求解集装箱装载问题的混合蚁群模拟退火算法

李想[1]，袁锐波[1*]，杨灏泉[2]
（1.昆明理工大学 机电工程学院，昆明 650504；2.云南柔控科技有限公司，昆明 650031）

摘要：目的 针对物流行业中存在的大规模、复杂、多规格货物的集装箱装载问题，提出一种基于塔装载启发式算法、二维装载点启发式算法、蚁群模拟退火算法的混合算法。**方法** 首先，采用塔装载启发式算法将三维待装箱装载成塔集，即将三维装箱问题降为二维装箱问题，有效降低集装箱的装载规模；其次，蚁群算法通过融入信息素选择更新策略，并利用自适应信息素挥发系数来提升算法整体的收敛速度，同时结合模拟退火算法对每代优秀路径集进行局部搜索，避免算法因收敛过快而陷入局部最优；最后，将蚁群模拟退火算法与二维装载点启发式算法相结合，优化每座塔的装载顺序和放置姿态，寻找最优的装载方案。**结果** 实验证明，在 250 组算例中，采用混合算法后，集装箱的平均空间利用率为 90.92%，优于其他 3 种对比算法。**结论** 设计的混合蚁群模拟退火算法适用于解决大规模集装箱装载问题。

关键词：三维装箱；大规模集装箱装载；启发式算法；蚁群算法；模拟退火算法
中图分类号：TB485.3 **文献标志码：**A **文章编号：**1001-3563(2024)11-0163-12
DOI：10.19554/j.cnki.1001-3563.2024.11.019

Hybrid Ant Colony Simulated Annealing Algorithm for Solving Container Loading Problems

LI Xiang[1], *YUAN Ruibo*[1*], *YANG Haoquan*[2]
(1. Faculty of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming 650504, China; 2. Yunnan Roukong Technology Co., Ltd., Kunming 650031, China)

ABSTRACT: The work aims to propose a hybrid algorithm of tower loading heuristic algorithm, two-dimensional loading point heuristic algorithm, and ant colony simulated annealing algorithm to address the container loading problem of large-scale and complex multi specification goods in the logistics industry. Firstly, the three-dimensional container was loaded into towers through the tower loading heuristic algorithm to reduce the three-dimensional packing problem to a two-dimensional packing problem, effectively reducing the loading scale of large-scale containers. Secondly, the ant colony algorithm incorporates a pheromone selection and update strategy and an adaptive pheromone evaporation coefficient to improve the overall convergence speed of the algorithm. At the same time, it combines with simulated annealing algorithm to perform local search on the set of excellent paths in each generation, avoiding the algorithm from falling into local optima due to too fast convergence. Finally, the ant colony simulated annealing algorithm was combined with a two-dimensional loading point heuristic algorithm to optimize the loading sequence and placement posture of each tower to find the optimal loading plan. The experiment showed that in 250 sets of examples, the average space utilization rate of the container in this algorithm was 90.92%, which was better than that of the three comparative algorithms. In conclusion, the hybrid ant colony simulated annealing algorithm designed in this article is very suitable for solving large-scale container packing problems.

KEY WORDS: three-dimensional container loading; large scale container loading; heuristic algorithm; ant colony; simulated annealing algorithm

收稿日期：2023-08-22
基金项目：云南省重大科技专项（202202AC080008）；中泰国际技术转移中心项目（GHJD-2022001）
*通信作者

货物的装载与运输是物流行业中的关键步骤之一。目前，集装箱装载大多由人工完成，在货物的种类和数量较少时，可以根据经验进行码放，在货物数量种类较多的情况下人工很难给出合理的装箱方案，车厢的空间利用率较低，导致运输成本上升。在物流行业自动化大趋势下，迫切需要实现装箱方案的智能化，因此研究可靠、稳定、高效的装箱算法很有必要。

三维装箱属于 C&P（Cutting and Packing）问题[1-2]的分支，也是一种经典的组合优化问题，所以很难采用精确算法求解 NP-hard 问题。目前，针对三维装箱算法的研究，国内外取得了一些成果。George 等[3]提出“层”的思想，即先按照一定规则将待装箱进行排序，再将箱子划分为层，将其简化为一维装箱问题。Eley[4]引入了“块”的思想，即首先将同种类的箱子按照同一姿态进行摆放，组合成块，然后将这些块优先装入容器，最后对剩余空隙用未打包成块的货物进行填充。Fanslau 等[5]将“块”的思想进行了推广，可以将不同货物打包成“块”。Menghani 等[6]提出了 2 种改进型遗传算法，并与启发式算法相结合，该算法具有不错的全局搜索能力，但其局部搜索能力较差。Bortfeldt 等[7]基于文献[3]的思想，结合遗传算法进行求解。Moura 等[8]首次引入了 RS（Residual-Space）思想，基于贪婪算法的思想进行装箱操作。Araya 等[9]采用双目标函数，基于多角度求解三维装箱问题。国内近些年也取得了不少成果。如，卓雪艳等[10]首先将三维装箱问题降为二维装箱问题，并且设计了一套自动化装箱系统。李伟等[11]将传统启发式算法与 GA、TS 元启发式算法相结合，求解了三维装箱问题。张德富等[12]将“块”的启发式算法与模拟退火算法结合，具有较高的填充率。张长勇等[13]针对机场行李码垛这种在线装箱形式设计了一种“关键点”方法，并结合机器学习算法来寻找最优方案。张长勇等[14]将文献[13]的“关键点”方法与改进粒子群算法相结合，提高了算法的快速性。刘胜等[15]利用树搜索算法解决三维装箱问题，填充率得到明显提升，但是其搜索时间较长。陈元文[16]在遗传算法中融入优先保持策略，解决了三维装箱问题。刑志伟等[17]设计了一种动态四叉树算法，在搜索过程中动态保留最优方案，对于强异构型货物具有不错的填充效果。

综上可知，目前的三维装箱算法无法应对大规模复杂集装箱装载问题，容易造成集装箱空间利用不充分，导致其运输成本提高。由此，文中针对上述问题设计混合蚁群模拟退火算法，以求解大规模集装箱装载。首先采用塔装载启发式算法生成塔集，将三维装箱问题降为二维装箱问题，有效降低装载规模，然后通过改进蚁群模拟退火算法，并与二维投影点启发式算法相结合，寻找塔的最优装箱方案。

1 问题描述

1.1 数学模型

给定一个长度 L、宽度 W 和高度 H 的集装箱，分别以集装箱的长、宽、高及左后下角点 O 建立三维坐标系（长为 x 轴、宽为 y 轴、高为 z 轴，点 O 为坐标原点）。现有若干个箱子集 $B=\{b_1, b_2, b_3, \ldots, b_n\}$，其中任意一个箱子 b_i 的长宽高分别为 l_i、w_i、h_i。此问题需在满足各项约束的前提下，使得集装箱内部的剩余空隙达到最小，且使集装箱尽可能装载现有的箱子。综上可知，问题的目标函数见式（1）。

