Journal of System Simulation

Volume 36 | Issue 3 Article 18

3-15-2024

Research on Hybrid Solution Algorithm for Layout Problem of Rectangular Parts with Multiple Constraints

Ye Liu
School of Mechanical Engineering, Jiangnan University, Wuxi 214122, China, 13151006307@163.com

Weixi Ji
School of Mechanical Engineering, Jiangnan University, Wuxi 214122, China; Jiangsu Provincial Key Laboratory of Food Manufacturing Equipment, Wuxi 214122, China, ji_weixi@126.com

Xuan Su
School of Mechanical Engineering, Jiangnan University, Wuxi 214122, China

Hongxuan Zhao
School of Mechanical Engineering, Jiangnan University, Wuxi 214122, China

Research on Hybrid Solution Algorithm for Layout Problem of Rectangular Parts with Multiple Constraints

Abstract

Abstract: A hybrid algorithm based on a cutting and matching algorithm and an improved ant colony algorithm was proposed to solve the layout problem of rectangular parts in the process of wood and glass blanking. A layout optimization model was established to maximize the mean square utilization and the remaining processing time; the ant colony algorithm was used as the layout sequence algorithm to determine the layout sequence of some parts and meet the processing time constraint. In order to improve the search efficiency of the ant colony algorithm, an adaptive pheromone updating strategy was proposed, and a hybrid mutation strategy based on genetic mutation and 2-opt mutation was introduced to enhance the local search capability. For the arrangement of parts on the workblank, in order to improve the mean square utilization of the workblank and meet the guillotine constraint, a cutting and matching algorithm was proposed to optimize the layout of rectangular parts. Finally, the improved algorithm was compared with other optimization algorithms by international standard test cases and actual enterprise cases, and the effectiveness of the proposed hybrid algorithm was verified.

Keywords

layout of rectangular parts, ant colony algorithm, guillotine, multiple constraints, hybrid mutation strategy

Recommended Citation

Liu Ye, Ji Weixi, Su Xuan, et al. Research on Hybrid Solution Algorithm for Layout Problem of Rectangular Parts with Multiple Constraints[J]. Journal of System Simulation, 2024, 36(3): 743-755.

第 36 卷第 3 期 系统仿真学报© Vol. 36 No. 3
2024 年 3 月 Journal of System Simulation Mar. 2024

多约束下矩形件排样问题的混合求解算法研究

刘野[1]，吉卫喜[1,2*]，苏璇[1]，赵宏轩[1]
(1. 江南大学 机械工程学院，江苏 无锡 214122；2. 江苏省食品制造装备重点实验室，江苏 无锡 214122)

摘要：针对板材和玻璃下料过程中存在的矩形件排样问题，提出了一种基于分割匹配算法与改进蚁群算法的混合算法进行求解。建立了以最大化均方利用率和剩余加工时间为目标的排样优化模型；利用蚁群算法作为排样顺序算法确定部分零件的排样顺序以满足零件的加工时间限制，为了提高蚁群算法搜索效率，提出了自适应信息素更新策略，引入基于遗传变异和2-opt变异的混合变异策略来增强局部搜索能力。针对于零件在毛坯上位置的排布问题，为提高毛坯的均方利用率同时又满足一刀切约束条件，提出分割匹配算法进行矩形件排布优化。将改后的算法与其他优化算法用国际标准测试案例和企业实际案例进行对比分析，验证了所提混合算法的有效性。

关键词：矩形件排样；蚁群算法；一刀切；多约束；混合变异策略
中图分类号：TP391.9；TP931　　文献标志码：A　　文章编号：1004-731X(2024)03-0743-13
DOI: 10.16182/j.issn1004731x.joss.22-1258
引用格式: 刘野, 吉卫喜, 苏璇, 等. 多约束下矩形件排样问题的混合求解算法研究[J]. 系统仿真学报, 2024, 36(3): 743-755.
Reference format: Liu Ye, Ji Weixi, Su Xuan, et al. Research on Hybrid Solution Algorithm for Layout Problem of Rectangular Parts with Multiple Constraints[J]. Journal of System Simulation, 2024, 36(3): 743-755.

Research on Hybrid Solution Algorithm for Layout Problem of Rectangular Parts with Multiple Constraints

Liu Ye[1], *Ji Weixi*[1,2*], *Su Xuan*[1], *Zhao Hongxuan*[1]
(1. School of Mechanical Engineering, Jiangnan University, Wuxi 214122, China;
2. Jiangsu Provincial Key Laboratory of Food Manufacturing Equipment, Wuxi 214122, China)

Abstract: A hybrid algorithm based on a cutting and matching algorithm and an improved ant colony algorithm was proposed to solve the layout problem of rectangular parts in the process of wood and glass blanking. *A layout optimization model was established to maximize the mean square utilization and the remaining processing time; the ant colony algorithm was used as the layout sequence algorithm to determine the layout sequence of some parts and meet the processing time constraint. In order to improve the search efficiency of the ant colony algorithm, an adaptive pheromone updating strategy was proposed, and a hybrid mutation strategy based on genetic mutation and 2-opt mutation was introduced to enhance the local search capability.* For the arrangement of parts on the workblank, in order to improve the mean square utilization of the workblank and meet the guillotine constraint, *a cutting and matching algorithm was proposed to optimize the layout of rectangular parts.* Finally, the improved algorithm was compared with other optimization algorithms by international standard test cases and actual enterprise cases, and the effectiveness of the proposed hybrid algorithm was verified.

收稿日期：2022-10-20　　修回日期：2022-12-16
基金项目：山东省重大科技创新工程基金(2019JZZY020111)
第一作者：刘野(1996-)，男，满族，硕士生，研究方向为智能制造，排样技术。E-mail：13151006307@163.com
通讯作者：吉卫喜(1961-)，男，教授，博导，博士，研究方向为智能化制造技术与系统、智能装备数字化与可靠性设计等。E-mail：ji_weixi@126.com

Keywords: layout of rectangular parts; ant colony algorithm; guillotine; multiple constraints; hybrid mutation strategy

0 引言

板材和玻璃下料过程中的多规格毛坯下料问题也称多规格毛坯矩形零件排样问题是典型的组合优化问题，在给定毛坯组内合理安排零件的顺序和布局，达到满足一刀切加工条件的同时实现毛坯的利用率最大的目的。一刀切约束定义为所有零件必须是边到边切割而成，通过垂直或者水平切割完成零件的加工成型，通常在玻璃和木材切割行业生产加工需要满足一刀切约束。在满足工艺条件下，提高企业材料资源利用率，减少材料浪费，能够很大程度上降低资源成本，在全球资源日益紧缺情况下能够显著提高企业经济效益。

