求解三维装箱问题的多层树搜索算法

刘 胜 [1, 2, 3] 沈大勇 [4] 商秀芹 [1, 2, 3] 赵红霞 [1, 2, 3] 董西松 [1, 2, 3] 王飞跃 [1, 2, 3]

摘 要 提出了一种求解三维装箱问题的多层树搜索算法, 该算法采用箱子–片–条–层–实体的顺序生成装载方案, 装载方案由实体表示. 该算法由 3 层搜索树构成. 第 1 层为三叉树, 每个树节点的 3 个分叉分别对应向实体中填入 XY 面平行层、XZ 面平行层、YZ 面平行层; 第 2 层为二叉树, 每个树节点的两个分叉分别对应向层内装载两个相互垂直的最优条; 第 3 层为四叉树, 用于将同种的多个箱子生成片. 在同时满足摆放方向约束和完全支撑约束的前提下, 该算法求解 BR12~BR15 得到的填充率高于现有装箱算法.

关键词 三维装箱, 墙构造, 水平层构造, 多层树

引用格式 刘胜, 沈大勇, 商秀芹, 赵红霞, 董西松, 王飞跃. 求解三维装箱问题的多层树搜索算法. 自动化学报, 2020, **46**(6): 1178−1187

DOI 10.16383/j.aas.c180795

A Multi-level Tree Search Algorithm for Three Dimensional Container Loading Problem

LIU Sheng[1, 2, 3] SHEN Da-Yong[4] SHANG Xiu-Qin[1, 2, 3] ZHAO Hong-Xia[1, 2, 3]
DONG Xi-Song[1, 2, 3] WANG Fei-Yue[1, 2, 3]

Abstract This paper presents a multiple-level tree search algorithm for the three dimensional container loading problem. This algorithm generates a loading plan in the box-piece-strip-layer-entity sequence. Hereby an entity denotes a loading plan. This algorithm consists of three levels of search tree. The first level is a ternary tree. In this tree the three branches of each node correspond to filling three layers which are parallel to the XY-plane, the XZ-plane, and the YZ-plane, respectively. The second level is a binary tree. In this tree the two branches of each node correspond to loading two orthogonal strips into a layer, respectively. The third level is a quad tree to search the best piece. In this tree the boxes of the same kind are integrated into a piece. The proposed algorithm achieves better filling rate for BR12~BR15 than the existing algorithms when the orientation constraint and the full-support constraint are satisfied.

Key words Three-dimensional container loading, wall building, horizontal layer building, multi-level tree search

Citation Liu Sheng, Shen Da-Yong, Shang Xiu-Qin, Zhao Hong-Xia, Dong Xi-Song, Wang Fei-Yue. A multi-level tree search algorithm for three dimensional container loading problem. *Acta Automatica Sinica*, 2020, **46**(6): 1178−1187

收稿日期 2018-11-29 录用日期 2019-03-25
Manuscript received November 29, 2018; accepted March 25, 2019
国家重点研究发展计划基金 (2018YFB1004803), 国家自然科学基金 (61773381, 61773382, 61533019), 湖南省科技重大专项基金 (2018GK1040), 广东省科技厅项目基金 (2017B090912001) 资助
Supported by National Research and Development Program of China (2018YFB1004803), National Natural Science Foundation of China (61773381, 61773382, 61533019), Hunan Science and Technology Project (2018GK1040), and Guangdong Science and Technology Project (2017B090912001)
本文责任编委 李力
Recommended by Associate Editor LI Li

1. 中国科学院自动化研究所复杂系统管理与控制国家重点实验室 北京 100190 2. 中国科学院自动化研究所北京市智能化技术与系统工程技术研究中心 北京 100190 3. 青岛智能产业技术研究院 青岛 266109 4. 国防科技大学系统工程学院 长沙 410073
1. State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190 2. Beijing Engineering Research Center of Intelligent Systems and Technology, Institute of Automation, Chinese Academy of Sciences, Beijing 100190 3. Qingdao Academy of Intelligent Industries, Qingdao 266109 4. College of Systems Engineering, National University of Defense Technology, Changsha 410073

三维装箱问题 (Three-dimensional container loading problem, 3D-CLP) 研究如何通过优化货物摆放来提高货箱空间利用率, 应用三维装箱算法不仅可以降低货物运输成本, 还可以减少运输工具在运输过程中的碳排放, 具有很好的社会经济效益.

三维装箱问题属于切割与布局问题 (Cutting and packing, CP)[1], 相关研究可以追溯到 20 世纪 80 年代[2]. 按照文献 [3] 的分类命名法, 三维装箱问题可以归为代号为 3/V/I 或 3/B/O 的 CP 问题. 按照文献 [4] 建议的分类命名法, 三维装箱问题归为三维单一背包问题 (Three-dimensional single knapsack problem, 3D-SKP) 或三维单一大物体摆放问题 (Three-dimensional single large object placement problem, 3DSLOPP). 依据文献 [5] 中定义, 本文研究的是考虑摆放方向约束 (C1) 和完全支撑约束 (C2) 的三维装箱问题. 在目前研究三

维装箱问题的文献中, 这也是考虑最广泛的两个约束.

本文结构如下: 第 1 节简要回顾三维装箱问题的研究进展; 第 2 节详细叙述本文提出的三维装箱算法的实现步骤; 第 3 节给出本文算法运行通用算例的结果, 并将结果与目前最新算法的运行结果相比较. 最后第 4 节总结了本文算法的特点, 并展望了下一步的研究思路.

1 研究综述

3D-CLP 是典型的 NP 难题[6−7], 不存在多项式时间复杂度的最优求解算法. 用传统的精确算法求解这类问题, 会发生 “组合爆炸” 的现象, 只适合解决箱子种类数量较小的装箱问题, Fekete 等[8] 给出了一种求解该问题的精确方法. 对于实际应用中规模比较大的装箱问题, 精确方法难以在可接受时间内求得最优解, 启发式方法依然是首选. 近年来有大量文献报道求解 3D-CLP 的启发式算法. 按照方法类型来分, 这些算法可以分为以下 3 种:

1) 经典启发式方法. 此类方法分为两大类, 一类是构造法, 即从一个空的方案逐步生成一个完全的装载方案, 文献 [9−13] 提出的方法可以归为此类方法, 文献 [14] 利用多面构建的方法, 显著提升了填充率, 文献 [15] 将拟人法与模拟退火算法相结合, 取得了很好的效果; 另一类是提升法, 即首先生成一个完全的装载方案, 然后不断地优化提升该方案.

2) 元启发式方法. 元启发式方法包括遗传算法、禁忌搜索、模拟退火、贪婪随机自适应等方法, 近年来在求解 3D-CLP 中得到广泛应用. 文献 [15−17] 利用遗传算法求解 3D-CLP. 文献 [18−19] 分别给出了一种求解 3D-CLP 的禁忌搜索方法. 文献 [15, 18, 20−21] 将模拟退火算法应用于求解 3D-CLP, 文献 [22] 给出了基于 Nelder-Mead 方法的局部搜索方法, 用以提升 3D-CLP 的解的质量. 文献 [23−24] 给出求解 3D-CLP 的贪婪随机自适应搜索方法 (Greedy randomized adaptive search procedure, GRASP).

