基于MILP和遗传算法的最优切割问题研究

# 摘要

本文针对窗边框生产中的最优切割问题，通过建立混合整数线性规划（MILP）模型和遗传算法，实现了对多约束条件下切割方案的优化求解。

对于问题 1，充分考虑了实际生产中订单需求的多样性，每个订单对窗框的宽度和高度都设定了明确的目标尺寸，同时为适应生产工艺的公差要求，还给出了相应的允许误差范围。此外，原材料的规格差异，包括不同的长度规格及其对应的单价，以及切割过程中不可忽视的锯口宽度。在此基础上，以实现利润最大化为目标，通过建立线性规划模型，全面且系统地考虑了原材料切割的各种可能组合方式。经过一系列的优化计算，最终确定了满足订单需求且利润最大化的最优切割方案。

问题 2 进一步贴合实际生产环境的复杂性，在问题 1 的基础上引入了原材料存在不规则缺陷这一关键因素。由于这些缺陷的位置和大小会直接影响切割后产品的质量，要求加工成品必须避开缺陷区域。针对这一问题，我们重新构建了数学模型，将避开缺陷区域作为核心约束条件融入到切割方案的规划中。在保证避开缺陷的同时，兼顾利润最大化和原材料的高效利用，通过遗传算法制定出符合实际生产需求的最优切割方案，并再次计算调整后的切割损失率和利用率。

问题 3 则基于附件中的详细缺陷数据以及更新后的订单需求，这些需求包含了不同订单对窗框的要求，包括订单量、尺寸目标和单价等信息。我们建立了更为综合和全面的数学模型，该模型充分考虑了约束条件对所求利润的影响。通过运用遗传算法对模型进行求解，最终得出了适用于该复杂场景的最优切割方案。

综上，我们运用混合整数线性规划，动态规划，综合模型等数学模型，讨论了在多种约束条件下的窗边框生产最优切割方案。本研究成果对于提高原材料利用率、降低生产成本、增强企业经济效益具有重要的实践意义。同时，研究过程中所采用的方法和思路，也为解决类似的生产加工优化问题提供了可借鉴的范例，有助于推动相关行业在生产管理和资源优化配置方面的发展。

关键词：窗边框生产，优化模型，混合整数线性规划（MILP），遗传算法

一、 问题重述

在进行窗边框的生产中，不仅需要将用户的目标尺寸和目标范围纳入考虑范围，还要求计算锯口宽度在切割中造成损失，以及调整切割方案，从而避开缺陷区域，在保证生产质量的同时最大化利用原材料，并考虑切割机的切割占用宽度。对于订单中包含多个有目标尺寸和允许误差（0.01 米）的加工需求，通过建立数学模型求出切割方案最优解，达到利润最大化，计算方案中整体的切割损失率和利用率。

问题一：原材料已知长度 5.5 米、6.2 米、7.8 米，单价为 18 元/根、22 元/根、28 元/根。在锯口宽度（0.005 米）的情况下，设计出最优切割方案，既能满足订单需求，又能最大化利润，并求出切割损失和利用效率。分别针对半价售出和废弃两种超量处理方式，设计切割策略最优解。

问题二：以第一问为基础，调整方案避开随机位置和大小的缺陷，综合考虑不确定性和风险，重新设计切割方案最优解，并对切割损失率与利用率评估。

问题三：优化问题二的模型，根据新的订单需求和原材料设计切割策略最优解，验证策略的有效性。

# 二、 问题分析

2.1 问题一的分析

在窗边框生产中，企业需根据多订单需求，包含窗框的宽度、高度及允许的误差，将长条形原材料切割为部件，每套窗框需两个宽度和两个高度部件，且每次切割需计入锯口宽度的损耗。问题 1 需要在三种不同长度的原材料中选择购买数量及切割方式以最大化利润，并计算整体切割损失率及利用率。核心挑战在于平衡订单完成量与原材料的成本约束，同时满足以下条件：

（1）考虑每次切割的锯口宽度对原材料的损耗，窗框的宽度和高度与切割损耗量之和不超过原材料总长度；（2）窗框的宽高量不超过订单需求量；（3)原材料和窗框的宽高均为整数，通过 MILP，以整数变量表示原材料使用量、订单中窗框的切割段数及订单完成量，线性目标函数为最大化利润，约束涵盖原材料使用量、订单需求量及窗框宽高量关系。模型简化假设窗框尺寸固定，余料不可再利用，从而将复杂非线性问题转化为线性规划，兼顾求解效率与生产实际。

2.2 问题二的分析

在问题 1 的基础上，问题 2 引入原材料的不规则缺陷约束，需调整切割方案以避开缺陷区域（已知起始位置和长度）。核心目标仍为利润最大化，但需额外考虑缺陷导致的可用材料减少及切割复杂性。解决方法需扩展问题 1 的 MILP 模型，并引入遗传算法建立新的模型，进行求解。

2.3 问题三的分析

问题 3 是在前两问的基础上增加了订单量和新的含缺陷的原材料，考虑到订单维度量较大，原材料的的种类数目多，普通优化类算法难以解决，为了求解该大规模优化模型，需要采用启发式遗传算法进行求解。

2.4 总思路图

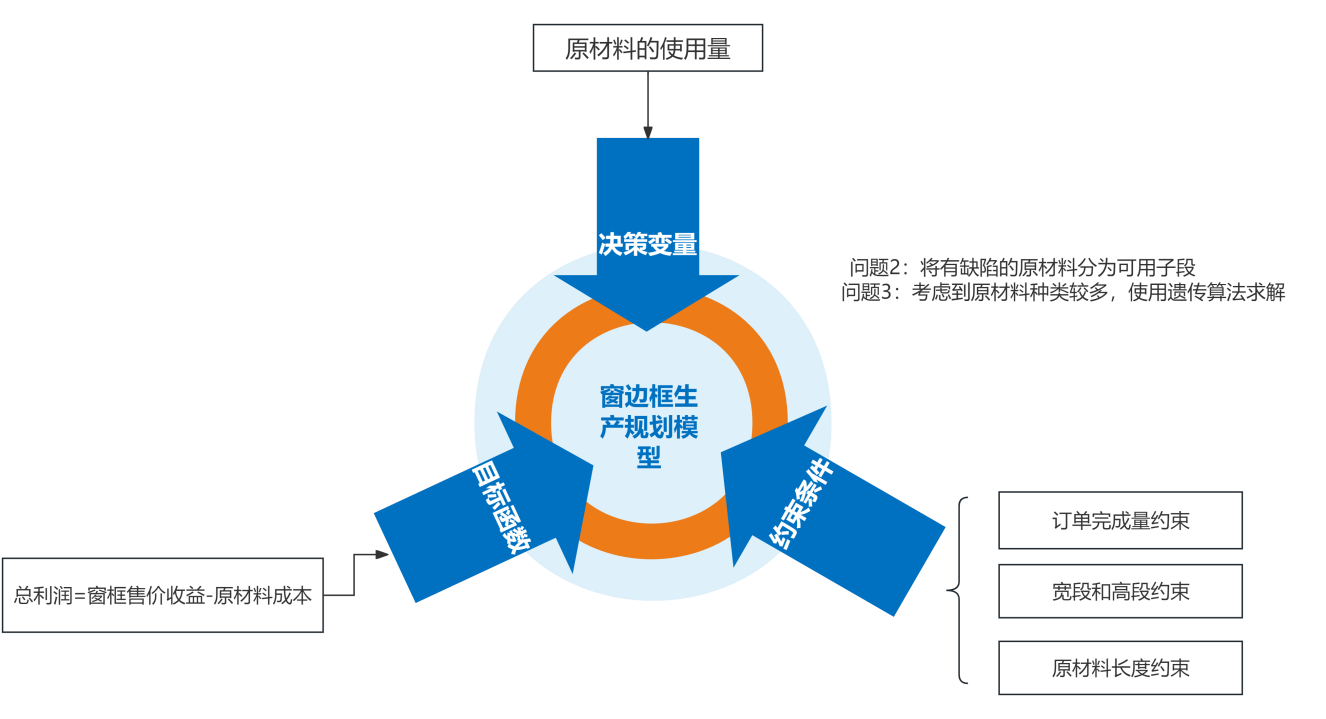


图 **1** 问题分析总思路图

# 三、 模型假设

1.原材料为均匀材质，除已知可能存在的不规则缺陷外，无其他影响切割和质量的因素。且原材料供应充足，无数量限制。

2.假设切割设备精度稳定，切割宽度固定为给定的 0.005 米，且切割过程中不会出现额外的误差或故障。

3.窗框的宽度和高度固定为目标值，忽略允许的±0.01 米调整，以简化模型。

4.不考虑切割顺序的影响，仅考虑总长度需求。

5.目标函数仅包含原材料成本和销售收入，无其他成本。

6.假设订单中的窗框尺寸和数量是准确且固定的，不存在订单变更或误差。

# 四、 符号说明

|  |  |
| --- | --- |
| 符号 | 说明 |
| 𝑥𝑖,𝑗,𝑘 | 第 i 种原材料切割的订单 j 的宽度（k=1）或高度（k=2）段数 |
| 𝐿𝑖 | 第 i 种原材料的总长度 |
| 𝑆𝑗,𝑘 | 第 j 种订单的宽度（k=1）或高度（k=2） |

|  |  |
| --- | --- |
| 𝑦𝑖 | 使用第 i 种原材料的根数 |
| 𝑧𝑗 | 订单 j 的完成套数 |
| 𝐶𝑖 | 第 i 种原材料价格 |
| 𝐶𝑗 | 第 j 种订单的价格 |
| 𝑀𝑗 | 第 j 种订单的订单量 |
| 𝑅𝐿 | 损失率 |
| 𝑅𝑢 | 利用率 |
| 𝑆𝑘 | 第 k 个缺陷起始位置 |
| 𝑙𝑘 | 第 k 个缺陷长度 |
| d | 锯口宽度 |

