

УДК 519.8

## АЛГОРИТМ ИМИТАЦИИ ОТЖИГА ДЛЯ РЕШЕНИЯ ЗАДАЧ ДВУМЕРНОЙ ПРЯМОУГОЛЬНОЙ УПАКОВКИ В КОНТЕЙНЕРЫ С ЗАПРЕЩЁННЫМИ ОБЛАСТЯМИ \*)

*А. С. Руднев*

**Аннотация.** Рассматривается задача двумерной прямоугольной упаковки в контейнеры с запрещёнными областями. Данная задача обобщает известную NP-трудную задачу упаковки в контейнеры. Разработаны кодирующие схемы для представления решений гильотинной и негильотинной задач, учитывающие специфику задачи с запрещёнными областями. На их основе разработан алгоритм имитации отжига для нахождения приближённого решения задачи. Начальное решение строится при помощи жадной эвристики, что позволяет начать поиск с низкой температуры. При каждой смене температуры осуществляется процедура уплотнения. Она позволяет концентрировать небольшие предметы на отдельных листах, что облегчает их разгрузку. Проведённые численные эксперименты свидетельствуют о высокой эффективности новых кодировок и о малой погрешности получаемых решений.

**Ключевые слова:** упаковка в контейнеры, кодирующая схема, имитация отжига.

### 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 запрещённых областей прямоугольной формы с фиксированными координатами. Каждый предмет имеет свойство, позволяющее ему либо пересекаться с запрещёнными областями, либо нет. Кроме того, имеется достаточное число контейнеров без запрещённых областей с одинаковыми размерами. Требуется с учётом запрещённых областей разместить

---

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

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 удаётся разместить в заданном списке контейнеров с запрещёнными областями, то используются дополнительные контейнеры без запрещённых областей. Предполагается, что при размещении предметы не пересекаются друг с другом и стороны предметов параллельны сторонам контейнеров. Данная задача имеет приложения при обработке листов прокатной стали [11] и является обобщением хорошо известной двумерной задачи упаковки в контейнеры [8].

Рассматриваются четыре варианта поставленной задачи двумерной упаковки в контейнеры (2BP): ориентированная задача (O) — повороты предметов не допускаются; неориентированная задача (R) — допускаются повороты предметов на 90 градусов; гильотинная задача (G) — каждый предмет может быть «вырезан» из контейнера при помощи рекурсивных разрезов от края до края; негильотинная задача (F) — допускаются произвольные размещения предметов в контейнерах.

Работа организована следующим образом. В разд. 1 приводится математическая постановка задачи. В разд. 2 описываются кодирующие схемы. В разд. 3 приводится схема модифицированного алгоритма имитации отжига для рассматриваемой задачи, определяются окрестности решений. В разд. 4 и 5 описываются алгоритмы построения начального решения и уплотнения упаковки. В разд. 7 приводятся результаты численных экспериментов.

## 1. Математическая постановка задачи

В данном разделе приводятся математические постановки задачи с произвольными и гильотинными разрезами. Вначале дадим математическую постановку задачи 2BP|R|F с запрещёнными областями.

Введём следующие обозначения:  $I = \{1, 2, \dots, n\}$  — множество предметов;  $w_i$  и  $h_i$ ,  $i \in I$ , — ширина и высота  $i$ -го предмета;  $d_i$  — бинарный параметр, принимающий значение 1, если  $i$ -й предмет может пересекаться с запрещёнными областями, и значение 0 в противном случае. Пусть  $M = M_1 \cup M_2$  — множество контейнеров;  $W_m$  и  $H_m$ ,  $m \in M$ , — ширина и высота  $m$ -го контейнера. Здесь  $M_1 = \{1, 2, \dots, l\}$  — это множество контейнеров с запрещёнными областями, а  $M_2 = \{1, 2, \dots, n\}$  — дополнительные контейнеры одинакового размера без запрещённых областей. Предполагается, что ширина и высота дополнительных контейнеров не меньше максимальных ширины и высоты предметов. Поэтому достаточ-

но  $n$  дополнительных контейнеров для существования допустимого решения задачи. Будем считать, что левый нижний угол каждого контейнера находится в начале координат. Обозначим множество запрещённых областей через  $I^0 = \{1, 2, \dots, K\}$ , а их координаты и размеры через  $x_k^0, y_k^0$  и  $w_k^0, h_k^0$ ,  $k \in I^0$ . Бинарный параметр  $q_{km}$  равен единице, если  $k$ -я запрещённая область находится в  $m$ -м контейнере, и нулю в противном случае.

