基于离散粒子群算法的生产过程决策问题分析

摘要

企业在生产一批产品时，需要综合考虑各环节的成本和利润，并做出多项决策，包括是否接受零配件，是否对零配件、半成品和成品进行检测，以及对不合格半成品、成品是否进行拆解。这些决策最终都会影响到生产成本和最终利润，因此针对不同情形，制定合适的决策方案对企业来说至关重要。

针对零配件的接收问题，由于零配件样本量较大，可以近似为**正态分布**，在 95% 和 90% 两种不同信度下，可以分别得到两个对应的单尾临界值。假设实际次品率标准差为 0.01 和 0.02，通过**样本容量计算公式**，计算出两种情形所需的最小抽样检测数分别为 900 和 225。利用**蒙特卡洛法模拟**实际次品率稍大于标称值时的情形，得到的结果符合题目要求。

针对 2 个零配件、1 道工序的决策问题，定义表示是否检测和拆解的决策变量，列出利润的函数表达式作为目标函数，建立一个最大化利润值的**优化模型**，并使用**离散型粒子群算法（DPSO）**求解，各种情形的最终结果已在论文中给出。对输入不同的零配件数进行单个成品利润的**灵敏度分析**，得到各样本中的方差基本维持在 0.1 以下，可以认为模型的鲁棒性较高。

针对多个零配件、多道工序的决策问题，n 个零配件、m 道工序的检测和拆解决策，可以根据每道工序不检测所导致的下一道工序中半成品不断增大的实际次品率来确定；对于 8 个零配件、2 道工序，同样可以定义表示是否检测和拆解的决策变量，建立最大化利润值作为目标函数的**优化模型**，使用 DPSO 算法求解，最终决策方案和依据的利润结果在论文中给出。

针对不确定次品率的决策问题，按照问题 1 中抽样检测方案，实际次品率服从正态分布，假设其标准差为 0.2，可得到不同次品率的**置信区间**，重新完成问题 2、3 时，以步长 0.005 遍历置信区间，以遍历结果作为实际次品率重新做出检测和拆卸决策，最终决策方案和依据的利润结果在论文中给出。

求解模型使用启发式算法 DPSO 显著提高了求解效率，并进行模型分析明确了改进方案与推广方向。

关键字：正态分布　蒙特卡洛模拟　优化模型　DPSO 算法　灵敏度分析

一、 问题重述

1.1 问题的背景

在电子产品制造领域，由于产品结构的复杂性和消费者对品质的高要求，所以生产过程中对零件与成品质量的把控尤为重要。且为了减少能源与不可再生资源的消耗，许多企业考虑逆向物流运作[1]，以回收不合格的产品。为了降低成本，电子产品生产企业需要在零件购买、成品检测、不合格品回收拆解等方面进行决策。

1.2 问题的要求

问题一：此问题要求设计一种检测次数尽可能少的方案，利用统计假设检验方法来确定样本量，以判断从供应商处购买的零配件次品率是否超过标称值。在 95% 的置信度下，如果次品率超过标称值则拒收；在 90% 的置信度下，如果次品率未超过标称值则接收。

问题二：根据已知的零配件和成品次品率，企业需在生产过程的各个阶段做出决策，包括是否检测零配件、是否检测成品、是否拆解不合格成品，以及如何处理用户退回的不合格品。决策需考虑检测成本、装配成本、市场售价、调换损失和拆解费用等因素，并对特定情境给出具体决策方案和依据。

问题三：针对 m 道工序、n 个零配件的多工序多零件生产流程，对于已给定的次品率，同样进行是否检测、如何处理半成品与成品中不合格品的决策；针对具体的两道工序，8 个零配件问题，并给出决策具体方案及指标。

问题四：基于抽样检测结果再次进行决策，即假设零配件、半成品和成品的次品率均通过抽样检测获得，重新完成问题二和问题三的决策过程，以评估次品率不确定性对决策的影响。

二、 问题分析

2.1 零件的抽样检测方案设计

情形（1）中，根据题目给出的 95% 的信度，可以确定一个次品率区间 (例如 $[0, \mu + 2\sigma]$)，实际次品率在这个区间中的可能性为 95%，即可以确保在 95% 的置信水平下，如果实际次品率超过 10%，能够检测到这个问题，并拒绝接受这批零配件。根据次品率区间右端点 P_c 计算出的样本容量的值，可以认为这是做出是否拒收的判断的最小样本容量, 当企业抽取不小于这个样本容量的零配件进行检测时，测得的次品率有 95% 的可信度，即我们的抽样检测方案。

情形（2）同理，计算出的最小样本容量可以确保在 90% 的置信水平下，如果实际次品率不超过 10%，能够检测到这个情况，并接收这批零配件。

2.2 生产过程的决策问题分析

对于企业在生产过程中遇到的是否检测、是否拆检等问题，需要给出具体的指标来进行决策。为了减小成本，提高收益与效率，我们将利润最大化定为决策依据，并尽可能减少拆解次数以提高效率。对于 6 种不同的情况，我们分别进行计算，用离散粒子群算法求解每种情况，得出利润最大的情况作为此种情况下的最优决策方案。

2.3 多工序多零配件生产的最优决策问题分析

针对多道工序与多零配件的生产过程，仅考虑次品率，根据次品率的变化来判断在生产过程中是否采取进一步的行动，如继续生产、检测或返工。此方案的主要目标是保证产品质量稳定，同时尽量减少冗余的操作。

针对两道工序、8 个零配件的生产过程，为了最大化生产利润，同时考虑工序间的质量要求、检测资源的有效配置以及不合格成品拆解的经济性，继续采用离散粒子群算法进行求解，通过多次迭代，最终输出一个最优或接近最优的生产决策方案。

2.4 基于不确定性次品率的生产决策问题分析

由于零配件、半成品和成品的次品率均通过抽样检测方法获得，是统计估计值，增加了决策过程中的不确定性。因此，在重新分析问题二和问题三时，需要考虑抽样误差对决策结果的影响。同时，决策模型应能灵活调整以适应不同置信水平下的次品率估计，从而确保在不确定性条件下仍能制定出高效的生产决策方案。

