中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

学　　校	中南大学
参赛队号	22105330309
队员姓名	1.孟维琦
	2.刘笑
	3.郭达辉

中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

题 目 二维三阶段方形件组批排样协同优化

摘 要

在大规模方形件的加工中设计一套完善的组批与排样协同优化的方案，具有重要的现实意义。本文针对该问题提出了一种基于特定需求的订单组批与排样迭代优化切割方法，具体如下：

针对子问题 1：建立了一个典型的**两阶段优化模型**，包括主问题（下料优化问题）和子问题（排样优化问题）。此两阶段优化问题包含两个不同层级的决策变量，分别对应第一阶段的决策和第二阶段的决策，在本文分别对应排样方式使用频次和排样方式中包含的特定 item 数目。最终实现对两阶段决策的联合优化，同时考虑第二阶段排样分布方式对第一阶段的影响。其中，**对于排样优化子问题：**首先根据 3H 排样方式，**引入加权因子、修正算子和 item 组合优化的方法优化排样方式**。加权因子与 item 的面积和板材的利用率具有相关性，加权因子愈大，排样过程优先被考虑；修正因子与组合优化根据多次枚举选择板材利用率最优作为结果，最终生成第一代排样方式，并作为已知变量返回至主问题求解。**对于下料优化主问题：**基于子问题已知的第一代排样方式，根据下料率采用价值更新(ISVC)算法动态更新 item 的价值，并计算当前目标值，设置板材用量基准对主问题目标值进行判断并在子问题中进行持续迭代，通过主问题与子问题的持续迭代直至达到最优结果。通过本方法，子问题 1 四个 item 数据排样优化后板材利用率分别为 93.86%、92.08%、94.07%、94.08%，优化性能突出。

针对子问题 2：首先确定此问题为基于“多品种小批量”定制化订单的组批与排样协同优化问题，针对性提出了基于定制化订单的聚类与排样协同优化的**双层优化模型**。此双层问题区别于两阶段问题的迭代求解，而是对两层工艺分别设置最优算法并求解，相同之处在于第二层问题输入同样基于第一层的输出。其中，**在上层模型：**①对订单采用满足生产工艺约束的凝聚层次聚类算法实现组批方案分类，获得具有同质化特点的多组组批方案，并作为输入进入**回归预测模型**。②考虑时效性要求，需要对排样方案进行“数据预处理”，将子问题 1 作为历史数据采用**随机森林算法**对排样方案最终结果建立回归预测模型并训练。**在下层模型：**利用上层模型训练后的预测模型对聚类的组批方案进行结果预测，并对预测结果进行升序排列，筛选最优的排样方案输入到下层模型(子问题 1 已建立的两阶段优化模型中)求解，确定最优排样方案结果并输出。通过本方法，五个数据集下的批组数为 39、38、32、32、42；板材利用率分别为 93.57%、92.44%、91.7%、90.04%、92%，体现了协同优化方案的可行性。

关键字：组批与排样优化；双层优化模型；价值更新算法；层次聚类算法；预测模型

目录

1. 问题重述

1.1 课题背景

伴随现代社会经济水平的提升及现代制造业的发展，市场需求越来越趋向于产品个性化和多样化定制。下料作为企业生产链中产品及零部件生产的第一道工序，如何提高材料利用率，降低原材料消耗，是企业减少资源和能源浪费，承担环境责任所要解决的关键问题。

排样优化本身属于下料问题，而下料问题又是一个典型的组合优化问题，排样方式本质上是一个决策问题，这属于运筹学的一个重要优化问题之一。针对用户的批量化订单要求，前期的订单处理和决策会直接影响到用料情况（材料利用率和板材使用量），因此，设计与之相关的优化决策模型至关重要。

同时，考虑到方形块(item)订单的个性化定制需求日益增加，相应面临“多种类小批量”的切割需要，但是这一些列的切割过程存在订单量大、排样繁琐等问题，迫切需要前期设计出一种复合此类订单需求的组批方案和排样下料优化方案，以期实现实际生产生活需要，并且保证实际运行效率和经济性要求。

1.2 问题重述

item 的排样是一个典型的二维切割问题。目前，针对 item 排样优化问题，主要的解决方案有线性规划模型、整数规划模型和启发式搜索算法三类[1-3]。根据切割工艺的不同，方形件的排样优化问题主要分为两类，一类是齐头切，另一类是非齐头切。考虑现阶段的工艺水平，目前采用较多的方法主要为齐头切。齐头切又可以细分为精确方式和非精确方式，主要有三种不同的类型：三阶段非精确(3NE)排样方式、三阶段匀质排样方式(3E)、三阶段同质排样方式(3H)。其中 3E 和 3H 排样方式可在三个阶段内切割出准确尺寸的方形件，因此都属于精确排样方式[4]。

由于上述个性化定制生产模式中的订单组批与排样优化至关重要，在假定只考虑齐头切的切割方式，并且规定切割阶段数为 3 阶且保证精确排样的基础上，我们需要解决如下两个问题：

问题 1：针对数据集 A，建立混合整数规划模型，对其进行排样优化，在满足生产订单需求和相关约束条件下，尽可能减少板材用量，同时能够输出排版方案。

问题 2：针对数据集 B，建立混合整数规划模型，先对其进行组批，再对每个批次进行独立排样优化，在满足生产订单需求和相关约束条件（问题 1 约束基础）下，尽可能减少板材用量，同时能够输出组批和排版方案。

2. 模型假设

为进一步实现模型的精简化，本文假定：

1. 为适合于自动化加工场景，每一阶段的切割可以采用多个刀具同时进行；
2. 为防止切割不当及后继加工预放的加工余量，默认为零；
3. 所有订单的交货期均相同；
4. 只考虑齐头切的切割方式；
5. 切割阶段数为 3 阶段，同一个阶段切割方向相同，相邻阶段切割方向相互垂直；
6. 排样方式为精确排样；
7. 假定板材原片仅有一种规格且数量充足；
8. 排样方案不考虑锯缝宽度（即切割的缝隙宽度）影响；
9. 子问题 2 中同一份订单仅出现在相同材质产品项，否则无法同时满足要求 1）和要求 2）。

3. 符号说明

符号	含义
G	当前代数
$G_{\max}$	最大代数
P	排样方案的3H排样方式集合
$R=[r_1,\cdots,r_m]$	item 的当前剩余需求向量，r_i 为 i 型 item 的剩余需求量
$D=[d_1,\cdots,d_m]$	item 的初始需求向量，d_i 为 i 型 item 的原始需求量
$V=[v_1,\cdots,v_m]$	item 的价值向量，v_i 为 i 型 item 的价值
$PNum$	当前最好排样方案所用的板材数量
a_{ij}	排样方式 p_j 中包含的 i 型 item 数量
f_j	排样方式 p_j 的使用频率
λ_1,λ_2	初始价值系数，其中 $\lambda_1 >> \lambda_2 > 1$
Q	长度基准
$e(x,i)$	同质条带 $x\times w_i$ 中最多可包含的 i 型 item 数量，$i=1,\cdots,m$
$v(x,i)$	同质条带 $x\times w_i$ 的最大价值，$i=1,\cdots,m$
e_i	同质条带 $x\times w_i$ 中包含的 i 型 item 有效数量，$i=1,\cdots,m$
v_i	同质条带 $x\times w_i$ 的有效价值，$i=1,\cdots,m$
$F(x,y)$	子段 $x\times y$ 的价值

$n(x,y,i)$	子段 $x\times y$ 中包含的 i 型 item 数量，$i=1,\cdots,m$
h_{ki}	段 $q_k\times W$ 中的 i 型 item 有效数量，$i=1,\cdots,m$
$f(x)$	子板 $x\times W$ 的价值
$\eta(x,i)$	子板 $x\times W$ 中包含的 i 型 item 数量，$i=1,\cdots,m$
m	测试样本数
n	测试样本按照订单分出的总类
y_i	样本实际数值
$\hat{y}_i$	预测模型预测结果
MAPE	平均绝对误差百分比
$RMSE$	误差平方根均值
p_i	方形快订单中某一种材质
q_j	方形快订单中另一种材质
C_p	订单 p_i 组成的簇，$i=1,\cdots,m$
C_q	订单 q_j 组成的簇，$j=1,\cdots,n$
$dist(C_p,C_q)$	簇类 C_p 和 C_q 之间的距离(相似度)
$\|C_p\|,\|C_q\|$	簇类 C_p 和 C_q 的订单个数
$H(n)$	簇间距离的最小方差矩阵
$C_x(n)$	订单凝聚层次聚类算法中的簇
P_{C_p}	C_p 簇类的聚类中心
P_{C_q}	C_q 簇类的聚类中心
n	逐次聚类合并次数
$D(n)$	各类订单簇之间的类平均距离矩阵
$G_d(n)$	初始状态下根据订单分类，d 为初始状态订单种类大小
T	类间距离阈值
D	含有 m 条排样数据样本集 $D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$
$E_t(x)$	第 t 个学习器

4. 问题一的分析及模型的建立与求解

4.1 问题分析

4.1.1 问题描述

减少板材原材料消耗是下料优化问题中最重要的目标。并且板材必须切割成块（方形件）以满足客户订单。由于技术限制，通常只允许平行于板材侧面的切割（正交切割）和从一个边界到另一个边界的切割（断头切割）。此外，考虑到三阶段排样方式可以较好地平衡材料利用率和切割复杂度，板材切割的阶段数（具有相同方向的一组切割——水平或垂直）设置为 3。本文切割问题三阶段切割方式为：在板材原材料被切割成条状（在第一阶段）后，每个条状然后被切割成垛状（第二阶段），最后每个垛中的方形件被第三组（水平）切割分开（第三阶段），得到最终的方形件。切割过程采用正交剪切，每一刀都与板材的长度方向或者是宽度方向平行。具体切割过程如图 4-1 所示。

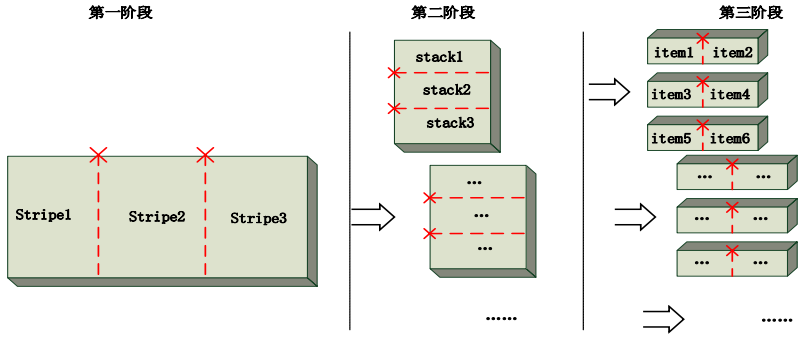


图 4-1 三阶段排样切割过程

三阶段排样方式主要有三种不同的类型：三阶段非精确(3NE)排样方式、三阶段匀质排样方式(3E)、三阶段同质排样方式(3H)。其中 3E 和 3H 排样方式可三个阶段内切割出准确尺寸的方形件，因此都属于精确排样方式。本文采用 3H 的精确排样方式，详细求解方式在后文介绍。

4.1.2 问题分析

子问题 1 本质上为排样优化问题，对应目标函数体现为板材用量最小化，满足的约束分别为：①在相同栈（stack）里的产品项（item）的宽度（或长度）应该相同；②最终切割生成的产品项是完整的，非拼接而成。两个约束本质上表现为三阶段切割的基本约束条件[5]。

二维三阶切割下料问题既要考虑到原材料板材成本，又需要考虑到切割工艺复杂度。对于顾客订单定制的 item 既要满足板材利用率最大（切割面积最大化），又需要考虑材料利用率最大化。对应转换可以建模为**两阶段优化问题**：一个**主问题**(Master problem)和一个**子问题**(Sub-problem)。其中，主问题为实现板材使用量最少为目标的二维三阶段切割**下料优化问题**，子问题为带约束的**三阶段排样优化问题**。

其中，第二阶段的排样优化问题仅考虑精确排样方式，对应 3E 和 3H 两种排样方案，但是考虑到“个性化定制”要求和求解建模的简单性、便捷性，本文最终采用三阶段精确排样方式中较为简单的 3H 排样方式[6-7]，以期实现板材利用率和加工复杂度之间的权衡。

4.2 模型建立

4.2.1 主问题：下料优化

本文研究的二维三阶段切割下料问题可描述为：原来板材尺寸为$L\times W$，库存数量足够多；待排样的 item 有m种，对应长、宽及需求量分别为(l_i,w_i,d_i)，其中i为 item 型号，$i\in I, I=\{1,\cdots,m\}$。需要确定一个合理的三阶段排样方案，使用数量最少的板材，排完全部的 item 订单。

以板材用量最少为目标的二维三阶段切割下料问题，可用公式(4-1)~(4-2)表示：

$$\min z=\sum_{j=1}^{K} f_j \tag{4-1}$$

$$s.t.\begin{cases}\sum_{j=1}^{K} a_{ij} f_j = d_i, i=1,2,\cdots,m\\ f_j\in N, j=1,2,\cdots,K\end{cases} \tag{4-2}$$

基于此问题的求解需要首先确定排样方案包含的所有的K种排样方式的集合：$P=[p_1,p_2,\cdots,p_K]$，其中，每种排样方式中所包含的i型 item 数量a_{ij}、排列位置和顺序，对应排样方式的使用频数$f_j, i=1,2,\cdots,m,\ j=1,2,\cdots,K$。考虑到最终为求解三阶段排样方案，且由三阶段同质(3H)排样方式(表现为带约束的排样优化)组成。最终采用迭代价值更新算法求解以上问题。子问题的3H排样方式生成算法作为3H排样方案的子算法，需要不断地重复调用并更新，产生多个不同的3H排样方式组合，直至排完所有 item 的订单，最终确定完整的优化方案。需要注意的是，最终排样方案结果好坏不仅与主问题本自身的求解算法有关，而且也需要对子问题的排样方式生成算法作为数据支撑，这正体现了两阶段优化的本质。