满足一刀切约束的多规格毛坯的矩形零件排样问题属于NP-Hard问题。目前为止国内外已经将启发式算法和部分智能优化算法应用于解决该问题上。文献[1]针对该问题提出了一种启发式算法BFS (best-Fit with stacking)，该算法从条带打包程序开始，将这些条带重新包装到面积递减的箱中。文献[2]针对于矩形装箱问题提出了蚁群算法与水平择优匹配算法相结合的混合算法，开拓了排样新思路。文献[3]叉树算法与矩形拼接算法相结合解决了一刀切约束下不同规模矩形件排样问题。文献[4]选择遗传算法与贪心算法混合使用优化排样顺序，又提出后检测策略排布零件以满足一刀切需求，算法结果相比文献算法得到较高的时间效率及求解质量。文献[5]针对变尺寸箱问题提出了两种源于动态规划思想的启发式方法，用于解决考虑一刀切约束和旋转的切割问题。文献[6]为二维变尺寸箱装箱问题(2-dimension variable-sized bin packing problem, 2DVSBPP)设计了一个混合启发式算法，提出了一种混合装箱算法，该算法与迭代模拟退火运行和二分搜索结合使用。文献[7]将 2DVSBPP的搜索空间划分为集合并对集合施加顺序，然后使用目标驱动的方法来利用这个划分解空间的特殊结构求解该问题。文献[8]考虑单矩形问题，提出了伪多项式和紧整数非线性公式，采用基于自下而上打包方法，实验结果表明所提出的模型及方法能够在合理的处理时间内获得最优或接近最优的解决方案。文献[9]提出了一种基于目标驱动和重建启发式方法，该算法与现有文献中算法进行对比后获得极具优势的排样效果。在企业大批量订单制造过程中，不同订单的送货日期和不同零件的加工工艺决定了零件在下料时可能有不同的时间限制。目前很少有这方面的文献，类似的问题有单一尺寸箱的带有截止日期的装箱问题，文献[10]介绍了带有到期日期的二维装箱问题，其中为每个矩形分配了一个到期日期，目标是最小化箱的数量和矩形的最大延迟。文献[11]考虑了正交二维装箱问题的双目标扩展，其中物品与到期日相关联，并且希望最小化物品的装箱数和最大延迟。文献[12]研究了物流行业带时间窗的装箱问题，并提出了一种最佳拟合启发式算法和最短路径解码器来解决该问题。

研究一刀切约束下带有加工时间限制的二维变尺寸箱排样问题(2DVSBPP with processing time and guillotine constraints, 2DVSBPP-PT-G)，因为在实际生产过程中，不同零件虽然在同样的毛坯上加工生成，但是由于客户的工艺要求和供货期的不同，为了不影响后续工艺的处理，每个零件都有一定的工序加工时间，所以其目的是在保证零件的加工时间不超过分配的加工时间和一刀切加工约束的情况下提高毛坯的利用率和零件剩余加工时间，对此本文提出了相应的解决方法。首先给出2DVSBPP-PT-G的数学模型，在信息素更新策略上改进蚁群算法确定零件排放顺序，同时提出了混合变异机制改善该算法，然后提出搜索匹配法确定零件在毛坯上的排放位置，最后利用

DOI: 10.16182/j.issn1004731x.joss.22-1258

该混合算法求出最优解。

1 问题描述与建模

2DVSBPP-PT-G 问题定义如下，给定一个零件集合 $N=\{1,2,\cdots,n\}$，包含 n 个零件，每个零件 i 都分配特定的加工时间 t_{si}，它定义了加工完成矩形零件 i 的最大时长。此外，有毛坯集合 $M=\{1,2,\cdots,m\}$，其中包含 m 个不同规格的毛坯，且对于任意一个零件，至少存在一个毛坯的长和宽要大于该零件的长宽，以此来保证每个零件都能排放在毛坯上。毛坯的加工时间由排布在毛坯上的零件的周长之和和平均加工速度来确定。由于受一刀切加工限制，所以毛坯的加工路径总长度不应该包括毛坯最外围四边长度，即在零件周长之和的基础上减去毛坯周长即为实际加工长度。在大批量下料过程中，下料生成的木材和玻璃零件往往需要后续的打孔、打胶、抛光等工序加工，当产品需求量大，订单时间周期短，零件加工耗时较长，为了不影响后续工序的进行和整体订单的工期，通过零件加工时间限制可以有效控制零件的加工顺序，因此引入零件剩余加工时间概念指导排样过程。毛坯的加工时间定义为毛坯在加工完成后所在的时间节点与第一个毛坯开始加工时的时间节点的时间差值，零件的剩余加工时间定义为零件的设定加工时间与零件所在毛坯的加工时间的差值。目标是在满足一刀切约束下保证零件在规定时间限制下加工完成，并且尽可能的提高毛坯利用率和最大剩余完成时间。

仅用最大化利用率(最小化毛坯使用数量)并不足以指导搜索过程[10]，因为大量的解决方案使用相同的毛坯数量。但是每种解决方案可能拥有不同的质量，因此需要更加准确的适应度计算函数来指导搜索。本文将毛坯利用率的平方的平均值定义为均方利用率[10]，以此指导算法搜索方向更有利于搜索到高质量的解决方案。

设毛坯 j 利用率函数为 U_j，均方利用率为 U_{ms}，毛坯 j 的加工时间 T_j，最大剩余加工时间函数为 $T_{\max}$，则有式(1)~(4)：

$$U_j=\frac{\sum_{i=1}^{n}X_{ij}R_i}{R_j}$$

$$\text{s.t.}\begin{cases}\sum_{i=1}^{n}X_{ij}R_i\leqslant R_j\\ X_{ij}\in\{0,1\}\\ j\in Q\\ i=1,2,\cdots,n\end{cases}\tag{1}$$

式中：n 为矩形零件个数；i 为零件编号；当零件 i 排放在毛坯 j 上时，$X_{ij}=1$，否则 $X_{ij}=0$；R_i 为零件 i 的面积；j 为毛坯编号；R_j 为毛坯 j 的面积；Q 为有效解决方案中使用毛坯的编号集合。

$$U_{\text{ms}}=\frac{\sum_{j\in Q}U_j^{\,2}}{R}$$

$$\text{s.t.}\begin{cases}U_j\in(0,1]\\ M\in(0,m]\end{cases}\tag{2}$$

式中：m 为毛坯个数；M 为有效解决方案中使用毛坯的数量。

$$T_j=\frac{\sum_{i=1}^{n}X_{ij}C_i-C_j}{V}$$

$$\text{s.t.}\begin{cases}V>0\\ j\in Q\end{cases}\tag{3}$$

式中：V 为平均加工速度；C_i 为零件周长；C_j 为零件所在毛坯周长。

$$T_{\max}=\sum_{k=1}^{M}\sum_{i=1}^{n}X_{ik}(t_{si}-kT_k)$$

$$\text{s.t.}\begin{cases}t_{si}-kT_k\geqslant 0\\ k=1,2,\cdots,M\\ k\in Q\end{cases}\tag{4}$$

式中：k 为有效解决方案中毛坯编号；$d_i-kT_k\geqslant 0$ 表示每个零件的加工完成时间不能超过零件的要求完成时间。

由于本算法有 2 个优化目标：主优化目标均方利用率和辅助优化目标剩余加工时间。为了方便算法运算，需要对 2 个目标进行归一化处理并根据目标重要性赋予一定的权重。权重的设置根

据实际生产情况来设置，如果不涉及零件紧急加工情况可设置权重为1，即只考虑材料利用率问题；否则可以根据需求适当减小权重值。其中涉及到剩余加工时间的极值设定，根据理想的情况下为每个零件都在和它规格相同的毛坯上加工，此时的剩余加工时间为最大值。理论最大剩余加工时间和最终的适应度函数分别为

$$T_{\mathrm{L\,max}}=\sum_{k=1}^{n}\sum_{i=1}^{k}(t_{si}-\frac{C_i}{V}) \tag{5}$$

$$F=\gamma U_{\mathrm{ms}}+(1-\gamma)\frac{T_{\max}}{T_{\mathrm{L\,max}}} \tag{6}$$

式中：γ为权重参数，取值为(0,1)。

2 排样顺序算法

蚁群算法(ant colony algorithm, ACA)已经广泛应用于组合优化问题，通过模拟自然界蚂蚁行为而形成的算法，是模拟蚂蚁搜索食物的最短路径的过程。零件的剩余加工时间指标主要取决于零件的排样顺序，并且零件的排样顺序也会对排样效果产生一定的影响。本算法通过改进信息素更新策略和增加混合变异策略来改进蚁群算法以便于更好地解决本文问题。

