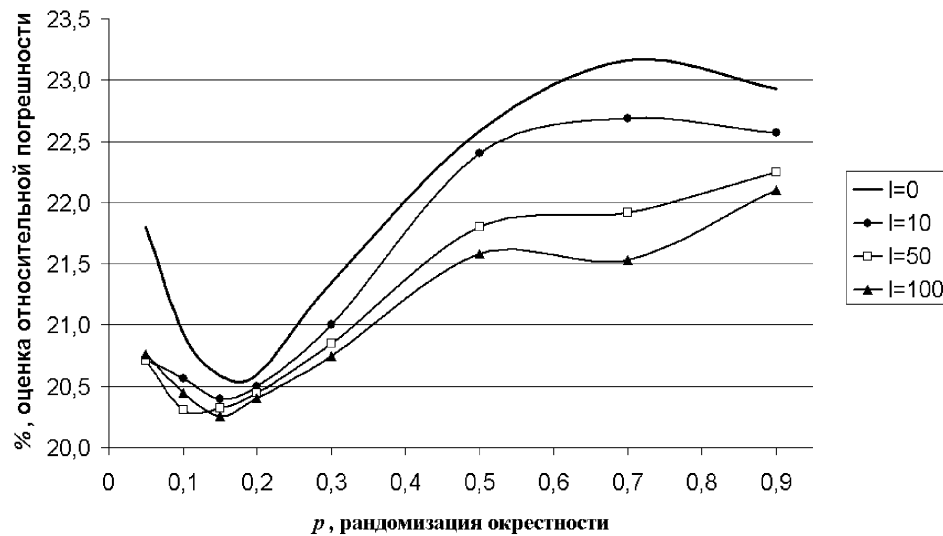


Рис. 6. Влияние длины списка запретов на относительную погрешность решений

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. 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.

5.2. 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 [15, 23, 31], в литературе для неё не приводится тестовых примеров. Поэтому примеры были сгенерированы случайным образом так, чтобы для каждого из них была известна некоторая верхняя оценка для 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 нарезаем 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. 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 (повороты прямоугольников на 90° разрешались). Начальное решение выбиралось случайным образом. Параметр рандомизации менялся от P_{\min} до P_{\max} , давая алгоритму возможность подстраиваться в процессе поиска. В предыдущем эксперименте показано, что при рандомизации 0,1 алгоритм способен за небольшое время найти достаточно хорошие решения, а начиная с 0,5, поведение алгоритма 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.

Т а б л и ц а 1

Значение длины полосы для задачи упаковки кругов и прямоугольников

Пример	Параметры примера				LB UB		TABU SEARCH		Отклонение (%)		Время (сек.)
	W	n	n_c	n_r			Среднее	Лучшее	от LB	от UB	
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	19	110	86	202,65	240	239,44	235,77	16,34	-1,76	1508

Величина шага ΔP равнялась 0,05. Количество итераций, совершаемых алгоритмом при каждом значении P , бралось обратно пропорциональным 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. 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 [15, 17, 19, 31, 32].

Т а б л и ц а 2

Длина полосы для задачи упаковки кругов

Пример	W	n_c	LB	CAGA [15]	S.Y. [31, 32]	B1.5 [19]	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 отведённое ему время, и из двух критериев остановки всегда срабатывало только ограничение по времени. В табл. 2 представлены решения генетического алгоритма CAGA [15], решения, полученные методом ветвей и границ S.Y. [31, 32], а также решения, полученные жадной эвристической процедурой B1.5 [19]. 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. 2. Среднее время работы алгоритма B1.5 составляет примерно 40 минут для примеров SY1–4 и 18 часов для примеров SY5–6 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 четырёх примеров SY1–4 алгоритму удалось найти новые рекордные решения. Для двух оставшихся наилучшими найденными остаются решения, полученные алгоритмом B1.5.

Т а б л и ц а 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)	15×40
C3(C31,C32,C33)	28(C31,C33),29(C32)	30×60
C4(C41,C42,C43)	49(C41,C42,C43)	60×60
C5(C51,C52,C53)	73(C51,C52,C53)	90×60
C6(C61,C62,C63)	97(C61,C62,C63)	120×80
C7(C71,C72,C73)	196(C71,C73),197(C72)	240×160

Для задачи упаковки прямоугольников в полосу были рассмотрены примеры с известным оптимальным значением целевой функции из работы [17]. 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). Рассмотрены два варианта задачи: с поворотами прямоугольников на 90 градусов и без них. В табл. 4 и 5 для данных примеров указаны лучшие результаты, полученные описанными в литературе алгоритмами [5, 6, 9, 10, 17, 18, 21, 22, 34], 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. 4 и 5 также указано среднее время работы алгоритма на примерах из каждого класса.

