第33卷 第4期 计 算 机 工 程 2007年2月
Vol.33 No.4 Computer Engineering February 2007
·人工智能及识别技术· 文章编号：1000—3428(2007)04—0160—03 文献标识码：A 中图分类号：TP301.06

求解矩形packing问题的贪心算法

陈端兵，黄文奇
(华中科技大学计算机科学与技术学院，武汉 430074)

摘 要： 在货物装载、木材下料、超大规模集成电路设计等工作中提出了矩形packing问题。对这一问题，国内外学者提出了诸如模拟退火算法、遗传算法及其它一些启发式算法等求解算法。该文利用人类的智慧及历史上形成的经验，提出了一种求解矩形packing问题的贪心算法。并对21个公开测试实例进行了实算测试，所得结果的平均面积未利用率为0.28%，平均计算时间为17.86s，并且还得到了其中8个实例的最优解。测试结果表明，该算法对求解矩形packing问题相当有效。

关键词： 矩形packing；贪心算法；占角动作

Greedy Algorithm for Rectangle-packing Problem

CHEN Duanbing, HUANG Wenqi
(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074)

【Abstract】 The rectangle-packing problem often appears in loading, timber cutting, very large scale integration design, etc. Many algorithms such as simulated annealing, genetic algorithm and other heuristic algorithms are proposed to solve it. This paper recommends an efficient greedy algorithm according to ten-thousand-year experience and wisdom of human being. Twenty-one public instances are tested by the algorithm developed. The average trim loss and runtime is 0.28% and 17.86s respectively. Furthermore, 8 instances are achieved optimum solutions. Experimental results demonstrate that the algorithm developed is fairly efficient for solving rectangle-packing problem.

【Key words】 Rectangle packing; Greedy algorithm; Corner-occupying action

对矩形packing问题，有许多方面的应用，如货物装载、超大规模集成电路的布局与布图规划、木材下料、网站和报刊杂志的版面布局等。对这一问题，国内外许多学者，如Hopper E[1]，Beasley J E[2]，Wu Y L[3]，Zhang D F[4]等，利用不同的思想提出了各种各样的算法，这些算法大致可分为两类：即随机优化算法和确定性构造算法。对于随机优化算法，其核心是设计一种能表示布局形姿态的数据编码；对于确定性的构造算法，关键在于确定一种放置矩形块的规则。

2004年Beasley J E提出了一种基于人口进化理论的启发式算法[2]。2002年Wu Y L等提出了基于最小自由度优先的启发式算法[3]，2005年Zhang D F等提出了一种基于分治策略的启发式算法[4]。近年来，一些学者把人类的经验和智慧加以形式化而得到拟人算法[5]。

受以上这些研究的启发，本文在文献[3, 5]的基础上，以最小化容器的面积未利用率为优化目标，提出了一种求解矩形packing问题的贪心算法。该算法的主要思想就是采用占角放置的原则，即放进矩形容器的矩形块始终占据一个角，在占角的前提下，还尽量占穴。利用本文提出的算法，对文献[1]给出的21个测试实例进行了实算测试，并和Heuristic1[3]与HH[4]进行了比较。Heuristic1，HH和本文算法所得结果的平均面积未利用率分别为：0.96%，0.88%和0.28%，平均计算时间分别为：1 379.97s，13.48s和17.86s；另外，用本文算法还得到了8个实例的最优解，Heuristic1可得到2个实例的最优解，HH可得到3个实例的最优解。测试结果表明，本文算法对求解矩形packing问题相当有效。

1 问题描述

已知一个宽为W_0，高为H_0的矩形容器R_0，以及n块矩形块$R_1, R_2, \ldots, R_n$，每一矩形块R_i的宽为w_i，高为h_i。容器的左下角和直角坐标平面的原点重合，容器的边和坐标轴平行。现在的问题是，要把这n块矩形块尽可能多地放进容器R_0中，也就是说，要使容器的面积未利用率尽可能得小，并满足如下限制条件：

