arXiv:2603.00484v1 [cs.CG] 28 Feb 2026

MergeDJD：一种面向二维不规则装箱问题的带零件合并的快速构造算法

周毅[a,c]，傅昊成[a]，刘一平[a,b]，毛健[a]，傅章华[b]，王宇毅[c,*]

[a] *University of Electronic Science and Technology of China, Chengdu, 611731, China*
[b] *The Chinese University of Hong Kong, Shenzhen, 518129, China*
[c] *CRRC Zhuzhou Institute 与 Tengen Intelligence Institute, Zhuzhou, 412000, China*

摘要

二维不规则装箱问题（2DIBPP）旨在将给定的一组不规则多边形（称为零件）无重叠地装入固定尺寸的矩形箱中，同时最大化箱利用率。尽管已有大量针对二维不规则装箱问题的元启发式算法，但由于构造启发式具有确定性且计算开销低，许多工业应用更倾向于采用这类方法。在此类方法中，L´opez-Camacho 等人提出的 DJD 算法是二维不规则装箱问题中最具竞争力的构造启发式之一。然而，对于切割实例，DJD 的效果较差，因为这类实例中的许多零件可以无缝地组合成更大的多边形。为解决这一问题，我们提出了 MergeDJD，这是一种新颖的构造算法，整合并扩展了 DJD 框架。MergeDJD 首先对实例进行预处理，通过迭代识别可组合成更大且更规则零件的零件组。随后，它采用改进版 DJD 算法，将放置策略加以改进，以便更好地处理非凸形状和组合形状，并将所有生成的零件装入箱中。针对 1,089 个知名基准实例的计算实验表明，MergeDJD 在其中 1,083 个实例上的表现始终优于 DJD，同时运行时间较短。值得注意的是，MergeDJD 在 515 个实例上取得了新的已知最优值。消融研究进一步证实了所提各组件的有效性。为便于复现——

*通讯作者*Email addresses:* zhou.yi@uestc.edu.cn（周毅），hfu622@std.uestc.edu.cn（傅昊成），liuyiping@cuhk.edu.cn（刘一平），maomaoxbx@163.com（毛健），fuzhanghua@cuhk.edu.cn（傅章华），yuyiwang920@gmail.com（王宇毅）

性并支持后续研究，我们已将完整实现开源，并提供了用于可视化装箱结果的接口。

关键词： 不规则装箱问题，构造启发式，零件合并策略，组合优化。

---

1. 引言

二维不规则装箱问题（2DIBPP）是一类众所周知且具有挑战性的工业组合优化问题。一般而言，该问题要求将给定集合中的不规则多边形零件放入多个矩形箱中，且彼此不重叠，同时最大化箱的利用率。二维不规则装箱问题在工业领域具有重要意义，尤其是在排样过程中，需要从矩形板材上切割二维零件 [5]。典型应用包括汽车与船舶制造 [26, 19], 服装业 [17], 以及玻璃生产 [16]。在这些场景中，高质量的装箱方案可以直接减少材料浪费，并带来可观的经济效益。

然而，从理论和实践角度来看，求解二维不规则装箱问题都极具挑战性。从理论上说，即使将所有零件限制为矩形（即二维装箱） [11]，或将问题设定为一维情形（即传统装箱） [9]，该问题仍然是NP难的。在实践中，精确求解方法仅适用于规模很小的实例，这限制了它们在现实应用中的价值。因此，研究者开发了各种启发式方法，以便在合理的计算时间内获得高质量的装箱方案。

1.1. 文献综述

本节总结现有的启发式算法。总体而言，这些方法可大致分为两类：构造算法 和改进算法。

构造算法。 构造算法通常从一个空解开始，逐步扩展部分解，直至得到完整的可行解。通过做出不可撤销的决策，构造启发式可以以较低的计算开销生成可行解。对于二维不规则装箱问题，构造算法需要做出两项启发式决策：零件到箱分配 和箱内定位。前者决定将零件分配到哪个箱，后者则确定零件在所选箱内的具体放置位置。

对于零件分配到箱阶段，递减首次适配 （FFD）按照面积递减顺序对零件排序，并将每个零件分配到第一个能够将其可行放置的箱中；如果不存在这样的箱，则开启一个新箱。递减最佳适配 （BFD）则选择放置后剩余空闲空间最小的箱。在每个箱内，通常按照面积递减顺序处理零件。FFD 和 BFD 最初都是为矩形零件装箱而提出的 [21]。它们被改编用于不规则零件 [15]。基于这些原则，DJD 启发式算法 [15]进一步优化放置顺序和箱体分配，以提高箱体利用率。

在箱内定位阶段，左下启发式算法会将零件放置在最低的可行位置；若有多个位置并列，则选择最靠左的位置 [4]。最大利用率启发式算法会在最早箱中选择使面积利用率最大的布局，而最小长度启发式算法则旨在最小化部分解外接矩形的长度 [2]。CAD启发式算法通过最大化已放置块与已有零件或箱子边界之间的邻接性，进一步提升布局质量 [23]。

由于高效且简单，构造启发式广泛用作独立方法，或作为更复杂改进算法的初始化组件。

改进算法。与通过一系列不可撤销的决策生成解的构造启发式不同，改进算法会在更充足的计算预算下，反复应用启发式规则进行迭代搜索。从这一角度看，二维不规则装箱问题的两类代表性改进算法是局部搜索和超启发式。

局部搜索从一个可行解（有时由构造启发式生成）出发，反复执行局部修改，例如扰动箱内的放置位置，或在不同箱子之间交换零件，以获得更优解。具体而言， [16]使用整数规划和混合整数规划模型构建了箱体分配与装箱模型，并结合局部改进机制。 [1]提出了*Jostle* 启发式算法，通过局部扰动零件位置和方向来改进初始放置。 [10, 26] 通过 FFD 和左下启发式算法生成初始解，然后在不同箱子之间应用重新定位和交换操作。近期研究还探索了替代表示和邻域设计，包括基于栅格的建模、重叠最小化和迭代空间收缩 [14, 25, 20]。此外， [24, 3]通过分解问题并采用局部改进和重新装箱策略，提高了箱利用率

。

另一方面，超启发式方法在启发式算法层面进行搜索，而非直接操作单个装箱。它们会迭代地选择、调整或组合低层启发式（通常是构造规则和局部改进移动），以生成一系列候选解；选择过程则由解质量的反馈引导 [22, 13, 6]。基于遗传算法的超启发式方法在 [22, 13]中被提出，用于为二维不规则装箱问题演化出有效的启发式组合。最近， [6]提出了一种动态分层超启发式框架，其中启发式选择由实例特征引导，并可递归调用简单启发式或其他超启发式。

1.2. 贡献与组织结构

正如 [15], 已经指出的，尽管黑盒方法可能产生质量更高的解，但实践者往往更青睐简单、确定且透明的算法，而非运行时间较长的黑盒方法。因此，由于行为可预测且计算开销低，简单的构造启发式在对时间和成本敏感的工业应用中颇具吸引力。受这一实际偏好的启发，本文继续研究二维不规则装箱问题的构造算法，并作出以下贡献。

- 我们发现，现有构造启发式算法在处理零件之间具有较强几何相容性的切割实例时存在一个关键局限。基于这一发现，我们提出一种合并策略，用于识别相容件并将其组合成更大、更规则的超形状。

- 我们为二维不规则装箱问题开发了一种新的构造启发式算法，即 *MergeDJD*，对 DJD 启发式算法进行了大幅扩展[15]。首先，MergeDJD 将所提出的合并策略整合为预处理阶段。此外，MergeDJD 采用改进的放置启发式算法（ICAD），以更好地处理非凸形状和组合形状，同时保留构造算法的确定性和低开销。

- 我们在 1,089 个基准实例上进行了大量计算实验。MergeDJD始终优于DJD，并在515个实例上取得了新的已知最优值。消融研究进一步验证了所提出组件的有效性。完整实现及可视化支持已公开发布于 https://github.com/hfu622/2DIBPP-MergeDJD。

本文结构如下。第 2 节详细介绍二维不规则装箱问题。在第 3节中，我们重新审视DJD并指出其局限性，随后介绍所提出的MergeDJD整体框架。接着，在第4节和5节中，我们分别详细介绍MergeDJD的两个主要组件：零件合并算法和ICAD放置启发式算法（位于DJD中）。第 6 节报告实验结果，并进行全面的比较与分析。最后，第 7 节总结全文。

2. 问题描述

我们正式定义二维不规则装箱问题（2DIBPP），并说明该问题中几何数据的表示方式。

二维不规则装箱问题的输入是一个元组 $(\mathcal{P},\ W,\ L,\ \mathcal{A})$，其中 $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ 是一组 n 零件，$W,\ L \in \mathbb{R}^+$ 分别表示每个箱子的宽度和长度，而 $\mathcal{A} \subseteq [0, 2\pi)$ 是一个有限的允许旋转角集合。

我们假设每个输入零件 $P \in \mathcal{P}$ 都是一个无孔的简单多边形，由按序排列的二维顶点表示。不失一般性，每个零件的第一个顶点固定在 (0, 0)，并称为参考点。顶点按顺时针或逆时针顺序给出。例如，如图 1 所示，若零件是一个三角形，则可表示为 $\langle(0, 0), (1, 1), (2, 4)\rangle$。每个箱子由角点序列 $\langle(0, 0), (L, 0), (L, W), (0, W)\rangle$ 表示。我们假设每个零件至少可以在一种允许的旋转角度下放入箱中。此外，我们假设可用箱子数量的上限为 b，且 $b \geq n$。

