基于贪心策略和遍历迭代算法的生产决策

摘要

本文主要研究生产过程中的决策问题，通过建立最小合格成品期望成本模型来帮助企业进行决策，以实现企业利润最大化。

针对问题一：使用**假设检验**的方法，由于样本量大且不具有分层性质，抽样方式可以选择随机抽样并利用中心极限定理，将二项分布用**正态分布**进行近似处理，在题目要求的不同的置信水平下通过求解拒绝域、控制样本误差得到样本数量的最小值。

针对问题二：问题二中建立**最小合格成品期望成本模型**，选取合格成品的期望成本作为指标，最小化合格成品期望成本作为目标函数。将生产过程分为零配件决策阶段和成品决策阶段，依次计算合格成品占比与总成本，同时考虑拆解成品情况下后续的成本计量。由于拆解不合格成品会导致生产过程进入无限循环，需要通过设定迭代过程停止条件终止计算。决策组合共有十六种，使用**遍历算法**计算每一种决策的期望成本，追求最小化合格成品期望成本，进而确定最优决策。最后对决策结果进行分析，并分析不同情境下最优决策不同的原因。

针对问题三：问题三中的从零件到半成品的决策过程、从半成品到成品的决策过程，均与问题二基本相同，但是需要根据阶段调整问题二中的模型，从零配件到半成品阶段不需要承担调换损失；从半成品到成品阶段不需要检测半成品是否合格。整个生产决策过程具有明显的贪心特征，因此可以分两个阶段（零件到半成品、半成品到成品），并结合**贪心策略**和问题二的**遍历算法**进行**迭代求解**，最后对决策结果进行分析。

针对问题四：问题四使用问题一的抽样检测方案，通过利用 Btea 分布与二项分布的**先验共轭**性质，进行**贝叶斯推断**得到估计的次品率，利用**蒙特卡洛模拟**得到一千组抽样，估计次品率后，更新问题二和问题三的次品率重新进行问题二和问题三的模型求解，并进行结果分析。

关键词：生产决策　假设检验　最小合格成品期望成本　贪心算法　蒙特卡洛模拟

一、 问题重述

1.1 问题背景

时代变迁，社会性大生产的背景下，专业化分工使得劳动要素被分配到不同的部门，如何让各个劳动要素的产品联通，让各个生产环节的要素流动对企业发展的顺畅与否起到了关键作用。

电子产业尤其如此，电子行业飞速发展和快速迭代的特性决定了这个行业对分工协作的执着。位于质量基础地位的原材料采购尤其重要，严密的零部件采购、精确的零部件质量检测都与产品的质量息息相关。与此同时，协调行业整体，对产业链的每一步的决策优化既可以降低生产成本，还可以优化生产流程、精简售后服务，乃是降本增效、达成事半功倍效果的优秀选择。为了达成这一目的，需要制作一份严谨又明确的决策方案，建立适当的数学模型，来达到生产过程决策的最优。

1.2 需要解决的问题

1. 在制定了抽样检测方案采购一批零部件时：
   (1) 建立抽样数量最少且能够满足置信水平要求的数学模型。
   (2) 基于模型制定怎样的误差标准能够达到效益的最大和成本的最优。
2. 在面对生产的各个环节时：
   (1) 建立四道环节决策规划最优数学模型。
   (2) 基于模型给出具体的决策方案并给出决策指标。
3. 在进行多步骤多变量的决策方案制定时：
   (1) 建立随机工序、随机零部件的通用决策方案模型。
   (2) 基于通用模型实现具体数字下的最优决策方案以及决策指标。
4. 当次品率为通过抽样检测方案得到时：
   (1) 建立模拟次品率下实现生产过程决策方案的数学模型。
   (2) 依据上述模型重新为企业制定新的决策方案并提供决策指标。

二、 问题分析

2.1 问题一的分析

仔细阅读题目之后，发现题目给的条件很少，换而言之，问题一对条件的约束很少，需要我们自己给出一些合理的设定来实现题目的要求。题目要求我们选择抽样检测的方案，抽样检测方案主要有随机抽样、系统抽样和分层抽样，介于题目没有给出限制反而是较为随性的提问，加上零部件具有较好的同一性分层的可能性较低，抽样检测方案可以选择随机抽样，而在随机抽样当中在总体取样的情况中二项分布非常贴切题意，再加

上背景中提到了这种电子产品非常畅销，我们不妨令样本的数量足够大，将符合二项分布的样本所构成的整体可以近似的看成符合正态分布，在标准化正态分布后可以正式进入到题目的解答环节。两题都可以用假设检验中的单边检验作答，利用标准化的正态分布与各自的置信水平做比较，解出对应的样本数量，可以发现样本数量仅和我们所期待的误差值相关，也就是说，只要我们给出一个期待的误差，我们就可以得到一个最小的样本量。

2.2 问题二的分析

零部件一一对应但又数量未定，是否会出现单个零件的缺失从而导致成品的无法组装；步骤三的重复步骤一、二是否与第一次决策固定；成品的拆解是否可以无限迭代，如果不可以那么以什么条件作结比较合理；零部件在接受了一次检验之后被拆解是否需要第二次检验验证自己的完好无损；成品的调换损失包含了替换合格成品的保证，纸面的数字损失背后还有哪些隐形损失。这些问题都会被纳入我们模型的考虑范围之内。

在明确了问题之后，可以发现这是一个多步骤多变量的动态规划问题，单种情况有16 种可能，除此之外还有调换重来的可能，蒙特卡洛和马尔可夫很难反映调换重来的情况。仔细观察后，发现产品的售价一定，那么我们将视角从一一列举可能得到最终最优成本的角度转移到降低平均期望成本，使得产品的期望成本下降，从而达到产品的期望利润最高，由此建立模型并给出决策方案和决策指标。

2.3 问题三的分析

问题三，在和问题二保持同样的前提和隐藏条件的同时，将问题二拓展将解法从具体引导至通用。由于未知数的存在，一一分析数值权衡利弊的想法灰飞烟灭，两层逻辑和从尾拆到头的可能性又将穷举的想法扼杀于摇篮中，但是尽管如此，有问题二的铺垫，问题三实际上就是两层包裹下的问题二，由于各个零部件各个半成品之间并没有直接相关的联系，我们采用贪心算法，在局部最优且互相独立的前提下完全可以实现整体的最优，那么问题三实际上转化成了将零件组成半成品，将半成品组成成品这两个环节，而这两个环节略作处理，例如将半成品看作问题二中无法调换的成品，将成品看作问题二中无法自我检测的顾客，如此处理就可以利用问题二的模型加上细节上的处理进行行之有效的分析。

2.4 问题四的分析

问题四和问题一息息相关，结合问题一的抽样检测方案下的前提下进行对问题二三的次品率的分析模拟才最贴合题意。在问题一中我们虽然用正态分布近似了总体的分布，但是在次品率的模拟上大可不必，为了最大程度地削减误差，我们采用与二项分布

形成共轭先验的贝塔分布来进行模拟，对于不同的先验分布也就是次品率，我们采用不同的 β，在计算模拟的次品率的同时引入对置信区间的计算，更为便捷地为检测与否提供参考借鉴，例如次品率落入较高的置信区间我们就需要采取检测从而降低相对成本的增加，而当次品率落入较低的置信区间内我们可以不检测来降低期望成本。将次品率带入原来问题二和三的数学模型中进行结果分析。

三、 模型假设

为了构建更为精确的数学模型，本文根据实际条件做出以下合理的假设或条件约束：

1. 由于电子产品畅销，假设零配件的数量足够大。
2. 成品市场不会达到饱和状态，不会产生滞销的情况出现。
3. 短期大量生产成品不会影响成品售价，同时各个环节的费用不会收到生产量影响。

四、 符号说明

本文中涉及到的主要变量符号说明：

符号	符号含义及说明
B_i	零配件 i 的购买单价
D_i	零配件 i 的检测成本
q_i	零配件 i 的次品率
p_i	进入装配环节的零配件 i 的次品率
C_i	零配件 i 的期望成本
A	成品的装配成本
D_f	成品的检测成本
H	不合格成品拆解费用
R	不合格成品调换损失
p_{old}	装配出错概率
p_{new}	成品次品率 (包括配件和装配两种原因)
C_{total}	成品总期望成本

注：未申明的变量以其在符号出现处的具体说明为准。

五、 问题一的模型建立与求解

5.1 问题分析

问题一的求解需要建立假设检验，首先需要设定相对应的原假设以及备择假设，通过抽样检测技术构建合理的检验统计量并给定显著性水平 α，根据检验统计量的概率分布，确定拒绝域的临界值。拒绝域是检验统计量可能取值的范围，当检验统计量的观测值落入此范围内时，拒绝原假设。将计算出的检验统计量与临界值进行比较，得到是否拒绝原假设的结果。

由于零部件具有较好分层的可能性较低，故抽样检测方案使用随机抽样是合理的。

5.2 假设检验模型的建立

5.2.1 假设的设定

第一小问

第一小问要在 95% 信度下认定零配件次品率超过标称值 $p_0 = 0.10$，显著性水平 $\alpha = 1 - 95\% = 0.05$，提出原假设 H_0 和备择假设 H_1 如下：

- **原假设** H_0**:** 零配件的次品率 $p \leq p_0$
- **备择假设** H_1**:** 零配件的次品率 $p < p_0$

第二小问

第一小问要在 90% 信度下认定零配件次品率不超过标称值 $p_0 = 0.10$，显著性水平 $\alpha = 1 - 90\% = 0.1$，提出原假设 H_0 和备择假设 H_1 如下：

- **原假设** H_0**:** 零配件的次品率 $p \geq p_0$
- **备择假设** H_1**:** 零配件的次品率 $p > p_0$

5.2.2 二项分布

配件次品率的标称值为 p_0，对样本进行随机抽样，检验结果应该符合二项分布。设 n 为样本数量，其中的次品数为 X，X 是二项分布随机变量，即

$$X \sim \text{Binomial}(n, p)$$

5.2.3 正态分布近似

题目中表示电子产品很畅销，可以理解为配件数量很大，这样可以根据 De Moive-Laplace 中心极限定理 [1]，

$$\lim_{n\to\infty} P\left(\frac{X - np}{\sqrt{np(1-p)}} \leq x\right) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt$$

二项分布可以使用正态分布进行近似处理，即

$$\frac{X-np}{\sqrt{np(1-p)}} \dot{\sim} \text{Normal(0,1)}$$

5.2.4 最小样本数的计算

第一小问

先假定原假设 H_0 成立，在显著性水平为 $\alpha = 0.05$ 情况下，构建检验统计量为

$$U = \frac{k-np_0}{\sqrt{np_0(1-p_0)}}$$

检验的拒绝域为检验统计量大于标准正态分布下信度为 95% 的分位点，

$U_{0.95} = 1.645$，即

$$U = \frac{k-np_0}{\sqrt{np_0(1-p_0)}} \geq U_{0.95}$$

整理得到

$$\frac{k}{n} - p_0 \geq U_{0.95} \cdot \sqrt{\frac{p_0(1-p_0)}{n}} \tag{1}$$

第二小问

构建检验统计量为

$$U = \frac{k-np_0}{\sqrt{np_0(1-p_0)}}$$

假设检验的信度为 90%，显著性水平为 $\alpha = 0.10$，原假设的拒绝域为

$$U = \frac{k-np_0}{\sqrt{np_0(1-p_0)}} \leq U_{0.10}$$

整理得到

$$\frac{k}{n} - p_0 \leq U_{0.10} \cdot \sqrt{\frac{p_0(1-p_0)}{n}} \tag{2}$$

5.3 问题一的求解

根据上述的分析，能够得到样本量最小值为

$$n = \frac{(U_\gamma)^2 \cdot p_0(1-p_0)}{(\frac{k}{n} - p_0)^2} \tag{3}$$

其中，第一小问中 $\gamma = 0.95$，第二小问中 $\gamma = 0.10$，通过设定误差值为 $\left|\frac{k}{n} - p_0\right|$，通常取 0.02，根据计算公式 (1) 和 (2)，分别可以求到第一小问和第二小问的最小样本量。具体结果

当误差值取为 0.02 时，第一小问的最小样本量为 609，第二小问的最小样本量为 370。当误差值取为 0.05 时，第一小问的最小样本量为 98，第二小问的最小样本量为 60。