3) 树搜索法. 树搜索或图搜索方法用于求解三维装箱问题, 形成了一批高效的求解算法. 文献 [25] 提出了求解 3D-CLP 的与或图算法. 文献 [26] 将分支定界法集成到求解 3D-CLP 的算法中. 文献 [7, 27−34] 都是基于树搜索的算法.

按照装载方案的构造方式来分, 这些启发式算法可以分为以下 6 种:

1) 墙构造法 (Wall building approach). 其特征是装载方案由一块块竖直的 “物品墙” 组成, “墙” 依次放入容器中, 其上下左右和后方五个方向需紧贴容器剩余空间的边以提高填充率, 所有 “墙” 的 4 个竖直面至少有一个和容器竖直面重合. 代表性方法有文献 [17, 29, 32−33].

2) 建堆法 (Stack building approach). 其特征是先将物品组合成若干个与容器等高的长方体 “堆”, 再将这些 “堆” 放置到容器中. 与墙构造法不同的是, 堆的 4 个竖直面不必和容器的竖直面重合, 但是堆的上下水平面必须和容器的上下水平面重合. 文献 [9] 和文献 [16] 正是基于这种方法.

3) 水平层构造法 (Horizontal layer building approach). 其特征是装载方案由若干块水平的长方体物品层组成, 层的 4 个竖直面和容器的竖直面重合. 代表性方法有文献 [10] 和文献 [26].

4) 块构造法 (Block building approach). 其特征是装载方案由若干个长方体 “块” 构成, 每个 “块” 由尺寸相同 (或相近) 且摆放方向相同的物品组成. 大多数算法采用这种构造方式. 如文献 [7, 19−20, 27, 31, 34] 中所述方法.

5) 完全切割法 (Guillotine cutting approach). 该方法利用完全切割树将容器分为若干个小的长方体空间, 切割树的叶子结点代表物品. 文献 [25] 正是基于这种方法.

6) 基于穴度的全局构造法 (Caving-degree-based global building approach). 该方法从全局考虑, 依次为每个物品在容器中寻找最佳放置位置, 再评价所有待放置物品放入后的效果, 选择效果最佳的物品放置到最佳位置, 从而获得高效装载方案. 文献 [11−13] 对该方法进行了深入的研究, 取得了一系列成果.

大多数的算法满足摆放方向约束和支撑约束, 除此以外, 还有一些文献提出了满足其他约束的算法. 文献 [34−35] 提出了带有运输优先级约束的算法. 文献 [30] 和文献 [36] 分别提出带有动态优先级和轴重约束的算法. 文献 [37] 算法同时满足运输优先级和完全运输约束. 此外还有大量文献对与 3D-CLP 相关的组合优化问题进行了研究. 文献 [38] 针对带容量约束的绿色车辆路径规划问题, 给出了带有竞争的文化基因算法. 文献 [39] 针对服装展切问题提出了一种反复贪婪算法. 文献 [40] 提出了电动汽车的混合调度策略, 文献 [41] 提出了一种高效的 3D 打印优化算法.

本文提出的多层树搜索算法是墙构造法和水平层法相结合的树搜索算法, 与文献 [32−33] 只有 1 层搜索树不同的是, 本文算法包括 3 层搜索树, 文献 [32−33] 算法只用到竖直层 (即 “墙”), 而本文算法同时包括竖直层和水平层. 本文将所提出算法称为多层树搜索算法 (Multi-level tree search, MLTrS).

2 多层树搜索算法

2.1 名词解释

3D-CLP 问题的形式化定义如下: 给定一个长、宽、高分别为 L, W, H 的长方体容器 C 和 n 个长方体箱子 b_1, b_2, $\cdots$, b_n, 用集合 B 表示这些箱子, 箱子 b_i $(i=1,2,\cdots,n)$的长为 l_i、宽为 w_i、高为 h_i, 箱子 b_i $(i=1,2,\cdots,n)$ 的体积为 $v_i=l_iw_ih_i$. 用 0-1 标志 zl_i、zw_i、zh_i 表示箱子 b_i $(i=1,2,\cdots,n)$ 的长宽高是否允许向上摆放 (0 表示不允许, 1 表示允许), 令

$$C=(L,W,H) \tag{1}$$

$$B=\{b_1,b_2,\cdots,b_n\} \tag{2}$$

$$b_i=(l_i,w_i,h_i,ul_i,uw_i,uh_i) \tag{3}$$

图 1 为坐标系以及容器、箱子尺寸和摆放方向示意图.

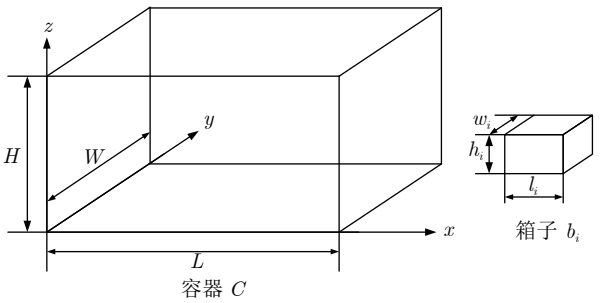


图 1 坐标系和容器、箱子尺寸和摆放方向示意图

Fig. 1 Examples of coordination, size, and orientations of container and box

设 F 为 B 的一个子集, 定义 F 中所有箱子体积之和为 V_F, 问题的目标是选择 B 的一个子集 F 使得 V_F 最大, 并且满足以下条件: 对任何箱子 $b_i\in F$, 在容器 C 中对应一个装填位置; 所有 F 中的箱子必须全部包含在 C 中; F 中任何两个箱子在 C 中不重叠; F 中箱子 b_i 的放置方向必须满足方向标志 ul_i, uw_i, uh_i 的要求. 定义子集 F 对应的容器 C 的填充率为 $FR_F=V_F/(LWH)$. 如果三维装箱问题带有完全支撑约束, 则容器中所有箱子的底部都必须与容器底部或其他箱子的顶部完全接触. 为了便于介绍算法, 进行如下定义:

定义 1. 片 (*piece*). 从长方体箱子的集合 B 取出完全相同的多个箱子进行摆放, 要求在 x 轴、y 轴、z 轴三个方向中的一个方向上只摆放一层, 且所有箱子在该方向上摆放尺寸相同, 则这些箱子的摆放构成一个 *piece*. 定义 *piece* 的三维坐标与尺寸为它的母长方体的三维坐标与尺寸 (在生成 *piece* 之前, 首先给定三维尺寸边界, 生成的 *piece* 不能超过该边界, 则该三维尺寸边界构成的长方体就是 *piece* 的母长方体). 分别以 $XS(piece)$、$YS(piece)$ 和 $ZS(piece)$ 表示 *piece* 的三维尺寸, 以 $VB(piece)$ 表示 *piece* 中所有箱子体积和, 定义 *piece* 的填充率 $Fr(piece)$ 为 $VB(piece)$ 除以 *piece* 的生成长方体体积. 如果 *piece* 在 x 轴方向只摆放一层, 则称为 *x_piece*, 同理可得 *y_piece* 和 *z_piece*, 图 2 为 *x_piece*, *y_piece* 和 *z_piece* 的示意图.