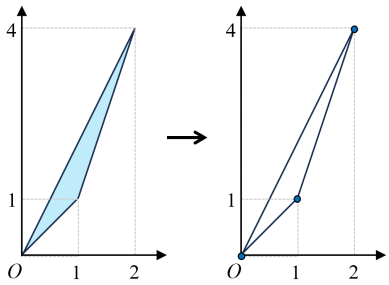

图 1：三角形零件可表示为 $\langle(0, 0), (1, 1), (2, 4)\rangle$。

二维不规则装箱问题要求在 $\mathcal{P}$ 中对零件进行可行且高成本效益的布局。零件子集 $\mathcal{P}' \subseteq \mathcal{P}$ 的布局由两个部分组成：每个零件的分配 到某个箱子，以及各零件在每个

箱子内的布局 。零件的布局同时指定其平移和旋转角$\alpha \in \mathcal{A}$。若每个零件恰好分配到一个箱子，所有零件均完全位于其所分配箱子的边界内，且分配到同一箱子的任意两个零件均不重叠，则该布局是可行的。

为了表示一个布局，我们将箱子索引为 $[b] = \{1, \ldots, b\}$，其中b 是可用箱子数量的上界。我们定义一个位置三元组，对应于零件 $P \in \mathcal{P}$，记作 $v_P = （i，\ \vec{v}，\ \alpha）$，其中 $i \in [b]$ 表示 P 被放置在第 i 个箱中，$\vec{v}$ 是平移（移动）向量，$\alpha \in \mathcal{A}$ 是旋转角。说明平移与旋转效果的示例见图 2。

子集 $\mathcal{P}' \subseteq \mathcal{P}$ 的一个布局 由位置三元组集合V表示，使得对于每个 $P \in \mathcal{P}'$，都存在唯一的 $v_P = （i，\ \vec{v}，\ \alpha）\in V$。若$\mathcal{P}' \subset \mathcal{P}$，则该布局是部分的；若 $\mathcal{P}' = \mathcal{P}$，则该布局是完整的。由于所有箱子都相同且可互换，我们不失一般性地假设箱子被连续使用：箱 i （$1 < i \leq b$）只有在箱 $i - 1$ 也被使用时才能出现在布局中。我们用 N 表示完整摆放中所使用的箱子数量。

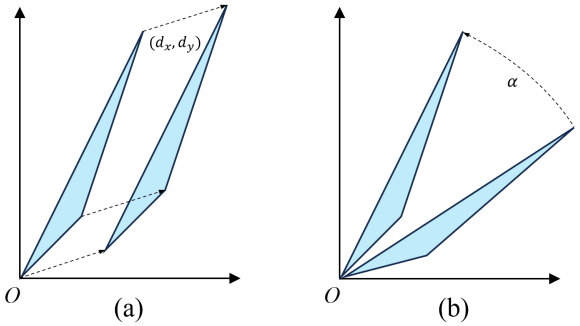

图2：（a）零件的所有点都平移了（d_x，d_y）。（b）零件以其参考点 O 为中心逆时针旋转 α 度。

2.1. 摆放度量

令 $Area(P) \in \mathbb{R}^+$表示零件 P 的面积，可使用标准多边形面积公式 [18] 计算。给定一个可行放置，令$\mathcal{P}_i$ 为分配到第 i 个箱中的零件集合。箱i 的利用率 定义为

$$U_i = \frac{\sum_{P \in \mathcal{P}_i} Area(P)}{W \times L}.$$

二维不规则装箱问题的主要目标是使所用箱子数量最少，即 N。此外，通常还会使用若干指标来评估

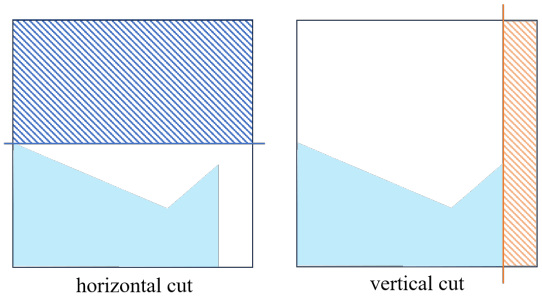

图 3：水平切割与垂直切割示意图。阴影区域表示可复用剩余区域。

完整且可行的布局质量，尤其是在比较使用相同数量箱子的解时。

- 该 F 指标。该指标定义为

$$F = \frac{\sum_{i=1}^{N} U_i^2}{N},$$

其中 U_i 是箱 i的利用率。给定两个具有相同 N的布局，该指标倾向于选择箱子利用率较高或几乎为空的方案。 F 指标由 [12] 提出，随后被广泛用于后续研究中 [26]。

- 该 K 指标。该 K 指标定义为

$$K = N - 1 + R^*,$$

其中 R^* 表示利用率最低的箱子的有效利用率。具体而言，对于利用率最低的箱子，首先通过施加水平或垂直断头台式切割，去除任何完全未被占用的可复用剩余区域，切割方向的选择以最大化剩余矩形区域为准。随后， R^* 的值计算为剩余矩形中零件所占面积与箱子面积之比。因此， K 可解释为完全利用的箱子数量加上利用率最低的箱子的分数利用率。图 3给出了一个示例。该指标尤其适用于区分使用相同箱子数量、但最后一个箱子装箱效率不同的解 [7, 26]。

3. MergeDJD 框架：二维不规则装箱问题

3.1. DJD 启发式算法综述

总体而言，DJD 算法以贪心的逐箱方式将零件装入箱子。所有零件首先按面积降序排序。随后，该算法依次开启箱子，并采用两阶段构造过程填充每个箱子。

对于给定的箱，DJD 首先根据单个零件执行贪心填充阶段。按照排序顺序，算法反复选择最大的未放置零件，并尝试将其放入当前箱中。可行性及精确的放置位置由一个称为构造性最大邻接法 (CAD)[15]的放置子程序确定。CAD 倾向于选择能最大限度增加与已放置零件及箱子边界邻接性的布局。若零件成功放置，则将其永久分配到当前箱中。该过程持续进行，直到没有其他零件能够放入，或箱子的占用面积达到箱子面积的三分之一，如算法1的第 4–8 行所示。

在初始贪心阶段之后，DJD 尝试使用剩余零件组成的小组进一步填充当前箱。具体而言，算法考虑由一个、两个或三个零件组成的有序组。对于每个组，都会调用 CAD，以测试该组能否共同放入箱中。一个称为废料的阈值参数用于控制可接受的剩余空白面积。如果某个组在当前废料阈值下能够达到足够高的利用率，则接受该组并将其永久放置。如果找不到可行的组，则逐步提高废料阈值，从而允许产生剩余空间更大的方案。该基于组的填充过程对应于算法 1的第 9–17 行。

当无法再放置其他零件或零件组时，当前箱子便会关闭，并开启一个新箱子。算法重复相同的步骤，直到所有零件都已分配到箱子中。所有布局和分配决策一经作出便不可更改。因此，DJD 是一种构造启发式，它会不可撤销地确定箱体分配以及每个箱子内零件的布局。

3.2. 二维不规则装箱问题的新算法

DJD 算法高效，在实验中几乎可实时运行 [15]。然而，在工业应用中，部分输入零件属于中间组件，之后还会通过胶粘、缝合或焊接重新组装。因此，这些零件具有互补边，可以无缝拼接成更大的形状，且不会留下间隙。这种情况经常出现在

算法 1 DJD 算法

输入： 一组零件 $\mathcal{P}$，每个箱的宽度为 W，长度为 L。
输出： 一个完整摆放 V。

$waste \leftarrow 0$，已摆放面积 $\Phi_i \leftarrow 0$，布局 $V_i \leftarrow \emptyset$，$\forall i \in [1, b]$
将 $\mathcal{P}$ 中的零件按面积降序排列
当 $\mathcal{P} \neq \emptyset$ **时执行**
    对于 $j = 1, \ldots, |\mathcal{P}|$ **执行** ▷ 将第 i 个箱填充至三分之一
        $flag,\ v_{P_j} \leftarrow \text{CAD}(i,\ \langle P_j \rangle)$
        如果 $flag =$ 成功 **则**
            $\mathcal{P} \leftarrow \mathcal{P} \setminus \{P_j\},\quad \Phi_i \leftarrow \Phi_i + Area(P_j),\quad V_i \leftarrow V_i \cup \{v_{P_j}\}$
        如果 $\Phi_i \geq W \times L/3$，**则跳出**
    当 $waste < W \times L - \Phi_i$ **时** ▷ 使用一组 i 填充第零件
        $\mathcal{G}_1 \leftarrow \mathcal{P}, \mathcal{G}_2 \leftarrow \mathcal{P} \times \mathcal{P}, \mathcal{G}_3 \leftarrow \mathcal{P} \times \mathcal{P} \times \mathcal{P}$
        $flag \leftarrow failure$
        对于每组有序块 $\vec{O} \in \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3$ **执行**
            $flag, \{v_P \mid P \in \vec{O}\} \leftarrow \text{CAD}(i,\ \vec{O})$
            如果 $flag =$ 成功且 $\Phi_i + \sum_{P \in G} Area(P) \geq W \times L - waste$ **then**
                $\mathcal{P} \leftarrow \mathcal{P} \setminus \{P \mid P \in \vec{O}\},\quad \Phi_i \leftarrow \Phi_i + \sum_{P \in \vec{O}} Area(P)$
                $V_i \leftarrow V_i \cup \{v_P \mid P \in \vec{O}\},\quad waste \leftarrow 0$，**中断**
        如果 $flag =$ 失败 **则** $waste \leftarrow waste + W \times L/20$
    $i \leftarrow i + 1$
返回 $V = \{V_1, V_2, \ldots, V_b\}$

所谓的切割实例中，其中零件是通过分割一块与箱尺寸完全相符的矩形板材获得的。

例如，图 4展示了一个切割实例，其中零件可以完美地组合成单个矩形。然而，DJD并未识别出这种结构，因为邻接性仅依据布局过程中的箱利用率来判定。

为解决这一局限，我们针对 DJD 引入了一种预处理算法，即部件合并算法。该算法尝试识别应当一同放置的零件，随后对合并后的零件应用部件放置算法。因此，如图5所示，整体算法 MergeDJD 由两个主要组件构成：(i) 部件合并和 (ii) 部件放置。如前所述，部件合并组件会迭代测试零件组合，并确定