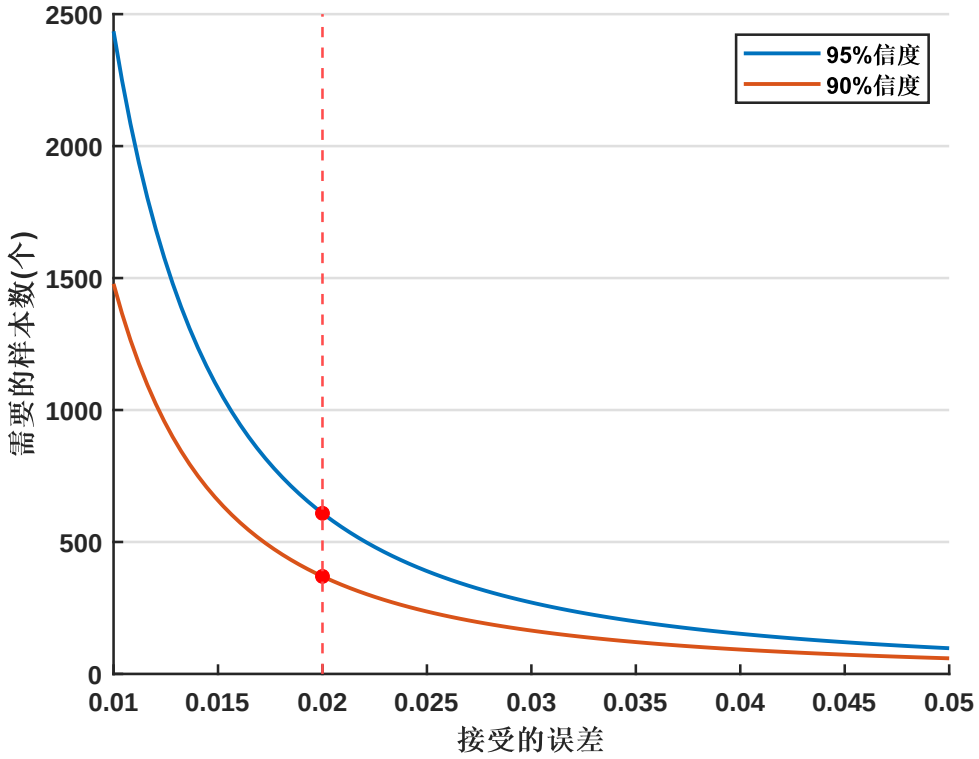

图 **1**　问题一求解结果

六、 问题二的模型建立与求解

6.1 最小化期望成本模型的建立

在问题二中，需要制定最优的决策方案，来实现公司利润的最大化。目标函数为

$$\max \text{利润} = \text{收入} - \text{成本}$$

由于零部件的最初数量不确定并且上述表达式中存在随机变量，故采用计算期望的方法进行目标函数的刻画，即

$$\max \text{合格成品的期望利润} = \text{合格成品的期望收入} - \text{合格成品的期望成本}$$

根据题目信息，市场的售价都是 56 元，是一个定值，故期望收入等于市场售价，这将问题转化成最小化期望成本，即目标函数变为

$$\min \text{合格成品期望成本} \tag{4}$$

决策变量一共有四个，分别是配件 1 是否检测 x_1、配件 2 是否检测 x_2、成品是否检测 x_3 和不合格品是否进行拆解 x_4，以上四个变量均为 $0-1$ 变量，执行时取 1，不执行时取 0。

问题二的解题流程图见图 2。

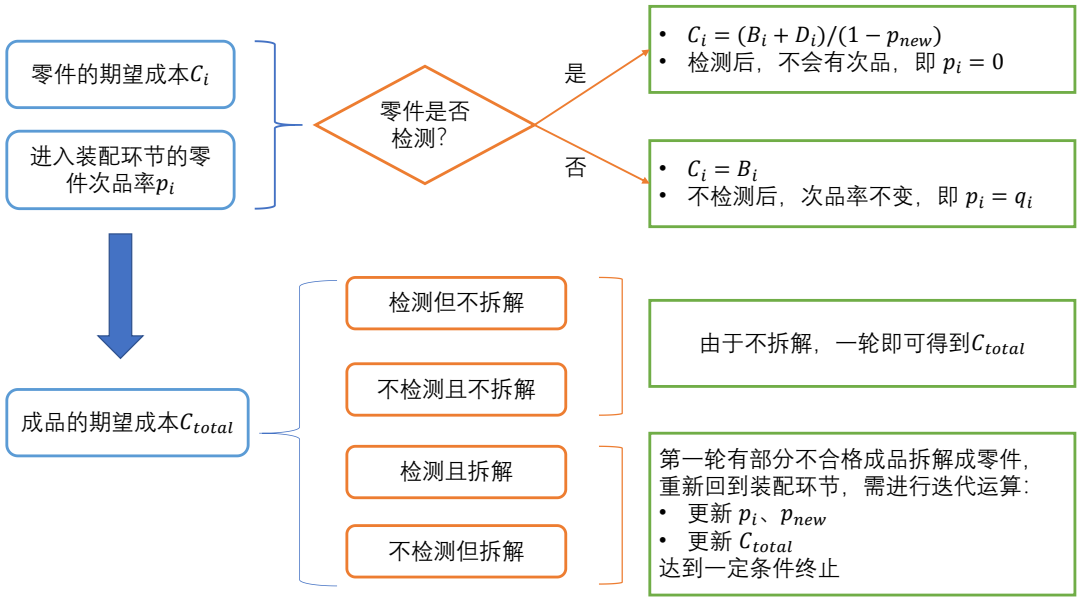

图 **2**　问题二解题流程图

6.2 合格成品期望成本的计算

6.2.1 零配件决策阶段

在问题二中，有 1 和 2 两种零部件，零部件 i 的购买单价为 $B_i(i = 1, 2)$，检测成本为 D_i，在未进行检测前的次品率为 q_i。需要对零部件进行的决策为是否进行检测，检测与否会影响到零部件的期望成本和检测后的次品率，具体情况分为以下两种情况：

1. **若检测零部件：**检测零部件需要检测费用，同时要将不合格零部件都进行丢弃，这时零部件 i 的期望成本为 $C_i = \frac{B_i + D_i}{1 - p_i}$，此时零部件 i 次品率为 $p_i = 0$。
2. **若不检测零部件：**不进行检测不会造成检测费用，并且不需要进行零部件的丢弃，这时零部件的期望成本为 $C_i = B_i$，次品率为 $p_i = q_i$。

6.2.2 成品决策阶段

A. 成品次品率

题目中仅提供成品装配出错概率，由于成品的不合格还有可能是因为配件中存在不合格配件，故现在需要计算成品中次品率 p_{new}。设零配件 i 的次品率为 q_i，成品中的不合格品分为两部分，一部分因为配件不合格造成成品不合格，占比为 $1 - \prod_{i=1}^{2}(1 - p_i)$，另一部分因为装配环节出错，占比为 $p_{old} \times \prod_{i=1}^{2}(1 - p_i)$，故成品次品率为

$$p_{new} = 1 - \prod_{i=1}^{2}(1 - p_i) + p_{old} \times \prod_{i=1}^{2}(1 - p_i) \tag{5}$$

B. 合格成品期望成本的计算

在成品的决策阶段中，成品的期望成本受到零部件成本和成品检测、拆解、次品调换等决策两部分的影响，决策不同导致成品的期望成本也会不同。设成品装配成本为 A，成品检测成本为 D_f，不合格成品拆解费用为 H，不合格成品调换损失为 R，根据决策类型将成品期望分为以下四种情况进行分析：

1. 检测成品并不拆解不合格成品

检测成品，所有成品都会产生零配件成本 $\sum_{i=1}^{2} C_i$，装配成本 A 以及检测成本 D_f。第一轮检测后的不合格成品占比 $= p_{new}$。

不拆解不合格成品，对不合格成品直接丢弃，这样意味着生产过程不会进入第二轮的装配拆解过程，即不会再次进入步骤 (1)、(2)、(3)、(4) 的生产过程，生产过程停止，最终形成的合格成品占比为 $1 - p_{new}$，成本为 $\sum_{i=1}^{2} C_i + A + D_f$。

因此合格成品总期望成本为

$$C_{total} = \frac{\sum_{i=1}^{2} C_i + A + D_f}{1 - p_{new}} \tag{6}$$

2. 不检测成品并不拆解不合格成品

不检测成品，所有成品均会产生零配件成本 $\sum_{i=1}^{2} C_i$ 和装配费用 A。不检测成品无法分辨其中的不合格成品，但是根据成品次品率的计算，能够推断出成品中不合格成品的占比即为 p_{new}，这部分不合格成品会被用户购买从而会产生一定的调换损失即为 $p_{new} \times R$。

不拆解不合格成品，对不合成成品直接丢弃，这样意味着生成过程不会进入第二轮的装配拆解过程，生产过程停止，最终生产的合格成品占比为 $1-p_{new}$，成本为 $\sum_{i=1}^{2} C_i + A + p_{new} \times R$。

因此合格成品总期望成本为

$$C_{total} = \frac{\sum_{i=1}^{2} C_i + A + p_{new} \times R}{1 - p_{new}} \tag{7}$$

3. 检测成品并拆解不合格成品

1) 第一轮装配过程

装配并检测成品，所有成品均会产生零配件成本 $\sum_{i=1}^{2} C_i$，装配成本 A 以及检测成本 D_f，并且能够区分开合格成品和不合格成品。此时第一轮装配过程结束，形成的不合格品比率 $p_{new}^{(1)} = p_{new}$，合格成品为 $1 - p_{new}^{(1)}$，阶段成本为 $\sum_{i=1}^{2} C_i + A + D_f$，因此可以得到第一轮后成品的期望成本

$$C_{total} = \frac{\sum_{i=1}^{2} C_i + A + D_f}{1 - p_{new}^{(1)}}$$

2) 第二轮拆解装配过程

拆解不合格成品，经过上一轮检测得到不合格成品占比 $p_{new}^{(1)}$，所需要的拆解费用为 $p_{new}^{(1)} \times H$，此时形成的配件 1 和配件 2 进行装配，需要更新配件 1 和配件 2 中的配件次品率。根据图 3 示意，成品中不合格成品分为因装配错误形成的不合格成品和因配件不合格形成的不合格成品，占比总和为 $(1-p_1)(1-p_2)p_{old}+1-(1-p_i)(1-p_2)=1+(1-p_1)(1-p_2)(p_{old}-1)$，次品配件占比为 p_i，则配件 i 在第二次进入装配迭代过程时更新的次品率为

$$p_i^{(2)}=\frac{p_i}{1+(1-p_1)(1-p_2)(p_{old}-1)} \tag{8}$$

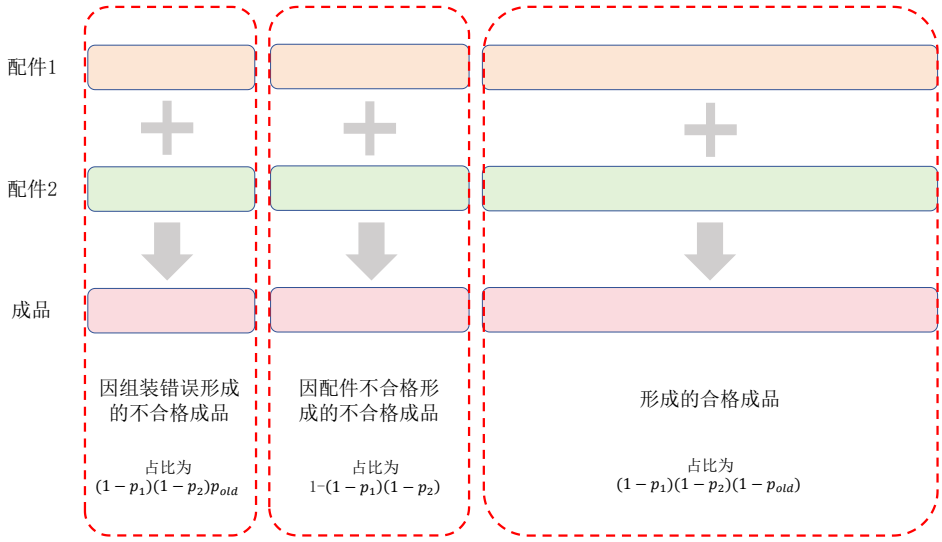

图 **3**　更新配件次品率计算示意图

利用新的配件次品率 $p_i^{(2)}$ 进行公式 (5) 的计算，得到第二次拆解装配过程中的成品次品率为 $p_{new}^{(2)}$。进入第二阶段的成品比率为 $p_{new}^{(1)}$，其中合格品比率为 $p_{new}^{(1)}\cdot(1-p_{new}^{(2)})$，不合格品比率为 $p_{new}^{(1)}\cdot p_{new}^{(2)}$。第二轮中，需要的阶段费用包括：不合格品的拆解费用、装配费用和成品检测费用。故第二轮的阶段成本为 $p_{new}^{(1)}\cdot(H+A+D_f)$。$C_{total}$ 可以更新为：

$$C_{total}=\frac{\sum_{i=1}^{2}C_i+A+D_f+p_{new}^{(1)}\cdot(H+A+D_f)}{1-p_{new}^{(1)}\cdot p_{new}^{(2)}}$$

