Контейнерно-ориентированное представление плана упаковки в задаче BPP

Т.Д. Тарасова
Факультет информатики и робототехники
Уфимский государственный авиационный технический университет
Уфа, Россия
e-mail: td_tarasova@mail.ru

А.Р. Усманова
Факультет информатики и робототехники
Уфимский государственный авиационный технический университет
Уфа, Россия
e-mail: адрес электронной почты

Аннотация

В работе рассматривается известная задача распределения одномерного ресурса - BPP (Bin Packing Problem). Авторы предлагают контейнерно-ориентированное представление решения данной задачи и исследование эффективности такого представления.

1. Введение

Задача BPP представляет собой известную задачу распределения одномерного ресурса, являющуюся NP-трудной [1]. Приведем содержательную постановку задачи линейной упаковки: дано множество $L=\{w_1,w_2,\ldots,w_n\}$ неотрицательных весов предметов и положительное число C - вместимость контейнера. Без потери общности веса предметов и вместимость контейнера будем считать целыми числами. Требуется найти разбиение множества L на минимально возможное число непересекающихся подмножеств, таких, что сумма весов в каждом подмножестве не превышает C. Если множество L интерпретировать как длины некоторых предметов, а не веса, и C как длину объекта, который будет подвергаться разрезанию на более мелкие заготовки, то можно говорить о задаче линейного раскроя. Математические постановки задач раскроя и упаковки либо идентичны, либо различаются незначительно, например, в задачах массового раскроя число предметов с различными длинами обычно невелико, зато помимо длины заготовки характеризуются еще требуемым количеством. В задаче же упаковки веса предметов в общем случае уникальны. Также в задачах раскроя решение обычно представляет собой матрицу неотрицательных чисел, а в задачах упаковки план упаковки есть бинарная матрица, содержащая только нули и единицы.

Труды Седьмой Всероссийской научной конференции «Информационные технологии интеллектуальной поддержки принятия решений», 28-30 мая, Уфа-Ставрополь-Ханты-Мансийск, Россия, 2019

В данной работе будет главным образом использована терминология задачи упаковки в контейнеры.

Приведем одну из математических формулировок рассматриваемой задачи.

Задача 1.
Задан вектор $L=\{w_1,w_2,\ldots,w_n\}$ весов предметов и n контейнеров емкости C. Сопоставим каждый предмет одному контейнеру таким образом, чтобы вес предметов в каждом контейнере не превышал число C, и число использованных контейнеров было минимальным. Полагаем, что контейнер, которому сопоставлен хотя бы один предмет, используется, в противном случае он не используется.

Пусть $y_j \in \{0,1\},\ j \in N = \{1,2,\ldots, n\}$

где $y_j = \begin{cases} 1, \text{если j - й контейнер используется} \\ 0, \text{иначе} \end{cases}$

$$x_{ij} \in \{0,1\} \quad \text{i, j} \in \text{N ,где}$$

$x_{ij} = \begin{cases} 1, \text{если i - й предмет сопоставлен j - му контейнеру;} \\ 0, \text{иначе} \end{cases}$

Минимизировать

$z = \sum_{j=1}^{n} y_j$ по отношению к

$\sum_{i=1}^{n} y_j x_{ij} w_i \le C,\ j \in N$ - ограничение, показывающее что суммарный все предметов, сопоставленных каждому контейнеру, не должен превышать его ёмкость.

$\sum_{i=1}^{n} x_{ij} = 1,\ j \in N$ - ограничение, требующее упаковки каждого из предметов.

Существует простая оценка снизу для числа контейнеров – нижняя граница

$$N_0 = \left\lceil \sum_{i=1}^{n} w_i / C \right\rceil \quad (*)$$

Можно утверждать, что если получено решение со значением целевой функции, равной N_0, то оно заведомо оптимально.

Особенности предлагаемого в данной работе алгоритма позволяют рассматривать родственную задачу расписания параллельных процессоров. Пусть имеется m идентичных процессоров и n работ с временами выполнения $w_1, w_2, .., w_n$. Требуется сопоставить каждую работу какому-либо процессору, так чтобы минимизировать максимальное время работы процессоров.