$$\max\ u=\frac{\sum l_i\times w_i\times h_i}{L\times W\times H} \tag{1}$$

式中：u 为空间利用率。

1.2 基本假设

在实际情况中，三维装箱问题较复杂。为了方便研究，做出以下假设。

1）集装箱和箱子都为规则的长方体。

2）所有箱子的质心为几何中心。

3）箱子不会随着彼此的堆叠而发生变形。

1.3 约束条件

1）体积约束：所有三维待装箱的体积必须小于或等于集装箱的体积。

2）重量约束：所有三维待装箱的重量都必须小于或等于集装箱的承重。

3）边界约束：要求装入集装箱的箱子全部在其内部。

4）方向约束：对于任意的三维待装箱，在摆放时只能以 3 边分别沿着 x 轴、y 轴和 z 轴平行摆放，总共有 6 种摆放姿态，如图 1 所示。

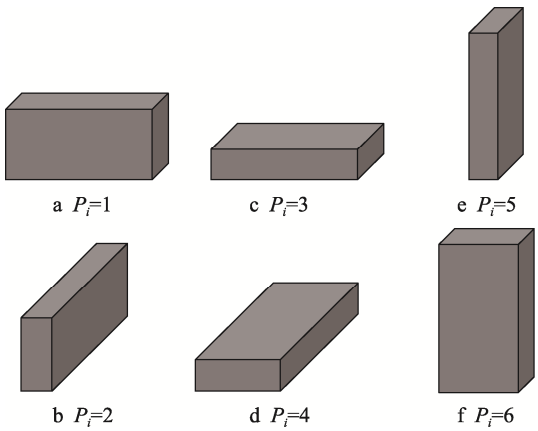


图 1 箱子的 6 种摆放姿态

Fig.1 Six postures for placing boxes

5）不重叠约束：装入集装箱的箱子彼此间不能发生重叠。

6）稳定性约束：对于每个在集装箱内部的箱子，其底面必须有集装箱底面或其他箱子的顶面作为支

撑，并且不能悬空。

7）完全切割约束：在约束 6）的前提下，要求箱体彼此之间堆叠形成的块可以被若干个水平面和竖直面切割成单个箱子，且这些用于切割的平面不能与块中的箱子相交。在实际情况中，此约束主要为了保证货物足够稳定，且利于叉车卸货。图 2a~b 所示为允许的放置状态，图 2c~d 为不允许的放置状态。

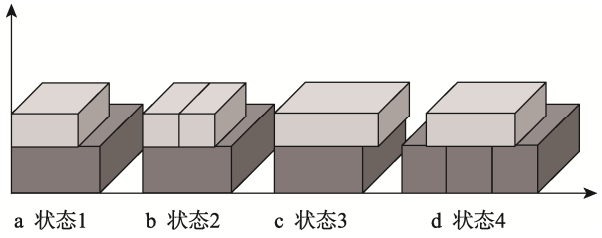


图 2　箱子的放置状态
Fig.2 Placement status of box

2　塔装载启发式算法

采用贪婪算法的思想，首先将所有待装箱按照一定规则排序，然后按照它们排列后的顺序依次沿着高度方向摆放，生成塔集，以此实现三维装箱问题到二维装箱问题转变（如图 3 所示），即达到降低装载规模的目的；且塔内空间的箱子采用最适应剩余空间启发式算法，该算法可以使单个塔内部的剩余空间得到充分利用，进而在求解大规模集装箱装载问题时不会因为降维而出现空间利用率降低的问题。

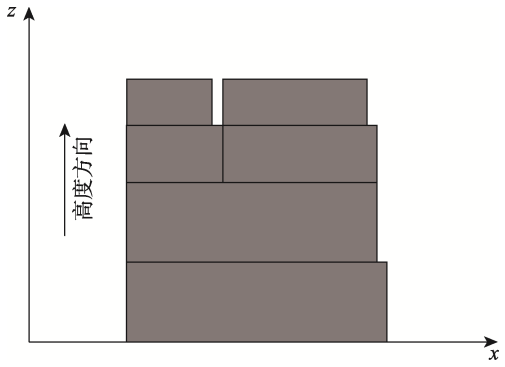


图 3　单个塔的正视图
Fig.3 Front view of a single tower

2.1　算法说明

基于有限的篇幅，对单个塔的生成进行说明即可解释塔装载启发式算法，具体步骤如下。

1）计算当前所有三维待装箱的长宽面、长高面及宽高面的面积，记 3 个面中面积最大的面为 maxarea 面。

2）将当前所有的三维待装箱按照其 maxarea 面的面积从大到小排序。

3）选择当前序列的第 1 个箱子为塔基，且将该箱子的 maxarea 面与该面距离集装箱箱顶的这段距离组成的空间作为后续箱子装载的初始剩余空间，如图 4 所示。

4）后续箱子按照步骤 2）中的顺序依次利用最适应剩余空间启发式算法（该算法将在 2.2 节说明）进行装载，直到塔内放不下任何箱子，或无任何待装箱为止，然后将该塔纳入塔集中，并记录该塔的信息，包括塔内已装载箱子的位置信息和整个塔内箱子的总体积等。

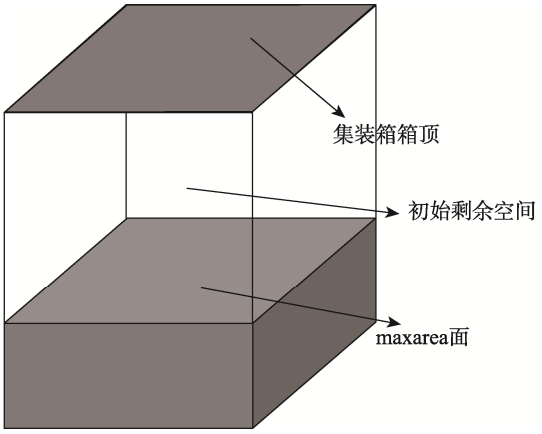


图 4　塔的初始剩余空间
Fig.4 Initial remaining space of tower

2.2　最适应剩余空间启发式算法

通过分析可知，箱子的装载主要有 3 个关键问题：一是箱子放置姿态及放置空间的选择；二是剩余空间的分割；三是剩余空间的合并。下面针对这 3 个关键问题进行设计。

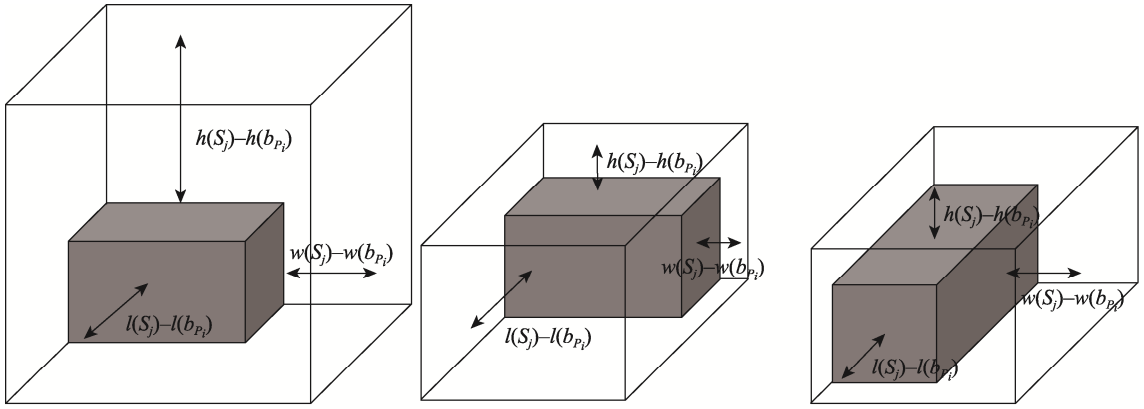
2.2.1　箱子放置姿态及放置空间的选择

对于当前装入塔内空间的箱子，它能以多种姿态装入多个剩余空间中，这样当前待装箱就有多种放置可能性，所以需要设计合理的规则为该箱子选择合适的放置空间及放置姿态。具体操作：在当前箱子能以某种姿态放入某一剩余空间时，此时箱子的摆放格局与该剩余空间越接近，则越倾向于选择这种情况下的放置空间及放置姿态。这样选择的原因是可以空出更大的剩余空间，为后续箱子的装载提供更大的可能性。下面给出具体的评价函数，见式（2）。

$$F_{ij}=-[l(S_j)-l(b_{P_i})+\varrho\,]\times[w(S_j)-w(b_{P_i})+\varrho\,]\times[h(S_j)-h(b_{P_i})+\varrho] \tag{2}$$

