基于概率统计的生产决策模型

摘要

本文旨在建立数学模型，综合考虑零配件次品率、成品检测以及用户退回情况等，重点分析设计高效的抽样检测方案，以控制零配件和成品的次品率，降低生产成本，并探讨对不合格品的处理策略，帮助企业进行合理决策。

对于问题一，本文建立以二项分布为基础的概率统计模型，基于正态近似推导样本量公式，求解临界值，并通过序贯检验进一步尝试减少检验次数。在置信水平分别为 95% 和 90% 的情况下，设置**容许误差 d 为 0.02**，得到所需的**最小样本量 n 分别为 609 和 370**。通过二项分布的累积分布函数，计算出决定拒绝零件需要检测出的**次品数量临界值 c 分别为 73 和 44**。此外，在序贯检验的基础上, 动态调整检测次数。**蒙特卡洛模拟**结果显示，当实际次品率在标称值上下浮动的情境下，该方法可显著减少实际检测次数，提高检验效率。

对于问题二，本文将零件生产问题建模为一个分支决策过程，构建了状态转移框架，建立了以**成本期望值**为目标函数的优化模型，其中成本包括零件采购、零件和成品检测、成品装配及潜在的拆解和调换成本等多项费用，还考虑了用户退货概率的影响。通过设定了四个决策变量（是否检测零件 1、零件 2 和成品、是否拆解成品），使用 **0-1 规划**，采用**暴力搜索法**对可能的决策组合进行求解, 求解结果见表 (1)。除了确定问题二中不同情况的决策方案，本文通过调整零件成本进行实验，发现模型在**单价较低时选择检测**，而在**单价较高时选择拆解**不合格成品。该模型有效反映了不同成本条件下的最优决策，可为实际生产提供参考。

对于问题三，对于表格给出的情况，本文在模型二的基础上，先将三个半成品的加工过程视为三个独立的成品制造过程，再将半成品视作零件来考虑成品制造过程。从最终成品的销售分析逐渐过渡到零件的检测和制造。利用 0-1 规划对模型进行优化，求解结果见表 (2)，该决策组合对应的正品成品的期望成本为 **116.524** 元。扩展到更复杂的情形，例如 m 道工序、n 个零配件的情况，只需要多次重复调用模型三，即可求解出最优决策。

对于问题四，本文将抽样检测的不确定性纳入决策影响因素，考虑利用样本方差和均值估计总体次品率分布的统计问题。将零件总体视作一个服从某种分布的随机变量，在估计出总体次品率的置信区间后，通过均匀采样和高斯采样生成多组次品率，并用模型求解出每组次品率对应的决策方案，再采用**最大后验估计（MAP）**原则得出最优决策方案。在样本量为 25，样本均值为 0.1，标准差为 0.07，信度为 95% 的情况下，进行 **100 组**采样，决策结果与问题二和问题三的决策一致，显示了在环境波动下决策方案的稳定性。

关键字：概率统计　假设检验　0-1 规划　搜索优化　后验估计

一、 问题重述

1.1 问题背景

在生产制造过程中，同时确保产品的质量且控制产品生产成本是企业长期保持竞争力的关键因素。然而，实际生产过程中存在着复杂的质量控制问题。例如，即使两个零配件均合格，由于装配过程中的不确定性，成品仍然可能存在缺陷。此外，不合格的成品可以通过拆解回收零配件，但这一过程会产生额外成本。因此，企业如何在采购、检测、装配、以及次品处理中优化决策，降低次品率和运营成本对企业至关重要。基于上述背景，需要解决下面四个问题：

1.2 问题要求

问题 1: 供应商提供的零配件存在一定的次品率，需要在控制成本的前提下，设计出检测次数最少的抽样检测方案，比较实际次品率和标称值的大小，以帮助企业决定是否接收这批零配件。

问题 2: 在已知两种零配件及成品的次品率情况下，企业有如下环节需要考虑：是否对零配件进行检测，如何处置不合格零配件，是否对成品进行检测，如何处理不合格成品。此外，还需要考虑用户退回造成的其它损失。结合上述情况，需要帮助企业对生产环节进行优化决策。

问题 3: 在生产过程中引入更多的工序和零配件，需要在已知各零配件、半成品及成品的次品率的基础上，扩展问题 2 中的决策模型，帮助企业进行更复杂的决策。

问题 4: 假设上述问题中的各类次品率均是通过抽样检测方法获得的，需要重新运用决策模型，帮助企业在增加了不确定性的情况下有效地控制生产成本和质量。

二、 问题分析

2.1 问题一分析

对于问题一，需要在尽可能少的检测次数下，通过抽样检测估计次品率，从而来决定是否接受供应商提供的零配件。当样本量 n 较大时，可以通过正态分布来近似样本比例的分布，从而简化计算。根据已知的的置信水平，可以得到其相应的显著性水平 α, 结合次品率标称值 p, 就可以对样本次品率进行单边区间估计。在本问中 n 未知的情况下, 设定允许的估计误差为 E, 则可以反推出所需的样本数量 n。

2.2 问题二分析

对于问题二，需要在企业生产过程中对零配件和成品进行决策，以降低总体成本并提高生产效率。决策包括是否检测零配件、是否检测成品、是否对不合格成品进行拆解及处理退回品的方式。可以构建一个分支决策过程模型，考虑零配件和成品的次品率、检测成本、装配成本、拆解成本及调换损失。通过建立目标函数最小化成本，并对决策变量进行 0-1 规划，求得最佳决策方案。

2.3 问题三分析

对于问题三，需要基于问题二进一步优化生产成本。可以通过适当对模型二进行改动，将问题三转化为连续两次使用模型二进行优化求解。注意到问题三中数据的特殊性，可以将半成品一和半成品二的制作过程进行等效。考虑所有可能的检测和拆解情景，通过建立目标函数最小化成本，并对决策变量进行 0-1 规划，求得最佳决策方案。

2.4 问题四分析

对于问题四，次品率不再是确定的，而是通过抽样检测获得的不确定性数据。因此，除了最小化成本外，还需考虑抽样误差对决策的影响。可以通过引入样本方差和均值来量化不确定性，并使用置信区间估计次品率的波动范围，再在该区间内均匀采样和高斯采样两种方法生成次品率样本，并代入原有模型进行决策分析。在得到的多组样本中，设计合理的方案来选择最优决策。

整体建模以及求解的工作如图 (1) 所示。

三、 模型假设

本文做出以下假设：

- **假设 1:** 在第一问中，假设总体中零件数量足够多。在进行抽样检测时，可视作无限总体抽样。
- **假设 2:** 可以采购到的零件数量足够多，不会出现零件功能不足的问题。
- **假设 3:** 拆解成品或半成品不会对零件造成损坏，而且零件库中的样本足够多，拆解放回的零件对总零件的次品率造成的影响可以忽略。
- **假设 4:** 生产出的产品全部可以卖出，不会出现滞销的情况。

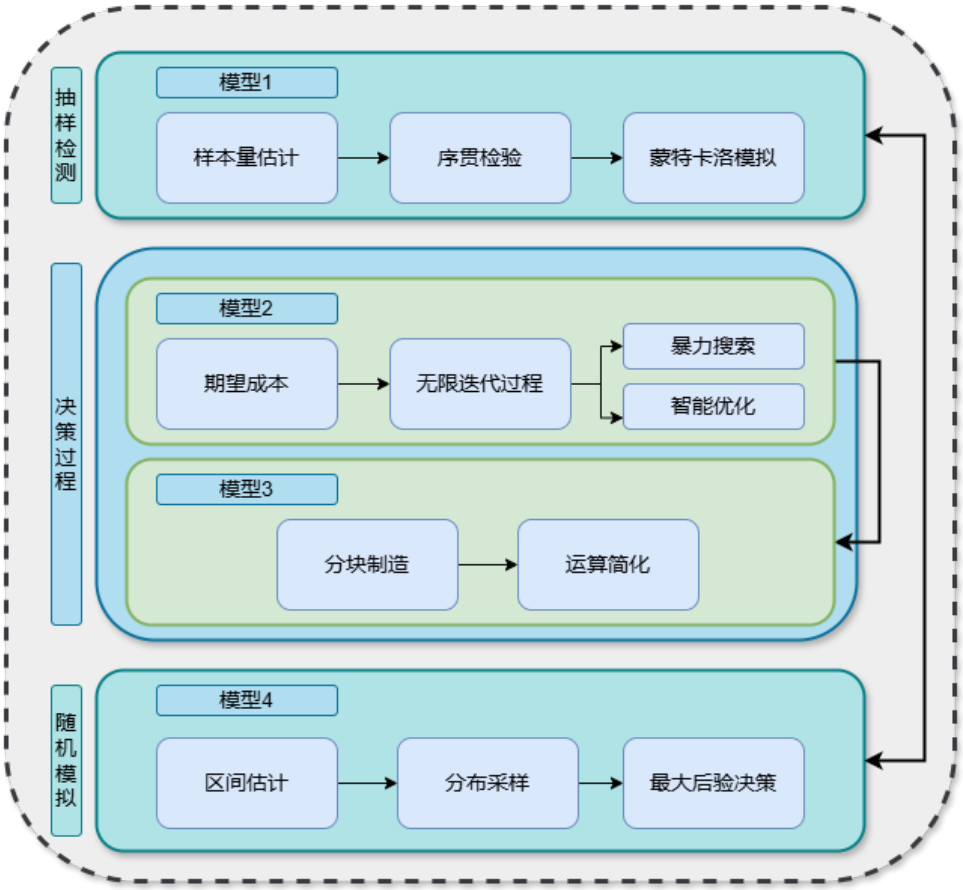


图 1　解题流程图

四、 符号说明

符号	说明
π	次品率标称值
Q	盈利
V	售价
$\mathcal{L}$	成本
D_i	决策变量
$\mathbb{E}[\cdot]$	期望算子
$\mathbb{O}(\cdot)$	符号算子
$x_{1:N}$	等价于 $x_1, x_2, \cdots x_N$
$\{p_i\}_{i=1}^{N}$	p_i 到 p_N 组成的集合

五、 问题一的模型的建立和求解

对于问题一，使用固定抽样和逐步抽样相结合的方案。先推导出一定容许误差下所需要的样本量，即最大检测次数。在此基础上，通过序贯检验使得在对所有样本进行检验之前，确定是否接受该批零件，从而进一步减少所需的检验次数。

5.1 模型建立

5.1.1 样本量确定

已知零配件次品率的标称值为 π，我们使用二项分布来模拟整个抽样过程。设从样本中抽取 n 个零配件，其中次品的数量为 X，则 $X \sim B(n, \pi)$。在这种情况下，样本中次品数量的期望值和方差分别为 $\mathbb{E}[X] = n\pi$ 和 $\mathbb{D}(X) = n\pi(1-\pi)$。