三、 模型假设

1. 假设供应商提供的零件数量较大，即样本容量大于 30，可将其看作正态分布 [2]。
2. 假设企业生产出的成品全部进入用户手中。
3. 忽略丢弃不合格零配件、半成品与成品对环境产生的影响。
4. 假设零件一与零件二经过检测、组成成品后，多余的零件不再计入后续的组装。
5. 假设多工序所得的成品拆解后为零件，而非拆解为半成品。
6. 假设企业无条件为用户调换的产品均已经过检测，为合格产品。
7. 假设采购的不同类型零件的个数相同。
8. 假设企业对不合格成品至多进行一次拆解。

四、 符号说明

表 1 列出了本文用到的部分符号，其它符号将在文中出现时进行解释。

表 **1**　符号说明

符号	符号说明
n_0	样本容量（抽检的样本个数）
α	显著性水平
P_0	标称值
ε	精确度
Z	正态分布分位数
S	总利润
N	成品数
h	成品售价
Q	成本
W	拆解得失
q_i （$i=1,2,3$）	零配件 1、2 与成品的次品率
p_i （$i=1,2,3,4,5$）	零配件 1、2 与成品的检测费用，调换损失，拆解费用
t	各个零配件的购入数量
T_1	零配件 1，2，3 的配对数
T_2	零配件 4，5，6 的配对数
T_3	零配件 7，8 的配对数
T_4	半成品 1，2，3 的配对数
P_{ij}	粒子在第 j 维度上更新其位置的概率

五、 模型的建立与求解

5.1 零配件的抽样检测方案设计

5.1.1 模型建立

假设抽样检测的零配件次品数服从二项分布，题目中供应商供应的零配件样本容量较大，可近似认为其实际次品率服从正态分布，即 $P \sim N\left(\mu, \sigma^2\right)$，则 [3]：

$$f(P)=\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(P-\mu)^2}{2\sigma^2}} . \tag{1}$$

其中均值 μ 为标称值，P 为实际次品率。依中心极限定理，则有如下检验统计量 [4]：

$$Z=\frac{P_c-P_0}{\sqrt{P_0\left(1-P_0\right)/n_0}}, \tag{2}$$

可得样本容量为：

$$n_0=\frac{Z^2 \cdot P_0 \cdot (1-P_0)}{\varepsilon^2}, \tag{3}$$

其中，

$$\varepsilon = P_c - P_0 . \tag{4}$$

P_c 为次品率置信区间 $[0, u+Z \cdot \sigma]$ 右端点。

5.1.2 模型求解

对于情形（1），由于实际次品率服从正态分布，可查找到 95% 置信水平对应的单尾临界值 $Z \approx 1.645$，均值 $u=0.1$，可假设其标准差 $\sigma=0.01$，则置信区间为 $[0, 0.11645]$，$P_c=0.11645$，可计算出最小样本容量的值。依据（3）式，可得 $n_0=900$，则当企业抽取不小于 900 个零配件进行检测时，可以确保在 95% 信度下，如果实际次品率超过标称值，能够检测到这个问题，并拒绝接收这批零配件。

对于情形（2），同样的，可查找到 90% 置信水平对应的单尾临界值 $Z \approx 1.282$，均值 $u=0.1$，可假设其标准差 $\sigma=0.02$，则置信区间为 $[0, 0.12564]$，$P_c=0.12564$，可计算出最小样本容量的值。依据（3）式，可得 $n_0=225$，则当企业抽取不小于 225 个零配件进行检测时，可以确保在 90% 信度下，如果实际次品率不超过标称值，能够检测到这个情况，并接收这批零配件。

5.1.3 模型检验与结果分析

使用蒙特卡洛模拟进行模型检验，matlab 进行编程。蒙特卡洛模拟是一种利用随机样本来解决复杂问题的统计方法，通过生成大量随机样本，并利用这些样本的统计特性来估算复杂系统的性质 [7]。本模型得到的最小抽样结果在实际运用时得到的实际次品率

分布情况可通过蒙特卡洛法进行模拟，在不同信度下，实际次品率稍高时，可能得到的次品率分布情况模拟结果结果如图 1 所示。

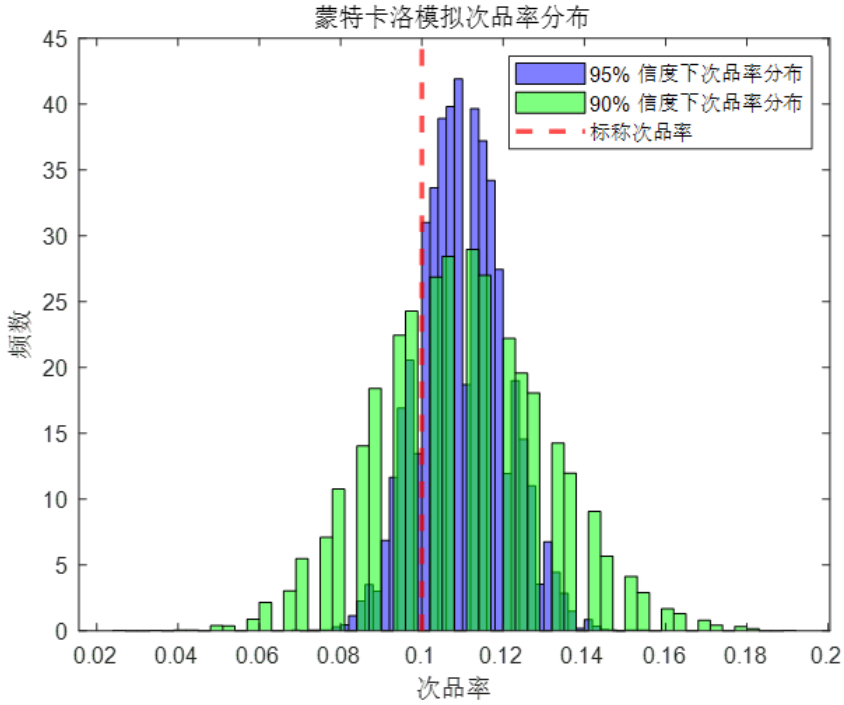

图 **1**　蒙特卡洛模拟示意图

在设定真实次品率为 0.11 时，输出结果为：

```
At 95% confidence level: Decision: Reject, Mean defect rate: 0.1099
At 90% confidence level: Decision: Reject, Mean defect rate: 0.1098
```

