队伍编号	21170450026
题号	D

钢材切割下料问题

摘要

根据订单的要求，工厂要用 10 种不同的原材料切割成 15 种不同规格的订单。故该原料切割问题为典型的优化模型。在切割过程中，要求使用的原料张数尽可能 少，同时提高成才率。故针对不同的订单需求及切割条件要求，得出 4 种相应的最佳切割方案。

问题一要求使用张数最少，同时要提高成材率。在满足五种卷料订单的基础上，尽可能的使用面积较大的原材料。故先将原料面积按降序排列，使用线性规划找出最佳宽度的材料匹配，先进行进行较大面积的原料切割，以此来减少使用张数，并求得最少原材料张数为 22 张。

问题二在问题一的要求上加上了 10 种板料的订单。根据板料的规格大小来看，板料长度普遍较短，可以充分利用切割卷料后的余料进行相应的切割，从而达到减少使用张数，并求得最少原材料张数为 22 张。

问题三在原有要求上增加了减少人为操作次数的要求，为减少排刀次数和小机器切割次数，要在同一张原料切割出尽可能多的相同订单，并直接将余料入库回收，不做二次切割，按照减少使用张数和减少人为操作进行整体线性规划。从而找出最佳切割方案，求得最少认为操作次数为 42 次。

问题四指出部分订单在长度上存在不同的浮动比例，使问题更加接近实际操作情况。针对不同订单的浮动比例，充分考虑所有的浮动情况太过复杂。利用 random 函数生成随机的浮动比例，将生成的浮动比例带入到数学模型中进行相应计算，得出最佳切割方案。其余范围内变化的浮动比例况均可类比此种方法得出切割方案。

关键词：优化模型、整体线性规划、最大收益、排刀组合

目录

一、问题重述

问题背景：

钢材制造业中往往会涉及到钢材的切割，钢材切割的剪切过程是在剪切台上完成的，钢材依次经过切头肩和圆盘剪进行切割。圆盘剪对纵向运动的原材料进行切割，并且剪切前需要进行排刀来满足订单的需要。而切头剪则实现将整个钢板横向完全切断，若不满足一刀切的要求则无法用切头剪进行切割。对于切割完原材料的剩余部分若满足余料要求则可以回收重新利用。现有 10 种长、宽、数量不等的原材料，针对 15 种订单要求需要解决以下问题：

问题一：

利用现有的 10 种原料，满足 5 种卷料的订单方案，在不考虑浮动比例的情况下，寻找一种切割方案，使所需的原材料张数最少，并且尽可能的提高总的成材率。

问题二：

利用现有的所有 10 种原料，满足所有的订单方案，同样在不考虑浮动比例的情况下，寻找切割方案，使所需的原材料张数最少，并且尽可能的提高总的成材率。

问题三：

由于圆盘剪的每次排刀都需要人力手动更换刀在排刀架上的位置，并且被转移到小机器上进行的切割操作也需要人为进行。利用所有原料，满足所有订单方案，在不考虑浮动比例的基础上，寻找切割方案，使所需的原材料张数最少，尽量减少换刀数和在小机器上切割数，并尽量提高总成材率。

问题四：

在问题二的基础上，实际情况中订单会指定浮动比例，使得交付的订单长度可以在原有要求的基础上存在上下的误差浮动。则在考虑浮动比例的情况下，寻找切割方案，使所需的原材料张数最少，并且尽可能的提高总的成材率。

二、问题分析

2.1 问题一的分析

问题一要在给出的所有原料中，使用最少的原料张数，满足五种卷料的订单方案，并尽可能提高成材率。订单的长、宽与原材料的长、宽是双向选择的关系。对于多变量且变量与变量相互影响的问题，通常采用以一个变量为研究对象，固定其他变量，从而找到最优解的方法。对于此题，订单的长与宽两个变量是固定组合变换，可视为一个变量，原材料的长与宽同理也可以视为同一变量。选择订单长为固定变量或订单宽为固定变量都可以将多变量问题转换为单变量问题从而求解。若采取将订单长作为固定变量，则先按照订单长度将订单在原材料上纵向排开；若采取将订单长作为固定变量，则先按照订单宽度将订单在原材料上横向排开。

为使所需的原材料张数最小，将原材料按面积进行从大到小排序，面积越大，则所能制作的订单越多，所需要的张数就越小。大面积订单选择大面积原料进行切割，小面积订单选择小面积原料进行切割，会使得面积利用率较高，故对订单进行同样的排序操作。由于卷料订单长度普遍较长，对于问题一可筛除长度小于订单长度最小值的原料，从而减少算法运算量。面积匹配后，按照固定订单长或固定订单宽的方法依次排放订单，即可求出排刀方案以及所需的订单张数。

2.2 问题二的分析

在问题一的基础上，除五种卷料订单，另外加上 10 种板料订单。不同于问题一，板料订单长度普遍较小，故不存在原料长度小于订单长度最小值的情况，需要将所有原料都考虑在内。并且在问题一中，符合标准的余料都不能满足卷料订单的需求，而在问题二中进行卷料切割时会产生较多符合标准并且能够满足板料的余料，将余料做为新的一种原料进行优先切割来满足板料，从而提高成材率并减少使用的原料张数。

2.3 问题三的分析

在问题二的基础上，考虑排刀次数以及小机器切割数的情况，为尽可能减少排刀次数，在使用一张原材料时尽可能不更换订单的排列，同时让排在原料末尾的订单尽量相同从而减少小机器切割数。问题二中为减少使用原料张数而将余料进行二次切割，而针对问题三，对余料进行二次切割会增加人工排刀次数以及小机器切割数，故对于问题三要放弃对余料的二次切割从而降低排刀次数以及小机器切割次数。

2.4 问题四的分析

在问题二的基础上，考虑指定的浮动比例的情况，更加符合钢材制造的过程中的实际情况。在订单需求表里给出了七种订单的浮动比，由于七种订单规格不同，浮动比也不同，这里固定七种订单的浮动比，固定订单宽度将订单在原料上进行排刀，再利用所建模型进行数据处理求得最终解。其他变化的浮动比可代入模型中求解，利用随机数生成范围内的浮动比即可代入求解。

三、模型假设与约定

（1）假设切割过程中不会有原料损耗。

（2）假设切割所得订单均为合格品。

（3）假设排刀时不发生人为错误。

（4）假设卷料在通过卷取机压臂成卷的过程中不存在长度和宽度的变化。

（5）假设符合长度和宽度的剩余原料即可作为余料，不考虑其他形变等因素。

四、符号说明及名词定义

符号	含义
j	原材料编号 j
i	订单编号 i
L_{yj}	原材料 i 的长度
W_{yj}	原材料 i 的宽度
L_{di}	订单 i 的长度
W_{di}	订单 i 的宽度
N_{yj}	所用原材料 j 的数量
N_{di}	订单 i 的数量
S_{yj}	原材料 j 的面积
S_{di}	订单 i 的面积
L_{rj}	原材料 j 所剩余料的长度
W_{rj}	原材料 j 所剩余料的宽度
X_j	原料 j 所用张数
C	成材率
S_{rj}	原材料 j 所剩余料面积
n	所需原材料的总张数
N_j	原材料 j 的总量
Z	排刀次数与小机器切割数之和
k	每张原材料排刀次数
m	小机器切割次数