# 五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 决策变量的确定

本文以使用原材料的根数为决策变量，设：

## 𝑦𝑖 ≥0

(1) 其中，yi表示使用第 i 种原材料的根数，其数值大于等于零，yi ∈N+。

5.1.2 利润最大化目标

该窗边框生产策略的主要目标是在尽可能提高原材料利用率的前提下，实现利润最大化，因此本文以总利润最大化作为模型的目标。

首先考虑原材料的成本，有:

|  |  |
| --- | --- |
| ∑𝑦𝑖 ⋅𝑐𝑖  其中，𝑐𝑖为第 i 种原材料的价格。  再考虑窗框销售的收益，有： | (2) |
| ∑𝑧𝑗 ⋅𝑐𝑗 | (3) |

𝑗

其中，zj表示订单 j 的完成数，cj表示第 j 种订单中窗框的单价。

因此，我们的目标函数如下：

|  |  |
| --- | --- |
| max𝑃=∑𝑧𝑗 ⋅𝑐𝑗 −∑𝑦𝑖 ⋅𝑐𝑖  𝑗 𝑖  其中，P 为总利润。综上，模型的目标为 | (4) |
| min𝑝=−∑𝑧𝑗 ⋅𝑐𝑗 +∑𝑦𝑖 ⋅𝑐𝑖 | (5) |

𝑗 𝑗

### 5.1.3 约束条件

为了更好地描述约束条件，本文将订单量及完成量、已使用原材料总长度与完成订单总长度进行了更细致的划分，完成订单总长度划分为：

1. 𝑆𝑗,𝑘，第 j 种订单的宽度（k=1）；
2. 𝑆𝑗,𝑘，第 j 种订单的高度（k=2）。

完成量划分为：

1.𝑥𝑖,𝑗,𝑘，第 i 种原材料用于切割订单 j 的宽度（k=1）的段数；

2.𝑥𝑖,𝑗,𝑘，第 i 种原材料用于切割订单 j 的高度（k=2）的段数。

约束 1：订单完成量约束

𝑧𝑗 ≤𝑚𝑗 (6)

其中，𝑚𝑗为第 j 种订单的完成量。

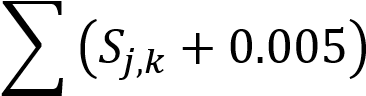
约束 2：宽段数量约束

∑𝑥𝑖,𝑗,1 ≥2𝑧𝑗 (7)

𝑖 约束 3：高段数量约束

∑𝑥𝑖,𝑗,2 ≥2𝑧𝑗 (8)

𝑗,𝑘 约束 4：原材料长度约束

𝑥𝑖,𝑗,𝑘 𝑦𝑖(9)

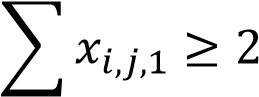
𝑗,𝑘

5.1.4 模型汇总综上，本文建立了以总利润为目标的窗边框生产中的最优切割模型。

## 𝑐𝑖

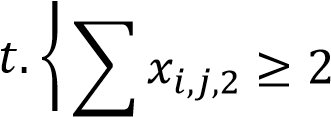
𝑗 𝑖

𝑧𝑗 𝑚𝑗



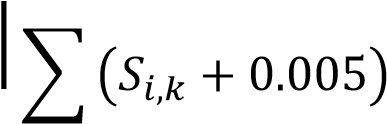
𝑧𝑗 (20)

𝑖

𝑠.𝑧𝑗

𝑗



𝑥𝑖,𝑗,𝑘 𝑦𝑖

{ 𝑗,𝑘

5.1.5 模型的求解与分析最终，利用 MATLAB 求解问题一的最优切割方案为使用 93 根长度为 5.5 米的原

材料，总成本为 1674 元，结果如下表：

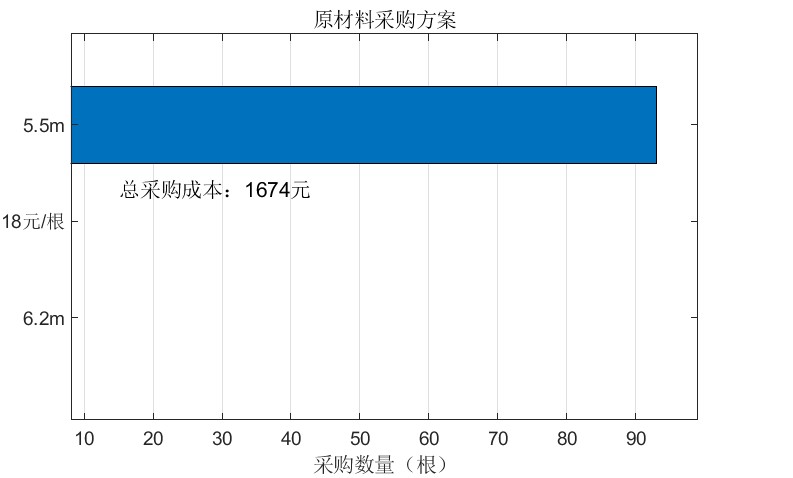


图 **2** 最优切割方案

采用该切割方案，能够实现总利润为 34026 元。各个订单对利润的具体贡献情况详见下图：

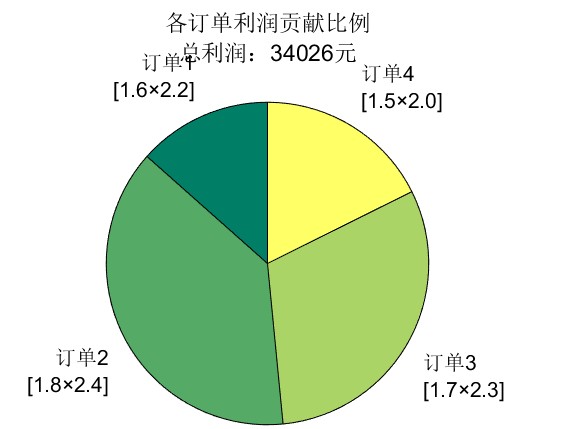
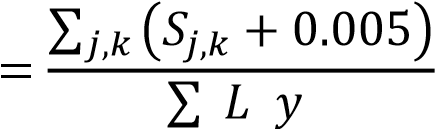


图 **3** 总利润

由此可见，订单 3 和订单 4 由于需求量大且对原材料利用率高，对生产总利润的贡献最为显著。同时，订单 1 和订单 2 虽然需求量相对较小，但通过设计合理的切割方案，也有效地提高了原材料的利用率，从而增加利润。此外，通过对比不同切割方案下的成本和利润，我们发现该最优切割方案在降低成本的同时，最大程度地提升了利润，体现了模型的有效性和实用性。

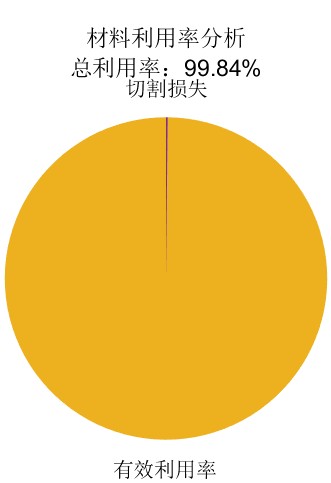
5.1.6 整体损失率与利用率

𝑅𝐿 =∑𝑖 𝐿𝑖 𝑦𝑖 −∑∑𝑗𝑖,𝑘𝐿(𝑖𝑆𝑦𝑗𝑖,𝑘 +0.005) (31)

𝑅𝑢  (12)