抽样得到的平均次品率相当接近于设定值，可以判断通过模型计算出的最小样本量很好的满足了企业要求。选取不同的标准差，可以得到不同的最小样本容量，标准差的选取取决于企业允许的最小误差，在一定的标准差下，企业可以使用简单随机抽样的方式随机抽取不低于一定量的零配件进行检测，此时可以满足企业做出决策所需的信度要求。

5.2 生产过程的决策问题分析

5.2.1 模型分析

在企业生产流程中，针对零配件及成品的检测与拆解决策问题，我们设定利润函数作为决策的核心依据，该函数综合考虑成品数量、销售价格、各项成本（包括购买成本、检测成本、装配成本、调换损失及拆解费用）以及拆解后可能获得的收益或损失。通过

对六种不同生产情况的分析，遍历每一种可能的检测与拆解策略组合，计算出相应的利润值。同时，为了兼顾成本效益与生产效率，采取减少拆解次数的方法，以进一步提升整体生产效率。

5.2.2 利润函数优化模型的建立

设零配件 1 与零配件 2 数量均为 t，则：

$$f(x) = S(t) = N \cdot h - Q + W \,. \tag{5}$$

设 X_i=0 或 1，i=1，2，3，4，分别表示是否对零配件 1、零配件 2、成品做检测与是否拆解。其中，当 X_i 的值为 0 时，表示不进行检测（或拆解）；当 X_i 的值为 1 时，表示进行检测（或拆解）。

建立一个优化模型：

$$\max f(x), \text{subject to } X_i = 0, 1, i = 1, 2, 3, 4 \,. \tag{6}$$

在步骤 1 中，由于零配件 1 与零配件 2 不一定都进行检测，所以零配件 1 与零配件 2 配对得的成品数为：

$$T = \min\left[(t - X_1 \cdot t \cdot q_1), (t - X_2 \cdot t \cdot q_2)\right] , \tag{7}$$

因为零配件 1 与零配件 2 的次品率独立，所以组装成成品的零配件 1、零配件 2 均为合格品的期望为：

$$E = (1 - q_1)^{1 - X_1} \cdot (1 - q_2)^{1 - X_2} \,, \tag{8}$$

成品中的实际合格品的数量为：

$$N_1 = E \cdot T \cdot (1 - q_3) \,, \tag{9}$$

而在步骤 2 中需要考虑是否对成品进行检测，检测出的不合格成品数量为：

$$t' = X_3 \cdot (T - N_1) \,, \tag{10}$$

则流入市场的成品数为:

$$N = T - t' \,. \tag{11}$$

对于从购入零配件到调换的总成本：

$$Q = Q_1 + Q_2 + Q_3 + Q_4 \,, \tag{12}$$

其中，假设步骤 4 中调换的产品经过了全部检测，为合格产品，所以，

$$\begin{cases} \text{购入成本} \quad Q_1 = 22t \,, \\ \text{检测成本} \quad Q_2 = X_1 \cdot t \cdot p_1 + X_2 \cdot t \cdot p_2 + X_3 \cdot T \cdot p_3 \,, \\ \text{装配成本} \quad Q_3 = 6T \,, \\ \text{调换费用} \quad Q_4 = (1 - X_3)\,(22 + 6 + p_1 + p_2 + p_3 + p_4) \cdot (N - N_1) \,. \end{cases} \tag{13}$$

对于步骤 3 中的拆解得失，为了兼顾成本效益与生产效率，进行简化，假设企业最多进行一次拆解，使此时的 $S = N \cdot h - Q$，令 $t = T \cdot q_3$，得出：

$$W = -T \cdot q_3 \cdot p_5 + S(T \cdot q_3)\,. \tag{14}$$

综上所述，求出 max S 对应的生产过程决策即为题解。

5.2.3 离散粒子群算法（DPSO）分析

在应对此复杂的生产环境时，蒙特卡洛模拟逼近统计量分布 [7] 或者遍历求解的方法，会使得计算量庞大。因此，我们考虑采用启发式算法，引入了离散粒子群算法，来缩小搜索空间。此算法启动于随机生成的粒子群，每个粒子均映射为一个的生产决策集合，涵盖零配件检测、成品检验及不合格品拆解决策等。通过计算各粒子的适应度（即基于当前决策的预估生产利润），算法在迭代循环中促使粒子向个体及全局最优解演进，直至收敛至最优或接近最优的决策方案。

在粒子群优化算法（PSO）对鸟群捕食行为的模拟中，每只鸟被命名为一个没有质量和体积的粒子 [8]，即一个由粒子组成的群体在 D 维搜索空间中以一定速度飞行，每个粒子在搜索时, 考虑到了自己搜索到的历史最好点和群体内（或邻域内）其他粒子的历史最好点，在此基础上变化位置（位置也就是解）[9]。

5.2.4 离散粒子群算法实现模型求解

在连续的粒子群算法中，粒子本身在飞行过程所经历过的最好位置称为个体最优值（$pBest$），整个群体目前所经历过的最好位置称为全局最优值（$gBest$）。粒子状态用 D 维速度 $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$ 和位置 $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$ 表示，则每个粒子根据下面的公式来更新自己的状态，从而产生新一代群体 [8]。

$$v_{id}^{k+1} = w v_{id}^{k} + c_1 r_{1d}^{k} \left(pBest_{id}^{k} - p_{id}^{k}\right) + c_2 r_{2d}^{k} \left(gBest_{id}^{k} - p_{id}^{k}\right)\,, \tag{15}$$

$$p_{id}^{k} = p_{id}^{k} + v_{id}^{k+1}\,, \tag{16}$$

其中，w 为惯性权值，c_1，c_2 为常数，称为学习因子，r_{1d} 和 r_{2d} 是［0，1］内的随机数，这里的 k 为迭代次数 [8]。

因为问题二中所形成的生产决策不连续，所以我们采用离散的粒子群算法来建模求解。

首先对粒子进行初始化定义一个粒子的 4 维位置为 $x_i = (x_{i1}, x_{i2}, \cdots, x_{i4})$,，定义一个粒子的 4 维速度为 $v_i = (v_{i1}, v_{i2}, \cdots, v_{i4})$，$i$ =1，2，…，n（n 为设定的粒子数）。

定义适应度函数：max $f(x)$.