3) 第三轮拆解装配过程至第 n 个拆解装配过程

第二轮检测出来的不合格成品将进入第三轮拆解装配过程，继续更新 p_i 和 $p_{new}^{(i)}$，得到 $p_i^{(3)}$ 和 $p_{new}^{(3)}$。第三轮拆解装配过程中的合格成品比率为 $p_{new}^{(1)}\cdot p_{new}^{(2)}\cdot(1-p_{new}^{(3)})$，不合格品比率为 $p_{new}^{(1)}\cdot p_{new}^{(2)}\cdot p_{new}^{(3)}$；阶段成本为 $p_{new}^{(1)}\cdot p_{new}^{(2)}\cdot(H+A+D_f)$。$C_{total}$ 可以更新为：

$$C_{total}=\frac{\sum_{i=1}^{2}C_i+A+D_f+(p_{new}^{(1)}+p_{new}^{(1)}\times p_{new}^{(2)})\cdot(H+A+D_f)}{1-p_{new}^{(1)}\cdot p_{new}^{(2)}\cdot p_{new}^{(3)}}$$

重复上述操作，便能够得到各个阶段的成本以及形成合格成品数。

综上所述，合格成品总期望成本为

$$C_{total} = \frac{\text{每一个装配拆解过程的成本之和}}{\text{每一个装配拆解过程形成的合格成品比率之和}}$$

代入表达式，根据规律整理可得

$$C_{total} = \frac{\sum_{i=1}^{2} C_i + A + D_f + (H + A + D_f) \times \sum_{i=1}^{n} \prod_{1}^{i} p_{new}^{(i)}}{1 - \prod_{i=1}^{n} p_{new}^{(n)}} \tag{9}$$

4. 不检测成品并拆解不合格成品

1) 第一轮装配过程

不检测成品，所有成品均会产生零配件成本 $\sum_{i=1}^{2} C_i$ 和装配费用 A，其中不合格成品的占比即为 p_{new}，其造成的调换损失即为 $p_{new}^{(1)} \cdot R$。此时第一轮装配过程结束，形成合格成品比率为 $1 - p_{new}^{(1)}$，阶段成本为 $\sum_{i=1}^{2} C_i + A + p_{new}^{(1)} \cdot R$。

2) 第二轮拆解装配过程拆解上一轮的不合格成品所需要的费用为 $p_{new}^{(1)} \times H$，利用公式 (5) 更新零件次品率 $p_i^{(3)}$，然后利用公式 (5) 进行得到成品次品率 $p_{new}^{(2)}$，第二轮中，形成的合格成品比率为 $p_{new}^{(1)} \cdot (1 - p_{new}^{(2)})$，阶段成本为 $p_{new}^{(1)} \cdot (H + A + R \times p_{new}^{(2)})$

3) 第三轮拆解装配过程至第 n 个拆解装配过程

对第二轮过程的不合格成品进入第三轮拆解装配过程，重复上述操作，便能够得到各个阶段的成本以及形成合格成品比率。

综上所述，合格成品总期望成本为

$$C_{total} = \frac{\sum_{i=1}^{2} C_i + A + R \times p_{new}^{(1)} + \left(H + A + R \times p_{new}^{(n)}\right) \times \sum_{i=1}^{n} \prod_{1}^{i} p_{new}^{(i)}}{1 - \prod_{i=1}^{n} p_{new}^{(n)}} \tag{10}$$

C. 拆解装配迭代过程的终止条件设定

成品决策阶段中，对于第三种决策（检测成品并拆解不合格成品）和第四种决策（不检测成品并拆解不合格成品），拆解不合格品会使得无限进入新的拆解装配过程，现需要设定迭代的终止条件。

根据对上述两种情况循环 100 次的结果分析，我们发现在第三种决策中 $\prod_{i=1}^{n} p_{new}^{(i)}$ 会以非常快的速度接近于 0，在第四种决策中则是 $p_{new}^{(i)}$ 会以非常快的速度接近于 1，对这两种情况的实际分析我们认为上述结果是合理的，决策三在拆解的前提下检测成品可以快速实现合格成品的输出，从而快速缩小可以进行拆解的不合格品，进而降低进入下一轮拆解的可能，决策四不检验成品从而为坏零部件提供生存空间，使得次品在组装中的出现概率大幅度增加，产品的次品率由此得到快速提升。为了保持成本计算的精确性，我们选择了 0.001 作为终止条件。当 $\prod_{i=1}^{n} p_{new}^{(i)} < 0.001$ 或者 $1 - p_{new}^{(i)} < 0.001$ 的情况下终止迭代过程。

6.3 模型求解

6.3.1 结果展示

问题二中，决策变量只有四个，最终决策的种类只有十六种，对于每一种决策都能够通过上述的过程计算出合格成品的期望成本，比较十六种情况中得到最小期望成本的决策即为最优决策组合。

通过**遍历算法**和**迭代算法**，计算得到六种情境下十六种决策方案的期望成本，结果如图 4 所示，具体的数值表格在附录 B 中体现。

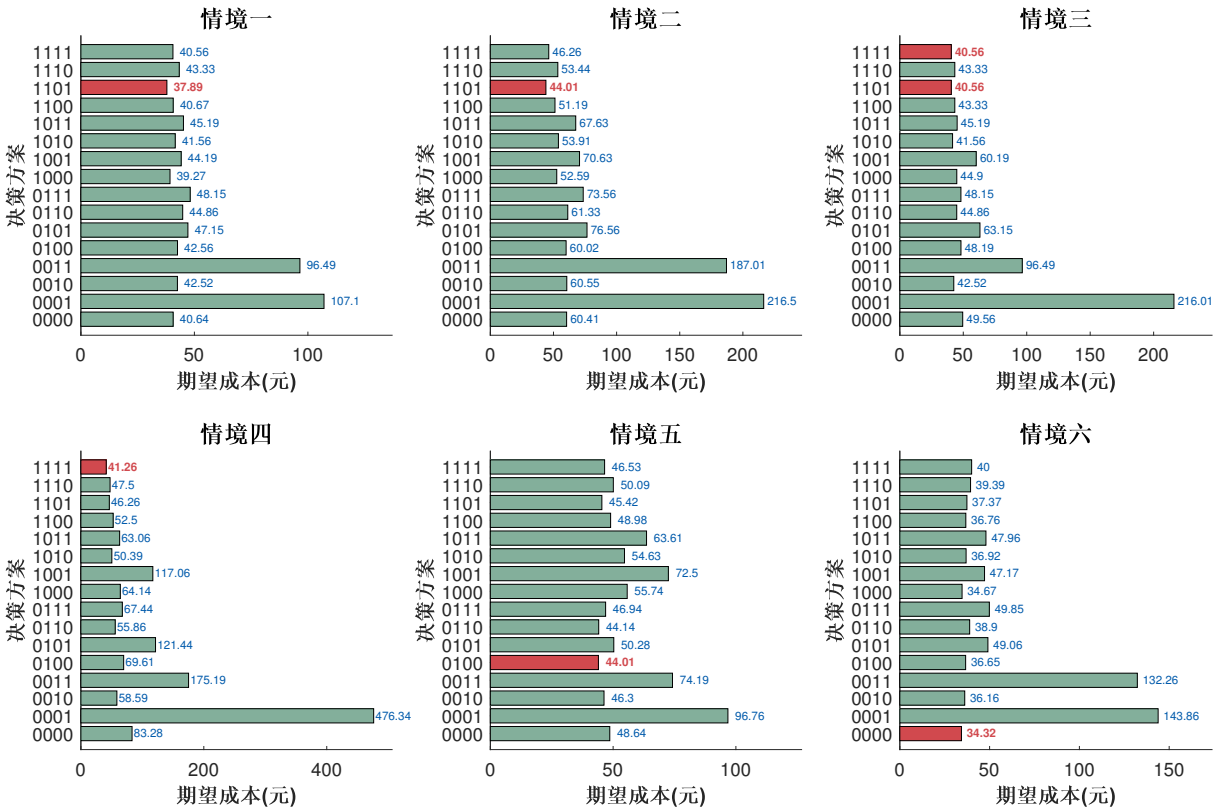

图 **4**　问题二中六种情境下不同决策方案的期望成本

对于每种情境的最优决策以及最优决策下的最小期望成本结果见表 1 所示。

情境	最优决策组合	最小期望成本
情境 1	1101，检测配件 1，检测配件 2，不检测成品，拆解不合格成品	37.89
情境 2	1101，检测配件 1，检测配件 2，不检测成品，拆解不合格成品	44.01
情境 3	1111，检测配件 1，检测配件 2，检测成品，拆解不合格成品 1101，检测配件 1，检测配件 2，不检测成品，拆解不合格成品	40.56
情境 4	1111，检测配件 1，检测配件 2，检测成品，拆解不合格成品	41.26
情境 5	0100，不检测配件 1，检测配件 2，不检测成品，不拆解不合格成品	44.01
情境 6	0000，不检测配件 1，不检测配件 2，不检测成品，不拆解不合格成品	34.32

表 **1**　问题二中六种情境的最优决策和最小期望成本

6.3.2 结果分析

1. 所有情境中，0001 和 0011 这两种决策的效果都很差，也即期望成本都很大，这是因为配件 1 和配件 2 都没有进行检测，但拆解不合格成品，使得不合格品拆解后装配所耗费的成本变得很大造成损失。
2. 情境三中最优有两个决策，即 1111 和 1101，原因是配件 1 和配件 2 都进行检测前提下，装配出错概率与调换费用的乘积刚好等于检测费用，这就使得两种方案的期望成本一样。进行推广，发现配件 1 和配件 2 都进行检测前提下，如果装配出错概率与调换费用的乘积大于检测费用，这时对于 x_3 是否进行成品检测的最优方案为检测成本；如果装配出错概率与调换费用的乘积小于检测费用，这时对于 x_3 是否进行成品检测的最优方案为不检测成本。
3. 所有情境中最优的决策几乎（除了情境 6）都选择了检测配件 2，原因固然有配件 2 的初始成本相对于检测费用高，但是配件 2 作为组成成品的一部分的角度和部件 1 是同等地位，所以检测与否的标准与其所占权重并没有太大的关系，成本的高低才是影响检测与否的重要标准。

七、 问题三的建立与求解

7.1 贪心策略和迭代求解的可行性

首先，问题三的决策过程分为两个阶段，从零配件到半成品的决策过程，以及从半成品到成品的决策过程。两者均与问题二中从零配件到成品的决策过程类似。虽然部分成本的处理上略有差异（例如：从零配件到半成品的决策过程不需要承担调换损失；从半成品到成品的过程不需要检测自身是否合格），但整体上分类讨论的思路与问题二相同。

其次，在整个生产链中，过去阶段的决策状态不会影响当前阶段的决策过程，各个

阶段的决策是相互独立的。又当前阶段产出的半成品的期望成本越低，一定会更有利于减少后续的生产成本，具有明显的贪心特征。

综合上述两点，可以采用贪心算法进行迭代求解，具体步骤分两阶段进行：

- **第一阶段**：计算使得各个半成品的期望成本达到最小的决策方案
- **第二阶段**：利用第一阶段求得的各个半成品的最小期望成本，以同样的方式迭代求解使得最后成品期望最小的决策方案

7.2 两道工序、八个零配件情况

7.2.1 从零配件到半成品的决策过程

A. 配件到半成品应用问题二模型的调整

从配件到半成品，相对于问题二中的模型，区别在于没有用户进行购买，不需要承担调换费用。需要对问题二的模型进行一定的调整（只有原来不检测的情况涉及到调换费用，因此只需调整不检测的情况），具体情况如下：

1) 半成品次品率调整

$$p_{new} = 1 - \prod_{i=1}^{\theta}(1-p_i) + p_{old} \times \prod_{i=1}^{\theta}(1-p_i) \quad (\theta = 2\ or\ 3) \tag{11}$$

2) 检测半成品并不拆解不合格半成品决策，期望成本不需要调整

3) 不检测半成品并不拆解不合格半成品决策，期望成本调整

$$C_{total} = \frac{\sum_{i=1}^{n} C_i + A}{1 - p_{new}} \tag{12}$$

4) 检测半成品并拆解不合格半成品决策，期望成本不需要调整

5) 不检测半成品并拆解不合格半成品决策，期望成本调整

$$C_{total} = \frac{\sum_{i=1}^{2} C_i + A + (H+A) \times \sum_{i=1}^{n} \prod_{1}^{i} p_{new}^{(i)}}{1 - \prod_{i=1}^{n} p_{new}^{(n)}} \tag{13}$$

B. 问题三配件到半成品的决策求解

观察半成品一和半成品二的生产要素可以发现，零配件 1、2、3 和配件 4、5、6 各个指标都相同，同时半成品一和半成品二在各个指标也都相同，所以半成品一和半成品二的决策过程是一样的，最优决策结果也是一样的。

故此时配件到半成品的过程，只需要进行半成品 1 和半成品 3 的最优决策求解，就能够得到此阶段的最优决策。

7.2.2 从半成品到成品的决策过程

A. 半成品到成品应用问题二模型的调整

通过上个阶段配件到半成品的最优策略求解，能够得到半成品 1、2、3 的次品率以及半成本的期望成本。

从半成品到成品，相对于问题二中的模型，区别在于半成品不需要检测自身是否合格，即不需要半成品的检测费用。但是计算成品期望成本的过程与问题二相同，因此不需要调整。

1) 成品次品率调整

$$p_{new} = 1 - \prod_{i=1}^{3}(1 - p_{半i}) + p_{old} \times \prod_{i=1}^{3}(1 - p_{半i}) \tag{14}$$