五、问题一模型建立与求解

5.1 数据解读与预处理

5.1.1 数据解读

根据附件三给出的工厂现有原料表以及附件四订单需求表，分析可知并不是每个原料都能满足订单需求，卷料订单长度普遍较大，而板料订单长度则较小，可以根据订单的长度来筛选原材料的规格，从而减少算法的运算量。

5.1.2 数据预处理

将原材料按照面积从大到小进行排序得到表 2-1

表 2-1　单个面积按降序排序的现有原料表

原料编号	长度	宽度	单个原料面积	库存	总面积
1	148623.91	1519.91	225894967	5	1129474835
5	75040.31	1573.71	118091686.3	3	354275058.8
6	138570.39	844.99	117090593.8	10	1170905938
7	98641.28	1184.54	116844541.8	12	1402134502
10	58023.82	1785.45	103598629.4	10	1035986294
9	104637.72	969.02	101396043.4	3	304188130.3
8	114074.27	879.02	100273564.8	9	902462083.3
3	75508.72	1232.32	93050905.83	8	744407246.6
2	32960.49	999.35	32939065.68	10	329390656.8
4	14091.52	920.62	12972935.14	2	25945870.28

将订单按照面积从大到小进行排序得到表 2-2

表 2-2　单个面积按降序排序的订单需求表

订单编号	长度	宽度	单个订单面积	需求量	总面积
1	44351.13	422.88	18755205.85	36	675187410.76
5	53479.79	332.29	17770799.42	18	319874389.54
3	54787.74	268.36	14702837.91	42	617519192.07
4	45284.39	277.70	12575475.10	32	402415203.30
2	39229.01	282.88	11097102.35	29	321815968.12
11	970.16	667.21	647300.45	34	22008215.42
7	896.09	714.72	640453.44	23	14730429.23
15	1243.03	471.25	585777.89	28	16401780.85

6	897.32	603.06	541137.80	38	20563236.37
10	752.61	641.45	482761.68	42	20275990.75
8	1096.33	435.84	477824.47	31	14812558.48
12	998.29	472.30	471492.37	25	11787309.18
13	1024.87	340.51	348978.48	24	8375483.61
9	890.53	343.08	305523.03	40	12220921.30
14	621.91	476.60	296402.31	22	6520850.73

5.2 模型的建立

问题一要求利用 10 种原料，在使所需原料张数最小的同时尽可能提高成材率的情况下，完成对 5 中卷料订单的切割。在数据预处理中，原材料与订单均已按照面积由大到小进行排序，使用的单个原材料面积越大，则使用张数越少。故针对问题一建立以下模型：

目标函数：

$$\text{Min n} = \sum_{j=1}^{10} X_j$$

满足订单需求的约束条件：

$$\begin{cases} L_{yj} \geq L_{di} & ① \\ W_{yj} \geq W_{di} & ② \\ \sum_{i=1}^{5} X_i * S_{yj} \geq \sum_{i=1}^{5} N_{di} * S_{di} & ③ \\ \text{Max c}=\dfrac{\sum_{i=1}^{5} N_{di} * S_{di} + \sum_{j=1}^{n} S_{rj}}{\sum_{i=1}^{5} X_i * S_{yj}} & ④ \\ N_{di} \leq N_j & ⑤ \end{cases}$$

其中：

j——原材料编号 j；i——订单编号 i；L_{yj}——原材料 i 的长度；W_{yj}——原材料 i 的宽度；L_{di}——订单 i 的长度；W_{di}——订单 i 的宽度；N_{di}——订单 i 的数量；S_{yj}——原材料 j 的面积；S_{di}——订单 i 的面积；L_{rj}——原材料 j 所剩余料的长度；W_{rj}——

原材料 j 所剩余料的宽度；X_j——原料 j 所用张数；C——成材率；S_{rj}——原材料 j 所剩余料面积；n——所需原材料的总张数；N_j——原材料 j 的总量。

在本问题背景下，5 种卷料长度普遍较长，根据约束条件中原材料的长度要长于订单长度就可以筛去部分原材料。再根据其他约束条件进行筛选比较，得出目标函数结果。

5.3 算法实施

利用以下算法对问题一模型进行求解：

⑴将原材料数据以及订单相关数据存储到列表中。

⑵按照单张面积从大到小分别对订单和原材料进行排序。

⑶定义相关的计数变量、决策变量。

⑷按照订单编号依次遍历所有订单。

⑸采取固定订单宽度的思想，先将原材料的宽度用订单填满。填满一列后再按照长度纵向填满订单，直到该张原材料的长度不足以放下任何订单长度，则调用下一张原材料。

算法流程如图 5-1：

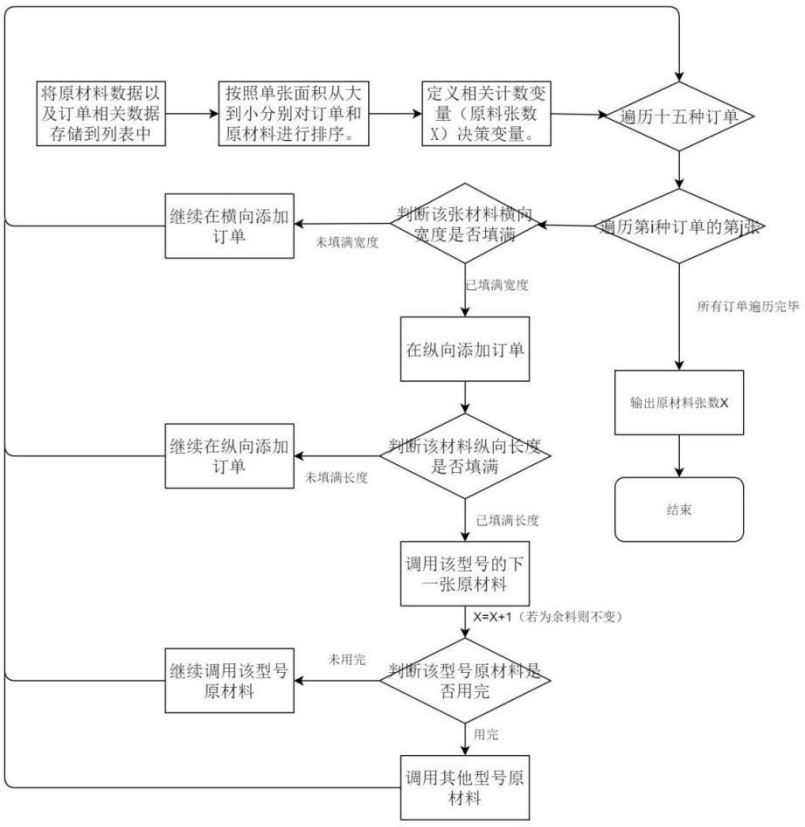

图 5-1　算法流程图

5.4 问题一的求解

由目标函数 Min n $=\sum_{j=1}^{10} X_j$求得完成 5 种卷料所需的最少原材料张数为 22 张，由公式 Max c=$\frac{\sum_{i=1}^{5} N_{di}*S_{di}+\sum_{j=1}^{n} S_{rj}}{\sum_{i=1}^{5} X_i*S_{yj}}$求得在原料张数为 22 张的情况下最大总成材率为 97.75%，具体原材料切割情况如表 5-2 所示：

