ВЕРОЯТНОСТНЫЙ ПОИСК С ЗАПРЕТАМИ ДЛЯ ЗАДАЧИ УПАКОВКИ В КОНТЕЙНЕРЫ

Ю. А. Кочетов, А. Р. Усманова

Аннотация

В работе предлагается новый вероятностный алгоритм поиска с запретами для задачи упаковки в контейнеры. Алгоритм использует рандомизированную окрестность квадратичной мощности и простое правило диверсификации для смены района поиска. Начальное решение строится с помощью процедуры локального спуска по окрестности экспоненциальной мощности. Один шаг такой процедуры имеет кубическую трудоемкость. Разработанный алгоритм тестировался на задачах из электронной библиотеки OR-Library. Результаты численных экспериментов свидетельствуют о высокой работоспособности алгоритма.

Введение

Задача упаковки в контейнеры состоит в следующем [2]. Даны n предметов и m контейнеров. Все контейнеры имеют одинаковую емкость $c \geq 0$, а каждый предмет i имеет вес $w_i \geq 0$. Требуется узнать, можно ли разместить все предметы по контейнерам так, чтобы суммарный вес в каждом контейнере не превышал его емкости. Сформулированная задача является NP-полной в сильном смысле и тесно связана со следующей задачей теории расписаний [5]. Даны n работ и m машин. Для каждой работы i известна длительность ее выполнения $w_i \geq 0$. Требуется найти такое разбиение множества работ по машинам, чтобы минимизировать момент завершения самой поздней работы. Легко заметить, что алгоритм решения этой оптимизационной задачи дает ответ на вопрос задачи упаковки в контейнеры. В дальнейшем центральную роль будет играть именно эта оптимизационная постановка задачи, а не другая широко известная постановка, когда требуется минимизировать число контейнеров для упаковки заданного набора предметов [2]. Мотивация такого выбора станет ясна позже.

Пусть $I = \{1, \dots, n\}$ и подмножества $I_1, \dots, I_m$ задают разбиение множества I на непустые подмножества. В качестве целевой функции выберем величину

$$F(I_1, \dots, I_m) = \sum_{k=1}^{m} \max(0, \sum_{i \in I_k} w_i - c),$$

определяющую суммарное превышение емкости контейнеров. Если $F(I_1, \dots, I_m) = 0$ для некоторого разбиения $I_1, \dots, I_m$, то ответ в исходной задаче положительный. Если же для всех разбиений $F(I_1, \dots, I_m) > 0$, то ответ отрицательный. Отметим, что аналогичными свойствами обладает и функция

$$G(I_1, \dots, I_m) = \max_{k \leq m} \sum_{i \in I_k} w_i,$$

задающая момент завершения самой поздней работы.

Для минимизации функции $F(I_1, \dots, I_m)$ предлагается вероятностный алгоритм поиска с запретами, который останавливается при достижении нуля и может работать сколь угодно долго, если соответствующего разбиения не существует. Разработанный алгоритм использует полиномиальные и экспоненциальные по мощности окрестности. Исследование их свойств и поведения алгоритма является основной целью данной работы.

1 Окрестности

Рассмотрим три окрестности, две из которых имеют полиномиальную мощность, а третья — экспоненциальную. Пусть $I_1, \dots, I_m$ — произвольное разбиение. Окрестностью $N_1(I_1, \dots, I_m)$ назовем множество всех разбиений, которые получаются из данного путем перемещения элемента из одного подмножества в другое. Если предположить, что в каждом подмножестве содержится в среднем n/m элементов, и каждый элемент может быть перемещен в любое из $m-1$ подмножеств, то мощность такой окрестности можно оценить в среднем величиной $O(n)$.

Окрестностью $N_2(I_1, \dots, I_m)$ назовем множество всех разбиений, которые получаются из данного путем фиксирования двух подмножеств, скажем I_l и I_k, выбора в каждом подмножестве по одному элементу $i_l \in I_l,\ i_k \in I_k$ и замене их одного на другой. Мощность такой окрестности оценивается в среднем величиной $O(n^2)$.

Введенные окрестности обладают следующим свойством, важным для локального поиска. Сопоставим каждому разбиению $I_1, \dots, I_m$ вершину некоторого графа и будем считать две вершины смежными, если одно разбиение принадлежит объединению введенных выше окрестностей другого разбиения. Такой граф иногда называют графом окрестностей или ландшафтом [10]. Из определений прямо следует, что этот граф связен. Рассмотрим теперь аналогичный граф для оптимизационной постановки, в которой требуется минимизировать число подмножеств. Уточним для этого случая основные определения. Будем рассматривать только такие разбиения, для которых вес каждого подмножества не превышает установленной емкости. Для каждого разбиения соседними являются все разбиения, получающиеся перемещением одного (N_1) или двух (N_2) элементов из разных подмножеств. Если в ходе таких преобразований появляется пустое подмножество, то оно удаляется из рассмотрения и величина m уменьшается на единицу. Целевая функция задачи — минимизация числа подмножеств. Покажем, что при таких естественных определениях граф окрестностей может не быть связным.