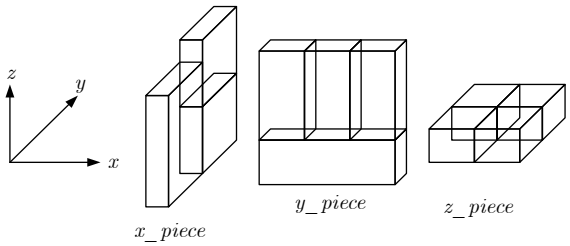


图 2 片的示意图

Fig. 2 Examples of pieces

定义 2. 条 (*strip*). 给定一组 *x_piece* (*y_piece*, *z_piece*), 从中选择一个子集, 在满足箱子摆放方向约束的前提下, 沿 x 轴 (y 轴, z 轴) 摞成一串, 形成条 *strip*. 定义 *strip* 的三维坐标与尺寸为它的母长方体的三维坐标与尺寸 (在生成 *strip* 之前, 首先给定三维尺寸边界, 生成的 *strip* 不能超过该边界, 则该三维尺寸边界构成的长方体就是 *strip* 的母长方体). 分别以 $XS(strip)$、$YS(strip)$ 和 $ZS(strip)$ 表示 *strip* 的三维尺寸, 以 $VB(strip)$ 表示 *strip* 中所有箱子体积和, 定义 *strip* 的填充率 $Fr(strip)$ 为构成 $VB(strip)$ 除以 *strip* 的最小包围长方体体积. 由 *x_piece* 沿 x 轴摞成一串构成的 *strip* 称为 *x_strip*, 同理可得 *y_strip* 和 *z_strip*, 图 3 为 *x_strip*, *y_strip* 和 *z_strip* 的示意图.

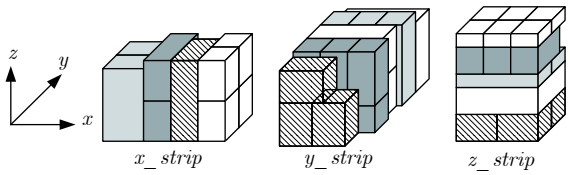


图 3 条的示意图

Fig. 3 Examples of strips

定义 3. 层 (*layer*). 给定一组 *x_strip* 与一组 *y_strip*, 从两组中选择一个子集, 在满足箱子摆放方向约束的前提下, 沿与 xy 面平行的面排列子集中的 *strip*, 所有摆放的 *strip* 的 z 坐标相同, 形成层 *layer*. 定义 *layer* 的三维坐标与尺寸为它的母长方

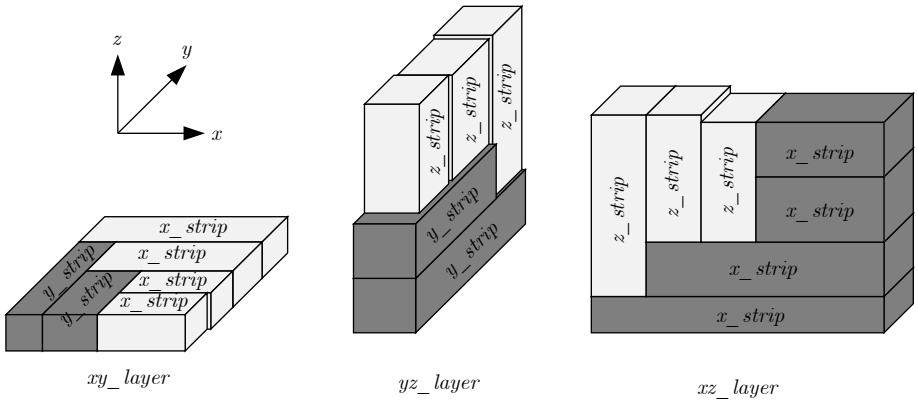


图 4　层的示意图

Fig. 4　Examples of layers

体的三维坐标与尺寸 (在生成 $layer$ 之前, 首先给定三维尺寸边界, 生成的 $layer$ 不能超过该边界, 则该三维尺寸边界构成的长方体就是 $layer$ 的母长方体). 分别以 $XS(layer)$ 、$YS(layer)$ 和 $ZS(layer)$ 表示 $layer$ 的三维尺寸, 以 $VB(layer)$ 表示 $layer$ 中所有箱子体积和, 定义 $layer$ 的填充率 $Fr(layer)$ 为 $VB(layer)$ 除以 $layer$ 的最小包围长方体体积. 由 x_strip 和 y_strip 沿与 xy 面平行的面排列形成的 $layer$ 称为 xy_layer. 同理可得 yz_layer 和 xz_layer, 图 4 为 xy_layer, yz_layer 和 xz_layer 的示意图.

定义 4. 装载实体 ($entity$). 给定容器 C 和箱子集合 B, 从集合 B 中选取一个子集 Q, Q 中的箱子全部装入容器 C, Q 中所有这些箱子的摆放位置和摆放方向整体构成一个装载实体 $entity$, 一个 $entity$ 即是一个装箱方案, 其母长方体就是容器 C 的最大内部空间长方体. 一个 $entity$ 可以分解为一组 $layer$, 每一个 $layer$ 可以分解为一组 $strip$, 每一个 $strip$ 可以分解为一组 $piece$, 每个 $piece$ 则由若干个完全相同的箱子组成. 定义 $VB(entity)$ 为 $entity$ 中所有箱子体积和, 定义 $entity$ 的填充率 $Fr(entity)$ 为 $VB(entity)$ 除以容器 C 的容积. 图 5 为 $entity$ 示意图.

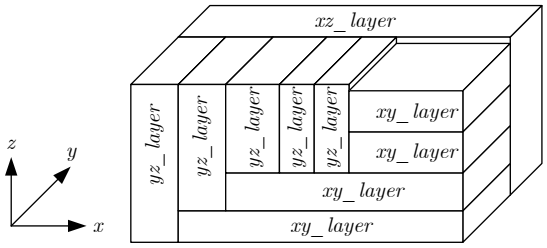


图 5　装载实体示意图

Fig. 5　Example of entity

2.2　算法流程

MLTrS 算法包含 3 层搜索树, 第 1 层树搜索 TrS_Entity 是三叉树, 也是算法的主过程, 用于搜索最优 $entity$; 第 2 层树搜索 TrS_Layer (包含 TrS_XY_Layer、TrS_XZ_Layer 和 TrS_YZ_Layer) 是二叉树, 用于搜索最优 $layer$; 第 3 层树搜索 TrS_Piece (包含 TrS_X_Piece、TrS_Y_Piece 和 TrS_Z_Piece) 是四叉树, 用于搜索最优 $piece$.