Введём переменные задачи. Пусть  $x_i$  и  $y_i$  — координаты левого нижнего угла  $i$ -го предмета. Бинарная переменная  $z_i$  принимает значение 1, если  $i$ -й предмет повернут на 90 градусов, и значение 0 в противном случае. Пусть бинарные переменные  $a_{ij}^L, a_{ij}^B$ ,  $i, j \in I$ , равны единице, если  $i$ -й предмет находится левее или 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 1, если  $i$ -й предмет находится соответственно левее, правее, ниже, либо выше  $k$ -й запрещённой области, и значение нуль в противном случае. Пусть бинарные переменные  $p_{im}$ ,  $i \in I, m \in M$ , 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. Бинарная переменная  $u_m$  принимает значение 1, если  $m$ -й контейнер используется, т. е. содержит хотя бы один предмет, и значение 0, если не используется. Тогда целевая функция, равная количеству используемых контейнеров, имеет вид

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

Зададим порядок использования контейнеров

$$u_m \geq u_{m+1}, \quad m \in M. \quad (2)$$

Использование контейнеров определяется с помощью неравенства

$$u_m n \geq \sum_{i=1}^n p_{im}, \quad m \in M. \quad (3)$$

Условие

$$\sum_{m=1}^{l+n} p_{im} = 1, \quad i \in I, \quad (4)$$

гарантирует, что каждый предмет находится только в одном контейнере.

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

$$x_i \geq 0, \quad y_i \geq 0, \quad i \in I, \quad (5)$$

$$x_i + w_i(1 - z_i) + h_i z_i \leq \sum_{m=1}^{l+n} p_{im} W_m, \quad i \in I, \quad (6)$$

$$y_i + h_i(1 - z_i) + w_i z_i \leq \sum_{m=1}^{l+n} p_{im} H_m, \quad i \in I. \quad (7)$$

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

$$x_i + w_i(1 - z_i) + h_i z_i \leq x_j + (1 - a_{ij}^L) W_{\max}, \quad i, j \in I, \quad (8)$$

$$y_i + h_i(1 - z_i) + w_i z_i \leq y_j + (1 - a_{ij}^B) H_{\max}, \quad i, j \in I, \quad (9)$$

$$a_{ij}^L + a_{ji}^L + a_{ij}^B + a_{ji}^B \geq p_{im} + p_{jm} - 1, \quad i, j \in I, \quad m \in M. \quad (10)$$

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

$$x_i + w_i(1 - z_i) + h_i z_i \leq x_k^0 + (1 - b_{ik}^L) W_{\max}, \quad i \in I, \quad k \in I^0, \quad (11)$$

$$x_k^0 + w_k^0 \leq x_i + (1 - b_{ik}^R) W_{\max}, \quad i \in I, \quad k \in I^0, \quad (12)$$

$$y_i + h_i(1 - z_i) + w_i z_i \leq y_k^0 + (1 - b_{ik}^B) H_{\max}, \quad i \in I, \quad k \in I^0, \quad (13)$$

$$y_k^0 + h_k^0 \leq y_i + (1 - b_{ik}^A) H_{\max}, \quad i \in I, \quad k \in I^0, \quad (14)$$

$$b_{ik}^L + b_{ik}^R + b_{ik}^B + b_{ik}^A \geq p_{im} + q_{km} - d_i - 1, \quad i \in I, \quad k \in I^0, \quad m \in M. \quad (15)$$

Сформулированные условия (1)–(15) определяют задачу с произвольными разрезами. Далее выпишем математическую постановку задачи упаковки с гильотинными разрезами. Для этого необходимо представить рекурсивный процесс разрезания прямоугольного контейнера на  $\bar{n}$  предметов в терминах математического программирования. Рассмотрим матрицу  $\bar{n} \times \bar{n}$ , представленную на рис. 1.