2.1 节点选择概率

在本算法中，每个节点代表零件的编号，每个解决方案的生成过程是基于连续向蚂蚁的移动路线插入未在移动路线中节点的过程，最开始蚂蚁的路径是空的，首先蚂蚁初始化在一个节点，然后按照式(7)选择下一个节点，蚂蚁从节点i移动到节点j的概率为

$$P_{ij}=\frac{\left[\eta_{ij}\right]^{\alpha}\left[\tau_{ij}\right]^{\beta}}{\sum_{l\in V_{\mathrm{unvisited}}}\left[\eta_{il}\right]^{\alpha}\left[\tau_{il}\right]^{\beta}},j\in V_{\mathrm{unvisited}} \tag{7}$$

式中：α、β是决定启发因素和信息素对算法性能影响的重要指标；τ_{ij}为节点i接下来选择节点j的信息素浓度，初始值默认为1；η_{ij}为节点i选择节点j的启发信息值；$V_{\mathrm{unvisited}}$是所有未在路径上的节点集合。

2.2 启发式信息值

当选定初始零件后，根据排样算法会自动关联选择一些零件，根据排样算法特性，不同的排样顺序不但会对毛坯的均方利用率产生较大影响还会对最大剩余时间指标产生显著影响。因此，针对于2个指标，零件的启发信息应该包含面积因素和时间因素两部分。为了提高利用率，应该尽可能优先选择更大面积的零件，当面积相同时要优先选择预期加工时间相近的零件。根据上述特性，对启发信息定义为

$$\eta_{iq}=\begin{cases}0, & i=q\\ \dfrac{S_{\max}+(S_q-S_i)}{S_{\max}}+\dfrac{1}{\sqrt{2\pi}}\exp\left(-\dfrac{(t_q-t_i)^2}{2}\right), & i\neq q\end{cases} \tag{8}$$

式中：$S_{\max}$为零件中的最大面积；S_i、S_q分别为零件i、q的面积；t_i、t_q分别为零件i、q的预期加工时间。

2.3 信息素与自适应信息素更新策略

所有蚂蚁遍历完所有节点后，要进行信息素更新，信息素的浓度大小是影响算法性能的重要指标。每次迭代后都要对信息素按照式(9)进行更新。

$$\tau_{ij}=(1-\rho)\tau_{ij}+\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{9}$$

式中：$1-\rho$为信息素蒸发系数；$\Delta\tau_{ij}^{k}$为第k条蚂蚁路径上节点i到节点j的累计信息素值。

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{F_k}{F_{\mathrm{best}}}, & (i,j)\in G_{\mathrm{best}}\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

式中：G_{best}为最优路径包含的节点集；F_k为第k个解的适应度值；F_{best}为当前全局最优适应度值。

信息素的蒸发系数$1-\rho$是决定算法效率的关键因素[13]，在标准算法中ρ是在区间(0,1)之间的随机数。合理的设置该参数有利于优化算法计算时间过长的缺陷，该值越大表示信息素蒸发越快，算法越容易收敛于局部最优；该值越小，信息素

DOI: 10.16182/j.issn1004731x.joss.22-1258

蒸发越慢，搜索范围更大，但是算法收敛时间越长。因此，本算法提出一种自适应信息素更新策略，计算过程中算法会根据种群平均适应度值动态地改变蒸发系数，随着种群个体整体向好发展，算法会提高信息素蒸发速度，从而加快算法收敛速度，反之则降低蒸发速度以便于扩大搜索范围，不易陷入局部最优。其表达式如式(11)所示：

$$\rho=\rho_{\min}+\frac{F_{\max}-F_{\text{avg}}}{F_{\max}}(\rho_{\max}-\rho_{\min}) \tag{11}$$

式中：$\rho_{\min}$、$\rho_{\max}$分别为最小和最大参数；$F_{\max}$为理论适应度上限；F_{avg}为当前种群的平均适应度值。

适应度上限为理论上最大适应度值，由式(6)可知，归一化后的适应度值最大值$F_{\max}$=1。由文献[14]定义可知，一般将种群规模分为3类：小规模(n≤150)、中规模(150<n≤400)和大规模(n>400)，在中小规模的问题中，一般设置$\rho_{\min}$=0.001、$\rho_{\max}$=0.1，在大规模问题中一般取$\rho_{\min}$=0.02、$\rho_{\max}$=0.5。

2.4 混合变异策略

为提高种群搜索能力，算法要在迭代初期扩大搜索范围，而在算法后期倾向于局部的精细搜索。然而，对搜索路径进行大规模的更改会影响到蚁群算法收集到的信息质量，因此本算法引入遗传算法的变异机制和经典的2-opt变异机制，并且随着迭代次数的增加动态地改变变异规则。遗传算法的变异机制针对一条蚁群路径进行多点交换变异，有益于扩大搜索范围，跳出局部最优。2-opt变异机制主要是针对路径中的两点进行位置交换，有利于针对解得邻域进行精细搜索，更有益于搜索最优值。因此利用轮盘赌算法在迭代过程中对2种变异机制进行选择，算法迭代初期，应该扩大蚁群搜索范围，因此应该更倾向于选择遗传变异机制，此时遗传变异机制被选择的概率应该更大，随着算法迭代次数增加减少遗传变异机制的选择概率，增加2-opt变异机制的选择概率，有利于算法获取全局最优值，每次迭代选用遗传变异机制的概率为

$$P=1-\frac{i}{I_{\max}} \tag{12}$$

式中：$I_{\max}$为最大迭代次数。

每次变异之后得到的结果与变异之前进行对比，选择适应度值更高的路径。考虑到算法性能，如果每次对所有蚂蚁的路径进行变异，会对算法计算速度产生较大影响，因此本算法根据迭代次数动态调整变异个体数。每代变异个数z由式(13)确定。

$$z=\left\lceil rand()\times\frac{I_{\max}-i+1}{2I_{\max}}\times Pop\right\rceil \tag{13}$$

式中：⌈ ⌉表示向上取整；$rand()$是取值在(0,1)之间的随机数；Pop为种群大小且$Pop\in\mathbb{N}$。

2.5 加工顺序的确定及结果有效性检查

算法产生的排样结果毛坯集后，要按照毛坯中零件工序加工时间对毛坯的加工顺序进行排序。在相同利用率解决方案的前提下，加入加工顺序排序方法，进行有效性检查后决定该解决方案是否保留，经过算法计算与筛选产生最终结果序列。然后根据获得的结果排序选取最大结果的那条路径作为本次迭代的最优解。加工顺序排序方法步骤如下：