当样本量 n 足够大时，二项分布可以用正态分布来近似表示，即 $X \sim N(n\pi, n\pi(1-\pi))$。由于样本次品率 $p = \frac{X}{n}$，因此样本次品率 p 也服从正态分布，即 $p \sim N\left(\pi, \frac{\pi(1-\pi)}{n}\right)$，$\sigma_p = \sqrt{\frac{\pi(1-\pi)}{n}}$。将样本次品率进行标准化，我们可以得到随机变量 Z:

$$Z = \frac{p - \pi}{\sqrt{\pi(1-\pi)/n}} \sim N(0, 1) \tag{1}$$

由于只需要检验次品率是否小于等于标称值，因此，该问题可被视作是一个单侧置信区间的问题。设给定的置信水平为 $1-\alpha$，则可以通过样本次品率 p 对总体次品率进行单侧区间估计 [1]，且估计误差 ϵ 为:

$$\epsilon = z_\alpha \sqrt{\frac{\pi(1-\pi)}{n}} \tag{2}$$

其中，z_α 为标准正态分布的上 α 分位点。

在公式 (2) 中，估计误差仅由总体次品率 π 和样本量 n 决定。由于总体次品率已知，通过调整样本数量 n，可以得到不同的估计误差。样本量越大，其估计误差就越小。因此，确定检测次数尽可能少的抽样方案，需要确定可容许的最大误差。设容许误差 d，即此时 $\epsilon = d$，则可推导出样本数量:

$$n = \frac{z_\alpha^2 \pi(1-\pi)}{d^2} \tag{3}$$

5.1.2 临界值确定

前文中已提到样本中的次品数量服从二项分布。在此基础上，计算该二项分布的累积分布函数可以根据已知条件确定临界值 c，使得当在样本中检测到的次品个数大于（或小于等于）c 时，拒绝（或接受）该批零件。临界值确定的具体过程将在下文中模型求解部分详细叙述。

5.1.3 序贯检验优化

在得到临界值 c 后，即可通过序贯检验进行动态决策，对所需要的检验次数进行进一步的缩小。为此，我们进行如下设置：

- n： 抽样的上限数量，即样本总数。
- n_i： 第 i 批次抽样的数量。
- $n^k_{\text{total}} = \sum_{i=1}^{k} x_i$： 第 k 批次检测完后，累计检测的产品数量。
- x_i： 第 i 批次抽样中检测到的次品数量。
- $x^k_{\text{total}} = \sum_{i=1}^{k} x_i$： 第 k 批次检测完后，累计检测到的次品数量。
- c： 接受该批产品的情况下，允许检测到次品数量的最大值。
- **停止条件 1:** 当 $x^i_{\text{total}} \geq c$，即第 i 轮结束已检测到的次品数超过临界值时。
- **停止条件 2:** 当 $n - n^i_{\text{total}} < c - x^i_{\text{total}}$, 即第 i 轮结束时，即使剩下的零件全为次品，检测到的次品总数也无法超过临界值。

在上述设置的基础上，可以进行如下的**动态决策**：

该动态决策过程可以进一步减少需要检验的样品次数。最终检测的样本数量 n^* 取决于实际的抽样情况，并不是一个定值，但是我们可以估计 n^* 的期望，即：

$$\mathbb{E}[n^*] = \sum_{n^*=1}^{n} = n^* \cdot P(n^*) \tag{4}$$

其中 $P(N)$ 为在第 N 次检验后停止检验的概率。实际检测次数的期望和真实的次品率有关。通过设定一个真实次品率的近似值，我们可以用蒙特卡洛模拟法估计该期望。

5.2 模型求解

该部分中，我们将通过给定的次品率标称 $\pi = 10\%$，分别针对两种情况进行求解。对于给定的置信水平 $1 - \alpha$, 可作出如下设置：

- H_0 : 次品率 $p \leq 10\%$
- H_1 : 次品率 $p > 10\%$
- 给定置信度对应的显著性水平 α。在该显著性水平下，样本中次品率的估计值显著高于标称值，则拒绝零假设 H_0 ，接受备择假设 H_1。
- 设置容许误差为总体次品率的 0.2 倍，即 $d = 0.02$。

Algorithm 1 零配件检测算法

输入: 总样本量 c, 条件 1 和条件 2
初始化: $i \leftarrow 1, x_{\text{total}} \leftarrow 0$
while 检测未结束 **do**
    if $i = 1$ **then**
        抽取 $n_i = c$ 个样本
    else
        抽取 $n_i = c - \sum_{j=1}^{i-1} x_j$ 个样本
    end if
    检测第 i 批次的样本，记录不合格品数量 x_i
    更新累计监测到的次品数量 $x_{\text{total}} \leftarrow x_{\text{total}} + x_i$
    if 满足条件 1 **then**
        停止检测，拒收该批零配件
        break
    else if 满足条件 2 **then**
        停止检测，接收该批零配件
        break
    else
        更新 $i \leftarrow i + 1$
    end if
end while

5.2.1 情况一

样本量确定: 对于情况一，给定置信水平为 95%，则 $\alpha = 0.05$。易知标准正态分布的上 0.05 分位点为 1.644854, 取 $z_\alpha = 1.645$，代入公式 (3) 可得:

$$n = \frac{1.645^2 \times 0.1 \times (1 - 0.1)}{0.02^2}$$

解得 $n = 608.86$，由于样本数量应为正整数，所以应随机检测至少 609 个样本。

临界值确定: 此时，我们希望确定一个临界值 c，使得:

$$P(X \leq c) \geq 0.95$$

已知，次品数量 $X \sim B(n, \pi)$，则有如下的累积分布函数:

$$P(X \leq c) = \sum_{x=0}^{c} \binom{n}{x} \pi^x (1 - \pi)^{n-x} \geq 0.95 \tag{5}$$

将 $n = 609, \pi = 0.1$ 代入公式 (5), 可以求解出临界值整数 $c = 73$。

序贯检验：第一轮中，选取 73 个样本进行检验，之后重复5.1.3中的逐步决策过程，直至达成停止条件。事实上，实际检验次数取决于该零件的实际次品率。如果用标称值 π 近似实际次品率，通过蒙特卡洛模拟法，进行 10000 次模拟实验，可近似求解出实际检验次数的期望 $\mathbb{E}[n^*] \approx 600$。

当使用标称值 10% 近似实际次品率时，序贯检验对减少实际检测次数的效果有限。但当实际次品率在标称值上下进行浮动时，逐步检测则可以更加明显地减少实际检测次数。然而，实际操作过程并不会像蒙特卡洛方法一样进行多次实验。在实际的单次检验过程中，序贯检验仍存在大幅度减少检测次数的可能性。关于实际次品率和检验次数期望的更详细信息和模型改进请见9.1。

5.2.2 情况二

样本量确定：对于情况二，给定置信水平为 90%, 则 $\alpha = 0.1$。易知标准正态分布的上 0.1 分位点为 1.281552, 取 $z_\alpha = 1.282$, 代入公式 (3) 可得:

$$n = \frac{1.282^2 \times 0.1 \times (1 - 0.1)}{0.02^2}$$

解得 $n = 369.79$，由于样本数量应为正整数，所以应随机检测至少 370 个样本。

临界值确定: 与情况一类似，可求解出情况二中的临界值 $c = 44$。

序贯检验：第一轮中，选取 44 个样本进行检验，之后重复5.1.3中的逐步决策过程，直至达成停止条件。如果用标称值 π 近似实际次品率，通过蒙特卡洛模拟法可近似求解出实际检验次数的期望 $\mathbb{E}[n^*] \approx 363$。

六、 问题二的模型的建立和求解

6.1 模型建立

问题二所述的零件的生产问题可抽象为如下所示的决策过程:

$$s_0 \xrightarrow{\pi_0} s_1 \xrightarrow{\pi_1} s_2 \xrightarrow{\cdots} \cdots \xrightarrow{\cdots} \longrightarrow s_T \tag{6}$$

在公式 (6) 所示的决策过程框架中，s_i 表示状态，s_T 表示最终状态，π_i 表示在当前状态下所采取的决策。以该框架为基础，将问题二建立成如 (2) 所示的分支决策过程。我们做出如下设置: 其中，圆圈内表示状态，箭头上表示决策或一些注释，C 表示检测，CN 表示不检测，T 表示通过，F 表示不通过，D 表示是否分解，P 表示概率转移。

假设零件充足，且生产出的产品无滞销情况。为帮助公司进行决策，可先考虑最大化单个成品售卖成功的利润 Q。

$$\max Q = V - \mathcal{L} \tag{7}$$

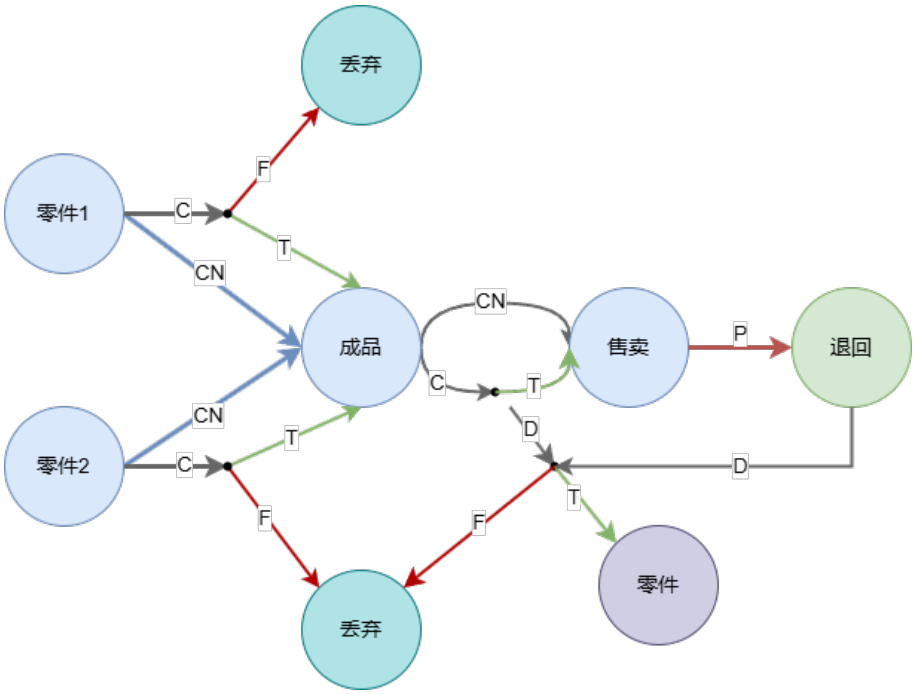


图 2　问题二决策过程

其中,V 为单个成品的售价，$\mathcal{L}$ 为成本，可进一步表示为:

$$\mathcal{L} = \ell_{\text{零件采购}} + \ell_{\text{零件检测}} + \ell_{\text{成品检测}} + \ell_{\text{零件装配}} \tag{8}$$

由于 V 为定值，上述问题可被转化为极小化成本的问题。为进一步明确该问题，我们作出如下设置:

- 零件 1、零件 2 的采购成本为 b_1, b_2。
- 零件 1、零件 2 以及成品的检测成本为 C_1, C_2。
- 成品的装配成本 b_3, 检测成本为 C_3，拆解成本与调换成本分别为 C_4, C_5。
- 零件 1、零件 2 以及成品的次品率为 p_1, p_2, p_3, 用户收到不合格商品后退回的概率为 p_c。
- 是否检测零件 1、是否检测零件 2、是否检测成品、是否对不合格成品进行回收分别为四个决策变量 D_1, D_2, D_3, D_4。如果确定检测（或回收），则对应变量为 1，反之为 0。

