Упаковка по корзинам

Лахтин Антон

Аннотация

Данная работа посвящена задаче упаковки по корзины — классической **NP**-трудной комбинаторной задаче. В работе исследуются границы её аппроксимируемости. В работе приводится доказательство того, что для любого сколь угодно малого $\varepsilon > 0$ не существует полиномиального алгоритма, строящего $(3/2-\varepsilon)$-приближенное решение, если $\mathbf{P} \neq \mathbf{NP}$. С другой стороны, в работе описывается и реализуется алгоритм, который для любого $\varepsilon > 0$ находит решение, использующее не более $(1+\varepsilon)OPT + 1$ корзин, где OPT — оптимальное число корзин.

1 Введение

Задача упаковки по корзинам является одной из фундаментальных проблем комбинаторной оптимизации и имеет многочисленные практические приложения в логистике, распределении памяти в вычислительных системах и других областях. Задача заключается в том, чтобы найти оптимальное количество контейнеров фиксированного размера, в которые можно поместить несколько объектов данных размеров.

Данная задача относится к классу **NP**-трудных, что делает поиск точного решения для больших входных данных нереалистичным. Поэтому значительное внимание в литературе уделяется разработке приближённых алгоритмов, которые за полиномиальное время находят решение, близкое к оптимальному. Особый интерес представляют аппроксимационные схемы, позволяющие получать решения с гарантированной точностью.

История изучения задачи восходит к задаче раскроя (Cutting Stock Problem), впервые системно исследованной в работе Гилмора и Гомори (1961) [1]. Как самостоятельная проблема, упаковка по корзинам была выделена и названа в начале 1970-х годов в контексте анализа алгоритмов распределения памяти (Гэри, Грэхем и Ульман, 1972 [2]). Вскоре была доказана её NP-полнота в работе Гэри и Джонсона (1975) [3], что сделало разработку приближённых алгоритмов основным направлением исследований.

В вопросе нахождения аппроксимационных алгоритмов значительная часть ранних результатов связана с анализом простых эвристик. В ранних работах Джонсона и др. (1974) [4] были предложены такие алгоритмы, как Next-Fit, First-Fit и Best-Fit, которые гарантируют использование не более чем $2OPT + 1$ корзин. Значительным прорывом стало создание асимптотической полиномиальной аппроксимационной схемы (APTAS) де ла Вегой и Люкером в 1981 году [5], которая для любого $\varepsilon > 0$ находит упаковку, использующую не более $(1+\varepsilon)OPT + O(1)$ корзин. Задача о получении аппроксимационной схемы, дающей оценку вида $(1+\varepsilon)OPT+1$, была решена позже с использованием более сложных методов, включая линейное программирование и техники округления.

Важность задачи подчёркивается не только её теоретической сложностью, но и широкой применимостью. Например, в облачных вычислениях она возникает при распределении задач по серверам с ограниченными ресурсами, в транспортировке — при загрузке контейнеров, в производстве — при раскрое материалов.

В данной работе рассматривается как теоретическая, так и практическая стороны задачи. Описывается доказательство того, что задача о $(\frac{3}{2}-\varepsilon)$-приближении является **NP**-трудной для любого $\varepsilon > 0$, что устанавливает нижнюю границу аппроксимируемости. Также строится и исследуется алгоритм, который для любого $\varepsilon > 0$ находит распределение данных элементов по не более чем $(1+\varepsilon)OPT + 1$ контейнерам.

2 Техническая часть

Формально задача упаковки по корзинам определяется следующим образом. Пусть задано множество n предметов и каждому предмету $i \in [n]$ сопоставлен размер $s_i \in (0, 1]$. Требуется найти минимальное число $k \in \mathbb{N}$ и разбиение $I = B_1 \cup B_2 \cup \cdots \cup B_k$ на непересекающиеся подмножества (корзины), такие что:

$$\forall j \in [k] : \sum_{i \in B_j} s_i \leq 1$$

Эту задачу обозначим за BIN PACKING. И если известны входные данные s, то такую задачу обозначим за BIN PACKING(s)

Задача об α-приближении – задача о нахождении разбиения на k корзин, для какого-то $k \leq \alpha OPT$.

Минимальное число корзин, которые необходимы для выполнения условий задачи будем обозначать как OPT. Для конкретного набора предметов J ответом на задачу BIN PACKING обозначим за $OPT(J)$.

