维普资讯 http://www.cqvip.com

一种求解集装箱装载问题的启发式算法*)

陈端兵[1] 黄文奇[2] 尚明生[1] 傅 彦[1]
(电子科技大学计算机学院 成都 610054)[1] (华中科技大学计算机学院 武汉 430074)[2]

摘 要 所谓集装箱装载问题,就是将若干大小不同的长方体盒子装进一个大小已知的长方体容器,其目标是最大化容器的积载率。对这一问题,国内外学者利用不同的哲学思想,提出了诸如遗传算法、模拟退火算法等求解算法。本文提出一种求解此问题的基于最大穴度优先原则的启发式算法。算法中使用了两个重要的策略:最大穴度原则和最小边度原则。用一些公开的算例对算法性能进行了实算测试,测试结果表明:算法所得结果的容器积载率高,是求解集装箱装载问题的有效算法。

关键词 Packing 问题,集装箱装载,启发式算法,穴度,边度

Heuristic Algorithm for Solving the Container Loading Problem

CHEN Duan-bing[1] HUANG Wen-qi[2] SHANG Ming-sheng[1] FU Yan[1]
(School of Computer Science, University of Electronic Science and Technology of China, Chengdu 610054, China)[1]
(School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China)[2]

Abstract The container loading problem is the problem of loading a series of boxes with different sizes into a fixed cuboid container. The objective is to maximize the volume utilization of the container. Researchers propose many algorithms such as genetic algorithm and simulated annealing to solve it based on different idea. This paper presents a heuristic algorithm based on maximum caving degree first principle for solving this problem. Two important strategies are considered in the algorithm proposed. One is the maximum caving degree principle, the other is the minimum edge degree principle. The performance of the algorithm is evaluated by some public test problems. High volume utilization is obtained by running the algorithm. Experimental results demonstrate that the algorithm proposed is fairly efficient for solving the container loading problem.

Keywords Packing problems, Container loading, Heuristic algorithm, Caving degree, Edge degree

1 引言

集装箱装载问题就是将若干大小不同的长方体盒子装进一个大小已知的长方体容器。通常情况下,优化的目标是最大化容器的装载效率,即最大化容器的积载率。在货运码头、物流、仓储等工作中均牵涉到这一问题。

集装箱装载问题属于 NP 难问题[1],因而求解此问题的算法大多是启发式算法。国内外学者利用不同的哲学思想提出了许多启发式算法,如 Loh 和 Nee[2], Bischoff 和 Ratcliff[3],许光泞和俞金寿[4],等等。

近年来,国内外许多学者都在致力于集装箱装载问题求解算法的研究。Bortfeldt 和 Gehring 于 2001 年提出了一种求解集装箱装载问题的混合遗传算法——CBGAS[5]。Hifi 提出了求解三维切割/装填问题的若干求解算法,他们使用爬山的策略,构造了几个启发式算法,这些算法在计算时间和结果精度上有一个较好的折衷[6]。2005 年,Moura 和 Oliveira[7] 在 GRM_{OD}(George and Robinson heuristic[8] 的改进版本)和 G_{RASP}(贪心随机自适应搜索算法[9])的基础上提出了一种新的求解算法——$GRM_{OD}G_{RASP}$。针对问题的特点,Gehring 和 Bortfeldt 提出了求解此问题的并行遗传算法[10],Bortfeldt 等提出了并行禁忌搜索算法[11]。2002 年,Pisinger 在泥瓦工砌墙的实践基础上提出了一种新的启发式算法[12]。2007 年,杨德荣和杨超提出了求解三维装箱布局的单向寻优搜索算法,并对两个典型算例进行了实算[13]。

本文在最大穴度优先的基础上,提出了一种求解集装箱装载问题的启发式算法,其优化目标是最大化容器的积载率。算法中使用了两个重要的策略。第一个策略就是放进容器的盒子始终占据一个角,甚至占穴,这一策略是文[14-16]的一个发展和提高。另一个策略是即将放进容器的盒子和其它已放进容器的盒子所形成的棱的数量要尽量地少。这样一来,放进容器的盒子就会抱得很紧,减少了空余空间,从而提高容器的积载率。我们用 Loh 和 Nee[2] 提出的 15 个测试算例 LN test problems、文[17]所使用的算例 Test1 和文[18]所使用的算例 Test2 对算法性能进行了实算测试。对算例 LN test problems,本文算法得到了其中 13 个实例的最优解,平均容器积载率为 70.6%;对算例 Test1 和 Test2,本文算法所得结果的容器积载率分别为 87.26% 和 92.08%。测试结果表明,本文提出的算法对求解集装箱装载问题是行之有效的。