MLTrS 算法主过程 TrS_Entity 的伪代码如算法 1 所示. 参数 $best\_entity$ 和 $entity$ 分别表示最好的装载实体和当前的装载实体, 参数 xr 、yr 和 zr 分别表示容器 C 剩余长方体空间的长宽高, 参数 B 是待装入的箱子集合. TrS_Entity 包括 4 个部分 (以大括号分隔), 第 1 部分创建 6 个空白的 $layer$, 以备后用; 第 2 部分是在剩余长方体空间内填入一个最优的 xy_layer; 第 3 部分是在剩余长方体空间内填入一个最优的 xz_layer; 第 4 部分是在剩余长方体空间内填入一个最优的 yz_layer.

算法 1. TrS_Entity 伪代码

TrS_Entity $(best\_entity, entity, xr, yr, zr, B)$

{ 创建空的 $xy\_layer, best\_xy\_layer, xz\_layer,$
$best\_xz\_layer, yz\_layer$ 和 $best\_yz\_layer$

{ for each feasible z_size for xy_layer
　TrS_XY_Layer$(best\_xy\_layer, xy\_layer, xr, yr, z\_size, B)$
　if $(Fr(xy\_layer) > Fr(best\_xy\_layer))$
　$best\_xy\_layer = xy\_layer$
$xy\_entity = entity \cup best\_xy\_layer$
if $(VB(xy\_entity) > VB(best\_entity))$
　$best\_entity = xy\_entity$
$xy\_B = B - best\_xy\_layer$ 中的箱子
TrS_Entity$(xy\_entity, xr, yr, zr - ZS(best\_xy\_layer), xy\_B)$

for each feasible y_size for xz_layer
  TrS_XZ_Layer$(best\_xz\_layer, xz\_layer, xr,$
    $y\_size, zr, B)$
  if $(Fr\,(xz\_layer) > Fr\,(best\_xz\_layer))$
    $best\_xz\_layer = xz\_layer$
$xz\_entity = entity \cup best\_xz\_layer$
if $(VB\,(xz\_entity) > VB\,(best\_entity))$
  $best\_entity = xz\_entity$
$xz_B = B - best_xz_layer$中的箱子
TrS_Entity$(xz\_entity, xr, yr -$
  $YS\,(best\_xz\_layer)\,, zr, xz\_B)$

for each feasible x_size for yz_layer
TrS_YZ_Layer$(best\_yz\_layer,$
  $yz\_layer, x\_size, yr, zr, B)$
if $(Fr\,(yz\_layer) > Fr\,(best\_yz\_layer))$
  $best\_yz\_layer = yz\_layer$
$yz\_entity = entity \cup best\_yz\_layer$
if $(VB\,(yz\_entity) > VB\,(best\_entity))$
  $best\_entity = yz\_entity$
$yz_B = B - best_yz_layer$中的箱子
TrS_Entity$(yz\_entity, xr -$
  $XS\,(best\_yz\_layer)\,, yr, zr, yz\_B)$

以 xy_layer 为例, 它的 x 轴向尺寸和 y 轴向尺寸分别等于 xr 和 yr, 而 z 轴向尺寸可能等于某个箱子在 z 轴方向可能的尺寸, 这样 xy_layer 的可能的 z 轴向尺寸数量就非常多, 为了加速算法, 我们取尺寸权值最大的 5 个尺寸. 尺寸权值计算法定义如下: 初值为 0, 对于每一个箱子的每一个边长, 如果该边长等于该尺寸, 而且该边可与 z 轴平行放置, 则尺寸权值加上与该边垂直的箱子面的面积. xz_layer 和 yz_layer 的可行尺寸选择方法可以参照 xy_layer.

TrS_XY_Layer, TrS_XZ_Layer和 TrS_YZ_Layer 分别用来求 xy_layer, xz_layer 和 yz_layer, 由于三种求解方法是类似的, 限于篇幅, 以过程 TrS_XY_Layer 求解 xy_layer 为例展开叙述, 如算法 2 所示. 参数 $best\_xy\_layer$ 和 xy_layer 分别代表最优 xy_layer 和当前 xy_layer, 参数 xr, yr 和 zr 分别是剩余长方体空间的长宽高, 参数 B 表示待装入的箱子集合. TrS_XY_Layer 包括两个部分 (以大括号分隔), 第 1 部分是在剩余长方体空间内填入一个最优的 x_strip, 第 2 部分是在剩余长方体空间内填入一个最优的 y_strip.

算法 2. TrS_XY_Layer 伪代码

TrS_XY_Layer$(best\_xy\_layer, xy\_layer, xr, yr, zr, B)$

$best\_x\_strip =$ **CreateBest_X_Strip_For_XY_Layer**
  (xr, yr, zr, B)
$x\_xy\_layer = xy\_layer \cup best\_x\_strip$
if $(VB\,(x\_xy\_layer) > VB\,(best\_xy\_layer))$
  $best\_xy\_layer = x\_xy\_layer$
$x\_B = B - best\_x\_strip$ 中的箱子
TrS_XY_Layer$(best\_xy\_layer, x\_xy\_layer, xr, yr -$
  $YS\,(best\_x\_strip)\,, zr, x\_B)$

$best\_y\_strip =$ **CreateBest_Y_Strip_For_XY_Layer**
  (xr, yr, zr, B)
$y\_xy\_layer = xy\_layer \cup best\_y\_strip$
if $(VB\,(y\_xy\_layer) > VB\,(best\_xy\_layer))$
  $best\_xy\_layer = y\_xy\_layer$
$y\_B = B - best\_y\_strip$ 中的箱子
TrS_XY_Layer$(best\_xy\_layer, y\_xy\_layer, xr -$
  $XS\,(best\_y\_strip)\,, yr, zr, y\_B)$

Create_Best_X_Strip_For_XY_Layer 和 Create_Best_Y_Strip_For_XY_Layer 分别为 xy_layer 求解 $best\_x\_strip$ 和 $best\_y\_strip$. 类似的, Create_Best_X_Strip_For_XZ_Layer 和 Create_Best_Z_Strip_For_XZ_Layer 分别为 xz_layer 求解 $best\_x\_strip$ 和 $best\_z\_strip$, Create_Best_Y_Strip_For_YZ_Layer 和 $Create_$Best_Z_Strip_For_YZ_Layer 分别为 yz_layer 求解 $best\_y\_strip$ 和 $best\_\ z\_strip$. 这 6 种求解过程是类似的, 仅以过程 Create_Best_X_Strip_For_XY_$Layer$ 求解 $best\_x\_strip$ 为例展开叙述, 如算法 3 所示. 参数 xr, yr 和 zr 分别是剩余长方体空间的长宽高, 参数 B 表示待装入的箱子集合. 求解之前, $best\_x\_strip$ 的 y 轴向尺寸尚未确定, 可能等于某个箱子在 y 轴方向可能的尺寸, 这样 $best\_x\_strip$ 的可能的 y 轴向尺寸数量就非常多, 为了加速算法, 我们仅取在箱子边长 (该边长必须能平行 y 轴放置) 中出现频率最大的 5 个尺寸, 然后针对这 5 个尺寸, 分别求解出 5 个 x_strip, 从中选择填充率最高的作为最终结果返回.