4.2.2 子问题：排样优化

两阶段排样优化模型的子问题是带约束的三阶段排样问题：寻找 item 在板材上的 3H 排样方式，最大化板材中包含的 item 总价值，而所包含 item 的数量不能超过其剩余需求量。建立排样优化数学模型如式(4-3)~(4-4)：

$$z_p=m\sum_{i=1}^{m} a_i v_i, P\in 3H\text{排样方式} \tag{4-3}$$

$$s.t. a_i\le r_i, a_i\in N, i=1,\cdots,m \tag{4-4}$$

其中，z_p对应3H排样方式的总价值，a_i对应排样方式中所包含的i型 item 的数量，v_i对应i型 item 的价值，r_i为i型 item 的当前剩余需求量，$i=1,\cdots,m$。

3H 排样方式由同向段组成，段中只包含同向条带，限制同一条带上只排入同种类型同方向的 item，有利于简化切割操作。第一阶段一次性把板材切割成多个段，第二阶段将段切割成多个条带，第三阶段将条带切割出 item。由于采用同质条带，同一种类 item 之间的间距相同，简化了第三阶段的切割操作。同质条带中相同 item 相邻排放，也增加了订单加工的便利性，减少排样方式中排入的 item 种类和切割过程中所需要的 item 堆放堆数，便于加工车间提取 item 作进一步加工，确保整个生产过程的效率。

并且考虑到 3H 排样方式有两种不同方向：X 向和Y向。X 向 3H 排样方式第一阶段

沿垂直方向切割，段沿水平方向排列，段上的条带沿垂直方向排列，条带上的 item 沿水平方向排列。*Y* 向方式则与之相反。

图 4-2 对应 *X* 向 3H 排样方式下的切割过程进行了详细模型介绍，对应箭头表示刀割口、数字代表相应的三个阶段号。第一阶段：沿垂直切割线将板材切割成左右两个段，第二阶段：沿两条水平切割线将左段切割成三根水平条带，第三阶段：沿垂直切割线将左段最上条带前切成两个尺寸准确的 item。对于 Y 向 3H 排样方式的切割过程同理可得，不再赘述。

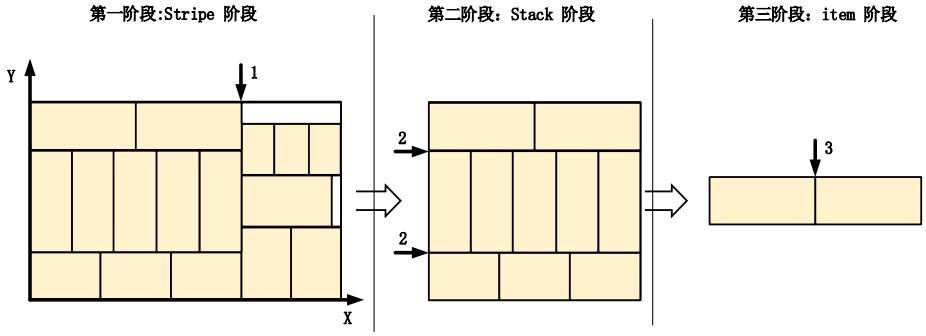


图 4-2 X 向 3H 排样方式

子问题排样方案的求解本身是一个决策问题，并作为整个两阶段模型的第二阶段求解，第二阶段问题的决策情况与第一阶段的求解相互关联，在两阶段共同求解下构成最佳决策。

4.2.3 两阶段优化模型：两阶段排样优化

关于此二维三阶段切割下料问题的求解，目前已有的枚举法对于本文海量数据并不适用，计算复杂度和时间成本较高。因此，本文重新设计并提出了新的求解算法，采用递推技术求解第二阶段排样子问题的近似解,通过动态调整价值算实现全局最优 3H 排样方案，继而进一步求解第一阶段下料主问题。具体流程如图 4-3 所示。

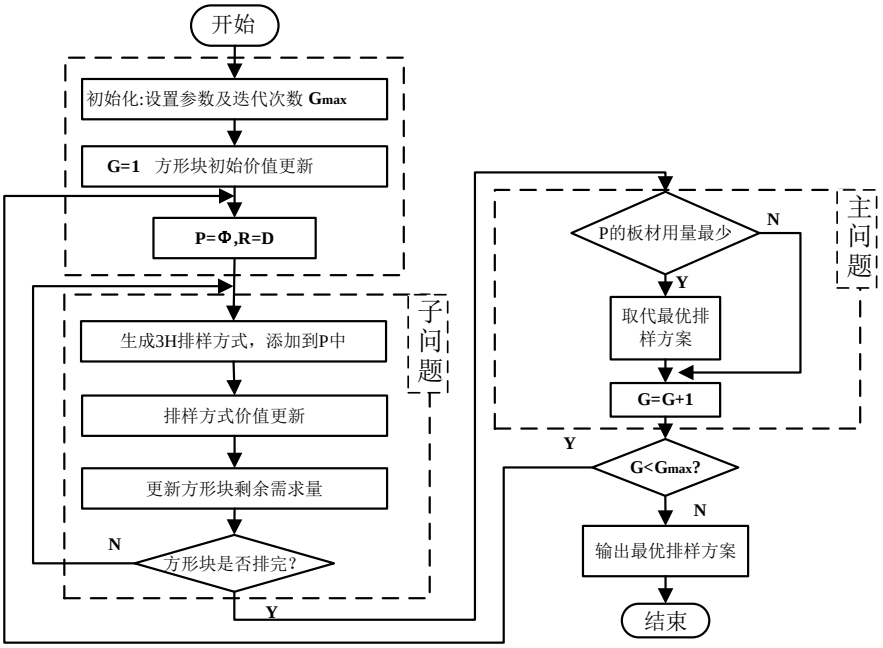


图 4-3 迭代求解价值更新算法流程图

4.3 模型求解及优化结果分析

4.3.1 数据组合处理

针对子问题 1，题目提供了 A1~A4 共四组数据，每组数据都包含了序号、材质、数量、长度、宽度和订单号这些信息，值得一提的是，每组数据中长宽相同的 item 只有少数几块，对于大部分 item 来说，长宽并不一致，这也就导致整个数据集变得复杂。为降排样复杂度，提高板子利用率，按照长宽信息对相关 item 进行拼接，减少 item 的种类，同时又保证融合后数据集，包含所有 item 的长、宽、订单号和材料的信息。

（1）拼接规则

由上述分析，对于 item 的拼接，可以采用长宽相等规则，共包含四种情况：①item1 的长等于 item2 的长，如图 4-4(a)所示；②item1 的长等于 item2 的宽，如图 4-4(b)所示；③item1 的宽等于 item2 的长，如图 4-4(c)所示；④item2 的宽等于 item2 的宽，如图 4-4(d)所示。

特别地，若 item1 和 item2 长宽相同，可采用长与长拼接，也可采用宽与宽拼接。

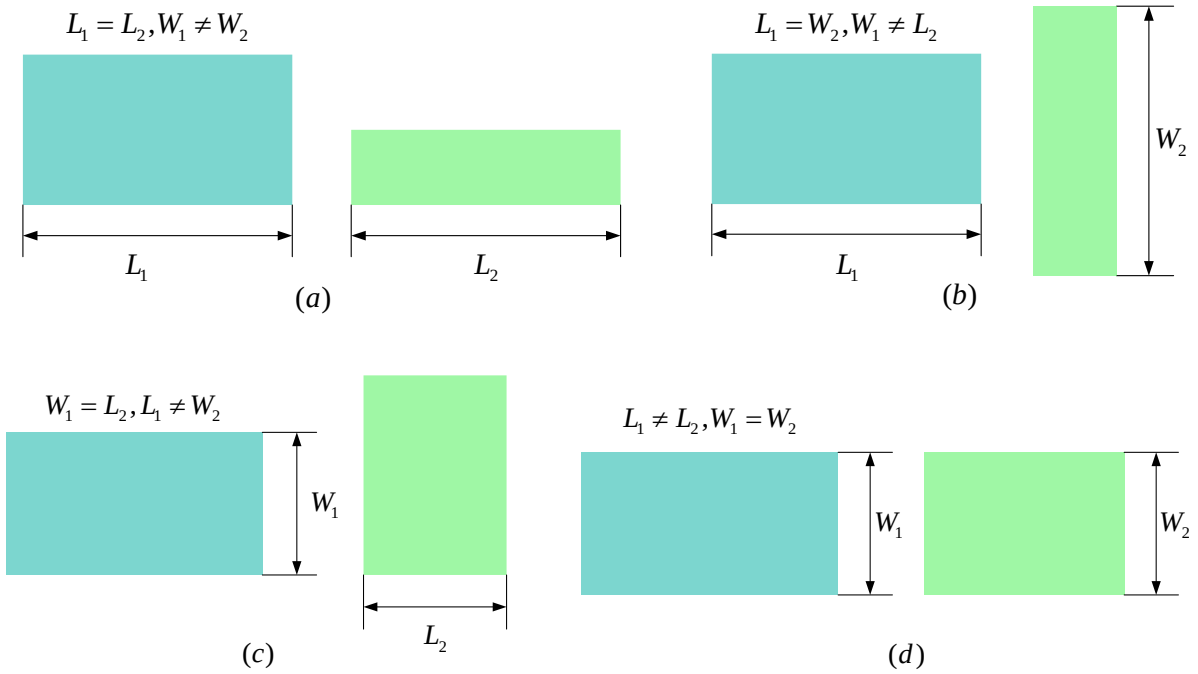


图 4-4 item 拼接规则

在拼接时，要保证拼接后的尺寸不能超过钢板尺寸，否则无法进行拼接。因为长宽属于相对而言，item 可以旋转，所以拼接尺寸约束可以表示为：

$$\begin{aligned} \max(L,W) &< L_0 \\ \min(L,W) &< W_0 \end{aligned} \tag{4-5}$$

式中，L 和 W 为拼接后的新 item 的尺寸。

（2）排列组合拼接

1）新 item 数据集生成

根据（1）中的拼接规则，可以得到很多类满足条件的 item 组合，为了保证组合的完整性，采用排列组合的方法生成所有拼接组合。

$$L_{new}^{i}=C_{l_i}^{2}\quad i=1,2,3,4$$
$$s.t.\begin{cases}\max(L,W)<L_0\\ \min(L,W)<W_0\end{cases} \tag{4-6}$$

式中，L_{new}^{i} 为在满足拼接规则约束中，所给第 i 组数据拼接后新数据集长度。

2）item 排列数据集筛选

在 1）中得到的新数据集中包含了重复 item 数据，需要进一步筛选。筛选规则为：每个 item 出现的次数不能超过它的需求量。根据上述规则，对新 item 数据集采用不放回抽取。

$$AB=N_E \tag{4-7}$$

A 为排样板材中每个 item 对应 id 的个数，B 为板材排样结果，N_E 为 item 需求量，用 item 的 id 进行区分。生成的排列组合 item 数据集，将其与剩余未拼接 item 数据集组合在一起，生成新 item 数据集，用于板材样式优化与后续求解计算。

4.3.2 排样优化求解

本文采用的迭代顺序价值修正(ISVC)算法属于启发式算法，以顺序启发式价值修正（SVC）作为基本框架。SVC 的基本思想是生成一种排样方式后，根据材料利用率对 item 价值进行修正。按新价值，对剩余 item 继续生成排样方式，直到全部 item 需求都满足为止。这个过程中顺序生成的排样方式直接组成排样方案。

ISVC 算法迭代生成一个 3H 排样方案涉及到 item 初始价值调整算子、3H 排样方式生成算法和排样方式的 item 价值修正算子的调用。首先介绍 ISVC 算法的程序流程图，然后简要介绍算法的主体框架，最后再详细介绍所调用的其他相关算子和子算法。ISVC 算法的伪代码如表 4-1 所示。

表 4-1 迭代求解价值更新算法伪代码

算法 1: 迭代求解价值更新算法 ISVCforCuttingPlan()		
	输入: 板材尺寸、item 种类、尺寸和订单需求等	
	输出: 3H 排样方案	
1	令 ***G*=1，P_{Num}=∞，**使用 item 初始价值调整算子初始化 item 价值。	
2	***P*** ←生成排样方案	
	1	**初始化参数: *j*=1，*P*=*Φ*，*R*=*D***
	2	调用 3H 排样方式生成算法 3*HPattern-Procedure()*，生成 3H 排样方式 p_i，并添加到当前排样方案中，其使用频率 f_i
	3	使用排样方式 item 价值修正算子调整 item 价值
	4	更新 item 的剩余需求量 $r_i=r_i-f_i a_{ij},(i=1,...m)$；
	5	若 $\exists r_i>0(i=1,...m)$ 则令 ***j*=*j*+1**，并转 **Step2.2**，否则转 **Step3**；
3	若板材用量小于 P_{Num}，则以当前排样方案作为最好排样方案，并更新 P_{Num}	
4	令 ***G*=*G*+1**，若 ***G*>G_{max}**，则算法终止，输出最好排样方案；否则转 **Step2**	

注：需要对 item 价值算子、3H 排样方式生成算法和 item 价值更新算子进行独立设计。

排样优化问题通过建立的两阶段优化模型进行求解，先进行排样求解，再优化利用率。每个板材的排样生成，由同一个板材的条带拼接而成。具体思路如下所述。

（1）分组

按照 item 的长宽进行型号分组，分为特大型号 s_g、大型号 b_g、小型号 s_m 三种。s_g 和 b_g 每种分类规则为

$$s_g = (L_i > 0.5L_0 \cap W_i > 0.5W_0) \cup (L_i > \alpha L_0) \cup (L_i > \beta W_0)$$
$$b_g = (L_i > \chi L_0) \cup (L_i > \delta W_0) \tag{4-8}$$

式中，α、β、χ 和 δ 为分组修正算子，因为 item 型号分类标准并不是唯一确定的，所以用于调节特大型号和大型号的分组，剩余种类分类为 s_m。在利用率优化中，修正算子动态调整。

（2）3H 排样方式生成算法：

由前述分析可知，3H 排样方式下对应两种方向：X 方向与 Y 方向。两种方向的排样方式类似（旋转后等效），求解方法也基本一致。本文选择其中的 X 向排样方式进行论述。