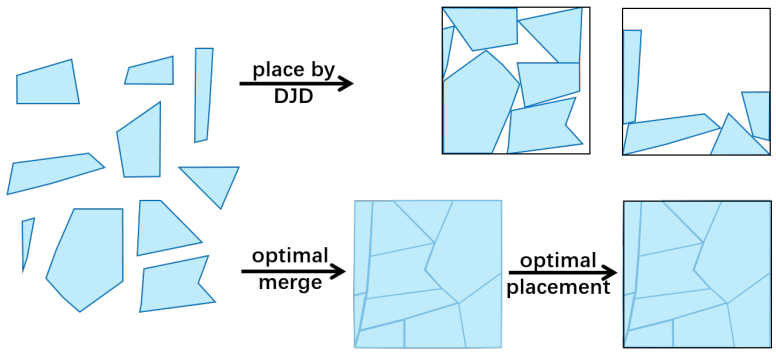

图4：切割实例中的所有零件可以合并成一个矩形。然而，DJD 未能识别出这一结构，导致解的质量较差。

它们是否能够组成一个较大的形状；如果可以，则用该形状替换原先各个独立的零件。部件放置组件则为合并后的零件集合确定完整且可行的布局。具体而言，它使用 DJD 确定箱体分配和放置顺序。不过，最终的放置位置由改进的 CAD 启发式算法（ICAD）计算得出。ICAD 在原始 CAD 的基础上加入了右下策略和额外的初始放置点。对于非凸零件和合并后的零件，它能实现更有效的布局。

4. 部件合并算法

给定一组零件，可以通过多种方式将它们组合成更大的复合部件。即使只考虑两个片段，不同的几何构型也可能产生不同的组合结果。在本节中，我们研究部件合并算法，该算法限制了两个零件的组合方式，同时保持了有效性。

直观而言，部件合并算法会贪心地尝试通过重叠角点来组合成对的零件。一旦两个零件成功组合，生成的组合件 便会取代原来的零件，并在后续迭代中作为单个实体处理。组合件由原始零件的点序列的并集表示。该过程支持多轮两两组合，并动态更新输入零件集合。对于由矩形多边形派生的切割实例，该算法原则上可以通过连续组合还原出矩形形状。请注意，在顺序过程中，组合件中零件的相对位置是固定的。

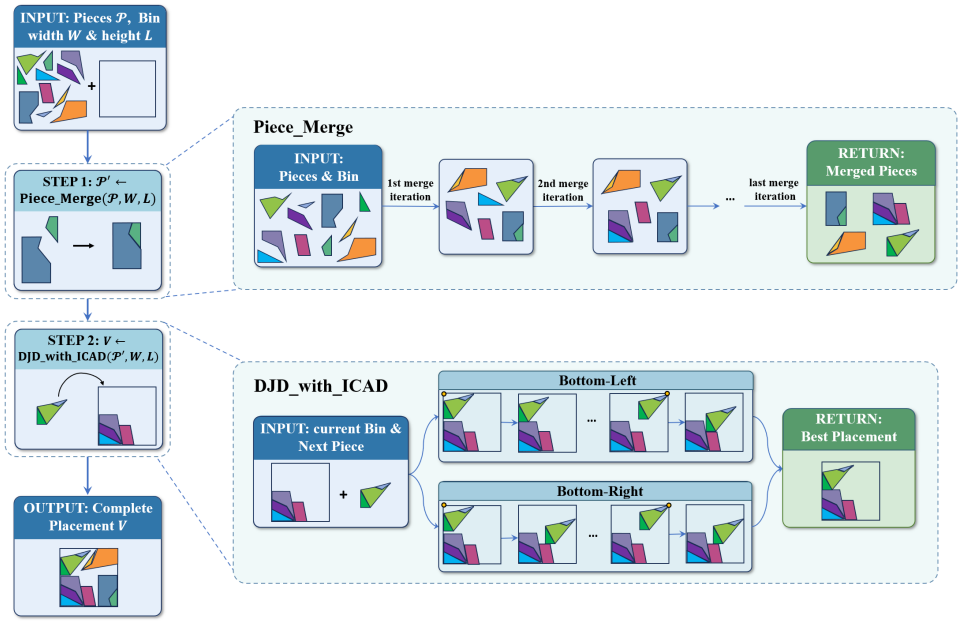

图5：所提出的 MergeDJD 算法的整体流程。片段合并过程详见算法2；带 ICAD 的 DJD 过程详见算法 1 和算法 3。

4.1. 部件合并算法详解

我们在算法 2中给出了部件合并算法的伪代码。具体而言，该算法维护一个零件工作集 $\mathcal{P}'$，其初始值等于输入零件集合 $\mathcal{P}$。只要至少存在一个可接受的组合件，它就会反复尝试组合成对的零件（第 2 行）。每次迭代开始时，会按面积降序对 $\mathcal{P}'$ 中的零件进行排序。然后，算法枚举所有无序对 (P_i, P_j)，其中 P_i 在排序列表中位于 P_j 之前。

从第 6 行到第 15 行，算法穷举检查两个零件 P_i 和 P_j 是否能在给定的箱体尺寸和阈值 T 下组合。对于一对零件（P_i，P_j），算法枚举 P_j 的所有允许的旋转 $\alpha \in \mathcal{A}$。对于每种旋转，都会考虑 P_i 的所有顶点（x_i，y_i）和 P_j 的所有顶点（x_j，y_j）。平移零件 P_j，使选定的两个顶点重合，从而得到候选组合件 P_k。如果 P_i 和 P_j 不重叠，且 P_k 的包围盒能放入空箱中，则 P_k 被视为可行的组合片段。可行组合片段的示例见图 6。

获得可行的组合片段后，将进一步评估其适应度值

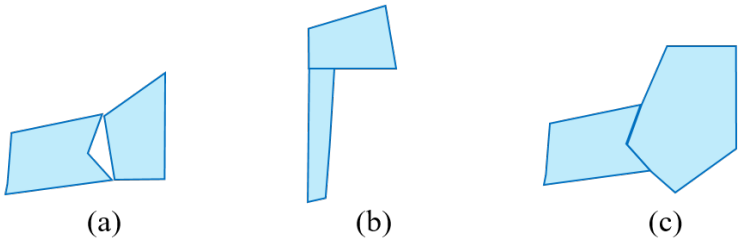

图 6：由两个片段合并而成的零件示例。(a) 两个片段在两个点处接触。(b) 两个片段沿一条线段重叠。(c) 两个片段在角处完美贴合。

。如果适应度值不低于阈值 T，则接受该组合件并终止枚举。从集合 $\mathcal{P}'$ 中移除原始零件 P_i和 P_j，并将组合件 P_k 插入集合中。然后，算法继续在更新后的工作集上尝试其他组合。

当 $\mathcal{P}'$ 中的任意一对零件都无法进一步合并时，过程便会结束（第2行）。由于每次成功合并都会使 $\mathcal{P}'$ 中的元素数量减少一个，因此合并尝试次数不会超过输入零件的数量 $|\mathcal{P}|$。随后返回最终的组合零件集合 $\mathcal{P}'$。

4.2. 评估组合件

给定两个零件 P_1 和 P_2，其中每个零件都可能是组合件，令$P_{1,2}$ 表示将它们合并后得到的组合件。我们定义一个适应度函数，用于评估用$P_{1,2}$ 替换 P_1 和 P_2 是否有利于后续布局。适应度函数定义如下：

$$f(P_1, P_2, P_{1,2}) = g_{vtx}(P_1, P_2, P_{1,2}) + g_{peri}(P_1, P_2, P_{1,2}) + g_{rect}(P_{1,2}). \quad (1)$$

函数 f 是三个分量等权重之和。前两个分量评估合并P_1 和 P_2所实现的结构简化，第三个分量则衡量组合件的全局规则性。注意，g_{rect} 仅取决于 $P_{1,2}$。

- g_{vtx}：顶点数减少。给定一个多边形 P，令 $|P|$ 表示 P的顶点数。顶点减少项定义为

$$g_{vtx}(P_1, P_2, P_{1,2}) = \max(0, \max(|P_1|, |P_2|) - |P_{1,2}|)\,.$$

计算组合件的点数时，例如$|P_{1,\ 2}|$，P_1 和 P_2 共有的相同顶点只计一次。

算法 2 零件合并算法

输入： 零件集合 $\mathcal{P}$，每个箱的宽度 W 和高度 L，
阈值参数 $T \in \mathbb{R}$。
输出： 合并后的新零件集合 $\mathcal{P}'$。

初始化 $\mathcal{P}' \leftarrow \mathcal{P}$, $cnt \leftarrow 0$
当 $|\mathcal{P}'| \neq cnt$ **时**
  按面积降序对 $\mathcal{P}'$ 中的零件进行排序，$cnt \leftarrow |\mathcal{P}'|$
  对于 $i = 1, \ldots, |\mathcal{P}'| - 1$ **执行**
    对于 $j = |\mathcal{P}'|, \ldots, i+1$ **执行**
      对于每种旋转 $\alpha \in \mathcal{A}$ **执行** ▷ 穷举检查是否两个片段可以合并
        将 P_j 顺时针旋转 α 度
        对于每个点 (x_i, y_i)，遍历 P_i **执行**
          对于每个点 (x_j, y_j)，遍历 P_j **执行**
            $d_x \leftarrow x_j - x_i$, $d_y \leftarrow y_j - y_i$
            固定 P_i，并沿向量 (d_x, d_y) 移动 P_j，得到 P_k
            如果 P_i 和 P_j 不重叠，则 $x_{\overline{P_k}} - x_{\underline{P_k}} \leq W$, $y_{\overline{P_k}} - y_{\underline{P_k}} \leq L$ 和 $f(P_i, P_j, P_k) \geq T$ **那么** ▷ 方程（1）
              $\mathcal{P}' \leftarrow (\mathcal{P}' \cup \{P_k\}) \setminus \{P_i, P_j\}$, **跳出**
          如果 $\{P_i, P_j\} \nsubseteq \mathcal{P}'$ **则跳出**
        如果 $\{P_i, P_j\} \nsubseteq \mathcal{P}'$ **则跳出**
返回 $\mathcal{P}'$

