软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(5):843−851 [doi: 10.3724/SP.J.1001.2011.03799]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

求解长方体 Packing 问题的捆绑穴度算法*

何 琨, 黄文奇+

(华中科技大学 计算机科学与技术学院,湖北 武汉 430074)

Cuboid Arrangement Approach Based on Caving Degree for Solving the Cuboid Packing Problem

HE Kun, HUANG Wen-Qi+

(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

+ Corresponding author: E-mail: wqhuangwh@gmail.com

He K, Huang WQ. Cuboid arrangement approach based on caving degree for solving the cuboid packing problem. *Journal of Software*, 2011,22(5):843−851. http://www.jos.org.cn/1000-9825/3799.htm

Abstract: Based on the caving degree method, this paper proposes a heuristic approach to solve the cuboid packing problem by incorporating the cuboid arrangement strategy. Experiments on 15 classic LN benchmark instances, performed by Loh and Nee in 1992, have shown that this approach has the potential of performing very well. In the difficult instance of LN2, it achieved a volume utilization of 98.2%, which is an improvement from the current best record by 1.6%. In another difficult instance of LN6, it achieved a volume utilization of 96.2%, which matches that of the current best record. For each of the other 13 instances, it maintains optimal layout that packs all cuboid items into the container, matching current best record. As a whole, the average volume utilization on the 15 LN instances is 70.96%.

Key words: three-dimensional packing; container loading; quasi-human; caving degree; cuboid arrangement

摘 要: 在穴度方法的基础上结合捆绑策略,为三维欧氏空间中长方体 Packing 问题的求解提供了一种高效的启发式算法.试算了由 Loh 和 Nee 于 1992 年提出的 15 个经典算例,对其中的困难算例 LN2,取得了 98.2%的空间利用率,比目前的最好纪录高 1.6 个百分点;对另一个困难算例 LN6,取得了 96.2%的空间利用率,与目前的最好纪录持平;对其他 13 个较为容易的算例均取得了最优的布局,与目前的最好纪录持平.总体而言,15 个算例的平均空间利用率为 70.96%,在整体空间利用率上达到了较好的效果.

关键词: 三维布局;装箱;拟人;穴度;捆绑

中图法分类号: TP301 **文献标识码**: A

Packing 问题(布局问题)属于高复杂度的典型的 NP 难度问题,研究的是一组物体在有限区域内互不重叠的优良布局,即如何设计高性能的求解算法,从而提供高质量的布局方案,以达到节约原材料、缩短施工周期、降低成本、提高效益等目的.1988 年,欧洲运筹学协会在 EURO IX/TIMS XXVII 国际会议上专门成立了 Cutting & Packing 问题兴趣小组 ESICUP(EURO special interest group on cutting and packing)(Cutting 问题其实就是

* 基金项目: 国家自然科学基金(60773194); 中央高校基本科研业务费(HUST 2010MS099)
收稿时间: 2009-06-19; 修改时间: 2009-08-12; 定稿时间: 2009-12-02

Packing 问题的另一种表述)[1],体现出国际学术界对 Packing 问题的高度重视.

Dyckhoff 于 1990 年提出了对 Packing 问题的一种很好的分类方法[2].随后,Wäscher 于 2007 年发表了改进的分类方法[3],用以解决随着 Packing 问题研究的发展,原有分类中逐渐暴露出的问题,并力求使新的分类更具一般性.Wäscher 分类的主要依据是:根据布局空间的维度,可分为一维(1D)、二维(2D)、三维(3D)和高维(*n*D)这 4 类;根据分配类型,可分为对所有容器选择放入的物体和对所有物体选择放入的容器两类;根据待放物体分类,可分为同构(待放物体尺寸完全相同)、弱异构(待放物体可根据尺寸分为若干较少的类型)和强异构(待放物体的尺寸很少相同)这 3 类;根据布局空间类型,可分为一个容器(尺寸固定,或有一至多条边的长度不固定)和多个容器(同构、弱异构、强异构)两大类;根据待放物体形状,可分为规则形状(矩形、圆形、圆柱形…)和不规则形状两大类.

对于待放物体的形状而言,国内外以对矩形(长方形、长方体)物体的布局研究为主[4].据 Sweeney 等人的统计[5],1940 年~1992 年间,相关的研究文献达 400 余篇,涉及 140 多个应用领域.其中,涉及非矩形件的只有 27 篇,涉及三维非矩形件的仅有 3 篇.这是由于矩形物体的布局问题在现实生活中的应用非常广泛,包括钢铁和木材的切割、集装箱的装入、轮船的装货、超大规模集成电路(VLSI)的设计、计算机内存空间的分配等等.

对于布局空间的维度而言,维度越高,问题的求解越困难,而即使一维 Packing 问题(即待放 item 的价值与大小相等的 KNAPSACK 问题)的求解也是 NP 难度的[6].