步骤1：首先复制当前解的路径并按照加工时间从小到大重新排序。

步骤2：选取蚁群路径中第一个零件所在的毛坯作为第一个加工的毛坯。

步骤3：根据式(3)计算该毛坯中所有零件的剩余加工时间，如果存在剩余加工时间为负数，则直接终止排序，返回步骤1。

步骤4：更新为排序零件集，从路径中剔除已排序毛坯中的所有零件编码。

步骤5：判断路径长度是否为0，如果不为0则转步骤2。如果路径中对应的所有零件都计算完毕，根据式(4)计算剩余时间总和并返回。

2.6 算法流程

混合算法总体流程图如图 1 所示。其中各步骤描述如下：

步骤 1：初始化种群，随机生成大小为N的种群，初始化各个参数和信息素浓度。

步骤 2：调用分割匹配算法生成解。

步骤 3：对步骤 2 的解调用加工排序算法获得毛坯加工次序。

步骤 4：计算并保存解得适应度值，判断种群个体未全部计算则转步骤 2 计算下一个个体，否则转步骤 5。

步骤 5：对得到的结果集按照适应度值排序，更新最优解。

步骤 6：判断如果没达到最大迭代次数则更新当前迭代次数并且更新信息素以及算法各个参数，转步骤 2。否则跳出循环输出最优解。

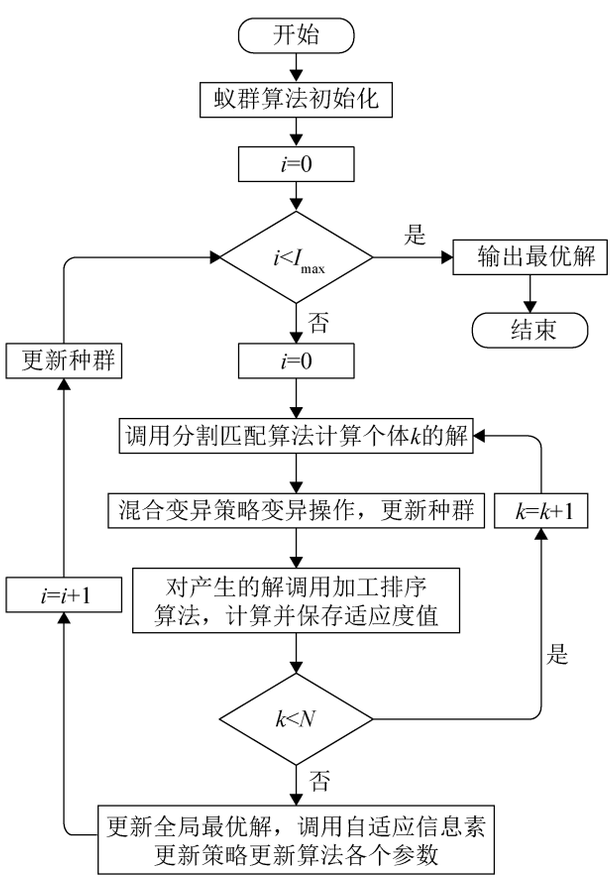


图 1 混合算法流程

Fig. 1 Flow chart of hybrid algorithm

3 分割匹配算法

确定了零件的排放顺序后，就要继续确定零件需要排放的毛坯以及在毛坯上的具体排放位置，在满足一刀切工艺约束的前提下，尽可能提高每个毛坯的利用率。在满足以上条件下，本文提出了一种基于矩形相似匹配和最大剩余面积分割的分割匹配算法。整个算法核心主要包括匹配搜索，分割定位，利用率优化三部分。

3.1 匹配搜索

首先读取现有的零件集和毛坯集，根据面积从大到小顺序进行排序，如果面积相等则按照宽度、高度依次从大到小排序，若两矩形相同则按照读取先后顺序排序，排序后的两组结果作为零件搜索集和毛坯搜索集。相似匹配搜索主要根据零件或者毛坯的面积和长宽进行搜索，若都相同则按序号返回，通过输入的目标矩形参数快速选择零件或者毛坯，根据式(14)返回索引值，如果没有零件符合条件，则返回−1。图 2 描述了零件匹配搜索算法的伪代码。

$$Index_j = \begin{cases} INDEX(\min(R_j - R_i)), R_j - R_i > 0 \\ INDEX(\min(L_i) \oplus \min(W_i) \oplus i), R_j - R_i = 0 \\ -1, \text{others} \end{cases} \tag{14}$$

式中：$INDEX()$为返回选定零件的索引函数；R_j为当前毛坯j的面积；R_i为零件i的面积；L_i，W_i分别为零件i的长和宽；符号⊕为前后递进判断，如果不符合前面条件则判断后面条件。

3.2 分割定位

通过匹配搜索算法找到的最合适的排样零件i排放到毛坯的最左下位置，为满足一刀切约束，放置完成后将剩余部分分割为 2 部分，一般来说会产生 2 种分割模式，如图 3 所示。

尽可能获取到较大的剩余面积有利于后续零件的填补排放，因此，要对 2 种分割模式进行面积计算与对比，对R_{ar}，R_{at}，R_{br}，R_{bt} 4 个矩形进行面积排序，若面积相等则根据宽度、高度依次进

DOI: 10.16182/j.issn1004731x.joss.22-1258

行排序，若两矩形完全相同则横向模式优先排序，最终选择排序第一个矩形所在的分割模式进行分割。选择模式表示为

$$E=\begin{cases}ER_{\mathrm{a}}, R=R_{\mathrm{a}*}\oplus L_{\mathrm{a}*}\oplus W_{\mathrm{a}*}\\ ER_{\mathrm{b}}, R=R_{\mathrm{b}*}\oplus L_{\mathrm{b}*}\oplus W_{\mathrm{b}*}\end{cases}$$

s.t.

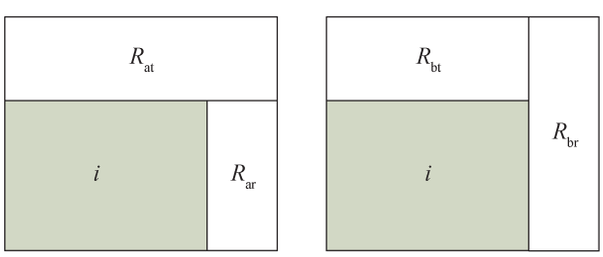
$$R=\begin{cases}\max(R_{\mathrm{ar}}, R_{\mathrm{at}}, R_{\mathrm{br}}, R_{\mathrm{bt}}), & R_{\mathrm{a}*}\neq R_{\mathrm{b}*}\\ \max(L_{\mathrm{ar}}, L_{\mathrm{at}}, L_{\mathrm{br}}, L_{\mathrm{bt}}), & R_{\mathrm{a}*}=R_{\mathrm{b}*}\\ \max(W_{\mathrm{ar}}, W_{\mathrm{at}}, W_{\mathrm{br}}, W_{\mathrm{bt}}), & R_{\mathrm{a}*}=R_{\mathrm{b}*}\text{and}L_{\mathrm{a}*}=L_{\mathrm{b}*}\\ R_{\mathrm{at}}, & \text{others}\end{cases} \tag{15}$$

式中：PR_{a}和PR_{b}分别为横向模式和纵向模式；*为通配符，这里代表a或者b；L和W分别为毛坯的长和宽。

```
输入：SORTEDITEMS, RECTANGLE/*SORTEDITEMS为有序零件组合，RECTANGLE为待排毛坯*/
输出：index/*返回零件索引*/
1   SAM(SORTEDITEMS, RECTANGLE)
2       {area=the area of RECTANGLE
3       if area<the minimum area of item in SORTEDITEMS
4             {return -1}
5       search the item from SORTEDITEMS
6       while(MAX(height of item,width of item)>
7           MAX(height of  RECTANGLE,width of RECTANGLE)
8           or MIN(height of item,width of item)>
9           MIN(height of  RECTANGLE,width of RECTANGLE))
10            {i=the index of item
11            i=i+1
12            if (i==the size of SORTEDITEMS)
13               {return -1}
14            item=the i-th of item in SORTEDITEMS
15            }
16          return the index if item
17      }
```