Задача 2. При заданных исходных данных найти x_{ij}

$x_{ij} \in \{0,1\}, i \in N=\{1,2,..,n\}, \; j \in M = \{1,2,..,m\}$,

где

$$x_{ij} = \begin{cases} 1, \text{если i - я работа сопоставлена j - му процессору;} \\ 0, \text{иначе} \end{cases}$$

$$\sum_{j=1}^{m} x_{ij} = 1, i \in N$$

при которых величина

$$T = \max_j (\sum_{i=1}^{n} x_{ij} w_i)$$

достигает минимума.

Очевидно, если получено решение задачи 2, не обязательно оптимальное, и $T \le C$, то оно является оптимальным решением задачи 1 со значением целевой функции $z = m$. В некоторых работах задача 1 именуется задачей упаковки контейнеров –1 (BPP-1), а задача 2 – задачей упаковки контейнеров-2 (BPP-2) [8]. На сегодняшний день известно много алгоритмов решения данной задачи, как точных, так и приближенных. Вследствие NP-полноты задачи наибольший интерес представляют эффективные приближенные алгоритмы, как простейшие эвристики, так и метаэвристики с использованием генетических и эволюционных алгоритмов, метода поиска с запретами, моделирования отжига и другие. Большинство таких алгоритмов представляют собой итерационный процесс, постепенно улучшающий некоторое начальное решение. На каждом шаге по текущему решению строится некоторое множество решений – окрестность соседних решений, затем из них выбирается по определенному критерию наилучшее. Важными факторами, влияющими на эффективность алгоритмов, является вычислительная сложность построения окрестности решений, и сложность оценки решений в окрестностях. В данной работе будет предложено контейнерно-ориентированное представление плана упаковки и показано преимущество такого представления по отношению к упомянутым факторам.

В работе проводились численные эксперименты на 8 тестовых наборах задач из электронной OR-библиотеки, предложенные Фолкнером [5], в каждом наборе имеется 20 случайным образом сгенерированных задач, объединенных общими характеристиками исходных данных. В таблице 1 приведены эти характеристики для каждого набора.

Табл. 1. Характеристики тестовых задач

№ тестового набора	n – число предметов	C – емкость контейнера	Интервал, содержащий веса предметов
1	120	150	[20,100]
2	250	150	[20,100]
3	500	150	[20,100]
4	1000	150	[20,100]
5	60	100.0	[20.0,50.0]
6	120	100.0	[20.0,50.0]
7	249	100.0	[20.0,50.0]
8	501	100.0	[20.0,50.0]

Данные тестовые задачи являются достаточно сложными и решения, получаемые простыми эвристическими алгоритмами в большинстве случаев далеки от оптимального. Для всех задач известны решения, полученные Фолкнером с помощью генетического алгоритма. Почти во всех случаях эти решения являются заведомо оптимальными, так как совпадают с нижней границей.

2. Общий алгоритм упаковки

Опишем сначала общий алгоритм упаковки, который в дальнейшем может быть модифицирован для многих метаэвристических методов - локального спуска, генетических и эволюционных алгоритмов. В частности, авторы использовали данную схему при реализации метода поиска с запретами [3,6,7]. Одной из характерных особенностей предлагаемого алгоритма является то, что задача упаковки решается в форме Задача 2. Другими словами, алгоритм начинает работу с некоторым начальным недопустимым (для Задачи 1) решением, в котором предметы сопоставленные нескольким контейнерам превышают его емкость. Цель алгоритма – преобразовать решение таким образом, чтобы оно стало допустимым в исходной Задаче 1. Выбор такой модели обусловлен, прежде всего, способом построения окрестности решений, описанным далее. На первом этапе получаем начальное решение для задачи BPP-1 (Задача 1) любым достаточно несложным эвристическим методом. Это может быть и классический FFD (First Fit Decreasing) или любые другие простые эвристики. Авторы в данном случае использовали вероятностную простую эвристику, описанную в [4] - «случайная перестановка с