本文研究长方体 Packing 问题,即在三维欧氏空间中,已知一个长、宽、高任意给定的长方体容器和有穷个长、宽、高分别任意给定的长方体木块,问如何将这些木块尽可能多地放入到容器中,使放入木块的总体积最大?要求放入的木块完全在容器内,棱平行于容器的棱,且木块间两两互不嵌入.作为问题的解,应指出哪些长方体被放入容器中,哪些长方体被留在容器外;并对每一个放入容器的长方体指明其在容器中的方向和位置.问题的优秀解对应着容器的空间得到了充分的利用.算法的性能指标为容器空间利用率的大小和实际计算时间的长短.另外,对于某些具体问题,还需考虑方向约束,比如,某些木块只能竖放、不能横放侧放等等.

长方体 Packing 问题需要在长、宽、高 3 个方向上综合考虑,是一个具有复杂约束条件的组合优化问题.它是三维 Packing 问题中最重要、最典型的问题之一,比一维、二维 Packing 问题要复杂得多,求解也要困难得多.另外,长方体 Packing 问题的高效求解对于求解一维、二维 Packing 问题以及其他三维 Packing 问题也具有借鉴意义.

目前,求解长方体 Packing 问题的方法包括砌墙(分层)法[7−12]、块排列法[13,14]、空间表达法[15]、堆构造法[16]、穴度法[17]、动态空间分解法[18]、序列三元组法[19]等.在此基础上,部分研究者结合遗传[9,16]、退火[12]、禁忌[13]、回溯[10,14,17]、随机搜索[11]等方法进一步改进算法的性能.目前,大部分方法多采用一种常规的途径,按照人们实际生活中可能的放置顺序进行布局,将物体从容器底部开始逐渐向上摆放、逐层邻接放入.当算例的异构性增强即不同尺寸待放长方体的类型越来越多时,算法的求解质量下降.

基于求解二维矩形 Packing 问题的穴度方法[20],我们提出了求解三维长方体 Packing 问题的穴度方法[17],每一步挑选一个长方体放入到容器内的一个“角”甚至是“穴”的位置,使该长方体与其他已放入的长方体或容器的壁尽量压紧.穴度方法采用的是一种反常规的,但却是抓住了事物更深刻本质的布局方法,并不一定逐层地放入物体,甚至可能先将容器的前后上下左右的空间填满,最后再填充容器的中心部分.算例的异构性越强,可选的占角动作越多,从而更有利于对强异构型算例的求解[17].但是对于同构或弱异构型的算例,由于可选的占角动作有限,穴度算法的求解质量并不够高.本文结合捆绑策略即块排列方法,将相同尺寸的长方体捆绑为一个更大的长方体块进行占角甚至占穴动作的试探,算例的同构性越强,可能的捆绑方案越多,越有可能找到贴切的动作来做,从而能够提高算法对同构或弱异构型算例的求解质量.

1 占角动作和穴度

在穴度方法中,我们提出了占角动作和穴度的概念.本节结合捆绑策略,重新定义占角动作和穴度等概念.

1.1 长方体块

定义 1(长方体块). 长方体块是指由同一类型的若干长方体,即 3 边边长和合法的放置方向均相同的若干长方体,按同一方向捆绑而成的一个大长方体块.组成该块的长方体之间没有任何空隙,即该长方体块的空间利用率为 100%.

一般而言,相同类型的长方体可合法放置的方向集也相同.

对 n 个尺寸为(l,w,h)的长方体,可以选用 1~n 个长方体进行捆绑以构成一个大的长方体块.表 1 给出了 8 个相同尺寸长方体的各种可能的捆绑方案.其中,第 1 列为组成该块的长方体的数目,即捆绑数 n_i;第 2 列为这些长方体可能组成块的不同方式.其中,n_l,n_w,n_h 表示捆绑到 l,w,h 方向的长方体数.显然,捆绑数 $n_i=n_l\times n_w\times n_h$.

Table 1 Different cuboid arrangements for 8 cuboids in the same size

表 1 8 个相同类型长方体的各种捆绑方案

n_i	$[n_l,n_w,n_h]$
1	[1,1,1]
2	[1,1,2], [1,2,1], [2,1,1]
3	[1,1,3], [1,3,1], [3,1,1]
4	[1,1,4], [1,2,2], [1,4,1], [2,1,2], [2,2,1], [4,1,1]
5	[1,1,5], [1,5,1], [5,1,1]
6	[1,1,6], [1,2,3], [1,3,2], [1,6,1], [2,1,3], [2,3,1], [3,1,2], [3,2,1], [6,1,1]
7	[1,1,7], [1,7,1], [7,1,1]
8	[1,1,8], [1,2,4], [1,4,2], [1,8,1], [2,1,4], [2,2,2], [2,4,1], [4,1,2], [4,2,1], [8,1,1]

1.2 占角动作

定义 2(格局). 设某一时刻容器内已放入若干长方体块,还有若干长方体在容器外待放,称为一个格局.当容器内尚未放入任何长方体块时,称为初始格局.容器内每增加一个长方体块,就形成一个新的格局.当所有长方体已放入容器内,或容器外剩余的长方体无法再放入时,称为终止格局.

