МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. М.В.ЛОМОНОСОВА

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

УДК 519.95; 007:159.955

Псиола В.В.

ОЦЕНКИ КАЧЕСТВА РАБОТЫ АЛГОРИТМА PACKER3D (ТЕОРИЯ И ПРАКТИКА)

Москва 2009 г.

Содержание

1 Введение

Всякий раз, когда требуется разместить товары на складе или загрузить их в транспортное средство, мы сталкиваемся с проблемой уложить груз так, чтобы он занимал как можно меньше места или поместился целиком в наименьшее количество контейнеров заданного объема, иными словами, с задачей об оптимальной упаковке (3-х мерной задаче о рюкзаке и 3-х мерной задаче об упаковке).

Для одномерных задач о рюкзаке и упаковке доказано, что они являются NP-полными [1]. Задачи о рюкзаке и упаковке в трехмерном пространстве заведомо сложнее их одномерных аналогов, но по-прежнему могут быть решены за полиномиальное время на недетерминированных вычислительных устройствах. Если есть алгоритм, решающий трёхмерную задачу, то его можно использовать для решения двухмерной задачи (для этого достаточно один линейный размер для всех объектов определить нулём). Аналогично решение одномерной задачи можно найти, используя алгоритмы для решений двухмерной или трёхмерной задачи. А значит решение одномерной задачи может быть сведено к решению двух или трёхмерной, то есть по трёх мерная и двух мерная задача является NP-полными по определению [1].

В работе [3] предложен алгоритм Packer3d для нахождения приближённого решения этих задач на основе эвристик за полиномиальное время. В данной работе приводятся теоретические и экспериментальные оценки работы этого алгоритма.

2 Проблематика

Как было сказано выше задачи об упаковке и рюкзаке, как в одномерном, так и двух- или трехмерном случаях относятся к классу NP-полных задач. NP-полными задачами называют труднорешаемые задачи, которые сводятся друг к другу за полиномиальное время. NP-полнота задач об упаковке и рюкзаке в одномерном случае показана в работе [1] (стр. 286 и

316 соответственно), а NP-полнота 2-х и 3-х мерных задач следует из очевидной сводимости одномерных задач к своим многомерным аналогам за полиномиальное время. На текущий момент остается открытым вопрос существования полиномиального алгоритма для решения NP-полных задач. Большинство исследователей сходятся во мнении, что такого алгоритма не существует. Поэтому, для решения задач подобного рода используются алгоритмы, дающие приближенное решение за быстрое (полиномиальное) время. В этом смысле становится актуальным вопрос оценки качества подобного рода алгоритмов. Ниже приведены теоретические (доказанные) оценки качества для подобного рода алгоритмов на одномерных задачах об упаковке и рюкзаке.

Одномерная задача об «упаковке в контейнеры» формулируется следующим образом: *задано конечное множество* $U = u_1, u_2, \ldots, u_n$ *«предметов» и «размеры»* $s(u) \in [0, 1]$ *для каждого предмета* $u \in U$ *(размер предмета представлен рациональным числом); требуется найти такое разбиение множества* U *на непересекающиеся подмножества* $U_1, U_2, \ldots, U_k$*, чтобы сумма размеров предметов в каждом подмножестве не превосходила* 1 *и чтобы* k *было наименьшим возможным.* Можно считать, что предметы, принадлежащие каждому множеству U_i упаковываются в один контейнер размера 1, а наша цель упаковать предметы множества U в как можно меньшее число контейнеров. Для этой задачи доказано [1], что наиболее очевидный эвристический алгоритм решения этой задачи под названием «в первый подходящий» в общем случае дает решение не хуже чем на **70%** от оптимального, а именно: *для любой индивидуальной задачи* I *об упаковке в контейнеры, если* $FF(I)$ *решение полученное алгоритмом «в первый подходящий», а* $OPT(I)$ *- наилучшее решение этой задачи, выполнено неравенство:*

$$FF(I) \leq \frac{17}{10} OPT(I) + 2$$

Более того, существуют индивидуальные задачи I*, для которых* $OPT(I)$

произвольно велико и

$$FF(I) \geq \frac{17}{10}(OPT(I) - 1).$$

Небольшая доработка этого алгоритма под названием «в первый подходящий в порядке убывания» (вариация «жадного алгоритма» [2]) позволяет улучшить погрешность до **22%** от оптимального [1], а именно *для любой индивидуальной задачи I об упаковке в контейнеры, если $FFD(I)$ решение полученное алгоритмом «в первый подходящий в порядке убывания», а $OPT(I)$ - наилучшее решение этой задачи, выполнено неравенство:*

$$FFD(I) \leq \frac{11}{9}OPT(I) + 4$$

Более того, существуют индивидуальные задачи I, для которых $OPT(I)$ произвольно велико и

$$FFD(I) \geq \frac{11}{9}OPT(I).$$

Пока не известны и скорее всего не существует алгоритмов решения задачи об «упаковке в контейнеры» имеющих принципиально лучшее качество [1]. 2-х мерные и 3-х мерные варианты этой задачи заведомо сложнее одномерной, так как одномерная может быть легко сведена к своим многомерным аналогам, поэтому оценки качества работы подобных алгоритмов в многомерном случае не могут быть лучше, чем вышеописанные.

Для задачи «о рюкзаке» ситуация несколько другая, и существуют полиномиальные алгоритмы, решающие задачу со сколько угодно хорошей, заранее определенной, точностью. Одномерная задача «о рюкзаке» формулируется следующим образом: *задано конечное множество U «предметов» и для каждого из $u \in U^+$ «размер» $s(u) \in Z^+$ и «стоимость» $v(u) \in Z^+$, а так же положительное число $B \geq max\{s(u) : u \in U\}$ - «емкость рюкзака». Требуется найти такое подмножество $U' \subseteq U$, что $\sum_{u \in U'} s(u) \leq B$ и величина $\sum_{u \in U'} v(u)$ принимает наибольшее возможное значение.* В данной работе, упоминая задачу о рюкзаке (особенно в ее 3-х мерном варианте), мы часта будем подразумевать, что $s(u) = v(u)$. Подобное ограничение оставляет задачу в классе NP-полных [1]. Наиболее

очевидный «жадный алгоритм» примененный для решения этой задачи имеет погрешность 2. Однако, в работе [1] показано, для любого, наперед заданного, числа $k \in Z^+$ есть простая модификация жадного алгоритма, которая за полиномиальное время позволяет решить задачу о рюкзаке с погрешностью не более чем $1 + (1/k)$. К сожалению, для такой модификации, полином, оценивающий скорость работы, имеет число k в показателе, что делает её неприменимой на практике. Однако, в работе [1] показано, что для задачи «о рюкзаке» может быть построено приближенное решение с аналогичной оценкой без использования алгоритма, содержащего k в показателе.

Хотя 2-х и 3-х мерные задачи об «упаковке в контейнеры» и «рюкзаке» имеют туже природу, но они заведомо существенно сложнее, тем более когда речь идет о существовании дополнительных ограничений на укладку предметов в контейнере. Поэтому кажется сомнительной возможность существования простых алгоритмов решения этих задач, для которых можно было бы доказать подобного рода оценки. Тем более, что для существенно более простых NP-полных задач зачастую до сих пор не найдено ни какого полиномиального алгоритма, асимптотическая погрешность которого была бы меньше бесконечности, например, для задачи о раскраске графа [1]. В таких ситуациях используют статистические оценки погрешности работы алгоритма, то есть вычисляется отклонения результата полученного алгоритмом от оптимального решения на некотором ограниченном наборе индивидуальных задач, сгенерированных в соответствии с каким-либо законом распределения. К сожалению подобные статистические оценки погрешности могут существенным образом отличаться от теоретических и дают слабое представление о качестве работы алгоритма на наиболее плохих индивидуальных задачах. В частности для алгоритмов «в первый подходящий» и «в первый подходящий в порядке убывания» решения задачи об «упаковке в контейнеры» оценка в среднем получается равной 1.07 и 1.02, в то время как в худшем случае она равна 1.7 и 1.22 соответственно [1]. Мало того, для каких-либо алгоритмов, решающих задачи об

«упаковки» и «рюкзаке» в многомерном случае, вычисление даже оценки в среднем является проблематичным, так как современные компьютеры позволяют найти наилучшее решение только для очень небольшого количество предметов. Выходом является сравнение результатов работы алгоритма не с наилучшим решением (которое сложно найти), а с недостижимым «идеальным» (в случае рюкзака - весь объем контейнера, а в случае упаковки - сумма объемов всех предметов). Однако, такого рода оценка погрешности совершенно не дает представления о качестве работы алгоритма в худшем случае. Для 3-х мерной задачи о рюкзаке несложно привести пример, когда величина наилучшего возможного решения равна 0.125 от «идеального». В этом случае сравнение результата работы любого алгоритма с «идеальным» решением даст погрешность более **86%**, хотя предложенное решение может быть сколь угодно близко к наилучшему.