*)国家自然科学基金资助项目 10476006,国家高技术研究发展计划(863 计划)2006AA01Z414,2007AA01Z440,国家 242 信息安全计划项目(2007B27),四川省应用技术研究与开发项目支撑计划(2008GZ0009)。**陈端兵** 讲师,主要研究方向为 NP 难问题高效求解、数据挖掘;**黄文奇** 教授,博士生导师,主要研究方向为 NP 难问题高效求解、人工智能、博弈等;**尚明生** 副教授,主要研究方向为数据挖掘、网络并行计算;**傅 彦** 教授,博士生导师,主要研究方向为数据挖掘、信息安全、模式识别等。

维普资讯 http://www.cqvip.com

2 问题描述

在实际的集装箱装载问题中，需要考虑许多限制条件[3]。本文只考虑方向约束，例如对一些特殊盒子(比如内装电视机的盒子)，只能沿某几个方向放置。

在上述假设条件下，集装箱装载问题可描述为：已知一个长方体容器，其长度为L，宽度为W，高度为H，以及n个盒子$B_i(i=1,2,\cdots,n)$，每一个盒子B_i的长度为l_i，宽度为w_i，高度为$h_i(i=1,2,\cdots,n)$。容器的前左下角位于三维直角坐标系的原点(0,0,0)，容器的6个面平行于三维直角坐标系的XY-,XZ-或YZ-坐标面。现在的问题是，要将这n个盒子尽可能“多”地放进容器中，这里“多”的度量是已经放进容器的盒子的体积之和，在放置过程中必须满足如下限制条件：

① 任一放进容器的盒子的每一个面必须和容器的某一个面平行；

② 任意两个已放进容器的盒子没有嵌入(即两盒子重叠的体积不能大于0)；

③ 任一放进容器的盒子不能超出容器的边界；

④ 任一放进容器的盒子必须满足方向约束。

3 算法基本框架

3.1 算法思想

在某一时刻，已经按放置规则向容器中放置了若干盒子，那么对还未放入的盒子，选择哪一个，放置到哪一个位置好呢？本文中，我们按照如下原则向容器中放置盒子：放进容器的盒子始终占据由三个先前已放进容器的盒子所形成的角，并且放置动作的穴度还要尽可能地大；若有多个穴度最大的动作，就挑选边度最小的动作(边度体现了放进容器中各盒子所形成布局的规整程度，在本节(5)中将给出边度的详细定义)。这样一来，放进容器的诸盒子就抱得非常紧凑，从而提高了容器的积载率。

3.2 基本概念

文献[16]中提出了针对二维矩形packing问题的占角动作和穴度的概念。本文中，我们对这两个概念加以扩充，以适用于集装箱装载问题。另外，本文还将提出边度这一重要概念。

(1)格局

在某一时刻，容器中已经互不嵌入地放了若干盒子，还有若干盒子在容器外面等待放入，这种状态，称为一个格局，如图1所示。若容器中还没有放任何盒子，此时的格局称为初始格局。若全部盒子都已互不嵌入地放进了容器或容器中已放不下任何盒子，此时的格局称为终止格局。

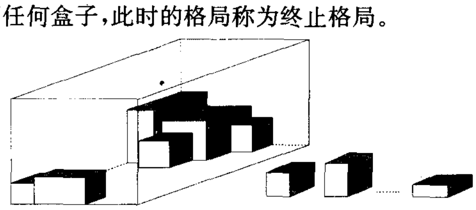

图1 格局

(2)占角动作

在某一格局下，若放进去的盒子B与容器中已有盒子(包括构成容器的6个盒子)中的某三个盒子贴面(即与盒子的面有重叠，并且重叠的面积大于0)，则称放此盒子的这种作法为一个占角动作。相应地，盒子B所在位置为该盒子的一个占角位置。若放进去的盒子B与已有盒子没有嵌入(即与其它盒子重叠的体积等于0)，不超出容器的边界，并且满足方向约束条件，则称这样的占角动作为合法占角动作。在图2中，盒子a,b,c,d,e是已经放置好了的盒子，盒子f在容器外面等待放入。若盒子f放在位置A或B，则放置盒子f的动作就是合法占角动作，放在位置A时，与盒子a,b,c,d贴面，放在位置B时，与盒子b,d,e贴面。