(1)任一放入容器的矩形块的边必须和容器的边平行；

(2)任意两块放入容器的矩形块没有嵌入。

2 算法基本思想与基本策略

2.1 基本思想

在某一时刻，已经按放置规则在矩形容器中放置了若干矩形块，那么对还未放入的矩形块，选择哪一块，放到哪一个位置好呢？中国很早以前就有这样一句警语“金角银边草肚皮”，从此警语中可悟出一个道理：放进去的矩形块最好是占据某个角，其次是贴边，最差的是悬空。若放进去的矩形块不仅占角，还和形成这个角以外的某些矩形块贴边，这样的动作就作得相当优美，对应的矩形块就占据了一个穴。可将前面提到的警语发展为“金角银边草肚皮，价值最高钻石穴”。

本文提出的算法就基于这种思想，即在放置矩形块时，总是占据某个角，而且动作的穴度还要尽量大，这样放置就可使矩形块之间挨得很紧，减小对空间的浪费。

2.2 基本概念

(1)格局

基金项目： 国家自然科学基金资助项目(10471051)；国家“973”计划基金资助项目(2004CB318000)

作者简介： 陈端兵(1971－)，男，博士生，主研方向：NP难问题高效求解；黄文奇，教授、博导

收稿日期： 2006-03-14 **E-mail：** hustcdb@yahoo.com.cn

在某一时刻，容器中已经互不嵌入地放了若干矩形块，还有若干矩形块在容器外面等待放入，这种状态，称为一个格局。若容器中还没有矩形块，此时的格局称为初始格局；若全部矩形块都按放置规则放进了容器或容器中已放不下任何矩形块，此时的格局称为终止格局。

(2)占角动作

在某一格局下，若即将放进去的矩形块 R 与容器中已有矩形块（包括构成容器的 4 块矩形块）中的某两块矩形块的不同方向的边有重叠(并且重叠的长度大于 0)，则称放此矩形块的这种作法为一个占角动作。若矩形块 R 按占角动作放进容器后，R 已有矩形块没有嵌入，并且不超出容器的边界，则称这样的占角动作为合法占角动作。如在图 1 中，阴影矩形块是已经放置好的矩形块，若将矩形块 1 放在位置 A（或位置 B，或位置 C，或位置 D），则放置矩形块 1 的动作就是合法占角动作；若放在位置 E 或位置 F，就不是占角动作。

特别地，若放进去的矩形块不仅占角，还和形成这个角以外的某些矩形块贴边，则称这样的放置动作为占穴动作。如在图 1 中，若矩形块 1 放在位置 A，那么矩形块 1 不仅占据了由矩形块 a 和 b 形成的角，还和矩形块 c 贴边，因而矩形块 1 占据了由矩形块 a、b、c 形成的穴，此时，放置矩形块 1 的动作就是占穴动作。

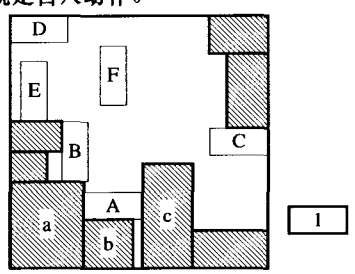

图 1 占角动作

(3)两矩形块间的距离

在两矩形块 R_i 和 R_j 中各取一点 $P_i \in R_i$，$P_j \in R_j$，设这两点间的曼哈顿距离为 D_{ij}，则两矩形块间的距离 $d = \min_{P_i \in R_i,\, P_j \in R_j}(D_{ij})$。例如，在图 2(a)~图 2(c)中，两矩形块间的距离分别为 0，l 和 l_1+l_2。

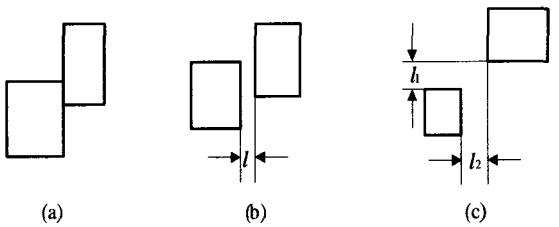

图 2 两矩形块间的距离

(4)占角动作的穴度

如图 3 所示，若将矩形块 R_i 按合法占角动作放进矩形容器之后，R_i 与形成这个角以外的其它矩形块（包括构成容器的 4 块矩形块）的距离的最小值为 $d_{\min}$，则此占角动作的穴度 C_i 定义为