基上述分析，可以重新定义速度与随机数的数乘为依随机数对应概率值，保留速度中的所有交换子，实现了粒子群算法向离散空间的映射 [10]。其步骤如下：

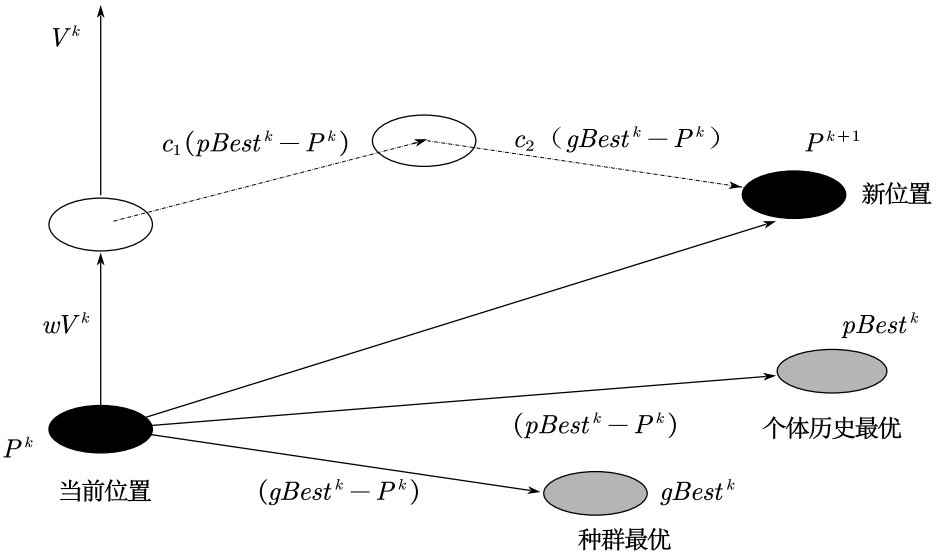

图 **2**　粒子的飞行过程示意图

（1）计算速度

即根据粒子的历史位置、最佳位置和全局最佳位置计算速度。

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_1 \left(p_{ij} - x_{ij}\right) + c_2 r_2 \left(g_j - x_{ij}\right) , \tag{17}$$

其中，c_1，c_2 分别为个体和社会学习因子，g_j 是全局最佳位置在第 j 维度的值。

研究表明，$w = 0.729$，$c_1 = c_2 = 1.494$ 时可以确保算法收敛 [8]。通过合理设置这些参数，可以显著提高粒子群优化算法的性能和效果。其中，w 用于控制粒子速度的惯性，常见的设置范围为 [0.4, 0.9]；c_1，c_2 分别控制对个体最佳位置和全局最佳位置的吸引力，常见设置为 1.5 到 2.0 之间。根据具体问题的规模和性质可进行调整，优化参数配置。

（2）应用 $sigmoid$ 函数

将速度值传递给 $sigmoid$ 函数, 以获得更新概率：

$$P_{ij} = \frac{1}{1 + e^{-v_{ij}}} , \tag{18}$$

保证其在［0，1］之间。

（3）决策更新

即根据求得的更新概率决定是否更新粒子的位置。

- 生成随机数: 为每个粒子生成一个随机数 r_{ij} 在 [0, 1] 区间内。
- 决定是否更新位置: 如果 r_{ij} 小于 P_{ij}，则更新粒子的位置; 否则保持当前位置不变。即，如果 $r_{ij} < P_{ij}$, 则 $x_{ij} = p_{ij}$.
- 检查停止条件：停止条件可以是达到最大迭代次数或适应度函数值不再显著改进。
- 输出最优解：返回全局最佳位置作为最优解。

5.2.5 模型求解

使用 python 进行求解，设购入零配件一、二的数量均为 t=100，设定粒子数 n=8，用离散粒子群算法对利润函数进行求解，得到的结果如表二所示：

表 **2**　两种零配件生产过程的决策结果

情况	零件一是否检测	零件二是否检测	成品是否检测	是否拆解	最大利润/元
1	否	是	否	是	2468
2	否	否	是	否	2052
3	是	否	是	否	2204
4	是	是	是	是	1672
5	是	否	是	否	2441
6	是	是	是	否	2098

5.2.6 灵敏度分析

对于输入不同的零配件数 t，可以进行单个成品利润的灵敏度分析。取 t =50，100，200，500，700，1000 的 6 种不同情况，分别算出对应的最大利润，而后求出每个成品的平均利润，得出平均利润的方差如表 3 所示。

表 **3**　平均利润的方差数据

	情况 1	情况 2	情况 3	情况 4	情况 5	情况 6
平均利润的方差	0	0.0125	0.1075	3.94e-31	1.58e-30	0.0233

发现平均利润的方差的值基本在 0.1 以下，可以认为此模型的鲁棒性较高。

5.3 多工序多零配件生产的最优决策问题

5.3.1 多工序多零件的模型分析

对于 m 道工序、n 个零配件的多工序多零件生产流程，未知零配件个数配对情况，同时未知每道工序的具体加工方式，题干仅给出零配件、半成品与成品的次品率作为已知条件，因此，假设此时生产方案决策只需考虑次品率，检测成本，拆解费用与调换成本等对此方案的决策无影响。

针对两道工序、8 个零配件的生产过程，以及各个流程的具体数据，我们依据问题二建立的模型，以设定利润函数作为决策的核心依据，并且减少拆解次数，对于每一生产过程的决策，采用问题二中的离散粒子群算法得出最优决策。

5.3.2 模型建立

根据问题二的模型，同理得到问题三的具体情形的利润函数优化模型与（5），（6）式同理。

设 X_i=0 或 1，i=1，2，$\cdots$，16（前 12 个变量表示检测决策，后 4 个表示拆解决策）。其中，当 X_i 的值为 0 时，表示不进行检测（或拆解）；当 X_i 的值为 1 时，表示进行检测（或拆解）。

