

# Упаковка прямоугольников в полосу модифицированным методом Нелдера-Мида с использованием генетического алгоритма

C.A. Мартинин M.B. Храпченко

**Аннотация.** Исследуется задача упаковки прямоугольников в полубесконечную полосу. Известно, что эта задача является NP-трудной. Предложен новый эвристический алгоритм упаковки с использованием модифицированного метода Нелдера-Мида, генетического алгоритма и линейного программирования. Проведенное экспериментальное исследование предложенного алгоритма на известных тестовых примерах демонстрирует его преимущества. Предложены перспективные направления его использования для упаковки произвольных выпуклых многоугольников. Показана принципиальная возможность распараллеливания данного алгоритма.

## 1. Введение

Задача упаковки прямоугольников в одну полубесконечную полосу (так называемая strip packing problem) возникает в различных сферах, начиная от элементарного раскroя материала и размещения микросхем до размещения вычислительных задач (приложений) на кластере.

Данная задача достаточно хорошо исследована [2,3,17]. Однако, в связи с тем, что рассматриваемая задача является NP-трудной, существует и продолжает появляться большое количество методов ее решения, основанных на различных подходах, позволяющих сократить объем вычислений. Для решения задачи были предложены методы линейного и динамического программирования [9], эвристические методы, наиболее известным из которых является Bottom-Left (BL) алгоритм [3], достаточно полный обзор методов приведен в [19]. Некоторое время назад появились работы, в которых использовались эволюционные методы, в частности, генетические алгоритмы (например, [5,6,12,29,30]). Известны также работы, использующие для решения рассматриваемой задачи алгоритмы имитации отжига [13] и искусственных нейросетей [22]. Все эти алгоритмы направлены на сокращение времени вычисления при сохранении приемлемых характеристик упаковки.

Кроме того, наряду с двумерной задачей в настоящее время вызывают интерес алгоритмы для случая произвольной размерности (см., например, [4]). Еще одно направление исследований связано с упаковкой различных геометрических фигур (например, сфер [8]). В последнее время возобновился интерес к упаковке кругов и квадратов в разнообразные виды областей (квадраты, правильные треугольники, круги). Хотя различные постановки задачи в этом направлении были сформулированы достаточно давно (см., например, [11, 18]), интерес к их решению продолжает оставаться высоким.

Предложенный в данной статье алгоритм основан на выборке при помощи генетического алгоритма подмножества прямоугольников, а затем их упаковке с использованием дополнительных построений (основанных на модифицированном алгоритме Нелдера-Мида [21]), которые позволяют в дальнейшем применить методы динамического и линейного программирования.

Как показывают теоретические исследования, широко известные эвристические алгоритмы полиномиальной сложности, например, Bottom-Left (BL) или Best-Fit Decreasing Height (BFDH), шельфовый, в которых осуществляется предварительное упорядочивание прямоугольников, на известных тестовых входных данных [7,15,16,23] не осуществляют оптимальную упаковку (что следует из известных оценок см., например, [19,24,25,28]). В то же время предложенный алгоритм на тех же тестовых данных позволяет получить наилучшую из теоретически возможных упаковок, что подтверждается экспериментальными данными.

Данный алгоритм позволяет при помощи однотипных вычислений производить упаковку в произвольном ( $n$ -мерном,  $n > 2$ ) пространстве и при соответствующей доработке (написании соответствующих формул ограничений) может быть использован для упаковки произвольных выпуклых многоугольников и сфер в произвольное число полос, а также производить упаковку произвольных выпуклых многоугольников в области различных видов: круги, квадраты, треугольники и т.д. Кроме того, алгоритм будет работать и в том случае, если некоторая подходящая упаковка для некоторой области уже имеется, и стоит задача на основе данной упаковки и дополнительного множества многоугольников построить упаковку для области иного вида. В этом случае можно изменить границы области упаковки, переписав граничные условия, и дальнейшая работа алгоритма будет производится для вновь построенной области.

## 2. Постановка задачи

Формально задача плотной упаковки прямоугольников на полубесконечной полосе для двумерного случая может быть сформулирована следующим образом. Данна последовательность прямоугольников  $R = \{R_1, \dots, R_n\}$ , для которых заданы  $w_i, h_i$  – ширина и высота  $i$ -го прямоугольника соответственно, и задана  $W$  – ширина полубесконечной полосы. Требуется найти 136

ортогональную упаковку без перекрытий данного набора прямоугольников, имеющую минимальную высоту. Под ортогональной упаковкой подразумевается упаковка, в которой прямоугольники расположены параллельно сторонам полосы. Любые вращения прямоугольников запрещены.

В основе алгоритма лежит последовательное выполнение шагов генетического алгоритма и алгоритма упаковки. Сначала производится выборка при помощи генетического алгоритма [1,10,20] подмножества прямоугольников для упаковки, затем выбранные прямоугольники упаковываются с использованием модифицированного алгоритма Нелдера-Мида [21]. Далее производится оценка результата упаковки. Если значение целевой функции генетического алгоритма удовлетворяет критериям, то данное подмножество прямоугольников считается упакованным. Далее последовательно повторяются шаги генетического алгоритма до завершения упаковки всех подмножеств прямоугольников из первоначального множества.

### **3. Основные понятия и используемые эвристики**

#### **3.1. Основные термины и шаги генетического алгоритма**

Генетические алгоритмы широко используются в качестве эвристических методов решения NP-трудных задач [1,10,20].

В генетических алгоритмах используются следующие термины и понятия:

Хромосома – вектор (последовательность), содержащий набор значений. Каждая позиция в хромосоме называется геном. Набор хромосом – вариант решения задачи.

Целевая (fitness) функция – функция оценки приспособленности решений.

Основные операторы генетического алгоритма:

Отбор – селекция (reproduction, selection) осуществляет отбор хромосом в соответствии со значениями их функции приспособленности.

Скрещивание (crossover) – операция, при которой две хромосомы обмениваются своими частями.

Мутация – случайное изменение одной или нескольких позиций в хромосоме.