其中，𝑅𝐿为原料损失率，𝑅𝑢为原料。

经过验证分析，原料利用率达到了 99.84%，而切割过程中的损失仅占 0.16%，充分满足了尽可能提高原料利用率这一目标。

 图 **4** 最优材料利用率

5.2 问题二模型的建立与求解

### 5.2.1 模型建立

基于问题一的分析，问题二进一步假设每批原材料均存在不规则缺陷，并且缺陷的大小和位置已知，鉴于缺陷会对材料的切割部分产生影响，为了搜索满足避开缺陷区域，满足订单需求且利润最大，采用遗传算法进行优化求解。其步骤如下：

1. 编码设计：可将切割方案进行编码，形成染色体。比如，使用整数编码，将每种原材料切割出不同订单窗框的数量进行排列组合。原材料有 3 种，订单有 4 种，可设计一个长度为 3×4 的整数数组作为染色体，数组元素表示每种原材料对应每个订单窗框的切割数量。
2. 适应度函数构建：根据题目要求利润最大化，适应度函数可以设定为总收益减去总成本。总收益是切割出的窗框按照单价计算的销售收益，总成本是使用的原材料成本。同时，要考虑惩罚项，若切割方案中存在窗框包含缺陷区域的情况，对适应度进行惩罚，降低其值，以引导算法避开缺陷区域。

max𝑃=∑𝑧𝑗 ⋅𝑐𝑗 −∑𝑦𝑖 ⋅𝑐𝑖 (43)

𝑗 𝑗

其中，P 为总利润。

1. 父代种群初始化：产生问题可能潜在解集的一个初始群体，我们将种群规模设定为50，生成父代种群。
2. 早熟判断：判断当前种群内部是否存在早熟现象，即若当前种群的最优适应度值对应的个体数量超过该种群规模的 10%,则表明可能存在收敛于局部最优解的情况。若存在早熟现象则在当前父代种群中插入一批新生成的基因序列个体后，重新进行适应度计算并排序。
3. 子代的生成：对父代进行选择、交叉和变异进化后进行生成子代种群，选择规则为优先选择适应度函数最小的交叉，其中交叉和变异需满足模型约束条件，我们将进化次数设定为50,变异概率设定为10%。
4. 合并：父代与子代合并，再次根据适应度函数进行排序，最终根据所设定的种群规模大小进行淘汰，生成新一代的种群。
5. 判断进化次数是否达到最大进化次数，若达到则退出，得到窗边框切割最优方案，否则转至4。
6. 遗传操作：选择操作依据适应度值，采用轮盘赌选择等方法，让适应度高的染色体有更大概率被选中进入下一代；交叉操作可以随机选择两个染色体，交换部分基因片段，生成新的切割方案组合；变异操作则对染色体的某些基因进行随机改变，防止算法陷入局部最优。
7. 算法流程：首先初始化种群，生成一定数量的随机切割方案作为初始染色体；接着不断迭代，在每一代中依次进行选择、交叉、变异操作，计算新个体的适应度；当满足设定的终止条件，如达到最大迭代次数或适应度收敛时，输出最优解，即最优切割方案。最后，依据得到的方案计算切割损失率和利用率。

### 5.2.2 模型求解

最终，问题二的最优切割方案为使用 23 根长度为 5.5 米的原材料、32 根长度为

6.2 米的原材料，25 根长度为 7.8 米的原材料。

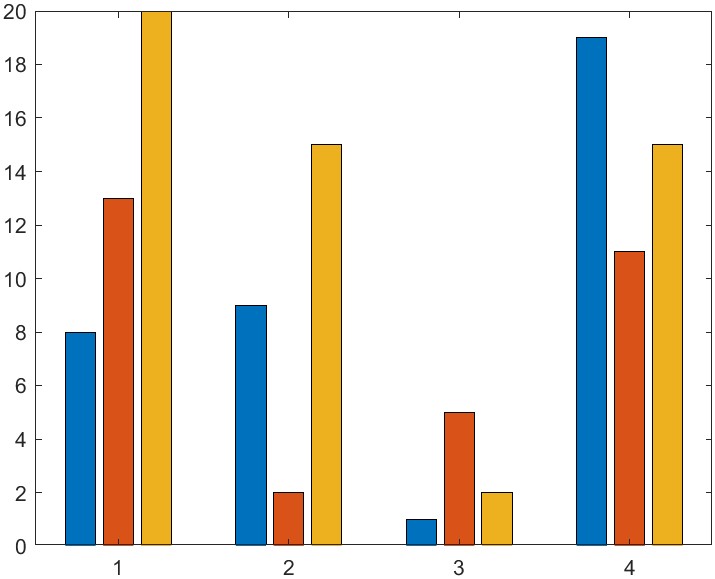


图 **5** 问题二中各订单所用原材料数

### 5.2.3 模型分析

经验证，该方案的原料利用率为 99.4%实现了材料利用率最大化。

 图 **6** 最优材料利用率

5.3 问题三模型的建立与求解

### 模型建立

1. 目标函数：利润等于总收益减去总成本，总收益为各订单销售收益之和，总成本为原材料采购成本之和，构建目标函数： max𝑃=∑𝑧𝑗 ⋅𝑐𝑗 −∑𝑦𝑖 ⋅𝑐𝑖 (54)
2. 约束条件：
3. 窗框数量约束：每种原材料生产各订单窗框数量之和不能超过订单需求量，即：

∑𝑥𝑖,𝑗,1 ≥2𝑧𝑗 (15)

𝑖

∑𝑥𝑖,𝑗,2 ≥2𝑧𝑗 (16)

𝑗,𝑘

1. 原材料长度约束：考虑锯口宽度，每种原材料切割用于生产窗框的材料总长度（含锯口）不能超过原材料长度。

∑(𝑆𝑗,𝑘 +0.005)𝑥𝑖,𝑗,𝑘 ≤(𝐿𝑖 +0.005)𝑦𝑖 (17)

𝑗,𝑘

3. 避开缺陷约束：切割部分不能包含缺陷区域，设切割起始位置为 t，对于每个缺陷(Sk, lk)，需满足t+2xi,j,1 +d≤Sk或t≥Sk +lk。实际计算时，对每种原材料，遍历可能切割方案，若切割部分与缺陷区域重叠则舍弃该方案。

𝑡+2𝑥𝑖,𝑗,1 +𝑑≤𝑆𝑘

(18)

𝑡≥𝑆𝑘 +𝑙𝑘 (19)

4.非负整数约束：𝑥𝑖,𝑗,1 ≥0，𝑥𝑖,𝑗,2 ≥0且为整数，𝑦𝑖 ≥0且为整数。

𝑥𝑖,𝑗,1 ≥0，𝑥𝑖,𝑗,1 ∈𝑁+ (20)

𝑥𝑖,𝑗,2 ≥0， 𝑥𝑖,𝑗,2 ∈𝑁+ (21)

𝑦𝑖 ≥0 ，𝑦𝑖 ∈𝑁+ (22)

5.遗传算法设计

（1）编码采用整数编码，个体编码为：

## [x11,x12,⋯,x1m,x21,x22,⋯,x2m,⋯x11,x12,⋯,xnm,y1,y2,⋯,ym]

其中，前 nm 个基因表示各订单在不同原材料上生产的窗框数量，后 m 个基因表

示各原材料使用数量。（2）初始种群生成

随机生成一定数量的个体，个体中基因值在满足约束条件的可行范围内随机取值，

如xi,j,1，xi,j,2在0到Mj之间，yi根据原材料长度和窗框尺寸估算可能取值范围后随机确定.