Из выше сказанного следует, что поиск даже примерных оценок погрешности работы алгоритмов решения 2-х и 3-х мерных задач об «упаковке» и «рюкзаке» очень не прост. И для всех алгоритмов существуют в некотором смысле «плохие» индивидуальные задачи, на которых вычисленная погрешность алгоритма будет большой. К счастью для оценки эффективности работы алгоритма на практике, достаточно вычислить среднюю погрешность по сравнению с «идеальным» решением, потому что в большинстве случаев на практике возникают задачи, параметры которых не сильно отклоняются от среднего значения. В любом случае на практике интересен быстрый алгоритм, который на **большинстве** поставленных задач дает хорошее решение, причем неважно, что на некоторых специальных примерах он дает большую погрешность. Именно их этих соображений, описанный в работе алгоритм мы будем оценивать по его средней погрешности по сравнению с «идеальным» решением.

Описанный в данной работе алгоритм, построен на основе совокупного использования различных разновидностей жадного «жадного» алгоритма, в котором функционал качества при выборе вариантов определяется специальным образом, на основе набора эвристик. Фактически описанный

алгоритм использует принцип «жадного» алгоритма в 3-х местах:

- выбор множества предметов одной высоты при сведении 3-х мерной задачи к 2-х мерной (см. 3.2);
- выбор 2-х мерного предмета для укладки в область (см. 3.4);
- выбор места для установки 2-х мерного предмета (см. 3.4).

Класс «жадных» алгоритмов достаточно хорошо изучен [1], [2] и даже определен тип задач, так называемые матроиды, на которых такие алгоритмы находят наилучшее решение [4]. Конечно NP-полные задачи не относятся к такому типу, однако все равно «жадный» алгоритм удобно взять за основу построения эффективного алгоритма их приближенного решения. В данной работе предложен пример такого алгоритма.

3 Алгоритма нахождения приближённого решения

Общая идея алгоритма решения задачи укладки заключается в следующем (см. рис. 1):

1. данные для укладки (список предметов) передаются алгоритму комбинирования предметов в прямоугольные блоки;
2. набор предметов комбинируются во всевозможные прямоугольные блоки, блоком называется набор предметов поставленных рядом, так что они образуют параллелепипед без щелей и пустот и представляющий собой аналог 3-х мерного предмета с другими размерами (см. 3.1);
3. сформированный набор блоков передается в алгоритм 3-х мерной укладки, который состоит из 3 частей:

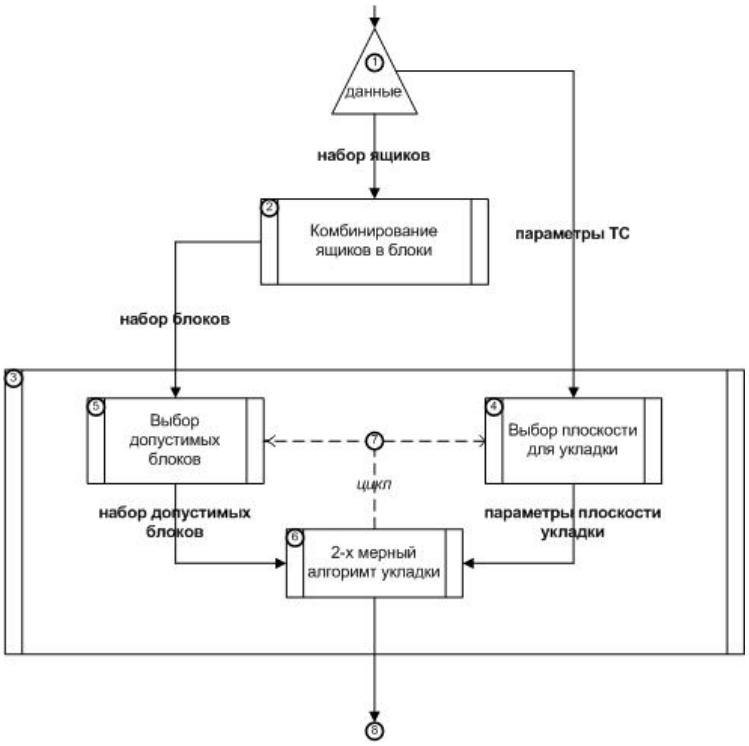

Рис. 1: Схема потоков данных алгоритма

(a) первая часть выбирает плоскость для укладки предметов, выбор плоскости для укладки является сведением задачи 3-х мерной укладки к задаче 2-х мерной укладки;

(b) вторая часть выбирает набор блоков, допустимых для укладки в текущую плоскость, при этом все эти блоки должны быть одной высоты для сведения 3-х мерной задачи для укладки к 2-х мерной задаче;

(c) третья часть является применением алгоритма 2-х мерной укладки к выбранной плоскости и допустимому набору блоков для укладки в эту плоскость;

4. алгоритм 3-х мерной укладки выполняется до тех пор, пока не уложены все предметы заказа и в контейнере остается свободный объем для укладки предметов;

5. после того как все предметы уложены или объем заполнен полно-

стью заполнен расчет оптимальной укладки предметов в контейнер считается завершенным.

3.1 Алгоритм комбинирования предметов

Алгоритм комбинирования предметов определяет способ формирования различных прямоугольных блоков из предметов. Для формирования всевозможных блоков используются два более простых алгоритма:

- формирования блоков из произвольного количества однотипных предметов одинаковой ориентации в пространстве;
- формирование блоков из двух разнотипных предметов, грани которых совпадают по размеру.

Эти два алгоритма применяются последовательно к результату предыдущего применения, при этом из результат удаляются блоки предметов, дублирующие друг друга. После того как последовательное применение этих шагов не образует новых блоков, приемлемых для укладки в контейнер, алгоритм формирования всевозможных блоков завершает работу.

3.2 Алгоритм 3-х мерной задачи (сведение к 2-х мерной)

Алгоритм решения 3-х мерной задачи представляет собой последовательное решение задачи 2-х мерной укладки предметов:

1. из предметов и прямоугольных блоков, сформированных на этапе 3.1, формируются наборы предметов одинаковой высоты, при этом учитываются все возможные положения предмета в пространстве, то есть каждый из предметов, имеющий разные линейные размеры, войдет в 3 набора, соответствующих каждому из его линейных размеров (если они все разные);

2. сформированные наборы сортируются в порядке убывания их функционала качества (см. 4.1), после чего считается, что первый набор является наиболее оптимальных для укладки в контейнер в текущий момент времени;

3. запускается цикл, в котором на каждом шаге в контейнер упаковывается наилучший (с точки зрения функционала 4.1) набор предметов:

   (a) в первую очередь вычисляется область укладки, в которую на данном шаге цикла должны быть помещены предметы (см. 3.3);

   (b) выбранная область определяется некоторым плоским основанием с ломаной границей и высотой, допустимой для укладки предметов (см. 3.3);

   (c) из отсортированного набора предметов выбирается ближайший набор, чья высота меньше или равна высоте области укладки;

   (d) для выбранных области и набора предметов запускается алгоритм решения 2-х мерной задачи укладки (см. 3.4);

   (e) после завершения укладки осуществляется проверка, если положение верхней грани предметов, установленных на этом шаге, совпадает с положением верхней грани предметов, образующих внутреннюю границу области, то объемы укладки сверху тех и других предметов, объединяются в один объем с общей внешней границей;

   (f) после того как предметы размещены в контейнере, меняется состав сформированных наборов укладки (из них исключаются уже размещенные предметы), поэтому запускается пересортировка всех наборов предметов в соответствии с убыванием функционала качества 4.1;

4. цикл завершается, когда в очередную область укладки не получилось поставить не одного предмета, при этом процедура расчета считается завершенной.