这里，针对 3H 排样构造方法与切割方案相反：方形件拼接成条带（第一阶段），有效条带组装成段（第二阶段），将段与段最终组合为板材原材料（第三阶段）。最终采用复杂度比较低的递推过程快速生成 3H 排样方式。子段向上拼接条带，最终递推可以得到段；子板向右拼接段，可以得到新的子板，最终递推得到板材，流程如图 4-5 所示。

对应详细流程为：①长度基准集合，②生成条带，③生成长段，④排样优化。

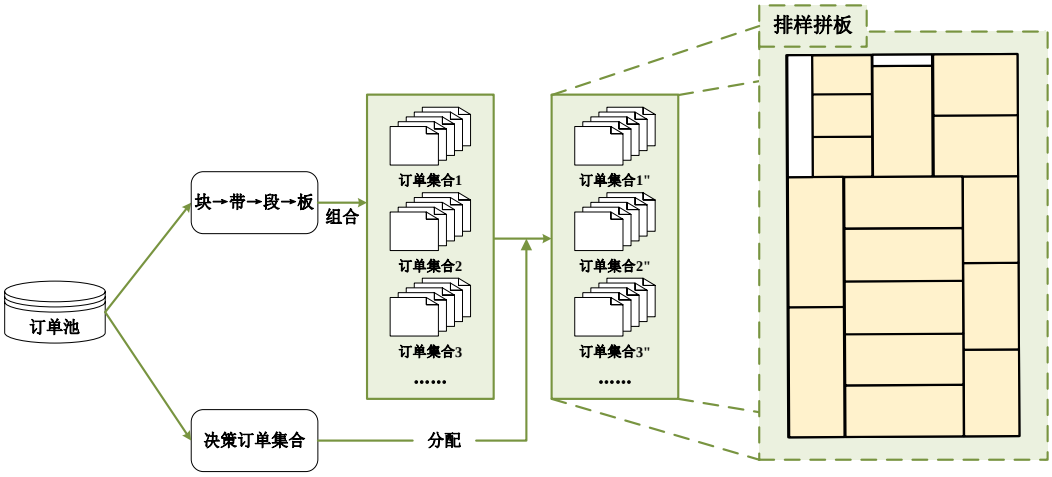


图 4-5 排样处理流程方式

1）长度基准集合：

采用本文的 3H 排样方式下，实际切割过程的切割位置的长度基准是 item 长度的整数倍，对应式(4-9)：

$$Q = \{x_{ik} \mid x_{ik} = kl_i, 0 \le k \le [r_i, [L / l_i]], k \in N, i = 1, ...m\} \cup \{L\} \tag{4-9}$$

排除重复元素，按递增顺序排列各元素，即 $Q = \{q_1, ..., q_m\}, q_1 = 0, ..., q_m = L$。其中，$m$ 表示元素数目，基于以上定义，在优化过程中仅考虑规范定义的长度条和段，继而实现节省计算时间的目的。

2）条带生成：

对于 i 型 item 组成的同质条带 $x \times w_i, x \in Q, i \in I_x, I_x = \{i \mid l_i \le x\}$，相应包含的 item 数量最多为 $e(x,i) = \min(x / l_i, r_i)$，相应情况下最大价值是 $v(x,i) = e(x,i)v_i$，对应 item 数量为 e_i，$e_i = 1, \cdots, e(x,i)$，相应条带价值是 $e_i v_i$。

条带组合包括 X 向和 Y 向两类，基于以上方式确定最大利用率相应方案实现条带生成。

$$V^{*}=\max(V^{X},V^{Y}) \tag{4-10}$$

同理，Y 向和 X 向的处理方法相似。

3）长段生成：

长段 $x\times W,x\in Q$ 由条带沿垂直方向拼接形成，其目标是在满足宽度约束和订单需求约束的基础上实现其所包含的条带最终价值最大化。相应约束为：长段中条带宽度之和小于等于原材料的宽度，长段中 item 数量小于等于当下的剩余需求订单数目。

长段中包含的条带可以由子段递推而得。如图 4-6 所示，子段 $x\times y$ 可由子段 $x\times(y-w_i)$ 和同质条带 $x\times w_i$ 向上拼接组成。设子段的价值和所包含的 item 数量分别为 $F(x,y),n(x,y,i),x\in Q,i=1,\ldots,m$。采用递推式(4-11)可从 $y=0$ 开始递推求子段的价值 $F(x,y)$，继而确定子段中的条带组合。

初始时，$F(x,0)=0,n(x,0,i)=0,x\in Q,i=1,\ldots,m$。由于有 item 需求的约束，子段中包含的 item 数量小于等于其剩余需求量。子段 $x\times(y-w_i)$ 中已含有的 i 型 item 数量为 $n(x,y-w_i,i)$，因此条带 $x\times w_i$ 中 i 型 item 的有效数量只有 $e_i=\min\{x/l_i,r_i-n(x,y-w_i,i)\},i\in I_y$，其中 $I_y=\{i|\ i\in I_x,w_i\le y\}$ 为可排入的 item 集合。

$$F(x,y)=m\left\{F(x,y-1),\max_{i\in I_y}\left[F(x,y-w_i)+e_iv_i\right]\right\},0<y\le W \tag{4-11}$$

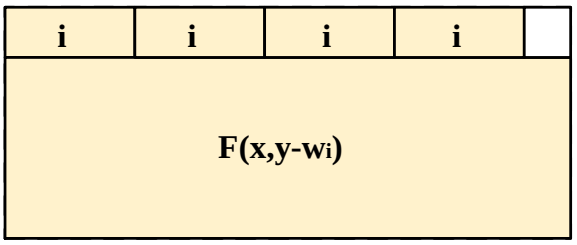


图 4-6 子段 x*y 的递推求解

子段 $x\times y$ 的初始解和子段 $x\times(y-1)$ 的解一致。当拼接的子段价值高于初始价值时，相应优化解更新为拼接结果。依次递推得到子段的价值后，可按如下规则确定子段中包含的各类型 item 数量，具体流程如表 4-2。

表 4-2 排样生成流程I

流程 1:	
1	***if*** $(F(y)=F(y-1))$
2	$\quad n(x,y,i)=n(x,y-1,i),i=1,\ldots,m$
3	***Else***
4	***if*** $(F(y)=F(y-w_k)+e_kv_k)$
5	$\quad n(x,y,k)=n(x,y-w_k,k)+e_k,n(x,y,i)=n(x,y-w_k,i),i\in I,i\ne k$
6	***end***

迭代之后的 $F(x,W)$ 即为段 $x\times W$ 的最终价值，其包含 i 型 item 数量为 $n(x,W,i)$，$i=1,\ldots,m$。同时，将段的 item 数量信息转存到 N_{ki} 中，$k=1,\ldots,M$，$i=1,\ldots,m$。

通过递推式(4-11)求得全部 M 个规范长度段的时间复杂度为 $O(mMW)$。

4）排样方式：

排样方式的确定表现为对第二阶段（子问题）的求解，根据 item 价值和剩余需求量，生成排样方式，最终目标函数为不同排样方式下所有 item 的总价值最大化。

3H 排样方式由规范长度段组成。将段沿水平方向排列组合，可得到板材 $L\times W$ 的排样方式。可采用子板拼接段，递推出板材中包含的段组合。设 $f(x)$ 为子板 $x\times W$ 的价值，$\eta(x,i)$ 为其包含的 i 型 item 的数量，$x=0,\cdots,L$，$i\in I$。子板 $x\times W$ 可由子板 $(x-q_k)\times W$ 和段 $q_k\times W$ 向右拼接而得（如图 4-6 所示），其中 $k\in K_x=\{k|\ q_k\le x\}$ 为可排入子板中的段序号。

子板 $(x-q_k)\times W$ 中已包含的 i 型 item 数量为 $\eta(x-q_k,i)$，由于 item 的需求约束，拼接上来的段 $q_k\times W$ 中的 i 型 item 的有效数量只能为 $h_{ki}=\min\{N_{ki},r_i-\eta(x-q_k,i)\},i\in I$，$k\in K_x$。递推式(4-12)可从 $x=0$ 开始推导 $f(x)$，确定子板中各段的组合。初始时 $f(0)=0,\eta(0,i)=0,i=1,\cdots,m$

$$f(x)=m\left\{f(x-1),\max_{k\in K_x}\left[f(x-q_k)+\sum_{i=1}^{m}h_{ki}v_i\right]\right\} \tag{4-12}$$

子板 $x\times W$ 的初始解和子板 $(x-1)\times W$ 的解相同，如果拼接子板 $(x-q_k)\times W$ 和段 $q_k\times W$ 得到的价值比初始价值高，则采用拼接结果，对应表 4-3 所示。

表 4-3 价值更新流程II

流程 2:	
1	***if*** $(f(x)=f(x-1))$
2	***do*** $\eta(x,i)=\eta(x-1,i),i=1,\ldots,m$
3	***else***
4	***if*** $(f(x)=f(x-q_k)+\sum_{i=1}^{m}h_{ki}v_i)$
5	***do*** $\eta(x,i)=\eta(x-q_k,i)+h_{ki},i=1,\ldots,m$
6	***end***

综上所述，按照 3H 方法，在 X 方向 item 详细的拼接流程如图 4-7 所示。

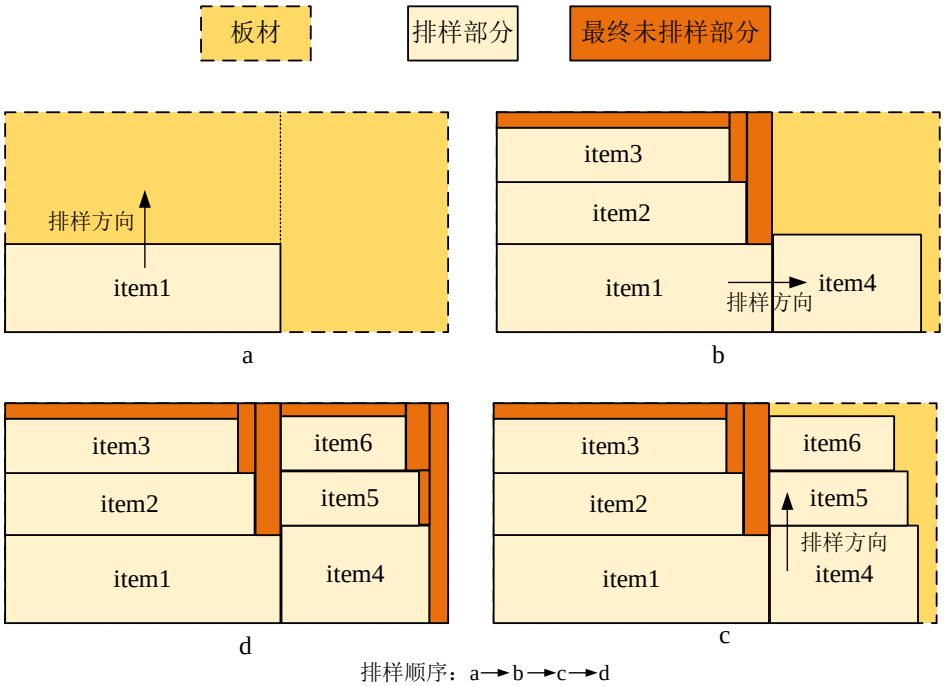


图 4-7 排样过程示意图

通过递推式(4-12)求得时间复杂度为$O\left(mML^2\right)$，生成一种排样方式的总时间复杂度为$O\left(mM\left(L^2+W\right)\right)$。

基于上述排样过程得到最终的排样结果如图 4-8 所示，包含 item 部分和浪费部分。

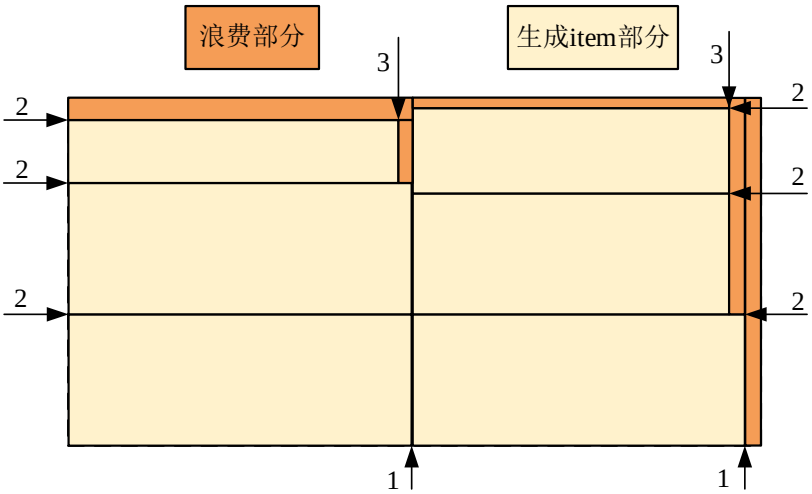


图 4-8 三阶段排样过程示意图

（3）排样方式 item 价值更新算子：

排样一般根据经验先排样大 item，尤其是超过板材一半尺寸的特大 item，因为这些大 item 或者是特大 item 很难与其他 item 组合成好的排样方式。如果只按材料利用率最大选择 item，这些 item 很可能由于一直都无法形成材料利用率大的排样方式而被保留到最后，影响到后期排样方式的材料利用率。item 初始价值调整的目的是对特大 item 或某一尺寸很大的 item 赋予更高的价值，使这些特殊 item 在 item 种类和数量都还较多时，就已优先排入，此时可形成的 item 组合较多，有利于形成较好的排样方式。

1）基于 item 大小的价值算子

方形快默认初始价值体现在面积，对应$v_i=l_iw_i$，对于面积较大的 item，对其进行价值系数赋值，用以后期进行价值调整，其中，设定$\lambda_1>>\lambda_2>1$，满足较大面积的 item 价值增加非常明显，对应权重最大化。具体表现为：

①初始价值：$v_i=l_iw_i$；

②若$(l_1\geq 0.5L)\wedge(w_1\geq 0.5W)$，则$\lambda_1 v_i\rightarrow v_i$；否则，若$(l_1\geq 0.5L)\vee(w_1\geq 0.5W)$，则$\lambda_2 v_i\rightarrow v_i$；