2) 检测成品并不拆解不合格成品决策，期望成本不需要调整

3) 不检测成品并不拆解不合格成品决策，期望成本不需要调整

4) 检测成品并拆解不合格成品决策，期望成本不需要调整

5) 不检测成品并拆解不合格成品决策，期望成本不需要调整

B. 问题三半成品到成品的决策求解

7.2.3 问题三的结果

通过上述的分析，对每一个阶段进行贪心算法的遍历迭代，能够求解到各个阶段的最优决策，结果见图 5 和图 6。

最终，问题三的最优决策，表 2 所示。

是否检测配件 1	检测	是否检测半成品 1	不检测
是否检测配件 2	检测	是否拆解半成品 1	检测
是否检测配件 3	检测	是否检测半成品 2	不检测
是否检测配件 4	检测	是否拆解半成品 2	检测
是否检测配件 5	检测	是否检测半成品 3	不检测
是否检测配件 6	检测	是否拆解半成品 3	检测
是否检测配件 7	检测	是否检测成品	检测
是否检测配件 8	检测	是否拆解成品	检测

表 **2**　问题三中的最优决策

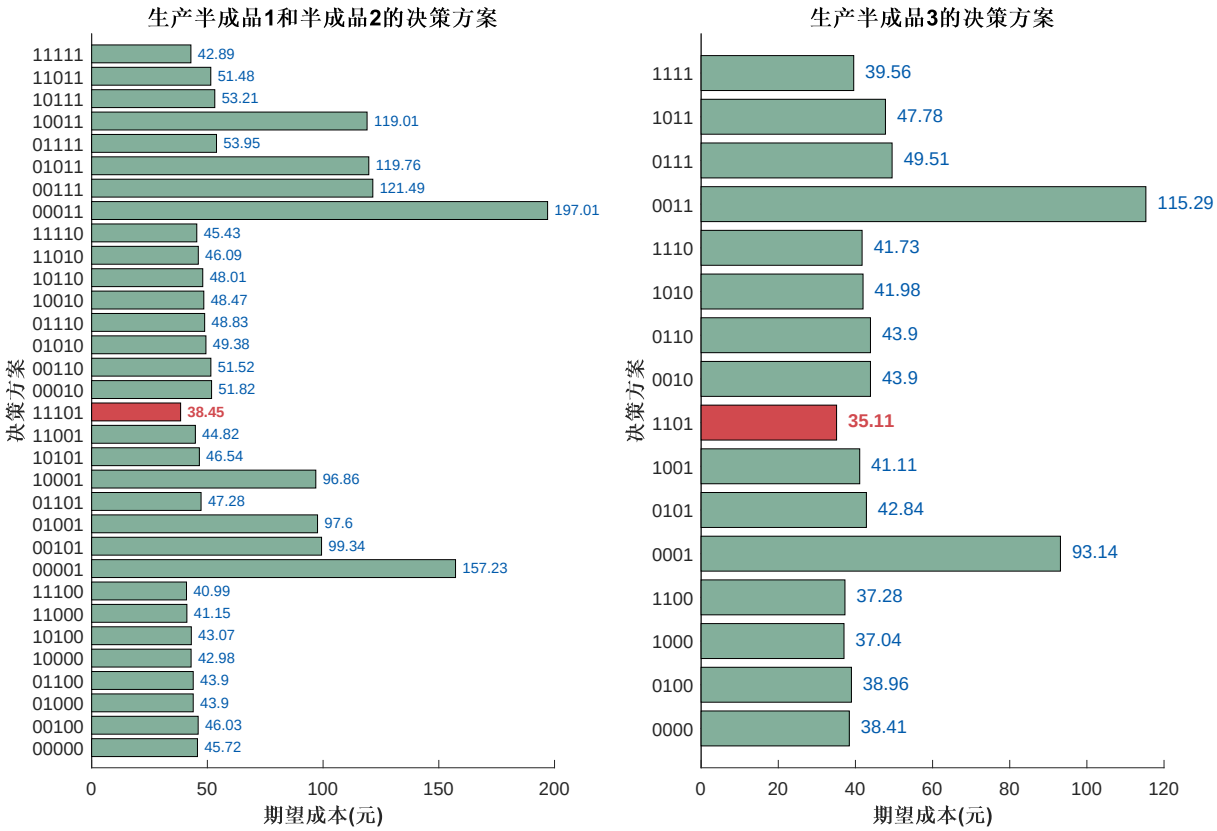

图 **5**　问题三配件到半成品过程的结果

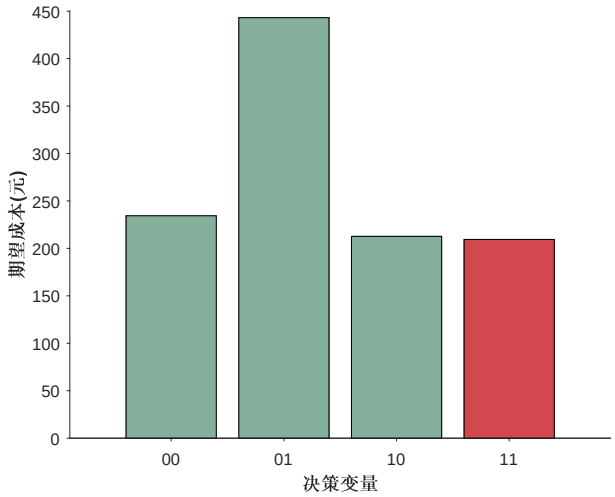

图 **6**　问题三半成品到成品过程的结果

7.3 m 道工序、n 个零配件情况

7.3.1 n 个零配件的处理

n 个零配件的处理，需要根据实际所处位置，进行和两道工序、八个零配件情况相似的模型调整，再进行阶段的最优决策求解。

7.3.2 m 道工序的处理

m 道工序的处理，从最小的零配件开始，进行调整的问题二模型的求解，得到半成品的次品率以及期望成本后，再进行半成品的装配过程，和两道工序、八个零配件情况类似处理即可。

八、 问题四的模型建立与求解

8.1 抽样检测方法的选择

8.1.1 模型的回顾与思考

回顾问题一中我们采用的是随机抽样方法，利用样本分布的二项分布，在样本总体较大基础上对总体采用正态分布近似，但是在问题四中我们需要得到的是自己模拟的次品率，如果引入正态分布、置信区间并以此来计算次品率，那么得到的结果必然是一个区间，这个区间不但会增加我们接下来对问题二三的建模难度同时也会因为问题二三中样本较小的数量导致误差的扩大，考虑到样本数量和误差，我们可以在问题四中直接采用二项分布进行建模。

8.1.2 贝叶斯推断与模型建立

贝叶斯推断核心公式：

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)} \tag{15}$$

其中：

P(A) 表示估计次品率

P(B) 表示抽样次品率

P(A | B) 表示后验概率

P(B | A) 表示似然函数

由于我们对样本采用的是二项分布，而贝塔分布又与二项分布先验共轭，因此我们对二项分布做贝叶斯更新，并不影响更新之后的分布符合情况，而通过更新出来的结果我们就可以得到新的次品率估计。

样本记为 $X\sim Binomial(n,p)$, 假设一次抽样得到了 k 个次品, 由此可以给出似然函数：

$$P(X=k|p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{16}$$

此时如果将 p 看成是一个分布，让 p 服从贝塔分布，即 p~$Beta(\alpha,\beta)$ 那么因为二项分布和贝塔分布先验共轭，进行更新之后的后验分布仍然服从贝塔分布：

$$p|B\sim Beta(\alpha+k,\beta+nk) \tag{17}$$

到此贝叶斯推断的所有工作结束，通过给定一个初始的 α 和 β 我们就可以实现对问题二三中每一种情况的次品率的模拟。

8.2 次品率的优化模拟

通过贝叶斯推断将二项分布和贝塔分布联系到一起得到的次品率是一个不固定的数值，每一次模拟的次品率并不相同，那么如何挑选出最为合适的次品率组呢？

我们这里采用的是置信水平判断法，在计算每一个次品率的同时检测该次品率的所处于的置信水平的区间范围，当所处的置信水平低于某个数值（这里采取的是 0.8）时就放弃这一轮的次品率，从而达到稳定高效的次品率输出组。

在保证了相对的质量之后，我们还应当对数量做出要求，利用蒙特卡洛模拟方法，随机出 1000 组次品率进入问题二三的模型中进行分析，以确保极端情况的出现概率最低，提高决策的正确性。

8.3 模型求解

8.3.1 问题二模型求解

分别将 1000 组次品率导入问题二原本的模型中，输出最优决策方案衣机作为决策依据的最低期望成本。

结果如图 7 所示。

情况 1、2、3、4 非常稳定，其中情况 3 虽然有两种不同的最优决策方案但实际上这两种最优决策方案在原本固定次品率的问题二中情况 3 本身就是有两种最优的决策方案，详情不再赘述。

重点来看出现了变化的情况 5 和情况 6。

情况 5 的两种决策区别在于成品是否检测，观察表格数据，排除变量之后我们可以将问题转化为：成品检验所产生的检测费用和成品不检验所带来的在顾客端产生的调换费用在什么情况下会接近。再将情况 4、5 相比较，我们发现能够与之相关的最大的变动在于零部件的次品率上升，零部件次品率上升带来的是不合格产品的比例上升，当不合格产品比例与调换费用相乘跟检测成品所带来的检测费用相近就会出现这种情况下的决策最优。

情况 6 也是类似的道理，由于相比于单价零部件 1 的检测费用偏高且与情况 3 相比，零部件中次品率较低，当不检测零部件 1 所产生的相对丢弃的成品价值和检测零部

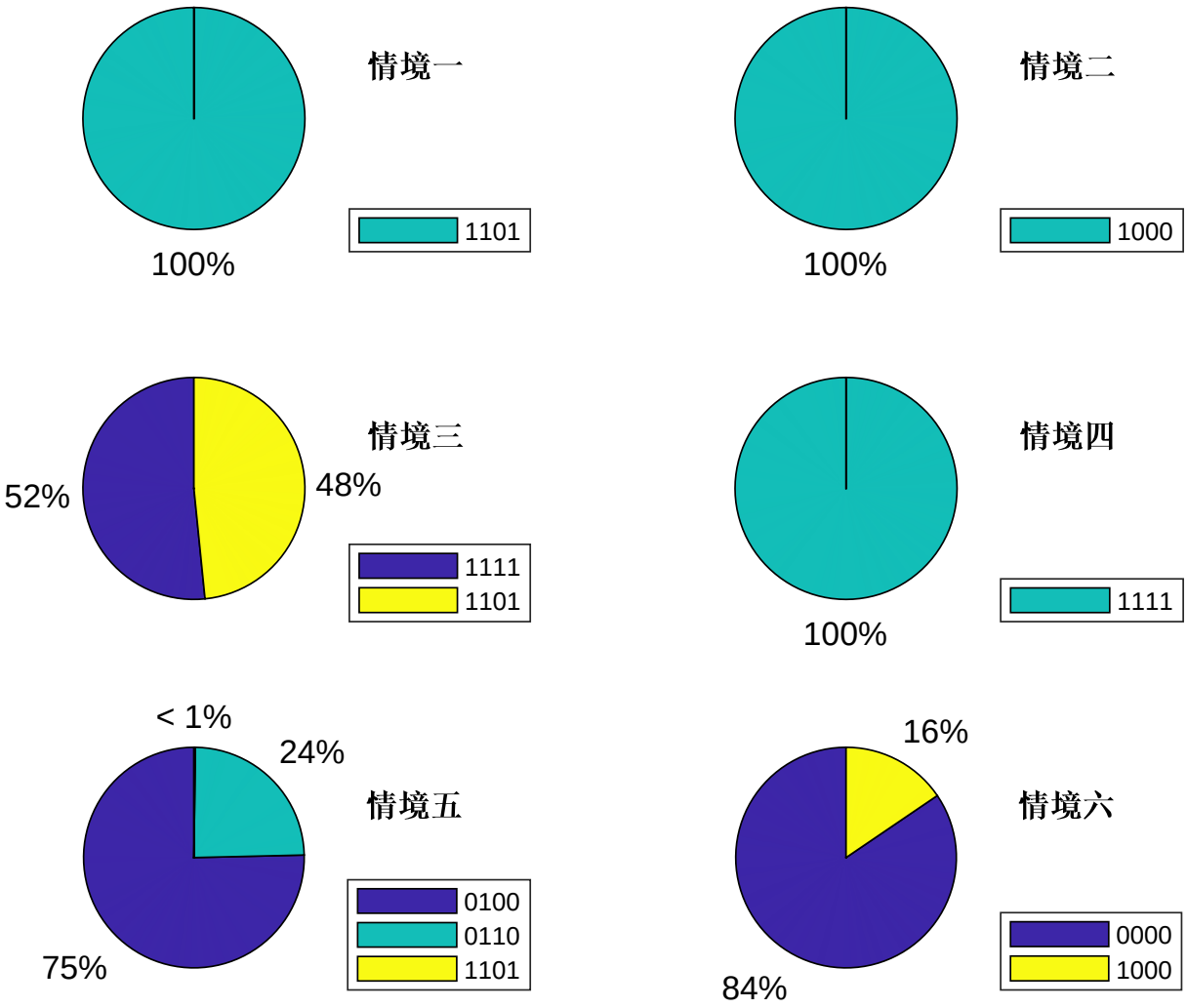

图 7　模拟计算问题二的结果

件 1 所产生的检测费用相近时就会出现两种不同的最优决策方案。

8.3.2 问题三模型求解

分别将 1000 组次品率导入问题三原本的模型中，输出最优决策方案衣机作为决策依据的最低期望成本。

结果如图 8 所示：

与问题二不同的是，问题三中次品率发生改变并没有影响或者说至少绝大部份情况是没有影响的，1000 次的样本数据模拟所导出的结果都是统一的。通过观测问题二中发生变化的情况 5 和情况 6 与问题三中的情况，我们发现，情况 5 和情况 6 在同一条件下都有与自己保持 0.1 级的差距的存在，而问题三中最接近的也有将近 1 级的差距，由此可见在随机事件中有一些差距是很难通过概率的方式予以弥补。