特别地，如果动作盒子(用于作占角动作的盒子)不仅占角，还和形成此角以外的其它已放进容器的盒子贴面，那么就称此占角动作为占穴动作。例如，在图2中，当盒子f放在位置A时，不仅占据了由盒子a,b,c形成的角，还与盒子d贴面，因而盒子f占据了由盒子a,b,c,d形成的穴，相应地，放置盒子f的动作就是一个占穴动作。

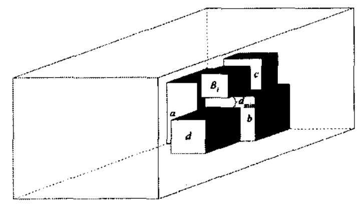

图2 占角动作

(3)两盒子间的距离

对两个给定的盒子B_i和B_j，它们之间的距离d_{ij}定义为

$$d_{ij}=\max(|x_{ci}-x_{cj}|-\frac{L_i+L_j}{2},0)+\max(|y_{ci}-y_{cj}|-\frac{W_i+W_j}{2},0)+\max(|z_{ci}-z_{cj}|-\frac{H_i+H_j}{2},0) \quad (1)$$

其中，$L_i=x_{ri}-x_{li}$，$W_i=y_{ri}-y_{li}$，$H_i=z_{ri}-z_{li}$，$L_j=x_{rj}-x_{lj}$，$W_j=y_{rj}-y_{lj}$，$H_j=z_{rj}-z_{lj}$；(x_{ci},y_{ci},z_{ci})，(x_{li},y_{li},z_{li})和(x_{ri},y_{ri},z_{ri})分别是盒子B_i的中心坐标、前左下角坐标和后右上角坐标；(x_{cj},y_{cj},z_{cj})，(x_{lj},y_{lj},z_{lj})和(x_{rj},y_{rj},z_{rj})分别是盒子B_j的中心坐标、前左下角坐标和后右上角坐标。

事实上，式(1)定义的两盒子间的距离是两点之间曼哈顿距离的推广。例如，在图3a-c中，两盒子之间的距离分别为0，l和$l_1+l_2+l_3$。

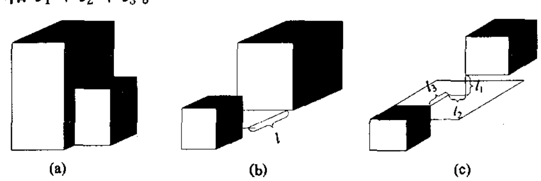

图3 两盒子间的距离

(4)一个盒子和多个其它盒子间的距离

对给定的盒子B和m个盒子组成的集合$\{B_i \mid i=1,2,\cdots,m\}$。令盒子$B$与$B_i$之间的距离为$d_i(i=1,2,\cdots,m)$，那么盒子$B$和$m$个盒子$B_1,B_2,\cdots,B_m$之间的距离定义为$\min(d_1,d_2,\cdots,d_m)$。

(5)占角动作的穴度

如图4所示，若将某一盒子B_i按合法占角动作放进容器之后，盒子B_i与形成这个角以外的其它已放进容器的盒子(包括构成容器的6个盒子)的距离为$d_{\min}$，则此占角动作的穴度C_i定义为

$$C_i=1-\frac{d_{\min}}{\sqrt[3]{l_i \cdot w_i \cdot h_i}} \quad (2)$$

式中，l_i，w_i和h_i分别为盒子B_i的长度、宽度和高度。

事实上，占角动作的穴度反映了即将放进容器的盒子和

距离此盒子最近的盒子(不包括形成此角的盒子 a,b,c)的贴近程度。根据此定义,占角动作的穴度不会大于 1。当作此占角动作的盒子占据了由 4 个或更多个盒子形成的穴时,占角动作的穴度就等于 1。当仅占据由 3 个盒子形成的角时,占角动作的穴度就小于 1。在放置盒子的过程中,我们总是挑选穴度最大的占角动作,并将对应的盒子按相应的方向放在容器中相应的位置上。