此时，设 p_i，i=1，2，$\cdots$，12，分别为零配件 1∼8，半成品 1、2、3 与成品的检测成本。

由于零配件 1∼8 不一定都进行检测，所以可求出配对数:

$$\begin{cases} T_1 = \min\left\{(t - 10\%\cdot X_1\cdot t), (t - 10\%\cdot X_2\cdot t), (t - 10\%\cdot X_3\cdot t)\right\}, \\ T_2 = \min\left\{(t - 10\%\cdot X_4\cdot t), (t - 10\%\cdot X_5\cdot t), (t - 10\%\cdot X_6\cdot t)\right\}, \\ T_3 = \min\left\{(t - 10\%\cdot X_7\cdot t), (t - 10\%\cdot X_8\cdot t)\right\}, \\ T_4 = \min\left\{(T_1 - 10\%\cdot X_9\cdot T_1), (T_2 - 10\%\cdot X_{10}\cdot T_2), (T_3 - 10\%\cdot X_{11}\cdot T_3)\right\}. \end{cases} \tag{19}$$

因为在此问中，所有零配件、半成品与成品的次品率均为 10%，所以为简化式子，我们用 $q_0 = 10\%$ 来统一表示各个零配件、半成品与成品的次品率。

对于半成品 1，组装成其的零配件 1、零配件 2、零配件 3 均为合格品的期望为：

$$E_1 = (1 - q_0)^{1-X_1}\cdot(1 - q_0)^{1-X_2}\cdot(1 - q_0)^{1-X_3}, \tag{20}$$

得到的半成品 1 中的实际合格品数量为：

$$N' = E_1\cdot T_1\cdot(1 - q_0), \tag{21}$$

同理可得半成品 2 中的实际合格品数量为：

$$N'' = (1 - q_0)^{1-X_4}\cdot(1 - q_0)^{1-X_5}\cdot(1 - q_0)^{1-X_6}\cdot T_2\cdot(1 - q_0), \tag{22}$$

半成品 3 中的实际合格品数量为：

$$N''' = (1 - q_0)^{1-X_7}\cdot(1 - q_0)^{1-X_8}\cdot T_3\cdot(1 - q_0). \tag{23}$$

成品数量为：

$$T = T_4 - 10\%\cdot X_{12}\cdot T_4, \tag{24}$$

其中成品的实际合格数量为：

$$N_1 = \min\left\{N', N'', N'''\right\}\cdot(1 - q_0). \tag{25}$$

而检测出的不合格成品数量为：

$$t' = X_{12} \cdot (T - N_1), \tag{26}$$

则流入市场的成品数为：

$$N = T - t'. \tag{27}$$

从购入零配件到调换的成本为：

$$\begin{cases} Q_1 = 64t, \\ Q_2 = \sum_{i=1}^{8} X_i P_i t + X_9 p_9 T_1 + X_{10} p_{10} T_2 + X_{11} p_{11} T_3 + X_{12} p_{12} T_4, \\ Q_3 = 8\left(T_1 + T_2 + T_3 + T_4\right), \\ Q_4 = 10\% \cdot (40 + 64 + 29)\left(1 - X_{12}\right)(N - N_1). \end{cases} \tag{28}$$

此时的拆解成本:

$$W = -6 \times 10\%\left(T_1 + T_2 + T_3\right) - 10 \times 10\% T_4 + S(10\%\left(T_1 + T_2 + T_3 + T_4\right)) \tag{29}$$

简化此时的 $S = N \cdot h - Q$，即最多只拆解一次。

5.3.3 模型求解

对于 m 道工序、n 个零配件的生产过程，可以根据次品率的变化来判断在生产过程中是否采取进一步的行动，如继续生产、检测或返工。此方案的主要目标是保证产品质量稳定，同时减少不必要的操作，如过多的检测或返工。即决策方案为，基于次品率来判断下一步的行动。

对于题干要求的具体情形，对粒子进行初始化定义一个粒子的 16 维位置为 $x_i = (x_{i1}, x_{i2}, \cdots, x_{i16})$,，定义一个粒子的 16 维速度为 $v_i = (v_{i1}, v_{i2}, \cdots, v_{i16})$，$i$ =1，2，…，n（n 为设定的粒子数），其他步骤与问题二相同。

使用 python 对此问进行求解，设 $t = 100$，n=800，迭代 1000 次，得到的接近最优的决策方案如表 4 所示，此时的最大利润为 9923 元。

表 **4**　**8** 种零配件，**2** 道工序生产过程的决策结果

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}
0	1	1	1	1	0	0	1	1	1	1	0	0	1	0	1

5.4 基于不确定次品率的生产决策问题分析

5.4.1 模型建立

因为此时问题 2 和问题 3 中零配件、半成品和成品的次品率均是通过抽样检测方法得到的，所以此时的次品率为一个区间。因为样本量较大，所以可把零配件、半成品和成品的次品率均看成正态分布，其均值 μ 为题干原给的次品率P_0，标准差 σ 可以自行设置。对得到的次品率区间进行步长搜索，重复问题 2 与问题 3 的模型，得出此时的最优决策方案。

实际次品率 P 服从正态分布，则同（1）式。假设置信水平为 Z，抽样检测得到的次品率区间为 $[-Z\cdot\sigma + P_0, Z\cdot\sigma + P_0]$。

5.4.2 模型求解

在 95% 置信水平下，可查找到对应的双尾临界值 $Z \approx 1.96$，均值为题目给出的次品率 p，假设其标准差 $\sigma = 0.02$，则置信区间为 $[-0.0392 + p, 0.0392 + p]$，假设每个零配件和成品的次品率都是线性相关的，以步长 $h = 0.005$ 从置信区间左端点开始对区间进行遍历。