由于零件和成品存在一定的次品率，成本 $\mathcal{L}$ 可被视作一个随机变量。为此，可以选择通过对成本均值 $\mathbb{E}[\mathcal{L}]$ 进行优化。经过常量分离后的目标函数如公式 (6.1) 所示:

$$\min \mathbb{E}[\mathcal{L}] = \mathbb{E}[\Delta\ell_{\text{零件采购}}] + \mathbb{E}[\ell_{\text{零件检测}}] + \mathbb{E}[\ell_{\text{成品检测}}] + \mathbb{E}[\Delta\ell_{\text{零件装配}}] + b_1 + b_2 + b_3 \tag{9}$$

其中，首次产品采购成本和成品装配成本 b_1, b_2, b_3 被单独列出，$\mathbb{E}[\Delta\ell_{\text{零件采购}}]$ 和 $\mathbb{E}[\Delta\ell_{\text{零件装配}}]$ 则表示除首次采购和装配之外可能存在的零件采购和成品装配成本。为

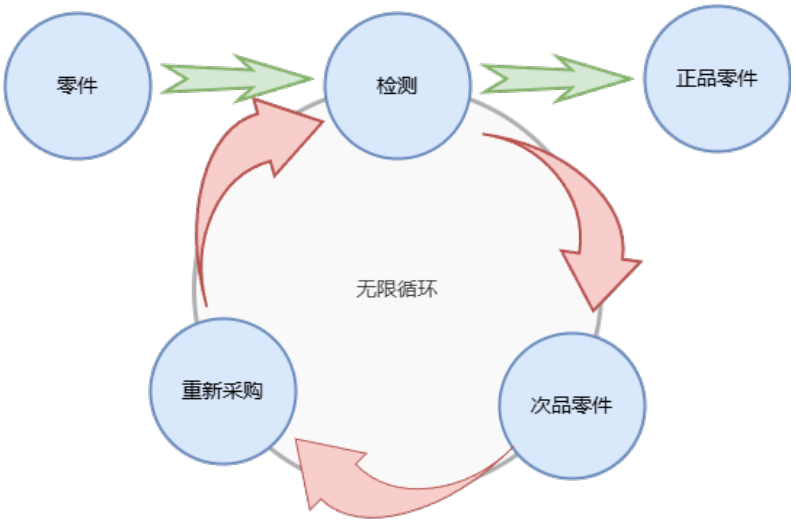


图 3　零件检测迭代过程

便于叙述，下文中称作额外的零件采购和成品装配成本。进一步考虑，额外的零件采购成本 $\mathbb{E}[\Delta\ell_{\text{零件采购}}]$ 可划分为两类：

1. 因为零件被**检测为不合格**，需要额外采购零件的成本，记为 $\mathbb{E}[\Delta\ell_{\text{检测采购}}]$。
2. 因为不合格成品**被丢弃**，需要额外采购零件的成本，记为 $\mathbb{E}[\Delta\ell_{\text{丢弃采购}}]$。

则额外的零件采购成本可表示为：

$$\mathbb{E}[\Delta\ell_{\text{零件采购}}] = \mathbb{E}[\Delta\ell_{\text{检测采购}}] + \mathbb{E}[\Delta\ell_{\text{丢弃采购}}]$$

下面，我们对目标函数进行更细致的分析。

6.1.1 零件检测

如果不对零件进行检测，该步骤不产生成本；而如果进行检测，则可能会发现次品，则需要采购新的零件进行制作。这一过程产生了上文提到的期望检测成本 $\mathbb{E}[\Delta\ell_{\text{检测采购}}]$。此外，使用新的零件后，还需要对其进行再次检测，而新的零件也可能存在次品，从而形成一个迭代过程，见图 (3)。该迭代过程可用以下表达式表示：

$$\mathbb{E}\left[\ell_{\text{零件检测}}\right] + \mathbb{E}[\Delta\ell_{\text{检测采购}}] = \sum_{i=1}^{2} D_i \left(C_i + p_i \left(b_i + C_i + p_i \left(b_i + C_i + p_i(\cdots)\right)\right)\right) \tag{10}$$

对该迭代过程求极限行为，有

$$\mathbb{E}\left[\ell_{\text{零件检测}}\right] + \mathbb{E}[\Delta\ell_{\text{检测采购}}] = D_1\left(C_1 + \frac{p_1\left(b_1 + C_1\right)}{1 - p_1}\right) + D_2\left(C_2 + \frac{p_2\left(b_2 + C_2\right)}{1 - p_2}\right) \tag{11}$$

6.1.2 成品检测

成品检测 $\mathbb{E}[\ell_{\text{成品检测}}]$ 同样可被分为两部分：

1. 对组装好的的成品进行检测。
2. 对用户退回的成品进行检测。

对用户退回的成品检测

如果成品检测未通过，则需要选择是否对当前成品拆解。如果选择不拆解而是直接丢弃，则需要再采购新的零件并完成成品组装，即产生 $\mathbb{E}[\Delta\ell_{\text{丢弃采购}}]$ 和 $\mathbb{E}[\Delta\ell_{\text{零件装配}}]$。该部分成本均值可表示为下式：

$$\mathbb{E}\left[\ell_{\text{成品检测}} + \Delta\ell_{\text{丢弃采购}} + \Delta\ell_{\text{零件装配}}\right] = D_3(C_3 + p_t\mathbb{E}[\ell_{\text{重做}}]) + (1 - D_3)p_t p_c(C_5 + \mathbb{E}[\ell_{\text{重做}}]) \tag{12}$$

其中，p_t 表示成品为次品的概率（不同于成品次品率)，$\mathbb{E}[\ell_{\text{重做}}]$ 为重做成品的成本。该公式表示：1）当检测成品时，需要检测成本。如果成品被检测为次品，则需重做成品；2）当不检测成品时，如果该成品是次品的且用户选择退还时，商家则需要支付调换损失并重做成品。6.1.3将对重做成品成本 $\mathbb{E}[\ell_{\text{重做}}]$ 进行进一步分析。

此外，成品为次品的概率 p_t 可在讨论零件检测情况的基础上，通过加法公式进行表示：

$$\begin{aligned} p_t =& D_1D_2p_3 + (1 - D_1)D_2[1 - (1 - p_1)(1 - p_3)] + D_1(1 - D_2)[1 - (1 - p_2)(1 - p_3)] \\ &+ (1 - D_1)(1 - D_2)[1 - (1 - p_1)(1 - p_2)(1 - p_3)] \end{aligned} \tag{13}$$

6.1.3 重做成本

当成品不合格时，需要按如图 (4) 所示的工作流程重新制造成品。考虑决策变量 D_4: 如果不拆解不合格成品，则只需要考虑制造新成品的成本。如果拆解不合格品，则需要考虑拆解费用，以及制作新成品的成本, 因为已经进行拆解，零件已经返回库中，所以需要在成本中减去零件采购的费用。重做成品的成本为:

$$\mathbb{E}\left[\ell_{\text{重做}}\right] = D_4[C_4 + \mathcal{L} - b_1 - b_2] + (1 - D_4)\mathcal{L} \tag{14}$$

6.2 模型求解

在上文分析的基础上，将上述公式进行线性组合得到符号表达式，并使用 Mathematica 软件进行处理，并得到损失函数的解析表达形式 (见附录C代码部分)，该优化问题为四个决策变量的 0-1 规划，即：

$$D_{1:4} = \underset{D_{1:4}}{\arg\min}\,\mathbb{E}[\mathcal{L}] \tag{15}$$

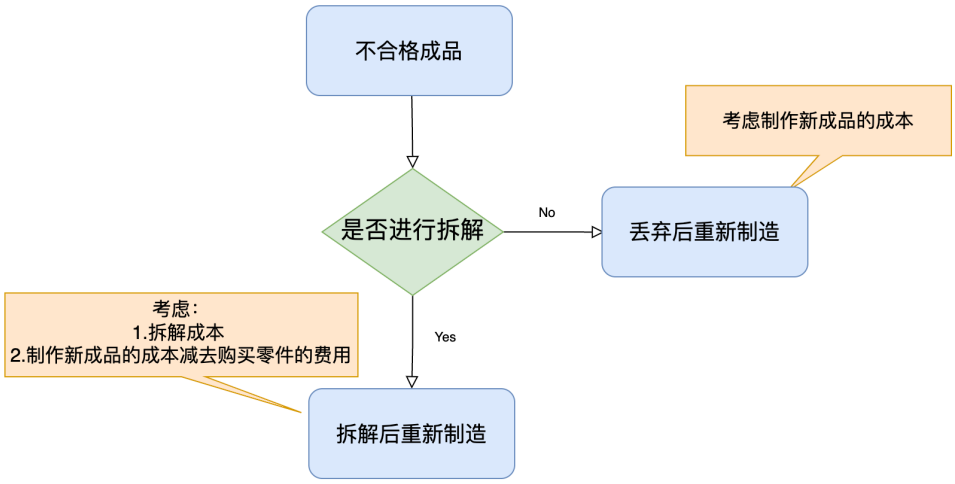


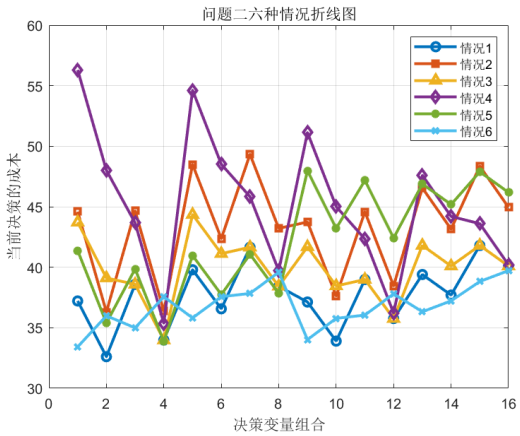
图 4　重做成品流程

采用暴力搜索算法对 16 种决策组合进行全面求解。注意，在公式 (14) 中，公式右侧部分存在损失函数 $\mathcal{L}$，导致最终的损失函数会以 $\mathcal{L} = f(\mathcal{L})$ 的形式出现，即出现递归。我们使用 Mathematica 的 $Solve$ 函数对其进行求解，发现该方程不可解。因此，在本问的优化中，令公式 (14) 中的损失函数为 $b_1 + b_2 + b_3$，为了方便描述，我们将公式右侧的 $\mathcal{L}$ 记作 L_1, 即此时 $\mathcal{L} = f(\mathcal{L}_1)$,$\mathcal{L}_1 = b_1 + b_2 + b_3$, 用以模拟最为简单的成品制造情况。在模型的分析与改进中，我们会对公式中的损失函数进行精确化，以证实模型目标函数在不同情况下的有效性。另外，考虑极端情况下的保守策略，对用户退回不合格产品的概率 p_c 设置为 1。针对问题二的六种情形，模型优化结果以及总成本如表 (1) 所示，表中“1”表示“是”，“0”表示“否”。