算法 3. Create_Best_X_Strip 伪代码

Create_Best_X_Strip_XY_Layer (xr, yr, zr, B)
  创建空的x_strip 和 $best\_x\_strip$
  for each feasible y_size for x_strip
  $x\_strip =$**Create_X_Strip** (xr, y_size, zr, B)

if $(Fr\,(x\_strip) > Fr\,(best\_x\_strip))$
$best\_x\_strip = x\_strip$
return $best\_x\_strip$

Create_X_Strip, Create_Y_Strip和 Create_Z_Strip 分别用来求 x_strip , y_strip 和 z_strip. 由于三种求解方法是类似的, 限于篇幅, 以过程 Create_X_Strip 求解 x_strip 为例展开叙述, 如算法 4 所示. 参数 xr , yr 和 zr 分别是剩余长方体空间的长宽高, 参数 B 表示待装入的箱子集合. 求解之前, 先将 B 中所有箱子组合成 x_piece , 再以 xr 为背包容量, 以 $XS\,(x\_piece)$ 为物品重量, 以 $VB\,(x\_piece)$ 为物品价值, 用动态规划方法求解一维背包问题, 得到 x_strip 并返回. 在为每一种箱子求解 x_piece 时, 分别以箱子长宽高为 x_piece 的 x 轴向尺寸, 预求解出 3 个备选项, 从中选择出填充率最高的, 然后去掉已经装入的箱子, 在进行新一轮的 x_piece 生成, 直到该种箱子装完. 这里要去掉箱子过大无法在给定空间中形成 x_piece 的情况.

算法 4. Create_X_Strip 伪代码

Create_X_Strip (xr, yr, zr, B)
创建空的 x_piece_list
for each box type b_i and its number num_i in B
while $(num > 0)$
$x\_piece\_a = \emptyset$
TrS_X_Piece$(x\_piece\_a, l_i, yr, zr, b, num)$
$x\_piece\_b = \emptyset$
TrS_X_Piece$(x\_piece\_b, w_i, yr, zr, b, num)$
$x\_piece\_c = \emptyset$
TrS_X_Piece$(x\_piece\_c, h_i, yr, zr, b, num)$
x_piece=the one with highest Fr among x_piece_a, x_piece_b和x_piece_c
将 x_piece 加入 x_piece_list
$num = num - x\_piece$ 中 b 的数量
$x\_strip =$ 一维书包(xr as 背包容量, x_piece_list as 物品列表)
return x_strip

TrS_X_Piece 、TrS_Y_Piece 和 TrS_Z_Piece 是三个树搜索过程, 分别求解 x_piece、y_piece和 z_piece. 三个过程是类似的, 限于篇幅, 仅以过程 TrS_X_Piece 求解 x_piece 为例展开, 如图 6 所示. 根据算法 4 可知在过程 TrS_X_Piece 中箱子在 x 轴向尺寸已经固定, 所以此时最多考虑两种摆放方向, 分别称为方向 1 和方向 2. 对于每个节点, 其第 1 个子节点是将箱子以方向 1 沿 y 轴向摆放一排; 第 2 个子节点是将箱子以方向 1 沿 z 轴向摆放一排; 第 3 个子节点是将箱子以方向 2 沿 y 轴向摆放一排; 第 4 个子节点是将箱子以方向 2 沿 z 轴向摆放一排. 图中的数字表示箱子是在树的第几层摆放到相应 $piece$ 中的. 实际求解过程中, 一旦找到填充率为 1 的 $piece$ 立即结束搜索, 从而加快计算速度.

2.3　算法加速

由于 MLTrS 算法包含三层搜索树, 因此计算量非常大, 为了加快算法运行速度, 本文采用了以下两种加速机制.

1) 记录每一个片、条、层的创建输入和输出, 形成历史库, 在创建一个新的片、条、层之前, 首先查找历史库中有无类似记录. 如果有, 则直接复制该记录对应的输出, 从而节约大量计算时间. 以创建一个 x_strip 为例, 假设输入中的三维空间长宽高分别为 x_s , y_s 和 z_s , 箱子集合为 B_i , 如果给定的历史库中存在一条 x_strip 创建记录, 其输入中的三维空间长宽高分别等于 x_s , y_s 和 z_s , 其输入输出箱子集合分别为 B_h 和 x_strip_h , 如果 B_i 是

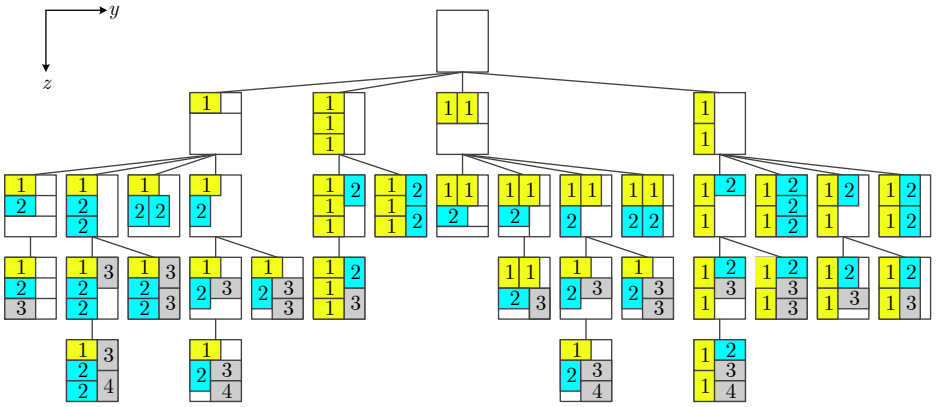


图 6　TrS_X_piece 搜索树
Fig. 6　The searching tree of TrS_X_piece

B_h 的子集, 且 x_strip_h 是 B_i 的子集, 则该记录是类似记录, 此时直接以 x_strip_h 作为求解结果.

2) 树搜索 TrS_Entity 在搜索最优 *entity* 时, 从某一节点向下搜索之前, 首先计算当前 *entity* 不可利用空间 (即容器中所有 *layer* 的母长方体体积和减去容器中所有 *layer* 中的所有箱子体积和) 占容器体积的百分比, 再用 1 减去该百分比得到当前树分枝可得结果的填充率上限 fr_UB. 如果 fr_UB 小于已找到的最优 *entity* 的填充率, 则不再沿该树分枝往下搜索, 从而减少计算时间.

2.4 对完全支撑的处理

按照第 2.2 节的算法流程求解出的装载方案满足摆放方向约束 (C1). 为了满足完全支撑约束 (C2), 需要对 MLTrS 算法流程进行如下调整.

1) 在创建片 (*piece*)时, 仅考虑箱子体积等于最小包围长方体体积的情况, 即 *piece* 构成一个完整的长方体, 不存在缺角. 最小包围长方体和母长方体的区别在于: 母长方体是在 *piece* 创建之前给定的长方体边界, *piece* 的最小包围长方体体积小于等于其母长方体体积.

2) 树搜索 TrS_Entity 在搜索最优 *entity* 时, 仅创建 yz_layer 和 xz_layer 两种 *layer*, 不创建 xy_layer, 因此 TrS_Entity 在考虑完全支撑时由三叉树变成了二叉树.