表 5-2　材料切割及成材率

材料编号	张数	订单 1	订单 2	订单 3	订单 4	订单 5	成材率
1	4	9	0	0	0	0	100
1	1	0	0	0	0	8	100
6	1	0	0	3	3	0	97.673
6	3	0	0	0	9	0	98.62
7	1	0	6	0	2	0	96.08
7	2	0	8	0	0	0	96.43
5	2	0	0	0	0	4	100
5	1	0	0	3	0	2	100
6	6	0	0	6	0	0	95.28
7	1	0	7	0	0	0	96.43

余料情况如表 5-3 所示：

表 5-2　可回库余料情况

材料编号	可回库余料 1 长	可回库余料 1 宽	可回库余料 2 长	可回库余料 2 宽	可回库余料 3 长	可回库余料 3 宽
1	148623.91	251.27	15570.52	1268.64		
1	75040.31	190.75	41664.33	1382.96		
6	38498.65	844.99				
6	2717.22	844.99				
7	6055.74	630.54	14127.53	630.54		
7	20183.26	1184.54				
5	75040.31	244.55	21560.52	244.55		
5	21560.52	664.58	20252.57	806.64	75040.31	104.05
6	2899.491	805.08				
7	20193.26	1184.54	39229.01	282.88		

具体切割方式如图 5-3 所示：

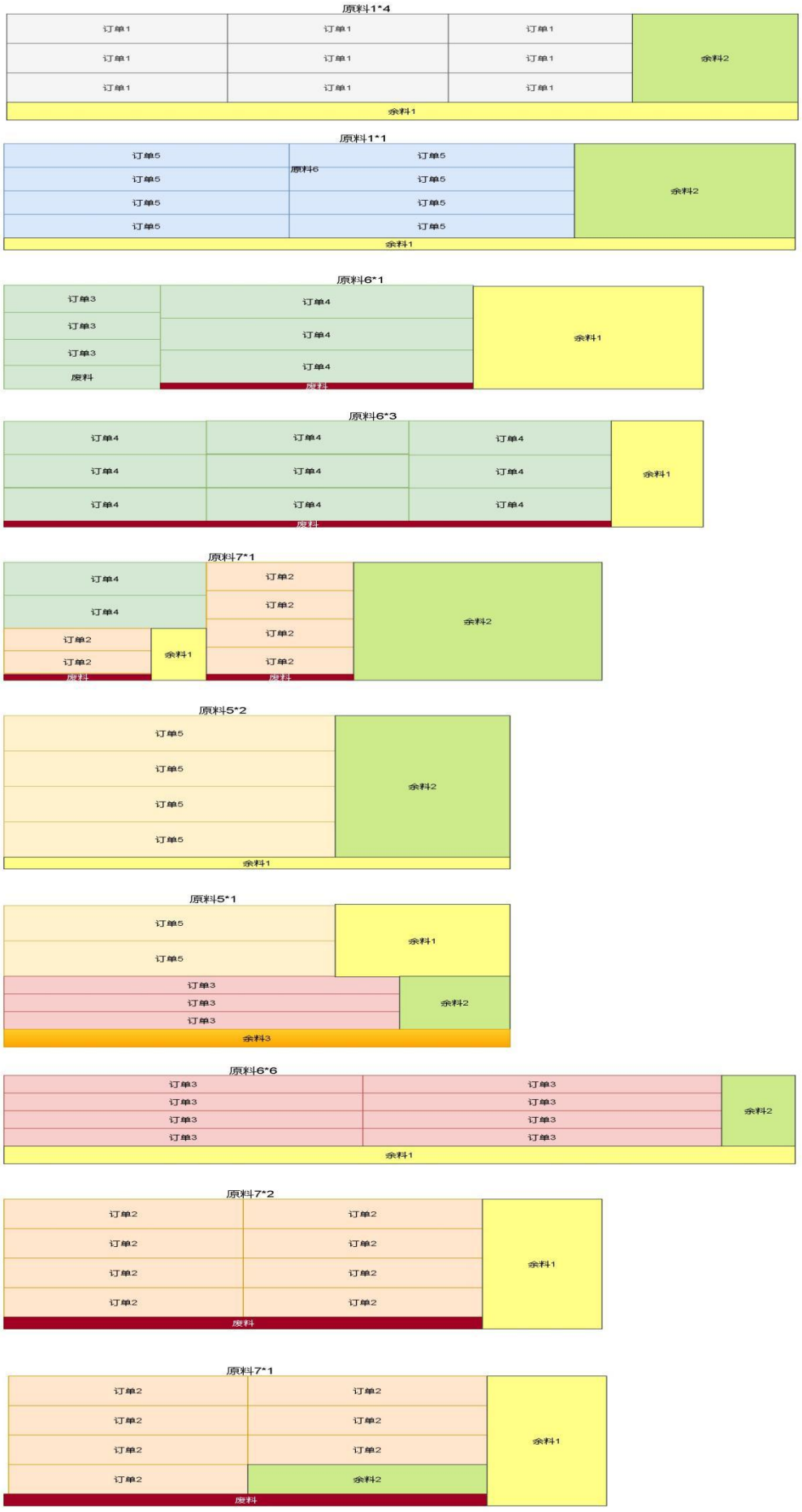
原料1*4
订单1
订单1
订单1
余料2
订单1
订单1
订单1
订单1
订单1
订单1
余料1
原料1*1
订单5
订单5
原料6
订单5
余料2
订单5
订单5
订单5
订单5
订单5
余料1
原料6*1
订单3
订单4
订单3
订单4
余料1
订单3
订单4
废料
废料
原料6*3
订单4
订单4
订单4
余料1
订单4
订单4
订单4
订单4
订单4
订单4
废料
原料7*1
订单4
订单2
订单4
订单2
余料2
订单2
订单2
余料1
订单2
订单2
废料
废料
原料5*2
订单5
订单5
余料2
订单5
订单5
余料1
原料5*1
订单5
余料1
订单5
订单3
余料2
订单3
订单3
余料3
原料6*6
订单3
订单3
余料2
订单3
订单3
订单3
订单3
订单3
订单3
订单3
余料1
原料7*2
订单2
订单2
订单2
订单2
余料1
订单2
订单2
订单2
订单2
订单2
订单2
废料
原料7*1
订单2
订单2
订单2
订单2
余料1
订单2
订单2
订单2
余料2
废料