$$C_i = 1 - \frac{d_{\min}}{\sqrt{w_i \cdot h_i}}$$

式中，w_i 和 h_i 分别为矩形块 R_i 的宽和高。

占角动作的穴度反映了即将放入的矩形块与容器中已有矩形块的抱紧程度，穴度越大，就抱得越紧，在选择占角动作时，就选穴度最大的占角动作。

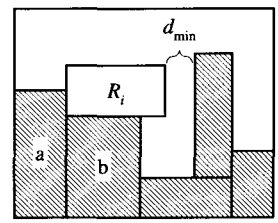

图 3 占角动作的穴度

(5)占角动作的贴边数

设某一占角动作所关联的矩形块为 R，那么与 R 贴边的矩形块块数(包括构成容器的 4 块矩形块)称为该占角动作的贴边数。在图 4 中，若将矩形块 1 放在位置 A，则对应占角动作的贴边数为 3；若放在位置 B，贴边数为 2；若放在位置 C，贴边数为 4。若有多个穴度最大的占角动作，就选贴边数最大的占角动作，例如，在图 4 的 3 个占角动作中，就选放在位置 C 的占角动作。

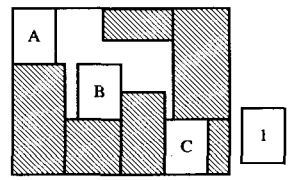

图 4 占角动作的贴边数

2.3 占角动作选择原则

(1)选择穴度最大的占角动作，若这样的占角动作有多个，转(2)；

(2)选择贴边数最大的占角动作，若这样的占角动作有多个，转(3)；

(3)随机选择一个占角动作。

3 算法的程序实现

Begin

输入矩形容器 R_0 以及 n 块矩形块 $R_1, R_2, \ldots, R_n$ 的宽和高，初始化得到初始格局 C_0。在下面的描述中，C 表示当前格局，C_{best} 表示截至目前得到的最好格局，u_{best} 表示格局 C_{best} 的面积未利用率，k 表示已经放入容器的矩形块块数。

For i:=1 to n

For j:=1 to 2

$C \Leftarrow C_0$;

If j=1 then

将矩形块 R_i 横放在容器的左下角;

Else

将矩形块 R_i 竖放在容器的左下角;

End if

R_i 放进容器之后，得到新的格局，仍记为 C，$k \Leftarrow 1$;

在格局 C 之下，枚举所有的合法占角动作，并计算每一个占角动作的穴度和贴边数;

While 存在合法占角动作且 $k < n$ do

按占角动作选择原则选择唯一一个占角动作;

作这个占角动作，作完之后得到新的格局，仍记为 C，$k \Leftarrow k+1$;

在格局 C 之下，枚举所有的合法占角动作，并计算每一个占角动作的穴度和贴边数;

End while

计算格局 C 的面积未利用率 u;

If $k=n$ then

输出 C 和 u，成功停机;

Else If $u<u_{best}$ then

$C_{best} \Leftarrow C$，$u_{best} \Leftarrow u$;

End if

End for

End for

输出 C_{best} 和 u_{best};

End

4 实算结果

对本文提出的算法，用 C#.net 语言编程实现，并对文献[1]给出的 21 个测试实例进行了实算测试。对每一个实例，矩形块的宽和高给定，容器的宽也给定，需要确定容器的高，使得所有矩形块都能放进容器。对每一个实例，其最优解的布局是完美布局，即所有矩形块都能放进容器，且容器的面积未利用率为 0%。测试实例的详细信息可参阅文献[1]。

将本文算法所得结果和算法 Heuristic1[3] 及 HH[4] 的计算结果进行了对比，如表 1 所示，表 1 中的计算时间为相应计算机的 CPU 时间。

对每一个实例，将容器的高度取为最优解的容器高度，然后用本文算法计算得到容器的面积未利用率及其布局结果。作为例子，图 5 给出了实例 C11，C13，C41，C51，C61 和 C71 的布局结果，在图 5 中，灰色区域是未利用的区域，实例 C11 和 C13 的布局是完美布局。