直观来说，顶点较少的形状更简单。因此，应优先选择能显著减少顶点数量的合并。在合并非凸零件时，这一准则尤其有效，因为表示凹结构通常需要许多顶点。

- g_{peri}：周长缩减。多边形的边是连接相邻顶点的线段。令 $Seg(P)$ 表示 P 中多边形的所有边界线段的集合。线段e的长度记为 $len(e)$。给定两个零件 P_1 和 P_2，组合件 $P_{1,2}$中的两条线段 e_i的重叠长度，e_j 通过 [15], 中的算法11计算，记为$overlap\ len(P_{1,2}, e_i, e_j)$。周长缩减项定义为

$$g_{peri}(P_1, P_2, P_{1,2}) = \sum_{e_i \in Seg(P_1)} \sum_{e_j \in Seg(P_2)} \left(\frac{overlap_len(P_{1,2}, e_i, e_j)}{\max(len(e_i), len(e_j))} \right)^2.$$

该项衡量了 P_1 和 P_2的周长在组合件中的重叠程度。由于周长较短的形状通常更简单，因此更倾向于选择能够缩短总周长的合并操作。

- g_{rect}：的全局规则性$P_{1,\ 2}$。令x和 x¯分别表示P_1顶点的 x坐标最大值和最小值，并以类似方式定义y 和 y 。$P_{1,\ 2}$ 的包络矩形是由这些极值定义的轴对齐矩形。规则性项定义为

$$g_{rect}(P_{1,2}) = \left(\frac{Area(P_{1,2})}{(\overline{x} - \underline{x})(\overline{y} - \underline{y})} \right)^2 .$$

该项用于评估组合件与矩形的相似程度。直观而言，形状越接近矩形，在后续装箱阶段就越有可能实现高效布局。

5. 改进 CAD 放置策略

如前所述，放置启发式，即 CAD 算法，在 DJD 算法中发挥着重要作用 [15]。CAD 算法最初由以下作者提出 [23, 8]。在 CAD 中，主要有两个关键步骤：设置初始位置和确定放置策略。

设置初始位置。对于待放入箱 i的零件 P 及其旋转 α ，CAD 构造一个二维点集合，记为$cand$，用作该零件参考点的潜在放置位置。首先，CAD 将箱的四个角(0，0)，(0，W)，(0，L)，(W，L)加入$cand$。对于箱中已放置的每个零件 P' ，CAD 会引入五个点($\bar{x}$，0)、(0，$\bar{y}$)、(x，$\bar{y}$)、($\bar{x}$，$\bar{y}$)、($\bar{x}$，y)，其中 $\bar{x}$ (或 x)和 $\bar{y}$(或 y)分别是 x 坐标和 y 坐标中的最大值（或最小值）。

放置策略。对于每个初始位置（x_0，y_0）$\in cand$，CAD 首先将零件 P 的参考点移动到该位置。移动后，如果任意两个零件都没有重叠区域，且 P 的坐标位于箱内，那么此次合并就是合法的。接下来，CAD 反复将零件P向下、再向左滑动，直到无法继续滑动为止。不失一般性，假设该位置为（x_t，y_t）。整个过程中，零件旋转固定为 α。此移动会生成一个位置三元组P，表示为（i，$\vec{v} =$（x_t，y_t） α）。并集形状的 g_{peri}度量

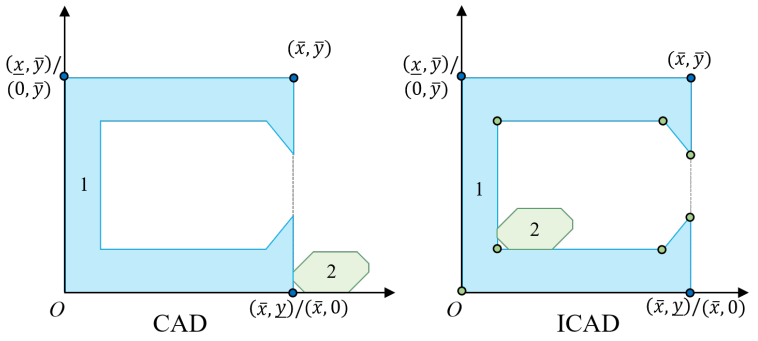

图7：初始位置影响的示意图。左：CAD生成的布局。放置零件1后，CAD为零件2生成的初始位置以蓝色标出。右：ICAD生成的布局。放置零件1后，ICAD为零件2生成的扩展初始位置以绿色标出。通过引入额外的初始位置，ICAD能够在凹区域内进行放置，从而提高利用率。

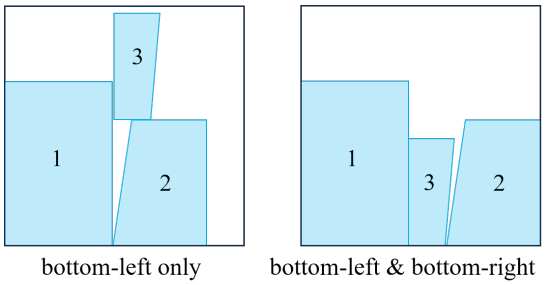

图 8：仅左下策略（CAD采用）和左下与右下混合策略（ICAD采用）生成的布局。在混合策略中，零件1和零件3采用左下策略，零件2采用右下策略，从而实现更紧凑的布局。

放置 P 后零件的度量根据位置三元组计算。CAD 枚举 P 的所有初始位置，并返回具有最高 g_{peri}的位置三元组。

5.1. 将*CAD*改进为*ICAD*

我们通过增加初始放置位置集合并引入额外的放置策略，改进了 CAD 算法。改进后算法的伪代码见算法3。对于初始位置，我们进一步增加了 $cand$ 以及已放置零件上的点。如图 7所示，CAD 中使用的 $cand$无法将零件放置在

算法3 ICAD 算法

输入： 一个箱 i，以及一组有序块 $\vec{O}$.
输出： 成功并返回放置位置，或失败。

$cand \leftarrow \{(0,\ 0),\ (0,\ L),\ (W,\ 0),\ (W,\ L)\}$
对箱 i 中的每个已放置块 P **执行**
    $cand \leftarrow cand \cup \{(0, y_{\overline{P}}), (x_{\underline{P}}, y_{\overline{P}}), (x_{\overline{P}}, y_{\overline{P}}), (x_{\overline{P}}, y_{\underline{P}}), (x_{\underline{P}}, 0)\} \cup P$
将当前装有已放置零件的箱表示为一个组合件 P_B
对于 $j = 1, \ldots, |\vec{O}|$ **执行**
    $cur_best \leftarrow -1$
    对于 $\alpha \in \mathcal{A}$ **执行**
        将零件 P_j 绕其参照点逆时针旋转 α 度，
        点。
        对于每个初始位置（x，y）$\in cand$ **执行**
            如果将 P_j 的参考点置于（x，y）后，任意两个零件都不
            重叠，则 $x_{P_j}, \overline{x_{P_j}} \in [0, W]$ 和 $y_{P_j}, \overline{y_{P_j}} \in [0, L]$ **那么**
                $v_{P_j,\ L} \leftarrow bottom\ left\ strategy\ (i,\ P_j,\ (x,\ y))$
                将应用 $v_{P_j,\ L}$ 后的箱表示为一个组合件
                P_{BL}
                如果 $g_{peri}(P_B,\ P_j,\ P_{BL}) > cur\ best$ **则**
                    $v_{P_j} \leftarrow v_{P_j,\ L}$, $cur_best \leftarrow g_{peri}(P_B,\ P_j,\ P_{BL})$
                $v_{P_j,\ R} \leftarrow bottom\ right\ strategy(i,\ P_j,\ (x,\ y))$
                表示将 $v_{P_j,\ R}$ 作为组合件应用后的箱
                P_{BR}
                如果 $g_{peri}(P_B,\ P_j,\ P_{BR}) > cur\ best$ **则**
                    $v_{P_j} \leftarrow v_{P_j,\ R}$, $cur_best \leftarrow g_{peri}(P_B,\ P_j,\ P_{BR})$
        如果 $cur\ best = -1$ **则返回** 失败
返回成功，$\{v_P \mid \forall P \in \vec{O}\}$

较大的凹区域内，导致利用率较低。相比之下，增加后的 $cand$ 使我们能够利用这一较大的凹区域，从而可能显著提高利用率。

在布局策略方面，我们通过采用向下再向右移动来改进 CAD，即将零件 P 先向下、再向右滑动，直至无法继续滑动。此移动会生成一个位置三元组，用于表示 P的布局。已放置块 P 被视为一个零件，而箱及其中所有先前放置的零件则被视为另一个零件。这两个零件构成一个组合件。随后使用 g_{peri} 度量对该组合件进行评估。如图 8所示，采用向下再向右移动