Пусть $c = 1, n = 90$ и

$$w_i = \begin{cases} 1/2, & i = 1, \dots, 30 \\ 1/3, & i = 31, \dots, 60 \\ 1/10, & i = 61, \dots, 90. \end{cases}$$

Положим $I_k = \{1/2, 1/3, 1/10\}, k = 1, \dots, 30$. Так как вес каждого подмножества не должен превосходить единицы, то данное разбиение имеет пустую окрестность $N_1(I_1, \dots, I_m)$, а все разбиения в $N_2(I_1, \dots, I_m)$ имеют тот же вид что и данное. Таким образом, в графе окрестностей нет пути от указанного разбиения до оптимального, содержащего 28 подмножеств:

$$I_k = \begin{cases} \{1/2, 1/2\} & k = 1, \dots, 15 \\ \{1/3, 1/3, 1/3\} & k = 16, \dots, 25 \\ \{1/10, \dots, 1/10\} & k = 26, \dots, 28. \end{cases}$$

Заметим, что требуемый путь появляется, если расширить первую окрестность, позволяя добавлять пустое подмножество и перемещать в него произвольный элемент.

Определим теперь третью окрестность. Выберем произвольным образом в каждом подмножестве I_k некоторый элемент i_k. Окрестностью $N_3(I_1, \dots, I_m, i_1, \dots, i_m)$ назовем множество всех разбиений, которые получаются всевозможными распределениями элементов $i_1, \dots, i_m$ по подмножествам $I_1, \dots, I_m$ так, чтобы в каждое подмножество попало ровно по одному элементу. Мощность такой окрестности равна $m!$. Положим

$$d_{kl} = \max(0, \sum_{i \in I_k} w_i - w_{i_k} + w_{i_l} - c),\ k, l \leq m.$$

Величина d_{kl} определяет превышение веса k-го подмножества, если из него удалить элемент i_k и добавить элемент i_l. Оптимальное разбиение в такой окрестности относительно функции F можно найти с трудоемкостью $O(m^3)$.

Введем булевы переменные

$$x_{kl} = \begin{cases} 1, & \text{если элемент } i_l \text{ добавляется в подмножество } I_k \\ 0 & \text{в противном случае.} \end{cases}$$

Задача поиска оптимального разбиения может быть представлена в виде задачи о назначениях:

$$\min \sum_{k=1}^{m} \sum_{l=1}^{m} d_{kl} x_{kl}$$

$$\sum_{k=1}^{m} x_{kl} = 1,\ l = 1, \dots, m$$

$$\sum_{l=1}^{m} x_{kl} = 1,\ k = 1, \dots, m$$

$$x_{kl} \in \{0, 1\},\ k, l = 1, \dots, m,$$

которая легко решается точно [3]. Аналогично можно найти разбиение, оптимальное относительно функции G [1].

Недостатком данной окрестности является то, что она не дает возможности менять мощности подмножеств $I_k,\ k \leq m$. Однако этот недостаток легко исправить, если в каждое подмножество добавить фиктивный элемент нулевого веса. Этот же прием делает первую окрестность частью второй.

2 Вероятностный поиск с запретами

Идея алгоритма поиска с запретами состоит в организации локального поиска таким образом, чтобы не останавливаться в точке локального оптимума, а "путешествовать"от одного локального оптимума к другому в надежде найти глобальный оптимум [6]. Механизм реализации этой идеи состоит из двух частей. Первая часть — список запретов. Он строится по предистории поиска и запрещает часть точек в окрестности текущего решения. Его цель исключить повторение уже пройденных точек. Вторая часть — рандомизация окрестности. Этот прием вносит элемент случайности, сокращает трудоемкость одного шага и заметно повышает эффективность алгоритма особенно на задачах большой размерности.

Алгоритм начинает свою работу с построения начального разбиения. В качестве такового может быть взято произвольное разбиение множества I на m непересекающихся подмножеств или разбиение, полученное одним из жадных алгоритмов. Такой алгоритм легко построить, например, адаптацией широко известного алгоритма FFD (перый подходящий с упорядочением)[2]: сортируем элементы по невозрастанию весов и укладываем их по очереди, выбирая каждый раз подмножество с минимальным весов. Полученное решение, как правило, не является допустимым и к нему применяется процедура локального спуска по экспоненциальной окрестности. Выберем в каждом подмножестве случайным образом один элемент и решим задачу о назначениях. Будем повторять эту процедуру до тех пор, пока несколько раз подряд не получим одно и тоже значение целевой функции. В этом случае будем считать начальное разбиение $I_1^0, \ldots, I_m^0$ построенным. Далее применяется вероятностный поиск с запретами с полиномиальными окрестностями.