3) 在创建 x_strip (y_strip) 时, 以所有箱子与 x_strip (y_strip) 母长方体的上表面接触面的最大正交内接长方形 (不超过接触面的边界, 且各边平行于容器的水平边的面积最大的长方形, 称为 x_strip (y_strip) 最大平顶长方形) 面积最大为选优标准, 而不是以 $Fr\,(x\_strip)$ ($Fr\,(y\_strip)$) 为选优标准. 当 x_strip (y_strip) 放入相应 $xz\_\ layer$ (yz_layer) 时, 仅 x_strip (y_strip) 的最大平顶长方形正上方的空间可以用于装载剩余的箱子.

4) 在创建 z_strip 时, 除最下方的 *piece* 外, 其他 *piece* 的下表面必须与其紧下方 *piece* 的上表面完全接触. 在生成所有 *piece* 且尚未创建 z_strip 时 (不失一般性, 对所有 *piece*, 令 $XS\,(piece) \geq YS\,(piece)$), 先将所有 *piece* 按照 $XS\,(piece)$ 值降序排列, 对于 $XS\,(piece)$ 相同的多个 *piece*, 再按 $YS\,(piece)$ 值降序排列, 从而得到一个新的 *piece* 列表. 在利用动态规划求解一维背包时, 就可以较为容易地剔除不满足完全支撑约束的解.

3 计算实验

本文 MLTrS 算法用 C#语言实现, 实验程序运行计算机的 CPU 为 Intel Xeon E5 2660@2.20 GHz, 内存为 16 GB, 操作系统为 64 位 Windows 7 Ultimate 版, 实验程序仅使用 CPU 的 8 个核中的一核. 本文的测试数据集来源于文献 [9], 该测试数据集包括 BR1~BR15, 共 15 个子集, 每个子集包含 100 个算例, 同一子集中每个算例包含的箱子种类数相同, BR1~BR15 中箱子的种类数分别为 3, 5, 8, 10, 12, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, 问题由弱异构到强异构逐渐过渡, 能够很好地测试算法在不同复杂度装箱问题中的性能. 我们在用 MLTrS 运行算例时, 针对不同复杂度的算例给定不同的计算时间限制, 限定时间 T 的计算式为: 当仅考虑摆放方向约束 (C1) 时, 对于弱异构算例 BR1~BR7, $T = 600$, 对于强异构算例 BR8~BR15, $T = 150 \times \sqrt{N}$, 当同时考虑摆放方向约束 (C1) 和完全支撑约束 (C2) 时, 对于弱异构算例 BR1~BR7, $T = 500$, 对于强异构算例 BR8~BR15, $T = 100 \times \sqrt{N}$, 其中 N 为算例的箱子种类数.

许多研究者全部或部分测试了 BR1~BR15. 其中组合启发式算法 (Combinational heuristics, CH)[16]、H_B(Heuristics of Bischoff) 算法[22] 测试了 BR1~BR7; A2(Algorithm 2) 算法[12] 测试了 BR8~BR15; GA_GB(Genetic algorithm of gehring and bortfeldt) 算法[16]、贪心随机自适应搜索算法 (Greedy random adaptive search procedure, GRASP)[23]、混合模拟退火算法 (Hybrid simulated annealing, HSA)[21]、maximal-space 算法[24]、基于整数拆分的树搜索算法 (Container loading tree search, CLTRS)[7]、FDA 算法 (Fit degree algorithm)[11]、多层启发式搜索算法 (Multiple layer heuristic algorithm, MLHS)[31] 测试了 BR1 ~ BR15 全部实例.

上述算法全部满足 C1 约束, 部分满足 C2 约束, 算法的计算结果直接来自于相应文献. 表 1 列出了这些算法与本文 MLTrS 算法对 BR1~BR7 的计算结果, 表 1 中所有数据表示每个算法针对一类实例所得到的平均填充率.

从表 1 可以看出, 针对 BR1~BR7 这 7 组弱异构算例, MLTrS 无论在满足 C1 约束还是在同时满足 C1 和 C2 约束的前提下, 都要弱于现有基于 Block building 构造方式的算法, 如 CLTRS[7]、MLHS[31], 从文献 [32–33] 和本文的算例测试结果来看, 基于 Wall-building 构造方式的算法对弱异构问题的求解性能弱于基于 Block building 构造方式的算法.

接下来观察本文算法对强异构算例的测试结果. 表 2 列出了各个算法对 BR8~BR15 的计算结果, 表 2 中所有数据表示每个算法针对一类实例所得到的平均填充率.

从表 2 可以看出, 在同时满足 C1 和 C2 约束的前提下, 当箱子种类数在 60 种以上时, 填充率超

表 1　各种算法对 BR1~BR7 的填充率比较

Table 1　Comparison of filling rates of BR1~BR7 by various algorithms

算法	约束	填充率 (%)							
		BR1	BR2	BR3	BR4	BR5	BR6	BR7	平均
GA_GB[16]	C1&C2	85.80	87.26	88.10	88.04	87.86	87.85	87.68	87.5
GRASP[23]	C1	93.52	93.77	93.58	93.05	92.34	91.72	90.55	92.65
FDA[11]	C1	92.92	93.93	93.71	93.68	93.73	93.63	93.14	93.53
HSA[21]	C1&C2	93.81	93.94	93.86	93.57	93.22	92.72	91.99	93.30
Maximal-space[24]	C1	93.85	94.22	94.25	94.09	93.87	93.52	92.94	93.82
A2[12]	C1	–	–	–	–	–	–	–	–
CLTRS[7]	C1	95.05	95.43	95.47	95.18	95.00	94.79	94.24	95.02
	C1&C2	94.51	94.73	94.74	94.41	94.13	93.85	93.20	94.22
MLHS[31]	C1	94.92	95.48	95.69	95.53	95.44	95.38	94.95	95.34
	C1&C2	94.49	94.89	95.20	94.94	94.78	94.55	93.95	94.69
MLTrS	C1	93.81	93.83	93.80	93.98	93.97	93.83	93.91	93.88
	C1&C2	93.48	93.41	93.79	93.85	93.43	93.65	93.63	93.61

表 2　各种算法对 BR8~BR15 的填充率比较

Table 2　Comparison of filling rates of BR8~BR15 by various algorithms

算法	约束	填充率 (%)								
		BR8	BR9	BR10	BR11	BR12	BR13	BR14	BR15	平均
GA_GB[15]	C1&C2	87.52	86.46	85.53	84.82	84.25	83.67	82.99	82.47	84.71
GRASP[23]	C1	90.26	89.50	88.73	87.87	87.18	86.70	85.81	85.48	87.69
FDA[11]	C1	92.92	92.49	92.24	91.91	91.83	91.56	91.30	91.02	91.9
HSA[21]	C1&C2	90.56	89.7	89.06	88.18	87.73	86.97	86.16	85.44	87.98
Maximal-space[24]	C1	91.02	90.46	89.87	89.36	89.03	88.56	88.46	88.36	89.39
A2[12]	C1	88.41	88.14	87.9	87.88	87.92	87.92	87.82	87.73	87.97
CLTRS[7]	C1	93.70	93.44	93.09	92.81	92.73	92.46	92.40	92.40	92.88
	C1&C2	92.26	91.48	90.86	90.11	89.51	88.98	88.26	87.57	89.88
MLHS[31]	C1	94.54	94.14	93.95	93.61	93.38	93.14	93.06	92.90	93.59
	C1&C2	93.12	92.48	91.83	91.23	90.59	89.99	89.34	88.54	90.89
MLTrS	C1	93.98	93.97	93.78	93.75	93.63	93.43	93.25	93.12	93.61
	C1&C2	91.82	91.69	91.55	**91.24**	**91.21**	**91.03**	**90.78**	**90.59**	**91.24**