3.3 Выбор области укладки для 3-х мерного алгоритма

На каждом шаге алгоритма 3-х мерной укладки (см. 3.2), в уже частично заполненном контейнере, необходимо выбрать некоторую область, в которую будут упаковываться предметы одинаковой высоты с помощью алгоритма 2-х мерной укладки. Эта область должна иметь плоское основание с ломаной границей. Подобные плоскости образуются в процессе работы алгоритма 3-х мерной укладки предметов, когда предметы и блоки одинаковой высоты устанавливаются рядом и на одному уровне. Поверхность таких предметов и образует некоторую область для укладки других предметов. Таким образом подобная область укладки определяется двумя параллельными прямыми (борта контейнера) и двумя прямоугольными ломаными (край установленных предметов и граница начальной области, куда они были установлены).

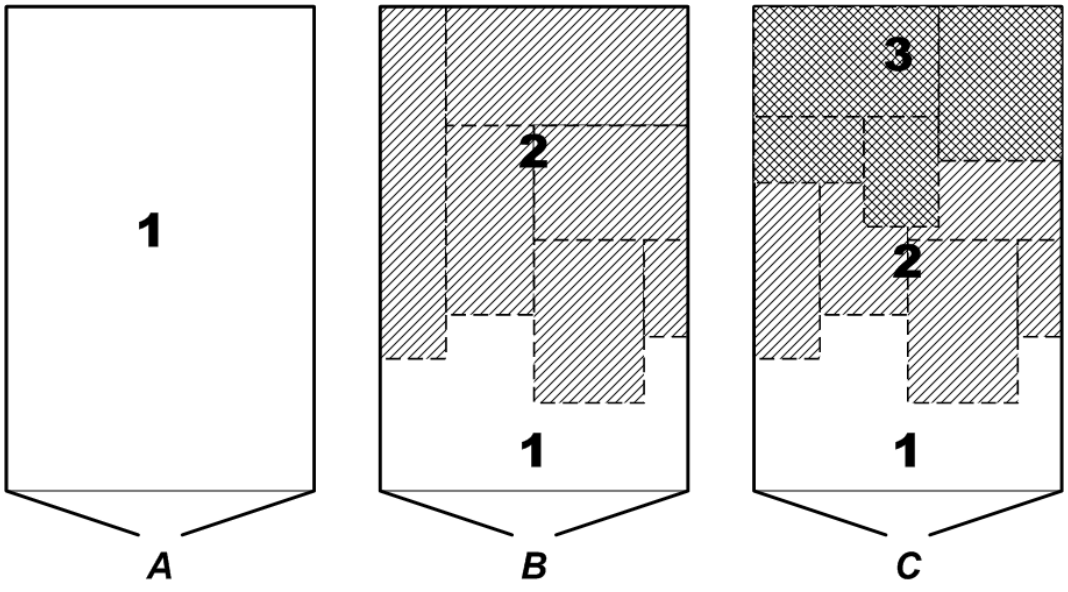

Рис. 2: Области укладки предметов

Алгоритм выбора такой плоскости следующий:

1. Изначально такой областью является все пространство контейнера (A1 на рисунке 2).

2. После установки в эту область нескольких предметов одинаковой высоты, становятся доступными для использования 2 области с плоским основанием: та что осталась от основания (B1 на рисунке 2) и та что сформирована установленными предметами (B2 на рисунке 2).

3. В качестве следующей для укладки области выбирается область над установленными на предыдущем шаге предметами (то есть на рисунке 2 последовательно заполняются области A1, B2, C3 и т.п).

4. В том случае, если сверху установленных предметов не удалось больше ничего поставить, выбирается оставшаяся область от предыдущего шага (C2 на рисунке 2).

5. Если не удалось заполнить и оставшуюся область, то осуществляется спуск еще на один шаг ниже (B1 на рисунке 2).

3.4 Алгоритм 2-х мерной укладки

Алгоритм укладки 2-х мерных предметов используется в алгоритме укладки 3-х мерных предметов 3.2. При этом 2-х мерный алгоритм должен вычислить оптимальную укладку 2-х мерных предметов в область, заданную двумя прямоугольными ломаными границами (внутренней и внешней). Боковые границы области определяются как прямые между концами внутренней и внешней ломаных. Ломаную линию будем называть прямоугольной, если все углы излома равны $+90^\circ$ или -90° (см. рисунок 3). Алгоритм 2-х мерной укладки заключается в выполнении следующих шагов:

1. все предметы сортируются в порядке убывания функционала качества (см. 4.2);

2. для каждого предмета в полученном списке выполняются следующие действия:

3. если площадь предмета больше пустой площади объема или его линейный размер больше максимального линейного размера объема

для укладки, то пропускаем этот предмет и переходим у следующему;

(a) для каждой из 4-х возможных ориентаций предмета в пространстве:

(b) для каждой из 4-х вершин предмета:

(c) для каждой из вершин внутренней ломанной и для каждой проекции вершин внешней ломанной на внутренюю:

   i. вычисляется новая область укладки, которая формируется из старой области и этим предметом, установленным в выбранной ориентации так, что выбранная вершина предмета совпадает с выбранной вершиной ломаной;

   ii. если в выбранном положении предмет пересекается с какой-либо границей области, то такое положение считается не допустимым и алгоритм переходит к следующему;

   iii. если выбранное положение предмета допустимо, то новая область укладки, сформированная этим положением, добавляется в список для анализа на шаге 3f.

(d) если для данного предмета нет ни одного допустимого положения, то алгоритм переходит к следующему предмету;

(e) после перебора всех возможных положений предмета, сформированные новые области укладки, которые получаются при установке предмета в каждом из таких положений, сортируются в порядке убывания функционала качества (см. 4.3);

(f) в полученном списке областей выбирается первая (с наибольшим значением функционала качества) и она определяет наилучшую позицию выбранного предмета в данной области, именно в этой позиции предмет и размещается;

(g) очевидно, что если предметы из списка, стоящие перед текущим не смогли поместиться в предыдущую область укладки, то

они не смогут поместиться и в новую, образованную этим установленным предметом, поэтому они исключается из дальнейшего перебора, а рассматриваются только следующие за текущим предметы.

4. цикл алгоритма 2-х мерной укладки завершается когда для какого-либо предмета найдено допустимое положение укладки, то есть в большинстве случаев пройдет только одна итерация этого цикла и будет выбрано наилучшее положение предмета, с наибольшим значением его функционала качества.

4 Использование функционалов качества

В различных местах алгоритма, для нахождения наилучшего решения, необходимо перебрать все возможные варианты сортировки некоторой последовательности элементов. К таким местам относятся шаг 2 алгоритма 3.2 и шаги 1 и 3е алгоритма 3.4. Однако подобный подход (полный перебор) приведет к неприемлемо долгому времени расчета. Поэтому, из некоторых эвристических соображений и на основе моделирования практических ситуаций формируется функционал качества элементов такой последовательности и все элементы сортируются в порядке убывания его значения. Отсортированные таким образом элементы, считаются единственным вариантом, который требуется рассмотреть. В каждом случае функционал качества представляет собой взвешенную сумму некоторых характеристик элемента. Набор этих характеристик и коэффициенты определяются как раз из эвристических соображений на основе анализа различных задач и вариантов укладки предметов.