能够生成更加紧凑的布局。

6. 计算实验

6.1. 数据集

表 1：JP1 和 JP2 实例的统计数据。所有实例的箱体尺寸均为 1000×1000。

JP1	件数	#实例数	最优 #箱子数	JP2	件数	#实例	最优箱数
Conv A	30	30	3	NConv A	35-50	30	3
Conv B	30	30	10	NConv B	40-52	30	10
Conv C	36	30	6	NConv C	42-60	30	6
Conv D	60	30	3	NConv F	35-45	30	2
Conv E	60	30	3	NConv H	42-60	30	12
卷积 F	30	30	2	NConv L	35-45	30	3
卷积 G	36	30	$\leq$15	NConv M	45-58	30	5
卷积 H	36	30	12	NConv O	33-43	30	7
Conv I	60	30	3	NConv S	17-20	30	2
Conv J	60	30	4	NConv T	30-40	30	10
Conv K	54	30	6	NConv U	20-33	30	5
Conv L	30	30	3	NConv V	15-18	30	5
Conv M	40	30	5	NConv W	24-28	30	4
Conv N	60	30	2	NConv X	25-39	30	3
卷积 O	28	30	7	NConv Y	40-50	30	6
卷积 P	56	30	8	NConv Z	60	30	12
卷积 Q	60	30	15				
卷积 R	54	30	9				
总计		540				480	

我们采用两个标准数据集，即拼图（*JP*）实例和嵌套实例，与现有研究相同 [26]。这些数据集可在 ESICUP 网站上获取 https://github.com/ESICUP/datasets。

- *JP*：JP 数据集（也称为 Terashima 数据集）包含两个实例集合 JP1 和 JP2，分别由 [15] 和 [13] 提出。这些实例中的零件均从若干箱中切割而成，因此最优解的 F 值为 100%。JP1 数据集包含 18 个类别，每个类别包含 30 个实例，共计 540 个实例。JP1 中的所有零件均为凸多边形。JP2 数据集包含 16 个类别，每个类别有 30 个实例，共计 480 个实例。与 JP1 不同，JP2 数据集同时包含凸零件和非凸零件。JP 实例的统计数据如表 1所示。

- 套料：该套料数据集源自23个不同的条带装箱问题实例，其中假设箱子的长度无限。该套料数据集包含凸多边形和非凸多边形。

为使这些实例适用于装箱问题，我们遵循 [16] 中的设置，并定义三种尺寸的箱子。令 d_{max} 表示给定实例中所有零件在初始方向上的最大长度或宽度。三种容器尺寸设置如下：小箱（**SB**）的长和宽均设为 $1.1d_{max}$，中箱（**MB**）设为 $1.5d_{max}$，大箱（**LB**）设为 $2d_{max}$。套料实例的统计信息见表2。

表 2：套料实例统计

实例	件数	箱体尺寸 $W = L$			实例	件数	箱体尺寸 $W = L$		
		SB	MB	LB			SB	MB	LB
albano	24	3337.4	4551	6068	poly3b	45	14.3	19.5	26
dighe1	16	72.6	99	132	poly4a	60	14.3	19.5	26
dighe2	10	77	105	140	poly4b	60	14.3	19.5	26
fu	12	15.4	21	28	poly5a	75	14.3	19.5	26
han	23	25.3	34.5	46	poly5b	75	14.3	19.5	26
jakobs1	25	8.8	12	16	shapes0	43	15.4	21	28
jakobs2	25	17.6	24	32	shapes1	43	15.4	21	28
mao	20	1206.7	1645.5	2194	shapes2	28	5.5	7.5	10
poly1a	15	14.3	19.5	26	衬衫	99	26	19.5	14.3
poly2a	30	14.3	19.5	26	swim	48	2133.6	2909.5	3879.3
poly2b	30	14.3	19.5	26	裤子	64	64.9	88.5	118
poly3a	45	14.3	19.5	26					

6.2. 基准算法

为评估 MergeDJD 的有效性，我们将其与原始 DJD 算法[15]进行比较。DJD 和 MergeDJD 均使用 Java 实现。所有实验均在一台配备英特尔® 酷睿™ i7-11800H @ 2.30GHz 处理器和 16 GB 内存的计算机上进行。

在收集已知最优值（BKV）的过程中，我们观察到 F 指标的计算在现有研究中并不总是一致。尽管 F 通常被定义为箱子 [12, 26], 利用率平方的平均值，如第2.1节所述，但一些现有工作似乎报告了采用其他计算方式得到的值，例如对利用率取平均值而不进行平方。因此，文献中报告的 F 值并不总是可以直接比较。针对若干套料实例，在 Yao 等人 [25], 和 Luo 等人 [14]所报告的结果中也可观察到类似差异。然而，由于缺少这些算法的源代码或其最终结果数据，不宜直接比较 F 这些工作中所报告的值。因此，在报告 F 和 K的已知最优值时，我们未纳入 Yao 等人 [25], Luo 等人 [14], 的结果，尽管这

可能会忽略一些更优的解。（事实上，这进一步凸显了该问题对开源实现和统一基准数据的需求。）此外，我们始终遵循 F指标在第2.1 节中给出的定义，并在所有实验中保持一致。

6.3. 性能比较

我们比较了基准算法在 JP1、JP2、nesting-SB、Nesting-MB 和 Nesting-LB实例上的整体性能。结果分别列于表 3、4、5、6和7 中。

表 3 JP1实例上的比较

粗体数值表示所比较算法中的最佳结果。带下划线的数值表示达到或优于已知最优值（BKV）的结果。为方便起见，BKV 的来源标记为 a: [26], b: [16], c: [3], d: [1], e: [10], f: [6], g: [15]。

实例 set	ave 件数	BKV		DJD			MergeDJD				
		F	K	F	K	time	平均件数 合并后	F	K	time	合并 time
Conv A	30	0.869^{f}	3.435^{a}	0.586	4.000	0.62	14.2	**0.763**	**3.371**	**0.30**	0.05
Conv B	30	1.000^{a}	10.000^{a}	0.899	10.900	0.51	20.0	**1.000**	**10.000**	**0.14**	0.01
Conv C	36	0.970^{f}	6.580^{a}	0.739	7.198	0.57	14.6	**0.917**	**6.207**	**0.24**	0.07
Conv D	60	0.610^{d}	3.596^{a}	0.428	5.000	**0.66**	30.7	**0.601**	**3.879**	1.98	0.41
Conv E	60	0.579^{a}	3.836^{a}	0.327	5.667	**0.63**	30.8	**0.503**	**4.285**	2.18	0.46
卷积 F	30	0.648^{f}	2.381^{a}	0.460	3.067	0.56	12.4	**0.616**	**2.435**	**0.31**	0.07
卷积 G	36	0.822^{a}	13.131^{a}	0.814	13.458	0.50	26.6	**0.828**	**13.118**	**0.25**	0.05
卷积 H	36	1.000^{a}	12.000^{a}	0.908	12.965	0.50	24.0	**1.000**	**12.000**	**0.16**	0.01
Conv I	60	0.697^{d}	3.250^{a}	0.621	4.000	**0.59**	60.0	**0.639**	**3.810**	8.96	0.01
Conv J	60	0.701^{b}	4.467^{a}	0.639	5.100	**0.59**	40.7	**0.680**	**4.875**	1.92	0.03
Conv K	54	0.923^{f}	6.641^{a}	0.664	7.800	0.58	33.8	**0.840**	**6.460**	**0.47**	0.15
Conv L	30	0.630^{d}	3.667^{a}	0.438	4.894	0.55	13.2	**0.716**	**3.492**	**0.50**	0.14
Conv M	40	0.723^{d}	5.936^{a}	0.540	7.155	0.62	16.6	**0.794**	**5.582**	**0.52**	0.17
Conv N	60	0.688^{b}	2.360^{a}	0.467	3.000	**0.82**	30.0	**0.519**	**2.793**	2.20	0.26
卷积 O	28	0.994^{f}	7.005^{a}	0.844	7.878	0.51	8.7	**0.898**	**7.345**	**0.32**	0.05
卷积 P	56	0.852^{b}	9.190^{a}	0.632	10.610	**0.66**	28.7	**0.768**	**9.243**	1.34	0.47
卷积 Q	60	1.000^{a}	15.000^{a}	**1.000**	**15.000**	0.49	30.2	**1.000**	**15.000**	**0.26**	0.01
卷积 R	54	0.946^{f}	10.064^{a}	0.716	10.963	0.77	26.0	**0.872**	**9.573**	**0.52**	0.20
平均值		0.814	6.808	0.651	7.703	**0.60**	25.6	**0.775**	**6.859**	1.26	0.14
min		0.579	2.360	0.327	3.000	0.49	8.7	**0.503**	**2.435**	**0.14**	0.01
Q1		0.690	3.614	0.485	4.921	0.52	15.1	**0.649**	**3.827**	**0.27**	0.03
Q2		0.837	6.258	0.636	7.176	0.58	26.3	**0.781**	**5.894**	**0.48**	0.07
Q3		0.964	9.798	0.795	10.827	**0.63**	30.6	**0.892**	**9.491**	1.78	0.19
max		1.000	15.000	**1.000**	**15.000**	**0.82**	60.0	**1.000**	**15.000**	8.96	0.47

表格 3-4 中的第 1-2 列分别给出每个实例集的名称及各实例集中的零件平均数量。第 3-4 列报告各实例集上的 BKV 结果，包括 F指标（越高越好）和K 指标（越低越好）。第 5-12 列汇总了各对比算法所得结果，包括算法的总运行时间（$time$）。对于 MergeDJD，我们还提供了合并后的零件平均数量及合并时间。最后，表格最后六行汇总了各列的结果，包括平均值、最小值、最大值以及三个四分位数 Q1、Q2 和 Q3。 Q_1和 Q_3 分别对应第 25 和第 75 百分位数，而 Q_2 表示中位数。这些四分位数以稳健的方式刻画了实验结果的集中趋势和离散程度。