图2　匹配搜索算法伪代码

Fig. 2　Pseudocode of matching search algorithm



(a) 横向模式　　(b) 纵向模式

图3　两种分割模式

Fig. 3　Two patterns of cutting

当出现零件与毛坯完美匹配的情况时会产生只有一个或者没有剩余毛坯的情况，大部分情况下每次排放一个零件就会产生最多2个子毛坯，每个子毛坯又可以当做新规格的毛坯继续按照同样的方式排放零件，按照式(16)循环递归直到不产生子毛坯即R_j=0或者子毛坯不能排放任何一个零件即R_j<min(R_i)，此时返回该毛坯和其包含的所有零件作为一组有效解，其伪代码如图4所示。

$$R_j=R_{j+1}+R_{j+2}+R_i \tag{16}$$

```
输入：RESULT, SORTEDITEMS, ITEMS/*RESULT空结果集，SORTEDITEMS有序零件集，ITEMS当前排样零件集*/
输出：RESULT/*排样后零件集*/
1    RC(RESULT, SORTEDITEMS, ITEMS)
2          {compute Rar，Rat，Rbr，Rbt
3           compute A,B←Rar，Rat，Rbr，Rbt
4           if the area A > 0
5              {BINA←A
6               itemA=SAM(SORTEDITEMS,BINA)
7               put item A into RESULT
8               delete item A from  ITEMS and SORTEDITEMS
9               RC(RESULT,RSORTEDITEMS, ITEMS)
10             }
11          if the area B > 0
12             {BINB←B
13              item B=SAM(SORTEDITEMS, BINB)
14              put item B into RESULT
15              delete item B from ITEMS and SORTEDITEMS
16              RC(RESULT,RSORTEDITEMS, ITEMS)
17             }
18          return RESULT
19         }
```

图4　矩形分割算法伪代码

Fig. 4　Pseudocode of rectangle cutting algorithm

3.3 利用率优化

分割算法产生一个可行解之后，需要对该结果进行利用率计算，此时需要给算法设定一个均方利用率指标值M，如果均方利用率小于M则要重新选择毛坯后对这种排放方案进行重新排样，直到遇到下述3种情况之一即可返回最佳结果：①该均方利用率大于等于M；②零件列表中没有剩余零件；③循环搜索到最后一个毛坯均方利用率仍未达到M，此时返回之前搜索的排样可行解集合中利用率最高的那个结果。这种优化策略依赖于毛坯的数量，所以在初始化毛坯列表时，毛坯数量必须是有限的且能容纳下所有零件。但是

随着毛坯数量的增加，总会出现一些规格相同的毛坯，这会造成一些重复运算，因此当开始运行优化策略时，判断毛坯是否与之前的毛坯规格相同，如果相同则跳过此毛坯，这样能够有效提升算法的运行速率。利用率优化算法伪代码如图5所示。

```
输入：SORTEDRECTS,SORTEDITEMS，ITEMS，RESULT，
Rete，R/*SORTEDRECTS为有序毛坯集合，SORTEDITEMS为
有序零件序列，ITEMS为当前排样序列，RESULT为当前结果
集，Rete为当前毛坯利用率，R为当前排样毛坯*/
输出：best/*best为提升后的零件集、毛坯、利用率参数集*/
1          SORTEDITEMS _COPY←SORTEDITEMS
2          ITEMS _COPY←ITEMS
3          best←Rete，R，RESULT
4          bIndex = the index of R
5          while(rate<M and SORTEDITEMS not null)
6           {bIndex = bIndex-1
7            if bIndex==-1
8              {delete items in RESULT from SORTEDITEMS
9               delete R from SORTEDRECTS
10              return best
11             }
12           ITEMS=ITEMS_COPY
13           SORTEDITEMS= SORTEDITEMS_COPY
14           R=SORTEDRECTS[bIndex]
15          delete other items in RESULT except to the first item
16           RC (RESULT,SORTEDITEMS,ITEMS)
17           compute rate ← RESULT
18           if rate>rate in Best
19             {best←Rete，R，RESULT
20              break
21             }
22          }
23         delete items in RESULT from SORTEDITEMS
24         delete R from SORTEDRECTS
25          return beat
```

图5 利用率优化算法伪代码

Fig. 5 Pseudocode of utilization optimization algorithm

3.4 分割匹配算法流程

分割匹配(cutting and matching, CM)算法主要流程图如图6所示。各步骤描述如下：

步骤1：输入矩形零件序列和矩形毛坯序列。

步骤2：选择矩形零件序列的第一个矩形。

步骤3：调用匹配搜索算法SAM和矩形分割算法RC计算求解，同时更新矩形零件序列(除零件)。

步骤4：执行利用率优化算法。

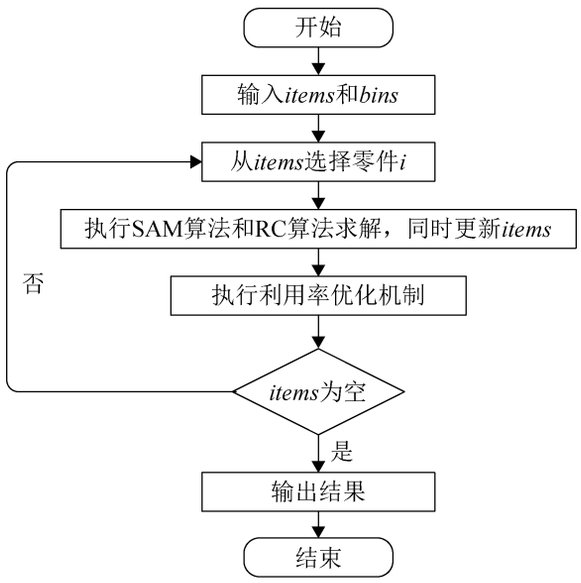


图6 CM算法流程

Fig. 6 Flow chart of CM algorithm

4 实验计算与对比

混合算法(IACA-CM)及对比算法采用Python3.7语言在IntelliJ IDEA平台上编写，均在8G RAM， Intel(R) Core(TM) i5-6200U CPU (2.30GHz)配置的设备上运行。算法运行的标准数据集采用文献[1]中的国际通用数据集M系列、Nice&Path系列和PisingerSigurd系列数据集和南通某玻璃幕墙企业板材下料实际数据集(KBH项目一楼L1到十楼L10板材下料)。实验分为两部分，第一部分设计3组实验验证算法有效性，第1组实验用来验证改进蚁群算法有效性，从Nice&Path系列数据集中按照规模选择Path50i(小规模)、Path300i(中规模)、Path500i(大规模)3组测试的数据集进行对比分析；第2组实验验证算法收敛性；第3组实验是验证分割匹配算法有效性，对3种规模数据集仿真实验。第二部分对比分析本文算法与其他算法(文献[1]中的BFS(best-fit with stacking)算法、文献[6]中HHA(hybrid heuristic algorithm)算法和文献[9]中的GDRR(goal-driven ruin and

DOI: 10.16182/j.issn1004731x.joss.22-1258

recreate)算法)，通过特定数据集进行对比分析。

通常零件的最小加工时间的设定为单个毛坯的加工时间[12]，由于本文问题中每个毛坯的加工时间并不确定，为了使零件的预期加工时间尽可能符合要求，本文利用第 2 节的搜索匹配算法针对每个毛坯进行排样处理，然后根据式(3)计算出毛坯加工时间并取所有毛坯的平均加工时间作为计算最小预期加工时间 t_{min}，根据式(3)取得最大值 t_{max}。零件的加工时间从服从[t_{min}, t_{max}]的均匀分布的随机数中选取。实验的数据集都要求在 600 s 限制下运行得出结果，且每个数据集运行 50 次取得的均方利用率平均值作为对比指标，均方利用率指标值设定为 M=0.9，主优化目标权重设定为 γ=0.7，算法其他参数设置为 α=1、β=2，种群大小设置为 200，迭代次数设置为 60，这种设置方式在很多研究成果中已经证实可以获得良好效果[15-17]。