Для доказательства теоремы 3.1 опишем постановку классической **NP**-трудной задачи PARTITION. Дано конечное множество $A = \{a_1, a_2, a_3, ...a_n\}$ с целыми положительными весами $s(a_i)$. Требуется определить, существует ли подмножество $A' \subseteq A$ такое, что

$$\sum_{a_i \in A'} s(a_i) = \sum_{a_i \in A \setminus A'} s(a_i)$$

Задачу PARTITION с входными данными A и s обозначим за PARTITION(A, s).

План доказательства работы алгоритма, дающего разбиение на $(1+\varepsilon)OPT+1$ корзин описан в книге Вазирани [6]. Ниже приведены доказательства нескольких утверждений, необходимых для доказательства корректности алгоритма.

Лемма 2.1. *Пусть $\varepsilon > 0$ фиксировано, и пусть K — фиксированное неотрицательное целое число. Рассмотрим ограничение задачи упаковки в контейнеры (bin packing) на экземпляры, в которых размер каждого предмета составляет не менее ε, а количество различных размеров предметов равно K. Тогда существует полиномиальный по времени алгоритм, который оптимально решает эту ограниченную задачу.*

Доказательство. Количество предметов в одном контейнере ограничено величиной $\lfloor 1/\varepsilon \rfloor$. Обозначим это как M. Оценим количество различных возможных типов контейнеров.

Есть K типов предметов, добавим еще один тип – нулевой. В контейнере есть M одинаковых мест для предметов. Количество способов сопоставить M местам по одному из $K + 1$ типов без учета порядка равно $R = \binom{M+K}{M}$. Нулевой тип, который означает, что место остается пустым. При этом все возможные заполнения контейнеров тут учтены. Так что получаем, что различных типов контейнеров не более R.

Общее количество используемых контейнеров не более n. Количество способов сопоставить n контейнерам по одному из $R + 1$ типов без учета порядка равно $P = \binom{n+R}{R}$. Количество типов $R+1$, так как мы опять добавляем нулевой тип, который означает, что контейнера нет. То есть, количество возможных допустимых упаковок не более $P = \binom{n+R}{R}$, что является полиномиальным относительно n. Перечисление их и выбор наилучшей упаковки дает оптимальный ответ. $\square$

Лемма 2.2. *Пусть $\varepsilon > 0$ фиксировано. Рассмотрим ограничение задачи об упаковке в контейнеры на примеры, в которых каждый предмет имеет размер не менее ε. Существует приближенный алгоритм с полиномиальным временем работы, решающий эту ограниченную задачу с коэффициентом $(1 + \varepsilon)$.*

Доказательство. Пусть I – данный набор из n предметов. Упорядочим их по возрастанию весов и и разобьем их на $K = \lceil \frac{1}{\varepsilon^2} \rceil$ групп, в каждой из которых не более $Q = \lfloor n\varepsilon^2 \rfloor$. Заметим, что две группы могут содержать предметы одинакового размера.

Построим набор предметов J, округляя размер каждого предмета в большую сторону до размера наибольшего предмета в его группе. Пример J имеет не более K различных размеров предметов. Следовательно, по лемме 2.1 мы можем найти оптимальную упаковку для J. Очевидно, она также будет допустимой упаковкой для исходных размеров предметов. Покажем, что $OPT(J) \le (1 + \varepsilon)OPT(I)$.

Построим другой набор J', округляя размер каждого предмета из I в меньшую сторону до размера наименьшего предмета в его группе. Очевидно, что $OPT(J') \le OPT(I)$. Докажем, что разбиение для J' дает аналогичное разбиение для всех, кроме Q самых больших предметов набора J.

Пусть предметы отсортированы так: $s_1 \le s_2 \le s_3 \le ... \le s_n$. Пусть f_i – веса предметов в J', а h_i – веса предметов в J $(f_i \le a_i \le h_i)$. Заметим, что предметы с номерами i и $i + Q \le n$ находятся в разных группах, а значит $h_i \le f_{i+Q}$.

Рассмотрим набор предметов с индексами от 1 до $n-Q$. В J они имеют веса $(h_1, h_2, ...h_{n-Q})$ это поэлементно меньше, чем набор весов $(f_{Q+1}, f_{Q+2}, ...f_n)$. Эти веса можно распределить в не более чем $OPT(J')$ корзин, а значит все предметы из набора J, кроме Q наибольших можно разместить в $OPT(J')$ корзинах.