Все элементы этой матрицы над 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  $a_{\bar{n}1}, \dots, a_{\bar{n}\bar{n}}$ , содержащих в себе ровно один прямоугольный предмет.

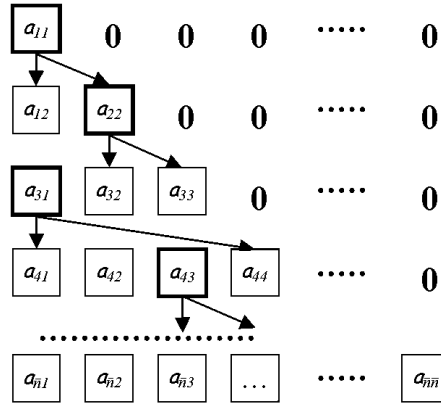


Рис. 1. Матрица гильотинного раскроя

Используем этот 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 —  $n \times n$ . Введём следующие переменные:  $\bar{X}_{ij}^m, \bar{Y}_{ij}^m, \bar{W}_{ij}^m, \bar{H}_{ij}^m, i, j \in I, m \in M$ , — координаты и размеры прямоугольных областей, соответствующих элементам матриц гильотинного раскроя. Бинарная переменная  $g_{ij}^m, 1 \leq j < i \leq n - 1, m \in M$ , 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. Введём аналогичную 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 бинарные переменные  $r_{ij}^m, i, j \in I, m \in M$ , 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

запрещёнными областями в гильотинном случае имеет вид

$$\min \sum_{m=1}^{l+n} u_m. \quad (16)$$

Условие (2), задающее порядок контейнеров, остаётся прежним:

$$u_m \geq u_{m+1}, \quad m \in M. \quad (17)$$

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

$$u_m n \geq \sum_{i,j=1}^n r_{ij}^m, \quad m \in M. \quad (18)$$

Каждый предмет находится только в одной области одного из контейнеров:

$$\sum_{m=1}^{l+n} \sum_{j=1}^n r_{ij}^m = 1, \quad i \in I. \quad (19)$$

Каждый предмет можно поместить в отдельную область:

$$\sum_{i=1}^n r_{ij}^m \leq 1, \quad j \in I, \quad m \in M. \quad (20)$$

Элементы  $a_{11}$  матриц гильотинного раскроя задаются параметрами контейнеров:

$$\overline{X}_{11}^m = 0, \quad m \in M, \quad (21)$$

$$\overline{Y}_{11}^m = 0, \quad m \in M, \quad (22)$$

$$\overline{W}_{11}^m = u_m W_m, \quad m \in M, \quad (23)$$

$$\overline{H}_{11}^m = u_m H_m, \quad m \in M. \quad (24)$$

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

$$\sum_{j=1}^i (g_{ij}^m + v_{ij}^m) \leq u_m, \quad 1 \leq i \leq n-1, \quad m \in M. \quad (25)$$

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

$$\overline{X}_{i+1,j}^m = \overline{X}_{ij}^m, \quad 1 \leq j < i \leq n-1, \quad m \in M, \quad (26)$$

$$\overline{Y}_{i+1,j}^m = \overline{Y}_{ij}^m, \quad 1 \leq j < i \leq n-1, \quad m \in M, \quad (27)$$

$$\overline{W}_{i+1,j}^m = \overline{W}_{ij}^m - v_{ij}^m s_i^m, \quad 1 \leq j < i \leq n-1, \quad m \in M, \quad (28)$$

$$\overline{H}_{i+1,j}^m = \overline{H}_{ij}^m - g_{ij}^m s_i^m, \quad 1 \leq j < i \leq n-1, \quad m \in M. \quad (29)$$