Рассмотрим задачу с поворотами прямоугольников. В табл. 4 представлены результаты, опубликованные в литературе [6, 9, 17, 18, 34] и полученные разработанным алгоритмом. Генетический алгоритм GA+BLF и алгоритм имитации отжига SA+BLF [17] 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 [9] составляло 160 секунд на вычислительной машине с процессором Pentium 2 GHz, и алгоритм запускался 10 раз на каждом примере. Алгоритм GA+IHR[34] тестировался на вычислительной машине с тактовой частотой 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} [6] для каждого примера запускался 10 раз по 1000 секунд.

Т а б л и ц а 4

Относительная погрешность для неориентированной упаковки
прямоугольников в полосу

Класс	GA+BLF [17]	SA+BLF [17]	SPGAL [9]	GA+IHR [34]	HRP [18]	H-SP _{1000s} [6]	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

Т а б л и ц а 5

Относительная погрешность для ориентированной упаковки
прямоугольников в полосу

Класс	Iori [22]	BF+SA [10]	SPGAL [9]	HM-SP [21]	GRASP [5]	H-SP _{1000s} [6]	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

В табл. 5 показаны результаты, полученные для ориентированной задачи прямоугольной упаковки в полосу [5, 6, 9, 10, 21, 22]. Алгоритм Iori [22] работал 300 секунд на каждом примере на машине с процессором Pentium III 800 Mhz. BF+SA [10] и GRASP [5] запускались 10 раз для каждого примера на компьютере Pentium 2 Ghz с ограничением по времени — 60 секунд на один запуск. Условия работы алгоритмов

SPGAL [9] и H-SP_{1000s} [6] для ориентированной задачи были такими же, как для неориентированной. Среднее время работы метаэвристики НМ-SP [21] 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. 4 и 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.

5.3. Использование пакета GAMS. В этой части приводятся результаты, полученные с помощью коммерческого пакета GAMS (<http://www.gams.com/>). Указанный пакет использует методы глобальной и локальной оптимизации и находит оптимальные решения для ряда задач, записанных в терминах математического программирования. Решение различных типов задач осуществляется с помощью подключаемых к GAMS решателей. Среди них такие известные, как BARON, CPLEX, LINDOGLOBAL, XPRESS и др. Целью эксперимента было выяснить, насколько эффективен данный пакет применительно к рассматриваемым в работе задачам. Рассмотрены примеры из предыдущего раздела. На каждом примере пакет работал 20 часов. В качестве решающего ядра выбран BARON, работа которого исследовалась в [23] на примере задачи упаковки кругов и выпуклых многоугольников. Это один из немногих решателей, способных находить глобальные оптимумы для задач нелинейной целочисленной оптимизации.

Т а б л и ц а 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	39	нет	20	да
	C1P2	37	нет	20	да
	C1P3	32	нет	20	да

Неориентированная задача упаковки кругов и прямоугольников в полосу, модель которой представлена в начале работы, оказалась для GAMS слишком сложной. В результате за отведённое 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 примеров SY1–4 упаковки кругов и для трёх примеров из класса C1 ориентированной упаковки прямоугольников. При решении последних двух задач пакетом использовались математические модели, имеющие более простой вид, чем общая.

В табл. 6 представлены результаты, полученные пакетом GAMS. Оптимальные решения, найденные для трёх примеров задачи упаковки кругов и прямоугольников, совпадают с решениями, полученными алгоритмом поиска с запретами. Качество допустимых решений, полученных для частных случаев задачи, заметно уступает решениям, полученным с помощью вероятностного поиска с запретами за гораздо меньшее время.

6. Заключение

Рассмотрена задача упаковки кругов и прямоугольников в полосу. Для её решения 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 нашёл новые рекордные значения целевой функции для четырёх примеров упаковки кругов в полосу. Представляет интерес разработка новой, менее трудоёмкой 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 [7] для задачи прямоугольной упаковки в контейнеры с запрещёнными областями. Это позволило сократить трудоёмкость декодирования на порядок и существенно ускорить работу алгоритма локального поиска, особенно на примерах большой размерности.