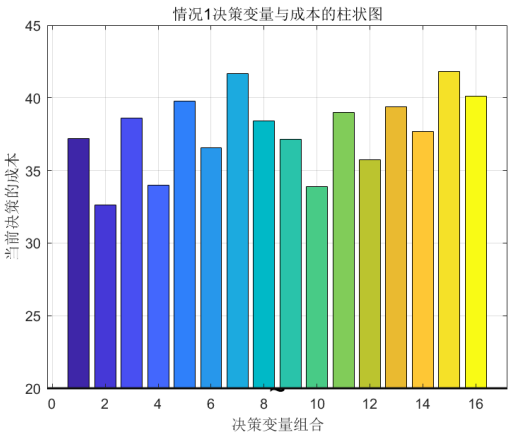
为更直观的展示决策变量与期望成本的变化情况，图 (5a) 绘制出了 16 种决策变量组合方式对应的所有情况期望成本，图 (5b) 展示出了绘制了情景 1 中 16 种决策组合所对应的期望成本。此外我们还使用智能优化算法对该问的 0-1 规划进行了求解，详情请参考支撑文件。

表 1　问题二决策表

情况	检测零件 1	检测零件 2	检测成品	拆解成品	成本 (元)
情况 1	0	0	0	1	32.607
情况 2	0	0	0	1	36.296
情况 3	0	0	1	1	33.981
情况 4	0	0	1	1	35.368
情况 5	0	0	1	1	33.872
情况 6	0	0	0	0	33.419



(a) 问题二六种情况折线图



(b) 问题二情况 1 期望成本柱状图

图 5　问题二结果图像

七、 问题三的模型的建立和求解

7.1 模型建立

基于模型二的分支决策过程，将更为复杂的问题三进行分步考虑: 先将三个半成品的制作看作三个独立的成品制作过程，再将制作好的半成品视作零件，此时该半成品的次品率与是否检测零件的决策变量相关，并非题目中给出的次品率。此外，还需要考虑半成品的拆解和丢弃过程。本部分将从整个成品的销售开始分析，逐渐过渡到零件的检测部分，我们画出了问题三的生产流程图，因为问题三比问题二复杂，为了直观的展示生产流程，对生产流程中的细节进行了简化，见图 (6)。

与问题二类似，考虑最小化成功生产单个正品成品进行销售的成本，其包括：组装成品前对半成品进行检测的成本，对成品检测的成本，成品装配成本和制造半成品的成本。建立目标函数如下:

$$\min \mathbb{E}[\mathcal{L}] = \mathbb{E}[\ell_{\text{半成品检测}}] + \mathbb{E}[\ell_{\text{成品检测}}] + \mathbb{E}[\ell_{\text{成品装配}}] + \mathbb{E}[\ell_{\text{半成品制造}}] \tag{16}$$

公式中，$\mathbb{E}[\Delta\ell_{\text{半成品制造}}]$，是除首次制造外额外的半成品制造成本，可表示为：$\mathbb{E}[\Delta\ell_{\text{半成品制造}}] = \mathbb{E}[\Delta\ell_{\text{检测制造}}] + \mathbb{E}[\Delta\ell_{\text{丢弃制造}}]$。其中包含两种情况，分别为:

1. 制作成品过程中，半成品被检测不合格而需要重新制造带来的成本，记为 $\mathbb{E}[\Delta\ell_{\text{检测制造}}]$。
2. 成品检为不合格被丢弃，需要重新制造的半成品带来的成本，记为 $\mathbb{E}[\Delta\ell_{\text{丢弃制造}}]$。

参照模型二的思路，对该目标函数进行分离得:

$$\min \mathbb{E}[\mathcal{L}] = \mathbb{E}[\ell_{\text{半成品检测}}] + \mathbb{E}[\ell_{\text{成品检测}}] + \mathbb{E}[\Delta\ell_{\text{成品装配}}] + \mathbb{E}[\Delta\ell_{\text{半成品制造}}] + \mathbb{E}[\mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3] + e_4 \tag{17}$$

其中，e_4 为成品的装配成本，$\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ 分别为制造出半成品 1，2，3 所需要的成本 (该成本是制造出一个半成品的成本，而不是制造出一个正品半成品的成本，因为在制造成

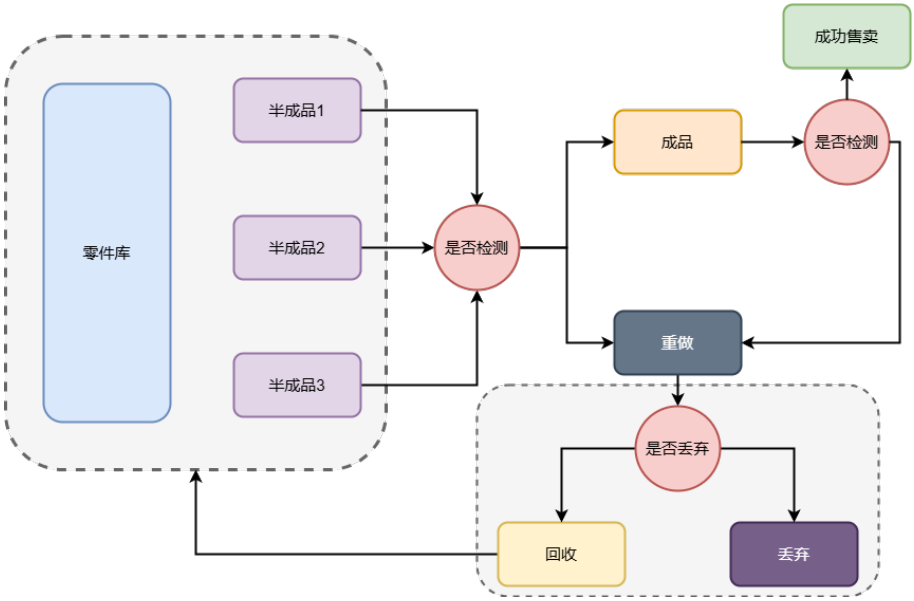


图 6　问题三生产流程图

品的环节之前并没有对半成品进行检测)。

7.1.1 成品组装过程中的半成品检测

在对成品组装过程中，可能需要对半成品进行检测。如果检测出不合格的半成品，则需要额外制造半成品。二者成本的期望之和为:

$$\mathbb{E}[\Delta\ell_{\text{半成品检测}}] + \mathbb{E}[\Delta\ell_{\text{检测制造}}] = \sum_{i=A}^{C} D_i(C_i + p^i(C_i + \mathbb{E}[\ell_{\text{半成品 i 重做}}] + p^i(\cdots))) \tag{18}$$

为更加清晰地描述问题，我们作出如下解释:

- $\sum_{i=A}^{C}$ 表示分别对 i 赋值 A, B, C 并对全式进行求和。
- D_i 为是否检测半成品 i 的决策变量。
- C_A, C_B, C_C 为检测半成品 $1, 2, 3$ 所需要的检测费用。
- p^i 为半成品 i 为次品的概率。
- $\mathbb{E}[\ell_{\text{半成品 i 重做}}]$ 为重做一个可能是次品的半成品 i 所需要的期望成本。

该公式表示：如果对半成品 i 进行检测，则需要一次的检测成本 C_i。由于半成品 i 存在一定的概率 p^i 为次品，如果半成品为次品的话，需要决定是否当前不合格半成品进行重做 (考虑拆解)。重做的半成品还可能是次品，于是就出现了公式中所示的无限循环过程。该流程如图 (7) 所示，对该无限循环递归过程求极限，有:

$$\mathbb{E}[\Delta\ell_{\text{半成品检测}}] + \mathbb{E}[\Delta\ell_{\text{检测制造}}] = \sum_{i=A}^{C} D_i \left(C_i + \frac{p^i(C_A + \mathbb{E}[\ell_{\text{半成品 i 重做}}])}{1 - p^i} \right) \tag{19}$$

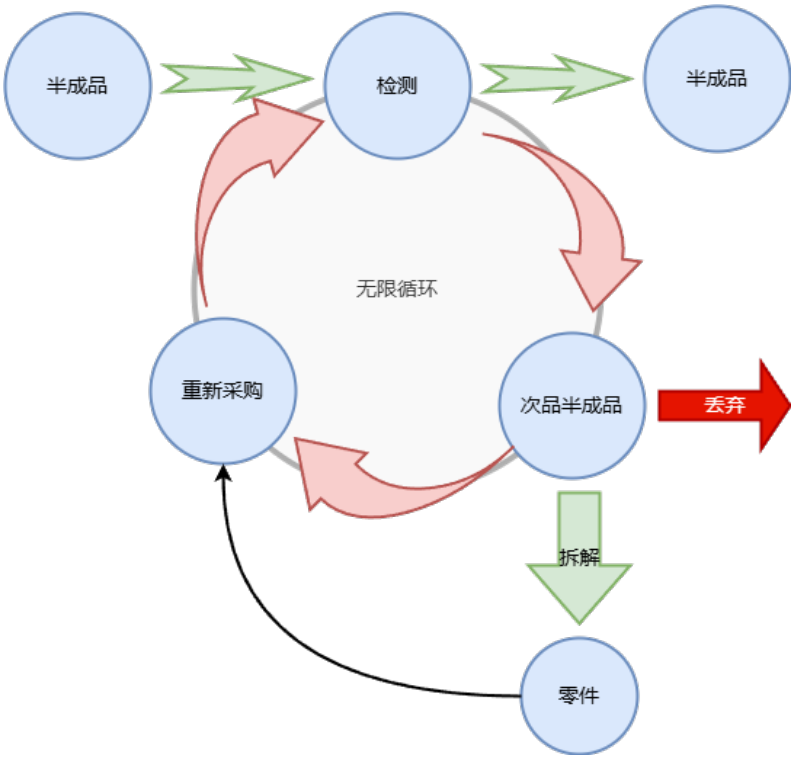


图 7　半成品检测过程

现在给出重做半成品的期望成本, 以重做半成品 1 举例，其期望成本为：

$$\mathbb{E}[\ell_{\text{半成品 1 重做}}] = D_a(d_1 + \mathbb{E}[\mathcal{L}_1] - b_1 - b_2 - b_3) + (1 - D_a)\mathbb{E}[\mathcal{L}_1] \tag{20}$$

其中，D_a 为是否进行半成品 1 拆解的决策变量，d_1 为半成品 1 的拆解成本，b_1, b_2, b_3 为零件 1，2，3 的采购成本，该公式的含义为：如果对半成品 1 进行拆解，首先需要拆解成本 d_1，后续生产半成品 1 需要新的成本 $\mathbb{E}[\mathcal{L}]$，因为拆解了不合格的半成品，拆下的零件可以返回零件库，所以在制造新的半成品 1 时，并不需要新的零件采购成本，于是减去零件采购成本 b_1, b_2, b_3；如果不进行零件拆解的话，需要的重做成本就是制造半成品 1 的期望成本。