定义 3(角区). 角区是由 3 个属于不同长方体块的平面相交于一点所围成的空闲空间.

容器的 6 个壁可视为 6 个特殊的、扁平的长方体块.因此,初始格局由 6 个特殊的长方体块和 8 个它们所围成的角区构成.

定义 3(占角动作). 当前格局下,将容器外若干相同类型的长方体组成一个长方体块,并放入到一个角区(该长方体块的某个顶点与角区的顶点重合,某 3 个面分别与角区的 3 个面重合且重合面积大于 0),动作长方体块不与任何已放入的长方体块嵌入,则称此动作为一个占角动作(corner-occupying action,简称 COA).称该动作长方体块为 COA 块.

因此,一个 COA 包含以下 5 个要素:

(1) 选择容器外哪种类型的长方体;

(2) 选择容器外多少个该类型的长方体进行捆绑;

(3) 对给定数量、给定类型的若干长方体,选择哪一种捆绑方案;

(4) 选择哪个角区放入;

(5) 选择哪个放置方向放入.

上述第 1 个要素确定了放入长方体的尺寸 l,w 和 h,第 2 个要素确定了捆绑数 n_i,第 3 个要素确定了捆绑到 l,w,h 方向的长方体数 n_l,n_w 和 n_h,这 3 个要素合起来确定了选择哪一个大的长方体块.第 4 个和第 5 个要素则确定了该长方体块的放入位置和放置方向.

可对每个长方体的不同放置方向定义一个方向数,如图 1 中右图所示.用(x_{i1},y_{i1},z_{i1})和(x_{i2},y_{i2},z_{i2})表示放入块 i 的左前下顶点坐标和右后上顶点坐标,其方向数即为组成块的长方体的方向数.在图 1 中,一个 COA 将由 3 个长方体捆绑而成的大长方体块 i 放入到了容器的右前上角,其$[n_l,n_w,n_h]$为[1,1,3],方向数为 4.

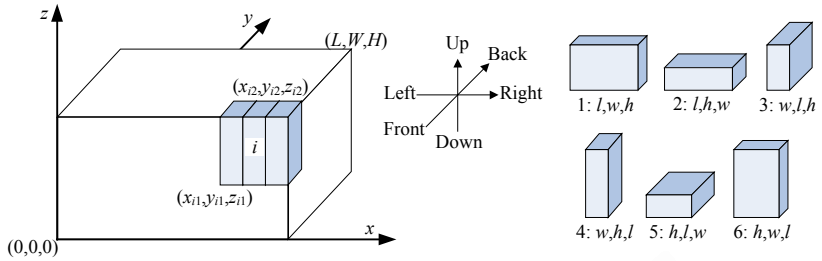

Fig. 1 An example of corner-occupying action

图 1 一个占角动作的示例

1.3 穴 度

相应地,穴度的定义也需作一定的调整.首先重新给出 COA 距离 d_i 的定义.

定义 4(COA 距离). 贴着做占角动作的长方体块的 k_i 个平面及其法方向(指向动作长方体块所在空间)指明了一个看似 $6–k_i$ 维的空间Ω,在此空间中可定义该 COA 的距离.令 B 表示与Ω重合的体积或面积大于 0 但不与动作长方体 i 相贴的长方体块的集合,则 COA 距离定义为 $\min\{d_{ij}|j\in B\}$.当 k_i=6 时,令 d_i=0.

定义 5(穴度). 占角动作的穴度 C_i 如公式(1)所示.它首先由其贴面数 k_i 决定,即有多少个长方体块的平面与该 COA 块相贴;然后由邻近度 ad_i 决定,即由 COA 距离 d_i、该 COA 块的 3 边边长和捆绑数 n_i 决定;最后由贴面率 r_i,即该 COA 块被贴的面积比率决定.

$$C_i=100\cdot k_i+10\cdot ad_i+r_i=100\cdot k_i+10\cdot\exp\left(\frac{-d_i}{\sqrt[3]{L_iW_iH_i}}\cdot n_i\right)+r_i \tag{1}$$

在新的穴度定义中,其贴面数 k_i、COA 距离 d_i、3 边边长 L_i 和 W_i 及 H_i、贴面率 r_i 的定义均与穴度算法的定义相似但又不同.这里观察的对象单元是长方体块,而不是单个的长方体.单个的长方体只是一类特殊的长方体块.