4.1 Функционал качества для наборов 3-х мерных предметов

Функционал качества (см. 4) набора 3-х мерных предметов одинаковой высоты определяется следующим образом:

$$\mathfrak{A} = \alpha_1 * h + \alpha_2 * \sqrt[3]{\sum_{i \in M} v(i)} + \alpha_3 * \sqrt{\sum_{i \in M} s(i)} + \alpha_4 * \sum_{i \in M} p(i)$$

$$+\alpha_5 * \sqrt[3]{\sum_{i \in N} q(i) * v(i)} + \alpha_6 * \sqrt{\sum_{i \in N} q(i) * s(i)} + \alpha_7 * \sum_{i \in N} q(i) * p(i)$$

$$+\alpha_8 * \sqrt[3]{\frac{\sum_{i \in N} q(i) * v(i)}{\sum_{i \in N} q(i)}} + \alpha_9 * \sqrt{\frac{\sum_{i \in N} q(i) * s(i)}{\sum_{i \in N} q(i)}} + \alpha_{10} * \frac{\sum_{i \in N} q(i) * p(i)}{\sum_{i \in N} q(i)}$$

$$+\alpha_{11} * \frac{\sum_{i \in N} q(i) * \tilde{p}(i)}{\sum_{i \in N} q(i)} + \alpha_{12} * \sqrt{\frac{\sum_{i \in N} q(i) * \tilde{s}(i)}{\sum_{i \in N} q(i)}}$$

$$+\alpha_{13} * \sqrt[3]{\sigma_v(N)} + \alpha_{14} * \sqrt{\sigma_s(N)} + \alpha_{15} * \sigma_p(N) + \alpha_{16} * \sigma_{\tilde{p}}(N) + \alpha_{17} * \sqrt{\sigma_{\tilde{s}}(N)}$$

где

h - высота предметов в наборе;

N - множество типов предметов и прямоугольных блоков предметов входящих в набор;

M - множество типов предметов, из которых состоят блоки, входящие в набор;

$v(i)$ - объем предмета или блока типа i;

$s(i)$ - суммарная площадь всех граней предмета или блока типа i;

$p(i)$ - суммарная длина линейных размеров предмета или блока типа i;

$q(i)$ - количество предметов или блоков типа i в наборе;

$\tilde{s}(i)$ - разница площадей максимальной и минимальной граней предмета или прямоугольного блока типа i;

$\tilde{p}(i)$ - разница максимального и минимального линейных размеров предмета или прямоугольного блока типа i;

$\sigma_x(Y)$ - среднеквадратичное отклонение функции x по всем элементам множества Y;

4.2 Функционал качества для 2-х мерных предметов

Функционал качества (см. 4) 2-х мерных предметов, который используется в алгоритме 4.5, определяется следующим образом:

$$\mathfrak{B} = \beta_1 * \frac{\sqrt{s(i)}}{c(j)} + \beta_2 * \sqrt{s(i)} + \beta_3 * p(i) + \beta_4 * \tilde{p}(i)$$

где

(i) - количество предметов, входящих в состав 2-х мерного предмета i;

$s(i)$ - площадь 2-х мерного предмета i;

$p(i)$ - периметр 2-х мерного предмета i;

$\tilde{p}(i)$ - разница длин минимального и максимального ребер 2-х мерного предмета i;

4.3 Функционал качества для областей укладки

Функционал качества (см. 4) областей укладки, который используется в алгоритме 3.4, определяется следующим образом:

$$\mathfrak{C} = \gamma_1 * \mathfrak{B} + \gamma_2 * \sqrt{S} + \gamma_3 * \sum_{i \in N} l_i + \gamma_4 * \sum_{i \in N \wedge i:2} l_i$$
$$+\gamma_5 * l_{min} + \gamma_6 * l_{max} + \gamma_7 * \mu_{l_i}(i \in N) + \gamma_8 * \mu_{l_i}(i \in N \wedge x_i > x_{i+1})$$

$$+\gamma_9 * \mu_{l_{i+1}}(i \in T) + \gamma_{10} * \sigma_{l_{i+1}}(i \in T) + \gamma_{11} * \mu_{min(l_i, l_{i+2})}(i \in T) + \gamma_{12} * \sigma_{min(l_i, l_{i+2})}(i \in T)$$

$$+\gamma_{13} * \mu_{|min(l_i, l_{i+2}) - l_{i+1}|}(i \in T) + \gamma_{14} * \sigma_{|min(l_i, l_{i+2}) - l_{i+1}|}(i \in T)$$

где

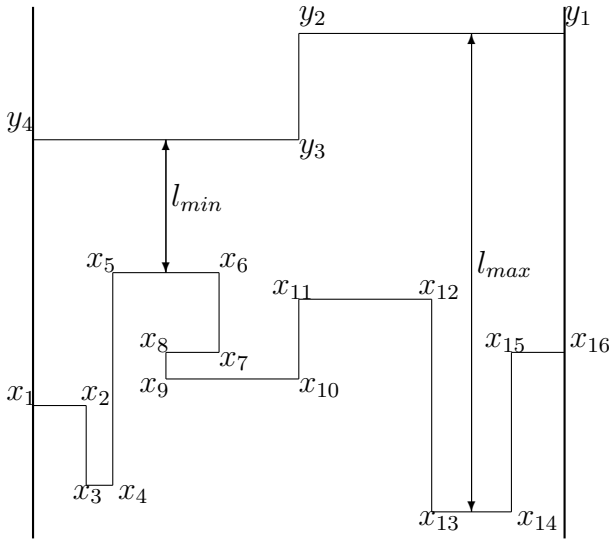

Рис. 3: Область загрузки

$\mathfrak{B}$ - значение функционала качества 2-х мерного предмета (см. 4.2), после установки которого образована данная область (см. 3.4);

S - площадь 2-х мерного предмета (см. 4.2), после установки которого образована данная область (см. 3.4);

N - множество вершин внутренней ломаной (исключая последнюю);

T - множество вершин внутренней ломаной, которые вместе с тремя последующими вершинами образуют тупик, под тупиком подразумевается последовательность из 3-х участков ломаных, крайние из которых имеют разную ориентацию (на рис. 3 это точки x_2, x_7 и x_{12});

l_i - длина отрезка (x_i, x_{i+1}) внутренней ломаной;

l_{min} - минимальное расстояние между внутренней и внешней границами области;

l_{max} - максимальное расстояние между внутренней и внешней границами области;

$\mu_x(Y)$ - математическое ожидание функции x по всем элементам множества Y;

$\sigma_x(Y)$ - среднеквадратичное отклонение функции x по всем элементам множества Y;

5 Теоретические оценки параметров работы алгоритма

5.1 Теоретическая оценка времени работы алгоритма

Предложенный алгоритм 3 является полиномиальным, то есть время его работы полиномиальным образом зависит от количества предметов в постановке задачи. Время работы алгоритма существенным образом зависит от двух параметров: k - количество типов предметов и $N = \sum_{i=1}^{k} n_i$ - количество предметов для укладки. Обозначим как P_{2d} - алгоритм решения 2-х мерной задачи о рюкзаке, описанный в разделе 3.4, а как P_{3d} - алгоритм решения 3-х мерной задачи о рюкзаке, описанный в разделе 3 [1]. Для доказательства оценки времени работы алгоритма нам потребуется доказать несколько дополнительных утверждений.

Определение 1 *Ломанная L, определённая своими вершинами (x_1, y_1), (x_2, y_1), ..., (x_n, y_n), называется прямоугольной, если $\forall i \in [1, n]$ либо $x_i = x_{i+1}$ либо $y_i = y_{i+1}$.*

Лемма 1 *Площадь замкнутой прямоугольной ломанной без самопересечений, определенной своими вершинами $L = \Big((x_1, y_1), (x_2, y_1), \ldots, (x_n, y_n)\Big)$, в любой системе координат равняется* [2]

$$\sum_{i=1}^{n} (x_{i+1} - x_i) * y_i$$

Доказательство: Рассмотрим произвольную прямоугольную ломанную, граница которой не имеет самопересечений (см. рис. 4А). Если к списку

[1] Эти алгоритмы одинаковым образом решают задачи о рюкзаке и задачу об упаковке, поэтому все сделанные оценки работы алгоритма будут верны для обеих задач.

[2] в качестве точки (x_{n+1}, y_{n+1}) берется точка (x_1, y_1).