7.1.2 成品检测

在对成品检测的的过程中，可能检测出不合格的成品，如果对不合格的成品采取丢弃处理的话，会产生上文中提到的丢弃采购 $\mathbb{E}[\Delta\ell_{\text{丢弃制造}}]$, 在对成品进行重做的过程中，还会产生多余的成品装配成本，即 $\mathbb{E}[\Delta\ell_{\text{成品装配}}]$, 该过程可以用以下公式表示：

$$\begin{aligned}\mathbb{E}[\Delta\ell_{\text{成品检测}}] + \mathbb{E}[\Delta\ell_{\text{丢弃制造}}] + \mathbb{E}[\Delta\ell_{\text{成品组装}}] = \\ D_D(C_D + p^4\mathbb{E}[\ell_{\text{成品重做}}]) + (1 - D_D)p^4 p_c(\mathbb{E}[\ell_{\text{成品重做}}] + C_9)\end{aligned} \tag{21}$$

其中，D_D 为是否检测成品的决策变量，C_D 为成品的检测成本，p^4 为成品为次品的概率，p_c 为用户退回不合格成品的概率，C_9 为成品调换损失，该公式的含义为：如果对

成品进行检测，首先需要检测成本，当检测出成品不合格时，需要对成品进行重做，公式 (21) 没有出现与公式 (18) 一样的重做成本期望无限递归的形式，是因为在最初的定义中，目标函数 $\mathcal{L}$ 就是制作出一个正品的成本，而不是制作出一个成品的成本，这与 $\mathcal{L}_1$ 的定义不同；如果不对成品进行检测，就可能卖出不合格成品并且被用户退回，需要支付调换损失和成品重做期望成本。成品重做期望成本的表达式如下所示：

$$\mathbb{E}[\ell_{\text{成品重做}}] = \mathbb{E}\left[D_d\left(\mathcal{L} - \mathcal{L}_1 - \mathcal{L}_2 - \mathcal{L}_3 + d_4\right) + (1 - D_d)\mathcal{L}\right] \tag{22}$$

其中，D_d 为是否拆拆解成品的决策变量，$\mathcal{L}$ 为本身的目标函数，即成功制作出一个正品成品所需要的成本，$\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ 为生产半成品 1、2、3 所需要的成本，d_4 为拆解成本。该公式的含义为：若拆解成品，首先需要拆解费用，然后需要制作一个新成品的成本，由于对不合格成品已经拆解，半成品 1，2，3 已经回收到了零件库中，并不需要制造额外的半成品；如果不拆解成品，则需要直接花费成品的期望成本。

7.1.3 半成品制造

在问题三模型建立之前，把三个半成品的加工看作三个独立的零件，现在来分析这三个零件的 “采购价格”，即它们的制造成本。由于制造几种半成品的分析方式类似，这里以制造半成品 1 来举例，半成品 1 的制造成本可以建模为：

$$\mathcal{L}_1 = b_1 + b_2 + b_3 + e_1 + \Delta\ell_{\text{零件采购}} + \ell_{\text{零件检测}} \tag{23}$$

其中，b_1, b_2, b_3 分别为零件 1、2、3 的采购价格，$\ell_{\text{零件检测}}$ 为检测三种零件的总成本，$\Delta\ell_{\text{零件采购}}$ 为因零件检测产生的多余采购成本，该公式并没有提到多余的零件装配成本，这是因为在制造半成品的过程中，并不会对半成品进行检测 (对半成品的检测在成品制造过程中考虑)，所以制造出来的半成品可能是次品，而且该半成品只能被装配一次，即半成品 1 的装配成本为 e_1。

零件检测和额外零件采购期望成本之和可以表示为：

$$\mathbb{E}[\Delta\ell_{\text{零件采购}} + \ell_{\text{零件检测}}] = \sum_{i=1}^{3} D_i(C_i + p_i(C_i + b_i + p_i(\cdots))) \tag{24}$$

其中，D_i 为是否检测零件 i 的决策变量，C_i 为检测成本，p_i 为零件 i 为次品的概率，b_i 为采购零件 i 的成本，该期望成本的计算过程已在上文中多次提到，可以求得其极限状态为：

$$\mathbb{E}[\Delta\ell_{\text{零件采购}} + \ell_{\text{零件检测}}] = \sum_{i=1}^{3} D_i\left(C_i + \frac{p_i(C_i + b_i)}{1 - p_i}\right) \tag{25}$$

7.1.4 次品概率的计算

上文使用的概率符号 p^1, p^2, p^3, p^4，依次为半成品 1、2、3 和成品的次品概率 (考虑了决策变量之后的次品概率)，这几种次品概率的计算方式都一致。此处以计算半成品

的次品的概率 p^3 为例，因为该半成品由两个零件装配，描述最为简单，该概率可以用公式表示为：

$$\begin{aligned} p^3 =& D_7 D_8 p_C + (1-D_7) D_8 \left(1-(1-p_7)(1-p_C)\right) + D_7 (1-D_8) \left(1-(1-p_8)(1-p_C)\right) \\ &+ (1-D_7)(1-D_8) \left(1-(1-p_8)(1-p_7)(1-p_C)\right) \end{aligned} \tag{26}$$

其中，D_7, D_8 分别为决策零件 7、8 是否检测的决策变量，p_7, p_8 分别为零件 7、8 的次品概率，p_C 为将正品零件装配为半成品 3 的次品率。该公式考虑了所有的检测情况：当零件 7、8 都被检测时，半成品 3 的次品概率为 p_C；当只有一个零件被检测时：例如只有零件 7 被检测时，这时半成品为正品的概率为 $(1-p_8)(1-p_C)$，即未检测的零件 8 为正品且装配正确，那么半成品为次品的概率为 $1-(1-p_8)(1-p_C)$；当零件 7、8 都未被检测时，半成品为次品的概率为 $1-(1-p_7)(1-p_8)(1-p_C)$。

7.2 模型求解

上文已对损失函数 (16) 中的所有变量进行分析，将其代入并进行 0-1 规划：

$$\{D_{1:4}, D_{A:D}\} = \underset{\{D_{1:4}, D_{A:D}\}}{\arg\min} \ \mathbb{E}[\mathcal{L}] \tag{27}$$

与问题二类似，公式 (22) 中也出现了递归形式，经 Mathematica 的 $Solve$ 函数求解发现该损失函数没有解析解。为简化求解，令公式右侧递归的 $\mathcal{L}$ 为最简单的成品制造形式进行计算，即公式右侧有 $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 + e_4$，$e_4$ 为成品装配成本。该公式描述了没有任何检测的成品制作情况，如此设置，我们便可以使用 Mathematica 对该损失函数进行计算，得到其解析形式 (其解析形式见附录 c 代码部分)。

现在着眼于问题三的具体情景，这里共有 16 种决策变量，将决策空间全部遍历完共需 2^{16} 次目标函数的计算，文章发现，半成品 1 与半成品 2 所需要的零件成本以及检测成本，自身的装配成本、检测成本、拆解成本都完全一致，那么，与半成品 1 和半成品 2 相关的最优决策必然一致：即相关的零件检测决策一致、半成品检测决策一致、半成品拆解决策一致，在损失函数中将这几个决策变量进行合并，会发现减少了 5 个决策变量，即现在只需要计算 2^{11} 次损失函数，就可以遍历完全部的决策空间。另外，考虑极端情况下的保守策略，对用户退回不合格产品的概率 p_c 设置为 1。通过这种简化方式，使用暴力搜索的方式对该损失函数进行优化，求得的最优决策如表 (2) 所示，其中“1”表示“是”，“0”表示“否”，在这种决策情况下，制作出正品成品所需要的期望成本为 116.524 元。

表 2　问题三决策表

决策变量	是/否	决策变量	是/否
是否检测零件 1	1	是否检测半成品 1	0
是否检测零件 2	1	是否检测半成品 2	0
是否检测零件 3	1	是否检测半成品 3	0
是否检测零件 4	1	是否检测成品	1
是否检测零件 5	1	是否拆解半成品 1	0
是否检测零件 6	1	是否拆解半成品 2	0
是否检测零件 7	1	是否拆解半成品 3	0
是否检测零件 8	1	是否拆解成品	1

此外，根据已经建立的问题三的零件-半成品-成品制造模型，对于 m 道工序、n 个零配件，已知零配件半成品和成品的次品率的抽象问题，也可以重复的利用该概率模型求解最终期望成本，从而进行规划得到最优决策方案。

八、 问题四的模型的建立和求解

8.1 模型建立

相对于问题二和问题三中确定的次品率，问题四中的次品率是通过抽样检测得到的。此时，最小化成本不再是确定决策的唯一量度，而是需要引入抽样检测的不确定性对决策造成的影响。

题目中并未给出明确的抽样检验情况，这为衡量不确定性造成的影响创建了一定的发挥空间。引入方差可以很好地量化该问题中隐含的不确定性。将总次品率视作服从某一分布的随机变量，通过样本方差和均值对总体进行估计。通过引入次品率的一个置信区间，可以进一步考虑不确定性对决策结果的影响，再在该区间中生成多组次品率样本分别进行决策，对结果进行统计分析后可以得出最优的决策方案。

假设本问题中抽样检验得到的样本均值 $\bar{p}$, 样本方差为 S^2。由于总体分布情况未知，在样本容量为 n 时，我们选取自由度为 $(n-1)$ 的 t 分布来估计总体的情况。

设置信水平为 $1-\alpha$, 可得:

$$P\left(-t_{\alpha/2,n-1} \leq \frac{\bar{p}-\mu}{S/\sqrt{n}} \leq t_{\alpha/2,n-1}\right) = 1-\alpha \tag{28}$$

对上述公式变形，可以得到对总体均值在 $1-\alpha$ 置信度下的一个估计区间:

$$\left(\bar{p} - t_{\alpha/2,n-1}\frac{S}{\sqrt{n}}, \bar{p} + t_{\alpha/2,n-1}\frac{S}{\sqrt{n}}\right) \tag{29}$$

设实际的次品率在该估计区间中进行浮动，则可以在该区间中进行对次品率采样。为此，我们设计两种采样方案：

方案 1: 在该区间中，在次品率的估计区间进行均匀采样，得到多组采样次品率。

方案 2: 设该区间为一个正态分布的三倍标准差区间，即 $(\mu-3\sigma,\mu+3\sigma)$，在该区间内进行高斯采样。

在采样出多组次品率后，将其分别代入问题二和三的模型中进行求解。可以求解出多个不同的决策方案。设第 i 个决策方案的第 j 个决策变量为 $\mathbf{a_{ij}}$, 设最终选取的第 j 个决策变量方案为 a_j，则选取策略可用下式表示：

$$a_j^* = \arg\max_{a_j} \sum_{i=1}^{m} \mathbb{O}(\mathbf{a}_{ij} = a_j) \tag{30}$$

其中，$\mathbb{O}(\cdot)$ 在括号内情况为“真”时取 1，括号内情况为“假”时取 0。该决策变量选取策略的直观意义就是对决策变量取众数，即贝叶斯推断中最大后验估计 (MAP) 意义下的最优估计，为模型提供明确的统计意义，可以很大程度的反应引入不确定性后的最优决策。