图 5-3 原料切割方式

5.5 问题一的结果分析

根据所建立的模型以及算法对问题一进行求解，得到要完成的五种卷料订单，在尽可能提高成材率的情况下，最少需要 22 张原材料，切割方式及用料情况见附件提交结果表格。

六、问题二模型建立与求解

6.1 模型建立

问题二在问题一的基础上还需满足 10 种板料的订单要求，为减少原材料的使用张数，可将余料作为新的原材料进行订单的切割，同时提高了成材率。

目标函数为：

$$\text{Min n} = \sum_{j=1}^{10} X_j$$

满足订单需求的约束条件：

$$L_{yj} \geq L_{di} \quad ①$$

$$W_{yj} \geq W_{di} \quad ②$$

$$\sum_{i=1}^{15} X_i * S_{yj} \geq \sum_{i=1}^{15} N_{di} * S_{di} \quad ③$$

$$\text{Max c=} \frac{\sum_{i=1}^{15} N_{di} * S_{di} + \sum_{j=1}^{n} S_{rj}}{\sum_{i=1}^{5} X_i * S_{yj}} \quad ④$$

$$N_{di} \leq N_j \quad ⑤$$

$$L_{rj} \geq L_{di} \quad ⑥$$

$$W_{rj} \geq W_{di} \quad ⑦$$

其中：L_{rj}——原材料 j 所剩余料的长度；W_{rj}——原材料 j 所剩余料的宽度。

问题二板料长度较短，故认为对板料进行切割时不会产生符合标准的余料，余料来源均为切割卷料时产生的，将余料也作为新的原材料参与切割，从而尽可能的降低使用原材料的张数。

6.2 算法实施

在问题一实现算法的基础上，将余料作为原料进行切割，步骤如下：

⑴将原材料数据以及订单相关数据存储到列表中。

⑵按照单张面积从大到小分别对订单和原材料进行排序。

⑶定义相关的计数变量、决策变量。

⑷按照订单编号依次遍历所有订单。

⑸采取固定订单宽度的思想，先将原材料的宽度用订单填满。填满一列后再按照长度纵向填满订单，直到该张原材料的长度不足以放下任何订单长度，则调用下一张原材料。

⑹判断原材料剩余成分是否符合可回收余料标准，若符合则作为新的原料优先使用。

问题二算法实施流程图如图 6-1：

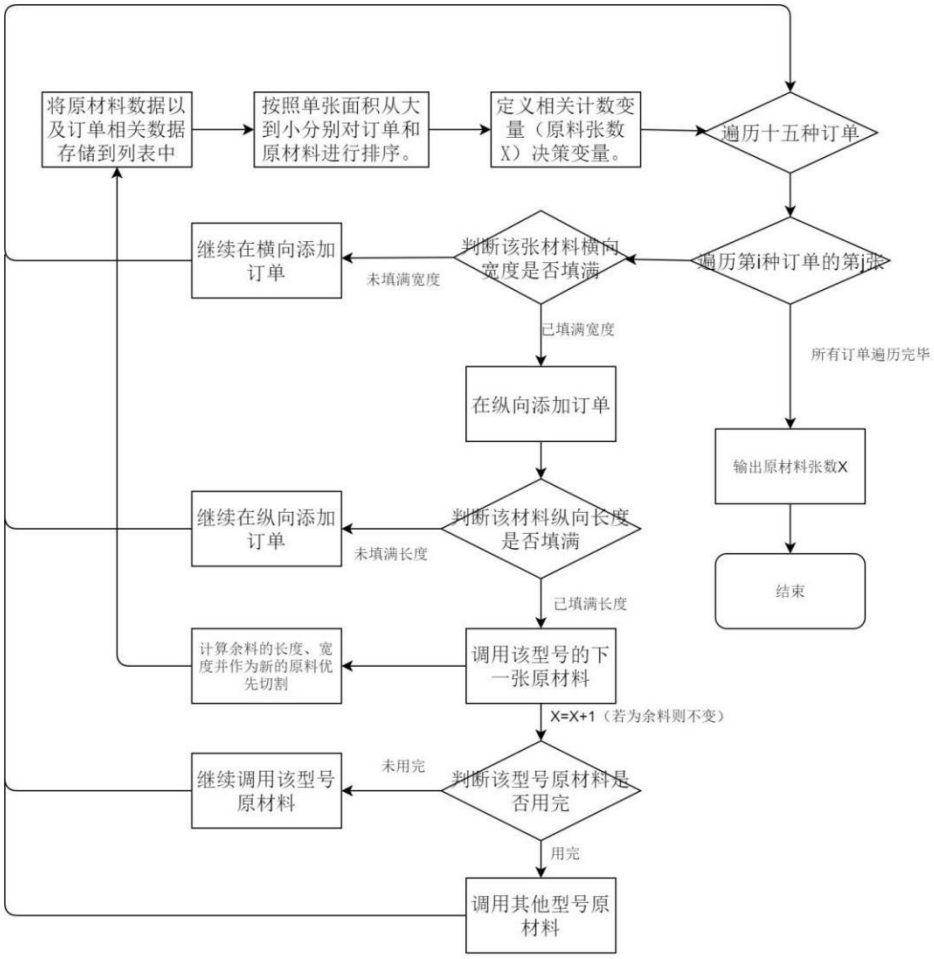

图 6-1 问题二算法实施流程图

6.3 问题二的求解

由目标函数 $\text{Min n} = \sum_{j=1}^{10} X_j$ 求得完成 15 种订单所需的最少原材料张数，在完成卷料订单之后，利用切割卷料订单后产生的余料进行二次切割，满足板料订单的需求，部分订单切割情况如表 6-2 所示：

表 6-2 部分订单切割情况

材料编号	张数	订单 4	订单 5	订单 6	订单 7	订单 8	订单 9	订单 10	成材率
1	4	0	0	0	0	0	40	42	97.43
1	1	0	8	0	23	0	0	0	87.73
6	1	3	0	13	0	0	0	0	86.51
6	3	9	0	0	0	0	0	0	98.62
7	1	2	0	0	0	0	0	0	96.08
7	1	0	0	22	0	0	0	0	86.18

部分余料情况如表 6-3 所示：

表 6-3 部分余料情况

材料编号	可回库余料 1 长	可回库余料 1 宽	可回库余料 2 长	可回库余料 2 宽	可回库余料 3 长	可回库余料 3 宽
1	148623.91	251.27				
1	75040.31	190.75				
6	0	0				
6	2717.22	844.99				
7	6055.74	630.54	14127.53	630.54		
7	0	0				
5	75040.31	244.55	21560.52	244.55		

5	21560.52	664.58	20252.57	806.64	75040.31	104.05
6	2899.491	805.08				
7	39229.01	282.88				

6.4 问题二的结果分析

在问题一模型以及算法的基础上针对新的约束条件作出相应修改，对问题二进行求解，得出要完成十五种订单，在尽可能提高成材率的情况下，最少需要 22 张原材料，切割方式及用料情况见附件提交结果表格。

七、问题三模型建立与求解

7.1 模型建立

在问题二的基础上，尽可能的减少排刀次数和小机器切割数。即同一张原材料的排刀方式尽可能不变。为达到此目的，将相同的订单排在同一张原材料上，并减少对余料的切割次数。故问题三区别于问题二，不考虑余料的二次切割，把符合标准的余料回收入库即可。