В результате разреза добавляется новая область:

$$\overline{X}_{i+1,i+1}^m = \sum_{j=1}^i ((g_{ij}^m + v_{ij}^m) \overline{X}_{ij}^m + v_{ij}^m \overline{W}_{i+1,j}^m), \quad 1 \leq i \leq n-1, \quad m \in M, \quad (30)$$

$$\overline{Y}_{i+1,i+1}^m = \sum_{j=1}^i ((g_{ij}^m + v_{ij}^m) \overline{Y}_{ij}^m + g_{ij}^m \overline{H}_{i+1,j}^m), \quad 1 \leq i \leq n-1, \quad m \in M, \quad (31)$$

$$\overline{W}_{i+1,i+1}^m = \sum_{j=1}^i (g_{ij}^m \overline{W}_{ij}^m + v_{ij}^m s_i^m), \quad 1 \leq i \leq n-1, \quad m \in M, \quad (32)$$

$$\overline{H}_{i+1,i+1}^m = \sum_{j=1}^i (v_{ij}^m \overline{H}_{ij}^m + g_{ij}^m s_i^m), \quad 1 \leq i \leq n-1, \quad m \in M. \quad (33)$$

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

$$x_i \geq \overline{X}_{nj}^m - \left(1 - \sum_{j=1}^n r_{ij}^m\right) W_{\max}, \quad i \in I, \quad m \in M, \quad (34)$$

$$y_i \geq \overline{Y}_{nj}^m - \left(1 - \sum_{j=1}^n r_{ij}^m\right) H_{\max}, \quad i \in I, \quad m \in M, \quad (35)$$

$$w_i(1 - z_i) + h_i z_i \leq \overline{W}_{nj}^m + \left(1 - \sum_{j=1}^n r_{ij}^m\right) W_{\max}, \quad i \in I, \quad m \in M, \quad (36)$$

$$h_i(1 - z_i) + w_i z_i \leq \overline{H}_{nj}^m + \left(1 - \sum_{j=1}^n r_{ij}^m\right) H_{\max}, \quad i \in I, \quad m \in M. \quad (37)$$

Ограничения (11)–(14) из группы условий, исключающих пересечения предметов и запрещённых областей, находящихся в одном контейне-

ре, не изменятся. Ограничение (15) следует переписать в переменных  $r_{ij}$ :

$$b_{ik}^L + b_{ik}^R + b_{ik}^B + b_{ik}^A \geq \sum_{j=1}^n r_{ij}^m + q_{km} - d_i - 1, \quad i \in I, \quad k \in I^0, \quad m \in M. \quad (38)$$

Выписанные в терминах частично-целочисленного программирования формулировки позволяют использовать коммерческое программное обеспечение для нахождения оптимальных решений рассматриваемых задач. При этом в задаче с гильотинными разрезами ограничения (28)–(33) 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. п. 6.3). Поэтому основные усилия направлены на разработку приближённых алгоритмов.

## 2. Кодирование схемы

Важную роль в алгоритмах упаковки играет способ кодирования решений. Рассматриваются две кодирующие схемы, использовавшиеся ранее в задачах проектирования интегральных микросхем [7, 10]. Содержимое каждого контейнера представляется в виде кодирующей структуры. В настоящей статье предложены алгоритмы декодирования, преобразующие такую структуру в упаковку предметов с учётом запрещённых областей.

Задачи с гильотинными разрезами возникают в приложениях, когда исходный материал необходимо разрезать при помощи гильотины. Структура 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. Постфиксная запись такого арифметического выражения, когда операторы стоят сразу после пары операндов, на которую они действуют, называется *польской записью* [10]. Любая польская запись имеет вид  $\alpha = \alpha_1 \alpha_2 \dots \alpha_{2n-1}$ , при этом

- (i) каждый символ  $\alpha_i$ ,  $1 \leq i \leq 2n-1$ , является либо предметом, либо оператором склейки;
- (ii) каждый предмет встречается в записи ровно один раз;
- (iii) для любого  $k$ ,  $1 \leq k \leq 2n-1$ , количество операторов склейки в последовательности  $\alpha_1 \dots \alpha_k$  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 [7]. Для представления LB-компактных решений будем использовать ориентированные деревья. Вершинам дерева соответствуют прямоугольные предметы. Рёбра 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 определённости в качестве родителя выбирается самый нижний предмет. Заметим, что среди оптимальных решений всегда найдётся LB-компактное. Поэтому далее будут рассматриваться только такие решения.