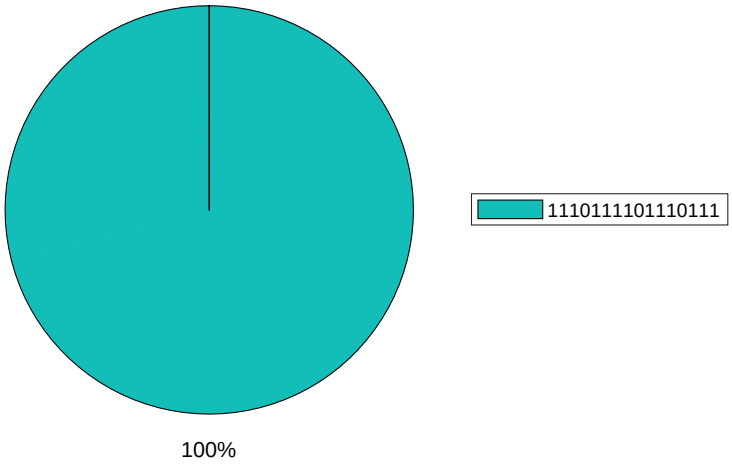

图 **8**　模拟计算问题三的结果

8.4 结果分析

第四问改变二三中的次品率并且采取随机抽样的方式，可以看作是一种以数据来模拟现实的情况。在问题二中各个不同的情况在问题四中就不再是一个个单独的个体，这些情况都可以看作是彼此的对照组，而我们所建立的模型便是在为这些组进行对照实验，结果虽然无法详细的反映过程，但是从结果去推断过程中发生的变化，根据这些变化来思考初始数据的效用。哪怕只是 10% 到 5% 的细微变化，对于决策对象的整体来说产生的变化将是巨大的，甚至有些模拟出来的次品率相较于原本只发生了不到 1% 的变化，这些细小的变化在日常生活中会为我们提供另一种截然不同的可能。看似只是改变了选取次品率的方式，实际上整个模型都因为这个的改变而变得环环相扣，并且创造了新的决策方案，这些方案是无法在一个完全给定的环境要素中产生的。就像问题二中哪个 0.002 的 $1-1-0-1$，哪怕他可能只是一种较为极端的方式产生的误差，但是这个结果同样在那种情景下成为了最优的决策方案。也是这种极端的产生让模型具有更多的可能。

九、 模型的评价与推广

9.1 模型的评价

9.1.1 优点

1. 模型将所用的每一个变量所引用的每一个函数都一一作了详细的解释，使得模型上下贯通便于理解。
2. 将调换产生的循环所带来的每一次决策步骤合理规划，使得受检测的零部件不用再检测，从而降低了循环的成本。
3. 模型没有针对某种特别特殊的情况作出特定情况下的特定解答而是提供了通用的解

决方案，具有广泛的应用范围。

4. 设定了巧妙的跳出循环的条件，加快了模型运转速度，又不失精确度。

9.1.2 缺点

1. 在问题二中使用遍历迭代算法，如果成品工序进行修改，该模型也要做出相对应的修改，具有一定局限。

9.2 模型的推广

1. 本模型实现了从原材料采购开始到售后调换的全局决策最优，在短期较不复杂的生产决策指导中可以发挥显著作用。问题四的模拟次品率更是直接利用数据的随机性质实现了映射现实中次品率的情况，为生产者提供了切实的管理工具。
2. 决策事件在我们日常生活中较为常见，本模型通过改变相应的参数，步骤或者引入新的变量可以满足日常大多数的决策需求。又由于采用了最小化期望成本的方式，在经济数字化的现在很多产品和服务可以转化为成本作为计量的方式，从而使得本模型在经济活动中具有一定的适用性。
3. 除了提供决策的方案，本模型同时提供方案所需要的最小期望成本，能够在很多临界问题，例如投资组合问题和红线问题方面具有应用的前景。从而实现从决策优化模型向更加广阔的方向发展。

参考文献

[1] 王静海, 范恩贵. 中心极限定理在抽样推断中的应用 [J]. 张家口农专学报,1995,(01):26-30.

[2] 王成, 罗玉波. 二项分布参数区间估计——基于十三种方法的模拟比较 [J]. 统计与管理,2021,36(03):24-31.DOI:10.16722/j.issn.1674-537x.2021.03.005.

[3]《贝塔分布、多项分布与二元正态分布》；Morris H Degroot；李永镇；李邵阳；高专学报；1994-04-10

[4]《二项分布的近似计算与应用举例》；万祥兰；科技视界；2019-08-15

[5]《大型电力系统可靠性评估中的马尔可夫链蒙特卡洛方法》；石文辉；别朝红；王锡凡；中国电机工程学报；2008-02-05

[6]《隐状态个数未知的隐马尔可夫多元正态分布的贝叶斯推断》；刘鹤飞; 王坤; 蒋成飞；统计研究；2017-12-25

附录 A　支撑材料文件列表

文件名	类型	简介
Q1/q1.m	Matlab 代码文件	问题一假设检验源代码
Q2/q2.py	Python 代码文件	问题二最小化期望成本决策代码
Q2/q2.m	Matlab 代码文件	问题二结果可视化代码
Q2/result.xlsx	Excel 文件	问题二的决策结果
Q3/q3_1.xlsx	Python 代码文件	问题三第一阶段的决策过程代码
Q3/q3_2.py	Python 代码文件	问题三第二阶段的决策过程代码
Q3/q3_1.m	Matlab 代码文件	问题三第一阶段结果可视化代码
Q3/q3_2.m	Matlab 代码文件	问题三第二阶段结果可视化代码
Q3/Q3_result_A.xlsx	Excel 文件	问题三第一类半成品的决策结果
Q3/Q3_result_B.xlsx	Excel 文件	问题三第二类半成品的决策结果
Q3/Q3_result_C.xlsx	Excel 文件	问题三第三类半成品的决策结果
Q4/q4_1.py	Python 代码文件	问题四中重做问题一的代码
Q4/q4_2.py	Python 代码文件	问题四中重做问题二的代码
Q4/q4_1.m	Matlab 代码文件	问题四中重做问题一的可视化代码
Q4/q4_2.m	Matlab 代码文件	问题四中重做问题二的可视化代码

附录 B　问题二的数值表格结果展示

决策情况	情境一	情境二	情境三	情境四	情境五	情境六
0000	40.64	60.41	49.56	83.28	48.64	34.32
0001	107.1	216.5	216.01	476.34	96.76	143.86
0010	42.52	60.55	42.52	58.59	46.3	36.16
0011	96.49	187.01	96.49	175.19	74.19	132.26
0100	42.56	60.02	48.19	69.61	44.01	36.65
0101	47.15	76.56	63.15	121.44	50.28	49.06
0110	44.86	61.33	44.86	55.86	44.14	38.9
0111	48.15	73.56	48.15	67.44	46.94	49.85
1000	39.27	52.59	44.9	64.14	55.74	34.67
1001	44.19	70.63	60.19	117.06	72.5	47.17
1010	41.56	53.91	41.56	50.39	54.63	36.92
1011	45.19	67.63	45.19	63.06	63.61	47.96
1100	40.67	51.19	43.33	52.5	48.98	36.76
1101	37.89	44.01	40.56	46.26	45.42	37.37
1110	43.33	53.44	43.33	47.5	50.09	39.39
1111	40.56	46.26	40.56	41.26	46.53	40

附录 C　问题一源代码

3.1　假设检验 (Matlab)

```
E = linspace(0.01, 0.05);
u_95 = norminv(0.95, 0, 1);
u_10 = norminv(0.1, 0, 1);
n_95 = u_95^2 * 0.1 * 0.9 ./ E.^2;
n_10 = u_10^2 * 0.1 * 0.9 ./ E.^2;
plot(E, n_95,'LineWidth',1.5)
```

```
hold on
plot(E, n_10, 'LineWidth',1.5)
xline(0.02, '--r', 'LineWidth', 1); % '--r' 表示红色虚线
plot(0.02, u_95^2 * 0.1 * 0.9 / 0.02^2, 'r.', 'MarkerSize', 18)
plot(0.02, u_10^2 * 0.1 * 0.9 / 0.02^2, 'r.', 'MarkerSize', 18)
xlabel('接受的误差')
ylabel('需要的样本数(个)')
set(gca, 'box', 'off', 'YGrid', 'on', 'linewidth', 1, 'fontweight', 'bold')
legend("95%信度", "90%信度")

exportgraphics(gcf, 'q1.eps', 'ContentType', 'vector')
```

