ИНФОРМАТИКА

2011 апрель-июнь № 2

УДК 621.129.13

Да Юн Цао

ЭФФЕКТИВНЫЙ BEST-FIT-АЛГОРИТМ ДЛЯ РЕШЕНИЯ ЗАДАЧ ДВУХМЕРНОЙ ОРИЕНТИРОВАННОЙ УПАКОВКИ В КОНТЕЙНЕРЫ

Рассматривается задача двухмерной упаковки в контейнеры (2D-BPP), которая заключается в минимизации числа одинаковых больших прямоугольников, используемых для упаковки конечного набора прямоугольников. Предлагается эффективный Best-Fit-алгоритм (IBF), основанный на методе вогнутого угла, для решения 2D-BPP. Вычислительный эксперимент по оценке эффективности алгоритма в сравнении с четырьмя классическими алгоритмами показывает, что IBF получил лучшие результаты почти для всех тестовых примеров за меньшее время.

Введение

Даны n прямоугольных предметов с размерами $w_i \times h_i$, $i = 1, \ldots, n$, и неограниченное количество одинаковых контейнеров размером $W \times H$. Задача 2D-BPP состоит в том, чтобы минимизировать число требуемых контейнеров, в которые будут упакованы все предметы без перекрытия. Без потери общности будем считать, что все исходные данные являются целыми положительными числами. Исследуемая задача, как известно, является NP-трудной в сильном смысле [1] и имеет много практических приложений, таких как сокращение стандартизированных единиц фонда в мебельной и стекольной промышленности, упаковка на полках, на транспорте и т. д. Более подробные описания 2D-BPP можно найти в [2–4].

В настоящей работе предлагается эффективный *Best-Fit*-алгоритм для задачи 2D-BPP, когда предметы и контейнеры имеют фиксированную ориентацию, параллельную координатным осям. Такое название алгоритма обусловлено тем фактом, что алгоритмы упаковки, в которых описываются правила выбора позиции для размещения очередного предмета из множества допустимых позиций, называются *Best-Fit*. Приводится сравнение предлагаемого алгоритма с другими алгоритмами. Эксперименты по 36 тестам из известной литературы с размерностью до 120 предметов показывают эффективность и устойчивость предлагаемого подхода.

1. Основные понятия

Определим множество O прямоугольных областей O_i, каждая из которых принадлежит внутренности некоторого контейнера, как множество максимальных по включению прямоугольных областей, не пересекающихся с уже размещенными в контейнерах предметами.

Различные прямоугольные области из множества O, принадлежащие одному и тому же контейнеру, могут пересекаться. Множество O меняется после размещения в контейнере очередного предмета.

В каждой из таких областей из O рассмотрим углы и определим их типы.

Если вершина угла принадлежит сторонам двух размещенных предметов, сторонам одного предмета и одной стороне контейнера или углу контейнера, то такой угол будет называться действительным углом (*RCC*) и обозначаться как C^+.

Если вершина угла принадлежит сторонам только одного размещенного предмета или только одной стороне контейнера, то такой угол будет называться притворным углом (*SCC*) и обозначаться как C^-.

На рис. 1 изображен контейнер, в котором множество O состоит из шести областей с углами (C_1, C_2, C_{16}, C_8), (C_{10}, C_{12}, C_8), (C_9, C_5, C_{13}, C_8), (C_{11}, C_{12}, C_{14}, C_7), (C_3, C_4, C_{14}, C_{15}), (C_5, C_6, C_7).

Пусть $\{C_1, \ldots, C_m\}$ обозначает набор углов контейнера. Каждый угол в контейнере является кандидатом на размещение угла нового предмета.

При размещении нового предмета p_t в угол C_j определим величины.

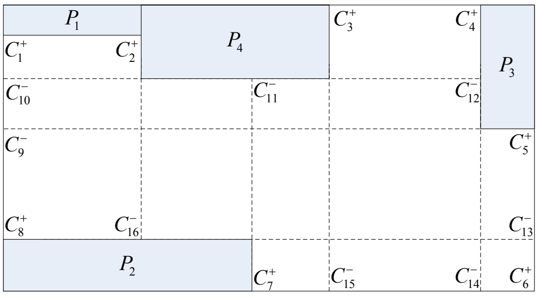

Рис. 1. Действительный (*RCC*) и притворный (*SCC*) углы

Пусть r соответствует количеству сторон уже размещенных предметов и сторон самого контейнера, которых касается p_t, а s соответствует количеству углов, исчезающих при размещении p_t. Тогда определим величину $Fit_A\_C_j(p_t)$ для оценки качества упаковки нового предмета p_t в контейнере в позиции C_j по формуле

$$Fit_A\_C_j(p_t) = 2r + \sum_{k=1}^{s} q_k \ , \tag{1}$$

где $q_k = 2$, если k-й угол является действительным углом, и $q_k = 1$ в противном случае.