В заключение автор выражает благодарность рецензентам за полезные замечания, которые помогли улучшить текст этой работы.

ЛИТЕРАТУРА

1. **Гончаров Е. Н., Кочетов Ю. А.** Вероятностный поиск с запретами для дискретных задач безусловной оптимизации // Дискрет. анализ и исслед. операций. Сер. 2. — 2002. — Т. 9, № 2. — С. 13–30.
2. **Гэри М. Р., Джонсон Д. С.** Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982. — 416 с.
3. **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.
4. **Стоян Ю. Г., Яковлев С. В.** Математические модели и оптимизационные методы геометрического проектирования. — Киев: Наук. думка, 1986. — 266 с.
5. **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.
6. **Araya I., Neveu B., Riff M. C.** An efficient hyperheuristic for strip-packing problems // Studies in Comput. Intelligence. — 2008. — Vol. 136. — P. 61–76.
7. **Beisiegel B., Kallrath J., Kochetov Y., Rudnev A.** Simulated annealing based algorithm for the 2D bin packing problem with impurities // Operations research proceedings, 2005. — Bremen: Springer-Verl., 2005. — P. 109–113.
8. **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.
9. **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.
10. **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.
11. **Dowsland K. A., Dowsland W. B.** Packing problems // Europ. J. Oper. Res. — 1992. Vol. 56. — P. 2–14.
12. **Dreo J., Petrowski A., Siarry P., Taillard E.** Metaheuristics for hard optimization: methods and case studies. — Berlin: Springer-Verl., 2005. — 369 p.
13. **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.

14. **Glover F., Laguna M.** Tabu search. — Dordrecht: Kluwer Acad. Publ., 1997. — 382 p.
15. **Hifi M., M'Hallah R.** A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes // Intern. Transaction in Oper. Res. — 2003. — Vol. 10. — P. 195–216.
16. **Hifi M., M'Hallah R.** Approximate algorithms for constrained circular cutting problems. // Comput. Oper. Res. — 2004. — Vol. 31. — P. 675–694.
17. **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.
18. **Huang W., Chen D., Xu R.** A new heuristic algorithm for rectangle packing // Comput. Oper. Res. — 2007. — Vol. 34. — P. 3270–3280.
19. **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.
20. **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.
21. **Ibaraki T., Imahori S., Yagiura M.** Hybrid metaheuristics for packing problems // Studies in Comput. Intelligence. — 2008. — Vol. 114. — P. 185–219.
22. **Iori M., Martello S., Monaci M.** Metaheuristic algorithms for the strip packing problem, — Dordrecht: Kluwer Acad. Publ., 2003. — P. 159–179.
23. **Kallrath J.** Cutting circles and polygons from area-minimizing rectangles // J. Global Optimization. — 2009. — Vol. 43. — P. 299–328.
24. **Kallrath J., Kochetov Y., Rudnev A.** Strip packing problem for circles and rectangles // 4th ESICUP Meeting. — Tokyo: University of Tokyo, 2007. — P. 20.
25. **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.
26. **Lin J. M., Chang Y. W.** TCG: A transitive closure graph-based representation for non-slicing floorplans // Proc. DAC, 2001. — Las Vegas: ACM, 2001. — P. 764–769.
27. **Lodi A., Martello S., Monaci M.** Two-dimensional packing problems: a survey // Europ. J. Oper. Res. — 2002. — Vol. 141. — P. 241–252.
28. **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.
29. **Lodi A., Martello S., Vigo D.** Recent advances on two-dimensional bin packing problems // Discrete Appl. Math. — 2002. — Vol. 123/124. — P. 373–380.
30. **Lubachevsky B. D., Graham R.** Dense packing of congruent circles in rect-

angles with a variable aspect ratio. Discrete and computational geometry — the Goodman — Pollack Festschrift // Algorithms and combinatorics. — Heidelberg: Springer-Verl., 2003. — P. 633–650.

31. **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 // Intern. Trans. Oper. Res. — 1998, — Vol. 5, N 1. — P. 45–57.
32. **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.
33. **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.
34. **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.

Руднев Антон Сергеевич,
e-mail: anton.rudnev@gmail.com

Статья поступила
5 ноября 2008 г.
Переработанный вариант —
29 мая 2009 г.