2）基于材料利用率的价值算子

以 item 总价值最大为目标生成排样方式时，相同面积的 item，价值不同排样的优先级也不尽一致，体现为价值愈大愈容易被选中排样。价值修正公式的目是根据当前的排样效果，人为地增加面积比较大的或者是“待舍弃”的 item 的价值，使这些组合困难的 item 比其它 item 具有更高的优先权。这里的“待舍弃”是指材料利用率低的排样方式中的 item。公式(4-13)给出了 item 的价值修正公式。

$$v_i\leftarrow g_1v_i+g_2\left(l_iw_i\right)^{\beta}/u_j \tag{4-13}$$

式中i型 item 的价值由旧价值和修正价值两部分构成，g_1和g_2分别是更新前价值的权值和更新后价值的权值，其中$g_2=\alpha a_{ij}/d_i, g_1=1-g_2, \alpha$和$\beta$为确定权值的参数。$u_j$为排样方式$p_j$的材料利用率，即$u_j=\left(\sum_{i=1}^{m}l_iw_ia_{ij}\right)/\left(LW\right)$。参考文献[7]，设定参数可参考的取值范围：$\alpha\in[0.6,0.9]$，$\beta$略大于 1。

item 的修正价值与其面积和材料利用率存在关联：

①$\left(l_iw_i\right)^{\beta}$且$\beta>1$表明修正价值与 item 的面积成正比关系。大的 item 修正后的价值增

量比普通 item 更大，排样方式中被选中的概率同比更大。

②u_j的作用体现了修正价值和材料利用率之间属于反比关系。其中，需要注意的是，排样中未用到的 item 相应地价值不变，对应$a_{ij}=0$。材料利用率过低说暗示组合困难，需重新排样。

item 的价值修正会影响之后排样方式的优先级选择。多次的价值修正后，相应排样方式下的总价值会愈来愈大。相比之下，材料利用率高的排样方式所包含的小 item 价值则基本不变。这种结果能够体现难组合的大 item 相比比较容易组合的小 item 存在更高排样顺序。

修正后的 item 价值直接进入下一代排样方案的迭代中，从而下一轮的迭代可以直接获得已有的 item 排样优先等级，并以不同的 item 信息生成新的排样方式及排样方案。

基于利用率的排样生成算法如图 4-9 所示。

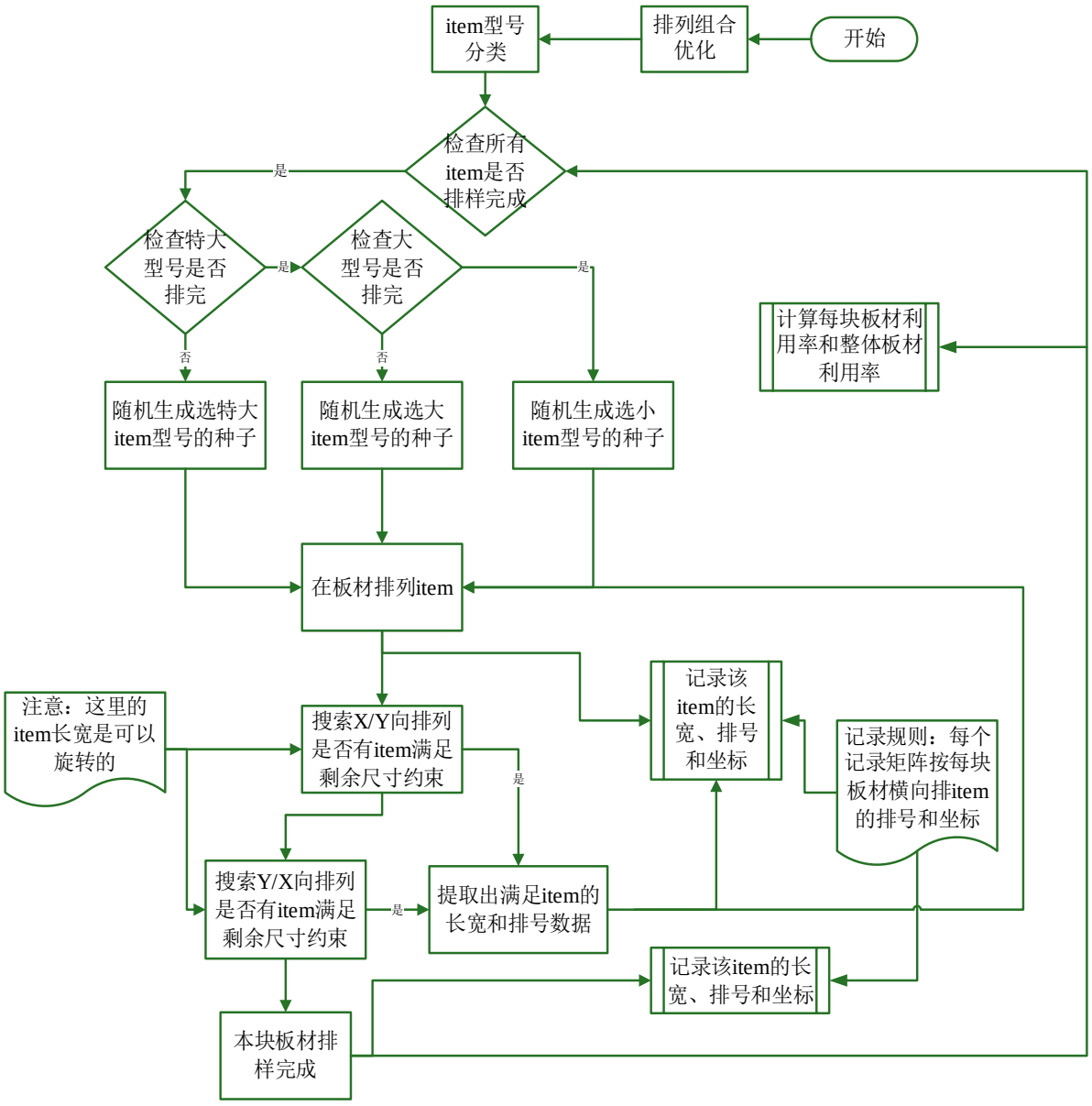


图 4-9 数据集生成流程图

关于 3H 方式生成的伪代码如表 4-4 所示。

表 4-4 3H 方式生成算法伪代码

算法 2: 3H 生成算法 3HPattern-Procedure()	
	输入: 板材尺寸、新 item 种类、尺寸，订单需求和价值
	输出: 3H 排样方案
1	***While***($D \neq \varnothing$)
2	item = D(1,:)
3	检验拼接的两个 item 是否达到寻求量
4	是：删除所有包含 item1 和 item2 的数据行
5	否：只删除 D 第一行

4.3.3 排样优化结果

（1）排样结果

因为修正算子、排列组合和样本排列的随机性都会影响到最终的排样结果和利用率，所以对 item 进行 5000 次实验，以消除上述因素对结果的影响。以板材利用率为目标对 5000 次结果进行筛选，选取利用率最高的作为排样优化的结果。5000 次实验必然会出现相同利用率的结果，再结合切割刀数等因素综合考虑，选择出最终排样结果。

图 4-10 显示了板材利用率和板材使用数量，可以看到，利用率均高达 90%以上。

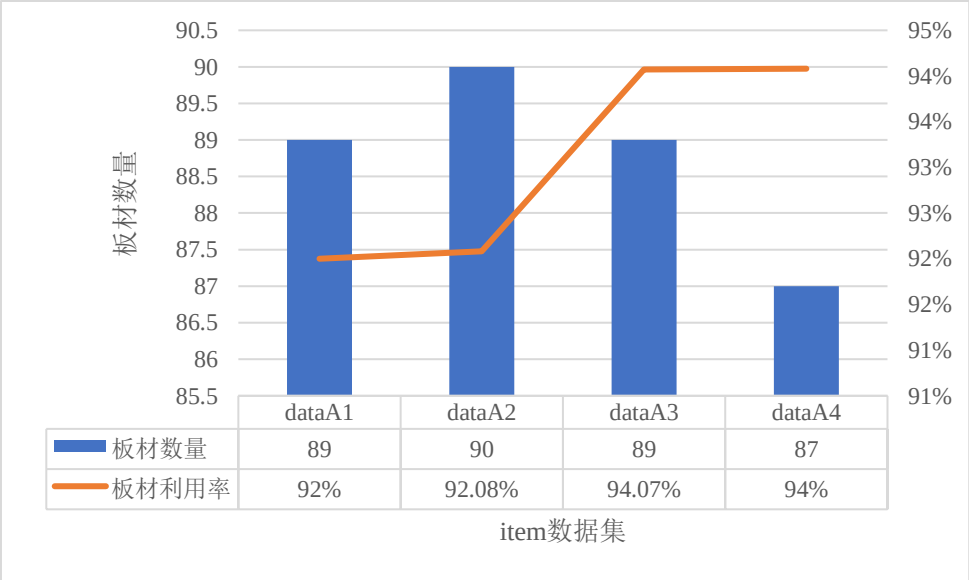


图 4-10 板材利用率结果

排样方式最终采用 X 向 3H 排列，在考虑 item 可旋转的情况下，X 向排列的利用率几乎等同于 Y 向排列，因为数据集明显长度特大数据更多，所以最终采用 X 向排列。

在 X 向排列中，Y 向条带生成的结果如图 4-11 所示。每一块板材中，不同颜色代表不同的 item，图中数字代表该块 item 的订单号，矩形的长度代表 item 的实际宽度。由图可知 X 向排列，条带生成接近于板材的宽度。同理，X 向排列的条段生成，接近于板材的长度。

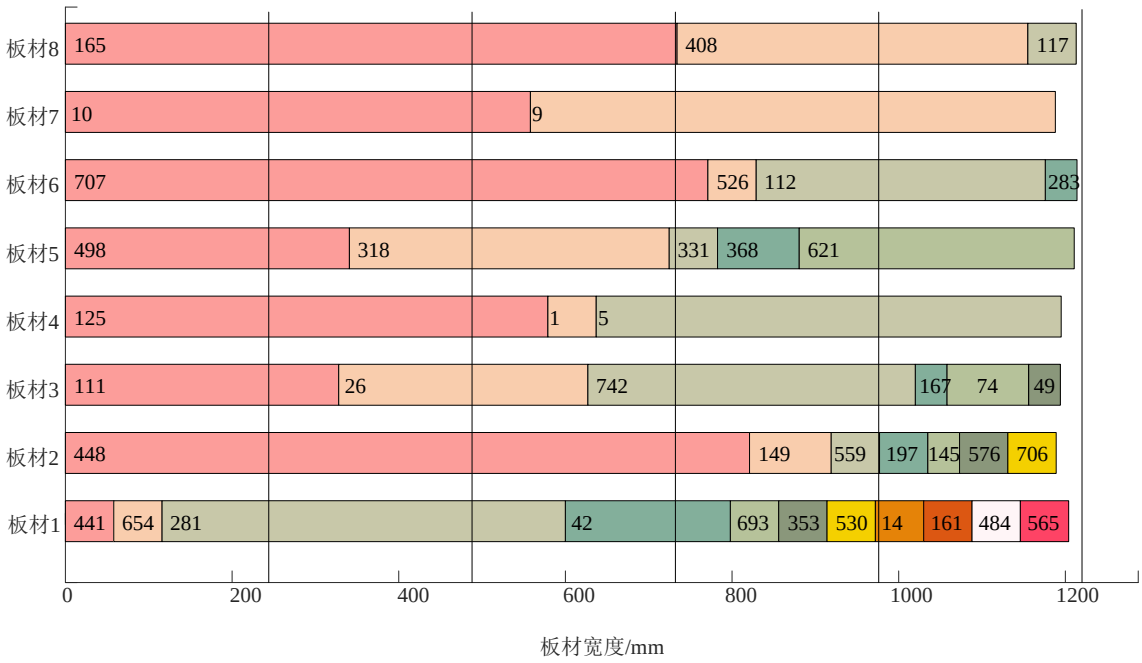
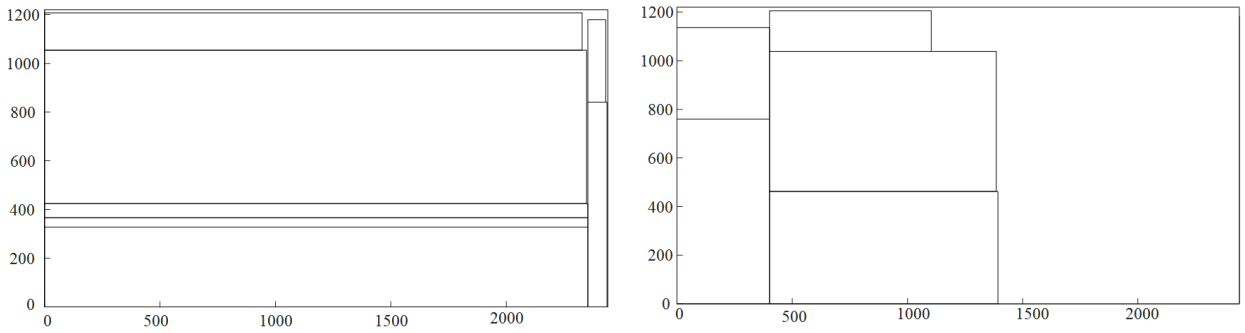


图 4-11 纵向排列条带生成图

按上述排样方法，根据记录的 item 在板材中的坐标，得到拼接后的板材形状如图 4-12 所示。结合表 4-5 所示数据，可以看出，在排样初始时，板材的利用率很高，但随着 item 的不断拼接，每块板材的利用率开始下降。究其原因，后面排样的 item 在同一块板材的匹配度降低。所以，应该优先排特大型号板子、其次大型号和小型号板子。



(a) 第 7 块板材　　(b)第 67 块板材

图 4-12 item 拼接结果图

表 4-5 每组数据集板材利用率

数据集	90%	<80%,90%>	<50%,80%> /板材序号	<50% /板材序号	板材数
dataA1	76	12	1/89	0	89
dataA2	65	13	2/(89,90)	0	90
dataA3	77	11	1/49	1/89	89
dataA4	81	6	0	0	87