Если существуют два угла C_i и C_j с равными значениями величин $Fit_A\_C_i(p_t)$ и $Fit_A\_C_j(p_t)$, определим параметр *Touch Length* (*TL*) следующим образом. Пусть предмет p_t будет касаться своей нижней стороной других предметов или стороны контейнера, тогда величина $p_t\_TL_B$ соответствует суммарной длине касания. Аналогично определим величины $p_t\_TL_T$, $p_t\_TL_L$ и $p_t\_TL_R$ для верхней, левой и правой сторон p_t соответственно (рис. 2).

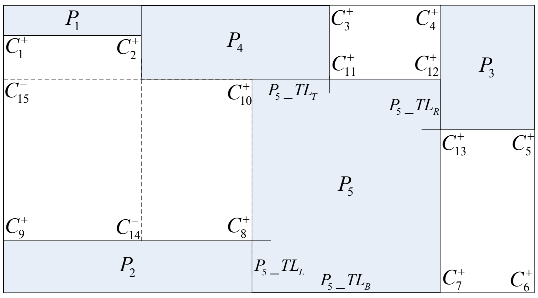

Рис. 2. Параметр *Touch Length* края

Для размещения предмета выбирается тот угол, у которого величина $Fit_B\_C_j(p_t)$, вычисляемая по формуле (2), максимальна:

$$Fit_B\_C_j(p_t) = p_t\_TL_L + p_t\_TL_T + p_t\_TL_R + p_t\_TL_B \, . \tag{2}$$

2. Эффективный Best-Fit-алгоритм

Алгоритм начинается с расчета нижней оценки L_0 по формуле [5]

$$L_0 = \left\lceil \frac{\sum_{i=1}^{n} w_i \times h_i}{W \times H} \right\rceil . \tag{3}$$

После вычисления L_0 алгоритм *IBF* первоначально сортирует все предметы по невозрастанию площадей. Вначале L_0 контейнеров считаются активными. При упаковке p_t исследуются все углы всех активных контейнеров. Если не существует позиции для упаковки p_t, то активизируется новый контейнер, в котором p_t упаковывается в левый нижний угол.

Если существуют позиции для размещения p_t, вычисляются значения $Fit_A\_C_j(p_t)$ и $Fit_B\_C_j(p_t)$ и определяется позиция с максимальным значением $Fit_A\_C_j(p_t)$.

Процесс повторяется до тех пор, пока все предметы не будут упакованы.

3. Вычислительный эксперимент

В данном разделе приводятся результаты вычислительного эксперимента. Алгоритм был реализован с использованием языка C++, сравнение выполнено на ноутбуке IBM PC T400 с частотой 2,26 ГГц и 2048 МБ оперативной памяти. Тесты beng1-beng8 взяты из работы [6]. Тесты cgcut1-cgcut3 описаны в [7], gcut1-gcut13 и ngcut1-ngcut12 – в [8, 9]. В таблице величина *LB* является нижней границей L_4, предложенной в [5].