式中：$l(S_j)$、$w(S_j)$、$h(S_j)$为第 j 个剩余空间沿着 x、y、z 轴的长度；$l(b_{P_i})$、$w(b_{P_i})$、$h(b_{P_i})$为当前待装箱以第 P_i 种姿态放入时，箱子沿着 x、y、z 轴的长度；ϱ 为补正参数，当 $l(S_j)-l(b_{P_i})$、$w(S_j)-w(b_{P_i})$ 或 $h(S_j)-h(b_{P_i})$为 0 时，也能选出评价值最高的放置方式；F_{ij}为当前待装箱以第 P_i 姿态放入第 j 个剩余空间时的评价值，该值越大则越好。

下面举例说明，图 5b、c 所示的待装箱与剩余空



a 放置姿态1　　b 放置姿态2　　c 放置姿态3

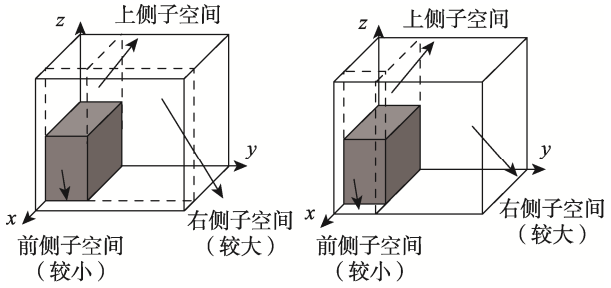
图 5　放置规则示例

Fig.5 Example of placement rules

间形状的接近程度都比图 5a 高，即图 5b~c 的评价值都比图 5a 的评价值高。图 5b、c 为同一剩余空间下 2 种放置姿态，由于图 5b 可以产生较大的剩余子空间，所以图 5b 的评价值比图 5c 高。综上可知，图 5b 为 3 种放置方式中最好的方式。

2.2.2　剩余空间的分割

通过评价函数选择箱子的放置姿态及放置空间后，在约束 6）和约束 7）的作用下，该放置空间能以 2 种方式进行空间分割，如图 6 所示。2 种分割方式都会将该剩余空间分为上侧子空间、右侧子空间和前侧子空间。在 2 种分割方式中，上侧子空间都一样，而前侧子空间和右侧子空间不一样。无论采用哪种分割方式，前侧子空间和右侧子空间均会呈现一大一小的情况，如图 6 所示。



a 分割方式1　　b 分割方式2

图 6　剩余空间分割示例

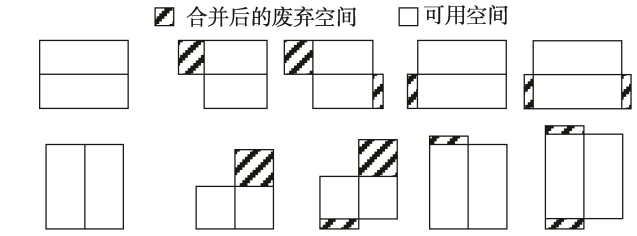
Fig.6 Example of remaining space segmentation

最适应剩余空间启发式算法的剩余空间分割规则：在空间分割后，产生的较大子空间越大越好，这样后续箱子放入塔内的可能性越大；产生的较小子空间越小越好，这样可以减少小空间无法装载导致的空间浪费。从上述思想可知，只要从 2 种剩余空间分割方式中选出较大子空间最大、较小子空间最小的分割方式即可。

以图 6 举例说明，首先，分别选出 2 种分割方式下的较大子空间（如图 6 分割方式所示，较大子空间均为右侧子空间）；其次，比较这 2 个较大子空间的体积，选择体积大的划分方式作为该剩余空间最终的分割方式，显然图 6b 的右侧子空间的体积比图 6a 的右侧子空间的体积要大，因此选择图 6b 这种剩余空间分割方式作为最终分割方案。将分割后的剩余空间纳入剩余空间集。

2.2.3　剩余空间的合并

选择分割方式后，将后续所有待装箱都尝试装入剩余空间集中的每个剩余空间，将剩余空间集中无法装入任何后续待装箱的剩余空间作为废弃空间。对废弃空间进行合并，最大化利用塔内空间，避免空间浪费。实现空间合并的基本条件：2 个空间必须相邻；2 个空间底面在同一平面上；合并后空间的体积必须变大。最适应剩余空间启发式算法的剩余空间合并方式的俯视图如图 7 所示。



a 无多余空间合并　　b 有多余空间合并

图 7　剩余空间的合并

Fig.7 Merge of remaining space

3　二维装载点启发式算法

将所有三维待装箱通过塔装载启发式算法装载成塔，现需要通过每座塔的底面确定每座塔在集装箱内的放置位置，即将三维装箱问题降为二维装箱问题进行求解，这里针对二维装箱问题提出了二维装载点启发式算法。在进行算法设计前，还需要对一些情况

进行说明，以便理解后续算法，具体如下。

1）在装载开始时，将原点 O 作为初始装载点，且装载点都有对应的装载面积。初始装载点的装载面积为整个集装箱的底面积，如图 8a 所示。

2）在装载二维待装箱时，将二维待装箱的左后角点贴合装载点进行装入，如图 8b 所示。

在二维装载点启发式算法中，单个二维待装箱装载前后主要有 5 个阶段。

1）装载点的选择。二维待装箱相当于每座塔的底面，为了使塔与塔之间更紧凑，在选择二维待装箱的装载点时，尽量选择接近原点 O 的装载点，即选择与原点 O 的直线距离最短的装载点。

2）装载点的生成。确定装载点后，将二维待装箱放入，并沿着 x、y 轴生成 2 个新的装载点，如图 8b 所示。

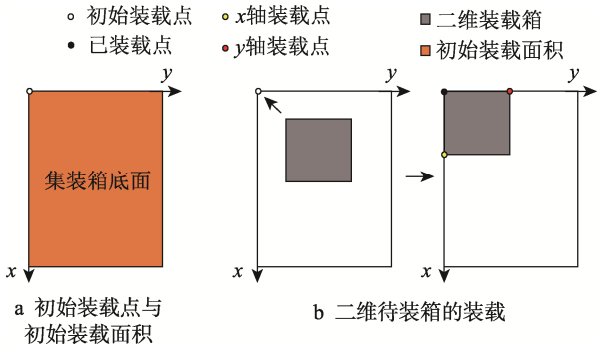


图 8　二维装载点启发式算法的说明

Fig.8 Explanation of 2D loading point algorithm

3）装载点的投影。为了使面积利用率更高，在每次生成新装载点后尝试进行投影。其中，x 轴装载点尝试沿着 y 轴负方向进行投影，y 轴装载点尝试沿着 x 轴负方向进行投影，如图 9a 所示。

4）装载点面积的修改。在前待装箱选择装载点并装入后，需要判断对已有装载点的装载面积是否存在遮挡现象。若存在遮挡，则需要对装载面积进行修改，如图 9b~c 所示。

5）重复装载点的删除。新产生的装载点在投影后，可能与已经存在的装载点重复。若重复，则去除该投影后的装载点，如图 9d 所示。

4　混合蚁群模拟退火算法

相关研究表明，蚁群算法解决组合优化问题的效果良好，但路径上信息素浓度的积累周期较长，导致算法收敛缓慢，为此引入了信息素选择更新策略和自适应信息素挥发系数。改进后的算法存在陷入局部最优的风险，为此再引入模拟退火算法进行局部搜索，以确保改进后的算法在保持快速收敛的同时，还能够兼顾全局和局部搜索能力。