对于问题 2 的各个情形，用 python 进行求解，遍历得到的利润结果如图 3 所示；对于问题 3 的情形，每个遍历结果得到的利润结果如图 4 所示。

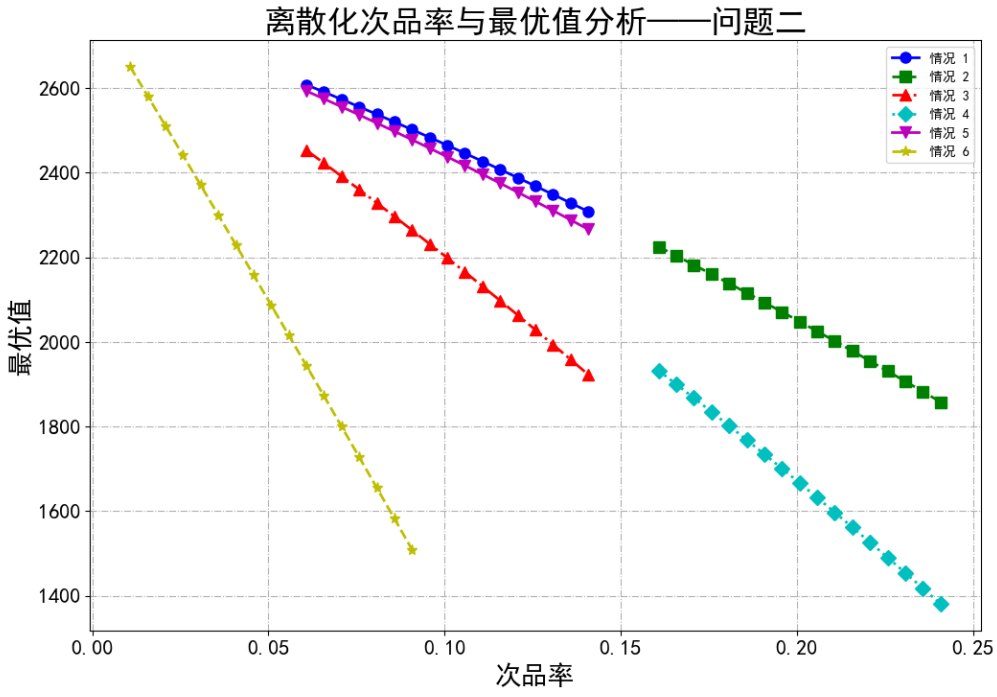

图 **3**　问题二的次品率区间分析示意图

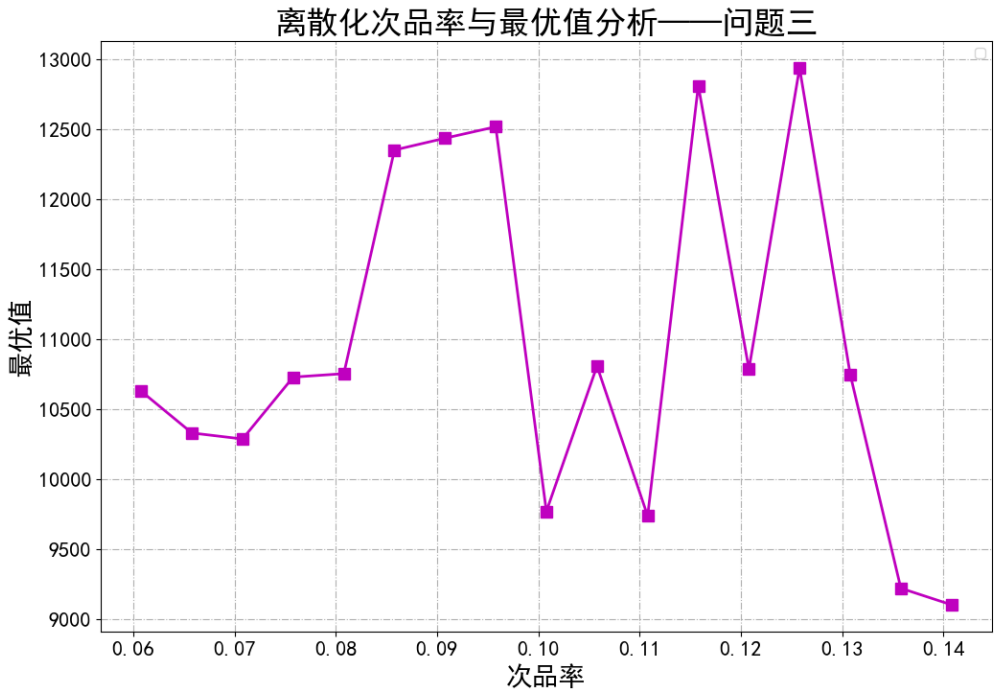

图 **4**　问题三的次品率区间分析示意图

对于问题 2 与问题 3 中以步长 $h = 0.005$ 搜索得到的次品率区间下的最优决策与最大利润全部数据，分别放入支撑材料的表格 1.xlsx 与表格 2.xlsx 中。问题 3 中部分数据如表 5 所示。

表 **5**　问题 **3** 中次品率区间下的最优决策与最大利润的部分数据

次品率	X_i，i=1，2，…，16 的最优决策方案	最优值
0.0908	[0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1]	12438.68518
0.0958	[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]	12520.09998
0.1008	[0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1]	9771.605783
0.1058	[1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 1]	10810.50003
0.1108	[0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1]	9739.241245

六、 模型的推广，应用与改进

6.1 模型的推广与应用

在生产决策问题中，往往需要同时考虑多个变量（如零配件的次品率、检测成本、拆解费用、生产工序等）以达到整体最优。本文包括了基于正态分布进行抽样检测以达到是否拒收（接收）零件的模型，从理想化流程（次品率确定，工序简单）到多工序多零配件再到不确定次品率的生产决策模型、离散粒子群算法求解模型。

此模型可以很容易地扩展到更复杂的生产流程中，全面考虑生产过程中的各种因素，能够适用于不同规模和复杂度的生产系统，为企业提供更加全面的决策支持。无论是零配件的次品率、检测成本、购买单价，还是市场售价和调换损失等，都可以通过调整算法的输入参数来适应不同的生产环境和市场需求。这使得模型在多种生产场景下都能有效运行，提供合理的决策支持。

随着市场环境的变化和生产条件的改善，生产过程中的各项参数可能会发生变化。粒子群算法具有良好的动态调整能力，能够根据新的参数和约束条件重新进行优化计算，为企业提供新的决策方案。这使得模型能够实时响应市场和生产环境的变化，保持决策的有效性和及时性。

6.2 模型的改进

1. 为了简化问题，我们只考虑拆解不合格品一次的情况，但是题干表示拆解过程不会对零配件造成损坏，所以在拆解损失较小的情况下，多次拆解的情况有可能盈利，可以考虑对拆解次数的情况进行完善。
2. 对于问题二、三的具体情况求解，我们采用离散粒子群算法进行求解，还可使用其他启发式算法如：模拟退火算法、遗传算法等，对此题的最优决策方案进行对比。
3. 对于问题三的 m 道工序、n 个零配件的多工序多零件生产流程，我们忽略除次品率以外（如检测成本、拆解成本）的影响，仅建立了一个大致的生产决策方案，并未定量计算。可以尝试使用弹性网 (Elastic Net) 方法构建多工序制造过程质量关系模型，并基于该模型进行关键质量特性识别 [11]，建立次品率与多工序多零件制造的具体模型。
4. 对于问题四中的次品率区间，我们选择设定步长遍历区间内的情况，而实际区间为连续区间。且次品率在置信区间中应为随机变化的，而我们假设各零配件的次品率线性相关。

参考文献

[1] 王发鸿, 达庆利. 电子行业再制造逆向物流模式选择决策分析 [J]. 中国管理科学,2006,(06):44-49.DOI:10.16381/j.cnki.issn1003-207x.2006.06.009.