基于问题二的模型，以减少排刀次数和小机器切割数为优化目标，设定优化条件，建立优化模型。

目标函数为：$\text{Min n} = \sum_{j=1}^{10} X_j$

满足订单需求的约束条件：

$$L_{yj} \geq L_{di} \quad ①$$

$$W_{yj} \geq W_{di} \quad ②$$

$$\sum_{i=1}^{15} X_i * S_{yj} \geq \sum_{i=1}^{15} N_{di} * S_{di} \quad ③$$

$$\text{Max c=} \frac{\sum_{i=1}^{15} N_{di} * S_{di} + \sum_{j=1}^{n} S_{rj}}{\sum_{i=1}^{5} X_i * S_{yj}} \quad ④$$

$$N_{di} \leq N_j \quad ⑤$$

$$L_{rj} \geq L_{di} \quad ⑥$$

$$W_{rj} \geq W_{di} \quad ⑦$$

$$\text{Min Z=} \sum_{j=1}^{10} k_j * m_j \quad ⑧$$

其中：Z——排刀次数与小机器切割数之和；k——每张原材料排刀次数；m——小机器切割次数。

问题三在问题二的基础上，需要尽可能的减少排刀次数和小机器切割数，于是在问题二模型的基础上，添加 Min Z= $\sum_{j=1}^{10} k_j * m_j$ 的约束条件。从而达到题目要求。

7.2 算法实施

在问题二实现算法的基础上，余料不进行二次切割，从而减少小机器切割次数和排刀次数，对小机器切割次数和排刀次数进行最小值约束，步骤如下：

⑴将原材料数据以及订单相关数据存储到列表中。

⑵按照单张面积从大到小分别对订单和原材料进行排序。

⑶定义相关的计数变量、决策变量。

⑷按照订单编号依次遍历所有订单。

⑸采取固定订单宽度的思想，先将原材料的宽度用订单填满。填满一列后再按照长度纵向填满订单，直到该张原材料的长度不足以放下任何订单长度，则调用下一张原材料。

⑹分别对小机器切割次数和排刀次数进行累加，最终合并求和。

算法流程图如图 7-2：

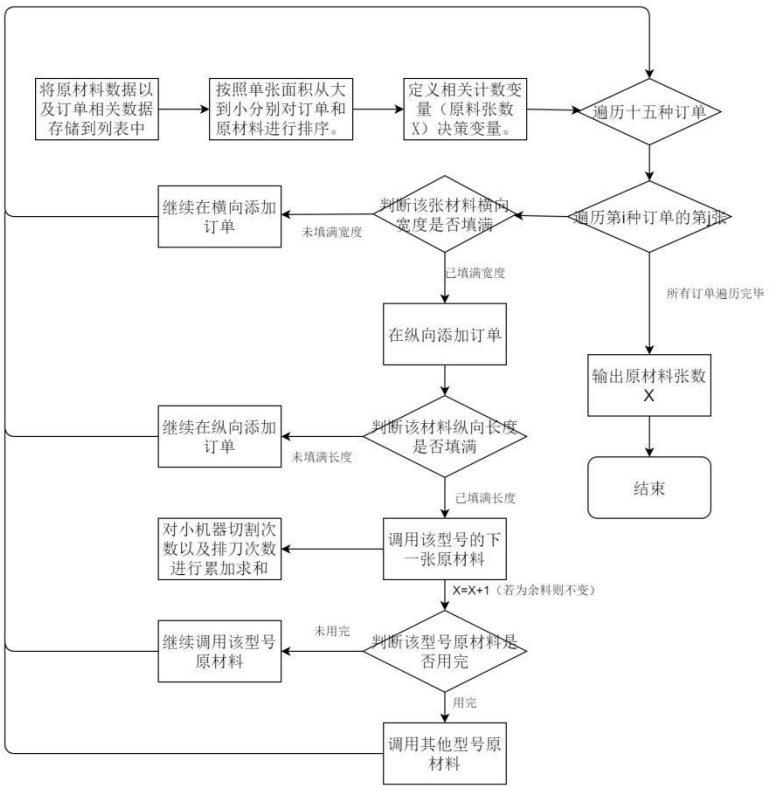

图 7-2 问题三算法流程图

7.3 问题三的求解

由目标函数 Min n $=\sum_{j=1}^{10} X_j$求得完成 15 种订单所需的最少原材料张数，但在条件 Min Z= $\sum_{j=1}^{10} k_j * m_j$的约束下，不对余料进行二次切割，尽可能减少排刀次数和小机器切割数，满足订单的需求，部分订单切割情况如表 7-3 所示：

表 7-3 问题三部分订单切割情况

张数	订单 4	订单 5	订单 6	订单 7	订单 8	订单 9
4	0	0	0	0	0	0
1	0	8	0	0	0	0
1	3	0	0	0	0	0
3	9	0	0	0	0	0
1	2	0	0	0	0	0
2	0	0	0	0	0	0
1	0	0	0	0	8	40

排刀次数及小机器切割数如表 7-4 所示：

表 7-4 问题三排刀次数及小机器切割数情况

材料编号	张数	换刀次数	小机器切割次数
1	4	0	1
1	1	0	1
6	1	1	2
6	3	0	1
7	1	1	1
7	2	0	1
5	2	0	1

5	1	0	2
6	6	0	1
7	1	1	2
7	1	4	5
7	1	8	9
切割总数		15	27

7.4 问题三的结果分析

在问题二模型以及算法的基础上针对排刀次数以及小机器切割数的约束条件作出相应修改，对问题三进行求解，得出要完成十五种订单，在尽可能提高成材率并减少排刀次数以及小机器切割数的情况下，最少需要 24 张原材料，排刀总数为 15 次，小机器切割数为 27 次，人工操作总数为 42 次，切割方式及用料情况见附件提交结果表格。

八、问题四模型建立与求解

8.1 模型建立

在问题二的基础上，钢材制造的过程中的实际情况，对订单长度规定额外的浮动比例，寻找最优的切割方案。由于浮动比例变化情况较多，只针对一组浮动比例进行模型求解，其他情况可随机生成符合范围内的浮动比例代入此模型求解。