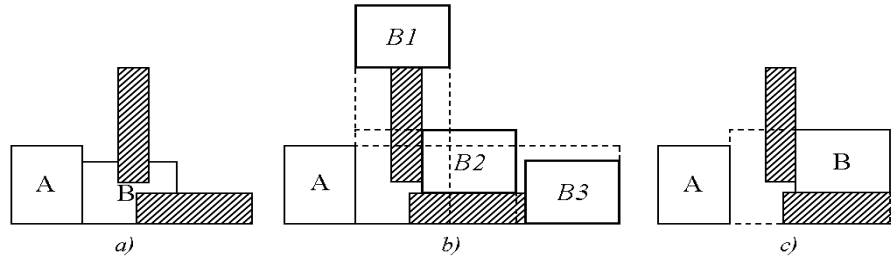


Рис. 2. Декодирование с учётом запрещённых областей

В задачах упаковки с запрещёнными областями требуется, чтобы полученное при декодировании решение было допустимым, т. е. с запрещёнными областями не должны пересекаться предметы, которым это не позволено по условию задачи. С этой целью разработана жадная процедура [1], позволяющая на каждом шаге алгоритма декодирования с помощью комбинаций сдвигов вверх и вправо находить наилучшее допустимое положение предмета относительно запрещённых 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. Пусть предмет  $B$  занимает недопустимое положение и пересекается с запрещёнными областями (рис. 2(a)). Тогда с помощью сдвигов вверх и вправо найдём допустимые положения  $B1$ ,  $B2$  и  $B3$  (рис. 2(b)). Положение  $B2$  характеризуется минимальной площадью сдвига, поэтому разместим в нём предмет  $B$ , как в наилучшем из возможных (рис. 2(c)). Пусть  $k$  — число

запрещённых 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, трудоёмкость данной процедуры для каждого контейнера составляет  $O(k^2)$ .

Декодирование польской записи выполняется за линейное время однократным просмотром записи и последовательной склейкой предметов соответствующими операторами, если запрещённые области отсутствуют [10]. Алгоритм декодирования выполняет следующие действия. Польская запись  $\alpha$  просматривается слева направо. Если очередной символ записи  $\alpha_i$  является предметом, то помещаем его в стек. Если  $\alpha_i$  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 запрещёнными областями алгоритм декодирования становится более сложным и трудоёмким. 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  $\bar{n}$  раз просмотреть польскую запись и найти допустимое положение каждого предмета. Здесь  $\bar{n}$  — это число предметов, содержащееся в рассматриваемом контейнере. Время работы алгоритма в этом случае становится равным  $O(\bar{n}^2 + k^2)$ .

Декодирование ориентированного дерева выполняется поиском в глубину [7]. Для каждого предмета координата  $x$  определяется его родителем, а координата  $y$  — при помощи контура частичной упаковки. Контур состоит из уже размещённых предметов, покрывающих частичную упаковку сверху. Его использование позволяет на порядок сократить время декодирования. Если предмет занимает недопустимое положение, т. е. пересекается с запрещённой областью, то для него выполняется процедура поиска наилучшего допустимого положения, описанная выше, и только затем продолжается декодирование. Время работы алгоритма декодирования составляет  $O(\bar{n} + k^2)$ .

### 3. Имитация отжига

Алгоритм имитации отжига представляет собой вероятностную процедуру локального поиска и принадлежит широко известному классу алгоритмов, называемых *метаэвристиками* [9]. Принцип работы алгоритмов локального поиска основан на понятии окрестности решения.

Окрестностью текущего решения называется множество всевозможных решений, полученных путём однократного применения некоторой операции к текущему решению. Алгоритм имитации отжига осуществляет вероятностный локальный поиск, совершая шаги как улучшающие целевую функцию, так и ухудшающие её. Однако с увеличением числа итераций и уменьшением параметра температуры  $T$  вероятность совершения ухудшающего шага становится меньше. Параметр  $T$  является ключевым в данном алгоритме. При фиксированной температуре вероятность сделать шаг, значительно ухудшающий текущее решение, меньше, чем шаг с небольшим ухудшением. С ростом числа итераций температура уменьшается по закону геометрической прогрессии  $T = rT$ . Коэффициент охлаждения  $r$  выбирается достаточно близким к единице,  $r \in [0,8; 0,99]$ . При каждом значении температуры алгоритм выполняет заранее заданное число шагов порядка мощности окрестности. В качестве критерия остановки может служить время работы алгоритма, число выполненных итераций, либо состояние, когда в течение заданного числа итераций алгоритм не находит новых рекордов.