1. 适应度函数

以目标函数值作为适应度函数，即Fitness = Z = ∑j zj ⋅cj −∑j yi ⋅ci，适应度值

越大表示个体越优。

1. 选择操作

采用轮盘赌选择法，计算每个个体适应度占总适应度的比例，作为被选择概率，概率越大的个体越可能被选中进入下一代。

1. 交叉操作

选择两个个体，随机确定交叉点，交换交叉点后基因片段，产生两个新个体。交叉过程中需检查新个体是否满足约束条件，不满足则重新交叉。

1. 变异操作

对个体基因以一定变异概率进行变异，随机改变基因值，同样检查变异后个体的可行性，不可行则重新变异。

1. 算法流程

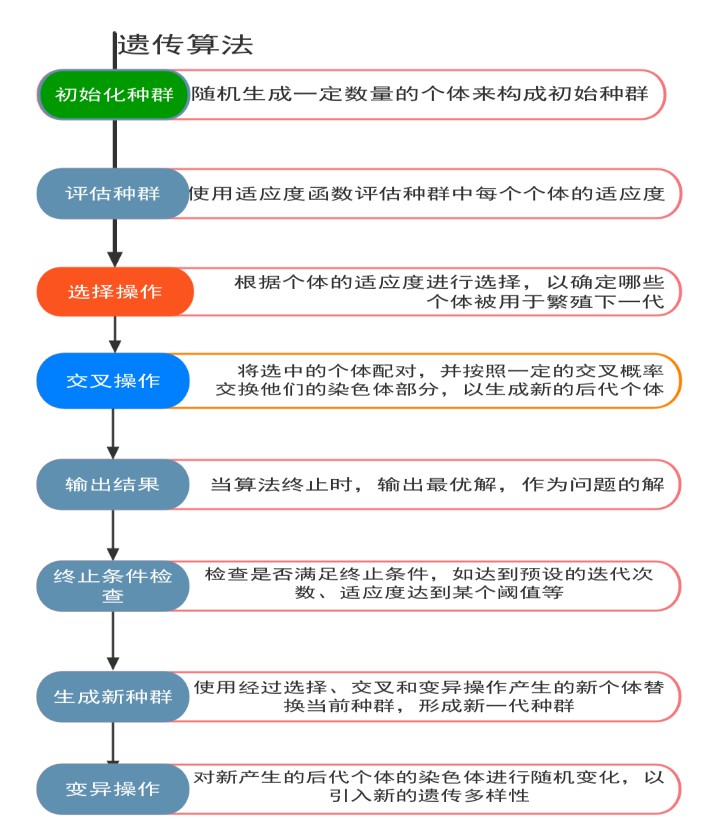


图 **7** 算法流程图

5.3.2 遗传算法求解与分析

使用 MATLAB 编程实现遗传算法，将附件缺陷数据和问题三订单需求数据预处理后输入模型。设置种群规模 50、遗传代数 50、交叉概率 0.8、变异概率 0.2。运行算法得到最优切割方案，计算切割损失率和利用率。根据结果分析不同原材料使用情况、各订单窗框生产分配，评估方案合理性，若不合理可调整遗传算法参数或优化模型重新求解。

得到如下结果：

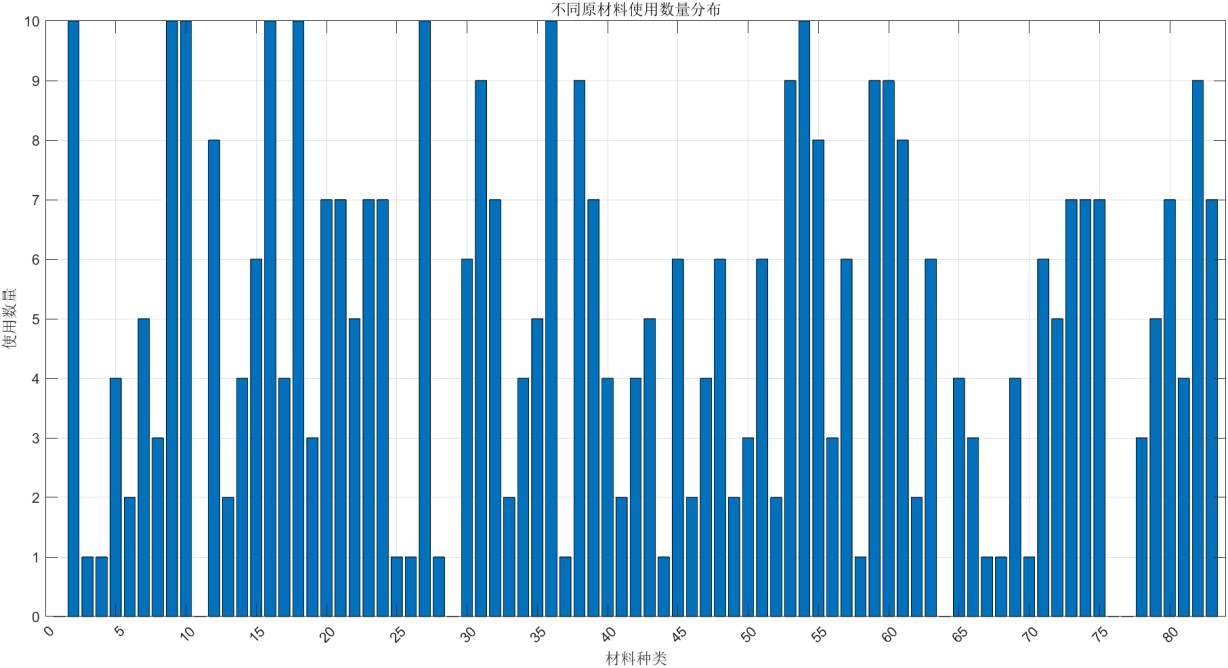


图 **8** 问题三材料最优使用情况

根据上图我们取得了材料的最优切割方案，又因为总收益已知为 161800 元和材料单价已知，最终得到总利润为 152699.92 元

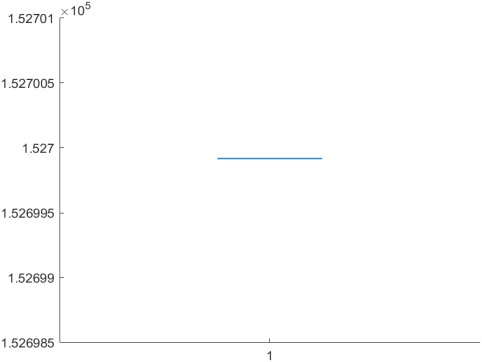


图 **9** 最优切割方案总利润

5.4 模型的改进

1. 编码方式改进：如采用动态编码，在算法运行过程中根据问题特点与进化情况调整编码方式，便于更好的适应搜索过程。
2. 初始化种群改进：增加种群多样性：采用多种群初始化方法，从不同区域采样生成初始种群，或者将种群大小扩大，如将我们上面解决问题时的种群大小由 50 提

至 100。

1. 交叉算子改进：根据个体适应度和种群的进化情况，自适应地调整交叉概率，或结合多种不同的交叉方式，如单点交叉、多点交叉、均匀交叉等，在算法中根据实际情况动态选择合适的交叉方法。
2. 变异算子改进：当种群进化停滞时，适当的增加变异率以跳出当前局部最优，反之则降低变异率以保持种群优秀基因。还可以在变异操作时，对当前个体领域内进行搜索，选择一个更优个体作为变异后的结果。
3. 终止条件改进：根据问题难度和算法运行情况，动态调整终止条件。对于困难问题可以放宽终止条件。除了达到最大迭代次数或满足一定的阈值条件外，还可以考虑种群的多样性、进化速度来盘肚腩算法是否收敛。
4. 与其他算法结合：
5. 与局部搜索算法结合：将遗传算法与局部搜索算法（如模拟退火、禁忌搜索等）相结合，利用遗传算法进行全局搜索，找到一个较好的搜索区域，然后利用局部搜索算法在该区域内进行精细搜索，以提高算法的收敛速度和求解精度。
6. 与神经网络结合：将遗传算法用于训练神经网络的权重或结构，利用遗传算法的全局搜索能力来优化神经网络的参数，提高神经网络的性能。

5.5 结果分析

* + 1. 问题一结果分析

通过 MILP 模型求解，得到以下重要指标总利润：34026 元利用率：99.84% 损失率：0.16%

分析：

高利润订单（例如酒店客房）应优先处理，以符合预期目标。

原材料的利用率较高，这主要得益于 MILP 模型精确地分配了切割方案，从而减少了剩余材料的产生；

切割过程中的损失主要源于锯口的宽度以及未能重新利用的余料。

* + 1. 问题二结果分析

引入缺陷约束后，遗传算法优化结果如下：总利润：34162 元（较问题一提高 0.4%）原材料利用率：99.4% 分析：

遗传算法通过适应度函数直接最大化利润，可能优先分配更多原材料到高单价订单（如酒店客房单价 680 元/套），尽管利用率略低，但收益提升显著。

选择低成本、高有效区域的原材料，优化成本结构；灵活调整切割方案，降低锯口损耗。

* + 1. 问题三结果分析

针对附件中大规模订单数据，优化结果如下：总成本：161800 元总利润：152699.92 元



图 **10** 问题三部分材料使用数量（详见附录）

对比分析：利润随订单量增加呈线性增长，但利用率下降，因缺陷和锯口损耗累积；

遗传算法在复杂约束下仍保持较高搜索效率，但需权衡计算时间与解的质量。

# 六、 模型的优缺点与改进

6.1 模型的优点

6.1.1 MILP 即混合整数线性规划

MILP 即混合整数线性规划是一种数学优化技术，其优点如下：

1.精确性和灵活性：能在给定的约束条件下，找到全局最优解，这对于要求精确决策的问题至关重要，如资源分配、生产计划等场景，可确保方案在理论上是最优的。可灵活处理各种线性约束条件，包括等式约束和不等式约束，能适应多种实际问题的建模需求，如产量限制、预算约束、时间窗口等。