Сопоставление эвристического алгоритма *IBF* с другими алгоритмами

Номер теста	Название теста	*N*	*LB*	*FFF*	*FBS*	*TS*	Время, с	*EA*	Время, с	*IBF*	Время, с
1	beng1	20	4	4	4	4	0,01	4	0,02	4	0,006891*
2	beng2	40	6	7	7	7	100,02	–	–	7	0,031282*
3	beng3	60	9	10	9	9	0,01	9	0,01	9	0,072011*
4	beng4	80	11	12	12	12	100,06	11	7245,07	11	0,134953*
5	beng5	100	14	16	15	14	0,01	14	0,02	14	0,25086*
6	beng6	40	2	2	2	2	0,01	2	0,01	2	0,028775*
7	beng7	80	3	3	3	3	0,01	3	0,02	3	0,14189*
8	beng8	120	5	5	5	5	0,01	5	0,02	5	0,387235*
9	cgcut1	16	2	2	2	2	0,01	2	0,01	2	0,003608*
10	cgcut2	23	2	3	3	2	0,01	2	0,01	2	0,006958*
11	cgcut3	62	23	26	26	23	0,01	23	0,04	23	0,135586
12	gcut1	10	4	5	5	5	100,02	5	0,02	5	0,001785*
13	gcut2	20	6	7	7	6	50,11	6	0,02	7	0,00748
14	gcut3	30	8	9	8	8	0,01	8	0,01	8	0,017005*
15	gcut4	50	13	15	15	14	100,03	14	3,73	14	0,068204*
16	gcut5	10	3	4	4	4	100,02	3	0,01	3	0,002822*
17	gcut6	20	6	8	8	7	100,02	7	0,90	7	0,006538*
18	gcut7	30	10	12	12	12	100,01	11	0,28	11	0,018789*
19	gcut8	50	12	15	14	14	100,03	–	–	14	0,055807*
20	gcut9	10	3	3	3	3	0,01	3	0,02	3	0,001746*
21	gcut10	20	7	8	8	8	100,03	7	1,22	8	0,006865
22	gcut11	30	8	10	10	9	100,03	9	909,70	9	0,018518*
23	gcut12	50	16	17	17	16	23,56	16	0,17	16	0,065784*
24	gcut13	32	2	2	2	2	0,01	2	0,01	2	0,019977*
25	ngcut1	10	2	3	3	3	100,02	3	0,13	4	0,001261
26	ngcut2	17	3	4	4	4	100,02	4	0,85	4	0,004182*
27	ngcut3	21	3	4	4	4	100,02	3	2,38	4	0,005509
28	ngcut4	7	2	2	2	2	0,01	2	0,01	2	0,000818*
29	ngcut5	14	3	4	4	3	0,01	3	0,01	3	0,002467*
30	ngcut6	15	2	3	3	3	100,02	3	68,98	3	0,003334*
31	ngcut7	8	1	2	2	1	0,01	1	0,01	1	0,001048*
32	ngcut8	13	2	2	2	2	0,01	2	0,01	2	0,002011*
33	ngcut9	18	3	4	4	4	100,02	3	0,67	4	0,006092
34	ngcut10	13	3	3	3	3	0,01	3	0,01	4	0,001994
35	ngcut11	15	2	3	3	3	100,02	2	3,53	3	0,007886*
36	ngcut12	22	3	4	4	4	100,02	3	2,07	3	0,002799*

Примечание: * означает, что *IBF* получил наилучшие результаты среди пяти алгоритмов.

Предлагаемый алгоритм сравнивается со следующими алгоритмами:

1. Двумя эвристическими алгоритмами *FFF* и *FBS*.

FFF (Finte First-Fit) и *FBS* (Finite Best Strip) описаны в работе [10], а результаты их расчетов – в [11]. Из таблицы видно, что почти все результаты упаковки предложенным алгоритмом не хуже, чем *FFF* и *FBS*.

2. Метаэвристическим алгоритмом *TS* – гибридным алгоритмом, основанном на *Tabu search* [11].

Отметим, что в литературе результаты тестирования получены как первоначальная верхняя граница. Установлено, что *IBF* может получить тот же результат с меньшим временем почти для всех тестов (рис. 3).

Алгоритм *TS* был закодирован в Fortran 77 и выполнен на компьютере Silicon graphics INDY R4000sc с частотой 100 МГц, который в 22,6 раз медленнее используемого процессора, поэтому результаты были умножены на 22,6 (рис. 3). Это не совсем корректно с точки зрения трудоемкости алгоритма, но авторов здесь интересует качество полученных решений.

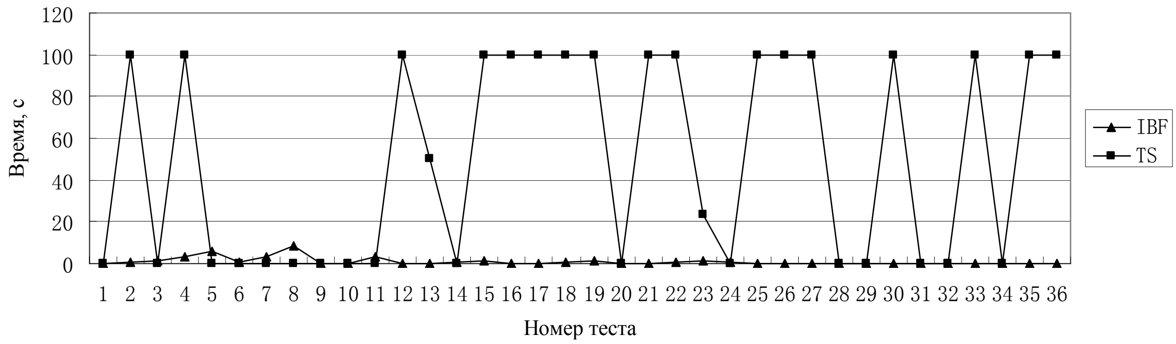

Рис. 3. Разница во времени между *IBF* и *TS* с запретами на основе метаэвристики для 2D-BPP