表中展示了子问题 1 所给四个数据集的排样信息，包括板材数量、利用率、item 长宽、横纵坐标、以及 item 排样的 x/y 方向的长宽。表展示的为部分结果，全部结果展示如附件（cut_progrom.csv）所示（部分结果展示于表 4-6 中）。

表 4-6 子问题 1 排版方案结果

原片材质	原片序号	产品 id	产品 x 坐标	产品 y 坐标	产品 x 方向长度	产品 y 方向长度
ZQB-0218S	1	30639	0	0	1964	395
ZQB-0218S	1	30471	0	395	1951	577
ZQB-0218S	1	30228	0	972	1913	38
ZQB-0218S	1	30856	0	1010	1888	199
ZQB-0218S	1	30365	1964	0	476	665
…	…	…	…	…	…	…
…	…	…	…	…	…	…
ZQB-0218S	87	30747	6271.5	0	281.5	569
ZQB-0218S	87	30977	6271.5	569	259	489
ZQB-0218S	87	30500	6850.5	0	253	380.8
ZQB-0218S	87	30461	6850.5	380.8	222	280
ZQB-0218S	87	30244	6850.5	660.8	215	449

（2）修正算子

所以修正算子的适当选取与利用率呈强相关性关系，最终经过 5000 次的选择，确定修正算子如表 4-7 所示。

表 4-7 不同数据的修正算子结果

数据集	γ	α	β	χ	δ
dataA1	50	0.95	0.95	0.5	0.5
dataA2	56	0.9	0.9	0.45	0.4
dataA3	57	0.9	0.9	0.4	0.45
dataA4	55	0.85	0.85	0.5	0.4

由表可知，在 dataA1 中 χ 和 δ 最小，说明在 dataA2、dataA3 和 dataA4 分类给小型号的数量过多，所以需要通过则修正算子调整型中型型号的数量。

（3）排列组合

生成的组合数据集，将其与剩余未拼接 item 数据集组合在一起，生成新 item 数据集，用于排样优化与后续板材利用率求解计算。对排列组合 item 数据，进行 5000 次的不断优化，最终确定如图 4-13 所示的组合数据。

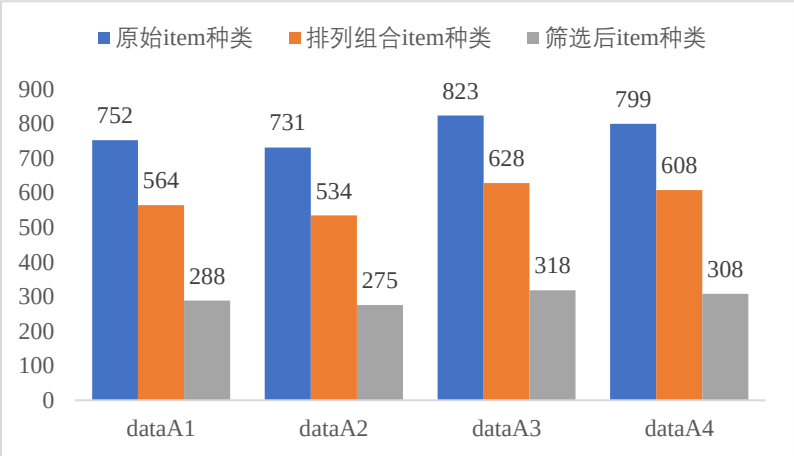


图 4-13 子问题 1 中四组数据原始 item 种类和处理后的 item 种类

可以看到，满足尺寸约束的 item 拼接组合并不多，经过筛选后，可以比较有效地减少数据量，进而可以减少求解复杂度和时间成本。

（4）item 型号分类

在表 4-7 所示的修正算子中，item 型号在优化前和优化后的分类结果如表 4-8 所示：

表 4-8 优化前后分类对比分析

数据集	优化前			优化后		
	小型号	大型号	特大型号	小型号	大型号	特大型号
dataA1	529	84	139	280	97	174
dataA2	517	66	148	191	86	179
dataA3	611	73	139	205	112	188
dataA4	620	66	113	220	97	174

（5）排样方法比较

1）利用率比较

经过排列组合优化后，与传统算法对比，利用率结果如图 4-14 所示。结果显示，采用排列组合的优化方法，可以提升 7~8%的板材利用率。而且数据量越大，经组合处理后，算法的稳定性也有所提升，传统算法波动范围平均为 75%~85%，而经过优化后的算法，可以将波动范围降低至 85%~92%。

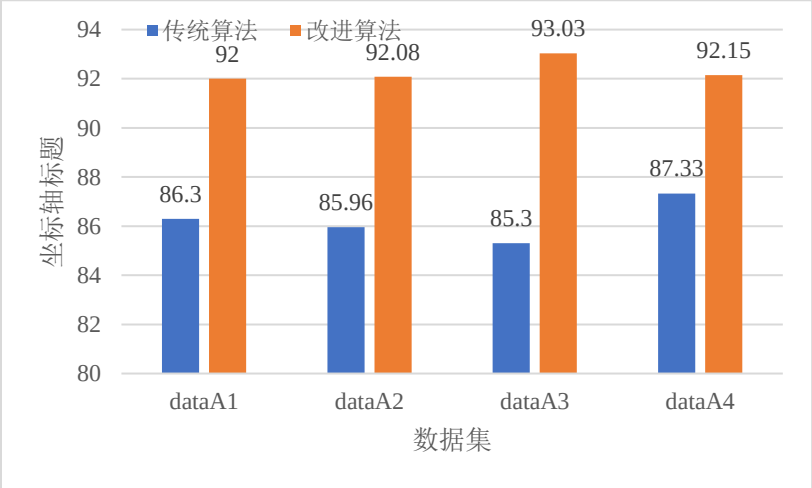


图 4-14 传统算法与改进算法板材利用率的对比结果

2）算法稳定性分析

由于每次修正算子不同和求解方法的随机性，算法表现的稳定性也是需要考虑重要因素之一。如表 4-9 所示，通过不同参数多次重复实验，改进算法排样的板材利用率可以稳定到 90%左右，但传统算法上下波动超过 8%。所以改进算法稳定性更好。

表 4-9 基于子问题 dataA1 的算法稳定性比较

算法种类	修正算子组合 A		修正算子组合 B	
	求解次数	稳定范围	求解次数	稳定范围
传统算法	500	91±2	500	90±3
改进算法	500	86±4	500	85±5

5. 问题二的分析及模型的建立与求解

5.1 问题分析

5.1.1 问题描述

方形件大规模个性化定制生产中，订单组批与排样优化之间存在强耦合关联，在订单交货期、原料下料率、设备利用率等方面存在时间或空间维度上的不可协调性，从而导致两个问题求解计算的非收敛式迭代，计算时间长，材料利用率低，组批不合理等。

订单组批问题是研究如何实现订单的批量化生产，需要考虑订单材质、板件数量、交货期等，并按实际生产约束等进行订单组合，获得最优的组批方案，以提高产线效率，缩短订单完工期。下料排样是板式产品的首道工序，对板件进行排版优化以最大化原材料利用率，直接影响生产批次构成与原料成本。孤立的考虑这两个问题，会出现最优组批方案对应的板材开料利用率低，对应的订单完工期达不到要求等问题。因此，需要对订单组批与排样问题进行协同优化，以利于实现生产过程的系统性优化，同步提高材料利用率和生产效率，减少批次，降低成本[8]。

5.1.2 问题分析

子问题 2 的约束都基于子问题 1 并与之相容，在满足子问题 1 约束的基础上进一步要求：将生产订单进行组批，考虑材料种类、批次大小、订单数量、材料利用率、生产设备产能与负荷等因素，实现订单生产的高效性，同时保证其个性化。

订单组批过程是一个典型的组合优化问题，按实际生产约束与交货期等要求进行订单组合，如果把所有可能的排列组合方式进行例举，对于少量订单组合时或许可行，但是对于订单量比较大时，穷举所产生的组批方案种类将非常庞大。而排样优化问题是一个 NP-Hard 问题，排样问题的搜索与求解空间很大，计算耗时长。如果按常规方式将订单组批后，直接调用板材排样优化算法，排样结果（利用率）一旦不理想则进行下一次迭代（更换组批方案)，这种迭代方法将无法确保求解过程的收敛速度及解的质量，最终导致整个求解过程变得非常漫长[9-10]。

排样下料优化预测问题是指对于每个排样优化任务不需要运行或调用排样优化算法计算排样结果，而是直接利用输入的排样任务数据（包括切片信息和原片信息）之间的数值关系及特征对排样优化结果（包括使用原料情况和下料利用率等）进行预测，通过机器学习模型使得预测的排样结果不断逼近实际排样结果的一类预测问题。

本文针对该问题提出了一种基于聚类和预测下的订单组批与排样迭代优化方法。首先提出一种满足生产工艺约束的订单凝聚层次聚类算法对订单进行组批，再利用随机森林预测模型对组批后的订单进行排样利用率预测，按预测的效果选择好的组批方案进行排样，最终确定预测下的最优排样方案对其进行实际两阶段排样优化。

5.2 模型建立

5.2.1 上层：基于聚类与预测的组批优化模型

（1）上层模型 part Ⅰ：聚类优化模型

方形件的组批问题目前采用较多的解决方法是层次聚类算法，其原理是假设每个订单都是单独的簇类，然后在算法运行的每一次迭代中找出相似度较高的簇类进行合并，该过

程不断重复，直到达到预设的簇类个数 K，即将方形件完成组批。

根据所提供数据，每份订单中都包含几种材质的方形件，每个订单数据包含了材质种类、方形件尺寸和数量。为使具有相同材质的订单组合在一起，需要建立各订单之间材质相似性的度量方法，再根据聚类算法将订单进行组批。

1）订单数据预处理与相似性分析

生产订单的组批通常按其订单相似性，将不同订单组成若干批次，以解决需求个性化与生产高效性之间的矛盾。而订单组批考虑的因素包括：材料种类、批次大小、订单数量、材料利用率、生产设备产能与负荷、设备加工时间与效率等诸多因素。组批的订单数量如果太少，则会导致材料利用率过低，生产切换时间长，设备效率低；组批订单数量如果太多，订单完生产工期较长，订单分拣难度增加，交期得不到保证。通常将同型号材质、交货期相近、相似工件尺寸的订单尽量放在同一批次，以减少加工设备的调整次数，提高材料利用率，提高生产效率。

本题假设所有订单的交货期均相同，因此，批次的定义为完成若干订单全部任务且不含任何不完整订单任务的订单集合。

2）订单凝聚层次聚类模型

凝聚层次聚类是一种自下而上的算法[11]，如图 5-1 所示。首先将每个订单都视为一个簇，然后开始按一定规则，将相似度高的簇进行合并，最后所有订单都形成一个簇或达到某一个条件时，算法结束。例如，当阈值距离设置为 *dist*3时，可以将订单分为{[1，2，3，4]，[5，6，7]，[8，9，10，11]}3 类。凝聚层次聚类的每一步都是着力于将两个最相似的簇合并，达到了局部最优，但是，由于该操作无法根据之后的结果，对之前的聚类结果进行更新，所以，凝聚层次聚类不一定能通过每一次的局部最优，最后达到全局最优。因此，一般而言，我们可以先用其他的一些聚类方法（例如 K-Means）等，先将样本数据聚合成初步的簇群，然后再利用凝聚层次聚类对簇群进行聚类。

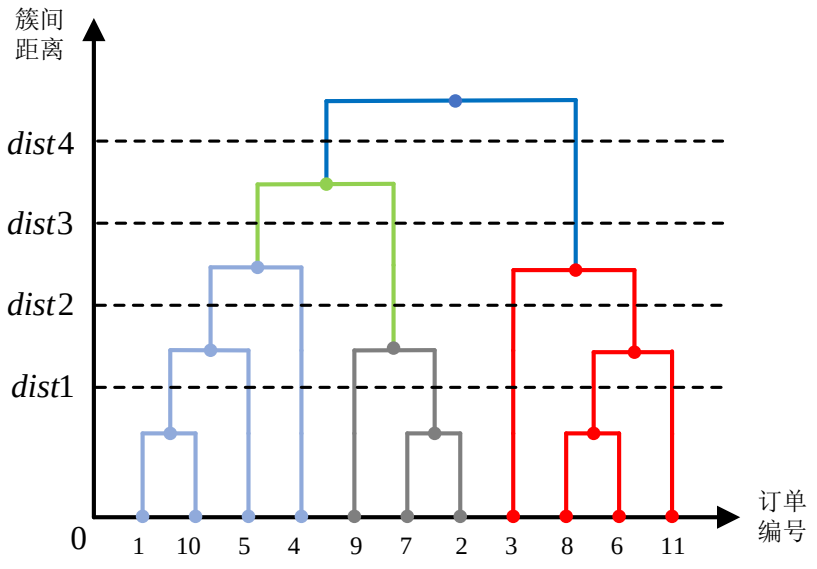


图 5-1 凝聚层次聚类算法示意图

由于上述中的聚类方式没有考虑生产约束的限制，聚类后的批次会出现材料种类偏多等问题，导致组批后无法投放生产，因此需要在聚类的过程中加入实际生产约束条件。在层次聚类的过程中，每次合并前先判断两个批次的订单合并后是否满足如下条件：合并后批次内产品项面积总和上限 *max_item_area*=250(m^2)；订单数量 *max_item_num*≤1000 等。如果满足这些约束条件，则两个类合并；如果不满足合并条件，再寻找次小的类间距

离批次进行合并，直到都不能合并为止。

（2）上层问题 partⅡ：排样下料优化预测模型

为适应本文小批量、多品种、大规模的订单进行多单生产，避免出现仅采用聚类而导致的部分板材下料率低、计算时间长、生产效率低与交期不能满足等问题，提出了一种基于历史数据驱动的板材下料率预测模型，并通过预测模型进行订单组批与排样的迭代优化方法。该方法的核心是利用预测模型对组批方案进行排样结果预测，根据预测结果从所有组批方案中选择较优（前 10%）的组批方案进行真实的排样优化，在不降低解质量的基础上达到快速切割的效果。随着预测模型训练数据的增加，预测精度也将不断提高，则可以加大剪支的范围，从而实现更快的迭代。