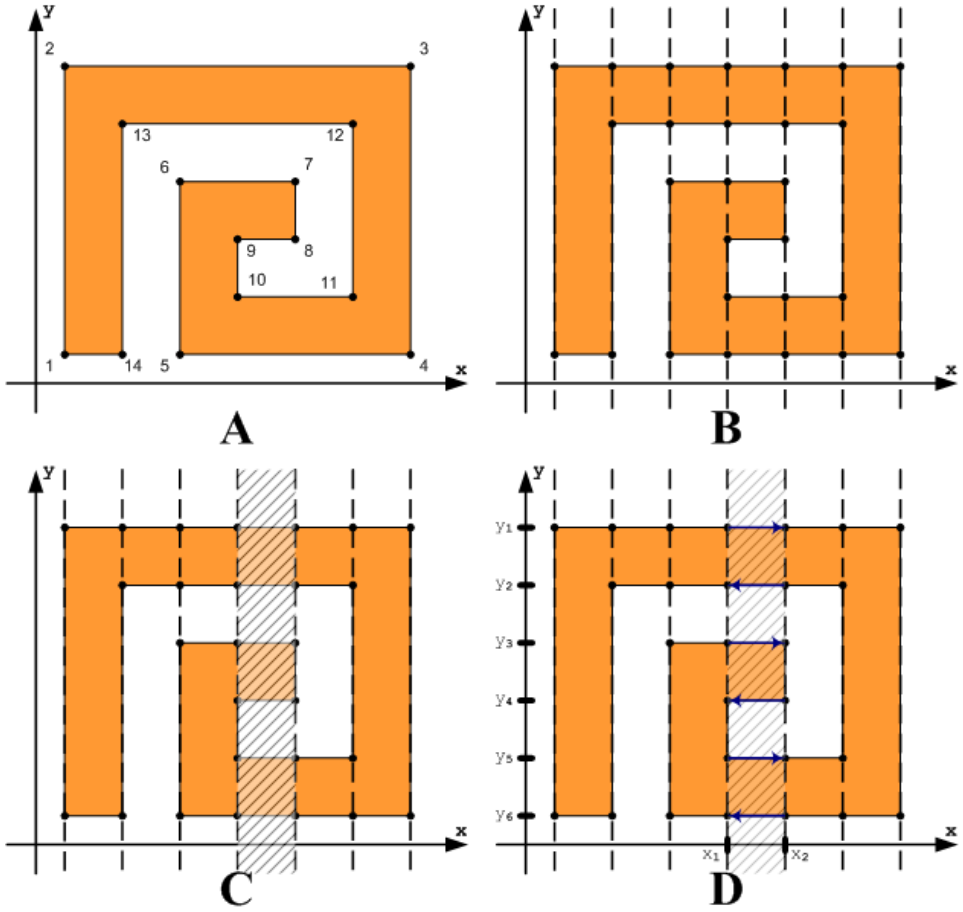

Рис. 4: Площадь прямоугольной ломаной

точек излома ломанной добавить любую точку (x', y') на горизонтальном участке границы ломаной, то это не изменит площади ломанной, и значение формулы с учетом новой точки не измениться, этот факт следует из формулы $(x' - x_i) * y_i + (x_{i+1} - x') * y_i = (x_{i+1} - x_i) * y_i$. Это значит, что лемму достаточно доказать для прямоугольной ломанной с произвольным количеством дополнительных точек, добавленных на горизонтальных участков. Через каждую точку ломанной проведем вертикальные прямые, и добавим к первоначальной ломанной все точки пересечения этих прямых с горизонтальными участками (см. рис. 4B).

Докажем лемму для полученной ломанной с дополнительным набором точек. Между любыми соседними проведенными вертикальным прямыми площадь ломанной разбивается на несколько прямоугольников (см.

4C). Площадь каждого такого прямоугольника вычисляется по формуле $\sum_{i=1}^{n}(x_{i+1}-x_i)*y_i$, а значит и суммарная площадь всех таких прямоугольников вычисляется по той же формуле $\square$.

Лемма 2 *Количество 3-х мерных прямоугольных блоков, у которых все входящие предметы одинаково ориентированы в пространстве, состоящих из n однотипных 3-х мерных предметов можно оценить как* $O(n \times ln^2(n))$.

Доказательство: Оценим $S_2(k)$ - количество пар (y,z), таких что $(y \times z) \leq k$. Из рисунка 5 очевидно неравенство $\sum_{i\in 2}^{k}\frac{1}{i} \leq \int_{i\in 1}^{k}\frac{1}{i}$, а значит:

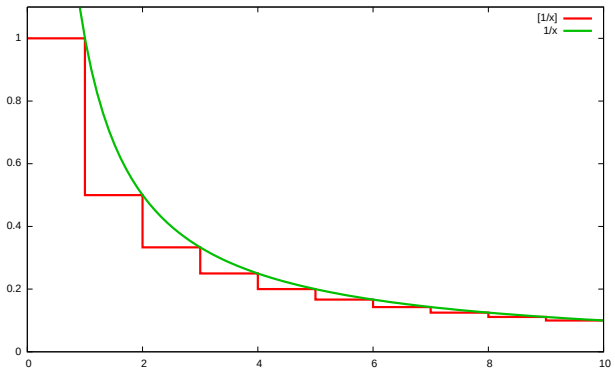

Рис. 5: Сумма $1/x$ меньше интеграла

$$S_2(k) = \sum_{y=1}^{k}\left[\frac{k}{y}\right] \leq \sum_{y=1}^{k}\frac{k}{y} = k\left(1+\sum_{y=2}^{k}\frac{1}{y}\right) \leq$$

$$\leq k\left(1+\int_1^k \frac{1}{y}dy\right) = k\left(1+ln(y)\Big|_1^k\right) = k(1+ln(k))$$

Пусть $S_3(n)$ - количество троек (x,y,z), таких что $(x \times y \times z) \leq n$. Из неравенства $(x \times y \times z) \leq n$ следует, что $(y \times z) \leq \frac{n}{x}$, а значит:

$$S_3(n) \leq \sum_{x=1}^{n} S_2\left(\frac{n}{x}\right) = \sum_{x=1}^{n}\frac{n}{x}\left(1+ln\left(\frac{n}{x}\right)\right) = \sum_{x=1}^{n}\frac{n}{x}\left(1+ln(n)-ln(x)\right) \leq \sum_{x=1}^{n}\frac{n(1+ln(n))}{x}$$

$$= n(1+ln(n))\sum_{x=1}^{n}\frac{1}{x} \leq n(1+ln(n))(1+ln(n)) \sim O(n \times ln^2(n))$$

Значит, количество блоков из n предметов тоже можно оценить как $O(n \times ln^2(n))$ $\square$.

Лемма 3 *Время вычисления функционала* $\mathfrak{C}$*, который определяется в пункте 4.3, оценивается как* $O(n)$*, где* n *- суммарное количество вершин внутренней и внешней границы.*

Доказательство: для оценки времени работы функционала достаточно определить время работы каждого из параметров, взвешенной суммой которых является функционал. Так как количество этих параметров фиксировано и не зависит от n, то время вычисления всего функционала будет оцениваться эквивалентно времени вычисления самого сложного параметра. Каждая характеристика функционала либо на зависит от количества вершин ломаной, либо вычисляется как сумма некоторых характеристик длин отрезков ломаной с последующим применением одноместной функции, не зависящей от n. А это значит, сложность вычисления каждой из характеристик и всего функционала $\mathfrak{C}$ оценивается как $O(n)$ $\square$.

Лемма 4 *Время вычисления функционала* $\mathfrak{A}$*, который определяется в пункте 4.1, оценивается как* $O(n)$*, где* n *- количество предметов в множестве, к которому применяется функционал.*

Доказательство: для оценки времени работы функционала достаточно определить время работы каждого из параметров, взвешенной суммой которых является функционал. Так как количество этих параметров фиксировано и не зависит от n, то время вычисления всего функционала будет оцениваться эквивалентно времени вычисления самого сложного параметра. Каждая характеристика функционала либо на зависит от количества предметов набора, либо вычисляется как сумма некоторых характеристик каждого предмета из набора с последующим применением одноместной функции, не зависящей от n. А это значит, сложность вычисления каждой из характеристик и всего функционала $\mathfrak{A}$ оценивается как $O(n)$ $\square$.

Теорема 1 *Если* $k = N$ *(все предметы разных типов), то время работы алгоритма* P_{2d} *можно оценить как* $O(N^3)$*.*

Доказательство: функционал качества $\mathfrak{B}$, который определяется в пункте 4.2, не зависит от количества предметов, а вычисляется на основе параметров одного предмета. Таким образом время выполнения шага 1 алгоритма оценивается как время вычисления функционала $\mathfrak{B}$ для каждого предмета - $O(N)$ и время сортировки предметов в порядке убывания значения этого функционала - $O(N*log(N))$, что в сумме дает $O(N*log(N))$. В процессе работы алгоритма возникают ломанные границы, которые образуются установленными предметами. Очевидно, что количество вершин в такой границе не может превышать $4*N$, так как любая вершина ломаной границы совпадает с одной из вершин ранее установленного предмета, а значит их не может быть больше чем суммарное количество всех вершин всех предметов. Таким образом на шаге 3 алгоритма для каждого предмета проверяется не более чем $2*4*4*N = 32*N \sim O(N)$ возможных вариантов установки предмета (2 варианта поворота предмета, 4 варианта вершины предмета, $4*N$ вариантов вершин ломанной). Из леммы 3 следует, что время вычисления функционала $\mathfrak{C}$ для каждого предмета оценивается как $O(N)$, а значит полное время поиска наилучшего положения предмета на шаге 3 оценивается как $O(N^2)$. Из критерия остановки цикла 4 следует, что в большинстве случаев поиск наилучшего положения будет производится для одного или двух предметов, однако в худшем случае этот поиск будет осуществляться для N предметов, а значит оценка времени работы всего алгоритма $O(N*log(N))+N*O(N^2)$, что эквивалентно $O(N^3)$ $\square$.