过了目前已知的所有算法. 在同时满足 C1 和 C2 约束的前提下, 与 CLTRS[7]、MLHS[31] 算法的对比如下: 当箱子种类数为 70 时, MLTrS 填充率分别提高 1.7% 和 0.62%; 当箱子种类数为 80 时, MLTrS 填充率分别提高 2.05% 和 1.04%; 当箱子种类数为 90 时, MLTrS 填充率分别提高 2.52% 和 1.44%; 当箱子种类数为 100 时, MLTrS 填充率分别提高 3.02% 和 2.05%. 另外, 针对 BR8~BR15 总平均值, MLTrS 分别提高 1.36% 和 0.35%. 在满足 C1 约束的前提下, MLTrS 比 CLTRS[7] 提高 0.73%, 略高于 MLHS[31].

由此可见, 本文算法在箱子种类数大于等于 70, 且要求满足摆放方向约束和完全支撑约束时, 填充率高于现有主流算法, 如 CLTRS[7]、MLHS[31] 等, 而且当箱子种类数越多, 本文算法求得的填充率比现有主流算法求得的填充率高得越多. 而对于箱子种类数小于 70 的算例, 本文算法要弱于 CLTRS[7] 和 MLHS[31], 并且箱子种类数越少, 本文算法求得的填充率比现有主流算法求得的填充率低得越多. 这表明相对于现有主流算法, 如 CLTRS[7]、MLHS[31], 本文算法更适合求解箱子种类数较多的算例, 即强异构算例.

4　结束语

本文提出了三维装箱问题的多层树搜索算法, 采用墙构造法和水平层构造法相结合的装载方案构造方式. 该算法包括 3 层搜索树, 第 1 层搜索树也是最顶层搜索树, 用于搜索最优装载方案, 该树为

三叉树, 每个树节点的三个分叉分别对应生成平行于 XY 面的层、生成平行于 XZ 面的层、生成平行于 YZ 面的层; 第 2 层搜索树用于为第 1 层树的每个节点搜索最优的层, 该树为二叉树, 每个树节点的两个分叉分别对应生成两个相互垂直的条; 第 3 层搜索树用于将每一类箱子生成片, 该树为四叉树, 由于生成片时限定空间比较小, 同类型的箱子数量相对小, 四叉树搜索速度可以接受. 本文提出的多层树搜索算法搜索量比较大, 难以在可接受时间内完成对整个搜索树的遍历, 为此采用了加速策略, 并且限定算法的运行时间. 利用通用算例的运行结果显示本文算法具有一定的竞争力. 我们在运行该算法时也发现一些问题, 例如当箱子体积相对于容器容积越小时, 树搜索深度就越深, 计算时间就越长, 有时甚至不能在限定时间内生成一个完整的装载方案, 因此本算法还有进一步改进的余地.

References

1 Dyckhoff H, Finke U. *Cutting and Packing in Production and Distribution*. Heidelberg: Physica-Verlag, 1992

2 George J A, Robinson D F. A heuristic for packing boxes into a container. *Computers and Operations Research*, 1980, **7**(3): 147−156

3 Dyckhoff, H. A typology of cutting and packing problems. *European Journal of Operational Research*, 1990, **44**(2): 145−159

4 Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 2007, **183**(3): 1109−1130

5 Bortfeldt A, Wäscher G. Constraints in container loading — a state-of-the-art review. *European Journal of Operational Research*, 2013, **229**(1): 1−20

6 Scheithauer G. Algorithms for the container loading problem. In: Proceedings of the 1991 on Operations Research. Berlin, Heidelberg, Germany: Springer, 1992. 445−452

7 Fanslau T, Bortfeldt A. A tree search algorithm for solving the container loading problem. *INFORMS Journal on Computing*, 2010, **22**(2): 222−235

8 Fekete S P, Schepers J, Van der Veen J C. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 2007, **55**(3): 569−587

9 Bischoff E E, Ratcliff B S W. Issues in the development of approaches to container loading. *Omega*, 1995, **23**(4): 377−390

10 Bischoff E E, Janetz F, Ratcliff M S W. Loading pallets with non-identical items. *European Journal of Operational Research*, 1995, **84**(3): 681−692

11 He K, Huang W Q. An efficient placement heuristic for three-dimensional rectangular packing. *Computers and Operations Research*, 2011, **38**(1): 227−233

12 Huang W Q, He K. A caving degree approach for the single container loading problem. *European Journal of Operational Research*, 2009, **196**(7): 93−101

13 He K, Huang W Q. A caving degree based flake arrangement approach for the container loading problem. *Computers and Industrial Engineering*, 2010, **59**(2): 344−351

14 Lim A, Rodrigues B, Wang Y. A multi-faced buildup algorithm for three-dimensional packing problems. *Omega*, 2003, **31**(6): 471−481

15 Zhang De-Fu, Wei Li-Jun, Chen Qing-Shan, Chen Huo-Wang. A combinational heuristic algorithm for the three dimensional packing problem. *Journal of Software*, 2007, **18**(9): 2083−2089
(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)

16 Gehring H, Bortfeldt A. A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 1997, **4**(5–6): 401−418

17 Bortfeldt A, Gehring H. A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 2001, **131**(1): 143−161

18 Sixt M. *Dreidimensionale Packprobleme. Losungsverfahren Basierend auf den Meta-Heuristiken Simulated Annealing und Tabu-Suche*. Frankfurt am Main: Europaischer Verlag der Wissenschaften, 1996.

19 Bortfeldt A, Gehring H, Mack D. A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 2003, **29**(5): 641−662

20 Mack D, Bortfeldt A, Gehring H. A parallel hybrid local search algorithm for the container loading problem. *International Transactions in Operational Research*, 2004, **11**(5): 511−533

21 Zhang De-Fu, Peng Yu, Zhu Wen-Xing, Chen Huo-Wang. A hybrid simulated annealing algorithm for the three-dimensional packing problem. *Chinese Journal of Computers*, 2009, **32**(11): 2147−2156
(张德富, 彭煜, 朱文兴, 陈火旺. 求解三维装箱问题的混合模拟退火算法. 计算机学报, 2009, **32**(11): 2147−2156)

22 Bischoff E E. Three-dimensional packing of items with limited load bearing strength. *European Journal of Operational Research*, 2006, **168**(3): 952−966

23 Moura A, Oliveira J F. A GRASP approach to the container-loading problem. *IEEE Intelligent Systems*, 2005, **20**(4): 50−57

24 Parreno F, Alvarez-Valdes R, Oliveira J F, Tamarit J M. A maximal-space algorithm for the container loading problem. *INFORMS Journal on Computing*, 2007, **20**(3): 412−422