1）数据特征提取

排样下料优化预测中，批组的数据量决定了机器学习的上限，对应预测方式（具体预测算法）应可能与之相近。预测模型的优化效果依赖于算法对数据集的处理，本质上表现为对特征值的提取，这对历史数据提出了一定要求，因此需要设计尽可能体现各自特点的特征数据作为输入。问题中提供的数据包含订单的尺寸大小和数量，原板材信息等，而模型中需要体现的优化目标为下料利用率、板材使用数量等，为实现模型特征参数尽可能最优，需要提取与目标相关的主要特征值。

本文针对板材下料预测模型特点，选择特征值时尽可能体现：①item 与条带之间的关系，②item 本体的大小特征，③条带与段之间的关系。基于以上关系提取包含有：①面积数据集（板材与 item 个体的面积比值），②周长数据集（板材周长与 item 个体的面积比值），③长边数据集（板材长边与 item 长边的比值），④短边数据集（板材短边与 item 短边的比值），⑤长宽比数据集（板材与 item 长宽比的比值）⑥数量数据集（item 个体数量大小）

基于以上六个数据集合，进行与目标相关度较高的特征值提取，首先生成体现数据集的 8 个特征值：①数据集合的平均值，②数据集合的最大值，③数据集合的最小值，④数据集合的中位数，⑤数据集合的标准差，⑥顺序排序下前后 10%个数据求和再作比值，⑦顺序排序下前后 25%个数据求和再作比值，⑧顺序排序下前后第 25%个数据的比值。

基于以上数据集合及相关特征参数，确定 6*8=48 个特征值，并且还要补充包含有板材大小（长与宽）、方形快订单数目、item 订单种类与材质、组批数目、切割总面积，合计 6 个特征值，因此最终确定提取特征值总数为：54+7=61 个。

2）模型选择

为使得模型拥有稳定性高，可靠性强，泛化能力强的回归模型，采用基于决策树的集成学习算法模型作为回归预测。选择随机森林集成学习模型[12]，此模型下随机森林的机器学习模型如表 5-1 所示。

表 5-1 随机森林学习模型介绍

模型名称	输入参数	输出参数	数据集划分
随机森林	61 个特征值数据 特征数据降维处理	板材使用量及利用率	训练/测试集： 0.25

集成学习通过结合多个预测优化结果改善集成学习器的泛化能力。随机森林以决策树形式作为基学习器，对训练模型的优化至关重要，其参数影响着学习效率。在模型最优参数配置过程中，采用交叉验证实现最优参数辨识并返回回归模型继续学习，最终确定最优参数。

对应回归模型参数组合寻优设置如表 5-2 所示：

表 5-2 回归模型参数组合寻优设置

预测模型	参数	含义	取值区间	步长	板材
随机森林	max_depth	决策树深度	（5，45）	10	10
	n_estimators	基学习器数目	（150，200）	10	150

3）模型评价指标

针对以上下料预测模型，采用两个指标来表征模型最终预测结果的优劣，分别为：平均绝对误差百分比（MAPE）和误差平方根均值（RMSE）。

平均绝对误差百分比：其大小用以判定模型的鲁棒性，数值越小越稳定，并且能够体现实际结果同预测结果之间的差距，对应公式(5-7)。

$$\mathrm{MAPE}=\sum_{i=1}^{n}\left|\frac{\mathrm{y}_i-\hat{y}_i}{\mathrm{y}_i}\right|\times\frac{100}{m} \tag{5-7}$$

其中，m 表示测试样本数，y_i 表示样本实际标签，$\hat{y}_i$ 表示预测模型预测结果。

误差平方根均值：其表示预测值与真实值误差平方根的平均值，体现了对于数据中较大或较小的数据的情况，能够对模型的精确度做出定量的结果展示，对应公式(5-8)。

$$RMSE=\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y_i-\hat{y}_i\right)^2} \tag{5-8}$$

5.2.2 下层：两阶段排样优化模型

基于上层模型模型预测后优化出的最终排样方案数据集，此数据集的性质与子问题 1 的数据特点完全一致，区别在于组批数量较大，因此需要优化步骤较多，但是底层求解模型与子问题 1 相同。

考虑到此处下层优化模型选择的两阶段排样优化模型，其与子问题 1 的求解模型完全一致，模型底层算法与优化框架一致，此处不再重复说明，详细流程如图 4-3 所示。

5.2.3 双层优化模型：基于聚类和预测的组批、排样协同优化

考虑上文建立的上层层次凝聚聚类及排样下料优化预测模型和下层两阶段排样优化模型，确定子问题 2 的最终求解模型：基于聚类与预测模型的组批、排样协同优化模型以及两阶段优化模型，相关流程如图 5-2 所示。

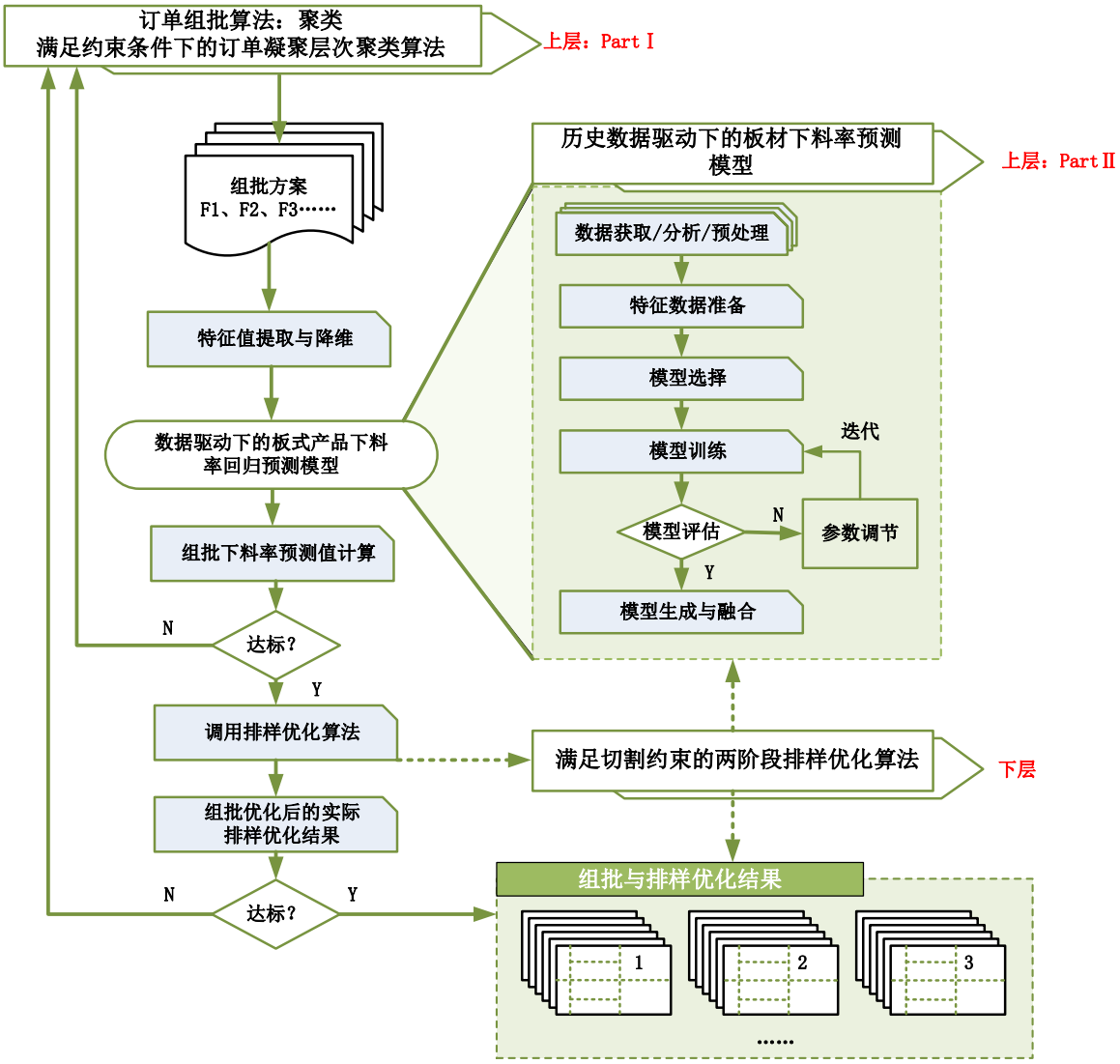


图 5-2 基于聚类和预测的组批与排样协同优化模型流程

5.3 模型求解及优化结果分析

5.3.1 组批优化求解

由上述分析可知，凝聚层次聚类算法是基于簇间的相似度进行的，确定簇与簇之间的相似度是该算法的要点，而这里的相似度是由簇间距离来确定的，簇间距离短的两个簇的相似度高，簇间距离长的相似度低。

（1）样本间相似度计算

由数据分析可知，每个订单都含有几种（两种及以上）的材质，因此属于一个典型的多指离散型。为了能将具有相同材质的订单组合在一起，这里采用基于杰卡德相似性系数（Jaccard similarity coefficient，JSC）的方式来评估两个订单之间材质的相似程度。杰卡德相似系数是衡量两个（订单材质）集合的相似度的一种指标，它表示为两个订单集合 p 和 q 的材质种类交集在 p，q 的并集中的比例，可以表示为：

$$dist(p,q)=\frac{p\wedge q}{p\vee q} \tag{5-1}$$

（2）簇间相似度的计算

基于现实计算考虑，并且融合实际应用可行性、便捷性，本文的簇间距离采用最小方差法。最小方差法：簇类C_p和C_q的距离等于两个簇类所有样本对距离平方和的平均，与均链接算法很相似，数学表达式为：

$$dist(C_p,C_q)=\frac{1}{|C_p|\cdot|C_p|}\sum_{p_i\in C_p,q_j\in C_q}(dist(p_i,q_j))^2 \tag{5-2}$$

综上介绍，基于生产工艺约束的凝聚层次聚类算法流程如表 5-3 所示：

表 5-3 基于生产工艺约束的凝聚层次聚类算法流程

算法 1：订单凝聚层次聚类算法 AHC()	
1	假设初始状态下 ***m*** 个订单都各为一个簇类：***C*1(0)，*C*2(0)，…，*Cm*(0)**计算每个簇类之间的相似度，得到 ***m**m*** 维相似矩阵
2	计算簇间距离的最小方差矩阵 ***H*(*n*)**，找出距离矩阵中最小的元素及其对应的两个簇 ***Cx*(*n*)**和 ***Cy*(*n*)**
3	考虑生产条件约束 如果步骤 **2** 中得到的两个簇合并后满足约束，就进行合并，形成新的簇 ***C*1(*n*+1)，*C*2(*n*+1)，…，**否则返回 **2** 找出次小元素对应的簇进行合并检验
4	计算合并后新簇之间的距离 ***H*(*n*+1)**
5	调至步骤 **2**，重复计算及合并
	结束条件：1）设置类间距阈值 ***T***，当距离最小方差矩阵的最小分量超过了 ***T*** 时，算法停止；2）所有簇合并后都不满足生产工艺约束条件时，算法停止。

5.3.2 组批、排样协同优化求解

（1）基于随机森林的排样下料优化预测回归求解

利用回归模型对板材订单任务的历史数据（来源于子问题 1）进行训练学习，将子问题 2 中的组批方案作为输入，对排样优化结果（板材利用率和数量）进行预测，筛选出前 10%组批方案通过方案 1 求解最优排样方案。

通过随机森林对此排样问题的 61 组特征进行回归学习，得到模型的 MAPE 及 RMSE 评估值输出。排样下料预测问题的随机森林具体步骤如表 5-4 所示：

表 5-4 排样下料预测模型中随机森林算法运行流程

算法 2：随机森林 RandomForestGegressor()	
	输入：含有 ***m*** 条排样数据样本集$D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$ **输出：**强分类器
1	如果$t=1,t=2,\ldots,t=T$
2	***P*** ←生成排样方案 1 训练集D中进行第t次有放回的随机采样，更新$D_1^{'}=\{(x_1^{'},y_1^{'}),\{(x_2^{'},y_2^{'}),\ldots,\{(x_m^{'},y_m^{'})\}$ 2 用$D_t^{'}$训练第t个学习器$E_t(x)$，期间随机选择节点样本特征并择优处理，用以决策树划分
3	对T个学习器输出的预测值算术平均处理并进行 ***MAPE*** 等价值判定
4	输出回归预测值

注：重点调节随机森林框架参数 *n_estimatorrs*, *obb_score* 和 *boostrp*。

（2）基于数据驱动的组批与排样协同优化

基于数据驱动的下料率预测模型筛选组批方案的完整流程框架如图 5-3 所示。

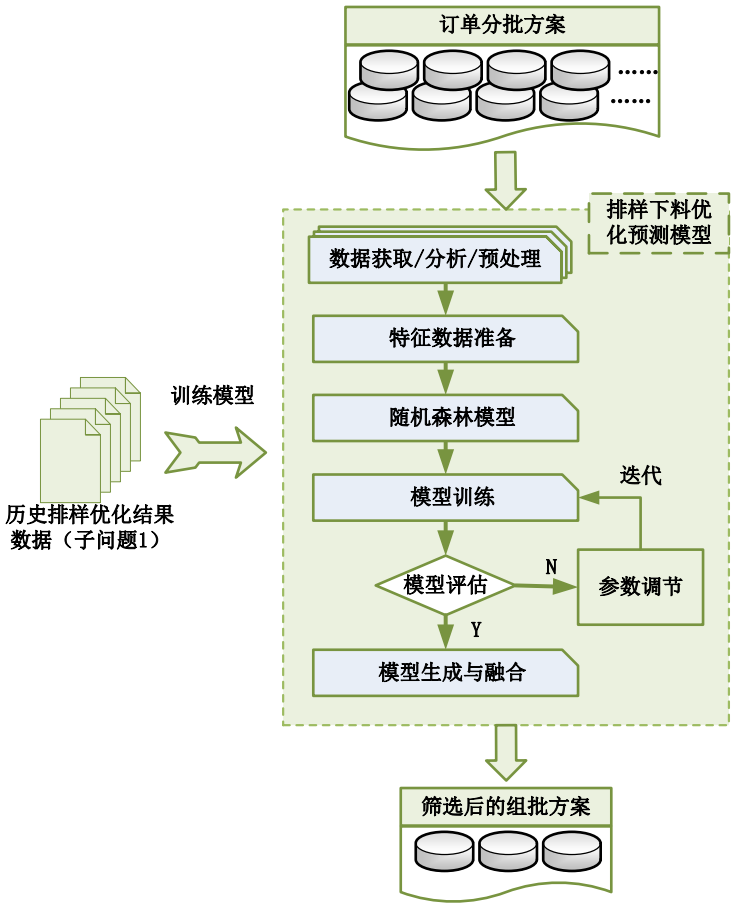


图 5-3 基于数据驱动的下料率预测模型

模型解释：首先，基于数据驱动下的预测模型进行回归训练，得到满足场景要求的预测模型。其次，对基于凝聚层次聚类模型下的组批方案进行整理，并输入预测模型进行组批方案排样结果预测，筛选出价值排序靠前的几组方案。最后，对筛选出的组批方案带入子问题 1 的两阶段迭代排样优化模型中求解出最终切割方案。