[2] 贾俊平. 统计学 [M]. 北京：清华大学出版社, 2004.

[3] 梁之舜，邓集贤, 等. 概率论与数理统计 [M].2 版北京: 高等教育出版社，1988.

[4] 智冬晓, 许晓娟, 张皓博.z 检验与 t 检验方法的比较 [J]. 统计与决策,2014,(20):31-34.DOI:10.13546/j.cnki.tjyjc.2014.20.007.

[5] 胡良平. 如何正确运用 Z 检验——Z 检验的基本概念与前提条件 [J]. 四川精神卫生,2020,33(05):418-421.

[6] 张志辉. 二项分布一致性检验问题研究 [J]. 系统仿真技术,2021,17(04):241-243.DOI:10.16812/j.cnki.cn31-1945.2021.04.005.

[7] 魏艳华, 王丙参, 邢永忠. 基于蒙特卡洛方法的假设检验问题探讨 [J]. 统计与决策,2018,34(24):75-78.DOI:10.13546/j.cnki.tjyjc.2018.24.017.

[8] 沈林成, 霍霄华, 牛轶峰. 离散粒子群优化算法研究现状综述 [J]. 系统工程与电子技术,2008,(10):1986-1990+1994.

[9] Kennedy J. The particle swarm: social adaptation of knowledge[C]. In: IEEEInternational Conference on Evolutionary Computation, Indianapolis, IN, USA,1997:303-308.

[10] Clerc M • Discrete Particle Sw arm Optimization, Illustrated by the T raveling Salesman Problem [oL] .http://www • mau-ricecierc • net./ ,2000.

[11] 王宁, 闫娜, 徐友真, 等. 复杂多工序制造过程关键质量特性识别 [J]. 统计与决策,2021,37(08):177-180.DOI:10.13546/j.cnki.tjyjc.2021.08.039.

附录 A　支撑材料列表

序号	文件名	材料说明
1	problem1.m	基于正态近似检验的问题一全部代码
2	问题二.py	基于离散粒子群算法的问题二全部代码
3	问题三.py	问题三的全部代码
4	问题四 _2.py	问题四中重复问题二进行次品率误差分析的代码
5	问题四 _3.py	问题四中重复问题三进行次品率误差分析的代码
6	灵敏度分析.py	对问题二中不同 t 取值进行灵敏度分析的代码
7	表格 1.xlsx	问题四中重复问题二的次品率区间下的利润与最优解数据
8	表格 2.xlsx	问题四中重复问题三的次品率区间下的利润与最优解数据
9	灵敏度表格.xlsx	灵敏度分析得到的平均利润方差数据

附录 B　蒙特卡洛模拟的 matlab 主要代码

```
% 定义参数
P0 = 0.1;
n_simulations = 10000; % 蒙特卡洛模拟的次数
sample_size_95 = 900; % 根据95%信度计算出的样本量
sample_size_90 = 225; % 根据90%信度计算出的样本量
true_defect_rate = 0.11; % 真实的次品率（假设比标称次品率稍高）

% 模拟抽样过程
results_95 = binornd(sample_size_95, true_defect_rate, [n_simulations, 1]) /
    sample_size_95;
results_90 = binornd(sample_size_90, true_defect_rate, [n_simulations, 1]) /
    sample_size_90;

% 计算在不同信度下的次品率分布
mean_rate_95 = mean(results_95);
mean_rate_90 = mean(results_90);

% 使用95%和90%信度水平进行假设检验
alpha_95 = 0.05;
alpha_90 = 0.10;
```

```
    critical_value_95 = prctile(results_95, 100 * (1 - alpha_95));
    critical_value_90 = prctile(results_90, 100 * (1 - alpha_90));

    if critical_value_95 > P0
    decision_95 = 'Reject';
    else
    decision_95 = 'Accept';
    end

    if critical_value_90 > P0
    decision_90 = 'Reject';
    else
    decision_90 = 'Accept';
    end

    fprintf('At 95%% confidence level: Decision: %s, Mean defect rate: %.4f\n',
        decision_95, mean_rate_95);
    fprintf('At 90%% confidence level: Decision: %s, Mean defect rate: %.4f\n',
        decision_90, mean_rate_90);
```