附录 D　问题二源代码

4.1　最小化期望成本模型 (Python)

```
import pandas as pd
import matplotlib.pyplot as plt



def get_mean_cost(x1, x2, x3, x4):
"""得到该决策情况下【成品的期望成本】
:param x1: 是否检测零配件1
:param x2: 是否检测零配件2
:param x3: 是否检测成品
:param x4: 是否拆解不合格品
零件相关变量:
- q1, q1(零件的次品率)
- p1, p2(进入装配环节的零件的次品率)
- B1, B2(零件的采购价格)
- D1, D2(零件的检测价格)
- C1, C2(零配件的期望成本)
成品相关变量
- p_old(初始的成品次品率)
- p_new(最后的成品次品率)
- A(组装费用)
- R(调换损失)
- H(拆解费用)
- D_f(成品的检测成本)
-
- C_total(成品的期望成本)
"""
# <----零件的期望成本---->
C1 = x1 * (B1 + D1) / (1 - q1) + (1 - x1) * B1 # 零配件 1 的期望成本(采购 + 检测)
p1 = (1 - x1) * q1 # 如果检测, 进入装配环节的零件 1 的次品率为 0, 否则为 q1
```

```
C2 = x2 * (B2 + D2) / (1 - q2) + (1 - x2) * B2 # 零配件 2 的期望成本(采购 + 检测)
p2 = (1 - x2) * q2 # 如果检测，进入装配环节的零件 2 的次品率为 0，否则为 q2

# <----最后成品的期望成本----->
p_new = 1 - (1 - p1) * (1 - p2) + (1 - p1) * (1 - p2) * p_old # 最后的成品次品概率
p_total_new = p_new
"""
四种情况下的总期望成本
decison1 = x3 * (1 - x4)   : [检测但不拆解] 不拆解即不合格品直接丢掉，（两个零件的期望成本 +
    组装费用 + 成品检测成本）/ 合格的成品比率
decison2 = (1 - x3) * (1 - x4) : [不检测且不拆解]
    用户发现不合格品后直接丢掉，（两个零件的期望成本 + 组装费用 + 调换损失）/ 合格的成品比率
decison3 = x3 * x4         : [检测且拆解] 第一轮：两个零件期望成本 + 组装费用 +
    检测成本；后续几轮：（拆解费用 + 组装费用 + 检测费用） * 进入该轮的比率
decison4 = (1 - x3) * x4   : [不检测但拆解] 第一轮：两个零件的期望成本 +
    组装费用；后续几轮：（拆解费用 + 调换损失）* 进入该轮的比率
"""
decison1 = x3 * (1 - x4)
decison2 = (1 - x3) * (1 - x4)
decison3 = x3 * x4
decison4 = (1 - x3) * x4
# 先看不拆解的情况，不需要迭代
if decison1 or decison2:
C_total = decison1 * (C1 + C2 + A + D_f) / (1 - p_new) + decison2 * (C1 + C2 + A + p_new * R)
    / (1 - p_new)
return C_total
# 拆解的情况，需要迭代
if decison3 or decison4:
C_total_cost = decison3 * (C1 + C2 + A + D_f) + decison4 * (C1 + C2 + A + p_new * R)
C_total_num = 1 - p_new
# 在迭代的过程种，成品的期望成本越来越大，直到大于售价，循环停止
while True:
# 更新 p1，p2，p_new
"""
p1：这一阶段零件1的次品率
p2：这一阶段零件2的次品率
p_new：这一阶段最后的成品次品率
p_total_old：进入这一阶段的概率
p_total_new：进入下一阶段的概率
"""
p1_old, p2_old, p_total_old = p1, p2, p_total_new
p1 = p1_old / (1 - (1 - p1_old) * (1 - p2_old) * (1 - p_old))
p2 = p2_old / (1 - (1 - p1_old) * (1 - p2_old) * (1 - p_old))
p_new = 1 - (1 - p1) * (1 - p2) + (1 - p1) * (1 - p2) * p_old # 最后的成品次品概率
p_total_new *= p_new
cost = decison3 * (H + A + D_f) + decison4 * (R * p_new + H + A) # 每次迭代所需要的费用
C_total_cost += cost * p_total_old
```

```
C_total_num += p_total_old * (1 - p_new)
C_total = C_total_cost / C_total_num
print(f'[{variable}]p1={p1}, p2={p2}, p_new={p_new}, p_total_old={p_total_old},
    p_total_new={p_total_new}, C_total_cost={C_total_cost}, C_total_num={C_total_num},
    C_total={C_total}')
if p_total_new < 1e-3 or 1 - p_new < 1e-3:
return C_total



if __name__ == '__main__':
# 第二题的表格数据
params = [
{"q1": 0.1, "q2": 0.1, "p_old": 0.1, "B1": 4, "B2": 18, "A": 6, "D1": 2, "D2": 3, "D_f": 3,
    "P": 56, "R": 6, "H": 5},
{"q1": 0.2, "q2": 0.2, "p_old": 0.2, "B1": 4, "B2": 18, "A": 6, "D1": 2, "D2": 3, "D_f": 3,
    "P": 56, "R": 6, "H": 5},
{"q1": 0.1, "q2": 0.1, "p_old": 0.1, "B1": 4, "B2": 18, "A": 6, "D1": 2, "D2": 3, "D_f": 3,
    "P": 56, "R": 30, "H": 5},
{"q1": 0.2, "q2": 0.2, "p_old": 0.2, "B1": 4, "B2": 18, "A": 6, "D1": 1, "D2": 1, "D_f": 2,
    "P": 56, "R": 30, "H": 5},
{"q1": 0.1, "q2": 0.2, "p_old": 0.1, "B1": 4, "B2": 18, "A": 6, "D1": 8, "D2": 1, "D_f": 2,
    "P": 56, "R": 10, "H": 5},
{"q1": 0.05, "q2": 0.05, "p_old": 0.05, "B1": 4, "B2": 18, "A": 6, "D1": 2, "D2": 3, "D_f": 3,
    "P": 56, "R": 10, "H": 40}
]
# 16种决策情况
variables = [
(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1),
(0, 1, 0, 0), (0, 1, 0, 1), (0, 1, 1, 0), (0, 1, 1, 1),
(1, 0, 0, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 0, 1, 1),
(1, 1, 0, 0), (1, 1, 0, 1), (1, 1, 1, 0), (1, 1, 1, 1),
]
result = []
for param in params:
q1 = param['q1'] # 零配件1的次品率
q2 = param['q2'] # 零配件2的次品率
p_old = param['p_old'] # 初始的成品次品率
B1 = param['B1'] # 零配件1的采购成本
B2 = param['B2'] # 零配件2的采购成本
A = param['A'] # 组装成本
D1 = param['D1'] # 零配件1的检测成本
D2 = param['D2'] # 零配件2的检测成本
D_f = param['D_f'] # 成品的检测成本
P = param['P'] # 售价
R = param['R'] # 调换损失
H = param['H'] # 拆解费用
```

```
temp = []
for variable in variables:
temp.append(get_mean_cost(*variable))
result.append(temp)

df = pd.DataFrame(result)
df.columns = ["0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1000", "1001",
    "1010", "1011",
"1100", "1101", "1110", "1111"]
# 创建一个2行3列的子图布局
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))

# 将DataFrame分为3个子DataFrame
df1 = df.iloc[:2] # 前2行
df2 = df.iloc[2:4] # 中间2行
df3 = df.iloc[4:] # 最后2行

# 创建一个2行3列的子图布局
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))

# 绘制第一个子图
df1.plot(kind='barh', ax=axes[0, 0], legend=False)
axes[0, 0].set_title('Subplot 1')
axes[0, 0].set_xlabel('Values')
axes[0, 0].set_ylabel('Category')

# 绘制第二个子图
df2.plot(kind='barh', ax=axes[0, 1], legend=False)
axes[0, 1].set_title('Subplot 2')
axes[0, 1].set_xlabel('Values')
axes[0, 1].set_ylabel('Category')

# 绘制第三个子图
df3.plot(kind='barh', ax=axes[0, 2], legend=False)
axes[0, 2].set_title('Subplot 3')
axes[0, 2].set_xlabel('Values')
axes[0, 2].set_ylabel('Category')

# 隐藏剩余的子图
for ax in axes.flat:
if not ax.has_data():
ax.set_visible(False)

# 调整子图之间的间距
plt.tight_layout()
plt.show()
```

4.2 决策结果的可视化 (Matlab)

```
%% 导入电子表格中的数据
% 用于从以下电子表格导入数据的脚本:
%
%    工作簿: D:\Desktop\2024数模国赛\result.xlsx
%    工作表: Sheet1
%
% 由 MATLAB 于 2024-09-07 20:40:35 自动生成

%% 设置导入选项并导入数据
opts = spreadsheetImportOptions("NumVariables", 17);

% 指定工作表和范围
opts.Sheet = "Sheet1";
opts.DataRange = "A2:Q7";

% 指定列名称和类型
opts.VariableNames = ["Var1", "VarName2", "VarName3", "VarName4", "VarName5", "VarName6",
    "VarName7", "VarName8", "VarName9", "VarName10", "VarName11", "VarName12", "VarName13",
    "VarName14", "VarName15", "VarName16", "VarName17"];
opts.SelectedVariableNames = ["VarName2", "VarName3", "VarName4", "VarName5", "VarName6",
    "VarName7", "VarName8", "VarName9", "VarName10", "VarName11", "VarName12", "VarName13",
    "VarName14", "VarName15", "VarName16", "VarName17"];
opts.VariableTypes = ["char", "double", "double", "double", "double", "double", "double",
    "double", "double", "double", "double", "double", "double", "double", "double", "double",
    "double"];

% 指定变量属性
opts = setvaropts(opts, "Var1", "WhitespaceRule", "preserve");
opts = setvaropts(opts, "Var1", "EmptyFieldRule", "auto");

% 导入数据
result = readtable("D:\Desktop\2024数模国赛\result.xlsx", opts, "UseExcel", false);

%% 转换为输出类型
result = table2array(result);

%% 清除临时变量
clear opts
labels = ["0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1000", "1001",
    "1010", "1011", "1100", "1101", "1110", "1111"];
titles = ["情境一", "情境二", "情境三", "情境四", "情境五", "情境六"];
figure('Position', [50, 50, 1000, 600])
for i = 1: 6
subplot(2, 3, i)
% 找出最小值及其索引
```

```
[minValue, minIndex] = min(result(i, :));
maxValue = max(result(i, :));
hold on
for j = 1: 16
h = barh(labels(j), result(i, j));
if j == minIndex || i == 3 && j == 16
set(h(1),'facecolor', [209 73 78]/255)
else
set(h(1), 'facecolor', [131 175 155]/255)
end
ylabel("决策方案", 'FontWeight','bold')
xlabel("期望成本(元)", 'FontWeight','bold')
xlim([0 maxValue + 30])
title(titles(i), 'FontWeight','bold','FontSize',12)
end
for k = 1:16
if k == minIndex || i == 3 && k == 16
text(result(i,k) + 3, k + 0.06, num2str(round(result(i,k) * 100)/100), 'FontSize', 6,
    'Color',[209 73 78]/255, 'FontWeight','bold')
else
text(result(i,k) + 3, k + 0.06, num2str(round(result(i,k) * 100)/100), 'FontSize', 6,
    'Color',[0 90 171]/255)
end
end
hold off
end
% sgtitle("六种情境下不同决策方案的期望成本", 'FontWeight', 'bold')
exportgraphics(gcf, 'q2Pic.eps', 'ContentType','vector')
```

附录 E　问题三源代码

5.1　第一阶段的最小化期望成本模型 (Python)

```
from typing import List, Dict
import math
import pandas as pd


def get_mean_cost1(partDetects: Dict[int, int], semiproductDetect: int, disassembly: int) ->
    float:
"""第一阶段，输入决策情况，输出期望成本
:param partDetect: 是否检测零件
:param semiproductDetect: 是否检测半成品
:param disassembly: 是否拆解半成品
-------------------------------------------
```

```
零件相关变量:
- p: 进入装配环节的次品率
- C: 零件的期望成本
"""
# <-----先算零件的期望成本----->
C = {}
p = {}
for k, v in partDetects.items():
C[k] = v * (B[k - 1] + D[k - 1]) / (1 - q[k - 1]) + (1 - v) * B[k - 1] # 零配件的期望成本
p[k] = (1 - v) * q[k - 1] # 进入装配环节的次品率
# <-----再算成品的期望成本----->
# 最后的成品次品率
p_temp = list(map(lambda x: 1 - x, p.values()))
p_new = 1 - math.prod(p_temp) + math.prod(p_temp) * p_old
p_total_new = p_new
# 四种情况
decision1 = semiproductDetect * (1 - disassembly) # 检测但不拆解
decision2 = (1 - semiproductDetect) * (1 - disassembly) # 不检测且不拆解
decision3 = semiproductDetect * disassembly # 检测且拆解
decision4 = (1 - semiproductDetect) * disassembly # 不检测但拆解
# 不拆解，不需要迭代
if decision1 or decision2:
C_total = decision1 * (sum(C.values()) + A + D_f) / (1 - p_new) + decision2 * (sum(C.values())
    + A) / (1 - p_new)
return C_total
# 拆解，需要迭代
if decision3 or decision4:
cost = decision3 * (H + A + D_f) + decision4 * (H + A) # 每次迭代所需要的费用
C_total_cost = decision3 * (sum(C.values()) + A + D_f) + decision4 * (sum(C.values()) + A)
C_total_num = 1 - p_new
C_total = C_total_cost / C_total_num
n = 0
p = p.values()
while True:
"""
p: 这一阶段各个零件的次品率
p_new: 这一阶段最后的成品次品率
p_total_old: 进入这一阶段的概率
p_total_new: 进入下一阶段的概率
"""
p_temp1, p_total_old = p, p_total_new
p_temp2 = list(map(lambda x: 1 - x, p_temp1))
p = [mp / (1 - math.prod(p_temp2) * (1 - p_old)) for mp in p_temp1]
p_temp3 = list(map(lambda x: 1 - x, p))
p_new = 1 - math.prod(p_temp3) + math.prod(p_temp3) * p_old
p_total_new *= p_new
C_total_cost += cost * p_total_old
```

```
C_total_num += p_total_old * (1 - p_new)
C_total = C_total_cost / C_total_num
if p_total_new < 1e-3 or 1 - p_new < 1e-3:
return C_total


if __name__ == '__main__':
B = [2, 8, 12, 2, 8, 12, 8, 12] # 各个零件的采购成本
D = [1, 1, 2, 1, 1, 2, 1, 2] # 各个零件的检测成本
q = [0.1] * 8 # 各个零件的次品率
A = 8       # 装配成本
D_f = 4     # 半成品检测成本
H = 6       # 半成品拆解成本
p_old = 0.1 # 初始的成品次品率
variablesA = [
{1: 0, 2: 0, 3: 0},
{1: 0, 2: 0, 3: 1},
{1: 0, 2: 1, 3: 0},
{1: 0, 2: 1, 3: 1},
{1: 1, 2: 0, 3: 0},
{1: 1, 2: 0, 3: 1},
{1: 1, 2: 1, 3: 0},
{1: 1, 2: 1, 3: 1},
]
variablesB = [
{4: 0, 5: 0, 3: 0},
{4: 0, 5: 0, 6: 1},
{4: 0, 5: 1, 6: 0},
{4: 0, 5: 1, 6: 1},
{4: 1, 5: 0, 6: 0},
{4: 1, 5: 0, 6: 1},
{4: 1, 5: 1, 6: 0},
{4: 1, 5: 1, 6: 1},
]
variablesC = [
{7: 0, 8: 0},
{7: 0, 8: 1},
{7: 1, 8: 0},
{7: 1, 8: 1},
]
params = [
(0, 0),
(0, 1),
(1, 0),
(1, 1),
]
result = {}
```

```
for variable in variablesA:
temp = []
for param in params:
temp.append(get_mean_cost1(variable, *param))
choice = ''.join([str(v) for v in variable.values()])
result[choice] = temp
dfA = pd.DataFrame(result)
dfA.index = ['00', '01', '10', '11']
resultA = {}
for index, row in dfA.iterrows():
for col in dfA.columns:
value = row[col]
resultA[f'{col}{index}'] = value
resultDf = pd.DataFrame(resultA, index=[0])
resultDf.T.to_excel("./Q3_result_A.xlsx")

result = {}
for variable in variablesB:
temp = []
for param in params:
temp.append(get_mean_cost1(variable, *param))
choice = ''.join([str(v) for v in variable.values()])
result[choice] = temp
dfB = pd.DataFrame(result)
dfB.index = ['00', '01', '10', '11']
resultB = {}
for index, row in dfB.iterrows():
for col in dfB.columns:
value = row[col]
resultB[f'{col}{index}'] = value
resultDf = pd.DataFrame(resultB, index=[0])
resultDf.T.to_excel("./Q3_result_B.xlsx")

result = {}
for variable in variablesC:
temp = []
for param in params:
temp.append(get_mean_cost1(variable, *param))
choice = ''.join([str(v) for v in variable.values()])
result[choice] = temp
dfC = pd.DataFrame(result)
dfC.index = ['00', '01', '10', '11']
resultC = {}
print(dfC)
for index, row in dfC.iterrows():
for col in dfC.columns:
value = row[col]
```

```
resultC[f'{index}{col}'] = value
resultDf = pd.DataFrame(resultC, index=[0])
resultDf.T.to_excel("./Q3_result_C.xlsx")
```