表 5-5 组批与排样双层协同优化流程

流程 1:	
	输入: 满足约束的订单凝聚层次聚类组批
	输出: 组批与排样优化结果
1	基于生产约束的订单凝聚层次聚类算法进行订单组批优化，获得多个可行的订单组批方案 ***F***
2	对每一个组批方案进行特征提取和降维，即每个组批方案中的每个批次中相同材质的板材作为一个最小预测样本，再基于排样特征值提取与降维方式生产样本数据
3	将每个组批方案中所有的预测样本数据输入到预测模型中进行排样结果预测，计算组批方案中所有批次中所有材料的排样预测结果
4	对所有组批方案的下料率预测值进行计算与评估，判断是否满足要求，如不满足直接返回步骤 **1**，调整组批方案

当 “满足评估要求” **执行** //下一步

否则 返回步骤 **1**

结束

5 选择部分预测利用率较高的组批方案，调用子问题 **1** 的排样优化算法，进行真实排样优化，计算优化结果

6 判断已排样的优化结果是否满足要求，如果不满足，则返回步骤 1，调整与优

当 “满足结果要求” **执行** //下一步

否则 返回步骤 **1**

结束

5.3.3 协同优化结果

（1）订单簇相似度计算结果

以所给数据集 dataB1 为例，订单相似度计算结果如图 5-4 所示：

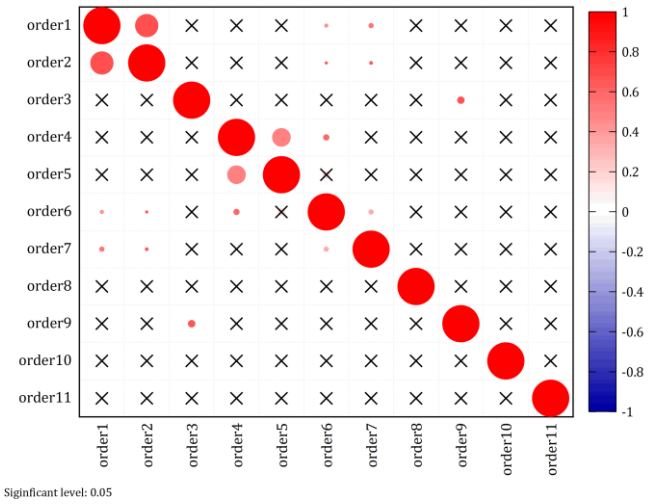


图 5-4 部分订单相似度计算结果展示

图中，红色圆越大，代表两个 order 之间的相似度越高，×代表两个 order 相似度为零，没有相同材质的 item。

（2）批次结果

由表可知，在满足面积约束和组批个数约束的限制下，通过层次聚类和随机森林方法，求解上文建立的组批模型中，每组数据的组批结果如表 5-6 所示：

表 5-6 每组数据下的组批结果

	dataB1	dataB2	dataB3	dataB4	dataB5
item 总数	26811	17952	18028	18526	27901
组批结果	39	38	32	32	42

（3）板材利用率

按上述方法，求得子问题 2 五组数据集的板材利用率如图 5-5 所示。限于篇幅限制，详细的组批方案和排样方式见附件。

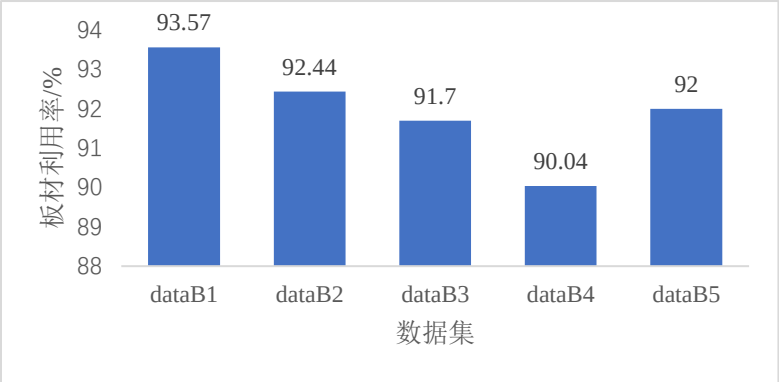


图 5-5 子问题 2 板材利用率

6. 模型的评价

6.1 模型的优点

（1）提出了基于迭代修正算法的两阶段切割下料优化方案，采用自下而上的优化理念，利用迭代过程的价值权重算子更新，尽可能地实现材料利用率最大化和板材使用量最小化。

（2）子问题 1 基本优化方案的材料利用率比较适合实际工业效率要求（80%），对于工业实践具有一定指导意义。

（3）子问题 1 在基本优化方案的基础上，对小 item 进行组合成大块，继而与其他大 item 继续排样的优化方案，实现了最高利用率达到 90%以上。

（3）对于子问题 2 的组批与排样优化问题，通过聚类获取工业生产的组批方案，建立了基于历史数据的随机森林预测模型来选择最优的组批方案，而后通过两阶段排样优化模型进行实际排样切割作业。最终结果符合工业生产的基本原则，该方案大幅度提高了工业计算速度、材料利用率，非常适合“多品种小批量”类的个性化订单需求。

（4）求解模型具有很强的可扩展性，对于处理其他维度和不同阶段的切割下料问题存在一定普适性，模块之间的封装调用便于结果的个性化可视化展示。

6.2 模型的缺点及改进

（1）针对子问题 1 在基础方案上的优化模型能够满足所有切割条件要求，但存在一定坐标的误差，未来需要对模型进行精细化处理，实现 item 的准确定位跟踪。

（2）子问题 1 的求解方案，对于各板材而言均为局部最优，但是从全局角度出发，对于所有使用的板材而言可能不是全局最优解。未来模型可以进一步考虑对切割余料的批量化处理，实现预料二次价值最大化。

（3）针对子问题 2 的基于随机森林的排样下料预测模型虽然在预测方面存在优势，但是对于模型的历史数据集主要参考子问题 1 的排样方案，数据集的预测效果可能存在一定误差，并且回归模型存在过拟合的风险。未来需要结合工厂的实际排样方案数据对预测模型进行训练和参数调节。

7. 参考文献

[1] Silva E， Alvelos F， Valério de Carvalho J M. An integer programming model for two-and three-stage two-dimensional cutting stock problems [J]. European Journal of Operational Research，2010， 205(3): 699-708.

[2] Cui Y D， Huang B X. Reducing the number of cuts in generating three-staged cutting patterns [J]. European Journal of Operational Research， 2012， 218: 358-365.

[3] Puchinger J， Raidl G R. Models and algorithms for three-stage two-dimensional bin packing [J]. European Journal of Operational Research， 2007， 183(3): 1304-1327.

[4] 陈秋莲，王成栋.二维剪切排样的束搜索启发式算法[J].计算机工程与应用，2017，53(09):236-239+257.

[5] Qiulian Chen， Yaodong Cui & Yan Chen (2016) Sequential value correction heuristic for the two-dimensional cutting stock problem with three-stage homogenous patterns， Optimization Methods and Software， 31:1， 68-87

[6] Qiang Liu， Hao Zhang， Jiewu Leng & Xin Chen (2019) Digital twin-driven rapid individualised designing of automated flow-shop manufacturing system， International Journal of Production Research，57:12， 3903-3919

[7] 陈秋莲. 二维剪切下料问题的三阶段排样方案优化算法研究[D].华南理工大学，2016.

[8] 张磊. PCB 定制生产的组批与拼版联合优化方法研究[D].广东工业大学，2019.DOI:10.27029/d.cnki.ggdgu.2019.000092.

[9] 杨玉丽，崔耀东，景运革，张青凤.生成矩形毛坯最优三块排样方式的精确算法[J].机械设计与制造，2008(09):11-13.

[10]黄少丽，杨剑，侯桂玉，崔耀东.解决二维下料问题的顺序启发式算法[J].计算机工程与应用，2011，47(13):234-237.

[11] 石剑飞，闫怀志，牛占云.基于凝聚的层次聚类算法的改进[J].北京理工大学学报，2008(01):66-69.

[12] 方匡南，吴见彬，朱建平，谢邦昌.随机森林方法研究综述[J].统计与信息论坛，2011，26(03):32-38.