因其贴面数 $k_i\geqslant 3$,邻近度 $0<ad_i\leqslant 1$,贴面率 $0<r_i\leqslant 1$,系数 100,10 和 1 使穴度首先由其贴面数决定,然后由邻近度决定,最后由贴面率决定.穴度 C_i 的值越大,与该 COA 块相贴的其他长方体块越多,该 COA 块与其他已放入块的距离越近,被其他放入块贴住的表面积越大,因此 COA 块放入的位置越像一个“穴”.长方体块放入的位置越像“穴”,该长方体块与其他已放入块的关系就越紧密.

当两个 COA 的它贴面数 k_i 和 COA 距离 d_i 相同时,有着较大平均尺寸的长方体块有较大的穴度.因此,我们加入捆绑数 n_i,用于调整 COA 距离 d_i 与长方体块平均尺寸的比值,降低由大量小长方体捆绑而成的块的放入优先级.例如,若两个 COA 有相同的它贴面数 k_i、COA 距离 d_i 和尺寸(L_i,W_i,H_i),则由较少长方体组成的块有更大的穴度.这有利于将体积较大的、较难放入的长方体优先放入容器.

2 捆绑穴度算法

捆绑穴度算法(cuboid arrangement algorithm based on caving degree,简称 CACD 算法)将相同尺寸的长方体捆绑为一个更大的长方体块,然后每一步挑选一个长方体块以指定的方向(横、竖、侧放等)放入容器中的一个“角”甚至是“穴”的位置,使得放入块与其他已放入的大长方体块或容器的壁尽量压紧,即在容器中占据了当前被利用机会最小的空间部分.

2.1 基本算法

基本算法 $CACD_0$ 的执行过程是从当前格局开始,每一步挑选一个穴度最大的占角动作来做,往容器内放入一个长方体块,从而形成新的格局,如此反复,直到终止格局.

基本算法 $CACD_0$ 可描述如下:

算法 1. $CACD_0$.

1. 从当前格局出发,对每个角区{
   对待放长方体的每种类型{
   对该类待放长方体的每种捆绑方案所构成的大长方体块{
   对每个合法的放置方向{
   将该长方体块暂时以当前指定的方向放入当前角区;
   若动作合法,则利用公式(1)计算其穴度.
   }}}}
2. 依字典序按以下优先规则挑选一个最好的 COA 来做:
   (1) COA 块之穴度:大优先.
   (2) COA 块之体积:大优先.
   (3) COA 块之长边长:大优先.
   (4) COA 块之短边长:大优先.
   (5) COA 块的 x_{i1},y_{i1},z_{i1} 中与容器长边对应之坐标:小优先.
   (6) COA 块的 x_{i1},y_{i1},z_{i1} 中与容器中边对应之坐标:小优先.
   (7) COA 块的 x_{i1},y_{i1},z_{i1} 中与容器短边对应之坐标:小优先.
   (8) COA 块之方向数:小优先.
   (9) 构成 COA 块的小长方体之体积:大优先.
   (10) 构成 COA 块的小长方体之长边长:大优先.
   (11) 构成 COA 块的小长方体之短边长:大优先.
3. 更新角区列表和待放长方体列表.
4. 重复步骤 1~步骤 3,直至终止格局.

第 2 步的规则(1)选择一个穴度最大的占角动作来做.规则(2)~规则(4)确定了放入块的尺寸,即 COA 的第 3 个要素,使得有着较大尺寸的长方体块的 COA 被优先考虑.规则(5)~规则(8)确定了大长方体块的放入位置和放置方向,即 COA 的第 4 个和第 5 个要素.规则(9)~规则(11)确定了构成大长方体块的小长方体的捆绑数量和尺寸,即 COA 的第 1 个和第 2 个要素.当穴度最大的 COA 有多个时,规则(2)~规则(11)可唯一地确定一个占角动作.

2.2 回溯策略

在 $CACD_0$ 的基础上加入回溯策略,得到增强算法 $CACD_1$.$CACD_1$ 每一步对穴度较大的若干个占角动作做进一步的观察,将做此动作后若执行基本算法 $CACD_0$ 可得到的空间利用率作为预期结果,然后从中选择一个预期结果最好的动作来做.

算法 2. $CACD_1$.

1. 从当前格局出发,对每个角区{
   对待放长方体的每种类型{
   对该类待放长方体的每种捆绑方案所构成的长方体块{
   对每个合法的放置方向{
   将该长方体块暂时以当前指定的方向放入当前角区;
   若动作合法,则利用公式(1)计算其穴度.
   }}}}
2. 用 $CACD_0$ 的优先级规则对所有合法的 COA 排序,对排在前面的 N 个 COA,将做此动作后若执行基本算法 $CACD_0$ 可得到的空间利用率作为预期结果.
3. 选择预期结果最好的 COA 来做,若出现僵局,则选择排在前面的 COA.
4. 更新角区列表和待放长方体列表.
5. 重复步骤 1~步骤 4,直至终止格局.