Подмножество $I' \subset I$ назовем тяжелым, если его вес превышает границу c. Определим рандомизированную окрестность

$$N'(I_1, \ldots, I_m) \subseteq N_1(I_1, \ldots, I_m) \cup N_2(I_1, \ldots, I_m),$$

установив разные вероятности для тяжелых и всех остальных подмножеств. Рассмотрим два параметра p_1, p_2. Параметр p_1 будет задавать вероятность включения разбиениям $I'_1, \ldots, I'_m$ в окрестность $N'(I_1, \ldots, I_m)$, если это разбиение было получено из $I_1, \ldots, I_m$ изменением одного из тяжелых подмножеств. В противном случае соответствующую вероятность будем задавать параметром p_2. Все разбиения включаются в окрестность $N'(I_1, \ldots, I_m)$ независимо друг от друга с вероятностью p_1 или p_2.

Шаг алгоритма состоит в переходе от текущего решения к следующему, наилучшему в окрестности $N'(I_1, \ldots, I_m)$ относительно некоторой вспомогательной функции. Выбор такой функции зависит от специфики задачи. Иногда в качестве таковой можно взять исходную целевую функцию. Понятно, что функция $G(I_1, \ldots, I_m)$ для этих целей не подходит. Имея два максимальных по весу подмножества, получим в окрестности почти все решения с одинаковым значением. Аналогичный эффект наблюдается и в случае, когда требуется минимизировать число подмножеств [11].

Пусть элемент i_k заменяется на i_l. В качестве вспомогательной целевой функции возьмем разность

$$f(i_k, i_l) = |\sum_{i \in I_k} w_i - \sum_{i \in I_l} w_i| - |\sum_{i \in I_k} w_i - \sum_{i \in I_l} w_i + 2w_{i_l} - 2w_{i_k}|.$$

Величина $f_(i_k, i_l)$ задает изменение разности весов подмножеств при замене i_k на i_l. При выборе очередного решения в рандомизированной окрестности будем искать разбиение с максимальным значением вспомогательной функции. Трудоемкость такой процедуры совпадает по порядку величины с мощностью окрестности $N_2(I_1, \ldots, I_m)$. В [9] рассматривалась близкая по смыслу вспомогательная функция.

Обозначим через $Tabu(s)$ множество запретов, полученных за последние s шагов алгоритма. Это множество состоит из пар (i_k, I_l), которые запрещают возвращать элемент i_k в подмножество I_l. В окрестности любого разбиения $I_1, \ldots, I_m$ список запретов порождает множество запрещенных разбиений $N_{tabu}(I_1, \ldots, I_m)$, которые удаляются из рандомизированной окрестности. Общая схема алгоритма поиска с запретами может быть представлена следующим образом.

Вероятностный поиск с запретами

1. Построить начальное разбиение $I_1^0, \ldots, I_m^0$ и положить $F^* = F(I_1^0, \ldots, I_m^0), Tabu(s) := \emptyset, t := 0$.
2. Пока не выполнен критерий остановки делать следующее.
   2.1. Сформировать окрестность $N'(I_1^t, \ldots, I_m^t) \setminus N_{tabu}(I_1^t, \ldots, I_m^t)$.
   2.2. Если она пуста, то положить $I_k^{t+1} := I_k^t,\ k \leq m$,
   иначе найти пару $i_{k'}, i_{l'}$, на которой достигается $\max f(i_k, i_l)$.
   2.3. Построить разбиение $I_1^{t+1}, \ldots, I_m^{t+1}$ по паре $i_{k'}, i_{l'}$.
   2.4. Если $F^* > F(I_1^{t+1}, \ldots, I_m^{t+1})$, то $F^* := F(I_1^{t+1}, \ldots, I_m^{t+1})$.
   2.5. Положить $t := t + 1$ и обновить список запретов $Tabu(s)$.

Напомним, что в качестве критерия остановки выступает равенство нулю целевой функции. Длина списка запретов $s \geq 0$ и коэффициенты рандомизации p_1, p_2 являются управляющими параметрами алгоритма. Выбор их значений зависит от размерности задачи и мощности окрестности.