混合蚁群模拟退火算法有 2 个阶段：将三维待装箱通过塔装载启发式算法装载成塔；将蚁群模拟退火算法与二维装载点启发式算法相结合，优化每座塔的装载顺序和放置姿态，进而得出每座塔的最优装箱方案。下面针对蚁群模拟退火算法进行说明。

4.1　蚁群算法

4.1.1　初始化设置

为了方便说明，将每座塔的底面作为二维待装箱进行处理。将每个二维待装箱都作为一个节点，其数

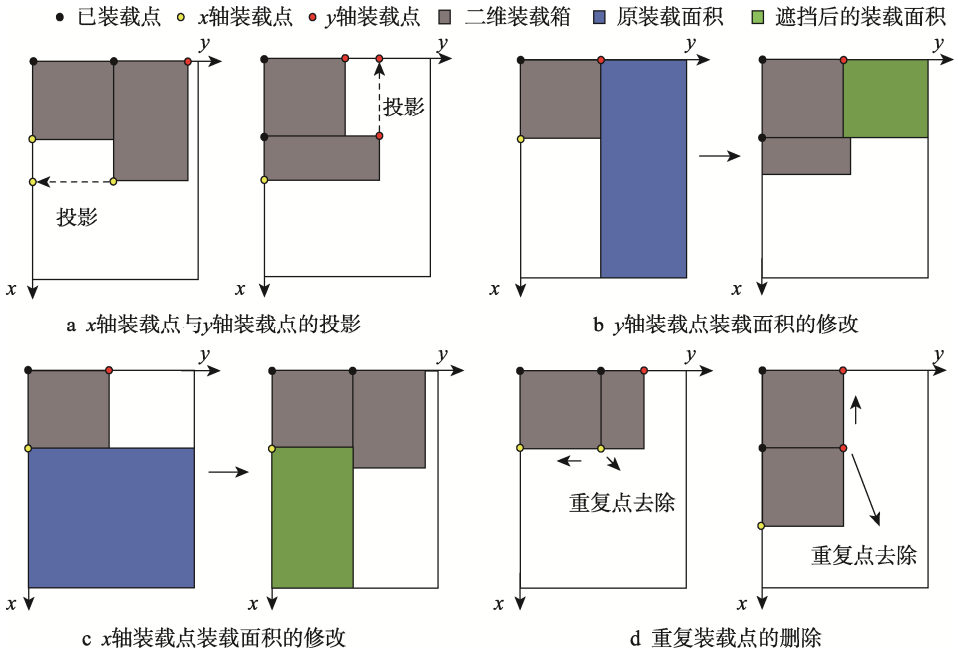


图 9　装载点的投影、装载面积的修改和重复装载点的删除

Fig.9 Projection of loading points, modification of loading area, and deletion of duplicate loading points

量为 a，蚂蚁的数量为 c；令 $\tau_i^*(t)$ 为第 t 代节点 i 上的首选箱子信息素，令 $\tau_{ij}(t)$ 为第 t 代节点 i 与节点 j 之间的后续箱子选择信息素，令 $\phi_i^{r_i}(t)$ 为第 t 代节点 i 的第 r_i 种姿态的箱子姿态选择信息素。其中，每个节点对应的二维待装箱都有 2 种摆放姿态，如图 10 所示。在 t=0 时，将每个节点上的首选箱子信息素、箱子姿态选择信息素，以及节点与节点之间的后续箱子选择信息素都设置为常数，即 $\tau_i^*(0)=\text{constant}$，$\phi_i^{r_i}(0)=\text{constant}$，$\tau_{ij}(0)=\text{constant}$。

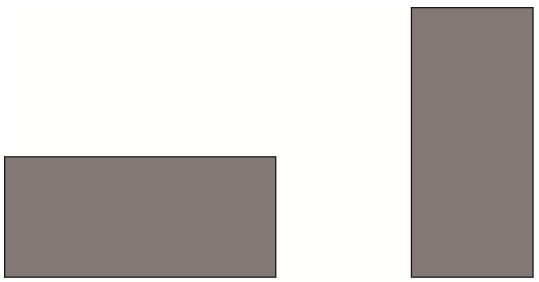


图 10　二维待装箱的 2 种放置姿态
Fig.10 Two placement postures for two-dimensional packing

4.1.2　状态转移规则

每只蚂蚁会根据每个节点上存留的首选箱子信息素浓度和首选箱子概率转移函数来选择第 1 个要走的节点，也就是选择第 1 个要装载的二维待装箱。首选箱子概率转移函数见式（3）。

$$F_i^k(t)=\tau_i^*(t)\Big/\sum_{i=1}^{a}\tau_i^*(t) \tag{3}$$

式中：$F_i^k(t)$ 表示第 t 代蚂蚁 k 选择节点 i 作为首选节点的概率。

每只蚂蚁选择第 1 个节点后，会根据节点之间的后续箱子选择信息素浓度和后续箱子选择概率转移函数选择后续节点，进而确定蚂蚁的节点序列，即二维待装箱装载顺序。后续箱子选择概率转移函数见式（4）。

$$R_{ij}^k(t)=\begin{cases}\dfrac{\tau_{ij}(t)}{\sum\limits_{j\in B_j^k(t)}\tau_{ij}(t)} & j\in B_j^k(t)\\ 0 & j\notin B_j^k(t)\end{cases} \tag{4}$$

式中：$R_{ij}^k(t)$ 为第 t 代蚂蚁 k 从节点 i 到达未曾访问节点 j 的概率；$B_i^k(t)$ 为第 t 代蚂蚁 k 到达节点 i 后，剩余未到达的节点集合。

由于每个节点对应 2 种放置姿态，且这 2 种放置姿态都存留着箱子姿态选择信息素，即根据 2 种姿态上的姿态选择信息素浓度和箱子姿态选择概率转移函数来确定每个节点的放置姿态。箱子姿态选择概率转移函数见式（5）。

$$\begin{cases}D_{ki}^{r_i}(t)=\phi_i^{r_i}(t)\Big/\sum\phi_i^{r_i}(t)\\ \sum\phi_i^{r_i}(t)=\phi_i^1(t)+\phi_i^2(t)\end{cases} \tag{5}$$

式中：$D_{ki}^{r_i}$ 为第 t 代蚂蚁 k 在节点 i 上选择第 r_i 种姿态的概率。

根据概率转移函数，可使每只蚂蚁根据信息素的浓度确定节点序列和每个节点对应选择的放置姿态。与节点序列对应，每个节点对应选择的放置姿态可以组成姿态序列，且姿态序列与节点序列构成蚂蚁的行走路径，后续可以通过二维装载点启发式算法确定路径对应装箱方案的空间利用率。

4.1.3　信息素更新规则

蚁群算法的收敛依赖信息素浓度的积累。为了更好地保留较优路径上的信息素，采用信息素选择更新策略：每代蚂蚁种群通过状态转移规则选择要走的路径，并计算路径对应装箱方案的空间利用率，只选择空间利用率前 30%蚂蚁走过的路径来更新 3 种信息素。

为了进一步在较好路径上留下更多的信息素，引入了自适应策略改进信息素挥发系数（自适应信息素挥发系数），具体操作：将当代与上代的空间利用率前 30%路径集的平均空间利用率进行比较，若当代比上代的平均空间利用率大或相等，则将挥发系数增加 1 个步长，反之则减小 1 个步长。这样可以使当代的优秀路径留下更多的信息素，以此引导后代蚂蚁选择质量较好的路径。挥发系数自适应更新公式见式（6）。

$$\begin{cases}\rho(t)=\rho(t-1)-\beta & V_u(t)<V_u(t-1)\\ \rho(t)=\rho(t-1)+\beta & V_u(t)\geqslant V_u(t-1)\end{cases} \tag{6}$$