Генетический алгоритм состоит из последовательности шагов:

Шаг 0. Формирование случайной популяции, состоящей из определенного набора хромосом (особей).

Далее итеративно повторяются шаги 1-3.

Шаг 1. Селекция и скрещивание.

Шаг 2. Мутация.

Шаг 3. Отбор наилучших с точки зрения целевой функции хромосом и замена ими исходных хромосом в популяции.

Процесс работы генетического алгоритма продолжается вплоть до выполнения некоторого критерия останова (как правило, количества итераций). В каждом следующем поколении возникают новые решения задачи. Среди них могут присутствовать как плохие, так и хорошие, но благодаря отбору, число приемлемых решений будет возрастать.

#### **3.2. Метод Нелдера-Мида**

Метод симплексов (метод Нелдера-Мида) – достаточно часто используемый алгоритм нелинейной оптимизации, представляет собой численный метод для поиска минимума целевой функции в многомерном пространстве. Идея метода заключается в вычислении целевой функции в вершинах симплекса и последовательного перемещения симплекса в направлении оптимальной точки. Этот метод является одним из наиболее быстрых и наиболее надежных не градиентных методов многомерной оптимизации.

Симплекс или N-симплекс определяется как выпуклая оболочка N+1 точек, не лежащих в одной гиперплоскости. Эти точки называются вершинами симплекса.

Основные шаги метода:

**Шаг 1.** Вычисление значения функции в каждой вершине симплекса.

**Шаг 2.** Преобразование симплекса. В применении к данному алгоритму – отражение вершины симплекса (выбранная вершина отражается относительно центра противоположной грани). При выборе вершины, в которой целевая функция принимает наилучшее значение (наименьшее отрицательное значение) алгоритм работает быстрее, но вероятность попадания в локальный минимум возрастает. При выборе вершины, в которой целевая функция принимает наихудшее значение (наибольшее отрицательное значение), алгоритм работает медленнее, но надежнее.

**Шаг 3.** Если критерий завершения удовлетворен (см. ниже, в подробном описании метода), то поиск прекращается.

#### **3.3. Применение задачи линейного программирования**

После того как при помощи алгоритма Нелдера-Мида была получена упаковка, ее уплотнение производится с использованием линейного программирования. Высоту упаковки, полученную при помощи алгоритма Нелдера-Мида, примем за максимальную высоту. Минимальная высота полосы вычисляется как суммарная площадь прямоугольников, входящих в данную упаковку, деленная на ширину полосы. Зная ширину, максимальную и минимальную высоту полосы для упаковки, полученной при помощи алгоритма Нелдера-Мида, можно выписать соотношения, определяющие взаимное расположение прямоугольников и их расположение относительно границ полосы.

Сначала запишем соотношения для взаимного расположения прямоугольников (как соотношения между их центрами) в виде линейных неравенств, связанных отношением «ИЛИ» (1).

$$-x_i + x_j + \frac{w_i}{2} + \frac{w_j}{2} \leq 0$$

$$x_i - x_j + \frac{w_i}{2} + \frac{w_j}{2} \leq 0 \quad (1)$$

$$-y_i + y_j + \frac{h_i}{2} + \frac{h_j}{2} \leq 0$$

$$y_i - y_j + \frac{h_i}{2} + \frac{h_j}{2} \leq 0,$$

где  $i$  и  $j$  – номера прямоугольников.

Пусть  $H$  – высота упаковки, полученная при помощи алгоритма Нелдера-Мида. Тогда для каждого прямоугольника можно выписать неравенства (2) (как соотношения между границами полосы и центрами прямоугольников), связанные отношением «И»:

$$\frac{w_i}{2} - x_i \leq 0$$

$$x_i + \frac{w_i}{2} - W \leq 0 \quad (2)$$

$$\frac{h_i}{2} - y_i \leq 0$$

$$y_i + \frac{h_i}{2} - H \leq 0,$$

где  $i$  – номер прямоугольника.

Для определения совместности системы будем применять задачу ЛП.

Алгоритм:

**Шаг 1.** Для каждой пары  $i$  и  $j$  выбирается только одно из неравенств (1) и все неравенства (2). Если в (1) выполняется более одного неравенства, выбирается неравенство для  $y$ .

**Шаг 2.** Проверяется совместность системы. Если система совместна, то переходим на Шаг 3, иначе полагаем, что оптимальная упаковка найдена.

**Шаг 3.** Уменьшаем высоту полосы  $H$  и переходим на Шаг 1 с новой высотой  $H$ .

Таким образом, уменьшая высоту полосы методом дихотомии между полученной при помощи алгоритма Нелдера-Мида и минимальной, и решая задачу линейного программирования для определения совместности системы неравенств, можно получить минимальную высоту полосы, соответствующую данному порядку расположения прямоугольников.

## 4. Предлагаемый алгоритм

### 4.1. Основные шаги генетического алгоритма

#### Шаг 1. Сортировка.

На данном шаге основной задачей является упаковка тех прямоугольников, у которых ширина сопоставима с шириной полосы. Их упаковка очевидна. Таким образом, выбираются и упаковываются все прямоугольники, разность между шириной которых и шириной полосы меньше некоторого числа  $\delta$ , которое зависит от ширины полосы (в рассматриваемых примерах  $\delta$  была на три порядка меньше ширины полосы). Упакованные объекты выводятся из дальнейшего рассмотрения.

Следующие шаги непосредственно описывают работу генетического алгоритма.

#### Шаг 2. Формирование популяции.

Формируем подмножества прямоугольников (особи популяции). Случайным образом выбираем  $k$  прямоугольников. Параметр  $k$  выбирается исходя из следующих соображений: быстрота упаковки (время вычисления, зависящее от конкретных вычислительных ресурсов) и их размеры (если размер упаковываемых прямоугольников значительно меньше ширины полосы, то число прямоугольников в подмножестве должно быть по крайней мере таким, чтобы их суммарная ширина была не меньше ширины полосы).