2.可解释性：模型的结构和决策变量具有明确的物理意义，约束条件和目标函数基于线性关系，易于理解和解释，便于决策者与模型构建者进行沟通，从而更好地根据实际情况调整模型和决策。

3.成熟的求解算法及多目标优化能力：经过多年发展，已有许多成熟的求解算法和软件，如分支定界法、割平面法等，能高效地求解大规模的 MILP 问题，为实际应用提供了有力的技术支持。可通过设置多个目标函数，并根据实际需求设置优先级或权重，实现多目标的优化，帮助决策者在不同目标之间进行权衡，找到满足综合需求的最优解。

## 6.1.2 遗传算法

遗传算法是一种基于自然选择和基因遗传学原理的优化算法，具有以下优点：

1.全局搜索能力：遗传算法从一组随机生成的初始解开始搜索，通过交叉、变异等操作在解空间中进行广泛的探索，有较大机会找到全局最优解，而不像一些传统算法容易陷入局部最优。

2.对问题的适应性强：它不需要对问题有过多的先验知识，只需要定义好适应度函数来评价解的优劣即可。对于复杂的、难以用传统数学方法精确建模的问题，如多目标优化、非线性优化等，遗传算法能够有效地处理。

3.并行性：遗传算法天然具有并行性，可以同时处理多个解，即在搜索过程中同时对多个个体进行操作，这使得它非常适合在并行计算环境下运行，能够大大提高搜索效率，尤其对于大规模问题的求解具有很大优势。

6.2 模型的缺点

计算复杂度较高，当生产问题规模较大时，遗传算法需处理众多个体及复杂的基因编码，计算量显著增加，导致算法运行时间延长。

MILP 问题属于 NP-hard 问题，随着问题规模的扩大，例如产品种类的增多、生产设备数量的增加、订单数量的上升等，求解所需时间将呈指数级增长。对于一些实时性要求较高的生产优化场景，如紧急订单插入后的调度调整，MILP 可能需要较长时间进行求解。

6.3 模型的改进

6.3.1 遗传算法改进方案

1. 混合局部搜索机制

针对遗传算法易陷入局部最优，尤其在处理复杂约束，如原料缺陷位置时，可能过早收敛的不足，在遗传中嵌入局部搜索（如模拟退火或 2-opt 算法），对每代最优个体进行精细化调整。从而在算法找到的近似最优解附近进一步挖掘更优解，适用于原料缺陷分布复杂、切割方案需避开多个小缺陷的场景。

1. 自适应参数调整

对于原始方案中固定的交叉概率 Pc 和变异概率 Pm 难以适应不同进化阶段需求的不足，可根据可行解多样性动态调整参数：当可行解多样性高时，降低 Pc 和 Pm 以保留优良个体；当多样性低时，增加 Pc 和 Pm 以跳出局部最优。可达到无需人工调参，减少试错成本的效果，提高找到全局最优的概率。在对比实验多订单、多原料场景中，自适应参数的遗传算法比固定参数方案平均提升 10%-15% 的原料利用率。

## 6.3.2 **MILP** 改进方案

1. 模型分解与并行求解

对于 MILP 求解大规模问题时计算时间过长的问题，按订单类型或原料批次将原始问题分解为若干子题，并行求解后整合结果。改进后可将求解时间从小时级降至分钟级，甚至扩展到万级订单和原料，突破传统方案的规模限制。

1. 鲁棒优化建模

考虑到原方案中假设原料缺陷位置完全确定，而实际生产中存在测量误差，可引入松弛变量表示缺陷位置的不确定性，在目标函数中平衡利润与鲁棒性。由此生成的方案在缺陷位置波动时仍保持可行性，减少生产中断风险。

# 七、 参考文献

[1]任长清,武子棋,闫杰,等.基于遗传算法的被动式木窗材下料优化[J].森林工程,2025,4

1(03):595-602.

[2]贾芮博,杨凯,王天明,等.基于遗传算法的 5B70 铝合金铣削加工多目标参数优化[J]. 工具技术,2025,59(04):22-29.

[3]温帅方,宋青浩,孙丹,等.基于遗传算法的 W 形封严环结构优化[J/OL].航空动力学报,

1-9[2025-05-09].https://doi.org/10.13224/j.cnki.jasp.20240811.

[4]卢艺,卢苑,梁俊文,等.含抽水蓄能电网安全约束机组组合问题的混合整数线性规划算法[J].电力系统保护与控制,2019,47(03):39-47.

[5]胡明琼.混合整数线性规划近优解的一种解法[J].武汉冶金科技大学学报,1998,(01):1

19-122.

[6]孔海宁.钢铁企业副产煤气系统优化调度研究[D].天津大学,2011.

[7]高倩,杨知方,李文沅.电力系统混合整数线性规划问题的运筹决策关键技术综述与

展望[J].电工技术学报,2024,39(11):3291-3307.DOI:10.19595/j.cnki.1000-6753.tces.23047

8.

附录

|  |
| --- |
| 第一问代码 |
| % 订单数据：[订单量, 宽度目标, 高度目标, 单价] orders = [  10, 1.6, 2.2, 480;  20, 1.8, 2.4, 680;  20, 1.7, 2.3, 550;  15, 1.5, 2.0, 420;  ];    % 原材料数据：[长度, 单价]  materials = [  5.5, 18; 6.2, 22;  7.8, 28;  ];  num\_orders = size(orders, 1); num\_materials = size(materials, 1); num\_segment\_types = 2; % 1: 宽度，2: 高度  % 变量索引：y1-3, x1\_1\_1, x1\_1\_2,... x3\_4\_2, z1-4 total\_vars = num\_materials + num\_materials \* num\_orders \*  num\_segment\_types + num\_orders; y\_indices = 1:num\_materials; x\_base = num\_materials + 1;  z\_base = x\_base + num\_materials \* num\_orders \*  num\_segment\_types;    % 目标函数系数  f = zeros(total\_vars, 1);  f(z\_base:z\_base + num\_orders - 1) = -orders(:, 4); % 收益部分  f(y\_indices) = materials(:, 2); % 成本部分  % 整数变量标记（所有变量均为整数）  intcon = 1:total\_vars; |

|  |
| --- |
| % 约束矩阵  A = []; b = [];    % 1. 订单段数约束：sum(x\_{r,o,d}) >= 2z\_o  for o = 1:num\_orders  % 宽度段约束  row = zeros(1, total\_vars); for r = 1:num\_materials pos = x\_base + (r-1)\*num\_orders\*2 + (o-1)\*2 + 0; row(pos) = -1; % -sum(x) + 2z <= 0 end row(z\_base + o - 1) = 2; A = [A; row]; b = [b; 0];  % 高度段约束  row = zeros(1, total\_vars); for r = 1:num\_materials pos = x\_base + (r-1)\*num\_orders\*2 + (o-1)\*2 + 1; row(pos) = -1; end row(z\_base + o - 1) = 2; A = [A; row]; b = [b; 0]; end    % 2. z\_o <= 订单量 for o = 1:num\_orders row = zeros(1, total\_vars); row(z\_base + o - 1) = 1; A = [A; row]; b = [b; orders(o, 1)]; end    % 3. 原材料长度约束  for r = 1:num\_materials L = materials(r, 1); row = zeros(1, total\_vars);  row(y\_indices(r)) = -(L + 0.005); % 右侧项系数 for o = 1:num\_orders  % 宽度段  s\_width = orders(o, 2); pos = x\_base + (r-1)\*num\_orders\*2 + (o-1)\*2 + 0; |

|  |
| --- |
| row(pos) = s\_width + 0.005;  % 高度段  s\_height = orders(o, 3); pos = x\_base + (r-1)\*num\_orders\*2 + (o-1)\*2 + 1; row(pos) = s\_height + 0.005; end  A = [A; row]; b = [b; 0]; end    % 变量边界  lb = zeros(total\_vars, 1); ub = inf(total\_vars, 1);  ub(z\_base:z\_base + num\_orders - 1) = orders(:, 1); % z\_o <= 订单量    % 求解MILP  options = optimoptions('intlinprog', 'Display', 'off');  [sol, fval, exitflag] = intlinprog(f, intcon, A, b, [], [], lb, ub, options);    % 结果解析  if exitflag == 1 y = sol(y\_indices); z = sol(z\_base:z\_base + num\_orders - 1); total\_profit = -fval;  % 计算利用率与损失率  total\_used\_length = sum(y .\* materials(:, 1)); total\_segments = sum(sol(x\_base:end - num\_orders)); total\_cut\_loss = 0.005 \* (total\_segments - sum(y)); utilization = (total\_used\_length - total\_cut\_loss) /  total\_used\_length; loss\_rate = total\_cut\_loss / total\_used\_length;  disp('===== 最优方案 =====');  disp(['总利润: ', num2str(total\_profit), ' 元']);  disp(['原材料利用率: ', num2str(utilization \* 100, '%.2f'),  '%']); disp(['切割损失率: ', num2str(loss\_rate \* 100, '%.2f'),  '%']); else disp('无可行解'); end if exitflag == 1 |