25 Morabito R, Arenales M. An AND/OR-graph approach to the container loading problem. *International Transactions in Operational Research*, 1994, **1**(1): 59−73

26 Terno J, Scheithauer G, Sommerweiß U, Riehme J. An efficient approach for the multi-pallet loading problem. *European Journal of Operational Research*, 2000, **123**(2): 372−381

27 Eley M. Solving container loading problems by block arrangement. *European Journal of Operational Research*, 2002, **141**(2): 393−409

28 Hifi, M. Approximate algorithms for the container loading problem. *International Transactions in Operational Research*, 2002, **9**(6): 747−774

29 Pisinger D. Heuristics for the container loading problem. *European Journal of Operational Research*, 2002, **141**(2): 143−153

30 Lim A, Ma H, Xu J, Zhang X W. An iterated construction approach with dynamic prioritization for solving the container loading problems. *Expert Systems with Applications*, 2012, **39**(4): 4292−4305

31 Zhang De-Fu, Peng Yu, Zhang Li-Li. A multi-layer heuristic search algorithm three dimensional container loading problem. *Chinese Journal of Computers*, 2012, **35**(12): 2553−2561
(张德富, 彭煜, 张丽丽. 求解三维装箱问题的多层启发式搜索算法. 计算机学报, 2012, **35**(12): 2553−2561)

32 Liu S, Tan W, Xu Z Y, Liu X W. A tree search algorithm for the container loading problem. *Computers and Industrial Engineering*, 2014, **75**: 20−30

33 Liu Sheng, Zhu Feng-Hua, Lv Yi-Sheng, Li Yuan-Tao. A heuristic orthogonal binary tree search algorithm for three dimensional container loading problem. *Chinese Journal of Computers*, 2015, **38**(8): 1530−1543
(刘胜, 朱凤华, 吕宜生, 李元涛. 求解三维装箱问题的启发式正交二叉树搜索算法. 计算机学报, 2015, **38**(8): 1530−1543)

34 Ren J D, Tian Y J, Sawaragi T. A tree search method for the container loading problem with shipment priority. *European Journal of Operational Research*, 2011, **214**(3): 526−535

35 Wang N, Lim A, Zhu W B. A multi-round partial beam search approach for the single container loading problem with shipment priority. *International Journal of Production Economics*, 2013, **145**(2): 531−540

36 Lim A, Ma H, Qiu C Y, Zhu W B. The single container loading problem with axle weight constraints. *International Journal of Production Economics*, 2013, **144**(1): 358−369

37 Liu S, Shang X Q, Cheng C J, Zhao H X, Wang F-Y. Heuristic

algorithm for the container loading problem with multiple constraints. *Computers and Industrial Engineering*, 2017, **108**: 149−164

38 Wang L, Lu J W. A memetic algorithm with competition for the capacitated green vehicle routing problem. *IEEE/CAA Journal of Automatica Sinica*, 2019, **6**(2): 516−526

39 Shang X Q, Shen D Y, Wang F-Y, Nyberg T R. A heuristic algorithm for the fabric spreading and cutting problem in apparel factories. *IEEE/CAA Journal of Automatica Sinica*, 2019, **6**(4): 961−968

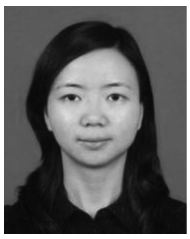
40 Luo C, Shen Z, Evangelou S, Xiong G, Wang F-Y. The combination of two control strategies for series hybrid electric vehicles. *IEEE/CAA Journal of Automatica Sinica*, 2019, **6**(2): 596−608

41 Shen Z, Shang X Q, Dong X S, Xiong G, Wang F-Y. A learning based framework for error compensation in 3D printing. *IEEE Transactions on Cybernetics*, 2019, **49**(11): 4042−4050

刘 胜 中国科学院自动化研究所复杂系统管理与控制国家重点实验室副研究员. 主要研究方向为组合优化, 智能物流, 智能制造.
E-mail: sheng.liu@ia.ac.cn
(**LIU Sheng** Associate professor at the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences. His research interest covers combinatorial optimization, intelligent logistics, and intelligent manufacturing.)

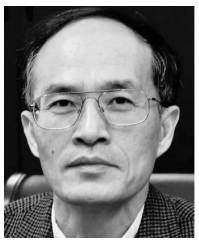
沈大勇 2011 年获国防科技大学系统工程学院系统工程学士学位, 2013 年获国防科技大学系统工程硕士学位, 2018 年获国防科技大学管理科学与工程博士学位. 主要研究方向为智能调度, 人工智能算法和并行社会系统. 本文通信作者.
E-mail: dayong.shen@nudt.edu.cn
(**SHEN Da-Yong** Received his bachelor and master degrees in system engineering from National University of Defense Technology (NUDT) in 2011 and 2013, respectively. He received his Ph.D. degree in management science and engineering from NUDT in 2018. His research interest covers intelligent scheduling, artificial intelligence algorithm, and parallel social systems. Corresponding author of this paper.)

商秀芹 中国科学院自动化研究所复杂系统管理与控制国家重点实验室副研究员. 2010 年获浙江大学控制理论与控制工程博士学位. 主要研究方向为智能制造, 工业过程建模与优化.
E-mail: xiuqin.shang@ia.ac.cn
(**SHANG Xiu-Qin** Associate professor at the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences. She received her Ph.D. degree in control theory and control engineering from Zhejiang University, in 2010. Her research interest covers intelligent manufacturing, industrial process modeling and optimizing.)



赵红霞 中国科学院自动化研究所复杂系统管理与控制国家重点实验室助理研究员. 主要研究方向为智能交通系统.
E-mail: hongxia.zhao@ia.ac.cn
(**ZHAO Hong-Xia** Assistant professor at the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences. Her main research interest is intelligent transportation system.)

董西松 中国科学院自动化研究所复杂系统管理与控制国家重点实验室副研究员. 2007 年获北京科技大学控制理论与控制工程博士学位. 主要研究方向为复杂系统的建模与控制, 智能交通系统.
E-mail: xisong.dong@ia.ac.cn
(**DONG Xi-Song** Associate professor at the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences. He received his Ph.D. degree from University of Science and Technology Beijing in 2007. His research interest covers modeling and analysis of complex systems, and intelligent transportation system.)



王飞跃 中国科学院自动化研究所复杂系统管理与控制国家重点实验室主任, 国防科技大学军事计算实验与平行系统技术研究中心主任, 中国科学院大学中国经济与社会安全研究中心主任, 青岛智能产业技术研究院院长. 主要研究方向为平行系统的方法与应用, 社会计算, 平行智能以及知识自动化.
E-mail: feiyue.wang@ia.ac.cn
(**WANG Fei-Yue** State specially appointed expert and director of the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences. Professor of the Research Center for Computational Experiments and Parallel Systems Technology, National University of Defense Technology. Director of China Economic and Social Security Research Center in University of Chinese Academy of Sciences. Dean of Qingdao Academy of Intelligent Industries. His research interest covers methods and applications for parallel systems, social computing, parallel intelligence, and knowledge automation.)