На тестируемых примерах на персональном компьютере (Pentium IV или аналогичном с процессором AMD) величина  $k$  находилась в пределах от 10-15 до 50. Аналогичным образом формируем подмножества до тех пор, пока все прямоугольники в совокупности не будут исчерпаны. Последнее подмножество будет, возможно, меньше  $k$ , но его в популяцию можно не включать. В итоге имеем популяцию размера  $\lfloor n/k \rfloor$ .

#### Шаг 3. Вычисление целевой функции.

Проводим упаковку подмножеств с использованием метода Нелдера-Мида. В качестве целевой функции берем разницу между суммой площадей упакованных прямоугольников и площадью полосы, содержащую данную упаковку. Далее необходимо определить, есть ли подмножества (особи популяции), упаковка которых не требует улучшения. Если есть

подмножество (подмножества) у которой целевая функция не больше некоторого числа  $\Delta$ , выраженного в процентах, то считается, что данная упаковка выполнена и данное подмножество исключается из дальнейшего рассмотрения (особь исключается из популяции). Число  $\Delta$  выбирается по возможности минимальным. Очевидно, что полное совпадение является крайне редким случаем, поэтому разумными границами для  $\Delta$  является  $0 \leq \Delta \leq 10\%(15\%)$ . Затем из оставшейся совокупности необходимо выбрать аналогичные наборы прямоугольников (даже если они входят в различные подмножества), поскольку для них упаковка уже найдена. После этого, если необходимо создать новые подмножества из оставшейся совокупности прямоугольников, возвращаемся к Шагу 2, в противном случае переходим к Шагу 4.

#### Шаг 4. Работа генетического алгоритма.

Работа генетического алгоритма начинается с турнирного отбора. В данном случае применяется самый простой вариант отбора. Процедура **Турнирный отбор** реализована для  $m=2$  и состоит из следующих шагов.

- Случайным образом выбираются два подмножества (две особи популяции) случайным образом и сравниваются по значению фитнес-функции. Оставляем лучшую.
- Аналогичным образом выбираем еще две особи, следя за тем, чтобы они не совпали с ранее выбранными. Сравниваем их таким же способом и оставляем лучшую.
- Полученную пару подвергаем скрещиванию. Случайным образом выбираем количество генов (прямоугольников), которыми особи будут обмениваться (не менее 1 и не более  $k-1$ ).

Случайным образом выбираем точку скрещивания (номер прямоугольника с которого будет произведен обмен прямоугольниками), меняем прямоугольники между подмножествами (особями популяции), считаем целевую функцию для двух вновь получившихся подмножеств. Замену потомками родителей будем производить в любом случае, когда нет совпадений в оставшейся части популяции. В результате получаем две новых особи популяции. Считаем для них целевую функцию (см. Шаг 3) и переходим к Шагу 5.

**Шаг 5. Итерации генетического алгоритма.** На каждой итерации повторяем Шаг 4 (число итераций в предложенном алгоритме зависит от размера популяции, предполагаем, что размер числа итераций равно размеру популяции\*10). Предполагается, что при скрещивании различных особей удастся получить упаковку, удовлетворяющую целевой функции. Если удалось получить соответствующую упаковку, то возвращаемся к Шагу 2. Если нет, то применяем мутацию.

**Шаг 6. Мутация.** Мутация должна выполняться достаточно редко, вероятность ее выполнения в данном алгоритме 1%. За счет нее пробуем улучшить упаковку путем внесения изменения в две особи. Мутацию проводим следующим образом: в случайно выбранной особи случайным образом выбираем прямоугольник и меняем его на произвольный прямоугольник другой произвольно выбранной особи. Вторая особь получает вместо своего прямоугольника прямоугольник первой особи. Считаем целевую функцию, (см. Шаг 3) и переходим к Шагу 5.

**Шаг 7. Изменение параметров генетического алгоритма.** Если предыдущие шаги не дали приемлемого результата и остались неупакованные прямоугольники, для которых не удалось построить упаковку, отвечающую данной целевой функции, пробуем изменить параметры генетического алгоритма, а именно число прямоугольников в особи (например,  $k \pm 1, k \pm 2, K$  и т.п.). Число  $k$  не должно быть меньше или больше некоторого разумного числа (в данной конкретной реализации не меньше 5, не больше 55). Возвращаемся к Шагу 2. В конце работы алгоритма проверяем, остались ли неупакованные прямоугольники. Упаковываем все оставшиеся прямоугольники по методу Неллдера-Мида. Заметим, что, если на Шагах 3 и 4 не возникают приемлемые упаковки, то следует изменить параметр  $k$  после первых нескольких итераций.

Далее заполняем полосу полученными упаковками, строим ограничения для задачи линейного программирования. Это позволяет осуществить общую упаковку полученных упаковок прямоугольников (об этом будет сказано ниже).

Таким образом на вход подаются значения  $n$  – общее число прямоугольников,  $k$  – число прямоугольников в упаковке:

Формирование популяции:

```

for i=1 to N /* N = ⌊n/k⌋ */
do
    Call RecPack() /* вызов программы упаковки*/
end

```

Итерации генетического алгоритма

```

for i=1 to i=NumberOfGeneration do
begin
    for j=1 to j=NumberOfTournamentSelection do
begin
        Call TournamentSelection()
        Call RecPack()
    end
    Call Mutation()
    Call RecPack()
end

```

## 4.2. Основные шаги модифицированного алгоритма Нелдера-Мида

Алгоритм используется для получения плотной упаковки  $k$  прямоугольников на шаге 2 генетического алгоритма. Дано  $k$  – число прямоугольников. Для описания взаимного расположения прямоугольников на плоскости будем использовать  $2k$ -мерное пространство. Координаты центров прямоугольников будем интерпретировать как точку  $2k$ -мерного пространства  $(x_1, y_1, x_2, y_2, \dots, x_k, y_k)$ . Здесь  $(x_1, y_1)$  – координаты центра первого прямоугольника,  $(x_2, y_2)$  – координаты центра второго прямоугольника и т.д.