|  |
| --- |
| %% 数据准备 % 原材料信息  material\_labels = cellfun(@(x)sprintf('%.1fm\n%d元/根',  materials(x,1), materials(x,2)),...  num2cell(1:num\_materials), 'UniformOutput', false); y\_values = sol(y\_indices);  % 订单信息  order\_labels = cellfun(@(x)sprintf('订单%d\n[%.1f×%.1f]', x,  orders(x,2), orders(x,3)),...  num2cell(1:num\_orders), 'UniformOutput', false); z\_values = sol(z\_base:z\_base + num\_orders - 1); order\_profits = z\_values .\* orders(:,4);  % 利润构成  profit\_components = [sum(order\_profits), -  sum(y\_values.\*materials(:,2))]; profit\_labels = {'订单收入','原材料成本'};  %% 可视化设置  colors = lines(7); % 使用MATLAB默认颜色  fig = figure('Position', [100 100 1200 800], 'Color', 'w');  %% 子图1：原材料采购分析  subplot(2,2,1) bh = barh(y\_values, 'FaceColor', 'flat'); bh.CData = colors(1:num\_materials,:); set(gca, 'YTickLabel', material\_labels,  'YDir','reverse',...  'FontSize',9, 'XGrid','on') xlabel('采购数量（根）') title('原材料采购方案') annotation('textbox', [0.15 0.75 0.3 0.05], 'String',... sprintf('总采购成本：%d元', sum(y\_values.\*materials(:,2))),...  'FitBoxToText','on', 'EdgeColor','none');  %% 子图2：订单完成情况  subplot(2,2,2) bg = bar([orders(:,1), z\_values], 'grouped'); set(bg(1), 'DisplayName','订单需求', 'FaceColor',  colors(5,:)) set(bg(2), 'DisplayName','实际完成', 'FaceColor',  colors(6,:)) set(gca, 'XTickLabel', order\_labels, 'FontSize',9,  'GridLineStyle',':') ylabel('完成数量') title('订单完成情况对比')  legend('show', 'Location','northwest') |
| %% 子图3：利润构成分析  subplot(2,2,3) pie(order\_profits, order\_labels);  title({'各订单利润贡献比例', sprintf('总利润：%d元', total\_profit)}) colormap(summer(num\_orders))  %% 子图4：材料利用分析  subplot(2,2,4) loss\_rates = [utilization, loss\_rate]; h = pie(loss\_rates, {'有效利用率', '切割损失'});  hp = findobj(h, 'Type', 'patch'); set(hp(1), 'FaceColor', colors(3,:), 'EdgeColor','none') set(hp(2), 'FaceColor', colors(4,:), 'EdgeColor','none') title({'材料利用率分析', sprintf('总利用率：%.2f%%', utilization\*100)})  %% 添加整体标题  annotation(fig, 'textbox', [0.4 0.94 0.2 0.05],...  'String', '切割方案优化结果可视化',...  'EdgeColor', 'none', 'HorizontalAlignment','center',...  'FontSize',14, 'FontWeight','bold') else disp('无可行解');  End |

|  |
| --- |
| 第二问代码 |
| % 定义订单数据（订单量, 宽度, 高度, 单价） orders = [  10, 1.6, 2.2, 480;  20, 1.8, 2.4, 680;  20, 1.7, 2.3, 550;  15, 1.5, 2.0, 420  ];  % 定义原材料数据（编号, 长度, 单价）  materials = [1,5.5,18;  2,6.2,22;  3,7.8,28;  ];  % 定义缺陷数据（原材料编号, 起始位置, 缺陷长度）  defects = [1,1.0,0.03;  1,2.5,0.04; |

|  |
| --- |
| 2,0.5,0.02;  2,1.8,0.05;  3,3.0,0.03;  ];  % 锯口宽度  cutWidth = 0.005;  % 遗传算法参数  popSize = 50; % 种群大小（减小以加快调试） genNum = 50; % 遗传代数（减小以加快调试） crossRate = 0.8; % 交叉率  mutateRate = 0.1; % 变异率  % 数据预处理  orderNum = size(orders, 1); materialNum = size(materials, 1); matIdx = materials(:, 1); % 原材料编号列表  % 初始化种群：[订单分配(4x83) 原材料使用(83)]  pop = zeros(popSize, orderNum\*materialNum + materialNum); for i = 1:popSize  % 订单分配：每个订单在各原材料上的生产数量（不超过订单量） for j = 1:orderNum for k = 1:materialNum pop(i, (j-1)\*materialNum + k) = randi([0, orders(j, 1)]); end end  % 原材料使用数量（初始化为0~10）  for k = 1:materialNum pop(i, orderNum\*materialNum + k) = randi([0, 10]); end end    % 遗传算法主循环  for gen = 1:genNum fprintf('Generation %d...\n', gen); fitness = zeros(popSize, 1); for i = 1:popSize  % 解析个体  orderAssign = reshape(pop(i, 1:orderNum\*materialNum),  orderNum, materialNum); matCounts = pop(i, orderNum\*materialNum+1:end);  % 检查订单总量是否超过需求  if any(sum(orderAssign, 2) > orders(:, 1)) fitness(i) = -Inf; |

|  |
| --- |
| continue; end  totalRevenue = sum(sum(orderAssign .\* repmat(orders(:,4),  1, materialNum))); totalCost = sum(matCounts .\* materials(:,3));  % 可行性检查：每个原材料的切割是否可行  feasible = true; for m = 1:materialNum matLength = materials(m, 2); numUsed = matCounts(m); if numUsed == 0  continue; % 未使用该原材料，跳过检查 end  % 获取该原材料的缺陷区间（转换为绝对位置）  matDefect = defects(defects(:,1)==matIdx(m), 2:3); %  [start, length]  % 收集该原材料上的所有切割需求：每个窗框需要2宽+2高，共4段 cuts = []; for j = 1:orderNum numFrames = orderAssign(j, m); if numFrames > 0 width = orders(j, 2);  height = orders(j, 3);  % 每个窗框需要2个宽度和2个高度段  cuts = [cuts; repmat(width, numFrames, 2); repmat(height,  numFrames, 2)]; end end  cuts = cuts(:); % 转换为列向量 numCuts = length(cuts); if numCuts == 0  continue; % 该原材料未切割任何段 end  % 计算总切割长度（含锯口）  totalCutLength = sum(cuts) + (numCuts - 1)\*cutWidth; if totalCutLength > matLength feasible = false; break;  end  % 排序切割段（优化排列，此处简单排序）  [~, sortIdx] = sort(cuts, 'descend'); % 降序排列以减少间隙 sortedCuts = cuts(sortIdx);  % 模拟切割位置，检查缺陷避让  currentPos = 0; |

|  |
| --- |
| for c = 1:numCuts segmentLen = sortedCuts(c);  endPos = currentPos + segmentLen;  % 检查是否与缺陷重叠  for d = 1:size(matDefect, 1) dStart = matDefect(d, 1); dEnd = dStart + matDefect(d, 2); if ~(endPos <= dStart || currentPos >= dEnd) feasible = false; break; end end if ~feasible break; end  currentPos = endPos + cutWidth; % 锯口宽度加在段后（除最后一个  段）  if currentPos > matLength feasible = false; break; end end if ~feasible break; end end if feasible fitness(i) = totalRevenue - totalCost; else fitness(i) = -Inf; end end  % 选择操作（轮盘赌，处理全无效解）  validIdx = find(fitness ~= -Inf); if isempty(validIdx)  fprintf('Warning: All solutions are infeasible in  generation %d\n', gen); break; end validFitness = fitness(validIdx); totalFitness = sum(validFitness); selectionProbs = validFitness / totalFitness; newPop = zeros(popSize, size(pop, 2)); for i = 1:popSize |

|  |
| --- |
| % 轮盘赌选择  r = rand; cumProb = cumsum(selectionProbs); selIdx = validIdx(find(cumProb >= r, 1)); newPop(i, :) = pop(selIdx, :);  % 交叉操作（单点交叉）  if i < popSize && rand < crossRate  crossPoint = randi([1, orderNum\*materialNum + materialNum -  1]); temp = newPop(i, crossPoint:end); newPop(i, crossPoint:end) = newPop(i+1, crossPoint:end); newPop(i+1, crossPoint:end) = temp; end  % 变异操作  for j = 1:size(newPop, 2) if rand < mutateRate  if j <= orderNum\*materialNum % 订单分配部分  % 确定订单和原材料索引  orderJ = floor((j-1)/materialNum) + 1; maxOrder = orders(orderJ, 1); newPop(i, j) = randi([0, maxOrder]);  else % 原材料使用数量部分  newPop(i, j) = randi([0, 20]); % 合理范围 end end end end pop = newPop; end    % 寻找最优解  [bestFitness, bestIdx] = max(fitness); bestSolution = pop(bestIdx, :);  orderAssign = reshape(bestSolution(1:orderNum\*materialNum),  orderNum, materialNum); matCounts = bestSolution(orderNum\*materialNum+1:end);    % 输出结果  fprintf('\n================= 最优解 =================\n'); fprintf('最大利润: %.2f 元\n', bestFitness);  fprintf('订单分配（行=订单，列=原材料，值=生产数量）:\n'); disp(orderAssign);  fprintf('原材料使用数量（索引对应materials第一列）:\n'); |