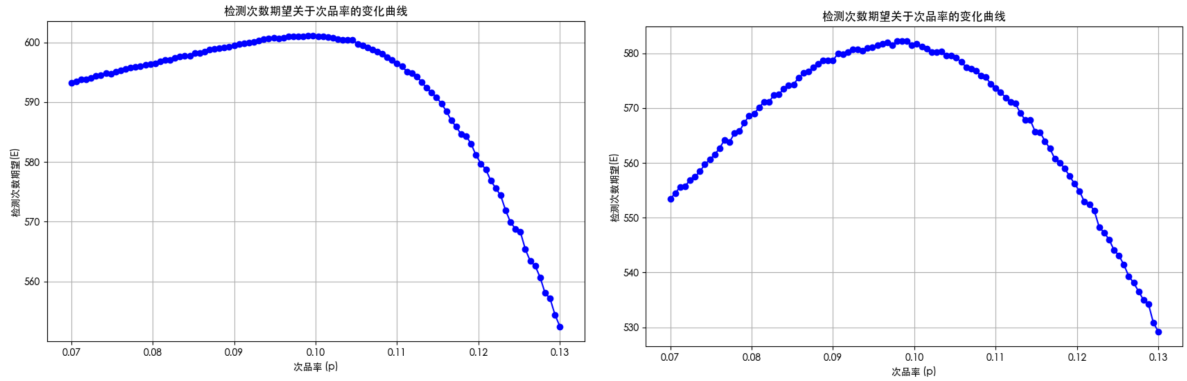
采样-决策的算法见 Algorithm 2，算法主体以均匀分布采样表示，括号内表示进行高斯采样所需要的额外信息，$\{p_i\}_{i=1}^n$ 表示生成的一组次品率，$\mathcal{U},\mathcal{N}$ 分别代表均匀分布和高斯分布。

Algorithm 2 采样-决策算法

1: **输入:** 估计区间 $[L,H]$, 采样数 n, 决策变量数量 N(如进行高斯采样，需要方差 σ^2)
2: $\{p_i\}_{i=1}^n \overset{\text{i.i.d}}{\sim} \mathcal{U}[L,H]$ $\quad \left(\{p_i\}_{i=1}^n \overset{\text{i.i.d}}{\sim} \mathcal{N}(\frac{L+H}{2},\sigma^2)\right)$
3: **while** 决策未结束 **do**
4: 对每一组次品率 p_i, 对成品制造过程进行决策 $\{a_j\}_{j=1}^N$
5: 记录最优决策变量
6: **end while**
7: 对所有的决策变量取最优估计 $a_j^* = \arg\max_{a_j} \sum_{i=1}^m \mathbb{O}(\mathbf{a}_{ij} = a_j)$
8: **输出:** 最优估计决策 $\{a_j^*\}_{j=1}^N$

8.2 模型求解

假设进行抽样检测的样本量 $n=25$, 样本均值 $\bar{p}=0.1$, 样本标准差 $S=0.07$。设置信度为 95%。根据公式（29）, 可以估计次品率的一个区间为：$(0.0711, 0.129)$。在算法中使用该估计区间，采样出 100 组不同的次品率，并使用模型进行决策。结果显示，最优决策方案与问题二、三中的决策结果相同，体现了模型在一定的环境波动下具有稳定性。为更清晰的展示采样决策过程，我们节选对问题二情况 1 进行**均匀采样-决策**的结果，展示在附录 A 表 (5) 中，表中“1”代表“是”，“0”表示“否”。



(a) 检验次数期望与实际次品率的关系一　　**(b) 检验次数期望与实际次品率的关系二**

图 8　检验次数期望与实际次品率的关系

九、 模型的分析与改进

9.1 模型一的分析和改进

当使用标称值 10% 近似实际次品率时，序贯检验对减少实际检测次数的效果有限。但当设置实际次品率为 11.5% 时，求解出实际检测次数的期望约为 589，这说明当实际次品率超过厂商宣称的标称值时，该方法可以帮助更快的停止检验，并拒绝接受零件。而当设置实际次品率为 0.085% 时，期望仍约等于 363。

对于情况一通过 python 绘制检测次数期望关于次品率的变化曲线，如图 (8a) 所示。结果说明，但当实际次品率在标称值上下进行浮动时，逐步检测则可以更加明显地减少实际检测次数。相较于次品率高于标称值的情况，在次品率低于标称值的情况下，模型并不能很好的提前停止检验接受该零件，这是停止条件 2 过于严苛导致的。可以设计更加宽泛的检验条件，如连续几个检验批次的次品率均低于标称值，则停止检测并接收零件。

如果连续 8 个检验批次的样品率均低于标称值的 90%，就判定该批样本合格。对于上述问题，可以得到新的曲线图 (8b)。此时可以发现，当次品率低于标称值时，模型可以提前结束检测。

在实际的生产过程中，可以结合具体情况，对停止条件进一步进行优化设置，以达到最优结果。

9.2 模型二误差分析与改进

在问题二中，因存在 $\mathcal{L} = f(\mathcal{L})$ 的无限递归问题，我们将等号右侧括号内的 $\mathcal{L}$ 以最简单的情况进行简化，即令 $\mathcal{L} = b_1 + b_2 + b_3$。为了方便描述，我们将其记为 $\mathcal{L} = f(\mathcal{L}_1)$,$\mathcal{L}_1 = b_1 + b_2 + b_3$, 即第一层迭代。进一步考虑第二层迭代，令 $\mathcal{L}_1 = f(\mathcal{L}_2)$,$\mathcal{L}_2 = b_1 + b_3 + b_3$, 此时的损失函数 $\mathcal{L} = f\left(f(\mathcal{L}_2)\right)$。对该损失函数进行搜索优化，得到的最优

决策如表 (3) 所示。

表 3　问题二决策表–高精度

情况	检测零件 1	检测零件 2	检测成品	拆解成品	成本 (元)
情况 1	0	0	0	1	33.8555
情况 2	1	0	0	1	38.5632
情况 3	0	0	1	1	35.6019
情况 4	1	0	1	1	37.5456
情况 5	0	0	1	1	35.9389
情况 6	1	0	0	0	34.1562

在更精确的计算下，是否检测和拆解成品决策情况并没有发生变化。在问题的更精确求解过程中，是否检验 1 的求解结果有所变化。此时，模型倾向于不检测单价较贵的零件 2，而且支持拆解成品。由于零件 2 单价高且检测费用相对较低，此现象似乎不符合直观的认知。对此，我们修改零件成本并进行了一系列的测试。结果显示：当零件单价小于 14 元时，程序倾向于检测零件，当零件单价大于 16 元时，程序倾向于不检测零件而拆解成品。

经过分析，我们认为该现象可能与公式 (10) 有关。由于存在无限迭代的过程，当零件的单价比较高时，会出现检测零件的期望过高的情况，于是模型会选择相信较低的次品率而不检测零件。如果成品是次品，对成品进行拆解则可以减少额外购买零件的成本。

9.3 模型三误差分析与改进

模型三存在与模型二一样的迭代精度问题，我们在这里也采取二层迭代的形式进行重新规划，得到的最优决策如表 (4) 所示，其中“1”表示“是”，“0”表示“否”，此时的成品制造期望成本为 120.716。对比表 (2) 和表 (2) 可以发现，当精度提高之后，最优决策并没有发生变化，说明模型在较低精度的迭代下已经可以找到最优决策。

表 4　问题三决策表–高精度

决策变量	是/否	决策变量	是/否
是否检测零件 1	1	是否检测半成品 1	0
是否检测零件 2	1	是否检测半成品 2	0
是否检测零件 3	1	是否检测半成品 3	0
是否检测零件 4	1	是否检测成品	1
是否检测零件 5	1	是否拆解半成品 1	0
是否检测零件 6	1	是否拆解半成品 2	0
是否检测零件 7	1	是否拆解半成品 3	0
是否检测零件 8	1	是否拆解成品	1

十、 模型的评价

10.1 模型的优点

- **优点 1：**模型结合了固定抽样和逐步抽样，在确定最大样本容量后进一步尝试减少检测次数。引入动态决策在实际生产过程中可以有效的减少检验次数，从而降低检验成本。
- **优点 2：**模型充分考虑了由于次品率而存在的循环制作过程，在每步决策中都考虑了其它步骤对其造成的影响。
- **优点 3：**模型在搜索空间内全面进行求解，可以准确的找到全局最优解。

10.2 模型的缺点

- **缺点 1:** 模型将问题一建模成一个无限样本的抽样问题，并用正态分布进行近似。在实际的生产过程中，样本数量可能并不能被视作无限。此时需要根据二项分布（或其他可能存在于实际过程中的分布）针对实际的样本量进行计算求解。
- **缺点 2：**由于模型充分考虑了各种情况，导致最终目标函数的解析式过于复杂，且目标函数存在于自身解析式中，不利于后续的化简。
- **缺点 3:** 由于模型目标函数以递归形式存在，以一定方法对其进行化简后求解可能导致求解精度有所下降。

参考文献

[1] 房祥忠. 点估计和区间估计的几种方法[J]. 中国统计, 2023(11):32-34.

附录 A　补充表格

表 5　问题四部分决策表

零件 1 采样次品率	零件 2 采样次品率	成品采样次品率	零件 1 检测	零件 2 检测	成品检测	成品拆解
0.1082	0.1157	0.0972	1	0	0	1
0.0802	0.0983	0.0980	0	0	0	1
0.0832	0.0897	0.1111	0	0	0	1
0.0716	0.0919	0.1261	0	0	0	1
0.0866	0.0968	0.1033	0	0	0	1
0.1092	0.0918	0.0795	0	0	0	1
0.1044	0.0846	0.0807	0	0	0	1
0.1106	0.1020	0.1159	1	0	0	1
0.0944	0.1258	0.0880	0	0	0	1
0.0844	0.0981	0.0734	0	0	0	1
0.0940	0.1046	0.0950	0	0	0	1
0.1190	0.0928	0.1248	1	0	0	1
0.0921	0.0762	0.0908	0	0	0	1
0.0744	0.0762	0.1103	0	0	0	1
0.0773	0.1152	0.1241	0	0	0	1
0.0921	0.0830	0.0731	0	0	0	1
0.1147	0.1026	0.0996	1	0	0	1
0.0973	0.0797	0.1114	0	0	0	1
0.1038	0.0813	0.1117	0	0	0	1
0.1151	0.0901	0.1282	1	0	0	1
0.1169	0.1080	0.0750	1	0	0	1
0.0970	0.1156	0.0778	0	0	0	1
0.1270	0.0959	0.0842	1	0	0	1
0.1192	0.1082	0.1093	1	0	0	1
0.0889	0.1216	0.1054	0	0	0	1
0.1054	0.1051	0.0969	0	0	0	1
0.0892	0.0935	0.1185	0	0	0	1
0.1101	0.1242	0.1252	1	0	0	1
		最终结果	0	0	0	1

附录 B　文件列表

文件名	功能描述
question1.py	问题一代码
question2.nb	问题二公式计算代码
question2_L.m	问题二公式设置代码
question2_OR.m	问题二搜索优化算法代码
question2_GA.m	问题二智能优化算法代码
question2_Plot1.m	问题二折线图代码
question2_Plot2.m	问题二柱状图代码
question3.nb	问题三代码
question4_U.m	问题四均匀采样重做问题二代码
question4_N.m	问题四高斯采样重做问题二代码
question4_U.nb	问题四均匀采样重做问题三代码
question4_N.nb	问题四高斯采样重做问题三代码
*.xlsx	代码运行需要的表格文件
支撑材料说明.txt	说明支撑材料文件夹中的各个文件功能