表4：JP2实例上的比较。粗体值表示所比较算法中的最佳结果。带下划线的值表示达到或优于已知最优值（BKV）的结果。

实例 set	ave 件数	BKV F	BKV K	DJD F	DJD K	DJD time	MergeDJD 平均件数 合并后	MergeDJD F	MergeDJD K	MergeDJD time	MergeDJD 合并 time
Nconv A	41.7	0.680[b]	3.527[a]	0.507	4.500	0.64	14.2	**0.680**	**3.527**	**0.44**	0.08
Nconv B	47.3	0.841[a]	10.808[a]	0.745	11.981	0.84	17.9	**0.746**	**11.646**	**0.27**	0.04
Nconv C	50.0	0.702[a]	7.007[a]	0.612	8.099	0.74	14.8	**0.880**	**6.311**	**0.36**	0.11
Nconv F	40.0	0.551[b]	2.413[a]	0.437	3.200	0.71	12.6	**0.578**	**2.511**	**0.49**	0.11
Nconv H	50.0	0.895[a]	12.654[a]	0.802	13.771	0.81	22.1	**0.828**	**13.315**	**0.27**	0.04
Nconv L	40.0	0.591[a]	3.771[a]	0.404	5.064	0.67	13.5	**0.718**	**3.495**	**0.48**	0.17
Nconv M	51.0	0.681[b]	6.073[a]	0.492	7.597	0.80	16.4	**0.790**	**5.557**	**0.51**	0.17
Nconv O	38.0	0.879[a]	7.488[a]	0.734	8.556	0.64	9.1	**0.861**	**7.516**	**0.31**	0.07
Nconv S	19.0	0.817[a]	2.185[a]	0.427	3.425	0.54	6.2	**0.842**	**2.233**	**0.30**	0.04
Nconv T	34.6	0.997[a]	10.019[a]	0.853	11.142	0.53	10.1	**0.995**	**10.018**	**0.23**	0.02
Nconv U	26.5	0.894[a]	5.242[a]	0.668	6.516	0.57	5.2	**0.975**	**5.046**	**0.28**	0.03
Nconv V	16.5	1.000[a]	5.000[a]	0.898	5.465	0.50	5.2	**0.950**	**5.117**	**0.23**	0.01
Nconv W	26.0	0.843[d]	4.364[a]	0.605	5.465	0.53	14.6	**0.931**	**4.137**	**0.24**	0.01
Nconv X	32.0	0.649[d]	3.579[a]	0.477	4.666	0.59	11.5	**0.967**	**3.038**	**0.28**	0.03
Nconv Y	45.0	0.743[a]	6.870[a]	0.628	7.965	0.72	14.4	**0.853**	**6.405**	**0.31**	0.07
Nconv Z	60.0	0.893[b]	12.843[a]	0.806	13.830	0.74	20.4	**0.897**	**12.655**	**0.49**	0.20
平均值		0.791	6.490	0.631	7.578	0.66	13.0	**0.843**	**6.408**	**0.34**	0.07
min		0.551	2.185	0.404	3.200	0.50	5.2	**0.578**	**2.233**	**0.23**	0.01
Q1		0.681	3.723	0.488	4.964	0.57	9.9	**0.779**	**3.519**	**0.27**	0.03
Q2		0.829	5.658	0.620	7.057	0.66	13.9	**0.857**	**5.337**	**0.30**	0.06
Q3		0.893	8.121	0.759	9.202	0.74	15.2	**0.936**	**8.141**	**0.45**	0.11
max		1.000	12.843	0.898	13.830	0.84	22.1	**0.995**	**13.315**	**0.51**	0.20

结果。请注意，每个套料数据集都只包含一个实例。因此，表格 5-7 中的第 1-2 列分别给出每个实例的名称及各实例中的零件数量。

我们观察到，在所有 JP1 和 JP2 切割实例中，MergeDJD 在 F 和 K 指标方面始终优于 DJD。此外，在两个数据集上，MergeDJD 在 34 个实例集中有 17 个达到或超过了 F 的已知最优值，并在 34 个实例集中有 20 个达到或超过了 K 的 BKV。鉴于 MergeDJD 的运行时间较短，尽管改进算法需要长得多的运行时间，这些结果仍具有竞争力。关于计算时间，我们观察到，当合并后的零件数量显著少于原始数量时，MergeDJD 甚至可能比 DJD 更快。这是因为零件数量的减少抵消了 MergeDJD 额外步骤带来的时间开销。尤其是，在 MergeDJD 中无法合并任何零件时（例如，在 Conv I 实例集中），MergeDJD 仍优于 DJD。这一结果表明，ICAD排样比 DJD 中原始的 CAD 排样更为有效。

在 SB、MB 和 LB 套料实例中，我们观察到，在 69 个实例中的 61 个实例上，就 F 指标而言，MergeDJD 优于 DJD。其中，MergeDJD 在 5 个实例中达到或超过了 F 的最佳已知值（BKV）。就 K 指标而言，MergeDJD 优于 DJD

表5：Nesting-SB 实例的比较。粗体数值表示所比较算法中的最佳结果。带下划线的数值表示达到或优于已知最优值（BKV）的结果。

实例 set	ave 件数	BKV		DJD			MergeDJD				
		F	K	F	K	time	平均件数 合并后	F	K	time	合并 time
albano	24	0.605[d]	4.677[a]	0.588	**4.911**	**1.07**	24	**0.589**	4.979	16.72	2.43
dighe1	16	0.457[a]	2.472[a]	0.276	3.873	**0.76**	15	**0.281**	**3.386**	1.30	0.36
dighe2	10	0.397[a]	2.390[a]	**0.367**	3.000	**0.74**	9	0.362	**2.870**	0.75	0.20
fu	12	0.448[d]	7.455[a]	**0.362**	7.649	0.48	12	**0.362**	**7.455**	**0.38**	0.08
han	23	0.529[a]	4.000[a]	0.401	5.000	**0.69**	18	**0.420**	**4.593**	5.21	2.81
jakobs1	25	0.433[d]	8.341[a]	**0.366**	**8.682**	**0.70**	15	0.288	9.682	2.54	1.65
jakobs2	25	0.418[d]	6.568[a]	0.327	7.909	**0.77**	21	**0.339**	**7.568**	2.84	1.13
mao	20	0.472[d]	3.623[a]	0.426	4.000	**0.82**	20	**0.429**	**3.856**	21.34	3.62
poly1a	15	0.456[a]	2.892[b]	0.277	3.979	**0.92**	15	**0.295**	**3.629**	1.67	0.44
poly2a	30	0.458[a]	5.782[a]	0.338	6.909	**0.54**	30	**0.343**	**6.715**	5.28	1.57
poly2b	30	0.419[d]	6.506[b]	0.314	7.909	**0.76**	30	**0.321**	**7.664**	5.25	1.93
poly3a	45	0.455[a]	8.779[a]	0.322	10.909	**0.93**	45	**0.324**	**10.629**	16.20	3.11
poly3b	45	0.456[d]	8.889[a]	0.365	9.951	**0.81**	43	**0.372**	**9.930**	12.78	3.83
poly4a	60	0.453[a]	11.961[a]	0.342	13.909	**0.98**	60	**0.354**	**13.419**	42.09	5.67
poly4b	60	0.431[a]	11.210[b]	0.360	12.937	**1.00**	58	**0.402**	**11.965**	21.98	6.79
poly5a	75	0.455[a]	14.460[a]	0.355	16.909	**1.35**	75	**0.365**	**16.889**	58.10	8.42
poly5b	75	0.479[e]	13.284[b]	0.378	14.993	**1.10**	73	**0.385**	**14.210**	45.26	10.72
shapes0	43	0.330[a]	11.534[a]	0.182	15.714	**0.90**	34	**0.233**	**13.974**	21.21	12.91
shapes1	43	0.330[a]	11.649[a]	0.182	15.714	**0.79**	34	**0.233**	**13.974**	25.80	12.93
shapes2	28	0.309[a]	19.727[a]	0.297	19.727	1.08	28	**0.297**	**19.727**	5.53	3.66
衬衫	99	0.666[e]	13.839[b]	0.409	15.909	**13.63**	91	**0.532**	**13.769**	456.13	26.81
swim	48	0.397[a]	8.771[a]	0.278	10.909	**3.62**	48	**0.322**	**9.982**	818.79	167.69
裤子	64	0.684[d]	4.850[a]	0.501	6.000	**2.07**	64	**0.671**	**5.000**	380.14	11.99
平均值		0.625	8.420	0.348	9.887	**1.59**	37.5	**0.370**	**9.385**	85.53	12.64
min		0.500	2.390	0.182	3.000	0.48	9	**0.233**	**2.870**	**0.38**	0.08
Q1		0.582	4.764	0.306	5.500	**0.76**	19	**0.309**	**4.990**	4.02	1.61
Q2		0.630	8.341	**0.355**	**8.682**	**0.90**	30	0.354	9.682	16.20	3.62
Q3		0.659	11.592	0.373	14.451	**1.08**	53	**0.394**	**13.594**	33.94	9.57
max		0.831	19.727	0.588	19.727	**13.63**	91	**0.671**	**19.727**	818.79	167.69

在 69 个实例中的 63 个实例上，并在 5 个实例中达到或超过了 K 的 BKV。总体而言，这些实例中只有少量零件被合并；尽管如此，MergeDJD 仍持续优于 DJD，表明即使可进行的合并有限，所提出的改进仍然有效。我们还观察到，对于零件数量较多的实例，MergeDJD 的运行时间显著增加。这是因为 ICAD 将已放置零件的点引入为候选初始位置，而候选初始位置的数量可能随着已放置零件数量的增加呈指数级增长。因此，验证每个初始位置可行性所需的时间大幅增加。缓解这一问题的一种可行方法是仅抽样一部分候选初始位置。