Разработанный алгоритм имитации отжига начинает свою работу с построения начального решения. Пусть число контейнеров в начальном решении равно  $m$  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 работа алгоритма делится на два уровня. Первым уровнем является параллельный локальный поиск. Для каждого контейнера цикл локального поиска повторяется заданное количество раз, равное  $v_j$ . 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 идёт в области допустимых и недопустимых решений, т. е. предметы могут выступать за границы контейнера. В этом случае в целевую функцию добавляется штраф, равный площади выступающих за границы контейнера частей предметов.

На втором уровне выполняется алгоритм *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 неза-

нятой площади первых  $m - 1$  контейнеров, где  $m$  — число используемых контейнеров. Данная оценка вычисляется следующим образом:

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

Заметим, что в данном выражении площадь запрещённых областей, на покрытие которой не хватает предметов с параметром  $d_i = 1$ , 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

ШАГ 1. Построить начальное решение  $S_1, \dots, S_m$ .

ШАГ 2. Задать начальную температуру  $T > 0$ .

ШАГ 3. Повторять, пока не выполнен критерий остановки.

3.1. Выполнить цикл  $v_j$  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.

ШАГ 4. Выдать лучшее найденное решение.

Для построения окрестностей необходимо определить операции над польскими записями и ориентированными деревьями. Для работы с 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 противоположными ( $+$  на  $*$ ,  $*$  на  $+$ ). Пусть  $\alpha = \alpha_1 \alpha_2 \dots \alpha_{2n-1}$  — польская запись. Заметим, что  $\alpha$  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  $\pi_1\pi_2\ldots\pi_n$  — некоторая перестановка предметов, а  $c_i$  — цепи суммарной длины  $n - 1$ . Два предмета называются *смежными* в  $\alpha$ , если они стоят рядом в перестановке  $\pi_1\pi_2\ldots\pi_n$ . Предмет и оператор называются *смежными*, если они стоят рядом в последовательности  $\alpha_1\alpha_2\ldots\alpha_{2n-1}$ . Будем использовать следующие три операции над польскими записями: перестановка двух смежных предметов, замыкание непустой цепочки операторов, перестановка предмета и смежного с ним оператора. Необходимо отметить, что в некоторых случаях перестановка предмета и смежного с ним оператора может привести к невыполнению условия (iii) в определении польской записи. При выборе соседнего решения такие операции рассматриваться не будут. Для построения окрестности решений, представленных в виде ориентированных деревьев, будем использовать следующие две операции: перестановка двух вершин дерева и перемещение листа дерева. Также при решении задачи упаковки неориентированного типа используется операция поворота предмета на 90 градусов. Окрестность гильотинной задачи обладает линейной мощностью. Мощность окрестности негильотинной задачи является квадратичной. Кроме того, введённые окрестности обладают свойством достижимости, т. е. для двух произвольных решений верно, что одно может быть получено из другого путём применения указанных операций за конечное число шагов.

#### 4. 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 удаётся, то в следующий, затем второй предмет — во второй контейнер, если не удаётся, 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  $\hat{\pi}+$ , либо  $\hat{\pi}*$  после одного из уже размещённых в контейнере предметов, где  $\hat{\pi}$  — 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 неразмещённых предметов. Трудоёмкость алгоритма равна  $O(n(nT_d + T_{LB}))$ , где  $T_d$  — трудоёмкость декодирования, а  $T_{LB}$  — трудоёмкость 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

ШАГ 1. Все контейнеры закрыты для использования.

ШАГ 2. Упорядочить список предметов по невозрастанию их площади.

ШАГ 3. Найти нижнюю оценку  $LB$  оптимального числа контейнеров для задачи со списком неразмещённых предметов.

ШАГ 4. Открыть первые  $LB$  закрытых контейнеров.

ШАГ 5. Разместить предметы в открытых контейнерах в заданном списке порядке.

ШАГ 6. Удалить размещённые предметы из списка.