问题四模型与问题二模型相似，对于规定订单长度额外浮动比例的长度可由公式计算得出：

$$F_{yj} = （1 + r） * L_{yj} \qquad ⑨$$

其中：F_{yj}——实际订单长度；r——浮动比例；L_{yj}——标准订单长度。

8.2 算法实施

在问题二实现算法的基础上，对规定浮动比例的订单长度进行修改，按照问题二的算法求得切割方案，步骤如下：

⑴将原材料数据以及订单相关数据存储到列表中。

⑵按照规定浮动比例修改标准订单长度。

⑶按照单张面积从大到小分别对订单和原材料进行排序。

⑷定义相关的计数变量、决策变量。

⑸按照订单编号依次遍历所有订单。

⑹采取固定订单宽度的思想，先将原材料的宽度用订单填满。填满一列后再按照长度纵向填满订单，直到该张原材料的长度不足以放下任何订单长度，则调用下一张原材料。

算法流程图如图 8-1 所示：

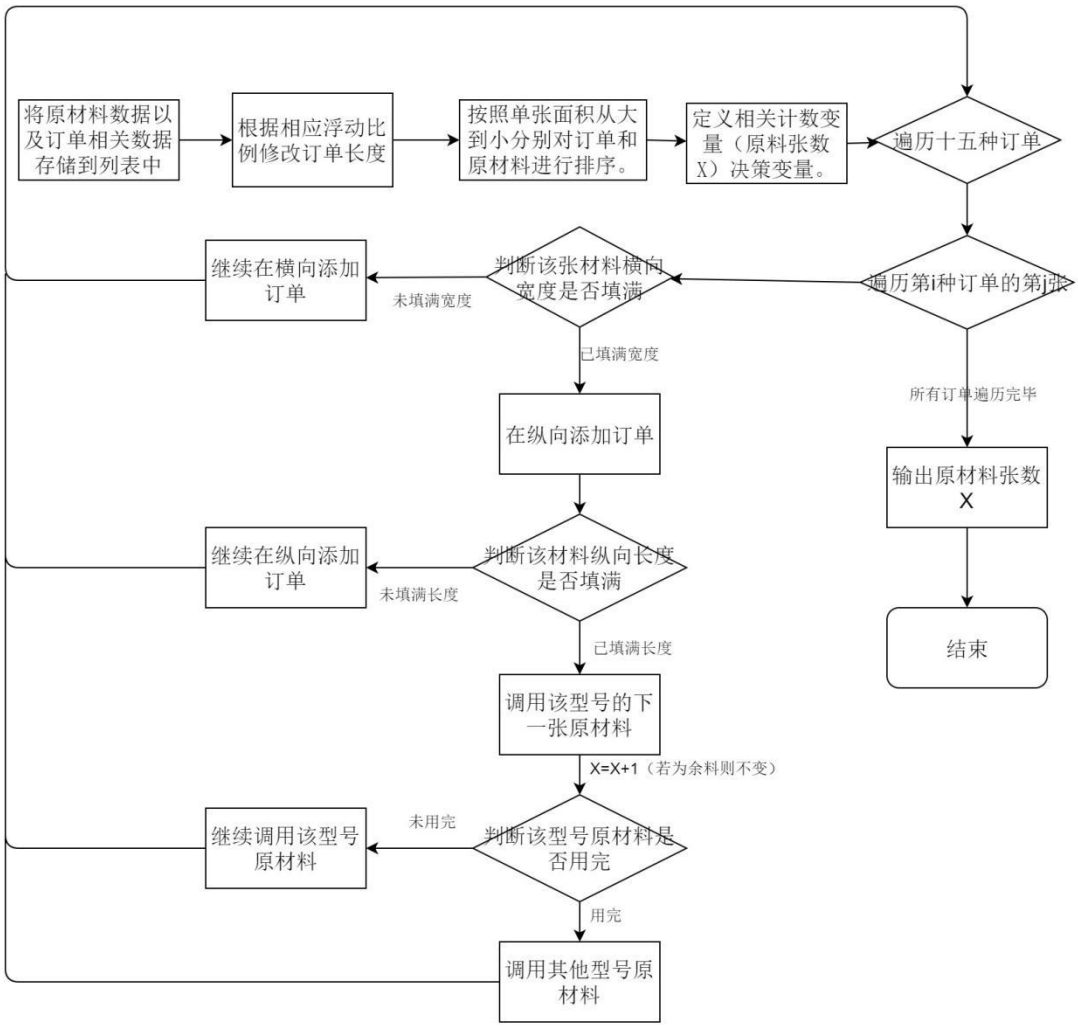

图 8-2 问题四的算法流程图

8.3 问题四的求解

目标函数与约束条件都同于问题二，在问题二的基础上考虑部分订单的浮动比例，满足所有订单要求，浮动比例规定如表 8-3 所示：

表 8-3 浮动比例规定

订单 1 浮比	订单 4 浮比	订单 5 浮比	订单 7 浮比	订单 8 浮比	订单 12 浮比	订单 15 浮比
4%	0%	-2%	2%	7%	3%	5%

部分订单切割情况如表 8-3 所示：

表 8-3 问题四部分订单切割情况

材料编号	张数	订单 8	订单 9	订单 10	订单 11	成材率
1	4	0	40	42	0	97.43
1	1	0	0	0	34	87.73
6	1	0	0	0	0	86.51
6	3	0	0	0	0	98.62
7	1	0	0	0	0	96.08
7	2	31	0	0	0	96.96

8.4 问题四的结果分析

在考虑到浮动比例的实际情况下，基于问题二的模型和算法，得出要完成十五种订单，在尽可能提高成材率的情况下，最少需要 22 张原材料。由于板料订单自身长度就比原料长度小很多，而浮动比例仅在 10%以内变化，故订单长度的变化相比于原材料几乎可以忽略不计，所以切割方式以及对原材料的需求基本没有发什么变化，但对成材率有着一定的影响。具体切割方式及用料情况见附件提交结果表格。