Для применения алгоритма Нелдера-Мида важнейшими шагами являются выбор начальной точки и критерий преобразования симплекса. В алгоритме, изложенном ниже, предложен новый способ выбора начальной точки, который позволяет двигаться к решению не выходя за пределы построенной допустимой области в пространстве размерности  $2k+2$ . Сложность выбора критерия преобразования симплекса связана с необходимостью предотвратить преждевременную остановку алгоритма из-за попадания симплекса в локальный минимум.

**Шаг 1.** Построение правильного симплекса в пространстве размерности  $2k + 2$ . При построении симплекса используем свойство, что проекция любой его вершины на противоположную грань совпадает с центром масс этой грани.

**Шаг 2.** Построение ограничений на расстояния между центрами прямоугольников (каждого с каждым). Ограничения для каждой пары имеют вид (1) (см. выше):

$$-x_i + x_j + \frac{w_i}{2} + \frac{w_j}{2} \leq 0$$

$$x_i - x_j + \frac{w_i}{2} + \frac{w_j}{2} \leq 0$$

$$-y_i + y_j + \frac{h_i}{2} + \frac{h_j}{2} \leq 0$$

$$y_i - y_j + \frac{h_i}{2} + \frac{h_j}{2} \leq 0,$$

Естественно, достаточно выполнения хотя бы одного из этих ограничений для пары точек  $i$  и  $j$ , а также соотношений (2).

В пространстве размерности  $2k+1$  выберем точку  $O_1$  с координатами  $(0, \dots, 0, C_1)$ ,  $C_1 > 0$ . Каждое такое ограничение (в пространстве размерности  $2k$ ) изменим так, чтобы в пространстве размерности  $2k+1$  оно проходило через точку  $O_1$ . В пространстве размерности  $2k+2$  выберем точку  $O_2$  с координатами  $(0, \dots, 0, C_1, C_2)$ ,  $C_2 > 0$ . Построим сферу с центром в точке  $O_2$  и радиусом  $R$  ( $R <$

$C_2$ ). Каждое ограничение (в пространстве размерности  $2k+1$ ) изменим так, чтобы в пространстве размерности  $2k+2$  оно касалось сферы с центром в точке  $O_2$ . Касательные ограничения строятся таким образом, чтобы точка  $O_2$  лежала в отрицательном (то есть допустимом) полупространстве для каждого ограничения. Как показывает опыт вычислений, параметры  $R$  и  $C_2$  должны быть связаны следующим соотношением:  $0.0005 \leq R/C_2 \leq 0.001$ . Параметр  $C_1$  можно выбрать приблизительно в 50-100 раз больше ширины полосы. В этом случае, как показывают результаты экспериментов, полученная упаковка будет наилучшей по сравнению с упаковками, которые были получены за то же время с другими параметрами.

**Шаг 3.** Построение ограничений для расстояний между центрами прямоугольников и сторонами полосы (для каждого прямоугольника со всеми сторонами). Эти ограничения преобразуем в ограничения пространства  $2k+2$ , как это было описано на предыдущем шаге.

**Шаг 4.** Поиск максимальной и минимальной высоты полосы.

**Шаг 5.** Вычисление текущей высоты полосы как среднего значения между максимальной и минимальной высотой полосы.

**Шаг 6.** Вписываем правильный симплекс в сферу радиуса  $r = \frac{R}{\sqrt{8}}$ , центр

которой находится в точке с координатами  $(0, \dots, 0, C_1-r, -r)$ . Увеличиваем размер допустимой области путем сдвига ограничений на  $R$ . Замкнутые полупространства, которые имеет вид  $x_{2k+1} \leq C_1 + R$  и  $x_{2k+2} \leq R$ , также становятся ограничениями.

**Шаг 7.** Вычисляем целевую функцию для каждой вершины симплекса. Целевая функция  $C$  – расстояние от вершины до ограничений. По построению точка не нарушает ограничение, если значение целевой функции отрицательно (чем меньше значение целевой функции, тем оно ближе к искомому решению). Уменьшаем размер допустимой области на величину, равную модулю целевой функции, умноженному на коэффициент  $Q$  ( $Q$  по порядку равно  $10^{-5}$ ).

**Шаг 8.** Сортировка вершин симплекса в зависимости от значения целевой функции от максимальных к минимальным значениям.

**Шаг 9.** Отражаем  $i$ -ую вершину симплекса относительно своей противоположной грани и получаем точку  $I$ . Если значение целевой функции для точки  $I$  строго отрицательно и разность между целевой функцией и сдвигом меньше  $\epsilon$  (чем меньше  $\epsilon$ , тем точнее работа алгоритма), то решение найдено. В этом случае проектируем точку  $I$  на гиперплоскость  $x_{2k+2}=0$  (точка  $I_1$ ). Затем находим пересечение прямой  $(I_1, O_1)$  с гиперплоскостью  $x_{2k+1}=0$  (точка  $I_2$ ). Точка  $I_2$  будет являться решением и алгоритм завершает работу.

**Шаг 10.** Если значение целевой функции в точке  $I$  отрицательно и  $i$ -ая вершина симплекса не была получена из точки  $I$  на предыдущем

преобразовании симплекса и выполнено хотя бы одно из следующих условий 1 или 2, то тогда  $i$ -ая вершина симплекса заменяется своей противоположной точкой и осуществляется переход на Шаг 8.

**Условие 1:** значение целевой функции в точке I меньше значения целевой функции в  $i$ -ой вершине симплекса.

**Условие 2:**  $2k+1$  координата точки I меньше  $2k+1$  координаты  $i$ -ой вершине симплекса.

В противном случае:

- Если не все вершины симплекса были проверены, то переход на Шаг 9.
- Если ни одна вершина симплекса не была улучшена, то урезание симплекса в два раза относительно вершины с наименьшей (лучшей) целевой функцией.

**Шаг 11.** Если длина ребра симплекса больше  $\varepsilon_s$  то переход на Шаг 7. В противном случае решение не может быть найдено и алгоритм завершает работу.

На практике при значении  $\varepsilon$  от 0.001 до 0.0001 за разумное время были получены упаковки, улучшить которые невозможно (см. таблицу 1). При этом  $\varepsilon_s = 10^{-9}$ .