图 5 部分实例的布局结果

从对比结果来看，本文算法所得结果的综合性能好于 Heuristic1 和 HH 所得结果。Heuristic1，HH 和本文算法所得结果的平均面积未利用率分别为：0.96%，0.88%和 0.28%，平均计算时间分别为：1 379.97s，13.48s 和 17.86s (表 1 最后 1 行所示)；另外，用本文算法还得到了 8 个实例的最优解，用 Heuristic1 可得到 2 个实例的最优解，用 HH 可得到 3 个实例的最优解。

表 1 Heuristic1，HH 和本文算法的对比

实例	矩形块数目	容器大小	本文算法		Heuristic1		HH	
			面积未利用率 (%)	计算时间 (s)	面积未利用率 (%)	计算时间 (s)	面积未利用率 (%)	计算时间 (s)
C11	16	20×20	0.00	0.04	2.00	1.48	2.00	0.00
C12	17	20×20	0.00	0.32	2.00	2.42	3.50	0.00
C13	16	20×20	0.00	0.06	2.50	2.63	0.00	0.00
C21	25	40×15	0.00	0.51	0.67	13.35	0.67	0.05
C22	25	40×15	0.00	0.16	0.00	10.88	0.00	0.05
C23	25	40×15	0.00	0.08	0.00	7.92	0.00	0.00
C31	28	60×30	1.11	0.97	0.67	23.72	0.67	0.05
C32	29	60×30	1.28	1.03	0.83	34.02	2.44	0.05
C33	28	60×30	1.44	0.90	0.78	30.97	1.56	0.05
C41	49	60×60	0.33	3.57	0.97	438.18	1.36	0.44
C42	49	60×60	0.00	0.14	0.22	354.47	0.78	0.44
C43	49	60×60	0.22	3.47			0.44	0.33
C51	73	60×90	0.11	7.73	0.30	1 417.52	0.44	1.54
C52	73	60×90	0.00	2.10	0.04	1 507.52	0.44	1.81
C53	73	60×90	0.15	7.71	0.83	1 466.15	0.37	2.25
C61	97	80×120	0.38	17.05	0.25	7 005.73	0.66	5.16
C62	97	80×120	0.06	15.09	3.74	5 537.88	0.26	5.33
C63	97	80×120	0.42	16.49	0.54	5 604.70	0.50	5.60
C71	196	160×240	0.20	101.38			1.25	94.62
C72	197	160×240	0.04	93.22			0.55	87.25
C73	196	160×240	0.13	102.97			0.69	78.02
平均值			0.28	17.86	0.96	1379.97	0.88	13.48

注：本文算法所用计算机为 Pentium 4 IBM R40 (2.0GHz CPU, 256MB RAM)，Heuristic1 所用计算机为 SUN Sparc20/71 (71MHz CPU, 64MB RAM)，HH 所用计算机为 Dell GX260 (2.4GHz CPU)。

5 结论

本文利用人类的智慧及历史上来形成的经验，以最小化容器的面积未利用率为优化目标，提出了一种求解矩形 packing 问题的贪心算法。

用本文提出的算法，对 21 个测试实例进行了实算测试，所得结果的平均面积未利用率为 0.28%，平均计算时间为 17.86s，并且还得到了 8 个算例的最优解。实算结果表明，本文提出的算法对求解矩形 packing 问题相当有效。

参考文献

1 Hopper E, Turton B. An Empirical Investigation of Meta-heuristic and Heuristic Algorithm for a 2D Packing Problem[J]. European Journal of Operational Research, 2001, 128(1): 34-57.

2 Beasley J E. A Population Heuristic for Constrained Two-dimensional Non-guillotine Cutting[J]. European Journal of Operational Research, 2004, 156(3): 601-627.

3 Wu Y L, Huang W Q, Lau S C, et al. An Effective Quasi-human Based Heuristic for Solving the Rectangle Packing Problem[J]. European Journal of Operational Research, 2002, 141(2): 341-358.

4 Zhang D F, Deng A S, Kang Y. A Hybrid Heuristic Algorithm for the Rectangular Packing Problem[C]//Proc. of the 5th International Conference on Computational Science. 2005: 22-25.

5 Huang W Q, Li Y, Akeb H, et al. Greedy Algorithms for Packing Unequal Circles into a Rectangular Container[J]. Journal of the Operational Research Society, 2005, 56(5): 539-548.