若参数 N 设为 1,则算法 $CACD_1$ 退化为 $CACD_0$.参数 N 的值越大,计算所需的时间越长,可能得到的空间利用率越高.本文的实验部分将参数 N 设为 20,以期在时间和优度上获得较好的平衡.对于待放长方体两两不同的

强异构的极端情形,捆绑穴度算法退化为纯粹的穴度算法.

3 实验结果

我们将捆绑穴度算法 $CACD_1$ 用 Java 标准版 V 1.6.0_03 编程,并在 Intel(R) Xeon(R) 2.33 GHz 处理器和 1GB 内存配置的计算机上进行了实例计算.

3.1 算例说明

Loh 和 Nee 于 1992 年提出了 15 个算例[21],简称为 LN 算例.它们被收录到著名算例库 OR-Library[22]中.这组算例的特点是:(1) 待放长方体高方向的边固定,只可以在长和宽的方向上作 90°的旋转.在实际应用中存在这样的约束,例如待放的长方体盒子内是装满液体的瓶子;(2) 待放长方体的类型有 6~10 种,为弱异构型的算例;(3) 待放长方体的总体积小于容器的体积;(4) 其中 13 个算例的最优解已知,可以将待放长方体全部装入,而另外两个困难算例 LN2 和 LN6 的最优解未知.

3.2 实验结果与讨论

3.2.1 空间利用率的比较

目前,计算了 LN 算例的高效算法包括 H_N[15],H_BR[7],H_B_*al*[8],GA_GB[16],TS_BG[13],HGA_BG[9],H_E[14],H_L_*al*[10],GRASP[11]和 H_W_*al*[18].它们与 $CACD_1$ 算法的计算结果比较见表 2.目前报道的较好结果是 Gehring 和 Bortfeldt 于 1998 年提出的 TS_BG 算法,其平均空间利用率为 70.85%.

Table 2 Computational results on the LN benchmarks (%)

表 2 LN 算例的计算结果比较(%)

Problem	H_N (1994)	H_BR (1995)	H_B_*al* (1995)	GA_GB (1997)	TS_BG (1998)	HGA_BG (2001)	H_E (2002)	H_L_*al* (2005)	GRASP (2005)	H_W_*al* (2008)	$CACD_1$ (2009)
LN1	62.5	62.5	62.5	62.5	62.5	62.5	62.5	62.5	62.5	62.5	62.5
LN2	80.7	90.0	89.7	89.5	96.6	89.8	90.8	80.4	92.6	90.7	**98.2**
LN3	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4
LN4	55.0	55.0	55.0	55.0	55.0	55.0	55.0	55.0	55.0	55.0	55.0
LN5	77.2	77.2	77.2	77.2	77.2	77.2	77.2	76.7	77.2	77.2	77.2
LN6	88.7	83.1	89.5	91.1	96.2	92.4	87.9	84.8	91.7	92.9	96.2
LN7	81.8	78.7	83.9	83.3	84.7	84.7	84.7	77.0	84.7	84.7	84.7
LN8	59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.4
LN9	61.9	61.9	61.9	61.9	61.9	61.9	61.9	61.9	61.9	61.9	61.9
LN10	67.3	67.3	67.3	67.3	67.3	67.3	67.3	67.3	67.3	67.3	67.3
LN11	62.2	62.2	62.2	62.2	62.2	62.2	62.2	62.2	62.2	62.2	62.2
LN12	78.5	78.5	76.5	78.5	78.5	78.5	78.5	69.5	78.5	78.5	78.5
LN13	84.1	78.1	82.3	85.6	85.6	85.6	85.6	73.3	85.6	85.6	85.6
LN14	62.8	62.8	62.8	62.8	62.8	62.8	62.8	62.8	62.8	62.8	62.8
LN15	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5
AVG	69.00	68.64	69.54	69.95	70.85	70.15	69.91	67.05	70.29	70.24	**70.96**

实验结果表明:(1) 对于困难算例 LN2,$CACD_1$ 取得了 98.2%的空间利用率,比目前的最好纪录高 1.6 个百分点;(2) 对于困难算例 LN6,$CACD_1$ 取得了 96.2%的空间利用率,与目前的最好纪录持平;(3) 对于其他的 13 个较容易的 LN 算例,$CACD_1$ 均取得了最优的布局,将待放的长方体全部放入,与目前的最好纪录持平;(4) 由于这 13 个算例的计算结果已经达到最优,不可能再作进一步的提高,受这 13 个算例计算结果的稀释,总体而言,$CACD_1$ 算法对 15 个 LN 算例取得了 70.96%的平均空间利用率,比目前的最好纪录高 0.11 个百分点.