Теорема 2 *Если $k = N$ (все предметы разных типов), то время работы алгоритма P_{3d} можно оценить как $O(N^4)$.*

Доказательство: так как $k = N$, то комбинирования предметов, определенное в разделе 3.1, не происходит, а значит время работы алгоритма P_{3d} определяется временем работы той части алгоритма, которая определена в пункте 3.2. На шаге 1 может быть сформировано не больше, чем $3*N$ множеств предметов одинаковой высоты, при этом количество

предметов в каждом таком множестве не более чем N, а суммарное количество предметов по всем множествам не более, чем $3 * N$. На шаге 2 происходит вычисление функционала качества для каждого множества и их сортировка. Из леммы 4 следует, что вычисления функционала $\mathfrak{A}$ для всех множеств оценивается как $3 * N * O(N) \sim O(N^2)$, а время сортировки - $O(3 * N * log(3 * N))$. То есть оценка времени выполнения шага 2 - $3 * N * O(N) + O(3 * N * log(3 * N)) \sim O(N^2)$. Для оценки времени выполнения одной итерации цикла 3 необходимо определить оценку времени выполнения каждой ее части:

- 3a - выбор плоскости укладки 3.3 строго детерминирован и не зависит ни от количества предметов, ни от количества уже установленных плоскостей, ни от их конфигурации.
- 3b - аналогично 3a.
- 3c - поиск ближайшего допустимого множества, в худшем случае выльется в перебор всех $3 * N$ множеств, а значит оценивается как $O(N)$.
- 3d - из теоремы 1 следует, что время укладки выбранного множества предметов в выбранную область оценивается как $O((3 * N)^3) \sim O(N^3)$.
- 3e - новое вычисление функционала $\mathfrak{A}$ делается за $O(N)$, и пересортировка за $O(N)$, так как достаточно переставить выбранное множество, порядок остальных не мог изменится.

Таким образом время выполнения одной итерации цикла 3 оценивается как $O(N) + O(N^3) + 2 * O(N) \sim O(N^3)$. Осталось вычислить максимальное количество итераций цикла. На практике количество множеств предметов одинаковой высоты и, соответственно, итераций цикла получается существенно меньшим количества предметов. Однако для оценки времени работы надо использовать наихудшую ситуацию, которая заключается в том, что в каждой итерации цикла мы укладываем только один предмет,

одного множества. Не уложить никакой предмет на очередной итерации мы не можем, так как это и является критерием остановки цикла 4. Таким образом максимальное количество итераций цикла - $3*N$, а значит полное время работы алгоритма 3.2 оценивается как $O(N^2)+3*N*O(N^3) \sim O(N^4)$ $\square$.

Теорема 3 *$\forall k, \forall(n_1, n_2, \ldots, n_k)$ время работы алгоритма P_{3d} можно оценить как $O(N^4 \times ln^8(N))$, где $N = \sum_{i=1}^{k} n_i$.*

Доказательство: для вычисления времени работы алгоритма необходимо вычислить количество блоков, который можно скомбинировать из указанного набора предметов. Из леммы 2 следует, что количество блоков, составленных из предметов i-го типа можно оценить как $O(n_i \times ln^2(n_i))$. А значит из неравенства

$$\sum_{i=1}^{k} \left(n_i \times ln^2(n_i)\right) \leq \sum_{i=1}^{k} \left(n_i \times ln^2(N)\right) = ln^2(N) \sum_{i=1}^{k} n_i = N \times ln^2(N)$$

следует, что количество блоков можно оценить как $O(N \times ln^2(N))$. Время работы алгоритма P_{3d}, примененного к полученному набору блоков и предметов, в соответствии с теоремой 2, можно оценить как $O((N \times ln^2(N))^4) \sim O(N^4 \times ln^8(N))$ $\square$.

Замечание: полученная оценка времени работы алгоритма носит сугубо теоретический характер, так как использование прямоугольных блоков, состоящих из большого количества предметов в алгоритме P_{3d} приводит к тому, что после каждой установки такого блока, количество оставшихся предметов уменьшается не на 1, а на количество элементов блока плюс количество тех блоков, которые уже не могут быть составлены из оставшегося количества предметов такого типа. К сожалению учесть этот факт в теоретических оценках представляется проблематичным, но большое количество экспериментов показывает, что время работы алгоритма P_{3d} с комбинированием блоков примерно равно времени работы алгоритма на тех же начальных предметах, но без комбинирования блоков. То есть при практическом использовании алгоритма можно ориентироваться

на оценку его скорости, сформулированную в теореме 2, которая равна $O(N^4)$.

5.2 Теоретическая оценка качества работы алгоритма

В разделе 2 показано, что вычисление точной теоретической оценки качества работы алгоритма по сравнению с «наилучшим» и даже по сравнению с «идеальным» решением является сложной проблемой даже в случае более простых NP-полных задач. К сожалению, найти такие оценки для предложенных алгоритмов P_{2d} и P_{3d} решения 2-х и 3-х мерных задач о рюкзаке и упаковке не удаётся. Однако, можно доказать некоторые факты, показывающие, что сам подход, который используется в описанный алгоритмах вполне эффективен и в некоторых специальных случаях дает высокое качество работы. А значит при должном конфигурировании предложенных алгоритмов, они могу эффективно решать задачи 2-х и 3-х мерной укладки предметов.

Определение 2 *Будем называть решение индивидуальной задачи о рюкзаке «идеальным», если объём упакованных предметов равен 100% объёма контейнера.*

Теорема 4 *Для любой индивидуальной 2-х мерной задачи о рюкзаке, у которой «наилучшее» решение является «идеальным», существуют такие функционалы качества* $\mathfrak{B}$ *и* $\mathfrak{C}$ *с линейной сложностью вычисления, при которых алгоритм* P_{2d} *вычисляет это «наилучшее» решение.*

Доказательство: пусть есть некоторая индивидуальная 2-х мерная задача о рюкзаке, и ее «оптимальное» решение, которое полностью заполняет 2-х мерный объем. Для любого предмета в этом решении, любая его вершина либо совпадает с какой-либо вершиной другого предмета, либо с вершиной объема для укладки, в этом легко убедится, если рассмотреть всевозможные конфигурации границ предметов в выбранной вершине. Из этого факта следует, что для указанного решения можно вы-

брать такую последовательность установки предметов, приводящую к этому решению, что на каждом шаге очередной предмет какой-либо своей вершиной будет устанавливаться либо в вершину другого предмета, либо в вершину объема. А это значит, что в данной последовательности на каждом шаге предмет своей вершиной будет устанавливаться в одну из вершин границы объема, оставшегося от установки предыдущих предметов. Пусть $(b_1, b_2, \ldots, b_n)$ - полученная последовательность предметов, а $\Big((l_1^1, l_1^2), (l_2^1, l_2^2), \ldots, (l_n^1, l_n^2))\Big)$ - соответствующие им линейные размеры. Определим следующие вспомогательные функции:

- $I_a(x)$ - одноместная функция-индикатор, которая принимает значение 1 только при $x = a$, а в остальных случаях равна 0.
- $I_{a,b}(x, y)$ - двухместная функция-индикатор, которая принимает значение 1 только при $x = a$ и $y = b$, а в остальных случаях равна 0.
- $I_{\vec{d}}(\vec{v})$ - многомерная функция-индикатор, которая принимает значение 1 только при равенстве векторов $\vec{d} = \vec{v}$, а в остальных случаях равна 0.