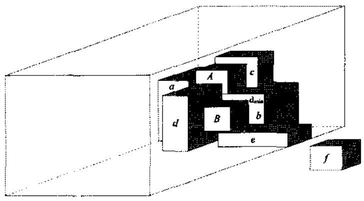

图 4 占角动作的穴度

(6)占角动作的边度

当占角动作所关联的盒子按占角位置放进容器之后,容器中所有盒子(包括当前盒子和先前已放进容器的盒子)之间由于贴面形成的棱和盒子自身未被占据的棱的总数称为此占角动作的边度。例如,当盒子 a 位于位置 A 时,对应占角动作的边度为 24,如图 5a 所示;如果位于位置 B,对应占角动作的边度为 27,如图 5b 所示。

边度越小,意味着放进容器的各盒子形成的布局越规整。因此,在放置过程中,若有多个占角动作的穴度都为最大,就选边度最小的占角动作。例如,在图 5 中,不论盒子 a 放在位置 A 还是位置 B,占角动作的穴度都为 1。但由于放在位置 A 所对应占角动作的边度更小,因此我们选择将盒子 a 放在位置 A 的占角动作。

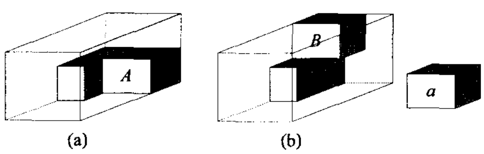

图 5 占角动作的边度

(7)放置方向的优先序

对一个即将放进容器的盒子,最多有 6 个可选的方向,即 $lwh, wlh, lhw, whl, hlw, hwl$。方向 lwh 意味着 $x_r-x_l=l$, $y_r-y_l=w$, $z_r-z_l=h$,而方向 whl 意味着 $x_r-x_l=w$, $y_r-y_l=h$, $z_r-z_l=l$,等等,这里 (x_l,y_l,z_l) 和 (x_r,y_r,z_r) 分别表示盒子前左下角和后右上角的坐标。本文中,放置方向的优先序取为 $lwh>wlh>lhw>whl>hlw>hwl$。

(8)点的优先序

对给定的两点 $P_1(x_1,y_1,z_1)$ 和 $P_2(x_2,y_2,z_2)$,如果满足如下三个条件中的一个,则称点 P_1 的优先序大于点 P_2 的优先序:

① $x_1<x_2$;

② $x_1=x_2$ 且 $y_1<y_2$;

③ $x_1=x_2$ 且 $y_1=y_2$ 且 $z_1<z_2$。

3.3 算法框架

在当前格局之下,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边度。然后选择穴度最大的占角动作并将对应的盒子依相应的方向放在容器中相应的位置上,直到容器外已没有盒子可放或容器内再也不能按放置规则放下任何盒子时为止。

需要注意的是,如果有多个穴度最大的占角动作,那么就按如下的选择原则挑选唯一一个占角动作:

(1) 挑选边度最小的占角动作;

(2) 挑选动作盒子体积最大的占角动作;

(3) 挑选动作盒子前左下角优先序最大的占角动作;

(4) 挑选具有最大方向优先序的占角动作;