параметром»-RPP. На следующем этапе все предметы из наименее заполненного контейнера перемещаются в другие контейнеры, и значение целевой функции уменьшается на единицу. Алгоритм перемещения довольно простой и состоит в следующем: самый тяжелый предмет из «уничтожаемого» контейнера перемещается во второй из наименее заполненных контейнеров, второй по возрастанию веса предмет перемещается в третий из наименее заполненных контейнеров и так далее, до тех пор, пока все предметы из данного контейнера не будут перемещены. Работа алгоритма демонстрируется на рис.1, где контейнеры расположены по возрастанию заполненности (суммарного веса сопоставленных им предметов) снизу вверх, а предметы в «уничтожаемом» контейнере расположены по возрастанию веса слева направо.

Сложность алгоритма можно считать константой, так как число предметов в контейнере не зависит от общего числа предметов, а только от отношения среднего веса предмета к емкости контейнера. В решаемых тестовых задачах это число не превосходит 5 для наборов задач 1-4 и 4 для наборов 5-8, а так как уничтожается самый легкий контейнер, то обычно число предметов в нем еще меньше этой границы.

Очевидно, в результате работы алгоритма «уничтожения» контейнера решение становится недопустимым, некоторые контейнеры переполнены. Поэтому на следующем этапе метод поиска с запретами пытается преобразовать полученное недопустимое решение в допустимое, где отсутствуют переполненные контейнеры (Рис. 1).

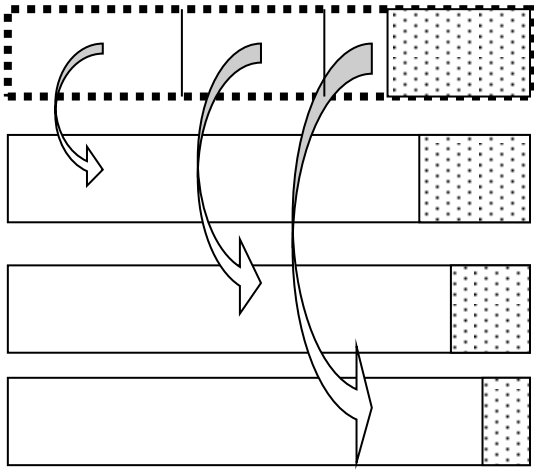

Рис. 1 Алгоритм «уничтожения» контейнера

Если это удается, то вновь уничтожается самый легкий контейнер и повторяется поиск допустимого решения методом TS. Процесс заканчивается, если на очередном шаге получено допустимое решение со значением целевой функции, совпадающим с нижней границей, или если не удалось получить допустимое решение. В последнем случае результатом будет предыдущее допустимое решение, полученное до уничтожения контейнера.

Приведем формальное описание общего алгоритма.

Алгоритм Упаковка

1. Положить номер шага i=0. Получить начальное решение x_0 простым эвристическим методом со значением целевой функции Nэвр. Положить текущее значение числа контейнеров N_i= Nэвр. Положить наилучшее значение целевой функции $N^*= N_i$.
2. Вызов алгоритма «уничтожения» контейнера. $N_i=N_i-1$.
3. Вызов основного алгоритма, позволяющего получить допустимое решение.
4. Положить i=i+1. Если удалось получить допустимое решение, то $N^*= N_i$, шаг 5. Иначе шаг 6.
5. Если $N_i=N_0$, где нижняя граница N_0 вычисляется по формуле (*), то получено оптимальное решение, остановка работы. Иначе шаг 2.
6. Остановка алгоритма с выдачей результата N*.

Основным элементом алгоритма является шаг 3, на котором непосредственно получается (или не получается) допустимое решения для заданного числа контейнеров. В данной работе мы не будем описывать полностью какой- либо алгоритм на этом шаге. Нашей целью является демонстрация преимущество контейнерно-ориентированного представления решения при

- реализации алгоритма уничтожения контейнера
- построении окрестности соседних решений
- вычислении оценочной функции для каждого решения в окрестности.