附录 C　关键代码

question1.py

```python
import numpy as np
import matplotlib.pyplot as plt

# 设置参数
n = 609  # 总样本数量
c = 73  # 允许的最大次品数量
num_simulations = 10000  # 模拟次数

#定义模拟函数
def simulate_detection(n, p, c):
    samples = np.random.binomial(n=1, p=p, size=n)
    np.random.shuffle(samples)

    total_defects = 0
```

```
    detection_count = 0
    accept_count = 0
    current_batch_size = c

    while detection_count < len(samples):
        # 抽取样本
        batch_samples = samples[detection_count:
   detection_count + current_batch_size]
        defects_in_batch = np.sum(batch_samples)
        total_defects += defects_in_batch
        detection_count += len(batch_samples)
        if (defects_in_batch / len(batch_samples) < 0.09):
            accept_count += 1

#判定是否满足停止条件
        if total_defects >= c:
            return detection_count
        if len(samples) - detection_count < c - total_defects:
            return detection_count
       #模型改进中新增的停止条件
        #if (accept_count == 9):
            #return detection_count

        current_batch_size = c - np.sum(samples[:
   detection_count])
    return detection_count


# 定义p的范围
p_values = np.linspace(0.07, 0.13, 100)
expected_detection_counts = []

# 计算每个p值对应的期望检测次数
for p in p_values:
    results = [simulate_detection(n, p, c) for _ in range(
```

```
    num_simulations)]
    expected_detection_count = np.mean(results)
    expected_detection_counts.append(expected_detection_count)

# 绘制期望检测次数关于p的变化曲线
plt.rcParams['font.family'] = 'Heiti TC'
plt.figure(figsize=(10, 6))
plt.plot(p_values, expected_detection_counts,color='b',marker=
    'o', linestyle='-')
plt.xlabel('次品率 (p)')
plt.ylabel('检测次数期望(E)')
plt.title('检测次数期望关于次品率的变化曲线')
plt.grid(True)
plt.show()
```

question2.nb

```
E1 = D1 (C1 + p1 (b1 + C1)/(1 - p1)) + D2 (C2 + p2 (b2 + C2)
    /(1 - p2));
E1 = D1 (C1 + p1 (b1 + C1)) + D2 (C2 + p2 (b2 + C2));
Pt = D1*D2*p3 + (1 - D1) D2 (1 - (1 - p1) (1 - p3)) + D1 (1 -
    D2) (1 - (1 - p2) (1 - p3)) + (1 - D1) (1 - D2) (1 - (1 - p1
    ) (1 - p2) (1 - p3));
E3 = D4 (C4 + L - b1 - b2) + (1 - D4) L;
E2 = D3 (C3 + Pt*E3) + (1 - D3) Pt*Pc (C5 + E3);
Loss = FullSimplify[E1 + E2 + b1 + b2 + b3];
```

question2_L.m

```
%% 2024B问题2~计算成本
clear;
clc;

% 成本矩阵: 零件1, 零件2, 成品检测, 拆解成本, 调换损失
C = [2,3,3,5,6;
     2,3,3,5,6;
     2,3,3,5,30;
     1,1,2,5,30;
```

```
      8,1,2,5,10;
      2,3,3,40,10];

% 零件1,2，成品的次品率
P = [0.1,0.1,0.1;
      0.2,0.2,0.2;
      0.1,0.1,0.1;
      0.2,0.2,0.2;
      0.1,0.2,0.1;
      0.05,0.05,0.05];

B = [4,18,6]; % 零件1,2 的购买单价，装配成本
D = [1,1,1,1]; % 决策变量
PC = 1; % 客户选择更换概率

% 定义 i 以选择不同的情况（1 到 6）
i = 1;

% 提取当前情况的成本和次品率
C_i = C(i, :); % 第 i 行的成本
P_i = P(i, :); % 第 i 行的次品率

% 公式 L 计算
L = (D(1) * (C_i(1) + B(1) * P_i(1)) / (-1 + P_i(1))) ...
  - (D(2) * (C_i(2) + B(2) * P_i(2)) / (-1 + P_i(2))) ...
  + D(3) * (C_i(3) - ((B(1) + B(2) - C_i(4)) * D(4) - (B(1) +
   B(2) + B(3))) ...
  * ((-1 + D(2)) * P_i(2) * (-1 + P_i(3)) + (-1 + D(1)) * P_i
   (1) * (1 + (-1 + D(2)) * P_i(2)) * (-1 + P_i(3)) + P_i(3)))
   ...
  - (-1 + D(3)) * (C_i(5) - (B(1) + B(2) - C_i(4)) * D(4) + (B
   (1) + B(2) + B(3))) ...
  * ((-1 + D(2)) * P_i(2) * (-1 + P_i(3)) + (-1 + D(1)) * P_i
   (1) * (1 + (-1 + D(2)) * P_i(2)) * (-1 + P_i(3)) + P_i(3)) *
    PC + B(1) + B(2) + B(3);
```

```
39
40 % L = B(1)+B(2)+B(3);
41 % 0-1 规划公式
42 output = - D(1) * (C_i(1) + B(1) * P_i(1)) / (-1 + P_i(1)) ...
43          - D(2) * (C_i(2) + B(2) * P_i(2)) / (-1 + P_i(2)) ...
44          + D(3) * (C_i(3) - ((B(1) + B(2) - C_i(4)) * D(4) - L
   ) * ((-1 + D(2)) * P_i(2) * (-1 + P_i(3)) ...
45          + (-1 + D(1)) * P_i(1) * (1 + (-1 + D(2)) * P_i(2)) *
    (-1 + P_i(3)) + P_i(3))) ...
46          - (-1 + D(3)) * (C_i(5) - (B(1) + B(2) - C_i(4)) * D
   (4) + L) * ((-1 + D(2)) * P_i(2) * (-1 + P_i(3)) + (-1 + D
   (1)) * P_i(1) * (1 + (-1 + D(2)) * P_i(2)) * (-1 + P_i(3)) +
    P_i(3)) * PC;
47
48 % 显示输出结果
49 disp(['Output (情况 ', num2str(i), '):']);
50 disp(output);
```

question2_OR.m

```
1  %% 2024B问题2-搜索
2  clear;
3  clc;
4
5  % 成本矩阵: 零件1, 零件2, 成品检测, 拆解成本, 调换损失
6  C = [2, 3, 3, 5, 6;
7       2, 3, 3, 5, 6;
8       2, 3, 3, 5, 30;
9       1, 1, 2, 5, 30;
10      8, 1, 2, 5, 10;
11      2, 3, 3, 40, 10];
12
13 % 零件1,2, 成品的次品率
14 P = [0.1, 0.1, 0.1;
15      0.2, 0.2, 0.2;
16      0.1, 0.1, 0.1;
```

```
     0.2, 0.2, 0.2;
     0.1, 0.2, 0.1;
     0.05, 0.05, 0.05];

% 零件1,2的购买单价，装配成本
B = [4, 18, 6];

% 客户选择更换概率
PC = 1;

% 选择不同情况的 i 值 (1到6)
i = 1; % 选择当前情况

% 提取当前情况的成本和次品率
C_i = C(i, :); % 第i行的成本矩阵
P_i = P(i, :); % 第i行的次品率矩阵

% 枚举所有可能的0-1组合（16种组合）
all_combinations = dec2bin(0:15) - '0'; % 生成16种二进制组合（4位）
num_combinations = size(all_combinations, 1);

% 初始化最优解
min_cost = Inf; % 初始化为无穷大
best_D = [];
cost_m = zeros(1,16);

% 遍历所有组合，计算每个组合的成本
for idx = 1:num_combinations
    D = all_combinations(idx, :); % 当前的0-1组合

    % 计算L值
    L = B(1)+B(2)+B(3)+(D(1) * (C_i(1) + B(1) * P_i(1)) / (-1 + P_i(1))) ...
      - (D(2) * (C_i(2) + B(2) * P_i(2)) / (-1 + P_i(2))) ...
```

```
50        + D(3) * (C_i(3) - ((B(1) + B(2) - C_i(4)) * D(4) - (B
    (1) + B(2) + B(3))) ...
51        * ((-1 + D(2)) * P_i(2) * (-1 + P_i(3)) + (-1 + D(1)) *
    P_i(1) * (1 + (-1 + D(2)) * P_i(2)) * (-1 + P_i(3)) + P_i(3)
    )) ...
52        - (-1 + D(3)) * (C_i(5) - (B(1) + B(2) - C_i(4)) * D(4)
    + (B(1) + B(2) + B(3))) ...
53        * ((-1 + D(2)) * P_i(2) * (-1 + P_i(3)) + (-1 + D(1)) *
    P_i(1) * (1 + (-1 + D(2)) * P_i(2)) * (-1 + P_i(3)) + P_i(3)
    ) * PC;
54     % 计算 output 成本
55     cost = - D(1) * (C_i(1) + B(1) * P_i(1)) / (-1 + P_i(1))
    ...
56             - D(2) * (C_i(2) + B(2) * P_i(2)) / (-1 + P_i(2))
    ...
57             + D(3) * (C_i(3) - ((B(1) + B(2) - C_i(4)) * D(4) -
     L) * ((-1 + D(2)) * P_i(2) * (-1 + P_i(3)) ...
58             + (-1 + D(1)) * P_i(1) * (1 + (-1 + D(2)) * P_i(2))
     * (-1 + P_i(3)) + P_i(3))) ...
59             - (-1 + D(3)) * (C_i(5) - (B(1) + B(2) - C_i(4)) *
    D(4) + L) * ((-1 + D(2)) * P_i(2) * (-1 + P_i(3)) + (-1 + D
    (1)) * P_i(1) * (1 + (-1 + D(2)) * P_i(2)) * (-1 + P_i(3)) +
     P_i(3)) * PC;
60     cost_m(idx) = cost;
61     % 如果当前成本比最优成本小，更新最优解
62     if cost < min_cost
63         min_cost = cost;
64         best_D = D; % 保存当前最优的决策变量组合
65     end
66 end
67
68 % 显示最优决策变量和最小成本
69 disp(['最优决策变量 D (情况 ', num2str(i), '):']);
70 disp(best_D);
71 disp(['最小总成本: ', num2str(min_cost)]);
```