3. Точным алгоритмом *EA* – алгоритмом типа ветвей и границ [5]. Результаты испытаний были получены на компьютере DIGI-TAL DECstation 5000/240 (5,3 Mflops). Точный алгоритм иногда может выполняться быстро, но из таблицы видно, что на некоторых тестах он работает очень медленно. Кроме того, следует отметить, что *EA* не может решить проблему для некоторых тестов, например beng2 и gcut8. Сравнение машинного времени и качества упаковки показало, что предложенный алгоритм является более предпочтительным, чем точный (который не дал результата за приемлемое время).

В процессе эксперимента обнаружено, что если сортировать последовательность *PS* по высоте предмета, то *IBF* может обеспечить лучшие результаты упаковки для случаев ngcut9 и ngcut12. Поэтому *IBF* может быть выполнен трижды в соответствии с приоритетами высоты, ширины и размера области отдельно, а затем следует выбрать лучший результат из этих трех решений.

Заключение

В работе построен эффективный Best-Fit-алгоритм для задачи двухмерной ориентированной упаковки в контейнеры. Эксперименты показали, что эвристический алгоритм получает удовлетворительные результаты за приемлемое время.

Список литературы

1. Garey, M.R. Computers and intractability / M.R. Garey, D.S. Johnson. – San Francisco, USA, 1979. – 338 p.

2. Harald, D. A typology of cutting and packing problems / D. Harald // European Journal of Operational Research. – 1990. – Vol. 44, № 2. – P. 145–159.

3. Lodi, A. Recent advances on two-dimensional bin packing problems / A. Lodi, S. Martello, D. Vigo // Discrete Applied Mathematics. – 2002. – Vol. 123, № 1–3. – P. 379–396.

4. Heike, H. An improved typology of cutting and packing problems / H. Heike, W. Gerhard, S. Holger // European Journal of Operational Research. – 2007. – Vol. 183, № 3. – P. 1109–1130.

5. Martello, S. Exact solution of the two-dimensional finite bin packing problem / S. Martello, D. Vigo // Management science. – 1998. – Vol. 44, № 3. – P. 388–399.

6. Bengtsson, B.E. Packing rectangular pieces – a heuristic approach / B.E. Bengtsson // The computer journal. – 1982. – Vol. 25, № 3. – P. 353–357.

7. Christofides, N. An algorithm for two-dimensional cutting problems / N. Christofides, C. Whitlock // Operations Research. – 1977. – Vol. 25, № 1. – P. 30–44.

8. Beasley, J.E. Algorithms for unconstrained two-dimensional guillotine cutting / J.E. Beasley // Journal of the Operational Research Society. – 1985. – Vol. 36, № 4. – P. 297–306.

9. Beasley, J.E. An exact two-dimensional non-guillotine cutting tree search procedure / J.E. Beasley // Operations Research. – 1985. – Vol. 33, № 1. – P. 49–64.

10. Berkey, J.O. Two-dimensional finite bin-packing algorithms / J.O. Berkey, P.Y. Wang // The Journal of the Operational Research Society. – 1987. – Vol. 38, № 5. – P. 423–429.

11. Lodi, A. Approximation algorithms for the oriented two dimensional bin packing problem / A. Lodi, S. Martello, D. Vigo // European Journal of Operational Research. – 1999. – Vol. 112, № 1. – P. 158–166.

Поступила 29.12.10

Белорусский государственный университет,
Минск, пр. Независимости, 4

Харбинский научно-технический университет,
Харбин Сюефулу, 52
e-mail: caodayong@hrbust.edu.cn

Dayong Cao

AN EFFICIENT BEST-FIT ALGORITHM FOR THE TWO-DIMENSIONAL ORIENTED BIN PACKING PROBLEM

The two-dimensional bin packing problem (2D-BPP) consists of minimizing the number of identical large rectangles (bins) used to pack a set of rectangles (items). In this paper the efficient best-fit algorithm (*IBF*) to solve 2D-BPP is proposed. We have tested the efficiency of our methods against four classical algorithms. The experiments demonstrated that the *IBF* provides more satisfied results for almost all test instances in a shorter time.