式中：$\rho(t)$、$\rho(t-1)$ 分别表示当代和上代的信息素挥发系数；β 为固定步长；$V_u(t)$、$V_u(t-1)$ 分别表示当代和上代的空间利用率前 30%路径集的平均空间利用率。

确定信息素选择更新策略和自适应信息素挥发系数，具体信息素的更新策略如下。

1）首选箱子信息素的更新。第 t 代，计算每个蚂蚁走过路径的空间利用率后，确定空间利用率前 30%路径中的每条路径第 1 个节点的类型，对这些节点上的首选箱子信息素进行追加，并对所有节点上的首选箱子信息素进行稀释，更新公式见式（7）。

$$\tau_i^*(t+1)=\begin{cases}\rho(t)\times\tau_i^*(t)+\sigma\times V_u(t) & i\in K(t)\\ \rho(t)\times\tau_i^*(t) & i\notin K(t)\end{cases} \tag{7}$$

式中：$K(t)$ 为第 t 代空间利用率前 30%路径中的每条路径第 1 个节点类型的集合；$\sigma(\sigma>0)$ 为节点 i 在集合 $K(t)$ 中出现的次数。

2）箱子姿态选择信息素更新。第 t 代，根据空

间利用率前 30%路径中的每条路径上的每个节点对应选择的放置姿态，在对应放置姿态上追加姿态选择信息素（例如节点 5 代表二维待装箱选择第 1 种放置姿态，则此时需要对节点 5 对应第 1 种放置姿态上的箱子姿态选择信息素进行追加），并且还需要对每个节点对应的 2 种姿态上的箱子姿态选择信息素进行稀释，更新公式见式（8）。

$$\phi_i^{r_i}(t+1)=\rho(t)\times\phi_i^{r_i}+\alpha\times V_u(t) \tag{8}$$

式中：$\alpha(\alpha\geq 0)$为空间利用率前 30%的蚂蚁群体中，经过节点 i 时选择第 r_i 姿态的蚂蚁数量。

3）后续箱子选择信息素的更新。第 t 代，确定空间利用率前 30%路径中的每条路径的节点顺序后，在对应节点之间追加后续箱子选择信息素，并且还需要对节点与节点之间的后续箱子选择信息素进行稀释，更新公式见式（9）。

$$\begin{cases}\tau_{ij}(t+1)=\rho(t)\times\tau_{ij}(t)+\Delta\tau_{ij}(t)\\ \Delta\tau_{ij}(t)=\sum\Delta\tau_{ij}^{k}(t),\ k\in N(t)\\ \Delta\tau_{ij}^{k}(t)=\delta\times V_u^{k}(t)\end{cases} \tag{9}$$

式中：$\Delta\tau_{ij}(t)$表示节点 i 与节点 j 之间追加的信息素；$\Delta\tau_{ij}^{k}(t)$表示蚂蚁 k 从节点 i 到达节点 j 后留下的信息素；$N(t)$表示空间利用率前 30%的路径；δ 为比例系数；$V_u^{k}(t)$为蚂蚁 k 走过路径的空间利用率。

4.1.4　蚁群算法步骤

蚁群算法的具体流程如图 11 所示，具体步骤如下。

1）初始化蚁群算法相关参数。

2）根据当前 3 种信息素的浓度，由式（3）~（5）确定每只蚂蚁的路径。

3）利用二维装载点启发式算法对每只蚂蚁的路径进行解码，并计算各个蚂蚁走过路径对应的空间利用率。

4）根据选择更新策略，计算当代空间利用率前 30%的路径平均值，并与上代空间利用率前 30%的路径平均值进行比较，再通过式（6）确定当代信息素挥发系数。

5）比较当代最优路径与全局最优路径的空间利用率。若当代最优路径的空间利用率比全局最优路径大，则将当代最优路径视为新的全局最优路径。

6）根据式（7）~（9）更新 3 种信息素的浓度。

7）重复步骤 2）至步骤 6），直至运行结束，输出全局最优路径。

4.2　模拟退火算法

在蚁群算法中，每代的每个蚂蚁会根据路径上的信息素水平来确定自己要选择的路径。然而，由于蚁群算法中加入了信息素选择更新策略及自适应信息素挥发系数，因此存在陷入局部最优的风险。为了避免该风险，需要在每代的每个蚂蚁计算其走过路径对应的空间利用率后，融入模拟退火算法对空间利用率前 30%的优秀路径进行局部搜索，试图寻找更加优秀的路径，这样可以有效提高算法的性能，避免陷入局部最优解。

