# 一、目标函数



其中是最小化浪费布料长度，即使用400cm布料的总长度-总需求长度, 和分别代表100cm裙子和60cm衣领总需求量。而表示对k大小的惩罚项，防止出现执行次数大于0的重复裁剪模式，详解见附言1

# 二、参数

|  |  |  |  |
| --- | --- | --- | --- |
| 裁剪方案 | 100cm数量 | 60cm数量 | 方案执行次数 |
| 1 | X11 | X12 | Y1 |
| 2 | X21 | X22 | Y2 |
| … | … | … | … |
| k | Xk1 | Xk2 | Yk |

其中参数X1, X2分别为每种裁剪方案中100cm和60cm的数量，Y为每种方案的裁剪次数，均为大于等于0的INTEGER类型

# 三、约束条件

约束1：每个裁剪模式的布料总长度不能超过400

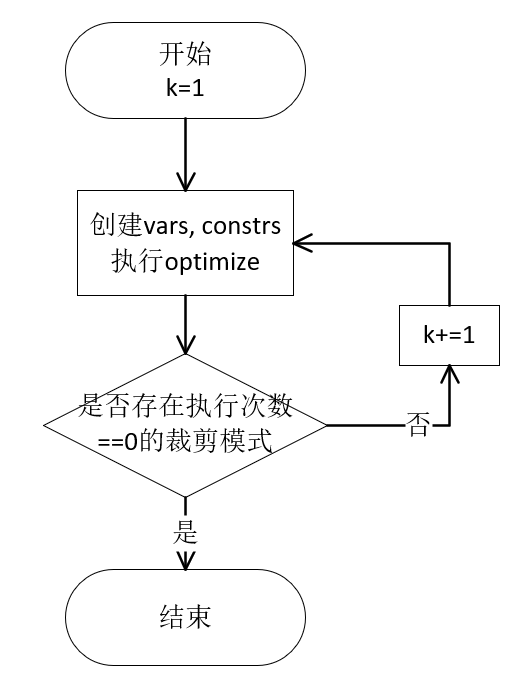


约束2：满足需求

即所有裁剪模式产生的总100cm数量大于等于需求量

即所有裁剪模式产生的总60cm数量大于等于需求量

# 四、模型运行流程



执行结果：

假设：每个公主需要12个100cm裙子，每个王子需要13个60cm衣服，每个布料400cm

若共有31个公主和52个王子：

则demand100=31×12, demand60=52×13，模型将从k=1开始，即只使用一种裁剪方案

开始：

k=1

最优解:

裁剪方案 1: 226.0 次, 100cm裁剪 2.0 次, 60cm裁剪 3.0 次

最小浪费布料总长度：12640.0

判断：此时不存在执行次数==0的裁剪方案，则k+=1，返回第一步

k=2

最优解:

裁剪方案 1: 59.0 次, 100cm裁剪 4.0 次, 60cm裁剪 0.0 次

裁剪方案 2: 136.0 次, 100cm裁剪 1.0 次, 60cm裁剪 5.0 次

最小浪费布料总长度：240.0

判断：此时不存在执行次数==0的裁剪方案，则k+=1，返回第一步

k=3

最优解:

裁剪方案 1: 59.0 次, 100cm裁剪 4.0 次, 60cm裁剪 0.0 次

裁剪方案 2: 0.0 次, 100cm裁剪 -0.0 次, 60cm裁剪 -0.0 次

裁剪方案 3: 136.0 次, 100cm裁剪 1.0 次, 60cm裁剪 5.0 次

最小浪费布料总长度：240.0

判断：此时存在执行次数==0的裁剪方案，找到最优解，运行结束。

之所以以存在执行次数等于0的裁剪方案为结束条件，详见附言2。

**附言1**

目标函数除了最小化浪费布料外，增加了这一惩罚项：

表示对k大小的惩罚项，防止出现执行次数大于0的重复裁剪模式

这一项包括两个点：

1.为什么要加上分子部分？

防止存在有执行次数>0的两个相同的方案，模型会把这两个方案的执行次数合并（这里很难用语言解释，可以看下面的增加/不增加惩罚项的对比，更加容易理解）

2.为什么要除以分母部分？

每个方案最多裁剪的成品为6个(400÷60≈6)，除6×k后，该惩罚项的大小总是小于1，**不会对问题目标最小化浪费产生影响。**

依然以文中31个公主和52个王子为例：

**不增加惩罚项**：

最优解：

裁剪方案 1: **56.0** 次, 100cm裁剪 4.0 次, 60cm裁剪 -0.0 次

裁剪方案 2: 136.0 次, 100cm裁剪 1.0 次, 60cm裁剪 5.0 次

裁剪方案 3: **3.0** 次, 100cm裁剪 4.0 次, 60cm裁剪 -0.0 次

最小浪费布料总长度：240.0

此时方案1和3是相同的，为了后续的循环结束条件（存在执行次数为0的方案），我们要把他们执行次数合并

**增加惩罚项：**

最优解:

裁剪方案 1: **59.0** 次, 100cm裁剪 4.0 次, 60cm裁剪 0.0 次

裁剪方案 2: 0.0 次, 100cm裁剪 -0.0 次, 60cm裁剪 -0.0 次

裁剪方案 3: 136.0 次, 100cm裁剪 1.0 次, 60cm裁剪 5.0 次

最优目标值: 240.55555555556202

最小浪费布料总长度：240.0

惩罚项: 0.5555555555555556

可以看到，之前的56次和3次已经合并为59次。

且根据上文解释，惩罚项始终是小于1的，不会对最小化浪费这个目标产生实质影响。

稍微改动一下目标函数，就可以极大的减少繁琐的约束条件，提高模型效率。

**附言2**

为什么要以存在执行次数等于0的裁剪方案为循环结束条件？

答：

因为当有k个裁剪方案，且存在执行0次的裁剪方案时，说明模型用k-1个裁剪方案就实现了最小化浪费，就算增加了裁剪方案个数，新的方案执行次数依然为0。用影子价格的思想解释的话，类似于“最大方案个数”这个约束的影子价格为0，即在此约束下模型已达到最优解，放宽此约束无法使模型得到更优值。

还是以文中31个公主和52个王子为例：

第二次循环(k=2)时的最优解：

裁剪方案 1: **59.0** 次, 100cm裁剪 4.0 次, 60cm裁剪 0.0 次

裁剪方案 2: **136.0** 次, 100cm裁剪 1.0 次, 60cm裁剪 5.0 次

第三次循环(k=3)的最优解

最优解:

裁剪方案 1: **59.0** 次, 100cm裁剪 4.0 次, 60cm裁剪 0.0 次

裁剪方案 2: 0.0 次, 100cm裁剪 -0.0 次, 60cm裁剪 -0.0 次

裁剪方案 3: **136.0** 次, 100cm裁剪 1.0 次, 60cm裁剪 5.0 次

此时结束条件满足，**停止**

若此时不停止，依旧循环，则得到：

k=4

裁剪方案 1: -0.0 次, 100cm裁剪 0.0 次, 60cm裁剪 0.0 次

裁剪方案 2: **59.0** 次, 100cm裁剪 4.0 次, 60cm裁剪 -0.0 次

裁剪方案 3: **136.**0 次, 100cm裁剪 1.0 次, 60cm裁剪 5.0 次

裁剪方案 4: -0.0 次, 100cm裁剪 0.0 次, 60cm裁剪 -0.0 次

我们可以看到，就算增加了最大裁剪方案k，模型的解依然与之前相同。此时就算继续循环至k=4，依然会得到相同的解，不妨结束。