(5) 挑选动作盒子编号最小的占角动作。

事实上,上面描述的是一个确定性的纯粹贪心放置办法。为了获得优度更高的解,我们在放置第一个盒子时,引入回溯的策略,将在 4.1 节对其进行详细描述。

4 算法

4.1 算法的伪代码描述

```
Begin
 用 C_best 和 U_best 分别表示当前最好解所对应的格局和容器的积载率,初始时,U_best=0。
  在初始格局 C_0 下,枚举所有的合法占角动作;
  For 每一个合法占角动作 do
  在初始格局 C_0 的基础上,将动作盒子依相应的方向"试放"1) 在容器中相应的位置上,得到新的格局 C_1;
  在格局 C_1 下,枚举所有的合法占角动作;
  While 存在合法的占角动作 do
    计算每一个占角动作的穴度和边度;
    选择穴度最大的占角动作;
    If 存在多个穴度最大的占角动作 then
      按占角动作选择原则挑选唯一一个占角动作(见 3.3 节(1)-(5));
    End if
    对选出的占角动作,将对应的盒子依相应的方向"试放"在容器中相应的位置上,得到新的格局,仍记为 C_1;
    在格局 C_1 下,枚举所有的合法占角动作;
  End while
  计算容器的积载率 U;
  If 所有盒子都放进了容器 then
    C_best⇐C_1,  U_best⇐U;
    退出 for 循环;
  Else if U_best<U then
    C_best⇐C_1, U_best⇐U;
  End if
  将所有"试放"进容器的盒子从容器中拿出来;
 End for
 输出最好解对应的格局,容器积载率和程序的运行时间;
End
```

4.2 计算复杂度分析

每放进一个盒子,将占据一个或多个角区,也将产生一些新角区,角区的数量正比于 n^2。为了向容器中放置一个盒子,需要在每一个角区对所有未放进容器的盒子进行"试放",因而有 $O(n^3)$ 个占角动作供候选。因此在最坏情况下,贪心放置过程(算法描述中的 while 循环)的时间复杂度为 $O(n^3\times n)=O(n^4)$。对整个算法而言,需调用 $O(n^3)$ 次贪心放置过程,因而在最坏情况下,算法的时间复杂度为 $O(n^4\times n^3)=O(n^7)$。

5 实算结果

对本文提出的算法,用 C#.net 语言编程实现,并对 Loh 和 Nee[2] 提出的 15 个测试算例 LN test problems,文献[17]和文献[18]所使用的算例 Test1 和 Test2 在 pentium 4 (256MB RAM/ 2.0GHz CPU)微型机上进行了实算测试,并将实算结果与 CBGAS[5], GRM_{OD} G_{RASP}[7] 以及文献[13,17,18]中的算法进行了比较,比较情况如表 1 和表 2 所示。

5.1 Test1 和 Test2

对算例 Test1,容器大小为 589.9cm * 238.8cm * 235.2

1) 所谓"试放",是指只是将盒子暂时放进容器,将来还会将它从容器右拿出来。

维普资讯 http://www.cqvip.com

cm，共有 30 个待放盒子，算例的具体数据见文献[17]中表 1。对算例 Test2，容器大小为 58dm * 24dm * 24dm，共有 30 个待放盒子，算例的具体数据见文献[18]中表 1。我们将本文算法所得结果的容器积载率和文献[13，17，18]进行了对比，如表 1 所示。对算例 Test1 和 Test2，本文算法所得结果的容器积载率分别为 87.26% 和 92.08%，与文献[13，17，18]所得结果相比，容器积载率有较大幅度的提高。

表 1　各算法在 Test1 和 Test2 上的实算结果对比

算例	文献[17]算法	文献[18]算法	文献[13]算法	本文算法
Test1	85.04%		85.86%	87.26%
Test2		83.3%	87.54%	92.08%

5.2　LN test problems

对算例 LN test problems，共有 15 个测试实例，即 LN01，LN02，…，LN15。对每一个实例，用本文算法求得了未放进容器的盒子个数和容器的积载率，如表 2 的第 6 列和第 7 列所示。我们将本文算法的性能和 CBGAS[5]，GRMODGRASP[7] 两个高效算法进行了对比。CBGAS 所用的计算机为 pentium PC(480MHz CPU)，GRMODGRASP 所用的计算机为 pentium IV(480MB RAM/ 2.4GHz CPU)。CBGAS，GRMODGRASP 和本文算法均可得到其中 13 个实例的最优解，所得结果的平均容器积载率分别为 70.1%，70.3% 和 70.6%，如表 2 所示。实算结果表明，本文算法所得结果比算法 CBGAS，GRMODGRASP 所得结果更优。