3.2.2 LN2 算例的计算结果

下面详细说明 $CACD_1$ 对 LN2 算例的计算结果.容器的尺寸为(3000,2000,1000)(不妨设单位为米),有 200 个待放的长方体,分为 8 种不同的类型.算法 $CACD_1$ 共装入 154 个长方体,它对每一类长方体的放入数量见表 3.

用 l_x,l_y,l_z 表示组成块的小长方体在 x,y,z 轴方向的长度,用 L_x,L_y,L_z 表示装入大长方体块在 x,y,z 轴方向的长度,用 n_x,n_y,n_z 表示大长方体块在 x,y,z 轴方向捆绑的长方体数目,表 4 给出了 LN2 详细的布局说明.$CACD_1$ 一共放入了 15 个大长方体块,其中每个长方体块由 1~20 个同类型的长方体捆绑而成.因 154 个长方体的布局图案

比较琐碎,我们给出由它们捆绑而成的 15 个大长方体块的布局图案,如图 2 所示.

Table 3 LN2 benchmark

表 3 算例 LN2

Type	*l*,*w*,*h*	Number	Packing number	Outside number
1	400,375,250	29	28	1
2	400,400,150	37	34	3
3	300,300,200	34	22	11
4	500,375,400	19	18	1
5	800,275,200	16	13	3
6	450,350,350	17	16	1
7	900,200,200	25	23	2
8	200,200,125	23	0	23
Total	–	200	154	46

Table 4 Packing layout of $CACD_1$ for LN2

表 4 $CACD_1$ 对算例 LN2 的布局说明

No.	Type	l_x,l_y,l_z	$n_i=n_x\times n_y\times n_z$	L_x,L_y,L_z	(x_{i1},y_{i1},z_{i1})	(x_{i2},y_{i2},z_{i2})
1	4	500,375,400	12=6×1×2	3000,375,800	(0,1625,200)	(3000,2000,1000)
2	3	300,300,200	10=10×1×1	3000,300,200	(0,1700,0)	(3000,2000,200)
3	7	200,900,200	15=15×1×1	3000,900,200	(0,800,0)	(3000,1700,200)
4	1	375,400,250	16=8×2×1	3000,800,250	(0,0,0)	(3000,800,250)
5	5	800,275,200	12=1×3×4	800,825,800	(0,800,200)	(800,1625,1000)
6	1	375,400,250	6=1×2×3	375,800,750	(0,0,250)	(375,800,1000)
7	2	400,400,150	20=1×4×5	400,1600,750	(800,25,250)	(1200,1625,1000)
8	2	400,400,150	10=1×2×5	400,800,750	(375,0,250)	(775,800,1000)
9	7	900,200,200	8=2×1×4	1800,200,800	(1200,1425,200)	(3000,1625,1000)
10	6	450,350,350	16=4×4×1	1800,1400,350	(1200,25,650)	(3000,1425,1000)
11	3	300,300,200	12=6×1×2	1800,300,400	(1200,1125,250)	(3000,1425,650)
12	4	500,375,400	6=2×3×1	1000,1125,400	(2000,0,250)	(3000,1125,650)
13	1	400,375,250	6=2×3×1	800,1125,250	(1200,0,400)	(2000,1125,650)
14	5	800,275,200	1=1×1×1	800,275,200	(1200,850,200)	(2000,1125,400)
15	2	400,400,150	4=2×2×1	800,800,150	(1200,0,250)	(2000,800,400)

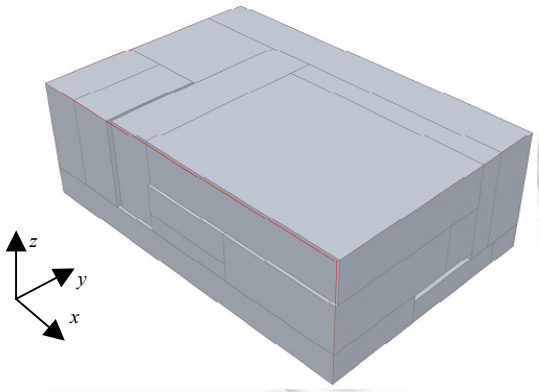

Fig.2 Packing layout build by cuboid blocks for the LN2 benchmark

图 2 LN2 算例的由长方体块组成的布局图案

3.2.3 计算时间的比较

H_BR 算法[7]和 H_L_*al* 算法[10]未给出计算时间的说明.H_B_*al* 算法[8]在 33MHz 的 PC 机上的计算时间为 2 秒(若待放长方体的类型有 3 种)~90 秒(若待放长方体的类型有 20 种),因 LN 算例待放长方体的类型有 6 到 10 种,故 H_B_*al* 算法对 LN 算例的计算时间约为 20 秒~30 秒.HGA_BG 算法[9]在 400MHz 的 PC 机上进行计算,对每个算例均设置了 500 秒的时间限制,但未报导具体的计算时间.H_E 算法[14]在 200 MHz 的 PC 机上进行计算,对每个算例的计算时间不超过 4 分钟.H_N 算法[15]在 33MHz 的 PC 机上进行计算,仅给出放入 200 个长方体

所需的时间为 1.4 分钟.GA_GB 算法[16]在 130MHz 的 PC 机上进行计算,对每个算例的计算时间不超过 4 分钟,但未报导具体的计算时间.H_W_*al* 算法[18]在一台奔腾 PC 机上运行,对每个算例的计算时间不超过 20 秒,但未报导该 PC 机的具体计算速度和每个算例的具体计算时间.