|  |
| --- |
| disp([matIdx, matCounts']);    函数  Li = [5.5; 6.2; 7.8]; % 原材料长度 yi = [1; 8; 10]; % 各材料使用数量 Sjk = { % 各材料的切割段（cell数组）  [1,1.47,2.99;], % 材料1的切割段  [ 0.5,1.28,4.35;], % 材料2的切割段  [3,4.77] % 材料3的切割段  };    % 调用函数  [R\_L, R\_u] = calculateCuttingMetrics(Li, yi, Sjk);  % 原调用代码后添加可视化  [RL, Ru] = calculateCuttingMetrics(Li, yi, Sjk); visualizeCuttingMetrics(Li, yi, Sjk, Ru, RL);    function [R\_L, R\_u] = calculateCuttingMetrics(Li, yi, Sjk,  cutWidth)  % 计算切割损失率和利用率  % 输入参数：  % Li: 原材料长度数组（列向量）  % yi: 原材料使用数量数组（列向量）  % Sjk: 切割段长度cell数组（每个cell对应一种原材料的切割段）  % cutWidth: 锯口宽度（默认0.005）  % 输出：  % R\_L: 总损失率  % R\_u: 总利用率    % 参数验证  if nargin < 4  cutWidth = 0.005; % 默认锯口宽度 end    % 确保输入为列向量  Li = Li(:); yi = yi(:);  % 计算总材料长度  totalMaterial = sum(Li .\* yi); |

|  |
| --- |
| % 初始化总有效切割长度  totalEffectiveCut = 0;  % 遍历每种原材料  for m = 1:length(Li)  % 当前材料的切割段  segments = Sjk{m};  % 跳过无切割段的情况  if isempty(segments) continue  end  % 计算当前材料的切割参数  nCuts = length(segments); % 切割段数量 totalSegments = sum(segments); % 纯切割段总长度  kerfTotal = (nCuts - 1) \* cutWidth; % 锯口总长度  % 累计有效切割长度（考虑使用次数）  totalEffectiveCut = totalEffectiveCut + ... (totalSegments + kerfTotal) \* yi(m); end    % 计算结果  R\_u = totalEffectiveCut / totalMaterial; % 利用率 R\_L = 1 - R\_u; % 损失率  % 显示结果  fprintf('========= 切割性能分析 =========\n'); fprintf('总材料长度: %.3f 米\n', totalMaterial); fprintf('有效切割长度: %.3f 米\n', totalEffectiveCut); fprintf('利用率 R\_u = %.2f%%\n', R\_u\*100); fprintf('损失率 R\_L = %.2f%%\n', R\_L\*100); fprintf('===============================\n'); end  % 示例数据  function visualizeCuttingMetrics(Li, yi, Sjk, R\_u, R\_L)  % 创建可视化画布  figure('Color','w','Position',[100 100 1200 800]);  % ========= 子图1: 各材料利用率对比 ========= subplot(2,3,[1 4]); hold on; |

|  |
| --- |
| % 计算各材料的独立利用率  materialUtilization = zeros(size(Li)); for m = 1:length(Li) if yi(m) == 0 materialUtilization(m) = 0; continue; end totalSeg = sum(Sjk{m}) + (length(Sjk{m})-1)\*0.005; materialUtilization(m) = totalSeg / Li(m); end    % 绘制堆叠柱状图  barData = [materialUtilization, 1-materialUtilization] .\*  100; hBar = bar(barData, 'stacked');  % 样式设置  set(hBar(1), 'FaceColor', [0.2 0.6 0.3],  'EdgeColor','none'); set(hBar(2), 'FaceColor', [0.8 0.3 0.2],  'EdgeColor','none'); xticks(1:length(Li));  xticklabels(arrayfun(@(x)sprintf('材  料%d\n%.1fm',x,Li(x)),1:length(Li),'uni',0)); ylabel('比例 (%)'); title('各材料利用率分解'); legend({'有效利用率','损耗率  '},'Location','northeastoutside');    % ========= 子图2: 切割段分布可视化 ========= subplot(2,3,2); hold on;    % 绘制所有切割段  colorOrder = lines(length(Li)); for m = 1:length(Li) if yi(m) == 0, continue; end segs = Sjk{m}; for s = 1:length(segs) rectangle('Position',[0 s-0.4 segs(s) 0.8],...  'FaceColor',colorOrder(m,:),...  'EdgeColor','none'); end |

|  |
| --- |
| text(max(segs)+0.1, s-0.1, sprintf('材料%d',m),...  'Color',colorOrder(m,:),'FontSize',8); end    % 图形设置  xlabel('切割长度 (米)'); yticks([]);  title('切割段长度分布');  xlim([0 max(cellfun(@max,Sjk))]);  % ========= 子图3: 整体指标仪表盘 ========= subplot(2,3,3); hold on;    % 绘制仪表盘  theta = linspace(0, pi, 100); plot(cos(theta), sin(theta), 'k', 'LineWidth',2); fill([cos(theta) 0], [sin(theta) 0], [0.9 0.95 1],  'EdgeColor','none');    % 绘制指针  pointerAngle = pi\*(1 - R\_u);  plot([0 cos(pointerAngle)], [0 sin(pointerAngle)], 'r',  'LineWidth',2);    % 添加标注  text(0, 0.2, sprintf('%.1f%%',R\_u\*100),... 'HorizontalAlignment','center','FontSize',20); text(-0.9,0.9,{'材料利用率'},'FontSize',12); axis equal off  % ========= 子图4: 材料使用量分布 ========= subplot(2,3,5:6);  % 准备数据  usageLength = Li .\* yi;  [~,sortIdx] = sort(usageLength, 'descend'); labels = arrayfun(@(x)sprintf('材料%d\n%.1f米  ',x,Li(x)),sortIdx,'uni',0);    % 绘制水平条图 |
| barh(1:length(Li), usageLength(sortIdx), 0.8,... 'FaceColor',[0.2 0.4 0.6], 'EdgeColor','none');  % 样式设置  set(gca,'YTick',1:length(Li),'YTickLabel',labels);  xlabel('总消耗长度 (米)'); title('材料消耗量排名');  grid on;    % ========= 添加全局标题 =========  sgtitle(sprintf('切割工艺分析报告 | 总利用率 %.1f%% | 总损耗  率 %.1f%%',...  R\_u\*100, R\_L\*100), 'FontSize',14); end |

|  |
| --- |
| 第三问代码 |
| % 定义订单数据（订单量, 宽度, 高度, 单价） orders = [  120, 1.6, 2.2, 480;  80, 1.8, 2.4, 680;  60, 1.7, 2.3, 550;  40, 1.5, 2.0, 420  ];  % 定义原材料数据（编号, 长度, 单价）  materials = [  1,5.5,17; 2,5.5,17.33; 3,6.2,20.59; 4,7,24.41; 5,7,24.05;  6,5.8,17.33; 7,6.5,22; 8,7.5,24.77; 9,6,19.83;  10,8.2,27.64;  11,6.8,23.32; 12,6.8,23.69; 13,5.6,17.66; 14,7.3,24.77;  15,6.1,19.83;  16,8,27.64; 17,5.9,18; 18,6.3,21.27; 19,7.8,26.57;  20,6.7,22.91;  21,5.4,16.68; 22,7.4,25.85; 23,6.9,22.91; 24,8.1,27.64;  25,7.6,26.2;  26,5.7,17.33; 27,6.4,22; 28,8.3,28.72; 29,6,18;  30,7.9,27.64;  31,5.5,16.68; 32,6.2,20.95; 33,7.1,23.69; 34,6.8,23.69; 35,5.8,17.66;  36,7.3,25.13; 37,6.9,24.05; 38,7.5,25.49; 39,5.6,17.33; 40,6.4,20.95; |