4.1 改进算法有效性验证

考虑种群规模不同，种群大小要根据零件规模大小相应变化，小规模种群大小设置为 50 中规模设置为 200 大规模设置为 500，标准蚁群算法信息素挥发参数设置为 ρ=0.05。分别测试 3 种规模下改进蚁群算法和标准蚁群算法性能对比，以及利用仿真结果证明分割匹配算法有效性。

4.1.1 改进蚁群算法有效性验证

改进蚁群算法只加入信息素更新策略并与标准蚁群算法进行对比，其中得到小中大规模结果如图 7 所示。

从图 7 可以看出，加入的信息素更新策略可以很好地对算法综合性能进行调控，相比于标准算法更快获得相对最优解，但是如图 7(c)所示，在大规模问题上，信息素更新策略对排样结果的影响明显变小，尤其在算法后期，也很对算法的影响越来越小,改进的算法与原生算法并没有明显的优势，因此需要在算法运行后期加入具有显著改进效益的变异策略。

改进蚁群算法只加入混合变异策略并与标准蚁群算法进行对比，其中得到小中大规模结果如图 8 所示。

根据图 8 可知，加入了混合变异机制的算法

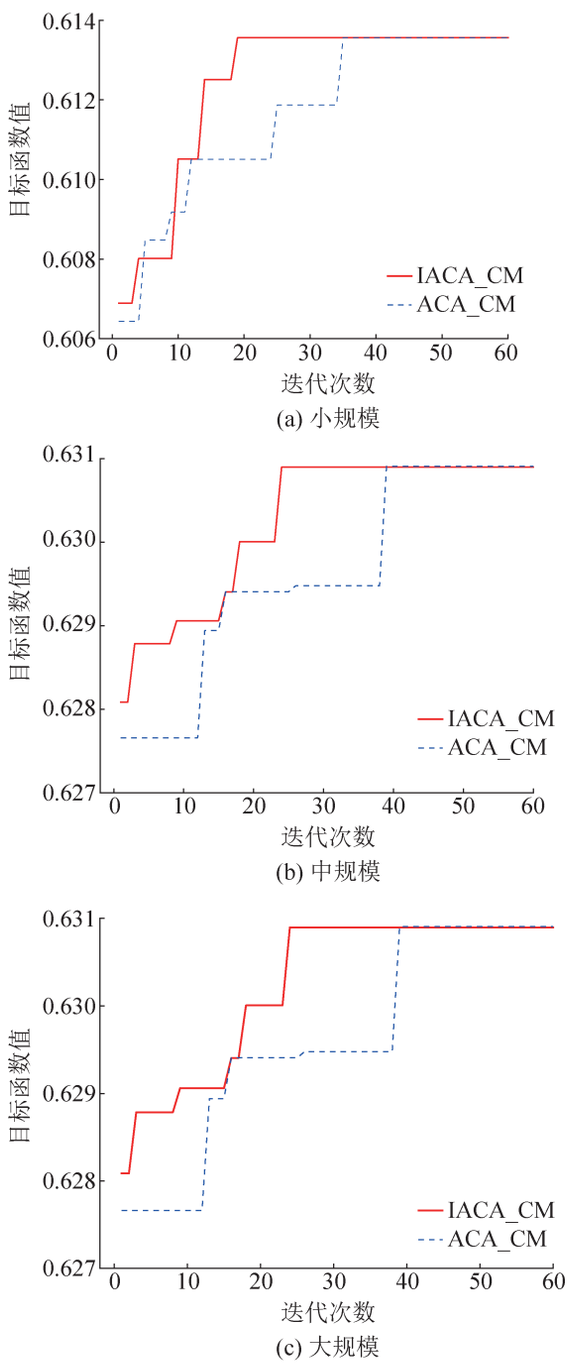


图 7　信息素策略改进 3 种规模对比

Fig. 7　Comparison of three scales of pheromone strategy improvement

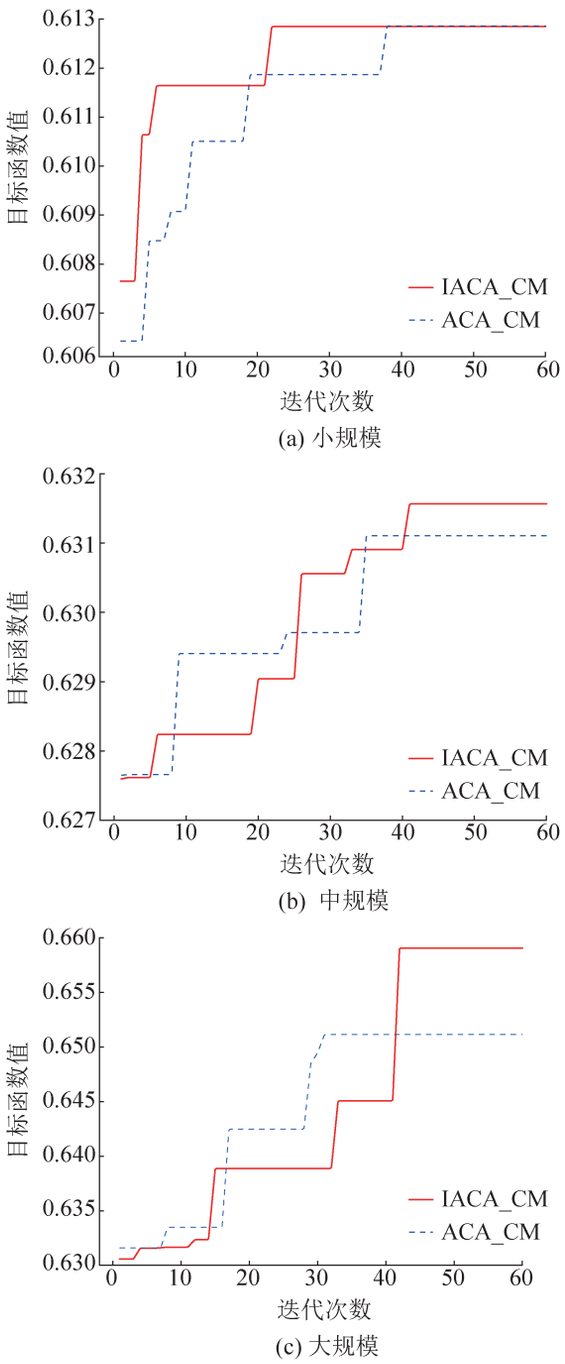


图8 变异策略改进3种规模对比

Fig. 8 Comparison of three scales of mutation strategy improvement

虽然在整体性能上并不具有太大优势，但是根据图8(b)和(c)所示，算法的最终结果明显好于标准算法，这说明了混合变异机制能够有效帮助算法跳出局部最优。

4.1.2 混合算法收敛性分析

为了更好的验证改进算法的性能，本节对算法的收敛性进行分析。图9为小中大3种数据集对应不同算法的收敛曲线。

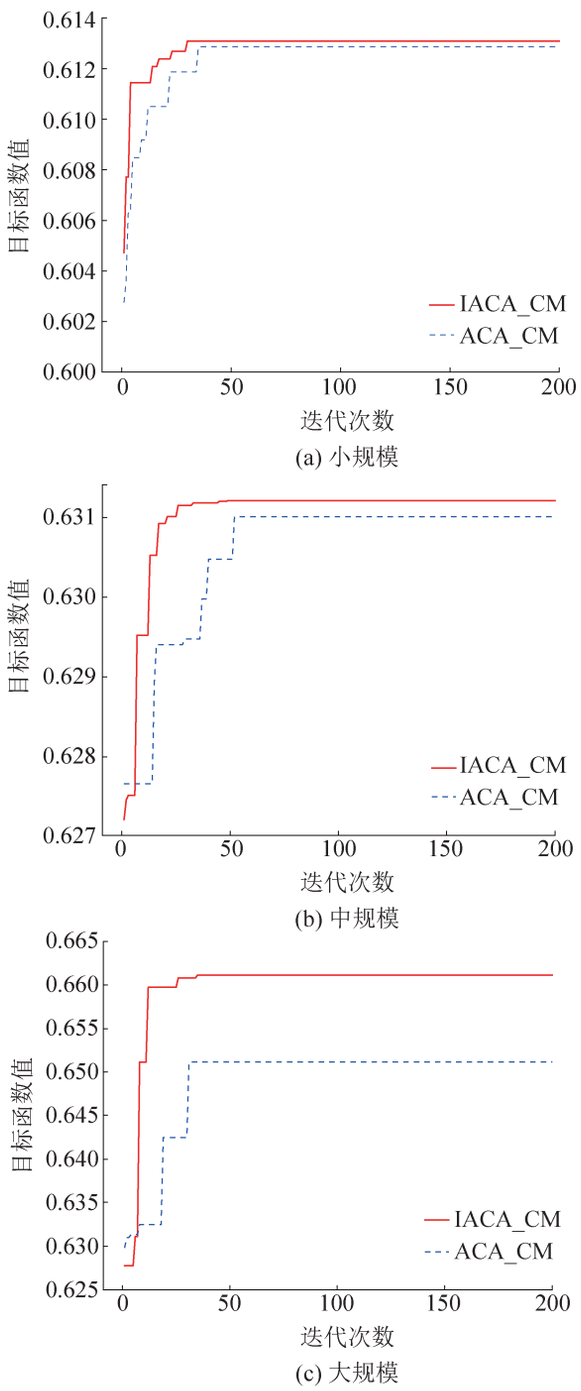


图9 算法收敛性3种规模对比

Fig. 9 Comparison of three scales of algorithm convergence

https://dc-china-simulation.researchcommons.org/journal/vol36/iss3/18
DOI: 10.16182/j.issn1004731x.joss.22-1258

由图 9 三组收敛曲线对比可以看出，IACA_CM 的收敛曲线均在 ACA_CM 的上方，说明 IACA_CM 在迭代初期可以迅速找到近似优解，这是由于信息素更新策略对算法影响最大，使其快速收敛；算法运行中期，渐渐受变异策略的影响，使其逐渐探索最优路径；算法后期，如图 9(c)所示，当检测到算法陷入停滞时，变异策略能够有效帮助算法跳出局部最优解，最终在全局最优解出收敛，而未加变异策略的算法则明显陷入局部最优解。

4.1.3 分割匹配算法有效性验证

由算法整体仿真实验得出大、中、小 3 种规模仿真结果如表 1 所示。表中给出 3 种规模下的毛坯使用规格及使用个数和利用率超过 90% 和 95% 的毛坯个数。由于篇幅有限，只展示小规模零件排样的效果仿真图，如图 10 所示。

表 1　仿真结果
Table 1　Results of simulation

规模	毛坯使用种类数	毛坯使用个数	利用率≥90%	利用率≥95%
小	6	7	6	6
中	6	27	23	20
大	6	68	61	59

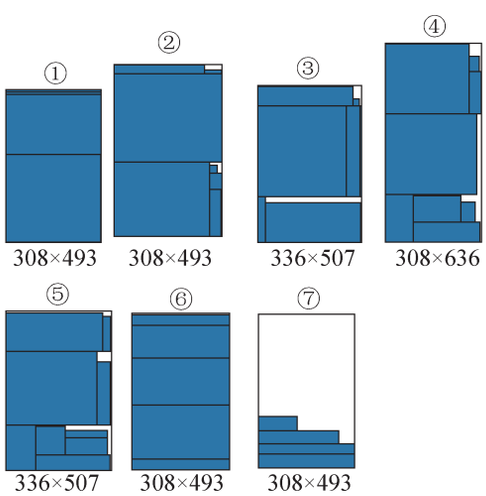


图 10　排样仿真图
Fig. 10　Layout simulation