目前取得最好结果的 TS_BG 算法[13]是在 200MHz 的 PC 机上进行计算,对每个算例均设置了 500 秒的时间限制,但未报导具体的计算时间.目前取得第二好结果的 GRASP 算法[11]在 2.4GHz 的计算机上的平均耗时为 8.7 秒.$CACD_1$ 算法在 2.33GHz 的计算机上的平均耗时为 549.2 秒,即 9.2 分钟.尽管 $CACD_1$ 算法的计算时间是 GRASP 算法的几十倍,但这仍然是比较合理的.而且我们认为,相对于计算时间的长短,如何取得高的空间利用率显得更为重要.

3.2.4 相关算法的分析

下面对目前取得最好结果的 TS_BG 算法[13]和取得第二好结果的 GRASP 算法[11]作进一步的分析.

TS_BG 算法[13]的基本思想是,将整个容器视为一个矩形空间并放入到矩形空间列表中,然后执行以下操作,直至矩形空间列表为空:

(1) 选择一个体积最小的矩形空间,在其底部放入一或两块由同类型长方体沿 *x*,*y* 方向捆绑而成的矩形平板,使其在 *x*,*y* 方向尽量放满该空间的底部.

(2) 将剩余空间分解为若干个两两无交的矩形空间,并更新矩形空间列表.

在步骤(1)中,根据长方体的不同类型和不同放置方向,当前的矩形空间有多个候选的矩形平板可放,这些候选方案按矩形空间底面积被覆盖的大小降序排列,而基本算法总是选择第 1 个方案.在此基础上结合禁忌搜索,搜索的邻域为对某个矩形空间选择其他的候选平板放入,然后继续执行基本算法得到最终的布局.

GRASP 算法[11]的基本思想是,将容器沿 *x* 方向分层,层厚等于某类型长方体按某方向放置时在 *x* 方向的长度,然后竖直放入由该类型长方体沿 *y*,*z* 方向捆绑而成的矩形平板;该层在 *y*,*z* 方向剩余的矩形空间分别选择由其他的同类型长方体在 *x*,*y*,*z* 方向捆绑而成的矩形块填充;若剩余空间不能放入任何长方体,则考虑与相邻层的剩余空间合并.在此基础上结合随机搜索,因矩形平板(块)按放入顺序组成的序列对应问题的一个解,因此其邻域为从该序列中随机取出一个,并将其后放入的所有长方体移出容器;然后从该矩形平板(块)对应空间的利用率较高的若干个填充方案中随机选择一个,接着继续执行基本算法得到最终的布局.

上述两种算法都是沿两个方向捆绑同类型的长方体以构成一个大的矩形平板,然后将它们全部由底向上水平放入或从左至右竖直放入.CACD 算法则是沿 3 个方向捆绑同类型的长方体以构成一个大的长方体块,然后通过穴度来统一比较不同占角动作的优劣;而且,长方体块在放入方向上没有作限制,也不要求逐层邻接放入.因此,CACD 算法在长方体块的构造、块的放入位置和放入方向上均更为灵活.

4 结 论

通过拟人途径,本文深入地研究了长方体 Packing 问题的高效求解算法.从人类社会的各种经验和智慧中吸取养分,并借鉴前人的成功经验,将擅长求解强异构型算例的穴度方法和擅长求解弱异构型算例的块排列法相结合,提出了捆绑穴度算法.利用国际上公开通行的 LN 算例集,与当前国内外文献中的代表性求解算法进行了比较.实验结果表明,本文提出的拟人型启发式算法具有较高的求解精度.

在下一步的工作中,我们将在保证求解质量的前提下,设法提高算法的计算速度;另外,可以考虑长方体 Packing 问题在实际应用中可能遇到的各种约束,如稳定性约束、重力分布约束、总承重约束等等.

致谢 本研究的实验计算得到了华中科技大学服务计算技术与系统实验室高性能计算中心(http://grid.hust.edu.cn/hpcc)的支持.