4.2.1　模拟退火算法描述

为了方便说明，将蚂蚁走过的路径中的节点序列和姿态序列称为路径序列。单个蚂蚁的路径序列模拟退火操作流程如图 12 所示，算法有内外 2 个循环。

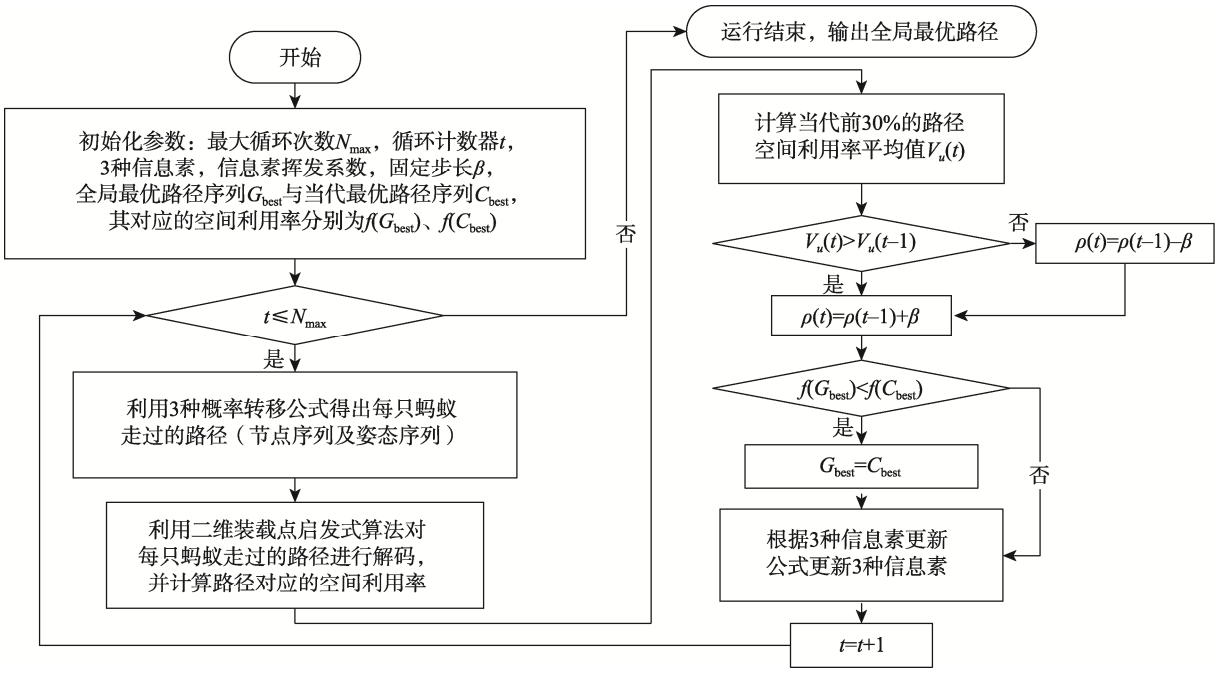


图 11　蚁群算法流程

Fig.11 Flowchart of ant colony algorithm

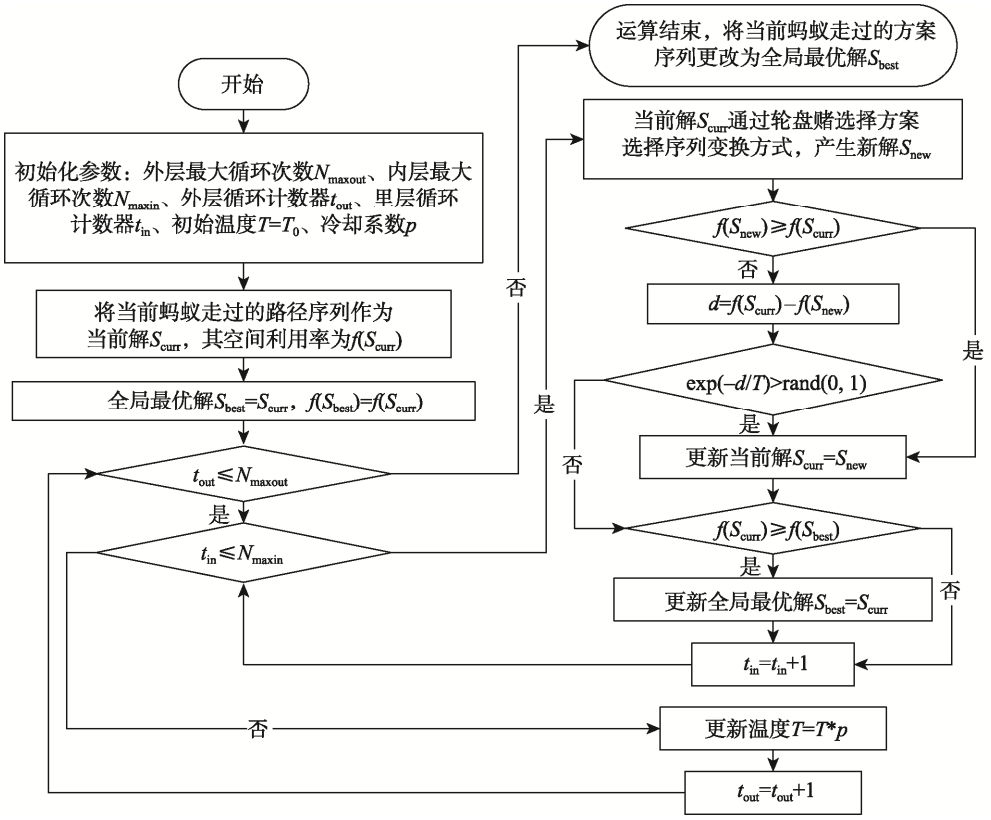


图 12　模拟退火算法流程

Fig.12 Flowchart of simulated annealing algorithm

每次内循环都需对蚂蚁当前路径序列随机选择 3 种序列结构的变换之一进行变换，若变换后路径序列空间利用率增大，则将变换后路径序列作为新当前路径序列，反之，则根据 Metroplis 准则，以一定概率接受变换后的路径序列作为新当前路径序列[18]；将新当前路径序列的空间利用率与最优路径序列进行比较，若新当前路径序列的空间利用率比最优路径序列的空间利用率大，则将新当前路径序列作为最优路径序列，反之亦然，直至内循环结束，再进行降温操作；在每次外循环中保留每次内循环的最优路径序列，直至外循环结束，最终输出该蚂蚁通过模拟退火操作后的最优路径序列。

4.2.2　序列结构变换

采用模拟退火算法使算法整体不会陷入局部最优的关键是将原优秀路径序列随机从 3 种序列结构变换方式中选择 1 种方式进行变换，即尝试在原优秀路径序列的领域内寻找空间利用率更高的优秀路径，具体变换方式如下。

1）单点交叉结构（被选择概率 40%）。随机在原有序列中插入 1 个点，然后将该点的前后序列进行互换，如图 13a 所示。

2）双点交叉结构（被选择概率 40%）。随机在原有序列中插入 2 个点，然后将位置靠前的插入点的前序列与位置靠后的后序列进行互换，如图 13b 所示。

3）逆转结构（被选择概率 20%）。随机在原有序列中插入 2 个点，将 2 个点之间的序列进行逆转，如图 13c 所示。

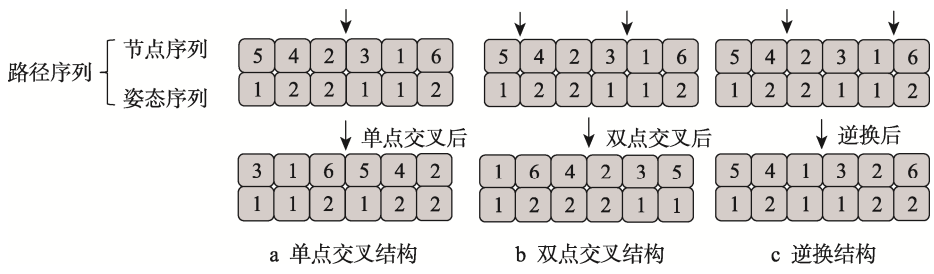


图 13　3 种序列变换结构

Fig.13 Three sequence transformation structures

4.3　混合蚁群模拟退火算法整体步骤

混合蚁群模拟退火算法流程如图 14 所示，具体步骤如下。

1）读取三维待装箱和集装箱的数据。主要包括三维待装箱的长度l_i/mm、宽度w_i/mm、高度h_i/mm、质量 m_i/kg，以及集装箱的长度L/mm、宽度W/mm、高度H/mm、承载质量 m/kg。

2）将三维待装箱通过塔装载启发式算法装载成塔。

3）初始化蚁群算法及模拟退火算法的相关参数。主要包括蚁群算法的最大循环次数 $N_{\max}$，3 种信息素浓度$\tau_i^*(0)$、$\tau_{ij}(0)$、$\phi_i^{r_i}(0)$，信息素挥发系数$\rho(0)$，固定步长 β，全局最优路径序列 G_{best} 和当代最优路径序列 C_{best}，以及模拟退火算法的内外层最大循环次数 $N_{\max in}$、$N_{\max out}$，初始温度 T_0，降温速率 p 等。

4）根据 3 种概率转移函数，见式（3）~（5），确定当代每只蚂蚁走过的节点序列和姿态序列。

5）将每只蚂蚁走过的节点序列和姿态序列通过二维装载点启发式算法进行解码，并计算每只蚂蚁走过路径的空间利用率。

6）采用 4.2 节提出的模拟退火算法对当代空间利用率前 30%的优秀路径进行局部搜索。

7）采用式（6）确定当代信息素挥发系数，并利用式（7）~（9）更新 3 种信息素，记录截至本代空间利用率最大的最优路径序列，并重复步骤 4）至步骤 7），直至循环结束，输出最优路径序列，即得到每座塔的最优装箱顺序和最优放置姿态。