3. Контейнерно-ориентированное представление плана упаковки

Опишем предлагаемое представление решения задачи. Алгоритмически план упаковки представляет собой тип «запись» со следующими полями: число контейнеров и список контейнеров. Элементы списка контейнеров также являются записями и содержат число предметов в контейнере, общий вес предметов и список номеров предметов, сопоставленных данному контейнеру. План не является допустимым и содержит переполненные контейнеры. В поле списка предметов указаны номера предметов, а в скобках соответствующие им веса. Веса предметов приведены здесь только для удобства прочтения, в действительности они хранятся в отдельном списке. Список контейнеров отсортирован по возрастанию веса контейнера, а список предметов в нем по возрастанию веса предметов (или, что то же самое, по

убыванию номеров предметов). Причем при изменении плана на каждой итерации не проводится сортировка, просто контейнер (или номер предмета) вставляется в существующий список на нужное место.

Организованный подобным образом план упаковки представляется более удобным, чем, например, приоритетный список предметов. Последний часто используется и представляет собой список номеров предметов, в котором они должны упаковываться по правилу FFD. Ранние эксперименты показали, что при использовании приоритетного списка тратится много времени на выяснение вопросов:

1) какие именно предметы упакованы в контейнер с данным номером
2) чему равен суммарный вес предметов в определенном контейнере
3) каков номер самого легкого или самого тяжелого предмета в контейнере и т.п.

После каждого изменения плана упаковки эти данные изменялись, и их нахождение требовало, как минимум, одного просмотра списка для каждого контейнера. В то время как в предлагаемом способе представления данных все характеристики, относящиеся к контейнеру, хранятся постоянно и изменяются сразу при перемещении предметов. Обращение к ним занимает только константное время, необходимое для обращения к оперативной памяти. Конечно, контейнерно-ориентированное представление упаковки занимает больше места, чем приоритетный список, но при современных ресурсах оперативной памяти это неудобство пренебрежимо мало по сравнению с выигрышем во времени.

4. Операторы преобразования решения

Ранее были упомянуты окрестности соседних решений. Покажем в данном разделе способ построения двух видов окрестностей. Первая окрестность N1 состоит из решений, получаемых из исходного перемещением одного предмета из одного контейнера в другой. Вторая окрестность N2 включает в себя решения, в которых два предмета из двух разных контейнеров меняются местами.

Для генерации решений из первой окрестности применяется оператор *op1(i_1,j_1,i_2)*, который перемещает j_1-й предмет из контейнера с номером i_1 в контейнер с номером i_2. При этом суммарный вес контейнеров пересчитывается следующим образом:

$$S'_{i_1} = S_{i_1} - w_{j_1}$$
$$S'_{i_2} = S_{i_2} + w_{j_2},$$

где S_{i_1}, S_{i_2} - веса контейнеров в исходном решении, S'_{i_2}, S'_{i_2} - веса контейнеров в соседнем решении, получающемся при применении *op1(i_1,j_1,i_2)* к исходному решению. Очевидно, эти вычисления есть фиксированное количество элементарных операций, не зависящее от размерности задачи. Удаление номера предмета из списка предметов, сопоставленных контейнеру, требует также фиксированное число операций присваивания, максимально равное числу предметов в контейнере, оно не зависит от размерности задачи, а только от отношения среднего веса предмета к емкости контейнера. Вставка номера предмета требует r операций сравнения и (k-r+1) операций присваивания, где k-число предметов в контейнере, а $r \in [1,k]$, и также число этих операций не зависит от размерности задачи.

Второй оператор *op2(i_1,j_1,i_2,j_2)* меняет местами j_1-й предмет из i_1-го контейнера с j_2-м предметом из i_2-го контейнера. Веса предметов пересчитываются по формулам

$$S'_{i_1} = S_{i_1} - w_{j_1} + w_{j_2}$$
$$S'_{i_2} = S_{i_2} - w_{j_2} + w_{j_1},$$

(3.4)