То есть $OPT(J) \le OPT(J') + Q \le OPT(I) + Q$. Все размеры в I не менее ε, значит $OPT(I) \ge n\varepsilon$. Поэтому $Q \le \varepsilon OPT(I)$, а значит $OPT(J) \le OPT(I)(1 + \varepsilon)$. $\square$

3 Основная часть

(a) Доказательство NP-трудности задачи о приближении

Основа доказательства взята из статьи [3].

Теорема 3.1. *Для любого $\varepsilon > 0$ задача о $(\frac{3}{2} - \varepsilon)$-приближении является NP-трудной.*

Доказательство. Построим сведение **NP**-трудной задачи PARTITION к задаче о $(\frac{3}{2} - \varepsilon)$-приближении. Для произвольной задачи PARTITION(A, s) построим входные данные для задачи BIN PACKING. Количество предметов: $n = |A|$. Размер i-го предмета: $s_i = \frac{2s(a_i)}{\sum_{a_j \in A} s(a_j)}$. Таким образом $\sum_{i=i}^{n} s_i = 2$.

Легко видеть, что для задачи BIN PACKING(s) $OPT = 2$ тогда и только тогда, когда PARTITION(A, s) имеет ответ «да». Также очевидно, что сводимость реализуется за полиномиальное время, ведь для нее необходимо только вычисление суммы $\sum_{a_j \in A} s(a_j)$ и значений s_i, и все это вычисляется за полином от длины ввода.

При этом, в случае $OPT = 2$ вопрос задачи о $(\frac{3}{2} - \varepsilon)$-приближении эквивалентен вопросу задачи BIN PACKING. То есть задача о $(\frac{3}{2} - \varepsilon)$-приближении имеет ответ 2 тогда и только тогда когда, когда PARTITION(A, s) имеет ответ «да».

Таким образом показана **NP**-трудность задачи $(\frac{3}{2} - \varepsilon)$-приближении.

$\square$

(b) Аппроксимирующий алгоритм

Теорема 3.2. *Для любого ε, $0 < \varepsilon < \frac{1}{2}$, существует алгоритм A_ε, который работает за время, полиномиальное от n, и находит упаковку, использующую не более $(1+2\varepsilon)OPT+1$ контейнеров.*

Доказательство. Пусть I – исходный набор объектов. Обозначим за I' набор, полученный из I удалением предметов размера меньше ε. По лемме 2.2 мы можем найти разбиение набора I', использующее не более $(1+\varepsilon)OPT(I')$ корзин. Затем будем распределять остальные предметы (размера меньше ε) по корзинам. Очередной предмет кладем в любую корзину, в которую он помещается. Если таких нет, то добавляем новую корзину и кладем туда.

Если новых корзин мы не добавляли, то мы использовали $(1+\varepsilon)OPT(I') \leq (1+\varepsilon)OPT(I)$ корзин. Иначе во всех корзинах, кроме возможно одной, суммарный размер объектов больше $1-\varepsilon$. Значит если всего корзин использовано M, то суммарный размер объектов больше $(M-1)(1-\varepsilon)$. Значит $OPT(I) \geq (M-1)(1-\varepsilon)$, то есть $M \leq \frac{OPT(I)}{1-\varepsilon} + 1 \leq OPT(1+2\varepsilon) + 1$. $\square$

В итоге получаем такой алгоритм для разбиения на не более, чем $(1+2\varepsilon)OPT+1$ корзин:

Algorithm 1 Алгоритм A_ε для упаковки в корзины

1: Удалить предметы размера меньше ε.
2: Округлить размеры предметов, чтобы получить постоянное число различных размеров (Лемма 2.2).
3: Найти оптимальную упаковку для округлённых предметов (Лемма 2.1).
4: Использовать полученную упаковку для исходных размеров предметов.
5: Упаковать предметы размера меньше ε жадным образом.

Важно отметить, что этот алгоритм помимо ответа позволяет получить и явное разбиение на соответсвующее количество корзин.

(c) Тестирование алгоритма

Описанный выше алгоритм имеет очень высокую вычислительную сложность, из-за этого тестирование алгоритма было возможно провести только на данных небольшого размера.