question3.nb

```
L = L1 + L2 + L3 + E1 + E2 + e4;
L1 = FullSimplify[b1 + b2 + b3 + e1 + p1 (C1 + b1)/(1 - p1) +
   p2 (C2 + b2)/(1 - p2) +p3 (C3 + b3)/(1 - p3)];
L2 = L1;
L3 = FullSimplify[ b7 + b8 + e3 + p7 (C7 + b7)/(1 - p7) + p8 (
   C8 + b8)/(1 - p8)];
E2 = 2*DA (CA + P1 (CA + Da (d1 + L1 - b1 - b2 - b3) + (1 - Da
   ) L1)/(1 - P1)) + DC (CC + P3 (CC + Dc (d3 + L3 - b7 - b8) +
    (1 - Dc) L3)/(1 - P3));
LL = L1 + L2 + L3 + e4;
E1 = DD (CD + P4 (Dd (LL - L1 - L2 - L3 + d4) + (1 - Dd) LL )
   ) + (1 - DD) P4*Pc (C9 + Dd (LL - L1 - L2 - L3 + d4) + (1 -
   Dd) LL);
P1 = FullSimplify[(1 - D1) (1 - D2) (1 - D3) (1 - (1 - p1) (1
   - p2) (1 - p3) (1 - PA)) + (1 - D1) (1 - D2) D3 (1 - (1 - p1
   ) (1 - p2) (1 - PA)) + (1 - D1) D2 (1 - D3) (1 - (1 - p1) (1
    - p3) (1 - PA)) + (1 - D1) D2*D3 (1 - (1 - p1) (1 - PA)) +
   D1 (1 - D2) (1 - D3) (1 - (1 - p2) (1 - p3) (1 - PA)) + D1
   (1 - D2) D3 (1 - (1 - p2) (1 - PA)) + D1*D2 (1 - D3) (1 - (1
    - p3) (1 - PA)) + D1*D2*D3*PA];
P2 = P1;
P3 = FullSimplify[D7*D8*PC + D7 (1 - D8) (1 - (1 - p8) (1 - PC
   )) + (1 - D7) D8 (1 - (1 - p7) (1 - PC)) + (1 - D7) (1 - D8)
    (1 - (1 - p7) (1 - p8) (1 - PC))];
P4 = (1 - DA) (1 - DB) (1 - DC) (1 - (1 - P1) (1 - P2) (1 - P3
   ) (1 - PD)) + (1 - DA) (1 -  DB) DC (1 - (1 - P1) (1 - P2)
   (1 - PD)) + (1 - DA) DB (1 - DC) (1 - (1 - P1) (1 - P3) (1 -
    PD)) + (1 - DA) DB*DC (1 - (1 - P1) (1 - PD)) + DA (1 - DB)
    (1 - DC) (1 - (1 - P2) (1 - P3) (1 - PD)) +  DA (1 - DB) DC
    (1 - (1 - P2) (1 - PD)) + DA*DB (1 - DC) (1 - (1 - P3) (1 -
    PD)) + DA*DB*DC*PD;
b1 = 2; b2 = 8; b3 = 12; b4 = 2; b5 = 8; b6 = 12; b7 = 8; b8 =
    12; e1 = 8; e2 = 8; e3 = 8; e4 = 8; C1 = 1; C2 = 1; C3 = 2;
    C4 = 1; C5 = 1; C6 = 2; C7 = 1; C8 = 2; CA = CB = CC = 4;
```

```
   CD = 6; C9 = 40; d1 = d2 = d3 = 6; d4 = 10; p1 = p2 = p3 =
   p4 = p5 = p6 = p7 = p8 = PA = PB = PC = PD = 0.1; Pc = 1; DB
    = DA;
(*定义决策变量的穷举集合，决策变量为 D1,D2,D3,D7,D8,DA,DC,DD,
   Da,Dc,Dd*)
decisions = Tuples[{0, 1}, 11];  (*11 个决策变量*)
(*计算每种组合下的 Loss 值*)
lossValues =
  Table[{D1, D2, D3, D7, D8, DA, DC, DD, Da, Dc, Dd} =
   decisions[[i]];
   (*计算对应组合下的 Loss 值*)Loss, {i, Length[decisions]}];
(*找到最小的 Loss 值*)
minLoss = Min[lossValues];
(*找到对应的最优决策组合*)
optimalDecision = decisions[[First@Position[lossValues,
   minLoss]]];
(*输出结果*)
{minLoss, optimalDecision}
{116.524, {{1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1}}}
```

question4_G.m

```
%2024B问题4求解第二问~高斯分布
clear;
clc;

% 成本矩阵：零件1，零件2，成品检测，拆解成本，调换损失
C = [2, 3, 3, 5, 6;
     2, 3, 3, 5, 6;
     2, 3, 3, 5, 30;
     1, 1, 2, 5, 30;
     8, 1, 2, 5, 10;
     2, 3, 3, 40, 10];

% 零件1,2，成品的次品率（6行3列矩阵）
P = [0.1, 0.1, 0.1;
```

```
    0.2, 0.2, 0.2;
    0.1, 0.1, 0.1;
    0.2, 0.2, 0.2;
    0.1, 0.2, 0.1;
    0.05, 0.05, 0.05];

% 零件1,2的购买单价，装配成本
B = [4, 18, 6];

% 客户选择更换概率
PC = 1;

% 情况(1到6)
i = 1;

% 提取当前情况的成本和次品率
C_i = C(i, :); % 当前情况的成本矩阵
P_i = P(i, :); % 当前情况的次品率矩阵

% 设置次品率的标准差为0.0967倍的次品率（针对情况1的0.1）
std_dev = 0.0967 * P_i; % 标准差基于次品率的0.07倍
if(i == 5)
std_dev(2) = 0.0967 * P_i(2) * 2;%情况5零件二单独赋值
end

% 生成符合高斯分布的次品率
num_samples = 100; % 假设生成100个随机次品率样本
P_samples = zeros(num_samples, 3);

%高斯分布Normal(均值，0.07*均值)
for j = 1:3
    P_samples(:, j) = normrnd(P_i(j), std_dev(j), [num_samples
   , 1]);
end

```

```
% 初始化存储每次次品率组合下的最优解
all_best_D = zeros(num_samples, 4);

% 枚举所有可能的0-1组合（16种组合）
all_combinations = dec2bin(0:15) - '0'; % 生成16种二进制组合（4位）
num_combinations = size(all_combinations, 1);

% 遍历每个次品率组合
for p_idx = 1:num_samples
    % 当前的次品率样本
    P_adjusted = P_samples(p_idx, :);

    % 初始化最优解
    min_cost = Inf; % 初始化为无穷大
    best_D = [];

    % 遍历所有组合，计算每个组合的成本
    for idx = 1:num_combinations
        D = all_combinations(idx, :); % 当前的0-1组合

        % 计算上一次LOSS值，迭代值
        L = B(1) + B(2) + B(3) + (D(1) * (C_i(1) + B(1) * P_adjusted(1)) / (-1 + P_adjusted(1))) ...
            - (D(2) * (C_i(2) + B(2) * P_adjusted(2)) / (-1 + P_adjusted(2))) ...
            + D(3) * (C_i(3) - ((B(1) + B(2) - C_i(4)) * D(4) - (B(1) + B(2) + B(3))) ...
            * ((-1 + D(2)) * P_adjusted(2) * (-1 + P_adjusted(3)) + (-1 + D(1)) * P_adjusted(1) * (1 + (-1 + D(2)) * P_adjusted(2)) * (-1 + P_adjusted(3)) + P_adjusted(3))) ...
            - (-1 + D(3)) * (C_i(5) - (B(1) + B(2) - C_i(4)) * D(4) + (B(1) + B(2) + B(3))) ...
            * ((-1 + D(2)) * P_adjusted(2) * (-1 + P_adjusted(3)) + (-1 + D(1)) * P_adjusted(1) * (1 + (-1 + D(2)) *
```

```
    P_adjusted(2)) * (-1 + P_adjusted(3)) + P_adjusted(3)) * PC;
76
77          % 计算成本
78          cost = - D(1) * (C_i(1) + B(1) * P_adjusted(1)) / (-1
    + P_adjusted(1)) ...
79                  - D(2) * (C_i(2) + B(2) * P_adjusted(2)) / (-1
    + P_adjusted(2)) ...
80                  + D(3) * (C_i(3) - ((B(1) + B(2) - C_i(4)) * D
    (4) - L) * ((-1 + D(2)) * P_adjusted(2) * (-1 + P_adjusted
    (3)) ...
81                  + (-1 + D(1)) * P_adjusted(1) * (1 + (-1 + D(2)
    ) * P_adjusted(2)) * (-1 + P_adjusted(3)) + P_adjusted(3)))
    ...
82                  - (-1 + D(3)) * (C_i(5) - (B(1) + B(2) - C_i(4)
    ) * D(4) + L) * ((-1 + D(2)) * P_adjusted(2) * (-1 +
    P_adjusted(3)) + (-1 + D(1)) * P_adjusted(1) * (1 + (-1 + D
    (2)) * P_adjusted(2)) * (-1 + P_adjusted(3)) + P_adjusted(3)
    ) * PC;
83
84          % 如果当前成本比最优成本小，更新最优解
85          if cost < min_cost
86              min_cost = cost;
87              best_D = D; % 保存当前最优的决策变量组合
88          end
89      end
90
91      % 存储该次品率组合下的最优决策
92      all_best_D(p_idx, :) = best_D;
93  end
94
95  % 计算每个决策变量的众数
96  final_decision = mode(all_best_D);
97
98  % 当前情况的最终决策变量
99  disp(['最终决策变量（情况 ', num2str(i), '):']);
```

```
disp(final_decision);

% 每个次品率组合下的最优解
disp(['每个次品率组合下的最优解:']);
disp(all_best_D);
```

question4_U.nb

```
(*定义初始次品概率*)
initialP = {p1, p2, p3, p4, p5, p6, p7, p8, PA, PB, PC, PD} =
   {0.1,
    0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1};
(*决策变量的穷举集合*)
decisions = Tuples[{0, 1}, 11];(*11个决策变量*)
(*均匀采样0.8到1.2倍的次品率区间*)
uniformSamples =
 Table[RandomReal[{0.8*p, 1.2*p}], {p, initialP}, {10}];
allBestDecisions = {};(*存储每次采样的最优决策组合*)

(*遍历每组采样次品率并计算最优决策*)For[sampleIdx = 1,
   sampleIdx <= 10,
 sampleIdx++,(*更新采样次品率*){p1, p2, p3, p4, p5, p6, p7, p8
   , PA, PB, PC,
   PD} = uniformSamples[[All, sampleIdx]];
 (*计算每种组合下的 Loss 值*)
 lossValues =
  Table[{D1, D2, D3, D7, D8, DA, DC, DD, Da, Dc, Dd} =
   decisions[[i]];
   Loss, {i, Length[decisions]}];
 (*找到最小的 Loss 值*)minLoss = Min[lossValues];
 (*找到最优决策组合并解除嵌套*)
 optimalDecision =
  decisions[[First@Position[lossValues, minLoss]]][[1]];
 (*保存最优决策组合*)AppendTo[allBestDecisions,
   optimalDecision];];

```

```
(*确认 allBestDecisions 的结构是正确的二维数组*)
Print["allBestDecisions 维度: ", Dimensions[allBestDecisions]];
(*计算每个决策变量的众数*)
finalDecision =
  Table[Commonest[allBestDecisions[[All, i]]][[1]], {i, 1, 11}];
(*输出每次采样的最优决策组合*)
Print["每次采样的最优决策组合:"];
Do[Print["第 ", i, " 次采样: ", allBestDecisions[[i]]], {i, 10}];
(*输出最终的众数决策组合*)
Print["最终的决策组合为: ", finalDecision];
```