и константы $S_1, S_1, \ldots, S_n$:

$$S_1 = 0, \forall m \in [2, n] S_m = \sum_{i=1}^{m-1} (l_i^1 * l_i^2)$$

Рассмотрим следующий функционал $\mathfrak{B}$, зависящий от линейных размеров исследуемого предмета x, y и суммарной площади ранее установленных предметов s:

$$\mathfrak{B}(x, y, s) = \sum_{i=1}^{n} (n - i + 1) * I_{S_i}(s) * I_{l_i^1, l_i^2}(x, y)$$

Очевидно, что время вычисления этого функционала не превосходит $O(N)$ и его применение к последовательности предметов $(b_1, b_2, \ldots, b_n)$ даст убывающую последовательность чисел $n, n-1, \ldots, 1$. То есть использование

подобного функционала в алгоритме P_{2d} приведет к тому, что предметы будут устанавливаться алгоритмом в заранее определенной последовательности. Функционала $\mathfrak{C}$ используется для выбора «наилучшей» границы при переборе вариантов установки предмета. Так как любая конфигурация предметов в заполняемом объеме образует только прямоугольную границу, то любую такую границу определить вектором $\vec{v}$ размерности $4N$, где координата i определяет длину i-го участка границы (если граница содержит меньше $4N$ участков, то оставшиеся координаты вектора устанавливаются в 0). Пусть $\vec{d}_1, \vec{d}_2, \ldots, \vec{d}_n$ - последовательность константных векторов, которые определяют границы, получившиеся последовательной установкой предметов $(b_1, b_2, \ldots, b_n)$ при формировании «оптимального» решения. В качестве $\mathfrak{C}$ выбираем функционал, который зависит от исследуемого варианта границы $\vec{v}$ и суммарной площади ранее установленных предметов s:

$$\mathfrak{C}(\vec{v}, s) = \sum_{i=1}^{n} (n - i + 1) * I_{S_i}(s) * I_{\vec{d}_i}(\vec{v})$$

Данный функционал, как и определенный ранее $\mathfrak{B}$ имеет оценку времени работы как $O(N)$, а его использование в алгоритме P_{2d} приведет к тому, что предметы будут устанавливаться в заранее определенные места. Таким образом предложенные функционалы $\mathfrak{B}$ и $\mathfrak{C}$ отвечают требованиям теоремы $\square$.

Аналогичное утверждение можно доказать для алгоритма P_{3d}, но сложность функционалов будет квадратичной.

Теорема 5 *Для любой индивидуальной 3-х мерной задачи о рюкзаке, у которой «оптимальное» решение является «идеальным», существуют такие функционалы качества $\mathfrak{A}$, $\mathfrak{B}$ и $\mathfrak{C}$ с квадратичной сложностью вычисления, при которых алгоритм P_{3d} вычисляет это «оптимальное» решение.*

Доказательство: пусть есть некоторая индивидуальная 3-х мерная задача о рюкзаке, и ее «оптимальное» решение, которое полностью заполняет 3-х мерный объем. Любой предмет в этом решении полностью опирается

либо на дно объема, либо на несколько других предметов, верхние грани которых находятся на одном уровне. Отсюда следует, что вся такая укладка может быть разбита на последовательные наборы $U_1, \ldots, U_r$ установленных предметов таким образом, чтобы в каждом наборе были предметы одной высоты, все предметы очередного набора были на одной высоте и полностью опирались либо на дно, либо на предметы ранее установленных наборов. Покажем, что можно подобрать таким образом функционал $\mathfrak{A}$, что алгоритм P_{3d} при таком функционале, будет выбирать наборы предметов в этом фиксированном порядке. Пусть набор U_i состоит из предметов $(b_{i1}, b_{i2}, \ldots, b_{in_i})$, а каждый предмет b_{ij} имеет линейные размеры $(l^1_{ij}, l^2_{ij}, l^3_{ij})$. Определим константы $V_1, V_2, \ldots, V_r$ следующим образом:

$$V_1 = 0, \forall m \in [2, r] V_m = \sum_{i=1}^{m-1} \sum_{j=1}^{n_m} (l^1_{ij} * l^2_{ij} * l^3_{ij})$$

Каждый набор предметов U_i однозначно определяется вектором размерности $3N$, координаты которого являются списком линейных размеров предметов $(b_{i1}, b_{i2}, \ldots, b_{in_i})$, отсортированные по уменьшению объема (если предметов меньше N, то оставшиеся координаты вектора устанавливаются в 0). Для последовательности наборов $U_1, \ldots, U_r$, определяющих оптимальное решение, соответствующие им вектора такого типа обозначим как $\vec{u}_1, \ldots, \vec{u}_r$. В качестве $\mathfrak{A}$ выбираем функционал, который зависит от вектора $\vec{w}$, заданного исследуемым набором границы, и суммарного объема всех ранее установленных предметов v:

$$\mathfrak{A}(\vec{w}, v) = \sum_{i=1}^{r} (r - i + 1) * I_{V_i}(v) * I_{\vec{u}_i}(\vec{w})$$

Время вычисления этого функционала $O(N)$ и его использование в алгоритме P_{3d} обеспечит выбор алгоритмом заранее определенной последовательности наборов предметов одной высоты. В качестве функционалов $\mathfrak{B}$ и $\mathfrak{C}$ нельзя явно использовать те, которые определены в теореме 4, так как в процессе работы алгоритма P_{3d} приходится решать несколько индивидуальных задач 2-х мерной укладки, каждая из которых требует особого определения этих функционалов. Однако, можно составить единые функ-

ционалы таким образом, чтобы на каждом шаге алгоритма P_{3d} они редуцировались в функционалы, определенные в теореме 4. Пусть $\mathfrak{B}_1, \ldots, \mathfrak{B}_r$ и $\mathfrak{C}_1, \ldots, \mathfrak{C}_r$ функционалы определенный способом, указанным в теореме 4 для решения каждой индивидуальной 2-х мерной задачи о рюкзаке, возникающей в процессе алгоритма P_{3d}. Тогда единые функционалы $\mathfrak{B}$ и $\mathfrak{C}$ помимо аргументов указанных в теореме 4 будут зависеть еще и от суммарного объема v наборов предметов, установленных ранее:

$$\mathfrak{B}(x, y, s, v) = \sum_{i=1}^{r} I_{V_i}(v) * \mathfrak{B}_i(x, y, s)$$

$$\mathfrak{C}(\vec{u}, s, v) = \sum_{i=1}^{r} I_{V_i}(v) * \mathfrak{C}_i(\vec{u}, s)$$

Так как значение r не превышает $4N$, а $\forall i \in [1, r]$ время работы функционалов $\mathfrak{B}_i$ и $\mathfrak{C}_i$ оценивается как $O(N)$, то время работы функционалов $\mathfrak{B}$ и $\mathfrak{C}$ - $O(N^2)$, а применение этих функционалов в совокупности с ранее определенным $\mathfrak{A}$ в алгоритме P_{3d} приведет к тому, что предметы будут расставлены в требуемой последовательности и на нужные места $\square$.

На самом деле из доказательств теорем 4 и 5 очевидным образом следует более сильное утверждение, которое показывает что потенциальное качество алгоритма P_{3d} при правильном подборе функционала достаточно велико:

Определение 3 *Решение называется* ***допустимым*** *когда все предметы установлены либо на дно объема, либо на объединение верхних граней других предметов и какое-либо из вертикальных ребер 3-х мерного предмета частично или полностью совпадает с ребром другого предмета или объема для укладки.*

Теорема 6 *Для любой индивидуальной 3-х мерной задачи о рюкзаке и любого* ***допустимого*** *решения этой задачи существуют такие функционалы качества $\mathfrak{A}$, $\mathfrak{B}$ и $\mathfrak{C}$ с квадратичной сложностью вычисления, при которых алгоритм P_{3d} вычисляет это решение.*

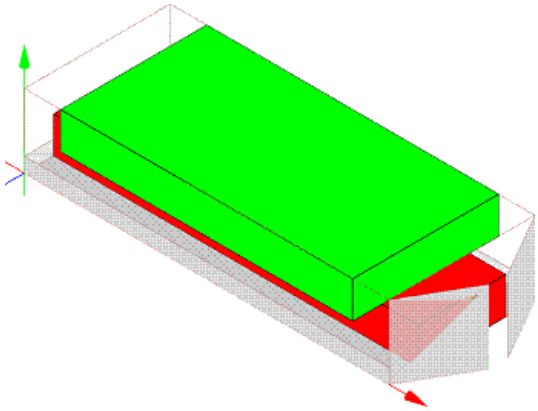

Рис. 6: Пример задачи, когда «наилучшее решение» решение нельзя свести к допустимому

Однако, не всегда «оптимальное решение» 3-х мерной индивидуальной задачи о рюкзаке является допустимым с точки зрения теоремы 6.