5.2 第二阶段的最小化期望成本模型 (Python)

```
def get_mean_cost2(finalProductDetect: int, disassembly: int) -> float:
"""第二阶段，输入半成品的决策情况，输出成品的期望成本"""
# <----半成品的期望成本---->
C1 = 38.446733562245115 # 半成品1的期望成本
C2 = 38.446733562245115 # 半成品2的期望成本
C3 = 35.11306686224178 # 半成品3的期望成本
p = q2 # 进入装配阶段的半成品次品率

# <----成品的期望成本---->
p_new = 1 - (1 - p) ** 3 + (1 - p) ** 3 * p_old2
p_total_new = p_new

# 四种情况
decison1 = finalProductDetect * (1 - disassembly)
decison2 = (1 - finalProductDetect) * (1 - disassembly)
decison3 = finalProductDetect * disassembly
decison4 = (1 - finalProductDetect) * disassembly

if decison1 or decison2:
C_total_cost = decison1 * (C1 + C2 + C3 + A2 + D_f2) + decison2 * (C1 + C2 + C3 + A2 + R *
    p_new)
C_total_num = 1 - p_new
C_total = C_total_cost / C_total_num
return C_total
if decison3 or decison4:
C_total_cost = decison1 * (C1 + C2 + C3 + A2 + D_f2) + decison2 * (C1 + C2 + C3 + A2 + R *
    p_new)
C_total_num = 1 - p_new
n = 0
while True:
p_total_old = p_total_new
p = p / (1 - (1 - p) ** 3 * (1 - p_old2))
p_new = 1 - (1 - p) ** 3 + (1 - p) ** 3 * p_old2
p_total_new *= p_new
cost = decison3 * (H2 + A2 + D_f2) + decison4 * (R * p_new + A2)
C_total_cost += cost * p_total_old
C_total_num += p_total_old * (1 - p_new)
C_total = C_total_cost / C_total_num
if 1 - p_new < 1e-3:
```

```
return C_total


if __name__ == '__main__':
q2 = 0.1   # 半成品的次品率
p_old2 = 0.2 # 初始的成品次品率（装错出错率）
A2 = 8     # 装配成本
D_f2 = 4   # 检测成本
H2 = 10    # 拆解费用
P = 200    # 市场售价
R = 40     # 调换损失
params2 = [
(0, 0),
(0, 1),
(1, 0),
(1, 1),
]
for param in params2:
print(f'[{param}]{get_mean_cost2(*param)}')
```

5.3 第一阶段结果的可视化 (Matlab)

```
load('./A.mat');
load('./C.mat')
x1 = [
"00000", "00100", "01000", "01100", "10000", "10100", ...
"11000", "11100", "00001", "00101", "01001", "01101", ...
"10001", "10101", "11001", "11101", "00010", "00110", ...
"01010", "01110", "10010", "10110", "11010", "11110", ...
"00011", "00111", "01011", "01111", "10011", "10111", ...
"11011", "11111"
];
x2 = ["0000", "0100", "1000", "1100", "0001", "0101", "1001", "1101", "0010", "0110", "1010",
    "1110", "0011", "0111", "1011", "1111"];
figure('Position',[50, 0, 1000, 700])
subplot(121)
[minValue, minIndex] = min(A);
hold on
for j = 1: 32
h = barh(x1(j), A(j));
if j == minIndex
set(h(1),'facecolor', [209 73 78]/255)
text(A(j) + 3, j + 0.06, num2str(round(A(j) * 100)/100), 'FontSize', 8, 'Color',[209 73
    78]/255, 'FontWeight','bold')
else
```

```
set(h(1), 'facecolor', [131 175 155]/255)
text(A(j) + 3, j + 0.06, num2str(round(A(j) * 100)/100), 'FontSize', 8, 'Color',[0 90 171]/255)
end
ylabel("决策方案", 'FontWeight','bold')
xlabel("期望成本(元)", 'FontWeight','bold')
title("生产半成品1和半成品2的决策方案", 'FontSize', 12, 'FontWeight', 'bold')
end
hold off
subplot(122)
[minValue, minIndex] = min(C);
hold on
for j = 1: 16
h = barh(x2(j), C(j));
if j == minIndex
set(h(1),'facecolor', [209 73 78]/255)
text(C(j) + 3, j + 0.06, num2str(round(C(j) * 100)/100), 'FontSize', 10, 'Color',[209 73
    78]/255, 'FontWeight','bold')
else
set(h(1), 'facecolor', [131 175 155]/255)
text(C(j) + 3, j + 0.06, num2str(round(C(j) * 100)/100), 'FontSize', 10, 'Color',[0 90
    171]/255)
end
ylabel("决策方案", 'FontWeight','bold')
xlabel("期望成本(元)", 'FontWeight','bold')
title("生产半成品3的决策方案", 'FontSize', 12, 'FontWeight', 'bold')
end
hold off

exportgraphics(gcf, 'q3Pic.eps', 'ContentType','vector')
```

5.4 第二阶段结果的可视化 (Matlab)

```
x = ["00", "01", "10", "11"];
y = [234.36, 443.16, 212.63, 209.41];
bar(x, y)

[minValue, minIndex] = min(y);
hold on
for j = 1: 4
h = bar(x(j), y(j));
if j == minIndex
set(h(1),'facecolor', [209 73 78]/255)
else
set(h(1), 'facecolor', [131 175 155]/255)
end
```

```
xlabel("决策变量", 'FontWeight','bold', 'Interpreter','tex', 'LineWidth', 2)
ylabel("期望成本(元)", 'FontWeight','bold')
end

set(gca, 'box', 'off', 'tickdir', 'out', 'ticklength', [.005 .005])
exportgraphics(gcf, 'q3_2.eps', 'ContentType','vector')
```

附录 F　问题四源代码

6.1　用蒙特卡洛模拟重新解决问题二 (Python)

```
import numpy as np
from scipy.stats import beta, binom
import pandas as pd
from collections import Counter
from tqdm import tqdm

d = []
alpha_prior = 1
beta_prior = 1

# 次品率
pro2_defective_rate =
    [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]
pro3_defective_rate = [0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1]
# 模拟抽样
n_samples = 609
n_total = 609
iter_num = 1000
params = [
{"q1": 0.1, "q2": 0.1, "p_old": 0.1, "B1": 4, "B2": 18, "A": 6, "D1": 2, "D2": 3, "D_f": 3,
    "P": 56, "R": 6, "H": 5},
{"q1": 0.2, "q2": 0.2, "p_old": 0.2, "B1": 4, "B2": 18, "A": 6, "D1": 2, "D2": 3, "D_f": 3,
    "P": 56, "R": 6, "H": 5},
{"q1": 0.1, "q2": 0.1, "p_old": 0.1, "B1": 4, "B2": 18, "A": 6, "D1": 2, "D2": 3, "D_f": 3,
    "P": 56, "R": 30, "H": 5},
{"q1": 0.2, "q2": 0.2, "p_old": 0.2, "B1": 4, "B2": 18, "A": 6, "D1": 1, "D2": 1, "D_f": 2,
    "P": 56, "R": 30, "H": 5},
{"q1": 0.1, "q2": 0.2, "p_old": 0.1, "B1": 4, "B2": 18, "A": 6, "D1": 8, "D2": 1, "D_f": 2,
    "P": 56, "R": 10, "H": 5},
{"q1": 0.05, "q2": 0.05, "p_old": 0.05, "B1": 4, "B2": 18, "A": 6, "D1": 2, "D2": 3, "D_f": 3,
    "P": 56, "R": 10, "H": 40}
]
# 16种决策情况
variables = [
```

```
(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1),
(0, 1, 0, 0), (0, 1, 0, 1), (0, 1, 1, 0), (0, 1, 1, 1),
(1, 0, 0, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 0, 1, 1),
(1, 1, 0, 0), (1, 1, 0, 1), (1, 1, 1, 0), (1, 1, 1, 1),
]
resultList = [[] for _ in range(6)]
counterList = [Counter() for _ in range(6)]

def get_mean_cost(x1, x2, x3, x4):
"""得到该决策情况下【成品的期望成本】
:param x1: 是否检测零配件1
:param x2: 是否检测零配件2
:param x3: 是否检测成品
:param x4: 是否拆解不合格品
零件相关变量:
- q1, q1(零件的次品率)
- p1, p2(进入装配环节的零件的次品率)
- B1, B2(零件的采购价格)
- D1, D2(零件的检测价格)
- C1, C2(零配件的期望成本)
成品相关变量
- p_old(初始的成品次品率)
- p_new(最后的成品次品率)
- A(组装费用)
- R(调换损失)
- H(拆解费用)
- D_f(成品的检测成本)
-
- C_total(成品的期望成本)
"""
# <----零件的期望成本---->
C1 = x1 * (B1 + D1) / (1 - q1) + (1 - x1) * B1 # 零配件 1 的期望成本(采购 + 检测)
p1 = (1 - x1) * q1 # 如果检测，进入装配环节的零件 1 的次品率为 0，否则为 q1
C2 = x2 * (B2 + D2) / (1 - q2) + (1 - x2) * B2 # 零配件 2 的期望成本(采购 + 检测)
p2 = (1 - x2) * q2 # 如果检测，进入装配环节的零件 2 的次品率为 0，否则为 q2

# <----最后成品的期望成本----->
p_new = 1 - (1 - p1) * (1 - p2) + (1 - p1) * (1 - p2) * p_old # 最后的成品次品概率
p_total_new = p_new
"""
四种情况下的总期望成本
decison1 = x3 * (1 - x4)   : [检测但不拆解] 不拆解即不合格品直接丢掉，（两个零件的期望成本 +
    组装费用 + 成品检测成本）/ 合格的成品比率
decison2 = (1 - x3) * (1 - x4) : [不检测且不拆解]
    用户发现不合格品后直接丢掉，（两个零件的期望成本 + 组装费用 + 调换损失）/ 合格的成品比率
decison3 = x3 * x4          : [检测且拆解] 第一轮：两个零件期望成本 + 组装费用 +
    检测成本；后续几轮：（拆解费用 + 组装费用 + 检测费用） * 进入该轮的比率
```

```
decison4 = (1 - x3) * x4   : [不检测但拆解] 第一轮：两个零件的期望成本 +
    组装费用；后续几轮：（拆解费用 + 调换损失）* 进入该轮的比率
"""
decison1 = x3 * (1 - x4)
decison2 = (1 - x3) * (1 - x4)
decison3 = x3 * x4
decison4 = (1 - x3) * x4
# 先看不拆解的情况，不需要迭代
if decison1 or decison2:
C_total = decison1 * (C1 + C2 + A + D_f) / (1 - p_new) + decison2 * (C1 + C2 + A + p_new * R)
    / (1 - p_new)
return C_total
# 拆解的情况，需要迭代
if decison3 or decison4:
C_total_cost = decison3 * (C1 + C2 + A + D_f) + decison4 * (C1 + C2 + A + p_new * R)
C_total_num = 1 - p_new
# 在迭代的过程种，成品的期望成本越来越大，直到大于售价，循环停止
while True:
# 更新 p1，p2，p_new
"""
p1：这一阶段零件1的次品率
p2：这一阶段零件2的次品率
p_new：这一阶段最后的成品次品率
p_total_old：进入这一阶段的概率
p_total_new：进入下一阶段的概率
"""
p1_old, p2_old, p_total_old = p1, p2, p_total_new
p1 = p1_old / (1 - (1 - p1_old) * (1 - p2_old) * (1 - p_old))
p2 = p2_old / (1 - (1 - p1_old) * (1 - p2_old) * (1 - p_old))
p_new = 1 - (1 - p1) * (1 - p2) + (1 - p1) * (1 - p2) * p_old # 最后的成品次品概率
p_total_new *= p_new
cost = decison3 * (H + A + D_f) + decison4 * (R * p_new + H + A) # 每次迭代所需要的费用
C_total_cost += cost * p_total_old
C_total_num += p_total_old * (1 - p_new)
C_total = C_total_cost / C_total_num
if p_total_new < 1e-3 or 1 - p_new < 1e-3:
return C_total

for _ in tqdm(range(iter_num)):
a = []
for i in range(0,18):
observed_defectives = binom.rvs(1, pro2_defective_rate[i], size=n_samples).sum()
alpha_posterior = alpha_prior + observed_defectives
beta_posterior = beta_prior + n_samples - observed_defectives
posterior_samples = beta.rvs(alpha_posterior, beta_posterior, size=10000)

mean_posterior = np.mean(posterior_samples)
```

```
a.append(mean_posterior)

c = [round(b, 4) for b in a]
new_c = [tuple(c[i:i+3]) for i in range(0, 18, 3)]
for idx, lt in enumerate(new_c):
params[idx]['q1'] = lt[0]
params[idx]['q2'] = lt[1]
params[idx]['p_old'] = lt[2]

result = []
for param in params:
q1 = param['q1'] # 零配件1的次品率
q2 = param['q2'] # 零配件2的次品率
p_old = param['p_old'] # 初始的成品次品率
B1 = param['B1'] # 零配件1的采购成本
B2 = param['B2'] # 零配件2的采购成本
A = param['A'] # 组装成本
D1 = param['D1'] # 零配件1的检测成本
D2 = param['D2'] # 零配件2的检测成本
D_f = param['D_f'] # 成品的检测成本
P = param['P'] # 售价
R = param['R'] # 调换损失
H = param['H'] # 拆解费用
temp = []
for variable in variables:
temp.append(get_mean_cost(*variable))
result.append(temp)

df = pd.DataFrame(result)
df.columns = ["0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1000", "1001",
    "1010", "1011",
"1100", "1101", "1110", "1111"]
minIds = tuple(df.idxmin(axis=1))
for idx, minId in enumerate(minIds):
resultList[idx].append(minId)

for idx, result in enumerate(resultList):
counterList[idx].update(result)
print(counterList)
```