References:

[1] ESICUP. EURO special interest group on cutting and packing. http://paginas.fe.up.pt/~esicup/tiki-index.php

[2] Dyckhoff H. A typology of cutting and packing problems. European Journal of Operational Research, 1990,44(2):145−159. [doi: 10.1016/0377-2217(90)90350-K]
[3] Wäscher G, Haubner H, Schumann H. An improved typology of cutting and packing problems. European Journal of Operational Research, 2007,183(3):1109−1130. [doi: 10.1016/j.ejor.2005.12.047]
[4] Cagan J, Shimada K, Yin S. A survey of computational approaches to three-dimensional layout problems. Computer-Aided Design, 2002,34(8):597−611. [doi: 10.1016/S0010-4485(01)00109-9]
[5] Sweeney PE, Paternoster ER. Cutting and packing problems: A categorized, application orientated research. Journal of Operation Research Society, 1992,43(7):691−706.
[6] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: Freeman, 1979.
[7] Bischoff EE, Ratcliff MSW. Issues in the development of approaches to container loading. OMEGA: The Int'l Journal of Management Science, 1995,23(4):377−390. [doi: 10.1016/0305-0483(95)00015-G]
[8] Bischoff EE, Janetz F, Ratcliff MSW. Loading pallets with non-identical items. European Journal of Operational Research, 1995, 84(3):681−692. [doi: 10.1016/0377-2217(95)00031-K]
[9] Bortfeldt A, Gehring H. A hybrid genetic algorithm for the container loading problem. European Journal of Operational Research, 2001,131(1):143−161. [doi: 10.1016/S0377-2217(00)00055-2]
[10] Lim A, Rodrigues B, Yang Y. 3-D container packing heuristics. Applied Intelligence, 2005,22:125−134. [doi: 10.1007/s10489-005-5601-0]
[11] Moura A, Oliveira JF. A GRASP approach to the container-loading problem. IEEE Intelligent Systems, 2005,20(4):50−57. [doi: 10.1109/MIS.2005.57]
[12] Zhang DF, Wei LJ, Chen QS, Chen HW. A combinational heuristic algorithm for the three-dimensional packing problem. Journal of Software, 2007,18(9):2083−2089 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2083.htm [doi: 10.1360/jos182083]
[13] Bortfeldt A, Gehring H. A Tabu search algorithm for weakly heterogeneous container loading problems. Operations Research Spektrum, 1998,20:237−250 (in German with English abstract). [doi: 10.1007/BF01539742]
[14] Eley M. Solving container loading problems by block arrangement. European Journal of Operational Research, 2002,141(2): 393−409. [doi: 10.1016/S0377-2217(02)00133-9]
[15] Ngoi BKA, Tay ML, Chua ES. Applying spatial representation techniques to the container packing problem. Int'l Journal of Production Research, 1994,32(1):111−123. [doi: 10.1080/00207549408956919]
[16] Gehring H, Bortfeldt A. A genetic algorithm for solving the container loading problem. Int'l Trans. in Operational Research, 1997, 4(5-6):401−418. [doi: 10.1111/j.1475-3995.1997.tb00095.x]
[17] Huang WQ, He K. A pure quasi-human algorithm for solving the cuboid packing problem. Science in China (Series F), 2009,52(1): 52−58. [doi: 10.1007/s11432-009-0005-0]
[18] Wang ZJ, Li, KW, Levy JK. A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach. European Journal of Operational Research, 2008,191(1):86−99. [doi: 10.1016/j.ejor.2007.08.017]
[19] Lu YP, Zha JZ. Sequence triplet method for 3D rectangle packing problem. Journal of Software, 2002,13(11):2183−2187 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/2183.htm
[20] Huang WQ, Liu JF. A deterministic heuristic algorithm based on Euclidian distance for solving the rectangles packing problem. Chinese Journal of Computers, 2006,29(5):734−739 (in Chinese with English abstract).
[21] Loh TH, Nee AYC. A packing algorithm for hexahedral boxes. In: Proc. of the Conf. of Industrial Automation. Singapore: Industrial Automation Association, 1992. 115−126.
[22] Beasley JE. OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society, 1990,41(11): 1069−1072.

附中文参考文献:

[12] 张德富,魏丽军,陈青山,陈火旺.三维装箱问题的组合启发式算法.软件学报,2007,18(9):2083−2089. http://www.jos.org.cn/1000-9825/18/2083.htm [doi: 10.1360/jos182083]
[19] 陆一平,查建中.三维矩形块布局的序列三元组编码方法.软件学报,2002,13(11):2183−2187. http://www.jos.org.cn/1000-9825/13/2183.htm
[20] 黄文奇,刘景发.基于欧氏距离的矩形 Packing 问题的确定性启发式求解算法.计算机学报,2006,29(5):734−739.

何琨(1972－),女,北京人,博士,副教授,CCF 会员,主要研究领域为 NP 难度问题现实求解,算法优化.

黄文奇(1938－),男,教授,博士生导师,主要研究领域为 NP 难度问题现实求解,算法优化.