随着箱体尺寸增大，我们发现合并的零件越来越多。具体而言，MB 和 LB 集合中的 dighe2 实例零件被合并为一个矩形。如图 9 (a) 所示，组合件代表这些零件最紧凑的布局。然而，与较不紧凑的布局（如图 9 (b) 所示）相比，F 指标保持不变，而较不紧凑的布局却能得到更好的 K 指标。这一反直觉的观察表明，当前的评估指标可能无法充分反映布局紧凑度，因此需要

表 6：Nesting-MB 实例的比较。粗体数值表示所比较算法中的最佳结果。带下划线的数值表示达到或优于已知最优值（BKV）的结果。

实例 set	ave 件数	BKV		DJD			MergeDJD				
		F	K	F	K	time	平均件数 合并后	F	K	time	合并 time
albano	24	0.532[d]	2.497[a]	0.493	3.000	1.11	24	**0.506**	**2.914**	10.44	2.62
dighe1	16	0.374[a]	1.333[b]	0.288	2.000	0.80	15	**0.305**	**1.566**	1.41	0.36
dighe2	10	0.823[a]	0.952[b]	0.263	2.000	0.62	1	**0.823**	**0.952**	0.53	0.01
fu	12	0.478[e]	3.571[b]	**0.442**	3.905	0.74	12	**0.442**	**3.571**	0.41	0.11
han	23	0.420[a]	2.203[b]	0.326	3.000	0.70	15	**0.338**	**2.725**	5.96	2.41
jakobs1	25	0.579[d]	3.250[a]	**0.522**	**4.000**	0.75	12	0.335	4.500	3.04	1.06
jakobs2	25	0.513[a]	3.000[a]	0.384	4.000	0.51	19	**0.423**	**3.333**	3.23	1.10
mao	20	0.420[a]	1.852[a]	0.311	3.000	1.05	20	**0.490**	**1.992**	42.71	3.76
poly1a	15	0.349[d]	1.497[a]	0.296	2.000	0.47	15	**0.301**	**1.872**	1.80	0.40
poly2a	30	0.521[a]	2.961[a]	0.307	4.000	0.58	30	**0.340**	**3.652**	4.08	1.46
poly2b	30	0.414[e]	3.420[a]	0.276	5.000	0.68	30	**0.376**	**3.949**	5.42	1.83
poly3a	45	0.455[a]	4.517[a]	0.306	5.975	0.75	45	**0.423**	**5.000**	10.12	3.33
poly3b	45	0.462[d]	4.515[a]	0.322	6.000	0.54	42	**0.423**	**5.000**	12.64	3.69
poly4a	60	0.519[a]	5.871[a]	0.332	7.933	0.91	60	**0.395**	**6.821**	18.98	5.57
poly4b	60	0.483[d]	5.751[a]	0.368	7.000	0.52	57	**0.383**	**6.205**	21.63	6.76
poly5a	75	0.476[a]	7.443[a]	0.366	8.974	1.52	75	**0.391**	**8.769**	59.88	8.58
poly5b	75	0.488[a]	6.555[a]	0.377	7.974	0.82	72	**0.405**	**7.513**	39.06	10.04
shapes0	43	0.398[a]	5.381[a]	0.229	8.000	1.21	29	**0.297**	**6.524**	41.44	10.11
shapes1	43	0.398[a]	5.381[a]	0.229	8.000	1.04	29	**0.297**	**6.524**	67.51	10.41
shapes2	28	0.351[e]	9.800[g]	**0.351**	**9.800**	0.47	28	**0.351**	9.933	5.889	3.113
衬衫	99	0.666[a]	6.740[a]	0.603	7.000	9.77	91	**0.604**	**6.974**	1010.93	26.92
swim	48	0.402[d]	4.411[a]	0.284	5.981	3.13	48	**0.369**	**4.989**	683.16	171.34
裤子	64	0.590[d]	2.635[a]	0.372	4.000	29.05	64	**0.577**	**3.000**	1259.89	11.73
平均值		0.483	4.160	0.350	5.328	2.51	36.2	**0.417**	**4.708**	143.92	12.47
min		0.349	0.952	0.229	2.000	0.47	1	**0.297**	**0.952**	0.41	0.01
Q1		0.408	2.566	0.292	3.452	0.60	17	**0.339**	**2.957**	3.66	1.28
Q2		0.476	3.571	0.326	5.000	0.75	29	**0.391**	**4.500**	10.44	3.33
Q3		0.520	5.566	0.375	7.466	1.08	52.5	**0.433**	**6.524**	42.07	9.31
max		0.823	9.933	0.603	**9.800**	29.05	91	**0.823**	9.933	1259.89	171.34

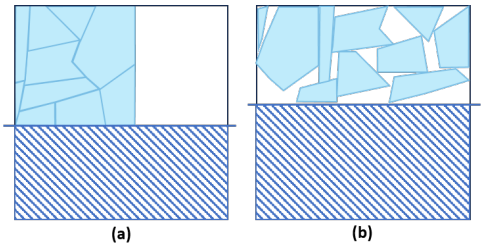

图 9：一个违反直觉的情形。给定一个大箱， F 指标无法区分布局 (a) 和 (b)，而 K 指标则优先选择不那么紧凑的布局 (b)。

更合适的评估标准。

6.4. 新组件的有效性

在本节中，我们分析 MergeDJD 中引入的新组件的有效性。如图 10所示，我们比较 MergeDJD 与三个消融变体，即 仅合并、仅 *ICAD* 以及原始 DJD，比较它们在所有实例组中的 F 指标和 K 指标。

表7：Nesting-LB实例上的比较。粗体数值表示所比较算法中的最佳结果。带下划线的数值表示达到或优于已知最优值（BKV）的结果。

实例 set	ave 件数	BKV		DJD			MergeDJD				
		F	K	F	K	time	平均件数 合并后	F	K	time	合并 time
albano	24	0.416[a]	1.374[a]	0.393	2.000	1.40	24	**0.396**	**1.627**	34.66	2.40
dighe1	16	0.329[a]	0.707[a]	0.329	1.000	0.50	15	**0.329**	**0.962**	0.80	0.33
dighe2	10	0.260[a]	0.587[b]	0.260	1.000	0.38	1	**0.260**	**0.714**	0.36	0.04
fu	12	0.526[a]	1.571[a]	0.485	2.000	0.49	11	**0.494**	**1.875**	0.31	0.04
han	23	0.412[d]	1.217[b]	0.255	2.000	0.50	15	**0.297**	**1.543**	4.67	2.97
jakobs1	25	0.617[d]	1.763[b]	**0.595**	**2.000**	0.50	11	0.342	2.375	3.49	1.18
jakobs2	25	0.464[a]	1.686[a]	0.444	2.000	0.44	19	**0.445**	**2.000**	2.28	1.09
mao	20	0.610[a]	0.961[a]	0.209	2.000	0.66	20	**0.278**	**1.129**	15.77	3.68
poly1a	15	0.368[a]	0.861[a]	0.367	2.000	0.54	15	**0.368**	**1.000**	1.31	0.43
poly2a	30	0.394[e]	1.632[a]	0.224	3.000	0.56	30	**0.371**	**2.000**	3.15	1.49
poly2b	30	0.459[d]	1.870[a]	0.240	3.000	0.49	30	**0.281**	**2.385**	5.27	1.81
poly3a	45	0.416[d]	2.452[a]	0.268	4.000	0.77	45	**0.375**	**2.960**	9.49	3.19
poly3b	45	0.418[d]	2.448[a]	0.370	3.000	0.44	42	**0.381**	**2.819**	11.55	3.80
poly4a	60	0.431[a]	3.274[a]	0.286	5.000	0.94	60	**0.381**	**3.961**	12.09	5.61
poly4b	60	0.415[a]	3.231[a]	0.352	4.000	0.62	57	**0.364**	**3.646**	24.04	6.91
poly5a	75	0.437[a]	4.231[a]	0.303	6.000	0.64	75	**0.383**	**4.963**	67.04	8.57
poly5b	75	0.476[a]	3.742[a]	0.342	5.000	0.69	71	**0.464**	**4.000**	34.32	9.80
shapes0	43	0.462[a]	3.000[a]	0.260	5.000	0.83	21	**0.281**	**3.679**	86.00	8.53
shapes1	43	0.462[a]	3.000[a]	0.260	5.000	0.89	21	**0.281**	**3.679**	110.97	8.37
shapes2	28	0.485[c]	4.700[b]	**0.437**	**4.800**	0.65	24	0.435	4.910	10.19	3.31
衬衫 swim	99	0.652[d]	3.916[a]	0.588	4.000	10.52	91	**0.591**	**4.000**	1849.41	26.95
裤子	48	0.416[d]	2.414[a]	0.238	2.998	1.85	48	**0.335**	**2.829**	1048.29	166.73
	64	0.462[d]	1.405[a]	0.408	2.000	21.09	64	**0.451**	**1.926**	2375.45	12.08
平均值		0.452	2.263	0.344	3.165	2.02	35.2	**0.373**	**2.651**	248.30	12.14
min		0.260	0.587	0.209	1.000	0.38	1	**0.260**	**0.714**	0.31	0.04
Q1		0.416	1.390	0.260	2.000	0.50	17	**0.313**	**1.751**	3.32	1.33
Q2		0.437	1.870	0.329	3.000	0.64	24	**0.371**	**2.385**	11.55	3.31
Q3		0.470	3.116	0.401	4.400	0.86	52.5	**0.416**	**3.679**	50.85	8.45
max		0.652	4.700	**0.595**	6.000	21.09	91	0.591	**4.963**	2375.45	166.73