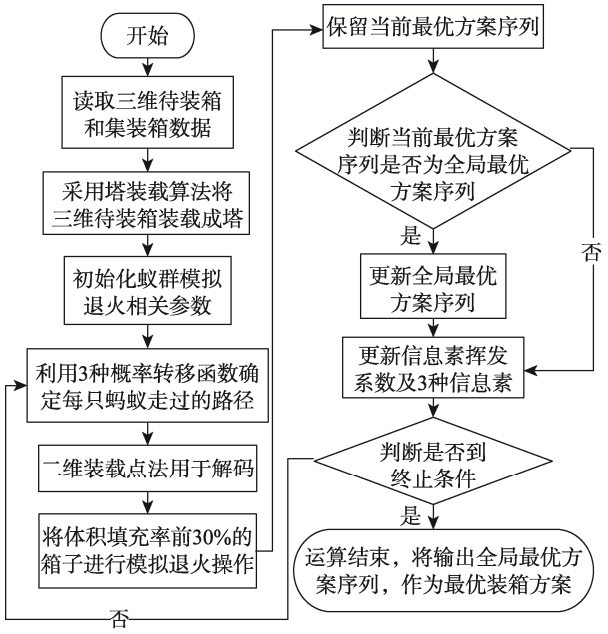


图 14　混合蚁群模拟退火算法流程

Fig.14 Flowchart of hybrid ant colony simulated annealing algorithm

5　实验测试

5.1　参数设置

本文算法采用 Matlab 2021a 实现，计算机配置：Intel® Core™ i7-7700HQ CPU @ 2.80 GHz ，内存为 8 GB，操作系统为 Windows 10。蚁群算法和模拟退火算法涉及的参数：蚂蚁种群数量 c，为二维待装箱数量的 3 倍；初始信息素挥发系数$\rho(0)$，为 0.6；固定步长 β，为 0.05；3 种信息素的初始值$\tau_i^*(0)$、$\tau_{ij}(0)$、$\phi_i^{r_i}(0)$，均为 1；蚁群算法最大循环次数 $N_{\max}$，为 100；模拟退火外层最大循环次数 $N_{\max\ out}$，为 30；内层循环次数 $N_{\max\ in}$，为 15；初始温度 T_0，为 10 ℃；降温速率 p，为 0.9。

5.2　算例比较

基于 Bischoff 在文献[19]中提出的箱子尺寸数据生成方法，利用计算机随机生成 10 组数据（BR1—BR10），共计 10 种类型的数据（每种类型有 25 组测试数据），且这 10 种类型数据中箱子种类由弱异构型到强异构型，可以反映算法的性能。这里将文献[20]提出的 3D-RSO 算法、混合传统模拟退火算法（HSA 算法）及混合传统蚁群算法（HAC 算法）进行比较分析。基于 10 种类型的数据，本文的混合蚁群模拟退火算法（HACSA 算法）与其他算法的平均空间利用率见表 1。

1）由于 3D-RSO 为启发式算法，将待装箱按照一定规则进行排序，直接进行装箱，所以相较于 HSA、HAC、HACSA 混合启发式算法，其空间利用率更低。

2）随着待装箱种类的增加，容器的废弃空间随之增多，所有算法的空间利用率都有所降低，求解问题的难度也随之增大。

3）从平均空间利用率来看，本文提出的 HACSA 算法的空间利用率高于其他算法。说明本文提出的算法具有可行性，且适用于解决三维装箱问题。

列出了 HACSA 对于 BR1—BR10 这些装箱数据的运行时间，以及空间利用率的最小值、最大值、平均值，见表 2。这里的运行时间为程序实际运行时间。从表 2 可以看出，算法整体的运行时间与箱子种类呈正相关，原因：随着箱子种类的增加，通过塔装载算法得到的剩余空间增多，选出箱子最适应放置姿态及最适应剩余空间的时间延长，导致算法整体的运行时间延长；随着箱子种类的增加，塔底面的种类也会增加，即二维待装箱的种类增加，采用蚁群模拟退火算法和二维装载点启发式算法寻找最优方案的难度随之增加，算法整体的运行时间随之增加。

表 1 5 种算法的空间利用率比较
Tab.1 Comparison of spatial loading rates among five algorithms

算法	填充率/%										
	BR1	BR2	BR3	BR4	BR5	BR6	BR7	BR8	BR9	BR10	平均
3D-RSO[20]	84.38	84.29	84.63	83.23	82.87	81.82	80.73	79.19	77.23	74.68	81.31
HSA	91.63	91.71	91.50	91.14	90.71	90.05	89.63	88.23	87.25	84.34	89.62
HAC	91.89	91.63	91.45	91.56	90.72	89.93	89.88	88.60	86.18	83.93	89.58
HACSA	92.28	93.37	92.49	92.14	91.37	90.70	90.27	89.73	88.56	88.32	90.92

表 2 混合蚁群模拟退火算法测试数据
Tab.2 Test details of hybrid ant colony simulated annealing algorithm

测试实例	运行时间/s			填充率/%		
	最小值	最大值	平均值	最小值	最大值	平均值
BR1	5.98	35.32	17.21	90.03	95.16	92.28
BR2	11.32	59.46	32.14	88.71	94.84	93.37
BR3	16.61	101.31	52.37	89.14	94.81	92.49
BR4	21.17	191.28	107.34	88.31	93.18	92.14
BR5	29.27	236.32	112.53	89.42	92.53	91.37
BR6	27.83	297.51	157.28	87.93	92.41	90.70
BR7	39.93	411.31	237.46	86.37	91.73	90.27
BR8	57.31	487.28	285.07	85.72	92.21	89.73
BR9	91.23	534.63	386.54	84.27	90.46	88.56
BR10	131.42	631.57	507.28	79.47	89.84	88.32
平均	43.21	298.60	189.52	86.94	92.72	90.92

5.3 实例验证

现采用云南某公司提供的货物订单信息进行实例验证，订单一共有 15 种货物，如表 3 所示，且采用的货车车厢的尺寸（长度×宽度×高度）为 6 310 mm× 2 450 mm× 2 675 mm，载质量为 1 000 kg。这些货物及车厢均符合前面的假设，将本文的 HACSA 算法与 HAC 算法、HSA 算法进行比较分析。

如图 15 所示，本文的 HACSA 算法在第 7 代时得到最佳的空间利用率，而 HSA 算法、HAC 算法分别在第 59 代、第 142 代得到各自的最佳空间利用率，即本文的 HIACSA 算法相较于 HSA 算法达到最佳空间利用率的迭代次数减少了 88.14%，相较于 HAC 算法达到最佳空间利用率的迭代次数减少了 95.07%，说明本文的算法具有更快的收敛速度。本文的 HACSA 算法的最佳空间利用率为 94.06%，而 HSA 算法、HAC 算法的最佳空间利用率分别为 92.32%、93.18%，即本文的 HACSA 算法相较于 HSA 算法的最佳空间利用率提升了 1.74%，相较于 HAC 算法的最佳空间利用率提升了 0.88%，说明本文算法具有不错的收敛精度。综上可知，本文的 HACSA 算法通过设置多种信息素、稀释系数，以及加入模拟退火算法，在保证收敛精度的同时，也保证了收敛速度，更加适合于大规模集装箱的装箱问题求解。

在装箱过程中，所有的货物需要使用 2 辆相同型号的货车才能全部容纳。值得注意的是，在第 1 节货车车厢装满后，才会开始将货物装入第 2 辆货车，因此第 2 节货车车厢的装载情况无法准确反映算法的性能。为了展示本文 HACSA 算法在此实例中的求解结果，给出了针对此实例求解出的第 1 节车厢的货物装载效果，如图 16 所示。

表 3　15 种货物详细信息
Tab.3 Detailed information of 15 types of goods

序号	长度/mm	宽度/mm	高度/mm	数量	质量/kg
1	445	233	550	50	4
2	455	230	543	50	5
3	528	262	287	250	4
4	455	243	570	75	8
5	475	345	306	50	6
6	398	235	538	5	3
7	482	260	278	5	2
8	505	370	176	250	4
9	340	211	389	15	3
10	453	240	553	30	5
11	463	248	556	30	4
12	391	242	536	20	6
13	497	360	185	45	3
14	457	231	540	35	4
15	488	253	282	20	4