|  |
| --- |
| 41,6.6,22; 42,7,24.05; 43,8,28; 44,5.9,19.83; 45,7.7,25.85;  46,6.5,22.41; 47,7.2,24.41; 48,6.1,20.59; 49,5.4,17; 50,8.2,27.64;  51,6.7,22.91; 52,7.8,26.57; 53,5.5,16.36; 54,6.6,22;  55,7,24.05;  56,5.8,17.66; 57,8,28; 58,6.9,23.69; 59,7.2,24.41;  60,6.3,21.27;  61,8.1,27.27; 62,5.6,17; 63,7.4,25.49; 64,6.1,19.83;  65,6.8,23.69;  66,7.3,25.13; 67,5.7,17.33; 68,7,24.41; 69,6.5,22; 70,5.8,17.33;  71,8.2,28; 72,7.5,25.85; 73,6,19.41; 74,7.7,25.85;  75,6.6,22;  76,6.2,21.27; 77,7.3,25.13; 78,5.5,16.68; 79,7,23.32;  80,6.9,23.69;  81,8,27.27; 82,7.4,25.85; 83,6.3,20.59  ];  % 定义缺陷数据（原材料编号, 起始位置, 缺陷长度） defects = [  1,1,0.3; 1,3,0.2; 2,2,0.4; 3,1.5,0.2; 3,4,0.3; 4,1.2,0.5;  5,2.3,0.3; 6,1,0.6; 7,2.8,0.4; 8,1.3,0.5; 9,2.1,0.3;  9,5,0.2;  10,1.1,0.2;11,3.1,0.4; 12,1.7,0.5; 13,2.5,0.3; 14,3,0.4;  15,1.9,0.3;  16,1.2,0.4;17,2.4,0.3; 18,0.8,0.3; 19,3,0.2; 20,2,0.5;  21,2.2,0.4;  22,1.6,0.3;23,2.7,0.4; 24,1.8,0.2; 25,0.9,0.3; 26,1.1,0.5; 27,2.9,0.2;  28,1.3,0.4;29,3.2,0.3; 30,2.3,0.5; 31,1.9,0.2; 32,2.5,0.4;  33,3,0.3;  34,2,0.2; 35,1.6,0.4; 36,1,0.3; 37,2.2,0.5; 38,2,0.4;  39,3.1,0.3;  40,1.5,0.2;41,1.9,0.3; 42,2.6,0.5; 43,1.1,0.2; 44,2.7,0.4;  45,3,0.3;  46,1.5,0.2;47,2,0.5; 48,2.9,0.3; 49,1.2,0.4; 50,2.1,0.5;  51,3.2,0.4;  52,1.7,0.3;53,1,0.4; 54,2.3,0.2; 55,2.5,0.3; 56,3,0.4;  57,2.4,0.3;  58,1.9,0.5;59,3.1,0.4; 60,2,0.3; 61,1.8,0.5; 62,2.1,0.2; 63,1.4,0.3;  64,2.6,0.4;65,2.5,0.2; 66,3,0.3; 67,1.2,0.5; 68,2.7,0.4; 69,3,0.3;  70,1.1,0.3;71,2.3,0.5; 72,2,0.4; 73,3.1,0.2; 74,1.5,0.3; 75,2.4,0.4; |

|  |
| --- |
| 76,1.8,0.5;77,2.5,0.3; 78,2.6,0.4; 79,1.7,0.2; 80,3,0.5  ];  % 锯口宽度  cutWidth = 0.005;  % 遗传算法参数  popSize = 50; % 种群大小（减小以加快调试） genNum = 50; % 遗传代数（减小以加快调试） crossRate = 0.8; % 交叉率 mutateRate = 0.1; % 变异率  % 数据预处理  orderNum = size(orders, 1); materialNum = size(materials, 1); matIdx = materials(:, 1); % 原材料编号列表  % 初始化种群：[订单分配(4x83) 原材料使用(83)]  pop = zeros(popSize, orderNum\*materialNum + materialNum); for i = 1:popSize  % 订单分配：每个订单在各原材料上的生产数量（不超过订单量） for j = 1:orderNum for k = 1:materialNum pop(i, (j-1)\*materialNum + k) = randi([0, orders(j, 1)]); end end  % 原材料使用数量（初始化为0~10）  for k = 1:materialNum pop(i, orderNum\*materialNum + k) = randi([0, 10]); end end    % 遗传算法主循环  for gen = 1:genNum fprintf('Generation %d...\n', gen); fitness = zeros(popSize, 1); for i = 1:popSize  % 解析个体  orderAssign = reshape(pop(i, 1:orderNum\*materialNum),  orderNum, materialNum); matCounts = pop(i, orderNum\*materialNum+1:end);  % 检查订单总量是否超过需求  if any(sum(orderAssign, 2) > orders(:, 1)) fitness(i) = -Inf; continue; |

|  |
| --- |
| end  totalRevenue = sum(sum(orderAssign .\* repmat(orders(:,4),  1, materialNum))); totalCost = sum(matCounts .\* materials(:,3));  % 可行性检查：每个原材料的切割是否可行  feasible = true; for m = 1:materialNum matLength = materials(m, 2); numUsed = matCounts(m); if numUsed == 0  continue; % 未使用该原材料，跳过检查 end  % 获取该原材料的缺陷区间（转换为绝对位置）  matDefect = defects(defects(:,1)==matIdx(m), 2:3); %  [start, length]  % 收集该原材料上的所有切割需求：每个窗框需要2宽+2高，共4段 cuts = []; for j = 1:orderNum numFrames = orderAssign(j, m); if numFrames > 0 width = orders(j, 2);  height = orders(j, 3);  % 每个窗框需要2个宽度和2个高度段  cuts = [cuts; repmat(width, numFrames, 2); repmat(height,  numFrames, 2)]; end end  cuts = cuts(:); % 转换为列向量 numCuts = length(cuts); if numCuts == 0  continue; % 该原材料未切割任何段 end  % 计算总切割长度（含锯口）  totalCutLength = sum(cuts) + (numCuts - 1)\*cutWidth; if totalCutLength > matLength feasible = false; break;  end  % 排序切割段（优化排列，此处简单排序）  [~, sortIdx] = sort(cuts, 'descend'); % 降序排列以减少间隙 sortedCuts = cuts(sortIdx);  % 模拟切割位置，检查缺陷避让  currentPos = 0; for c = 1:numCuts |

|  |
| --- |
| segmentLen = sortedCuts(c);  endPos = currentPos + segmentLen;  % 检查是否与缺陷重叠  for d = 1:size(matDefect, 1) dStart = matDefect(d, 1); dEnd = dStart + matDefect(d, 2); if ~(endPos <= dStart || currentPos >= dEnd) feasible = false; break; end end if ~feasible break; end  currentPos = endPos + cutWidth; % 锯口宽度加在段后（除最后一个  段）  if currentPos > matLength feasible = false; break; end end if ~feasible break; end end if feasible fitness(i) = totalRevenue - totalCost; else fitness(i) = -Inf; end end  % 选择操作（轮盘赌，处理全无效解）  validIdx = find(fitness ~= -Inf); if isempty(validIdx)  fprintf('Warning: All solutions are infeasible in  generation %d\n', gen); break; end validFitness = fitness(validIdx); totalFitness = sum(validFitness); selectionProbs = validFitness / totalFitness; newPop = zeros(popSize, size(pop, 2)); for i = 1:popSize  % 轮盘赌选择 |

|  |
| --- |
| r = rand; cumProb = cumsum(selectionProbs); selIdx = validIdx(find(cumProb >= r, 1)); newPop(i, :) = pop(selIdx, :);  % 交叉操作（单点交叉）  if i < popSize && rand < crossRate  crossPoint = randi([1, orderNum\*materialNum + materialNum -  1]); temp = newPop(i, crossPoint:end); newPop(i, crossPoint:end) = newPop(i+1, crossPoint:end); newPop(i+1, crossPoint:end) = temp; end  % 变异操作  for j = 1:size(newPop, 2) if rand < mutateRate  if j <= orderNum\*materialNum % 订单分配部分  % 确定订单和原材料索引  orderJ = floor((j-1)/materialNum) + 1; maxOrder = orders(orderJ, 1); newPop(i, j) = randi([0, maxOrder]);  else % 原材料使用数量部分  newPop(i, j) = randi([0, 20]); % 合理范围 end end end end pop = newPop; end    % 寻找最优解  [bestFitness, bestIdx] = max(fitness); bestSolution = pop(bestIdx, :);  orderAssign = reshape(bestSolution(1:orderNum\*materialNum),  orderNum, materialNum); matCounts = bestSolution(orderNum\*materialNum+1:end);    % 输出结果  fprintf('\n================= 最优解 =================\n'); fprintf('最大利润: %.2f 元\n', bestFitness);  fprintf('订单分配（行=订单，列=原材料，值=生产数量）:\n'); disp(orderAssign);  fprintf('原材料使用数量（索引对应materials第一列）:\n'); disp([matIdx, matCounts']); |