九、模型的评价

9.1 优点

（1）数据处理上，除了将实际情况中的硬性限制条件作为约束条件，还将成材率也作为约束条件，从而尽可能的满足题目要求。

（2）模型易于理解和操作，且算法的处理上选择用相对简单易懂的方式解决问题。

9.2 缺点

（1）对于余料的分析，部分余料没有进行实际的运算比较，对于算法的精度还有待提高。

（2）若数据过于庞大，算法不够优化，可能会需要处理较多数据，从而导致运算速度较慢。

十、参考文献

【1】马振华等编，现代应用数学手册——运筹学与最优化理论卷，北京：清华大学出版社，2002

【2】赵静等编，数学建模与数学实验， 北京：高等教育出版社，2002

【3】陈东炎，陈冬梅，王树忠，数学建模， 科学出版社，2007

十一、附录

1.第一题主程序代码

```
import copy
yL=[148623.91,32960.49,75508.72,14091.52,75040.31,138570.39,98641.28,114074.27,104637.72,58023.82]
yW=[1519.91,999.35,1232.32,920.62,1573.71,844.99,1184.54,879.02,969.02,1785.45]
yN=[5,10,8,2,3,10,12,9,3,10]
dL=[44351.13,39229.01,54787.74,45284.39,53479.79,897.32,896.09,1096.33,890.53,422.88,970.16,998.29,1024.87,621.91,1243.03]
dW=[422.88,282.88,268.36,277.70,332.29,603.06,714.72,435.84,343.08,641.45,667.21,472.30,340.51,476.60,471.25]
dN=[36,29,42,32,18,38,23,31,40,42,34,25,24,22,28]
yS=[]
dS=[]
for i in range(0,10):
    yS.append([yL[i]*yW[i],yL[i],yW[i],yN[i],i+1])
for i in range(0,5):
    dS.append([dL[i]*dW[i],dL[i],dW[i],dN[i],i+1])
y = sorted(yS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
d = sorted(dS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
y.reverse()
d.reverse()
y2=copy.deepcopy(y)
d2=copy.deepcopy(d)
result=1
n=0
x=0
k=y[n][2]
p1=[]
p2=[]
p3=[]
z=0
sd=0
sy=0
for i in range(0,5):
```

```
    for j in range(0,d[i][3]):
        z=z+1
        sd=sd+d[i][0]#算成材率
        p1.append(d[i][4])
        if y[n][2]<d[i][2]:
            p1.pop()
            p2.append(p1)
            p1=[d[i][4]]
            if j==0:
                y[n][1]=y[n][1]-d[i-1][1]
            else:
                y[n][1]=y[n][1]-d[i][1]
            if y[n][1]>=d[i][1]:

                y[n][2]=y2[n][2]
                y[n][2]=y[n][2]-d[i][2]
            elif y[n][1]<d[i][1]:
                sy=sy+y[n][0]#算成材率
                p3.append(p2)
                p3.append(y[n][4])
                p2=[]
                y[n][3]=y[n][3]-1
                result=result+1
                if y[n][3]==0:
                    n=n+1
                    y[n][2]=y[n][2]-d[i][2]
                else:
                    y[n][1]=y2[n][1]
                    y[n][2]=y2[n][2]-d[i][2]

        else:
            y[n][2]=y[n][2]-d[i][2]
        if i==4 and j==(d[i][3]-1):
                p2.append(p1)
                p3.append(p2)
                p3.append(y[n][4])
c=sd/sy
print(result,p3,c)
```

2. 第二题主程序代码

```
import copy
yL=[148623.91,32960.49,75508.72,14091.52,75040.31,138570.39,98641.28,114074.
27,104637.72,58023.82]
yW=[1519.91,999.35,1232.32,920.62,1573.71,844.99,1184.54,879.02,969.02,1785.
45]
yN=[5,10,8,2,3,10,12,9,3,10]
```

```
dL=[44351.13,39229.01,54787.74,45284.39,53479.79,897.32,896.09,1096.33,890.
53,422.88,970.16,998.29,1024.87,621.91,1243.03]
dW=[422.88,282.88,268.36,277.70,332.29,603.06,714.72,435.84,343.08,641.45,6
67.21,472.30,340.51,476.60,471.25]
dN=[36,29,42,32,18,38,23,31,40,42,34,25,24,22,28]
yuL=[15570.52,15570.52,15570.52,15570.52,41664.33,38498.65,2717.22,2717.22,
2717.22,6055.74,14127.53,20183.26,20183.26,21560.52,20252.57,2899.491,2899.
491,2899.491,2899.491,2899.491,2899.491,20193.26]
yuW=[1268.64,1268.64,1268.64,1268.64,1382.96,844.99,844.99,844.99,844.99,63
0.54,630.54,1184.54,1184.54,664.58,806.64,805.08,805.08,805.08,805.08,805.0
8,805.08,1184.54]
yuB=[1,1,1,1,12,6,62,62,62,7,72,73,73,5,52,63,63,63,63,63,63,74]
yuN=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
yS=[]
dS=[]
yuS=[]
for i in range(0,22):
    yuS.append([yuL[i]*yuW[i],yuL[i],yuW[i],yuN[i],yuB[i]])
for i in range(0,10):
    yS.append([yL[i]*yW[i],yL[i],yW[i],yN[i],i+1])
for i in range(0,15):
    dS.append([dL[i]*dW[i],dL[i],dW[i],dN[i],i+1])
y = sorted(yS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
d = sorted(dS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
yu = sorted(yuS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
y.reverse()
d.reverse()
yu.reverse()
z=0

for i in yu:
    y.insert(3+z,i)
    z=z+1
del y[0]
del y[0]
del y[0]
y2=copy.deepcopy(y)
d2=copy.deepcopy(d)
yu2=copy.deepcopy(yu)
result=0
n=0
x=0
k=y[n][2]
p1=[]
p2=[]
p3=[]
```

```
z=0
sd=0
sy=0
for i in range(5,15):
    for j in range(0,d[i][3]):
        z=z+1
        sd=sd+d[i][0]#算成材率
        p1.append(d[i][4])
        if y[n][2]<d[i][2]:
            p1.pop()
            p2.append(p1)
            p1=[d[i][4]]
            if j==0:
                y[n][1]=y[n][1]-d[i-1][1]
            else:
                y[n][1]=y[n][1]-d[i][1]
            if y[n][1]>=d[i][1]:

                y[n][2]=y2[n][2]
                y[n][2]=y[n][2]-d[i][2]
            elif y[n][1]<d[i][1]:
                sy=sy+y[n][0]#算成材率
                p3.append(p2)
                p3.append(y[n][4])
                p3.append(y2[n][2])
                p2=[]
                y[n][3]=y[n][3]-1
                if  n>21:
                    result=result+1
                if y[n][3]==0:
                    n=n+1
                    y[n][2]=y[n][2]-d[i][2]
                else:
                    y[n][1]=y2[n][1]
                    y[n][2]=y2[n][2]-d[i][2]

        else:
            y[n][2]=y[n][2]-d[i][2]
        if i==14 and j==(d[i][3]-1):
                p2.append(p1)
                p3.append(p2)
                p3.append(y[n][4])
c=sd/sy
print(result,p3,c)
```

3. 第三题主程序代码

```
import copy
yL=[148623.91,32960.49,75508.72,14091.52,75040.31,138570.39,98641.28,114074.
27,104637.72,58023.82]
yW=[1519.91,999.35,1232.32,920.62,1573.71,844.99,1184.54,879.02,969.02,1785.
45]
yN=[5,10,8,2,3,10,12,9,3,10]
dL=[44351.13,39229.01,54787.74,45284.39,53479.79,897.32,896.09,1096.33,890.
53,422.88,970.16,998.29,1024.87,621.91,1243.03]
dW=[422.88,282.88,268.36,277.70,332.29,603.06,714.72,435.84,343.08,641.45,6
67.21,472.30,340.51,476.60,471.25]
dN=[36,29,42,32,18,38,23,31,40,42,34,25,24,22,28]
yS=[]
dS=[]
for i in range(0,10):
    yS.append([yL[i]*yW[i],yL[i],yW[i],yN[i],i+1])
for i in range(0,15):
    dS.append([dL[i]*dW[i],dL[i],dW[i],dN[i],i+1])
y = sorted(yS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
d = sorted(dS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
y.reverse()
d.reverse()
y2=copy.deepcopy(y)
d2=copy.deepcopy(d)
result=1
n=0
x=0
k=y[n][2]
p1=[]
p2=[]
p3=[]
z=0
sd=0
sy=0
for i in range(0,15):
    for j in range(0,d[i][3]):
        z=z+1
        sd=sd+d[i][0]#算成材率
        p1.append(d[i][4])
        if y[n][2]<d[i][2]:
            p1.pop()
            p2.append(p1)
            p1=[d[i][4]]
            if j==0:
                y[n][1]=y[n][1]-d[i-1][1]
            else:
```

```
                y[n][1]=y[n][1]-d[i][1]
            if y[n][1]>=d[i][1]:

                y[n][2]=y2[n][2]
                y[n][2]=y[n][2]-d[i][2]
            elif y[n][1]<d[i][1]:
                sy=sy+y[n][0]#算成材率
                p3.append(p2)
                p3.append(y[n][4])
                p2=[]
                y[n][3]=y[n][3]-1
                result=result+1
                if y[n][3]==0:
                    n=n+1
                    y[n][2]=y[n][2]-d[i][2]
                else:
                    y[n][1]=y2[n][1]
                    y[n][2]=y2[n][2]-d[i][2]

        else:
            y[n][2]=y[n][2]-d[i][2]
        if i==14 and j==(d[i][3]-1):
                p2.append(p1)
                p3.append(p2)
                p3.append(y[n][4])
c=sd/sy
print(result,p3,c)
```