根据表 1 可得出，3 种规模零件排样中毛坯利用率占比 90% 以上的毛坯占总体使用毛坯的 85% 以上，其中利用率超 95% 以上也占绝大部分，而且，除去最后未排满的一个毛坯，其他毛坯整体效果会更为乐观。

由图 10 可以看出，除了因为零件数量不足而未排满的最后一个毛坯外，其他在的 6 个毛坯中，所有毛坯的利用率达到了 95% 以上，其中第 1 个和第 6 个毛坯利用率更是接近于 100%，由此说明本文排样算法在排样质量上具有一定参考价值。

4.2 算法性能对比

因为对比参数为均方利用率，所以用式(2)代替本项测试的适应度函数，并且式(8)中启发信息值也暂时屏蔽后半部分时间影响式。表 2~4 分别给出了 3 种数据集几种算法的均方利用率值对比，表中数据都是百分制数据，其中每组数据集中效果最好的算法结果都被加粗标出。

表 2　M 系列数据集的实验结果
Table 2　Experimental results of M series data set　%

数据集	IACA-CM	GDRR	BFS	HHA
M1	95.3	**96.8**	91.2	**96.8**
M2	**95.5**	94.4	81.0	91.4
M3	**96.7**	96.1	90.1	94.8
平均值	**95.83**	95.76	87.43	94.33

表 3　PisingerSigurd 系列数据集的实验结果
Table 3　Experimental results of PisingerSigurd series data set　%

数据集	IACA-CM	GDRR	BFS	HHA
I	**92.8**	88.5	78.5	83.7
II	**93.8**	93.1	73.2	92.7
III	**86.8**	84.5	70.2	75.0
IV	**88.2**	87.4	70.0	84.0
V	**83.0**	81.7	66.2	71.6
VI	85.2	**86.3**	64.2	82.5
VII	**84.8**	81.2	65.5	73.4
VIII	**83.7**	80.3	67.0	73.8
IX	**66.1**	60.8	54.2	55.7
X	**87.4**	86.5	75.8	81.5
平均值	**85.18**	83.03	68.18	76.89

表 4 Nice&Path 系列数据集的实验结果
Table 4 Experimental results of Nice&Path series data set
%

数据集	IACA-CM	GDRR	BFS	HHA
Nice25i	89.4	**99.6**	55.6	90.0
Nice50i	**90.7**	87.5	60.5	79.6
Nice100i	**88.9**	84.2	64.3	77.9
Nice200i	85.0	**85.7**	72.4	83.2
Nice300i	**89.3**	87.0	75.3	84.5
Nice400i	**89.0**	88.9	73.4	86.4
Nice500i	**89.7**	87.5	76.9	86.6
Path25i	92.9	**100**	62.2	94.5
Path50i	92.3	**96.2**	67.1	90.0
Path100i	**93.3**	91.2	69.7	87.4
Path200i	**92.1**	89.3	76.5	87.1
Path300i	**94.7**	91.3	76.3	90.5
Path400i	**93.2**	90.9	78.4	89.9
Path500i	**93.5**	93.3	75.6	88.4
平均值	**91.00**	90.90	70.30	86.86