表 2　本文算法与 CBGAS，GRMODGRASP 在算例 LN test problems 上的实算结果对比

实例	CBGAS[5] 剩余盒子个数	CBGAS[5] 积载率(%)	GRMODGRASP[7] 剩余盒子个数	GRMODGRASP[7] 积载率(%)	本文算法 剩余盒子个数	本文算法 积载率(%)
LN01	0	62.5	0	62.5	0	62.5
LN02	51	89.8	19	92.6	23	95.5
LN03	0	53.4	0	53.4	0	53.4
LN04	0	55.0	0	55.0	0	55.0
LN05	0	77.2	0	77.2	0	77.2
LN06	45	92.4	28	91.7	35	93.8
LN07	0	84.7	0	84.7	0	84.7
LN08	0	59.4	0	59.4	0	59.4
LN09	0	61.9	0	61.9	0	61.9
LN10	0	67.3	0	67.3	0	67.3
LN11	0	62.2	0	62.2	0	62.2
LN12	0	78.5	0	78.5	0	78.5
LN13	0	85.6	0	85.6	0	85.6
LN14	0	62.8	0	62.8	0	62.8
LN15	0	59.5	0	59.5	0	59.5
平均值		70.1		70.3		70.6

结束语　本文提出了一种求解集装箱装载问题的基于最大穴度优先原则的启发式算法。算法中使用了两个重要的策略：最大穴度原则和最小边度原则。用文献[17，18]所使用的两个算例 Test1 和 Test2 以及 Loh 和 Nee[2] 提出的 15 个测试算例 LN test problems 对算法性能进行了实算测试，对算例 Test1，Test2 和 LN test problems，算法所得的容器积载率分别为 87.26%，92.08% 和 70.6%。实算结果表明，本文提出的算法对求解集装箱装载问题是行之有效的。

参 考 文 献

[1] Scheithauer G. Algorithms for the container loading problem// Operations Research Proceedings. 1991:445-452

[2] Loh T H, Nee A Y C. A packing algorithm for hexahedral boxes // Proceedings of the Conference of Industrial Automation. 1992:115-126

[3] Bischoff E E, Ratcliff M S W. Issues in the development of approaches to container loading. Omega, the International Journal of Management Science, 1995, 23(3): 377-390

[4] 许光泞，俞金寿. 改进遗传算法求解三维集装箱装载问题. 华东理工大学学报(自然科学版)，2007，33(3)：425-428，444

[5] Bortfeldt A, Gehring H. A hybrid genetic algorithm for the container loading problem. European Journal of Operational Research, 2001, 131: 143-161

[6] Hifi M. Approximate algorithms for the container loading problem. International Transactions in Operational Research, 2002, 9: 747-774

[7] Moura A, Oliveira J F. A GRASPapproach to the container-loading problem. IEEE Intelligent Systems, 2005, 20(4): 50-57

[8] George A J, Robinson D F. A heuristic for packing boxes into container. Computers and Operations Research, 1980, 7(3): 147-156

[9] Resende M G C, Pitsoulis L S. Greedy randomized adaptive search procedure. Handbook of Applied Optimization, Oxford University Press, 2002

[10] Gehring H, Bortfeldt A. A parallel genetic algorithm for solving the container loading problem. International Transactions in Operational Research, 2002, 9: 497-511

[11] Bortfeldt A, Gehring H, Mack D. A parallel tabu search algorithm for solving the container loading problem. Parallel Computing, 2003, 29: 641-662

[12] Pisinger D. Heuristics for the container loading problem. European Journal of Operational Research, 2002, 141: 382-392

[13] 杨德荣，杨超. 三维装箱布局的单向寻优搜索法. 工程图学学报，2007(2)：17-22

[14] Wu Y-L, Huang W, Lau S, et al. An effective quasi-human based heuristic for solving the rectangle packing problem. European Journal of Operational Research, 2002, 141: 341-358

[15] Huang W, Li Y, Akeb H, et al. Greedy algorithms for packing unequal circles into a rectangular container. Journal of Operations Research Society, 2005, 56: 539-548

[16] Huang W, Chen D, Xu R. A new heuristic algorithm for rectangle packing. Computers & Operations Research, 2007, 34: 3270-3280

[17] 刘嘉敏，马广焜，黄有群. 基于组合的三维集装箱装入启发式算法的研究. 工程图学学报，2005(1)：22-25

[18] 王若恩，陈锦昌，郭水平. 三维空间最优装载模式的算法研究与实现. 工程图学学报，2005(5)：6-13