где S_{i_1}, S_{i_2} - веса контейнеров в исходном решении, S'_{i_2}, S'_{i_2} - веса контейнеров в соседнем решении, получающемся при применении *op2(i_1,j_1,i_2,j_2)* к исходному решению. Очевидно, число операций для изменения плана упаковки не зависят от размерности задачи, из соображений, изложенных выше для первой окрестности соседних решений. Можно показать, что число решений в каждой из окрестностей ограничено сверху величиной n^2. Основным способом сокращения вычислительного времени в предлагаемом методе является использование рандомизированных окрестностей. Авторы предлагают следующий вариант построения такой окрестности. - вероятность каждого соседнего решения быть включенным в рандомизированную окрестность зависит от веса контейнеров, из которых перемещаются предметы. Контейнеры разбиты на 3 группы:

1) $S1=\{ i : S_i = \max\limits_{k=1,m} S_k \}$

   – множество контейнеров с весом, равным максимальному весу контейнера

2) $S2=\{ i : C < S_i < \max\limits_{k=1,m} S_k \}$

   – множество контейнеров с весом, превышающим емкость C, но меньшим, чем максимальный вес контейнера

3) $S3=\{ i : C \geq S_i \}$

   – множество контейнеров с весом, меньшим и равным емкости C,

где S_i- суммарный вес заготовок, сопоставленных i-му контейнеру, m-число контейнеров.

Для получения допустимого решения, очевидно, в первую очередь, необходимо выбирать решения, изменяющие контейнеры из первой и второй групп, так как они не могут войти в допустимое решение. Решения, изменяющие только контейнеры из третьей группы, должны иметь самую низкую вероятность включения в рандомизированную окрестность. Под изменением контейнеров здесь понимается перемещение (окрестность N1) или обмен (окрестность N2) предметов из этих контейнеров, то есть изменение общего веса предметов, сопоставленных контейнеру.

В большинстве работ предлагается рассматривать решения только для контейнеров из первой или второй группы [9] или даже только первой [2]. Цель предложенного подхода заключалась в том, чтобы выяснить эффективность использованных в упомянутых работах стратегий. Действительно ли не имеет смысла перемещать предметы из двух контейнеров третьей или второй групп? Нужен ли обмен предметов из контейнеров первой и второй групп? Как показали результаты проведенного численного эксперимента, в некоторых задачах невозможно достижение оптимального решения, если не рассматривать перемещения двух предметов из контейнеров второй или третьей групп.

Разделение контейнеров на первую и вторую группы связано именно с выяснением поставленных вопросов. Хотя фактически после первого же локального оптимума обычно эти группы совпадают.

Соседнее решение в любой из окрестностей получается при изменении двух контейнеров i_1 и i_2. Для каждого решения вероятность p включения в окрестность определяется следующим образом:

$$p = \begin{cases} p1, \text{если } (S_{i_1} \in S1) \text{ or } (S_{i_2} \in S1) \\ p2, \text{если } (S_{i_1} \in S2) \text{ or } (S_{i_2} \in S2) \\ p3, \text{если } (S_{i_1} \in S3) \text{ and } (S_{i_2} \in S3) \end{cases} \quad (3.8)$$

Значения p1, p2 и p3 определяются экспериментально и зависят от размерности задачи. Чем больше число предметов, тем меньше должна быть вероятность p3, в то время как вероятности p1 и p2 не должны сильно уменьшаться. Это связано с тем, что число переполненных контейнеров после достижения локального оптимума не зависит от размерности задачи и обычно составляет 1-5 контейнеров.

5. Оценочная функция

Введение оценочной функции обусловлено тем, что очень часто возникает ситуация, когда много решений из окрестности имеют одно и то же лучшее значение целевой функции. Возникает необходимость введения иного критерия выбора следующего решения. Поскольку предложенный алгоритм упаковки решает задачу в форме BPP-2, целевая функция здесь есть вес самого тяжелого контейнера. И это значение может измениться, только если был перемещен предмет из него. Очевидно, при любых изменениях других контейнеров значение целевой функции не изменится. И поэтому сложно оценить эффективность соседних решений, полученных из текущего изменением весов других контейнеров, за исключением того случая, когда самым тяжелым становится какой-то иной контейнер (тогда произойдет ухудшение значения целевой функции).