附录 A 程序代码

（1）子问题一：两阶段优化模型

迭代排样优化算法：

```
function [efficient,order,area_all,axis_xy,x_y]= fuc_main()
clear
data = xlsread('result3.xlsx','Sheet3','D1:F586');
[order,str] = xlsread('result3.xlsx','Sheet3','G1:G586');
% for    i = 1:469
%         st = str{i};
%         order(i,1) = str2num(st(isstrprop(str{i},'digit')));
% end
order2 = [1:1:586]';
data = [data,order2];

area = [data(:,1).*data(:,2)];
L0 = 2440;   %板子长度
W0 = 1220; %板子宽度
S_area = L0*W0;
A = L0*W0;
% material = [];   %材料类型
% material_S = length(material);   %     一共有多少种材料
% material_L = [];        %每种材料下，项目的种类
material_SUM = ones(586,1); %每种材料下，每种项目的需求量
TP = [];    %项目类型（按照长宽和材料类型分类）     material_S * max{Ni}   Ni 表示每个材料中，按长款分类的大小。
leng = [data(:,1)];        %所有类型的板子长度
wid = [data(:,2)];        %所有类型的板子宽度
x = 0;                %当前项目的宽
y = 0;                %当前项目的长
 plate_x = 0;%当前板子的剩余宽
plate_y = 0;   %当前板子的剩余长
plate_wx = 0;   %当前板子的浪费长
plate_wy = 0;   %当前板子的浪费宽
wa_area = 0;     %当前板子浪费面积
%   总浪费面积通过 wa_area 的叠加   最后存入 plate_w
plate_num =1; %板子的频率
C = 0;     %切割成本
 % min z = sum(L0W0+Ck);
copy_data = data;
small = [];
teda = [];
```

```
zhongda = [];
j = 1;
k = 1;
m = 1;
%% 特大
for i = 1:586

    if ((leng(i)>0.5*L0 && wid(i)>0.5*W0) ) || (leng(i)>0.7*L0) ||( wid(i)>0.7*W0)
            teda(j,:) = data(i,:);
            j = j+1;
        continue
    elseif leng(i)>0.5*L0 || wid(i)>0.5*W0
        zhongda(k,:) = data(i,:);
        k = k+1;
        continue;
        else
            small(m,:) = data(i,:);
            m = m+1;
            continue;
    end
end




%F(x) = max(F(0), F())
% seed = randi([2,length(teda)-1]);
% area_state = teda(seed,1);
k=1;
copy_zhong = zhongda;
copy_small = small;
copy_teda = teda;
copy_data = data;
order = zeros(1000,1000);     %记录 order
axis_xy =zeros (1000,100);    %记录 xy 坐标
area_all = [];
xxyy = 1;
%% 初始横向排列
for i = 1:1:sum(leng.*1)
    if size(copy_data,1) == 0
        break
    end
    area_state = 0;
    sprintf('第%d 次循环',i)
```

```
axis_i = 1;
if (size(teda) > 0)
    seed = randi([1,size(teda,1)]);
    size_flag = teda(seed,1:2);

    plate_x = L0-teda(seed,1);
    plate_y = W0;
    order_i = 1;
    order(i,order_i) = teda(seed,3);
    x_y(xxyy,:) = teda(seed,1:2);
    xxyy = xxyy +1 ;
    area_state = size_flag(1)*size_flag(2);
    axis_xy(i,axis_i:axis_i+1) = [0,0];
    order_i = order_i + 1;
    axis_i = axis_i + 2;
    %% 子空间剩余面积\长\宽
    sub_area = W0*size_flag(1) - size_flag(1)*size_flag(2);
    sub_plate_y = W0 - size_flag(2);
    sub_plate_x = size_flag(1);
    %% 删除已用行
    copy_data(find(copy_data(:,3) == teda(seed,3)),:) = [];
    teda(seed,:) = [];
elseif(size(zhongda) >0)
    order_i = 1;
    if i<100
        seed = randi([1,size(zhongda,1)]);
    else
        [s,seed] = max(zhongda(:,1));
    end
    size_flag = zhongda(seed,1:2);
    plate_x = L0-zhongda(seed,1);
    plate_y = W0;
    order(i,order_i) = zhongda(seed,3);
    x_y(xxyy,:) = zhongda(seed,1:2);
    xxyy = xxyy + 1;
    area_state = size_flag(1)*size_flag(2);
    axis_xy(i,axis_i:axis_i+1) = [0,0];
    order_i = order_i + 1;
    axis_i = axis_i + 2;
   %% 子空间剩余面积\长\宽
    sub_area = W0*size_flag(1) - size_flag(1)*size_flag(2);
    sub_plate_y = W0 - size_flag(2);
    sub_plate_x = size_flag(1);
    %% 删除已用行
```

```
        copy_data(find(copy_data(:,3) == zhongda(seed,3)),:) = [];

        zhongda(seed,:) = [];

    else

        order_i = 1;
        if i<100
            seed = randi([1,size(small,1)]);
        else
             [s,seed] = max(small(:,1));
        end
        size_flag = small(seed,1:2);
        plate_x = L0-small(seed,1);
        plate_y = W0;
        order(i,order_i) = small(seed,3);
        x_y(xxyy,:) = small(seed,1:2);
        xxyy = xxyy + 1;
        area_state = size_flag(1)*size_flag(2);
        axis_xy(i,axis_i:axis_i+1) = [0,0];
        order_i = order_i + 1;
        axis_i = axis_i + 2;
       %% 子空间剩余面积\长\宽
        sub_area = W0*size_flag(1) - size_flag(1)*size_flag(2);
        sub_plate_y = W0 - size_flag(2);
        sub_plate_x = size_flag(1);
        %% 删除已用行
        copy_data(find(copy_data(:,3) == small(seed,3)),:) = [];

        small(seed,:) = [];

    end
    %% 坐标
    y_label = size_flag(2);
    x_label = 0;
    %% 找剩余可行值

    while(1)
        %% 还没有考虑 item 情况
        k = 1;
        able_data = [];
        able_ndata = [];
        for j = 1:size(copy_data,1)
            if copy_data(j,1)<= sub_plate_x && copy_data(j,2)<= sub_plate_y || copy_data(j,1)<= sub_plate_y
```

```
&& copy_data(j,2)<= sub_plate_x
                    if copy_data(j,1) == size_flag(1) && copy_data(j,2) == size_flag(2)
                        continue
                    end
                    able_data(k,1:3) = copy_data(j,:);
                    able_data(k,4) = copy_data(j,1) * copy_data(j,2);
                    k=k+1;
                end
            end
            able_data = fun_data(able_data,sub_plate_x);    %%
            if (size(able_data) == 0)
                break
            end

%               able_cha(:,1) = able_data(:,1) - size_flag(1);
%               able_cha(:,2) = able_data(:,2) - size_flag(2);

%         flag = [];
%         flag1 = [];
%         flag2 = [];
%         flag3 = [];
%         flag4 = [];
%         %% 按照横坐标找
%         flag1 = find((zhongda([1:length(zhongda)],1)) == teda(seed,1));
%         flag2 = find((small([1:length(zhongda)],1)) == teda(seed,1));
%         %% 按照纵坐标找
%         flag3 = find((zhongda([1:length(zhongda)],2)) == teda(seed,1));
%         flag4 = find((small([1:length(zhongda)],2)) == teda(seed,1));
%         flag = [flag1,flag2,flag3,flag4];

            able_ndata = sortrows(able_data,-1);
            sub_plate_y = sub_plate_y - able_ndata(1,2);

            order(i,order_i) = able_ndata(1,3);
            x_y(xxyy,:) = able_ndata(1,1:2);
            xxyy = xxyy + 1;
            area_state = area_state + able_ndata(1,4);
            order_i = order_i + 1;

            axis_xy(i,axis_i:axis_i+1) = [0,y_label];
            y_label = y_label + able_ndata(1,2);
            axis_i = axis_i + 2;
            copy_data(find(copy_data(:,3) == able_ndata(1,3)),:) = [];
            if size(teda,1)>0
```

```
            teda(find(teda(:,3) == able_ndata(1,3)),:) = [];
        end
        if size(zhongda,1)>0
            zhongda(find(zhongda(:,3) == able_ndata(1,3)),:) = [];
        end
        if size(small,1)>0
            small(find(small(:,3) == able_ndata(1,3)),:) = [];
        end

    end

    order(i,order_i-1) =    order(i,order_i-1) *1000;
    k = 1;

    %% 横向拼接

    while(1)
        y_label = 0;
        x_label = sub_plate_x + x_label;
        sub_plate_y = plate_y;
        sub_plate_x = plate_x;
        for j = 1:size(copy_data,1)
            if (copy_data(j,1)<= sub_plate_x && copy_data(j,2)<= sub_plate_y) || (copy_data(j,1)<=
sub_plate_y && copy_data(j,2)<= sub_plate_x)
                able_data(k,1:3) = copy_data(j,:);
                able_data(k,4) = copy_data(j,1) * copy_data(j,2);
                k=k+1;
                able_data = fun_data(able_data,sub_plate_x);
            end
        end
        if (size(able_data) == 0)
            break
        end
        k=1;
        flag = 0;
        while(1)
            able_data = [];
            able_ndata = [];
            for j = 1:size(copy_data,1)
                if (copy_data(j,1)<= sub_plate_x && copy_data(j,2)<= sub_plate_y) || (copy_data(j,1)<=
sub_plate_y && copy_data(j,2)<= sub_plate_x)
    %                 if copy_data(j,1) == size_flag(1) && copy_data(j,2) == size_flag(2)
    %                     continue
    %                 end
```

```
                able_data(k,1:3) = copy_data(j,:);
                able_data(k,4) = copy_data(j,1) * copy_data(j,2);
                k=k+1;
            end
        end
        able_data = fun_data(able_data,sub_plate_x);

        if (size(able_data) == 0)
            order(i,order_i-1) =    order(i,order_i-1) *1000;

            break
        end
        able_ndata = sortrows(able_data,-1);
        if flag == 0
            plate_x = plate_x - able_ndata(1,1);
            plate_y = plate_y;
            flag = 1;
        end
        sub_plate_y = sub_plate_y - able_ndata(1,2);
        area_state = area_state + able_ndata(1,4);
        order(i,order_i) = able_ndata(1,3);
        order_i = order_i + 1;
        x_y(xxyy,:) = able_ndata(1,1:2);
        xxyy = xxyy + 1;
        axis_xy(i,axis_i:axis_i+1) = [x_label,y_label];
        y_label = y_label + able_ndata(1,2);
        axis_i = axis_i + 2;
        copy_data(find(copy_data(:,3) == able_ndata(1,3)),:) = [];
    if size(teda,1)>0
        teda(find(teda(:,3) == able_ndata(1,3)),:) = [];
    end
    if size(zhongda,1)>0
        zhongda(find(zhongda(:,3) == able_ndata(1,3)),:) = [];
    end
    if size(small,1)>0
        small(find(small(:,3) == able_ndata(1,3)),:) = [];
    end
        k = 1;
    end
end
area_all(i,1) = [area_state];
area_all(i,2) = [area_state/(L0*W0)];
area_state = 0;
end
```

```
efficient(1,:) = sum(area_all(:,2))/size(area_all,1);
% efficient(1,:) = ;
% sprintf('总利用率%d',sum(area_all(:,2))/size(area_all,1))
item 排列组合拼接程序
```

（2）子问题二：双层优化模型

③ 凝聚层次聚类算法：

```
'''
    凝聚层次聚类算法:
    1. 获取所有样本的距离矩阵；
    2. 将每个数据点作为一个单独的簇；
    3. 基于最不相似(距离最远)样本的距离，合并两个最接近的簇；
    4. 更新样本的距离矩阵；
    5. 重复 2 到 4，直到所有样本都属于同一个簇为止。
'''

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import pdist，squareform
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram

# 参数
labels = ["15Age"， "20Age"， "35Age"， "45Age"， "55Age"， "60Age"]      # 编号
feature = ["location"]      # 特征
data = [15， 20， 35， 45， 55， 60]      # 数据

#通过 pandas 将数组转换成一个 DataFrame
dist_matrix = pd.DataFrame(data， columns = feature， index = labels)
print(dist_matrix)
print("==========================================")

# 只有一个特征(一维)和多个特征(高维)的求法是不一样的， 高维时需要生成对角距离矩阵， 一维则不需要
if(len(feature) != 1):
    # 高纬时运行以下程序
    # pdist: 计算两两样本间的欧式距离， 返回的是一个一维数组
    # squareform: 将数组转成一个对称矩阵
    dist_matrix = pd.DataFrame(squareform(pdist(dist_matrix， metric="euclidean"))， columns = labels， index = labels)
    print(dist_matrix)
```

```
    print("==========================================")

# linkage 是 scipy 自带的层次聚类算法， 输入:
result_clusters = linkage(dist_matrix.values， method = "complete"， metric = "euclidean")
"""
    输出由四列组成：
    第一字段与第二字段分别为聚类簇的编号，每生成一个新的聚类簇就在此基础上增加一对新的聚类簇进行标识
    第三个字段表示前两个聚类簇之间的距离
    第四个字段表示新生成聚类簇所包含的元素的个数
"""
clusters = pd.DataFrame(result_clusters， columns=["label 1"，"label 2"，"distance"，"sampleSize"]，
                                index=["cluster %d"%(i+1) for i in range(result_clusters.shape[0])])
print(clusters)
print("==========================================")

# 画图
dendr = dendrogram(result_clusters， labels = labels)
plt.ylabel("distance between two ages")
plt.savefig("result.png")
print("图片已保存")
plt.show()
```

```
'''
    Find_S 算法实现: 寻找极大特殊假设
'''

# 数据集合
x1 = ['sunny'， 'warm'， 'normal'， 'strong'， 'warm'， 'same' ，1]
x2 = ['sunny'， 'warm'， 'high'， 'strong'， 'warm'， 'same' ， 1]
x3 = ['rainy'， 'cold'， 'high'， 'strong'， 'warm'， 'change'， 0]
x4 = ['sunny'， 'warm'， 'high'， 'strong'， 'cool'， 'change'， 1]

xa = [x1， x2， x3， x4]
h = [ None， None， None， None， None， None ]

sum = 0
for each_data in xa:
    sum += 1
    if each_data[-1] == 0:
        print('第'， sum， '个样本， 规则是：'， h)
```

```
            continue
        else:
            for each in range(len(each_data[:-1])):
                if h[each] == None:
                    h[each] = each_data[each]
                elif h[each] != each_data[each]:
                    h[each] = '?'
            print('第', sum, '个样本， 规则是：', h)
```

④ 随机森林回归模型

```
"""
    随机森林是一种同质的集成学习算法，通过构建多个决策树，然后结合多个决策树的结果，得到更好的预测
"""

import pandas as pd
from sklearn.feature_extraction import DictVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV


def forest():
    # 加载数据
    titan = pd.read_csv("E:/Desktop/机器学习_新/数据集/泰坦尼克数据集/train.csv")

    # 构造特征值和目标值
    feature = titan[["Pclass", "Age", "Fare", "Sex"]]
    target = titan["Survived"]

    # 特征预处理
    # 查看有没有缺失值
    print(pd.isnull(feature).any())
    # 填充缺失值
    Age = feature.pop("Age")  # 取出，意思是取出来之后删除原来的
    Age = Age.fillna(Age.mean())
    feature.insert(0, "Age", Age)

    # 字典特征抽取
    dv = DictVectorizer()
    feature = dv.fit_transform(feature.to_dict(orient="records"))
    feature = feature.toarray()
```

```
    print(feature)
    print(dv.get_feature_names())

    # 划分数据集
    x_train, x_test, y_train, y_test = train_test_split(feature, target, test_size=0.25)
    print("训练集：", x_train.shape, y_train.shape)
    print("测试集：", x_test.shape, y_test.shape)

    # 建立模型
    rf = RandomForestClassifier()

    # 超参数搜索
    param = {"n_estimators":[10, 20, 30, 40], "max_depth":[25, 35, 45]}
    gc = GridSearchCV(rf, param_grid=param, cv=5)

    # 训练
    gc.fit(x_train, y_train)

    # 交叉验证网格搜索的结果
    print("在测试集上的准确率：", gc.score(x_test, y_test))
    print("在验证集上的准确率：", gc.best_score_)
    print("最好的模型参数：", gc.best_params_)
    print("最好的模型：", gc.best_estimator_)


if __name__ == "__main__":
    forest()



"""
    通过树形结构来实现分类的一种算法，关键在于如何选择最优属性
    通常用三种方式：信息增益（ID3）、增益率（C4.5）、基尼系数（CART）
"""

import pandas as pd
import numpy as np
from sklearn.feature_extraction import DictVectorizer
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split


def de_tree():
    # 加载数据
```

```
titan = pd.read_csv("E:/Desktop/机器学习_新/数据集/泰坦尼克数据集/train.csv")
# print(titan.shape)
# pd.set_option("display.max_columns", 100)  # 把 dataframe 中省略的部分显示出来
# print(titan.head(5))

# 构造特征值和目标值
feature = titan[["Pclass", "Age", "Fare", "Sex"]]
target = titan["Survived"]

# 特征预处理
# 查看有没有缺失值
print(pd.isnull(feature).any())
# 填充缺失值
Age = feature.pop("Age")  # 取出，意思是取出来之后删除原来的
Age = Age.fillna(Age.mean())
# print(feature)
# feature.drop("Age", axis=1, inplace=True)  # 删除一列
feature.insert(0, "Age", Age)
# print(pd.isnull(feature).any())

# 字典特征抽取
dv = DictVectorizer()
feature = dv.fit_transform(feature.to_dict(orient="records"))
feature = feature.toarray()
print(feature)
print(dv.get_feature_names())

# 划分数据集
x_train, x_test, y_train, y_test = train_test_split(feature, target, test_size=0.25)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.25)
print("训练集：", x_train.shape, y_train.shape)
print("验证集：", x_val.shape, y_val.shape)
print("测试集：", x_test.shape, y_test.shape)

# 建立模型
tree = DecisionTreeClassifier(max_depth=5)

# 训练
tree.fit(x_train, y_train)

# 验证
score = tree.score(x_val, y_val)
print("在验证集上的得分：", score)
```

```
    # 预测
    score_test = tree.score(x_test, y_test)
    print("在测试集上的得分：", score_test)
    predict = tree.predict(x_test)
    print("测试结果：", predict)

    # 保存树结构
    export_graphviz(tree, out_file="E:/Desktop/开题报告/tree.dot", feature_names=['Age', 'Fare', 'Pclass',
'Sex=female', 'Sex=male'])

    # 将保存的 dot 文件转成 png 文件，查看树结构
    # dot -Tpng tree.dot -o tree.png


if __name__ == "__main__":
    de_tree()
```

（附录仅展示部分代码，其余代码见附件）