Тестирование алгоритма было разбито на несколько частей. В первой части происходила проверка корректности алгоритма. Для разных значений ε, количества предметов и реального ответа (OPT) генерировалось разбиение OPT корзин размера 1 на несколько предметов так, что суммарный вес предметов был равен OPT, то есть все корзины полностью заполнены. Результат работы алгоритма представляет собой оптимальное количество корзин с необходимой точностью и разбиение предметов на корзины. После выполнения алгоритма проверялось, что найденное количество корзин удовлетворяет требованиям точности, а также проверялась корректность найденного разбиения на корзины. Таким образом было выявлено, что алгоритм работает корректно.

Во второй части проходило исследование зависимости времени работы алгоритма от значения ε и количества предметов. Для этого сначала фиксируется количество предметов значением $size = 10$ и для различных значений ε генерировалось по 20 выборок размера

$size$ с распределением $U(0,1)$, для каждой из них считалось время работы алгоритма, а затем все 20 полученных значений усреднялись. Для исследования зависимости времени работы алгоритма от количества предметов использовался тот же принцип. Для каждого количества предметов генерировалось по 20 выборок соответствующего размера с распределением $U(0,1)$, для каждой из них считалось время работы алгоритма, а затем все 20 полученных значений усреднялись. При этом использовался алгоритм с фиксированным значением $\varepsilon = 0.1$. Полученные зависимости представлены на графиках (Рис. 1). Можно сделать вывод, что при уменьшении ε время работы алгоритма значительно увеличивается. Также при увеличении размера выборки увеличивается время работы алгоритма. Обе зависимости вполне естественны.

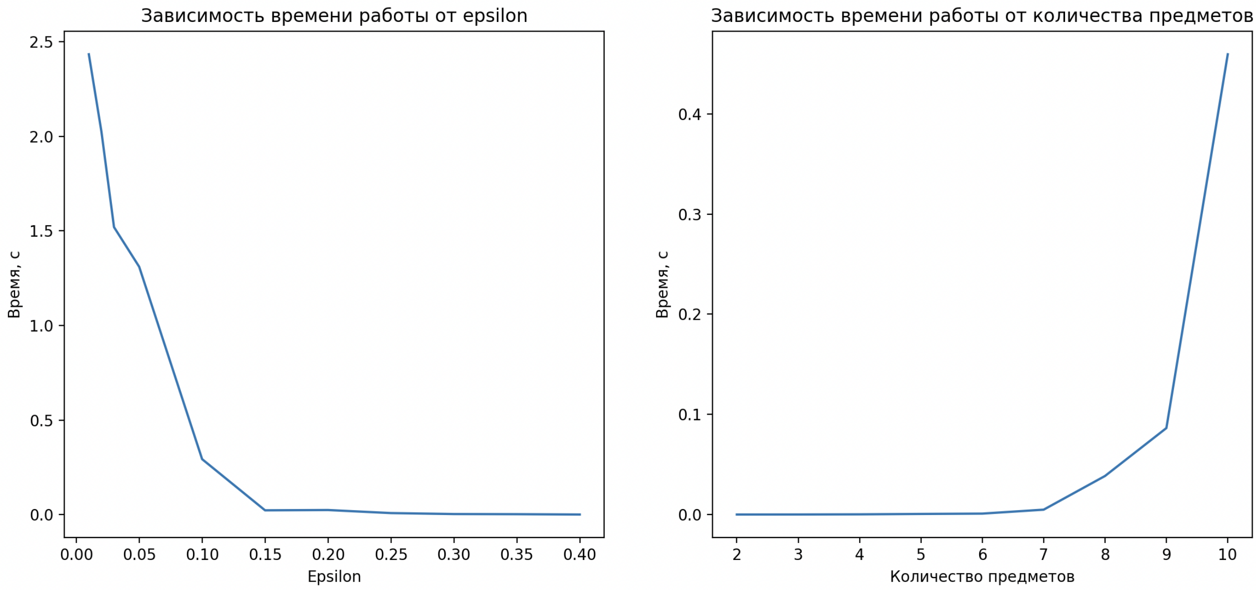

Рис. 1: Зависимости времени от значений ε и количества предметов для выборок с распределением $U(0,1)$

В третьей части проходило такое же исследование, как и во второй, но вместо равномерного распределения рассматривалось распределение $Beta(1,3)$. Для него вероятность получить более маленькие веса выше. Результаты тестирования представлены на графиках (Рис. 2). Полученные зависимости схожи с полученными ранее. Однако для такого распределения время работы алгоритма зачастую получается больше. Поэтому для исследования зависимости времени работы от значений ε фиксировалось количество предметов равное 9.