Следует заметить, что упаковка большого количества прямоугольников приводит к росту вычислительной сложности алгоритма. Поскольку число ограничений оценивается как  $k(k-1)/2 + 4k$ , очевидно, что пропорционально увеличивается и количество арифметических операций. На практике это приводит к значительному увеличению времени с ростом числа упаковываемых прямоугольников. Причем за счет того, что появляются дополнительные ограничения, связанные с возрастанием размерности пространства, длительность выполнения возрастает не квадратично, а еще с некоторым коэффициентом, зависящим от размерности задачи.

### 4.3. Алгоритм уплотнения упаковки

Поскольку для работы алгоритма Нелдера-Мида необходимо, чтобы симплекс имел ненулевую длину ребра, перед упаковкой прямоугольников требуется произвести их урезание. Впоследствии для того, чтобы вернуться к прежним размерам прямоугольников будет использоваться алгоритм уплотнения упаковки.

#### 4.3.1. Проверка существования допустимой области системы линейных неравенств

**Шаг 1.** Аналогично Шагу 2 алгоритма пункта 4.2 строим ограничения.

**Шаг 2.** Выбираем точку касания со сферой T на одном из ограничений. Строим  $2k-1$  гиперплоскостей, которые вместе с гиперплоскостью, соответствующей  $i$ -му ограничению находятся в общем положении и проходят через точку T. Обозначим такое множество гиперплоскостей через F, а сами гиперплоскости назовем дополнительными. Кроме того, построим гиперплоскость, проходящую через точку T и параллельную гиперплоскости  $x_{2k+2}=0$ . Пересечением этих  $2k+1$  гиперплоскостей будет являться прямая l.

**Шаг 3.** Находим ограничение, точка пересечения которого (точка P) с прямой l является ближайшей к точке T. Заменяем произвольную дополнительную гиперплоскость из множества F на гиперплоскость, которая соответствует найденному ближайшему ограничению. Если такая замена для множества F является первой, то при замене сдвигаем найденное ближайшее ограничение так, чтобы оно проходило через середину отрезка [T,P]. Вновь построенная точка P' является серединой этого отрезка, затем переносим точку T в точку P' и получаем новую прямую l.

**Шаг 4.** Если множество F еще содержит дополнительные гиперплоскости, то переходим на Шаг 3.

**Шаг 5.** В результате предыдущих действий будет получена точка T, через которую проходит  $2k+1$  ограничений. Эти  $2k+1$  находящихся в общем положении ограничений дают в своем пересечении прямую. Находим точки пересечения этой прямой с ограничениями. Выбираем то ограничение, расстояние до которого от точки T вдоль прямой минимально и у найденной точки пересечения  $x_{2k+2}$  координата меньше, чем у точки T. Найденная точка становится точкой T.

**Шаг 6.** Через точку T будет проходить  $2k+2$  ограничений. По очереди сдвигаем все ограничения, кроме вновь найденного и сдвинутого на Шаге 3, в его отрицательное полупространство. Так как все эти  $2k+2$  ограничения линейно независимы, то они в пересечении дадут точку, обозначим ее через P. Построим прямую l, проходящую через точки T и P. Пересечем эту прямую со всеми ограничениями, которые будут включать и ограничение, задаваемое гиперплоскостью  $x_{2k+2}=0$ . Будем рассматривать только те точки пересечения, у которых  $x_{2k+2}$  координата меньше, чем у точки T. Точкой P становится точка, ближайшая к точке T. Если перебрав  $2k$  ограничений точку P найти не удалось, то то допустимая область отсутствует и алгоритм завершает работу. Ограничение, при помощи которого была построена точка T, заменяется вновь найденным ограничением, при помощи которого была построена точка P. Точка P становится новой точкой T. Если точка T не принадлежит  $x_{2k+2}=0$ , возвращаемся к началу шага 6.

**Шаг 7.** Находим пересечение луча O<sub>1</sub>T с гиперплоскостью  $x_{2k+1}=0$ . Найденная точка пересечения есть искомое решение.

### 4.3.2. Алгоритм уплотнения упаковки

В основе алгоритма уплотнения упаковки лежит метод линейного программирования.

**Шаг 1.** Выбираем упаковку, генерируем ограничения, описывающие взаимное расположение всех прямоугольников относительно друг друга внутри упаковки и каждого прямоугольника относительно сторон полосы. Максимальной высотой будет полученная высота упаковки, а минимальной – теоретически возможная.

**Шаг 2.** Выбираем текущую высоту как среднее значение между минимальной и максимальной высотой.

**Шаг 3.** Находим точку внутри допустимой области.

**Шаг 4.** Если допустимая область существует, то максимальную высоту делаем текущей, иначе минимальную высоту делаем текущей. Если разность между минимальной и максимальной высотой больше некоторого числа  $\epsilon$ , переходим на Шаг 2, в противном случае алгоритм завершает свою работу.

Следует также заметить, что для увеличения скорости работы алгоритма можно увеличить допустимую область. Для этого необходимо увеличить ширину полосы (например, на 1-3%), а ширину прямоугольников уменьшить (на 1-2%). Далее алгоритм уплотнения упаковки, используя реальные размеры полосы, прямоугольников и данные о взаимном расположении прямоугольников, проверяет возможность данной упаковки. Если такая упаковка возможна, то решение найдено. Если увеличение ширины полосы и уменьшение ширины прямоугольников приводят к невозможности реализации такой упаковки, то упаковку необходимо проводить с исходными данными.

## 5. Результаты реализации

Для реализации алгоритма были написаны следующие подпрограммы: упаковка с помощью модифицированного алгоритма Нелдера-Мида, генетический алгоритм и уплотнение упаковки при помощи задачи линейного программирования. Для наглядного представления результатов тестирования основной программы упаковки была написана вспомогательная программа графического отображения расположения прямоугольников на плоскости. Используемый язык программирования – C++. Общий объем программного кода составляет порядка 5000 строк.

Сравнение проводилось со стандартно реализованными алгоритмами BL, BLDW и шельфовым.