ШАГ 7. Если список предметов пуст, то закончить работу алгоритма, иначе перейти на шаг 3.

При решении классических задач упаковки в контейнеры могут использоваться нижние оценки для минимального числа контейнеров, описанные в [2–4, 6]. В задачах упаковки с запрещёнными областями каждый контейнер имеет свои размеры и множество запрещённых областей, т. е. является уникальным. Поэтому применение стандартных нижних оценок не представляется возможным. Для случая с запрещёнными областями будем использовать обобщение нижней оценки  $L_0$ , которая также известна как непрерывная нижняя оценка для задач прямоугольной упаковки в контейнеры и вычисляется по формуле

$$L_0 = \left\lceil \sum_{i=1}^N w_i h_i / WH \right\rceil,$$

где  $w_i$  и  $h_i$  — ширина и высота  $i$ -го предмета, а  $W$  и  $H$  — ширина и высота контейнеров. Нижняя оценка для задач с запрещёнными областями

имеет вид

$$L_{IMP} = \min \left\{ m \mid \max \left( 0, \sum_{i=1}^n w_i h_i d_i - \sum_{j=1}^m \sum_{k=1}^K w_k^0 h_k^0 q_{kj} \right) + \sum_{i=1}^n w_i h_i (1 - d_i) - \sum_{j=1}^m \left( W_j H_j - \sum_{k=1}^K w_k^0 h_k^0 q_{kj} \right) \leq 0, \quad m \geq 0 \right\}.$$

Данная оценка аналогична непрерывной нижней оценке и указывает минимальное число контейнеров, которое необходимо взять с начала списка, чтобы разместить все предметы.

## 5. 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 нём предметы предметами 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  $T_d$  — трудоёмкость используемого декодера. При работе данного алгоритма предметы добавляются в контейнеры так же, как и при построении начального решения.

## 6. 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.

**6.1. Примеры прямоугольной упаковки с запрещёнными областями.** Для анализа работы алгоритма сгенерированы два класса примеров. В примерах первого класса (Class\_Preplaced) ни один предмет не может пересекаться с запрещёнными областями. Таким образом, каждая запрещённая область может рассматриваться как заранее размещённый предмет. Во втором классе (Class\_Impurities) каждой запрещённой области соответствует предмет таких же размеров. При этом допускается его пересечение с запрещёнными 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 запрещённых 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 запрещённых областей безотходным способом. Размеры дополнительных контейнеров брались равными  $300 \times 300$ . 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. 1 и 2 показаны результаты экспериментов. Решались четыре типа задач: с поворотами предметов и без, 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, так как 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.



Т а б л и ц а 1

Значение  $D_{m-1}$  (%) для примеров с заранее размещёнными предметами  
(Class\_Preplaced)

	$n$	IMP_10				IMP_40				IMP_70			
		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

**6.2. Примеры классической прямоугольной упаковки.** Для анализа работы алгоритма на классических задачах прямоугольной упаковки в контейнеры (т. е. без запрещённых областей) использовались тестовые примеры, предложенные в [8]. Всего рассматривается 500 примеров, разделённых 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  $[1, 100]$ ,  $W = H = 300$ .

Т а б л и ц а 2

Значение  $D_{m-1}$  (%) для примеров с запрещёнными областями  
(Class\_Impurities)

	$n$	IMP_10				IMP_40				IMP_70			
		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

В описанных шести классах размеры всех предметов выбираются из одного интервала. В следующих четырёх 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 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 четвёртого 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 [8]. При тестировании разработанного алгоритма имитации отжига на данных примерах использовались те же начальные значения параметров, что в предыдущем эксперименте. В качестве критерия останова также выступало время работы алгоритма, ограниченное 300 секундами.

В табл. 3 представлены средние значения отношений полученных верхних оценок к нижним, а также результаты, опубликованные в [8]. Для 69,7% примеров верхние оценки совпали с нижними. Это означает, что найдены оптимальные решения. Также с помощью разработанного алгоритма улучшены результаты, полученные ранее.