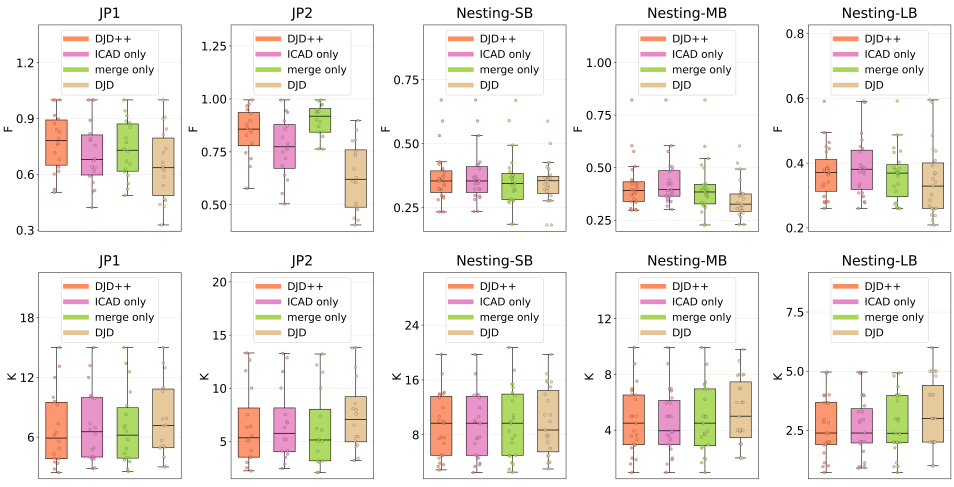

图 10：MergeDJD 改进措施的影响。

结果表明，与 DJD 相比，MergeDJD 中引入的两项改进对所有实例的 F 指标和 K 指标均有积极影响。

此外，我们观察到，合并是 MergeDJD 性能提升的主要来源，而 ICAD 主要增强了稳健性和整体效果。具体而言，在 JP1 上，MergeDJD 在 F 指标方面明显优于所有变体，其中位数更高，方差更小。在 JP2 上，仅合并变体的 F 值与 MergeDJD 相当，有时甚至略高，这表明 ICAD 在该数据集中的优势不那么显著。对于套料实例（SB、MB 和 LB），MergeDJD 的 F 值始终不低于另外三个变体，不过由于可合并的零件数量有限，性能差距相对较小。就 K 指标而言，在大多数实例组中，MergeDJD 的中位数通常低于另外三个变体。Nesting-MB 实例是个例外：仅 ICAD 变体的 K 值略优于 MergeDJD。

7. 结论

本文介绍了 MergeDJD，这是一种用于二维不规则装箱问题的新型构造启发式算法。MergeDJD 在 DJD 算法的基础上，引入了合并策略和改进的布局启发式算法 ICAD。合并策略将相容件组合成组合件。ICAD 改进了非凸零件和合并后的部件的布局。在标准基准实例上的实验结果表明，MergeDJD 在运行时间较短的情况下始终优于 DJD。消融研究表明，合并策略是观察到的性能提升的主要来源，而 ICAD 进一步增强了算法的稳健性和解质量。

未来的工作可以探索几个方向。可以使用基于学习的方法来指导合并或布局决策。例如，这类方法可以预测有前景的合并候选项，或根据几何特征对放置位置进行排序。这有助于减少不必要的尝试，同时保留算法的构造性。MergeDJD 的并行实现还可以进一步缩短其在大规模应用中的运行时间。例如，对于大型实例，可以并行评估多个合并候选项或放置位置。

致谢

本工作得到国家自然科学基金委员会项目62372093以及深圳市科技Pro-的资助

gram项目KJZD20240903095712016。作者感谢Zibo Zhou在早期原型实现方面提供的帮助。

关于在稿件准备过程中使用生成式人工智能和人工智能辅助技术的声明。

在准备这项工作期间，作者使用了 ChatGPT（OpenAI）进行语言润色并提升表达清晰度。使用该工具后，作者对内容进行了审阅，并根据需要进行编辑，同时对已发表文章的内容承担全部责任。

参考文献

[1] Ranga P Abeysooriya、Julia A Bennell 和 Antonio Martinez-Sykora。针对可自由旋转的二维不规则形状装箱问题的推挤启发式。国际生产经济学期刊，195:12– 26，2018。[2] Edmund Burke、Robert Hellier、Graham Kendall 和 Glenn Whitwell。针对二维不规则装箱问题的一种新型左下填充启发式算法。运筹学，54(3):587–601，2006。[3] Sifan Cai、Jie Deng、Loo Hay Lee、Ek Peng Chew 和 Haobin Li。针对有限旋转条件下二维不规则装箱问题的启发式算法。《计算机与运筹研究》，160:106398，2023。[4] Kathryn A Dowsland、Subodh Vaid 和 William B Dowsland。采用左下策略进行多边形放置的算法。《欧洲运筹学杂志》，141(2):371–381，2002。[5] Matheus Francescatto 和 Alvaro Neuenfeldt J´unior。二维装箱、下料和开放维度问题：实际需求综述。《计算机与运筹研究》，第107209页，2025。[6] Francesca Guerriero 和 Francesco Paolo Saccomanno。用于装箱问题的分层超启发式。软计算，27(18): 12997–13010，2023。ISSN 1432-7643。[7] Wei Han、Julia A Bennell、Xiaozhou Zhao 和 Xiang Song。满足一刀切约束的二维不规则形状装箱构造启发式算法。《欧洲运筹学杂志》，230 (3):495–504，2013。

[8]Mhand Hifi 和 Rym M'Hallah。二维布局问题的混合算法：规则形状和不规则形状的情形。*International Transactions inOperationalResearch*，10(3):195–216，2003。[9] Richard E Korf。一种用于最优装箱的新算法。载于 *Aaai/Iaai*，第731–736页，2002。[10] Qiang Liu、Jiawei Zeng、Hao Zhang 和 Lijun Wei。针对有限旋转条件下二维不规则装箱问题的一种启发式算法。载于人工智能理论与应用趋势。人工智能实践：第*33*届工业、工程及其他领域应用智能系统国际会议，*IEA/AIE 2020*，日本北九州，*2020*年*9*月*22—25*日，论文集第*33*卷，第268–279页。Springer，2020。ISBN 303055788X。[11] Andrea Lodi、Silvano Martello 和 Michele Monaci。二维装载问题：综述。《欧洲运筹学杂志》，141(2):241–252，2002。[12] Eunice L´opez-Camacho、Hugo Terashima-Mar´ín、Gabriela Ochoa 和 Santiago Enrique Conant-Pablos。通过主成分分析理解装箱问题的结构。国际生产经济学期刊，145(2):488–499，2013。[13]Eunice L´opez-Camacho、Hugo Terashima-Marin、Peter Ross 和 Gabriela Ochoa。用于求解装箱问题的统一超启发式框架。*Expert Systems with Applications*，41(15):6876–6889，2014。[14] Qiang Luo、Chunrong Pan、Hong Zhong 和 Yunqing Rao。采用精英策略的十进制人工蜂群算法，用于含不规则物品的下料问题。*Swarm and Evolutionary Computation*，97:102026，2025。[15] Eunice L´opez-Camacho、Gabriela Ochoa、Hugo Terashima-Mar´ın 和 Edmund K Burke。针对二维不规则装箱问题的一种有效启发式算法。*Annals of Operations Research*，206:241–264，2013。ISSN 0254-5330。[16]Antonio Martinez-Sykora、Ram´on Alvarez-Vald´es、Julia A Bennell、Rub´en Ruiz 和 Jos´e Manuel Tamarit。针对不规则装箱问题的数学启发式算法

可自由旋转的装箱问题。《欧洲运筹学杂志》，258(2):440–455，2017年。ISSN 0377-2217。[17] Hiroyuki Okano。用于二维自由形状装箱问题的基于扫描线的算法。《日本运筹学会杂志》，45(2):145–161，2002年。ISSN 0453-4514。[18] Franco P Preparata 和 Michael I Shamos。计算几何：导论。Springer Science & Business Media，2012年。[19]Wanda Rulita Sari、Gunawan Gunawan 和 Dimas Angga Fakhri Muzhoffar。船舶制造中套料系统优化的系统综述。载于《*AIP*会议论文集》，第3320卷，第020002页。AIP Publishing LLC，2025年。[20] Chao Tang、Shaowen Yao、Limei Lu、Shigang Zhang 和 Lijun Wei。带引导线的二维不规则排样问题的迭代压缩方法。《*IEEE Access*》，2024年。[21]Hugo Terashima-Mar´ın、CJ Far´ıas Z´arate、Peter Ross 和 Manuel Valenzuela-Rend´on。一种基于遗传算法的方法，用于为二维规则切割下料问题生成通用超启发式算法。载于第*8*届遗传与进化计算年度会议论文集，第591–598页，2006年。[22] Hugo Terashima-Mar´ın、Peter Ross、CJ Far´ıas-Z´arate、Eunice L´opez-Camacho 和 Manuel Valenzuela-Rend´on。用于解决二维规则与不规则装填问题的通用超启发式算法。*Annalsof Operations Research*，179:369–392，2010年。ISSN 0254-5330。[23] Anand Uday、Erik D Goodman 和 Ananda A Debnath。利用特征匹配和并行遗传算法对不规则形状进行套料。载于*Goodman*（编），遗传与进化计算会议：最新研究论文，第429–434页，2001年。[24] Zi Wang、Daofang Chang 和 Xingyu Man。考虑切缝距离及零件自由旋转的二维不规则装箱问题优化。《国际工业工程计算杂志》，13(4):491–506，2022年。[25]Shaowen Yao、Chao Tang、Hao Zhang、Songhuan Wu、Lijun Wei 和 Qiang Liu。针对钢铁行业提出的二维不规则多尺寸装箱问题的迭代倍增二分搜索。《计算机与运筹研究》，162:106476，2024年。

[26] Hao Zhang、Qiang Liu、Lijun Wei、Jiawei Zeng、Jiewu Leng 和 Duxi Yan。针对有限旋转的二维不规则装箱问题的迭代倍增局部搜索。《计算机与运筹研究》，137:105550，2022年。