Плотность упаковки вычисляется следующим образом: вычисляется общая площадь прямоугольников, затем вычисляется занимаемая данной упаковкой площадь на полосе (ширина полосы умножается на высоту упаковки) и берется их соотношение, выраженное в процентах.

В качестве тестовых данных были использованы известные тесты для двумерной задачи упаковки, подробно рассмотренные в [7,15,16,23], а также специальным образом подобранные примеры.

Для проведения численных экспериментов использовался персональный компьютер с процессором AMD Athlon 2600+ и 512Мб DRAM.

На рисунке 1 показан результат предложенного алгоритма, время работы алгоритма 13,7 секунды. Плотность упаковки 96% и является оптимальной для специальным образом подобранных восьми прямоугольников, размеры которых равны 2.95\*3.0, 4.95\*4.0, 6.95\*10.0, 0.95\*7.5, 4.95\*2.0, 0.95\*7.5, 4.95\*2.0, 0.95\*7.5 и шириной полосы 10.

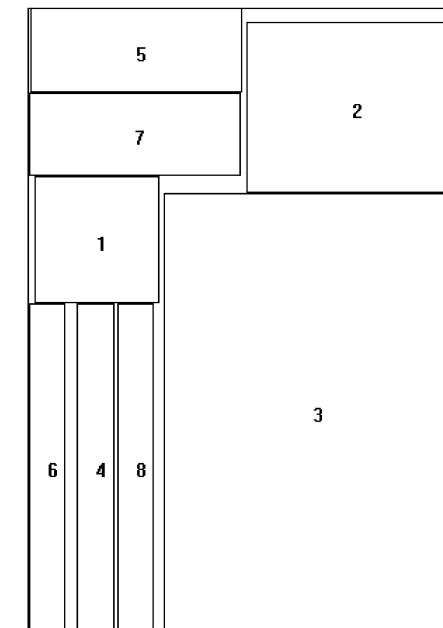


Рис. 1. Упаковка восьми специально подобранных прямоугольников.

Алгоритм BLDW выполнял бы упаковку в следующем порядке: прямоугольник 3 помещается в левом нижнем углу, далее создается новый уровень для прямоугольников 2, 5 и 7, затем прямоугольник 1 помещается на тот же уровень, что и прямоугольник 3, а для прямоугольников 4, 6 и 8 пришлось бы открывать новый уровень. Плотность упаковки – 63%.

Естественно, в случае применения алгоритма BL on-line имеется теоретическая возможность получения результата, аналогичного работе модифицированного алгоритма Нелдера-Мида в том случае, если

прямоугольники предварительно не сортировать по ширине, а подавать их на вход в соответствующем порядке (например, 2,5,7,3,1,4,6,8).

Но вероятность появления прямоугольников в необходимом порядке даже для примера из 8 штук составляет примерно 0,5%. Очевидно, с ростом числа прямоугольников она еще уменьшится.

Очевидно также, что и для алгоритма BFDH легко подобрать пример, когда он позволяет получить плотность упаковки 66,67%, в то время, как предложенный алгоритм дает возможность получить плотность упаковки 100%.

Таким образом в рассмотренных выше случаях предложенный алгоритм дает возможность получить наилучшую упаковку, в то время как остальные рассматриваемые алгоритмы получение наилучшей упаковки не гарантируют.

На рисунке 2 представлен результат упаковки тестового примера (см. [15] C1(P3)) из 16 прямоугольников. Известно, что оптимальная высота упаковки 20.

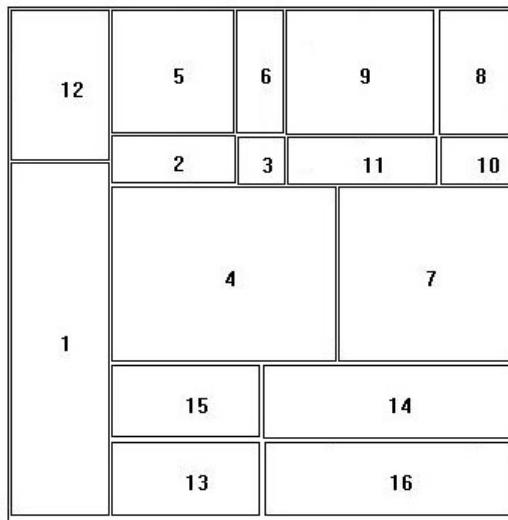


Рис. 2. Оптимальная упаковка 16 прямоугольников тест C1(P3).

В [7] приведены данные о выполнении различных тестов, предложенных в [15,16,23], в том числе и данного, которые показывают, что оптимальной плотности упаковки рассматриваемые алгоритмы не достигают (см таблицу 1), в то время как предложенный алгоритм позволяет ее получить.

**Таблица 1.** Фрагмент таблицы, приведенной в [7]. Сравнение в % наилучшей упаковки с упаковкой bottom-left.

Тест	C1			C2			C3		
	P1	P2	P3	P1	P2	P3	P1	P2	P3
Число прямоугольников	16	17	16	25	25	25	28	29	28
BL	45	40	35	53	80	67	40	43	40
BL-DW	30	20	20	13	27	27	10	20	17
BL-DH	15	10	5	13	73	13	10	10	13

Разумеется, с увеличением роста числа прямоугольников характеристики упаковки, полученной при помощи таких алгоритмов, как BL, BLDW, BLF, шельфового и др. улучшаются. Однако для тестов от 5 до 50 прямоугольников наилучшая из возможных упаковка была получена при помощи предложенного алгоритма.

Для получения окончательного вида упаковки, представленной на рисунках 1 и 2, была использована программа уплотнения упаковки, реализованная по соответствующему алгоритму (см. п. 4).

Ниже на рисунке 3 показан результат упаковки специальным образом подобранных 30 прямоугольников различной формы до и после применения уплотнения с использованием задачи линейного программирования. Высота левой упаковки до применения задачи линейного программирования (84%), правой – после применения задачи линейного программирования (95%), таким образом разница составляет 11%, что является существенной величиной.