Здесь предлагается оценочная функция, которая представляет собой изменение разницы весов двух контейнеров, из которых перемещаются предметы. Пусть S_{i_1}, S_{i_2} - веса контейнеров в исходном решении X, S'_{i_2}, S'_{i_2} - веса контейнеров в соседнем решении Y, вычисленные по формуле (3.4) или (3.5). Тогда оценка соседнего решения есть

$$\tilde{f}(Y) = \left|S_{i1} - S_{i2}\right| - \left|S'_{i1} - S'_{i2}\right|$$

Лучшим в окрестности будет решение, доставляющее максимум оценочной функции. Практический смысл этой функции в том, что она позволяет выбирать из окрестности в первую очередь решения, «выравнивающие» вес контейнеров. Очевидно, что большим достоинством этой функции является то, что для ее вычисления необходимо фиксированное число арифметических операций, не зависящих от размерности задачи. Во многих приложениях метода поиска с запретами этого достичь не удается [10,11].

6. Заключение

1. В данной работе была рассмотрена задача упаковки в контейнеры и описан общий алгоритм упаковки, использующий контейнерно-ориентированное представление плана упаковки. Данное представление позволило создать алгоритм уменьшения контейнера с постоянной сложностью, не зависящей от размерности исходной задачи.
2. Были предложены алгоритмы построения двух видов окрестностей соседних решений на основе рассматриваемого плана упаковки. Сложность получения каждого решения из окрестностей есть постоянная величина, благодаря используемому представлению решения.
3. Был предложен вероятностный алгоритм построения окрестностей, использующий разделение контейнеров на группы.
4. В работе описана оценочная функция для выбора очередного решения из окрестностей, также имеющая константную сложность.

Список используемых источников

1. Garey M.R. and Johnson D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Frecman, San Francisco. 1979.
2. Martello S.,Toth P. Knapsack Problems, Algorithms and Computer Implemetations. John Wiley and Sons Ltd.- England, 1990.
3. Кочетов Ю.А., Усманова А.Р. Вероятностный поиск с запретами для задачи упаковки в контейнеры.- *Сборник докладов XII Байкальской международной конференции*. 2001;22-27.
4. Усманова А.Р. Вероятностные жадные эвристики для задачи упаковки в контейнеры.- Сборник докладов Оптим-2001. Первая всероссийская научно-практическая конференция по вопросам решения оптимизационных задач в промышленности. Спб,2001.-:141-145.
5. Falkenauer E. A hybrid Grouping Genetic Algorithm for Bin Packing. Journal of Heuristics, Vol.2, No 1,1996.- pp.5-30.
6. Усманова А.Р. Поиск в экспоненциальной окрестности решений для задачи линейной упаковки. Сборник трудов международной конференции «Информационные технологии для интеллектуальной поддержки принятия решений», Май 21-25, Уфа, Россия, 2013. Т.2. С.189-194
7. Усманова А.Р. Структуры списка запретов в методе TS для задачи упаковки. Сборник трудов международной конференции «Информационные технологии для интеллектуальной поддержки принятия решений», Май 21-25, Уфа, Россия, 2013. Т.2. С.208-211
8. Scholl A., Klein R., Jurgens C. Bison: A fast hybrid procedure for exactly solving the one-dimensional bin-packing problem. Computers Ops.Res. Vol. 24, No 7, 1997.-pp. 627-645.
9. Стоян Ю.Г., Яковлев С.В. Математические модели и оптимизационные методы геометрического проектирования. – Киев: Наукова думка, 1986.-268 с.
10. Glover F., Taillard E., de Werra D. A user's guide to tabu search. Annals of Operation research. -Vol. 41,1993.- pp.3-28.
11. Johnson D.S., Demers A., Ullman J.D., Garey M.R., Graham R.L. Worst-case performance bounds for simple one dimensional packing algorithms. SIAM J. Comput., Vol.3, No 4,1974.- pp.229-325.
12. T. Fabarisov I., Yusupova N. I., Ding K., Morozov A., Janschek K.. Analytical toolset for model-based stochastic error propagation analysis: extension and optimization towards industrial requirements //Системная инженерия и информационные технологии–2019.–№1.–pp.63-66. URL:http://siit.ugatu.su/index.php/journal/article/view/7