В целом, исходя из полученных результатов, можно сделать вывод, что при увеличении количества предметов значительно увеличивается время работы алгоритма, что естественно, ведь он работает за $O(n^R)$, где n – количество предметов, а R – большая константа, описанная в Лемме 2.1. Также из определения R становится понятно, почему при фиксированном количестве предметов и уменьшении значения ε, значительно увеличивается время работы алгоритма.

Реализацию самого алгоритма и его тестирования можно найти по ссылке: (*ссылка на проект*).

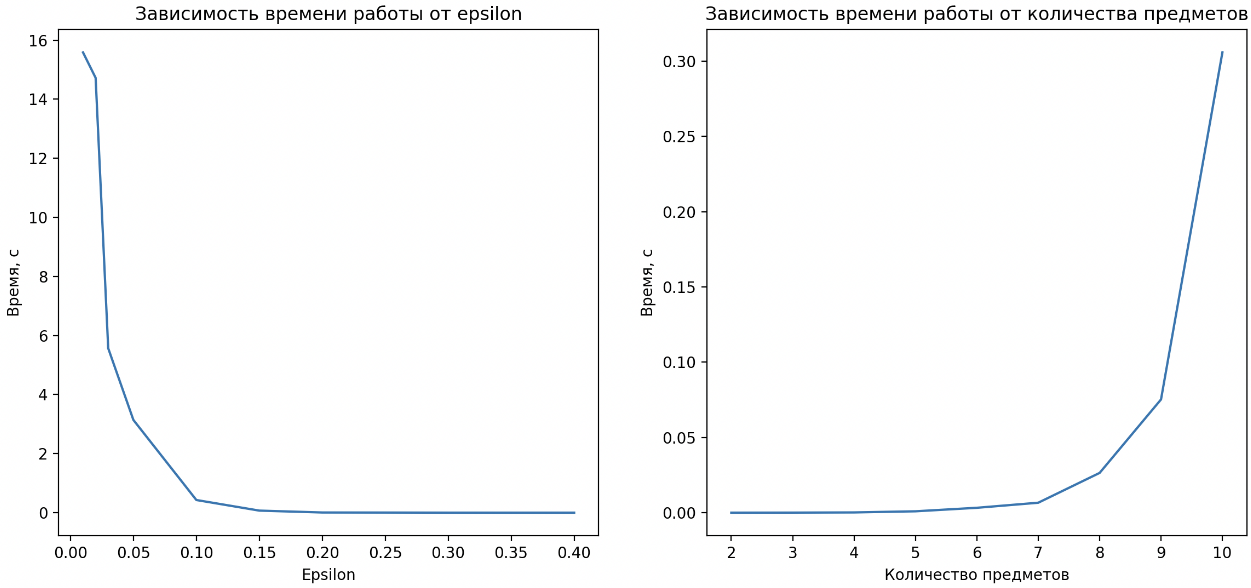

Рис. 2: Зависимости времени от значений ε и количества предметов для выборок с распределением $Beta(1, 3)$

4 Заключение

В данной работе был проведён комплексный анализ классической задачи комбинаторной оптимизации — упаковки по корзинам (Bin Packing Problem). Исследование установило теоретические границы аппроксимируемости задачи, конструктивный алгоритмический подход и практическую реализацию.

Было доказано, что в случае $\mathbf{P} \neq \mathbf{NP}$ задача не допускает полиномиального алгоритма с аппроксимационным коэффициентом лучше, чем $(\frac{3}{2} - \varepsilon)$ для любого $\varepsilon > 0$. Также был представлен и проанализирован алгоритм, который для любого $\frac{1}{2} > \varepsilon > 0$ находит упаковку, использующую не более $(1 + 2\varepsilon)OPT + 1$ корзин.

Однако на практике этот алгоритм работает очень долго даже для достаточно небольших размеров исходных данных. Поэтому для применения на практике, необходимы модификации в реализации.

5 Список литературы

[1] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[2] Michael R Garey, Ronald L Graham, and Jeffrey D Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing*, pages 143–150, 1972.

[3] M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

[4] David S Johnson, Alan Demers, Jeffrey D Ullman, Michael R Garey, and Ronald L Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974.

[5] W Fernandez de la Vega and George S Lueker. Bin packing can be solved within 1+ in linear time. *Combinatorica*, 1(4):349–355, 1981.

[6] V. V. Vazirani. Bin packing. In *Approximation Algorithms*, chapter 9. Springer, 2001.