Далее приведены данные о времени выполнения предложенного алгоритма и его сравнение по времени выполнения и плотности упаковки с алгоритмами: BL, модификацией BL алгоритма с предварительным упорядочиванием прямоугольников по невозрастанию ширины BLDW и шельфовым. Тестовые примеры были взяты из [15,16,23], время выполнения для предложенного алгоритма соответствовало случаю достижения наилучшей (теоретически возможной) упаковки.

Напомним, что для проведения численных экспериментов использовался персональный компьютер с процессором AMD Athlon 2600+ и 512Mb DRAM, на котором были получены приведенные ниже временные оценки. Очевидно, что при использовании ПК с большей частотой и оперативной памятью время вычисления снизится).

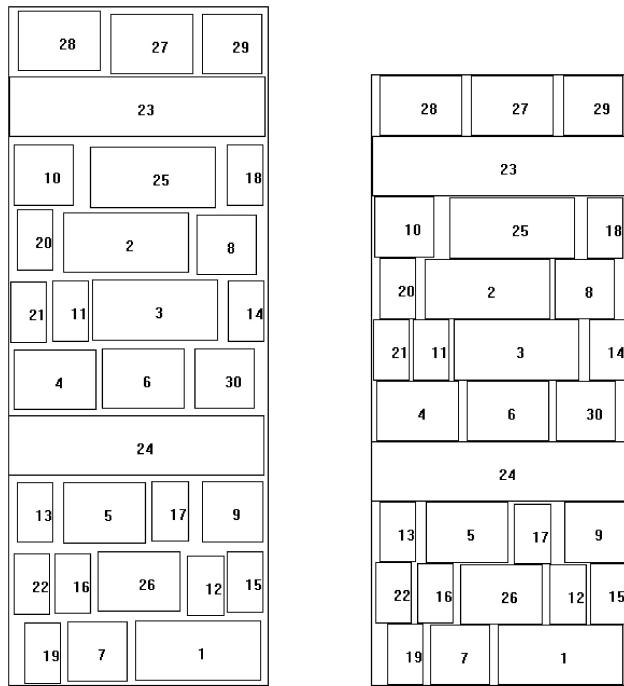


Рис. 3. Упаковка до и после уплотнения с использованием задачи ЛП.

В таблице 2 представлено среднее время работы алгоритма, поскольку в некоторых случаях обработка локальных минимумов может привести к увеличению времени работы. С другой стороны, некоторые упаковки могут быть подсчитаны за значительно меньшее время. Статистические данные также были получены на тестовых примерах [15, 16, 23] для различного числа прямоугольников.

**Таблица 2.** Среднее время работы модифицированного алгоритма Нелдера-Мида тестовые примеры из [15, 16, 23].

Число прямоугольников	Среднее время упаковки
15 – 20 штук	30 – 40 с
25 – 30 штук	1 – 5 мин
45 – 50 штук	10 – 40 мин
60 – 70 штук	3 час

Дальнейшее увеличение числа прямоугольников ведет к значительному увеличению времени работы модифицированного алгоритма Нелдера-Мида, поэтому в предлагаемом алгоритме используется генетический алгоритм, позволяющий осуществлять упаковку блоками по 20-30 штук и затем пытаться ее улучшить.

Поскольку время работы генетического алгоритма и задачи линейного программирования на порядок (или, в некоторых случаях, на два порядка) меньше, чем время работы модифицированного метода Нелдера-Мида, то при окончательной оценке времени выполнения предложенного алгоритма их оценкой времени можно пренебречь.

В следующей таблице 3 представлено среднее время работы алгоритмов и плотность упаковки с 16-17 прямоугольниками для предложенного алгоритма и алгоритмов BL и BLDW.

**Таблица 3.** Сравнение времени выполнения и плотности упаковки для различных алгоритмов по тестам [15, 16, 23].

Алгоритм	Время работы	Средняя плотность упаковки
BL	1с	70%
BLDW	1с	80%
Предложенный алгоритм	15с	97%

На небольшом примере видно, что предложенный алгоритм дает наилучшее решение, хотя и за значительно большее время. В таблице 4 приведено время работы алгоритмов со специально подобранными 10 прямоугольниками (см. Рис. 1), порядок следования прямоугольников 2,5,7,3,1,4,6,8.

В таблицах 3 и 4 в связи с небольшим числом прямоугольников генетический алгоритм не применялся.

**Таблица 4.** Время работы алгоритмов со специально подобранными 10 прямоугольниками

Алгоритм	Время работы	Плотность упаковки
BL	1с	96%
BLDW	1с	63%
Shelf	1с	51%
Предложенный алгоритм	13с	96%

В таблице 5 при упаковке 500 прямоугольников (тесты, предложенные в [23]) генетический алгоритм используется.

**Таблица 5.** Среднее время работы алгоритмов с 500 прямоугольниками по тестам [23].

Алгоритм	Время работы	Средняя плотность упаковки
BL	10 с	89%
BLDW	30 мин	89%
Предложенный алгоритм	60 мин	95%

В заключение заметим, что предложенный алгоритм может быть применен для произвольного числа прямоугольников и время его работы может быть легко спрогнозировано исходя из параметров алгоритма. Первоначальные параметры для работы генетического алгоритма следует выбирать в зависимости от соотношения размеров исходных прямоугольников и ширины полосы. В том случае, если размер прямоугольников не более, чем на порядок отличается от ширины полосы, первоначальное значение  $k$  можно выбрать, например, от 10 до 20. В этом случае время имеется возможность получить оптимальное решение за приемлемое время. Если ширина прямоугольников значительно меньше ширины полосы, число прямоугольников должно быть увеличено (в этом случае на практике достаточно взять  $\varepsilon=0.001$ , что значительно увеличивает скорость работы алгоритма упаковки).

Очевидно, что данный алгоритм обладает хорошими возможностями по распараллеливанию вычислений, поскольку после получения нескольких выборок для упаковки сама упаковка может проводится параллельно.