6.2 用蒙特卡洛模拟重新解决问题三 (Python)

```
import numpy as np
from scipy.stats import beta, binom
import pandas as pd
```

```python
from typing import List, Dict
import math
from tqdm import tqdm
from collections import Counter


# 结果保存
resultList = []
c = Counter()
a = []

alpha_prior = 1
beta_prior = 1

# 次品率
pro2_defective_rate =
    [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]
pro3_defective_rate = [0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1]
# 模拟抽样
n_samples = 609
n_total = 609
iter_num = 1000

# 需要的变量
B = [2, 8, 12, 2, 8, 12, 8, 12] # 各个零件的采购成本
D = [1, 1, 2, 1, 1, 2, 1, 2] # 各个零件的检测成本
q = [0.1] * 8 # 各个零件的次品率
A = 8       # 装配成本
D_f = 4     # 半成品检测成本
H = 6       # 半成品拆解成本
p_olds = [0.1] * 3 # 初始的成品次品率
variablesA = [
{1: 0, 2: 0, 3: 0},
{1: 0, 2: 0, 3: 1},
{1: 0, 2: 1, 3: 0},
{1: 0, 2: 1, 3: 1},
{1: 1, 2: 0, 3: 0},
{1: 1, 2: 0, 3: 1},
{1: 1, 2: 1, 3: 0},
{1: 1, 2: 1, 3: 1},
]
variablesB = [
{4: 0, 5: 0, 3: 0},
{4: 0, 5: 0, 6: 1},
{4: 0, 5: 1, 6: 0},
{4: 0, 5: 1, 6: 1},
{4: 1, 5: 0, 6: 0},
```

```
{4: 1, 5: 0, 6: 1},
{4: 1, 5: 1, 6: 0},
{4: 1, 5: 1, 6: 1},
]
variablesC = [
{7: 0, 8: 0},
{7: 0, 8: 1},
{7: 1, 8: 0},
{7: 1, 8: 1},
]
params = [
(0, 0),
(0, 1),
(1, 0),
(1, 1),
]

q2 = 0.1   # 半成品的次品率
p_old2 = 0.2 # 初始的成品次品率（装错出错率）
A2 = 8     # 装配成本
D_f2 = 4   # 检测成本
H2 = 10    # 拆解费用
P = 200    # 市场售价
R = 40     # 调换损失
params2 = [
(0, 0),
(0, 1),
(1, 0),
(1, 1),
]

def get_mean_cost1(partDetects: Dict[int, int], semiproductDetect: int, disassembly: int,
    type: int) -> float:
"""第一阶段，输入决策情况，输出期望成本
:param partDetect: 是否检测零件
:param semiproductDetect: 是否检测半成品
:param disassembly: 是否拆解半成品
----------------------------------------------
零件相关变量:
- p: 进入装配环节的次品率
- C: 零件的期望成本
"""
# <-----先算零件的期望成本----->
C = {}
p = {}
p_old = p_olds[type]
for k, v in partDetects.items():
```

```
C[k] = v * (B[k - 1] + D[k - 1]) / (1 - q[k - 1]) + (1 - v) * B[k - 1] # 零配件的期望成本
p[k] = (1 - v) * q[k - 1] # 进入装配环节的次品率
# <-----再算成品的期望成本----->
# 最后的成品次品率
p_temp = list(map(lambda x: 1 - x, p.values()))
p_new = 1 - math.prod(p_temp) + math.prod(p_temp) * p_old
p_total_new = p_new
# 四种情况
decision1 = semiproductDetect * (1 - disassembly) # 检测但不拆解
decision2 = (1 - semiproductDetect) * (1 - disassembly) # 不检测且不拆解
decision3 = semiproductDetect * disassembly # 检测且拆解
decision4 = (1 - semiproductDetect) * disassembly # 不检测但拆解
# 不拆解，不需要迭代
if decision1 or decision2:
C_total = decision1 * (sum(C.values()) + A + D_f) / (1 - p_new) + decision2 * (sum(C.values())
    + A) / (1 - p_new)
return C_total
# 拆解，需要迭代
if decision3 or decision4:
cost = decision3 * (H + A + D_f) + decision4 * (H + A) # 每次迭代所需要的费用
C_total_cost = decision3 * (sum(C.values()) + A + D_f) + decision4 * (sum(C.values()) + A)
C_total_num = 1 - p_new
C_total = C_total_cost / C_total_num
n = 0
p = p.values()
while True:
"""
p: 这一阶段各个零件的次品率
p_new: 这一阶段最后的成品次品率
p_total_old: 进入这一阶段的概率
p_total_new: 进入下一阶段的概率
"""
p_temp1, p_total_old = p, p_total_new
p_temp2 = list(map(lambda x: 1 - x, p_temp1))
p = [mp / (1 - math.prod(p_temp2) * (1 - p_old)) for mp in p_temp1]
p_temp3 = list(map(lambda x: 1 - x, p))
p_new = 1 - math.prod(p_temp3) + math.prod(p_temp3) * p_old
p_total_new *= p_new
C_total_cost += cost * p_total_old
C_total_num += p_total_old * (1 - p_new)
C_total = C_total_cost / C_total_num
if p_total_new < 1e-3 or 1 - p_new < 1e-3:
return C_total


def get_mean_cost2(finalProductDetect: int, disassembly: int) -> float:
"""第二阶段，输入半成品的决策情况，输出成品的期望成本"""
```

```
p = q2 # 进入装配阶段的半成品次品率

# <----成品的期望成本---->
p_new = 1 - (1 - p) ** 3 + (1 - p) ** 3 * p_old2
p_total_new = p_new

# 四种情况
decison1 = finalProductDetect * (1 - disassembly)
decison2 = (1 - finalProductDetect) * (1 - disassembly)
decison3 = finalProductDetect * disassembly
decison4 = (1 - finalProductDetect) * disassembly

if decison1 or decison2:
C_total_cost = decison1 * (C1 + C2 + C3 + A2 + D_f2) + decison2 * (C1 + C2 + C3 + A2 + R *
    p_new)
C_total_num = 1 - p_new
C_total = C_total_cost / C_total_num
return C_total
if decison3 or decison4:
C_total_cost = decison1 * (C1 + C2 + C3 + A2 + D_f2) + decison2 * (C1 + C2 + C3 + A2 + R *
    p_new)
C_total_num = 1 - p_new
n = 0
while True:
p_total_old = p_total_new
p = p / (1 - (1 - p) ** 3 * (1 - p_old2))
p_new = 1 - (1 - p) ** 3 + (1 - p) ** 3 * p_old2
p_total_new *= p_new
cost = decison3 * (H2 + A2 + D_f2) + decison4 * (R * p_new + A2)
C_total_cost += cost * p_total_old
C_total_num += p_total_old * (1 - p_new)
C_total = C_total_cost / C_total_num
if 1 - p_new < 1e-3:
return C_total

for _ in tqdm(range(iter_num)):
d = []
for i in range(0,12):
observed_defectives = binom.rvs(1, pro3_defective_rate[i], size=n_samples).sum()
alpha_posterior = alpha_prior + observed_defectives
beta_posterior = beta_prior + n_samples - observed_defectives
posterior_samples = beta.rvs(alpha_posterior, beta_posterior, size=10000)

mean_posterior = np.mean(posterior_samples)
d.append(mean_posterior)

f = [round(e, 4) for e in d]
```

```
# 更改次品率变量
q = f[0:8]
p_olds = f[8:11]
p_old2 = f[11]
# 最后的决策
phase1_choice = ''
phase2_choice = ''
# 计算生产三个半成品的期望成本和对应的决策方案
# 第一个半成品
result = {}
for variable in variablesA:
temp = []
for param in params:
temp.append(get_mean_cost1(variable, *param, 0))
choice = ''.join([str(v) for v in variable.values()])
result[choice] = temp
dfA = pd.DataFrame(result)
dfA.index = ['00', '01', '10', '11']
resultA = {}
for index, row in dfA.iterrows():
for col in dfA.columns:
value = row[col]
resultA[f'{col}{index}'] = value
min_A = min(resultA.values())
C1 = min_A
for k, v in resultA.items():
if v == min_A:
phase1_choice += k

# 第二个半成品
result = {}
for variable in variablesB:
temp = []
for param in params:
temp.append(get_mean_cost1(variable, *param, 1))
choice = ''.join([str(v) for v in variable.values()])
result[choice] = temp
dfB = pd.DataFrame(result)
dfB.index = ['00', '01', '10', '11']
resultB = {}
for index, row in dfB.iterrows():
for col in dfB.columns:
value = row[col]
resultB[f'{col}{index}'] = value
min_B = min(resultB.values())
C2 = min_B
```

```
for k, v in resultB.items():
if v == min_B:
phase1_choice += k

# 第三个半成品
result = {}
for variable in variablesC:
temp = []
for param in params:
temp.append(get_mean_cost1(variable, *param, 2))
choice = ''.join([str(v) for v in variable.values()])
result[choice] = temp
dfC = pd.DataFrame(result)
dfC.index = ['00', '01', '10', '11']
resultC = {}
for index, row in dfC.iterrows():
for col in dfC.columns:
value = row[col]
resultC[f'{col}{index}'] = value
min_C = min(resultC.values())
C3 = min_C
for k, v in resultC.items():
if v == min_C:
phase1_choice += k

# 第二阶段
result2 = {}
for param in params2:
result2[param] = get_mean_cost2(*param)

min_2 = min(result2.values())
for k, v in result2.items():
if v == min_2:
k = [str(kk) for kk in k]
phase2_choice = ''.join(k)

choice = phase1_choice + phase2_choice

resultList.append(choice)

c.update(resultList)
print(c)
```

6.3 问题二结果的可视化 (Matlab)

```
figure("Position", [20 20 600 500])
subplot(3, 2, 1)
pie(1000)
legend("1101", 'Location','southeastoutside')
title('情境一', 'FontWeight','bold', 'Position', [2, 0.3, 0])
subplot(3, 2, 2)
pie(1000)
legend("1000", 'Location', 'southeastoutside')
title('情境二', 'FontWeight','bold', 'Position', [2, 0.3, 0])
subplot(3, 2, 3)
pie([516 484])
legend("1111", "1101", 'Location', 'southeastoutside')
title('情境三', 'FontWeight','bold', 'Position', [2, 0.3, 0])
subplot(3, 2, 4)
pie(1000)
legend("1111", 'Location', 'southeastoutside')
title('情境四', 'FontWeight','bold', 'Position', [2, 0.3, 0])
subplot(3, 2, 5)
pie([716 232 2])
legend("0100", "0110", "1101", 'Location', 'southeastoutside')
title('情境五', 'FontWeight','bold', 'Position', [2, 0.3, 0])
subplot(3, 2, 6)
pie([845 155])
legend("0000", "1000", 'Location', 'southeastoutside')
title('情境六', 'FontWeight','bold', 'Position', [2, 0.3, 0])

exportgraphics(gcf, 'q_4_1.eps', 'ContentType', 'vector')
```

6.4 问题三结果的可视化 (Matlab)

```
pie(1000)
legend("1110111101110111", 'Location','eastoutside')
exportgraphics(gcf, 'q4_2.eps', 'ContentType','vector')
```