Т а б л и ц а 3

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

Класс	$n$	Поиск с запретами [8]				Имитация отжига			
		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
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. 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 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, так как она более чувствительна к локальным изменениям решения. Поэтому в табл. 4 указано время последнего нахождения решения с меньшим количеством контейнеров. Используя таблицу быстро-

действия ЭВМ [5], несложно заметить, что время нахождения последнего рекордного решения для алгоритма [8] на вычислительной машине с процессором AMD Athlon 1,44 GHz имело бы тот же порядок величины, что и время предложенного алгоритма.

**6.3. 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. Данный раздел содержит результаты, полученные с помощью коммерческого пакета GAMS (<http://www.gams.com/>), который позволяет находить оптимальные решения ряда задач, используя подключаемые модули BARON, CPLEX, LINDOGLOBAL, XPRESS и др. Расчёты велись на вычислительной машине с процессором Intel Celeron 1,8 GHz. Рассмотрены 500 тестовых примеров классической прямоугольной упаковки из предыдущего раздела.

Т а б л и ц а 4

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

Класс	Поиск с запретами [8]				Имитация отжига			
	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

На поиск решения пакету отводилось 20 часов для каждого примера. В качестве решающего ядра для модели с произвольными разрезами выбран CPLEX (версия 11.0). 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 приближённых решений (значение параметра *mireremphasis* равно 1). Рассматривались постановки без поворотов предметов. В результате расчётов оказалось, что модель с гильотинными разрезами является слишком сложной для пакета, поскольку не было найдено ни одной гильотинной упаковки для рассматриваемых примеров.

Т а б л и ц а 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 табл. 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 найдены допустимые решения, отмечены числами в скобках, указывающими, сколько примеров данной размерности в классе 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, оптимальность которых доказана с помощью пакета, а во втором  $ОПТ_{UB=LB}$  количество примеров, для которых найденная верхняя оценка совпала с известной нижней оценкой.

В результате вычислительного эксперимента решены 82% рассмотренных тестовых примеров. Для 20,4% примеров найденные решения являются оптимальным. Оптимальность ещё 13,4% следует из совпадения найденного решения с известной нижней оценкой. Как видно из таблицы, примеры размерностью 20 предметов достаточно эффективно решаются с помощью пакета. Сложность примеров значительно возрас-

тает с увеличением числа предметов. Начиная с 80 предметов, не всегда удаётся получить допустимое решение, не говоря уже о качественном приближённом решении.

## 7. Заключение

Рассмотрена новая прикладная задача прямоугольной упаковки в контейнеры с запрещёнными областями. Исследованы формулировки с гильотинными и произвольными разрезами. Выписаны математические постановки данных задач в терминах частично-целочисленного линейного и нелинейного 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 учётом запрещённых 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. Проведённые численные эксперименты показали, что алгоритм позволяет находить решения с малой погрешностью, в том числе и глобально оптимальные.

## ЛИТЕРАТУРА

1. Beisiegel B., Kallrath J., Kochetov Yu., Rudnev A. Simulated annealing based algorithm for the 2D bin packing problem with impurities // Oper. Res. Proc., 2005. — Heidelberg: Springer-Verl., 2006. — P. 109–113.
2. 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.
3. 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.
4. 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.
5. Dongarra J. J. Performance of various computers using standard linear equations software // Technical Report No. CS-89-85, University of Manchester, 2008. — 102 p.
6. Fekete S. P., Schepers J. On more-dimensional packing. Part II: Bounds // Technical Report No. 97.289, Universität zu Köln, 2000. — 20 p.

7. **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. (<http://eda.ee.ucla.edu/EE201A-04Spring/otree.pdf>)
8. **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.
9. **Osman I. H., Laporte G.** Metaheuristics: a bibliography // Ann. Oper. Res. — 1996. — Vol. 63 — P. 513–628.
10. **Wong D. F., Liu C. L.** A new algorithm for floorplan design // Proc. DAC, 1986. — Piscataway: IEEE Press, 1986. — P. 101–107.
11. **Zheng Y., Tang L.** Hybrid scatter search and tabu search for the mother plate design problem in the iron and steel industry // Proc. Comput. Sci. Optimization, 2009. — Washington: IEEE Computer Society, 2009. — P. 978–980.

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

Статья поступила  
1 сентября 2009 г.  
Переработанный вариант —  
14 апреля 2010 г.