Важным элементом локального поиска является стратегия диверсификации. Согласно этой стратегии [6] если рекорд F^* длительное время не меняется, то целесообразно перейти к другой части допустимой области, кардинально перестроив текущее решение. Заметим, что вспомогательная функция $f(i_k, i_l)$ выбрана так, чтобы максимально выравнивать веса подмножеств. Это приводит к тому, что в тяжелых подмножествах собираются элементы с малыми весами, которые уже редко перемещаются в другие подмножества. В связи с этим предлагается следующая схема диверсификации. Выбирается несколько тяжелых подмножеств, которые долго не менялись в ходе локального поиска. Из каждого такого подмножества выбираются два элемента и меняются на один элемент из "легкого"подмножества. Такая процедура сильно увеличивает суммарное переполнение, поэтому в завершение процедуры применяется локальный спуск по экспоненциальной окрестности.

3 Численные эксперименты

Разработанный алгоритм запрограммирован на языке ПАСКАЛЬ (Delphi 4.5) и тестировался на задачах из электронной библиотеки OR-Library [4]. Эти задачи разбиты на два класса [8], назовем их **A** и **B**. В каждом классе 4 группы, в каждой группе по 20 задач одной размерности:

Класс **A**: $n = 120, 250, 500, 1000$.
Класс **B**: $n = 60, 120, 249, 501$ (триплеты).

Первый вычислительный эксперимент касается экспоненциальной окрестности. На рисунке 1 показано типичное поведение функции F при выборе начального решения. На первых шагах наблюдается быстрое падение. На последних шагах значение функции меняется незначительно. Многие элементы возвращаются в те подмножества, из которых они были взяты. Процесс по сути останавливается. Следуя идеологии поиска с запретами, естественно запретить возвращение элементов или их части в исходные подмножества. Этой цели можно достичь, положив $d_{kk} = \infty,\ k \leq m$. Аналогичным способом можно реализовать любые запреты, что заметно повышает эффективность алгоритма. На рисунке 2 приведены результаты экспериментального сравнения двух вариантов алгоритма: без запретов и с запретом возвращать элементы в тяжелые подмножества. Сравнение этих двух вариантов свидетельствует о больших потенциальных возможностях такого подхода.

Поиск по экспоненциальной окрестности на начальном шаге является важным элементом алгоритма. Его исключение приводит к резкому росту трудоемкости, особенно на сложных задачах. В таблице 1 приведены время и число итераций алгоритма с использованием экспоненциальной окрестности и без нее. Результаты эксперимента приведены для класса **A**, $n = 1000$. Каждая из задач решалась 10 раз. Время счета здесь и далее приведено для PC Pentium-750. Аналогичные результаты были получены и на всех других размерностях обоих классов.

Третий вычислительный эксперимент касался выбора параметров p_1, p_2. В [9] рассматривался вариант $p_1 = 1, p_2 = 0$. Численные эксперименты подтвердили правильность выбора $p_2 = 0$. Даже незначительное увеличение p_2 приводит к росту общего числа итераций. Выбор $p_1 = 1$ оправдан только для задач малой размерности. При $n = 1000$ наилучшее значение p_1 не превосходит 0,5. В таблице 2 приведены результаты экспериментов для $p_1 = 1$ и $p_1 = 0,2$. В таблицах 3-10 приводятся результаты расчетов по всем тестовым задачам при наилучших значениях параметров.

Список литературы

[1] Гордон А. Я. Один алгоритм решения минимаксной задачи о назначении. В кн. Исследования по дискретной оптимизации, М.: Наука, 1976, с. 327-332.

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

[3] Диниц Е. А., Кронрод М. А. Один алгоритм решения задачи о назначении. Доклады АН СССР, 1969, т. 189, № 1.

[4] Beasley J. E. OR-Library: Distributing test problems by electronic mail. Jounal of the Operational Research Society, V. 41, No. 11, 1990, 1069-1072 (http://mscmga.ms.ic.ac.uk/info.html).

[5] Brucker P., Huring J., Werner F. Improving local search heuristics for some scheduling problems. Part II. Discrete Applied Mathematics, V. 72, 1997, 47-69.

[6] F. Glover and M. Laguna, *Tabu Search.* Kluwer Academic Publishers, Dordrecht, 1997.

[7] Aarts E. H. L., Lenstra J. K. (eds.) *Local Search in Combinatorial Optimization.* Wiley, Chichester, 1997.

[8] Folkenauer E. A hybrid grouping genetic algorithm for the bin packing. Journal of Heuristics, V.2, No. 1, 1996, 5-30.

[9] Hübscher R., Glover F. Applying tabu search with influential diversification to multiprocessor scheduling. Computers and Operations Research, V. 21, No. 8, 1994, 877-884.

[10] Reeves C. R. Landscapes, operators and heuristic search. Annals of Operations Research 86, 1999, 473-490.

[11] Scholl A., Klein R., Jürgens C. BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Computers and Operations Research, V. 24, No. 7, 1997, 627-645.