由表 2 可以看出，在 M 系列的 3 组数据集中，本文算法整体获得较好的排样效果，均方利用率均达到 95% 以上，算法效果比 BFS 和 HHA 有一定的优势，但对比 GDRR 算法获得的结果并没有显著优势。根据表 3 的 PisingerSigurd 系列的 10 组数据集的对比其他算法效果，总体上本文算法较其他算法更有优势，并且总体的平均值较 GDRR 高出 2%、比 BFS 算法高出 17%、比 HHA 算法高出 8.29%。在表 4 的 Nice&Path 系列数据集的对比结果中，虽然 GDRR 算法有个别数据集排样效果优于本文算法，但是从整体效果上来看，本文算法在 14 个数据集中有 10 个均方利用率最高，这体现出了本文算法较其他算法具有更好的稳定性。整体来说，本文算法在变尺寸箱排样效果上具有一定的竞争力。表 5 所示为基于企业实际板材生产下料数据进行测试对比的结果，本文算法在解决该实际问题上具有很明显的优势，由此可见，本文算法具有一定的生产实用价值。

表 5 企业板材下料数据集的实验结果
Table 5 Experimental results of enterprise wood blanking
%

数据集	IACA-CM	GDRR	BFS	HHA
KBH-L1	**95.3**	90.6	76.3	86.0
KBH-L2	**90.7**	89.5	69.5	77.2
KBH-L3	**93.4**	90.4	76.8	87.5
KBH-L4	**91.8**	88.2	70.0	83.1
KBH-L5	**94.1**	91.0	79.1	86.3
KBH-L6	**93.0**	86.9	71.7	79.7
KBH-L7	**93.2**	90.5	78.9	89.6
KBH-L8	**95.7**	92.0	77.5	89.7
KBH-L9	**94.5**	88.2	68.6	87.3
KBH-L10	**96.5**	91.8	72.5	90.2

5 结论

针对加工时间限制和一刀切约束下多规格毛坯矩形件排样问题提出混合求解算法。提出自适应信息素更新策略改进蚁群算法确定排样顺序，又引入了混合变异机制改善算法。提出分割排样算法决定排样位置，提高毛坯利用率。实验结果表明，分割排样算法能够获得相对较好的结果，IACA-CM 算法能够有效解决当前约束下的排样问题，改进的蚁群算法比标准蚁群算法在求解性能和质量上都具有更好的效果；IACA-CM 算法在排样质量上较其他文献算法也具有一定优势。但是当零件和毛坯规模更大时，算法的时间复杂度并不是非常理想。今后的研究重点将专注于大规模零件排样时算法的性能优化，以较少的计算时间完成大规模零件的排样。同时为异形件的拼接排样打下良好基础，基于本文算法思想并结合异形件的处理方法对异形件进行优化排样也是未来研究方向之一。

参考文献：

[1] Frank G Ortmann, Nthabiseng Ntene, Jan H van Vuuren. New and Improved Level Heuristics for the Rectangular Strip Packing and Variable-sized Bin Packing Problems

DOI: 10.16182/j.issn1004731x.joss.22-1258

[J]. European Journal of Operational Research, 2010, 203 (2): 306-315.

[2] 张怀宇, 杨根科, 白杰. 二维Strip Packing问题的嵌套启发式算法[J]. 系统仿真学报, 2012, 24(8): 1601-1605, 1623.
Zhang Huaiyu, Yang Genke, Bai Jie. Nested Heuristic Algorithm for Two-dimensional Strip Packing Problem [J]. Journal of System Simulation, 2012, 24(8): 1601-1605, 1623.

[3] 张子成. 基于矩形拼接的"一刀切"矩形排样优化设计[J]. 现代制造工程, 2018(4): 103-107, 157.
Zhang Zicheng. Optimal Design of Rectangular Layout Based on Rectangular Mosaic with Guillotine Constraints [J]. Modern Manufacturing Engineering, 2018(4): 103-107, 157.

[4] 刘诚, 孙远升, 花军, 等. 基于遗传-贪心混合搜索的人造板下料算法[J]. 林业工程学报, 2021, 6(4): 127-133.
Liu Cheng, Sun Yuansheng, Hua Jun, et al. Cutting of Wood-based Panel Based on Genetic Greedy Hybrid Search[J]. Journal of Forestry Engineering, 2021, 6(4): 127-133.

[5] Liu Ya, Chu Chengbin, Wang Kanliang. A Dynamic Programming-based Heuristic for the Variable Sized Two-dimensional Bin Packing Problem[J]. International Journal of Production Research, 2011, 49(13): 3815-3831.

[6] Hong Shaohui, Zhang Defu, Lau H C, et al. A Hybrid Heuristic Algorithm for the 2D Variable-sized Bin Packing Problem[J]. European Journal of Operational Research, 2014, 238(1): 95-103.

[7] Wei Lijun, Wee Chong Oon, Zhu Wenbin, et al. A Goal-driven Approach to the 2D Bin Packing and Variable-sized Bin Packing Problems[J]. European Journal of Operational Research, 2013, 224(1): 110-121.

[8] Mateus Martin, Reinaldo Morabito, Pedro Munari. A Bottom-up Packing Approach for Modeling the Constrained Two-dimensional Guillotine Placement Problem[J]. Computers & Operations Research, 2020, 115: 104851.

[9] Jeroen Gardeyn, Tony Wauters. A Goal-driven Ruin and Recreate Heuristic for the 2D Variable-sized Bin Packing Problem with Guillotine Constraints[J]. European Journal of Operational Research, 2022, 301(2): 432-444.

[10] Bennell J A, Lai Soon Lee, Potts C N. A Genetic Algorithm for Two-dimensional Bin Packing with Due Dates[J]. International Journal of Production Economics, 2013, 145(2): 547-560.

[11] Claudio Arbib, Fabrizio Marinelli, Andrea Pizzuti. Number of bins and Maximum Lateness Minimization in Two-dimensional Bin Packing[J]. European Journal of Operational Research, 2021, 291(1): 101-113.

[12] Liu Qiang, Cheng Huibing, Tian Tian, et al. Algorithms for the Variable-sized Bin Packing Problem with Time Windows[J]. Computers & Industrial Engineering, 2021, 155: 107175.

[13] Petr Stodola, Karel Michenka, Jan Nohel, et al. Hybrid Algorithm Based on Ant Colony Optimization and Simulated Annealing Applied to the Dynamic Traveling Salesman Problem[J]. Entropy, 2020, 22(8): 884.

[14] Petr Stodola, Pavel Otřísal, Kamila Hasilová. Adaptive Ant Colony Optimization with Node Clustering Applied to the Travelling Salesman Problem[J]. Swarm and Evolutionary Computation, 2022, 70: 101056.

[15] 胡蓉, 陈文博, 钱斌, 等. 学习型蚁群算法求解绿色多车场车辆路径问题[J]. 系统仿真学报, 2021, 33(9): 2095-2108.
Hu Rong, Chen Wenbo, Qian Bin, et al. Learning Ant Colony Algorithm for Green Multi-depot Vehicle Routing Problem[J]. Journal of System Simulation, 2021, 33(9): 2095-2108.

[16] Yan Yuzhe, Sohn H S, Reyes G. A Modified Ant System to Achieve Better Balance Between Intensification and Diversification for the Traveling Salesman Problem[J]. Applied Soft Computing, 2017, 60: 256-267.

[17] Sahar Ebadinezhad. DEACO: Adopting Dynamic Evaporation Strategy to Enhance ACO Algorithm for the Traveling Salesman Problem[J]. Engineering Applications of Artificial Intelligence, 2020, 92: 103649.