Данный алгоритм без модификаций может быть применен и для размерностей  $n>2$ . Для этого необходимо добавить ограничения для  $n$ -мерного случая в модифицированном методе Нелдера-Мида. Работа генетического алгоритма остается точно такой же. Также следует заметить, что изменив уравнения ограничений, мы можем применить данный алгоритм для упаковки произвольного типа выпуклых многоугольников в произвольного вида выпуклых областях.

## Литература

- [1] Back T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, New York, 1996.
- [2] Baker B.S., Brown D.J. and Katseff H.P. A 5/4 algorithm for two-dimensional packing. Journal of Algorithms, 2:348--368, 1981.

- [3] Baker B.S., Coffman E.G. Jr., Ronald L. Rivest: Orthogonal Packings in Two Dimensions. SIAM J. Comput. 9(4):846-855 (1980)
- [4] Bansal L., Correre J.R., Kenyon C. and Sviridenko M. Bin Packing in Multiple Dimensions: Inapproximability Results and Approximation Schemes, Mathematics of Operations Research v.31 (2006), pp. 31-49.
- [5] Bortfeldt A.; Gehring, H.: Two metaheuristics for strip packing problems. 5-th International Conference of the Decision Sciences Institute. Athen, Griechenland, 4.-7. Juli 1999.
- [6] Bortfeldt A. Hermann G., A Parallel Genetic Algorithm for Solving the Container Loading Problem International Transactions in Operational Research Vol. 9 Issue 4 Page 497 July 2002
- [7] Burke E.K., Kendall G. & Whitwell G., A New Placement Heuristic for the Orthogonal Stock-Cutting Problem, Operations Research, 52(4) (2004), 655-671.
- [8] Danny Z. Chen, Xiaobo (Sharon) Huy, Yingping Huang, Yifan Li, Jinhui Xuz Algorithms for Congruent Sphere Packing and Applications, Symposium on Computational Geometry 2001: 212-221.
- [9] Gilmore, P. C., Gomory, R. E., A Linear Approach to the Cutting-Stock Problem, Operations Research, 1961, Vol 9, pp 849-859.
- [10] Goldberg D.E., Genetic algorithms in Search, Optimization and Machine Learning., Reading, MA: Addison-Wesley Publishing Company, 1989.
- [11] Goldberg M., The packing of equal circles in a square, Math. Magazine 43, pages 24-30, 1970
- [12] Goodman E.D, Tetelbaum A.Y. and Kureichik V.M. A Genetic Algorithm Approach to Compaction, Bin Packing, and Nesting Problems, Release 3.01, Aug. 1, 1995, Technical Report 95-08-01, Intelligent Systems Laboratory and Case Center for Computer-Aided Engineering and Manufacturing, Michigan State University, 80pp.
- [13] Dagli, C.H.; Hajakbari, A. Simulated annealing approach for solving stock cutting problem Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on, Vol., Iss., 4-7 Nov 1990 Pages:221-223
- [14] Dagli, C. H., Poshyanonda, P., "New Approaches to Nesting Rectangular Patterns", Journal of Intelligent Manufacturing, 1997, Vol. 3, No. 3, pp. 177-190.
- [15] Hopper E. & Turton B.C.H., An Empirical Investigation of Meta-Heuristic and Heuristic Algorithms for a 2D Packing Problem, European Journal of Operational Research, 128(1)(2001), 34-57.
- [16] Hopper E. & Turton B.C.H., Problem Generators for Rectangular Packing Problems, Studia Informatica Universalis, 2(1) (2002), 123-136.
- [17] Kenyon, C., and Remila, E. A Near-optimal Solution to a Two-dimensional Cutting Stock Problem. Mathematics of Operations Research 25, 4 (Nov 2000), 645-656.
- [18] S. Kravitz. Packing cylinders into cylindrical containers. Mathematics Magazine, 40:65-71, 1967
- [19] Lodi A., Martello S., and Monaci M. Two-dimensional packing problems: a survey. European Journal of Operational Research, 141: 241-252,2002.
- [20] Mitchell M. An Introduction to Genetic Algorithms. The MIT Press, Cambridge, MA, 1996.
- [21] Nelder, J. A. and Mead. R. A Simplex Method for Function Minimization. Comput. J. 7, 308-313, 1965.
- [22] Poshyanonda, P.; Bahrami, A.; Dagli, C.H. Two dimensional nesting problem: artificial neural network and optimization approach Neural Networks, 1992. IJCNN., International Joint Conference on, Vol.4, Iss., 7-11 Jun 1992 Pages: 572-577 vol.4

- [23] Wang P.Y. & Valenzuela C.L., Data set generation for rectangular placement problems, European Journal of Operational Research, 134(2001), 378-391.
- [24] Головистиков А.В., Задачи двумерной упаковки и раскроя: обзор. Информатика, выпуск N 20, 2008, с. 18-33.
- [25] Жук С.Н. Онлайновый алгоритм упаковки прямоугольников в несколько полос с гарантированными оценками точности. Труды Института Системного программирования: Методы синтеза и анализа, т.12, 2007, с. 7-16.
- [26] Жук С.Н. Приближенные алгоритмы упаковки прямоугольников в несколько полос. Дискретная математика, т. 18:1, 2006, с. 91-105.
- [27] Канторович Л.В., Залгаллер В.А. Рациональный раскрой промышленных материалов. Наука. – Новосибирск, 1971, 299 с.
- [28] Кузюрин Н.Н., Поступов А.И. Вероятностный анализ шельфовых алгоритмов упаковки прямоугольников в полосу. Дискретная математика, т. 18:1, 2006, с. 76-90.
- [29] Мухачева Э.А., Мухачева А.С., Чиглинцев А.В. Генетический алгоритм блочной структуры в задачах двумерной упаковки. Информационные технологии. Машиностроение. -М.:1999, N 11, с. 13-18.
- [30] Мухачева А.С., Чиглинцев А.В., Смагин М.А., Мухачева Э.А.. Задачи двумерной упаковки: развитие генетических алгоритмов на базе смешанных процедур локального поиска оптимального решения. Приложение к журналу Информационные технологии. Машиностроение. -М.: 2001, N 10, 24 с.