На рисунке 6 показан элементарный пример индивидуальной 3-х мерной задачи о рюкзаке, оптимальное решение которой нельзя свести к допустимому решению. Оценить, насколько наилучшее допустимое решение в среднем хуже, чем оптимальное не представляется возможным, поэтому из теорем 4, 5 и 6 сложно оценить качество алгоритмов P_{2d} и P_{3d} относительно «наилучшего решения» или «идеального решения», однако статистические расчеты показывают, что в среднем алгоритмы дают приемлемое качество расчета даже относительно «идеального решения».

5.3 Экспериментальные оценки параметров работы алгоритма

При оценке времени и качества работы алгоритма наиболее интересными являются результаты его работы на большом количестве предметов. Так как задача является NP-полной, нахождение «наилучшего» решения для произвольной индивидуальной задачи с большим количеством предметов не представляется возможным. Эксперименты показали, что нахождения решения путём полного перебора становиться невыполнимо

долгим при количестве предметов больше 10. Это означает, что решение, найденное предложенным алгоритмом на произвольной индивидуальной задаче нельзя сравнить с «наилучшим». Поэтому для оценки времени и качества работы алгоритма будут использоваться некоторые специальные индивидуальные задачи, про которые точно известно что их «наилучшее» решение является «идеальным». То есть для задач, «наилучшее» решение которых заполняет объем контейнера на 100%. Для поиска таких индивидуальных задач достаточно «распилить» контейнер для укладки набором плоскостей, параллельных координатным плоскостям. При таком «распиливании» весь объем для заполнения распадается на набор предметов, которые гарантированно могут быть упакованы в этот контейнер и их суммарный объем равен объему контейнера. Очевидно, что для любого N и k легко подобрать такой набор плоскостей, что после «распиливания» этими плоскостями объема для укладки, получится индивидуальная задача в которой количество предметов больше N, а количество типов разных предметов больше k.

Для оценки качества и скорости алгоритма, описанного в 3 было проведено 1050 расчетов оптимальной укладки для индивидуальных задач, составленных по принципу описанному выше. Общее время эксперимента составило более 5*251397 секунд, что равно 14.5 суток непрерывных расчетов. В процессе эксперимента выбиралось некоторое число N - количество предметов и K - количество разных типов предметов ($K \leq N$). Для каждой пары N и K производилось 5 экспериментов и вычислялось среднее качество и скорость работы алгоритма.

На рисунке 7 показана зависимость качества алгоритма от количества предметов и типов различных предметов в задаче. Здесь под качеством понимается отношение суммарного объема предметов, упакованных алгоритмом, к объему контейнера. Исходя из особенностей формирования каждой индивидуальной задачи можно сказать, что это число определяет качество работы алгоритма относительно наилучшего решения. Из графика видно, что для при любом количестве предметов, если они все имеют

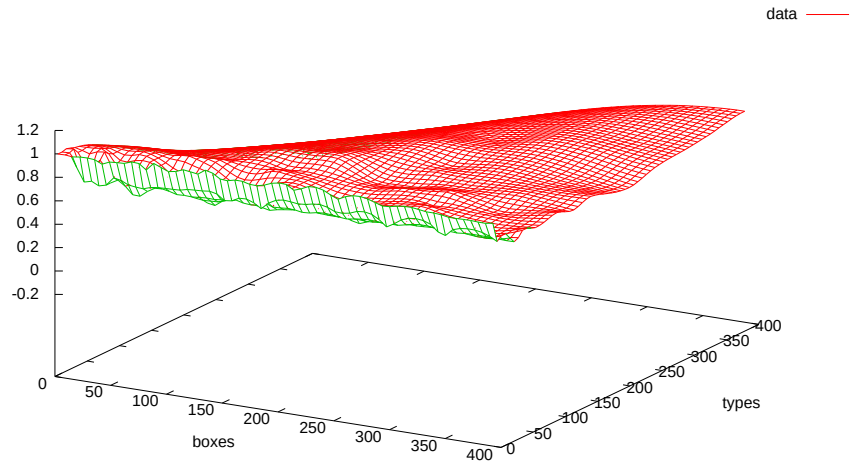

Рис. 7: Зависимость качества работы алгоритм от количества предметов и типов разных предметов

один размер, алгоритм находит наилучшее решение (что не удивительно). Для разноразмерных предметов качество алгоритма примерно одинаково для любого количества предметов и размеров и равно примерно **0.85**.

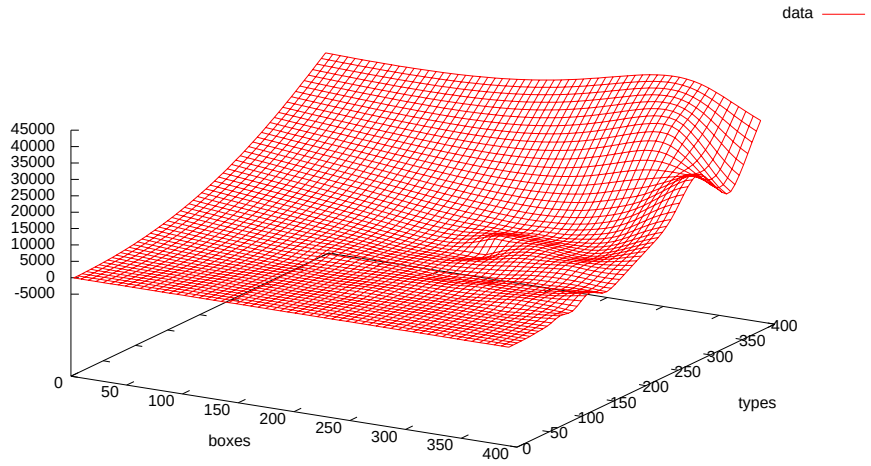

Рис. 8: Зависимость времени работы алгоритм от количества предметов и различных типов предметов

На рисунке 8 показана зависимость времени работы алгоритма от количества предметов и типов различных предметов в задаче. Видно, алгоритм работает практически мгновенно на любом количестве одинаковых предметов и время его работы увеличивается с возрастанием предметов и типов предметов. Как было показано в разделе 5.1 этот рост имеет полиномиальную природу.

6 Заключение

В работе предложен алгоритм нахождения приближённого решения задачи о рюкзаке в 3-х мерном случае и приведены теоретические и экспериментальные оценки скорости и качества работы этого алгоритма. Не смотря на то, что алгоритм не находит наилучшее решения задачи о рюкзаке, его высокая скорость и вполне приемлемый результат позволяют использовать его в реальных задачах, возникающих на практике. Это подтверждается тем, что программная система на базе этого алгоритма успешно внедрена и используется на практике на многих предприятиях. Помимо высокой скорости работы основным преимуществом алгоритма является его высокая адаптивность, позволяющая реализовывать модификации алгоритма для учета дополнительных ограничений и легко модернизировать его для нужд конечных пользователей.

Хотелось бы отметить особое участие в разработке алгоритма и выразить огромную благодарность моему научному руководителю Строгалову Александру Сергеевичу. Именно совместно с ним была выработана основная идея алгоритма, а его ценные советы и замечания помогли решить много принципиальных трудностей, возникавших при адаптации алгоритма к новым требованиям.

Список литературы

[1] *М. Гэри, Д. Джонсон* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

[2] *А.Ахо, Дж.Хопкрофт, Дж.Ульман* Построение и анализ вычислительных алгоритмов. М.:Мир 1979, 536 стр.

[3] *В.В. Псиола.* О приближенном решении 3-х мерной задачи об упаковке на основе эвристик. Интеллектуальные системы, 11, вып. 1-4, 2007

[4] *М. Липский* Комбинаторика для программистов. М.: Мир, 1988.

[5] *Т.Ч. Ху, М.Т. Шинг* Комбинаторные алгоритмы. Нижний Новгород: Изд-во Нижегородского госуниверситета им.Н.И.Лобачевского, 2004. - 330 с.