4. 第四题主程序代码

```
import copy
yL=[148623.91,32960.49,75508.72,14091.52,75040.31,138570.39,98641.28,114074.
27,104637.72,58023.82]
yW=[1519.91,999.35,1232.32,920.62,1573.71,844.99,1184.54,879.02,969.02,1785.
45]
yN=[5,10,8,2,3,10,12,9,3,10]
dL=[45681.66,39229.01,54787.74,45284.39,57223.3753,897.32,834.326,1074.4034,
890.53,422.88,970.16,1008.2729,1024.87,621.91,1342.47]
dW=[422.88,282.88,268.36,277.70,332.29,603.06,714.72,435.84,343.08,641.45,6
67.21,472.30,340.51,476.60,471.25]
dN=[36,29,42,32,18,38,23,31,40,42,34,25,24,22,28]
yS=[]
dS=[]
for i in range(0,10):
    yS.append([yL[i]*yW[i],yL[i],yW[i],yN[i],i+1])
for i in range(0,5):
    dS.append([dL[i]*dW[i],dL[i],dW[i],dN[i],i+1])
y = sorted(yS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
```

```
d = sorted(dS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
y.reverse()
d.reverse()
y2=copy.deepcopy(y)
d2=copy.deepcopy(d)
result=1
n=0
x=0
k=y[n][2]
p1=[]
p2=[]
p3=[]
z=0
sd=0
sy=0
for i in range(0,5):
    for j in range(0,d[i][3]):
        z=z+1
        sd=sd+d[i][0]#算成材率
        p1.append(d[i][4])
        if y[n][2]<d[i][2]:
            p1.pop()
            p2.append(p1)
            p1=[d[i][4]]
            if j==0:
                y[n][1]=y[n][1]-d[i-1][1]
            else:
                y[n][1]=y[n][1]-d[i][1]
            if y[n][1]>=d[i][1]:

                y[n][2]=y2[n][2]
                y[n][2]=y[n][2]-d[i][2]
            elif y[n][1]<d[i][1]:
                sy=sy+y[n][0]#算成材率
                p3.append(p2)
                p3.append(y[n][4])
                p2=[]
                y[n][3]=y[n][3]-1
                result=result+1
                if y[n][3]==0:
                    n=n+1
                    y[n][2]=y[n][2]-d[i][2]
                else:
                    y[n][1]=y2[n][1]
                    y[n][2]=y2[n][2]-d[i][2]
```

```
        else:
            y[n][2]=y[n][2]-d[i][2]
        if i==4 and j==(d[i][3]-1):
                p2.append(p1)
                p3.append(p2)
                p3.append(y[n][4])
c=sd/sy
print(result,p3,c)
yL=[148623.91,32960.49,75508.72,14091.52,75040.31,138570.39,98641.28,114074.
27,104637.72,58023.82]
yW=[1519.91,999.35,1232.32,920.62,1573.71,844.99,1184.54,879.02,969.02,1785.
45]
yN=[5,10,8,2,3,10,12,9,3,10]
dL=[44351.13,39229.01,54787.74,45284.39,53479.79,897.32,896.09,1096.33,890.
53,422.88,970.16,998.29,1024.87,621.91,1243.03]
dW=[422.88,282.88,268.36,277.70,332.29,603.06,714.72,435.84,343.08,641.45,6
67.21,472.30,340.51,476.60,471.25]
dN=[36,29,42,32,18,38,23,31,40,42,34,25,24,22,28]
yuL=[11578.93,11578.93,11578.93,11578.93,34177.16,38498.65,2717.22,2717.22,
2717.22,6055.74,14127.53,20183.26,20183.26,17816.935,20252.57,2899.491,2899.
491,2899.491,2899.491,2899.491,2899.491,20193.26]
yuW=[1268.64,1268.64,1268.64,1268.64,1382.96,844.99,844.99,844.99,844.99,63
0.54,630.54,1184.54,1184.54,664.58,806.64,805.08,805.08,805.08,805.08,805.0
8,805.08,1184.54]
yuB=[1,1,1,1,12,6,62,62,62,7,72,73,73,5,52,63,63,63,63,63,63,74]
yuN=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
yS=[]
dS=[]
yuS=[]
for i in range(0,22):
    yuS.append([yuL[i]*yuW[i],yuL[i],yuW[i],yuN[i],yuB[i]])
for i in range(0,10):
    yS.append([yL[i]*yW[i],yL[i],yW[i],yN[i],i+1])
for i in range(0,15):
    dS.append([dL[i]*dW[i],dL[i],dW[i],dN[i],i+1])
y = sorted(yS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
d = sorted(dS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
yu = sorted(yuS, key=(lambda x: [x[0],x[1], x[2],x[3],x[4]]))
y.reverse()
d.reverse()
yu.reverse()
z=0

for i in yu:
    y.insert(3+z,i)
    z=z+1
```

```
del y[0]
del y[0]
del y[0]
y2=copy.deepcopy(y)
d2=copy.deepcopy(d)
yu2=copy.deepcopy(yu)
result=0
n=0
x=0
k=y[n][2]
p1=[]
p2=[]
p3=[]
z=0
sd=0
sy=0
for i in range(5,15):
    for j in range(0,d[i][3]):
        z=z+1
        sd=sd+d[i][0]#算成材率
        p1.append(d[i][4])
        if y[n][2]<d[i][2]:
            p1.pop()
            p2.append(p1)
            p1=[d[i][4]]
            if j==0:
                y[n][1]=y[n][1]-d[i-1][1]
            else:
                y[n][1]=y[n][1]-d[i][1]
            if y[n][1]>=d[i][1]:

                y[n][2]=y2[n][2]
                y[n][2]=y[n][2]-d[i][2]
            elif y[n][1]<d[i][1]:
                sy=sy+y[n][0]#算成材率
                p3.append(p2)
                p3.append(y[n][4])
                p3.append(y2[n][2])
                p2=[]
                y[n][3]=y[n][3]-1
                if  n>21:
                    result=result+1
                if y[n][3]==0:
                    n=n+1
                    y[n][2]=y[n][2]-d[i][2]
                else:
```

```
                    y[n][1]=y2[n][1]
                    y[n][2]=y2[n][2]-d[i][2]

        else:
            y[n][2]=y[n][2]-d[i][2]
        if i==14 and j==(d[i][3]-1):
                p2.append(p1)
                p3.append(p2)
                p3.append(y[n][4])
c=sd/sy
print(result,p3,c)
```

注：代码均用 python 编写