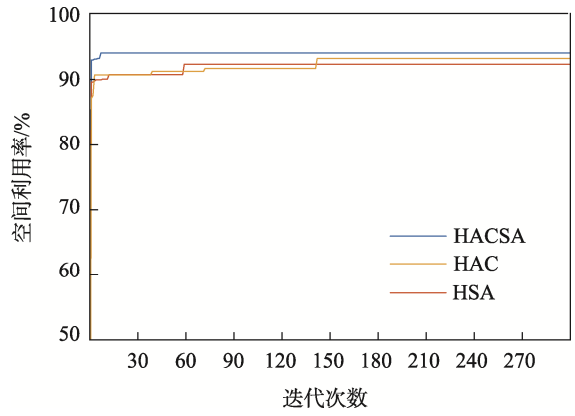


图 15　3 种算法迭代对比
Fig.15 Comparison chart of three algorithm iterations

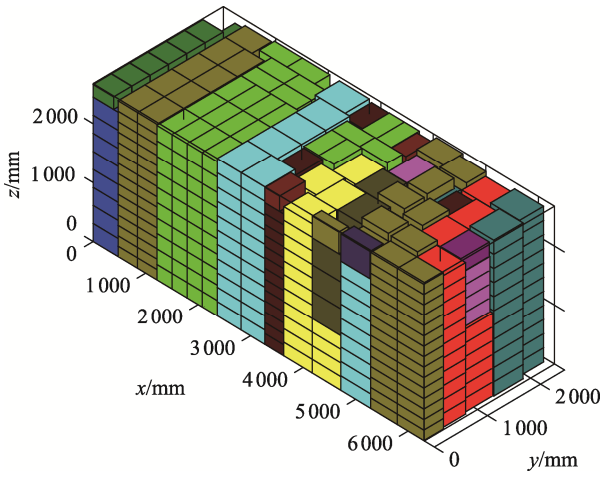


图 16　货物的三维装载图
Fig.16 3D loading diagram of goods

6　结语

针对三维装箱问题的求解，本文采用组合启发式算法结合蚁群模拟退火算法，经过从弱异构型到强异构型多组算例的测试，取得了不错的装箱效果，证明本文的蚁群模拟退火算法无论是全局搜索能力还是局部搜索能力都很强。HACSA 算法在面对弱异构型算例（BR1—BR5）时的平均空间利用率可以达到 92.33%。虽然面对强异构型算例（BR6—BR10）的求解时间明显延长，但是其空间利用率相较于具有同种约束的算法提高了 6.45%。基于目前三维装箱问题的多样性和随机性，在面对不规则箱体、不规则容器和特殊实际约束时，需要进行更深入的研究。

参考文献：

[1] DYCKHOFF H. A Typology of Cutting and Packing Problems[J]. European Journal of Operational Research, 1990, 44: 145-159.

[2] WÄSCHER G, HAUßNER H, SCHUMANN H. An Improved Typology of Cutting and Packing Problems[J]. European Journal of Operational Research, 2007, 183(3): 1109-1130.

[3] GEORGE J A, ROBINSON D F. A Heuristic for Packing Boxes into a Container[J]. Computers & Operations Research, 1980, 7(3): 147-156.

[4] ELEY M. Solving Container Loading Problems by Block Arrangement[J]. European Journal of Operational

Research, 2002, 141(2): 393-409.

[5] FANSLAU T, BORTFELDT A. A Tree Search Algorithm for Solving the Container Loading Problem[J]. INFORMS Journal on Computing, 2010, 22(2): 222-235.

[6] MENGHANI D, GUHA A. Packing Boxes into Multiple Containers Using Genetic Algorithm[J]. Journal of the Institution of Engineers (India): Series C, 2016, 97(3): 441-450.

[7] BORTFELDT A, GEHRING H. A Hybrid Genetic Algorithm for the Container Loading Problem[J]. European Journal of Operational Research, 2001, 131(1): 143-161.

[8] MOURA A, OLIVEIRA J F. A GRASP Approach to the Container-Loading Problem[J]. IEEE Intelligent Systems, 2005, 20(4): 50-57.

[9] ARAYA I, MOYANO M, SANCHEZ C. A Beam Search Algorithm for the Biobjective Container Loading Problem[J]. European Journal of Operational Research, 2020, 286(2): 417-431.

[10] 卓雪艳, 何勇, 吴灵芳, 等. 大型车辆配载与装箱优化方法及系统设计研究[J]. 机械设计与制造, 2019(10): 20-23.
ZHUO X Y, HE Y, WU L F, et al. Research on Cargo Allocation and Vehicle Loading Problem and System Design[J]. Machinery Design & Manufacture, 2019(10): 20-23.

[11] 李伟. 基于改进随机森林算法的多规格货物装载研究[D]. 淮南: 安徽理工大学, 2020: 32-38.
LI W. Research on Multi-Specification Cargo Loading Based on Improved Random Forest Algorithm[D]. Huainan: Anhui University of Science & Technology, 2020: 32-38.

[12] 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[J]. 计算机学报, 2012, 35(12): 2553-2561.
ZHANG D F, PENG Y, ZHANG L L. A Multi-Layer Heuristic Search Algorithm for Three Dimensional Container Loading Problem[J]. Chinese Journal of Computers, 2012, 35(12): 2553-2561.

[13] 张长勇, 吴智博, 王艳芳. 基于 K-means 的航空行李快速装箱算法[J]. 包装与食品机械, 2019, 37(3): 38-42.
ZHANG C Y, WU Z B, WANG Y F. Fast Container Loading Algorithm for Airline Luggage Registration Based on K-Means Clustering[J]. Packaging and Food Machinery, 2019, 37(3): 38-42.

[14] 张长勇, 张倩倩, 翟一鸣, 等. 基于改进粒子群算法的航空行李在线装载优化[J]. 包装工程, 2021, 42(21): 200-206.
ZHANG C Y, ZHANG Q Q, ZHAI Y M, et al. Online Luggage Loading Optimization Based on Improved Particle Swarm Algorithm[J]. Packaging Engineering, 2021, 42(21): 200-206.

[15] 刘胜, 沈大勇, 商秀芹, 等. 求解三维装箱问题的多层树搜索算法[J]. 自动化学报, 2020, 46(6): 1178-1187.
LIU S, SHEN D Y, SHANG X Q, et al. A Multi-Level Tree Search Algorithm for Three Dimensional Container Loading Problem[J]. Acta Automatica Sinica, 2020, 46(6): 1178-1187.

[16] 陈元文. 基于优先保持策略遗传算法的三维装箱问题[J]. 包装工程, 2021, 42(15): 211-218.
CHEN Y W. Three-Dimensional Container Loading Problem Based on Genetic Algorithm with Priority Retention Strategy[J]. Packaging Engineering, 2021, 42(15): 211-218.

[17] 邢志伟, 侯翔开, 李彪, 等. 基于动态四叉树搜索的民航行李车码放算法[J]. 北京航空航天大学学报, 2022, 48(12): 2345-2355.
XING Z W, HOU X K, LI B, et al. Civil Aviation Luggage Cart Stacking Algorithm Based on Dynamic Quadtree Search[J]. Journal of Beijing University of Aeronautics and Astronautics, 2022, 48(12): 2345-2355.

[18] KIRKPATRICK S, GELATT J C D, VECCHI M P. Optimization by Simulated Annealing[J]. Science, 1983, 220(4598): 671-680.

[19] BISCHOFF E E, RATCLIFF M S W. Issues in the Development of Approaches to Container Loading[J]. Omega, 1995, 23(4): 377-390.

[20] 尚正阳, 顾寄南, 唐仕喜, 等. 高效求解三维装箱问题的剩余空间最优化算法[J]. 计算机工程与应用, 2019, 55(5): 44-50.
SHANG Z Y, GU J N, TANG S X, et al. Efficient Residual-Space-Optimization Algorithm for Three Dimensional Container Loading Problem[J]. Computer Engineering and Applications, 2019, 55(5): 44-50.