附录 C　基于离散粒子群算法的第二问 python 主要代码

```
import itertools
import numpy as np


# 数据列表
data =[
[0.1, 4, 2, 0.1, 18, 3, 0.1, 6, 3, 56, 6, 5], # 情况 1
[0.2, 4, 2, 0.2, 18, 3, 0.2, 6, 3, 56, 6, 5], # 情况 2
[0.1, 4, 2, 0.1, 18, 3, 0.1, 6, 3, 56, 30, 5], # 情况 3
[0.2, 4, 1, 0.2, 18, 1, 0.2, 6, 2, 56, 30, 5], # 情况 4
[0.1, 4, 8, 0.2, 18, 1, 0.1, 6, 2, 56, 10, 5], # 情况 5
[0.05, 4, 2, 0.05, 18, 3, 0.05, 6, 3, 56, 10, 40] # 情况 6
]

# Sigmoid函数
def sigmoid(x):
return 1 /(1 +np.exp(-x))

# 定义适应度函数
def fitness_function(position, params):
# 解包每一行的数据作为函数的参数
概率1, 配件1单价, 成本1, 概率2, 配件2单价, 成本2, 概率3, 装配花费, 成本3, h, 调换损失, 拆卸成本 =params
x1, x2, x3, x4 =position

# 重置初始值
```

```
t =100
T =int(min((t -x1 *t *概率1), (t -x2 *t *概率2)))
t_ =int(x3 *T *概率3) #次品
# print(t_)
N =T -t_ #成品数
Q1 =22*t
Q2 =x1 *成本1*t +x2 *成本2*t +x3 *T *成本3
Q3 =6*T
Q4 =(1 -x3 ) *( 22 +6 +成本1+成本2+成本3+ 调换损失)* T*概率3

S =N *h -(Q1 +Q2 +Q3 +Q4)
# print(S)
# print('_____')

t= T *概率3
T =min((t -x1 *t *概率1), (t -x2 *t *概率2))
t_ =int(x3 *T *概率3)
N =T -t_ #成品数
Q1 =22*t
Q2 =x1 *成本1*t +x2 *成本2*t +x3 *T *成本3
Q3 =6*T
Q4 =(1 -x3 )* ( 22 +6 +成本1+成本2+成本3+ 调换损失)* T*概率3
SS =N *h -(Q1 +Q2 +Q3 +Q4)
W =-1 *T *成本3 *拆卸成本 +SS

S =S +W

return S;

# PSO算法
for idx, params in enumerate(data):
print(f"情况 {idx+1}")

# PSO参数
num_particles =20 # 粒子数量
num_dimensions =4 # 每个粒子的维度（每个粒子由 0 和 1 组成）
max_iterations =20 # 最大迭代次数
inertia =0.5 # 惯性权重
cognitive_constant =1.5 # 个体学习因子
social_constant =1.5 # 社会学习因子

# 初始化粒子的位置（0 或 1）和速度（可以初始化为较小的随机值）
position =np.random.randint(0, 2, (num_particles, num_dimensions))
velocity =np.random.uniform(low=-1, high=1, size=(num_particles, num_dimensions))

# 初始化粒子历史最佳位置和全局最佳位置
personal_best_position =np.copy(position)
personal_best_value =np.array([fitness_function(position[i], params) for i in range(num_particles)])
global_best_position =position[np.argmax(personal_best_value)]
global_best_value =np.max(personal_best_value)

# PSO主循环
for iteration in range(max_iterations): # 迭代
for i in range(num_particles):
# 更新速度
r1, r2 =np.random.rand(num_dimensions), np.random.rand(num_dimensions)
```

```
cognitive_velocity =cognitive_constant *r1 *(personal_best_position[i] -position[i])
social_velocity =social_constant *r2 *(global_best_position -position[i])
velocity[i] =inertia *velocity[i] +cognitive_velocity +social_velocity

# 通过Sigmoid函数获取更新概率
velocity_prob =sigmoid(velocity[i])

# 更新位置：根据概率来决定每个维度的0或1是否翻转
for d in range(num_dimensions):
if np.random.rand() <velocity_prob[d]:
position[i, d] =1 -position[i, d] # 翻转0或1

# 计算新的适应度值
fitness_value =fitness_function(position[i], params)

# 更新个人最佳位置
if fitness_value >personal_best_value[i]:
personal_best_position[i] =np.copy(position[i])
personal_best_value[i] =fitness_value

# 更新全局最佳位置
if np.max(personal_best_value) >global_best_value:
global_best_value =np.max(personal_best_value)
global_best_position =np.copy(personal_best_position[np.argmax(personal_best_value)])
```

附录 D　基于离散粒子群算法的第三问 python 主要代码

```
import numpy as np


# 数据列表
概率 =0.1
单价1,单价2,单价3,单价4,单价5,单价6,单价7,单价8 =2,8,12,2,8,12,8,12
零件成本1,零件成本2,零件成本3,零件成本4,零件成本5,零件成本6,零件成本7,零件成本8 =1,1,2,1,1,2,1,2

半成品装配成本 =8
半成品检测成本 =4
半成品拆解成本 =6

成品装配成本 =8
成品检测成本 =6
成品拆解成本 =10
售价 =200
调换 =40


# Sigmoid函数
def sigmoid(x):
```

```
    return 1 /(1 +np.exp(-x))

    # 定义适应度函数
    def fitness_function(position):
    # 解包每一行的数据作为函数的参数
    t=100
    x1, x2, x3, x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16 =position
    # 零配件1, 2, 3配对数
    T1 =min((t -x1 *t *0.1), (t -x2 *t *0.1), (t -x3 *t *0.1))
    T2 =min((t -x4 *t *0.1), (t -x5 *t *0.1), (t -x6 *t *0.1))
    T3 =min((t -x7 *t *0.1), (t -x8 *t *0.1))
    T4 =min((T1 -x9 *T1 *0.1), (T2 -x10 *T2 *0.10 ), (T3 -x11 *T3 *0.10 ))
    # 成品数
    N =T4 -x12 *T4 *0.10
    Q1 =64*t
    Q2 =t*(x1+x2+x3*2+x4+x5+x6*2+x7+x8*2)+ x9*半成品检测成本*T1 +x10*半成品检测成本*T2 +x11*半成品检测成本*T3 +
                                                    x12*成品检测成本 *T4
    Q3 =(T1 +T2 +T3 +T4) *8
    Q4 =(40 +64 +29)*(1 -x12)*T4 *0.10
    S =N * 售价 -(Q1+Q2+Q3+Q4)

    # 下面是拆解

    t =(T1 +T2 +T3 +T4) *0.10
    # 零配件1, 2, 3配对数
    T1 =min((t -x1 *t *0.1), (t -x2 *t *0.1), (t -x3 *t *0.1))
    T2 =min((t -x4 *t *0.1), (t -x5 *t *0.1), (t -x6 *t *0.1))
    T3 =min((t -x7 *t *0.1), (t -x8 *t *0.1))
    T4 =min((T1 -x9 *T1 *0.1), (T2 -x10 *T2 *0.10), (T3 -x11 *T3 *0.10))
    # 成品数
    N =T4 -x12 *T4 *0.10
    Q1 =64 *t
    Q2 =t *(x1 +x2 +x3 *2 +x4 +x5 +x6 *2 +x7 +x8 *2) +x9 *半成品检测成本 *T1 +x10 *半成品检测成本 *T2 +x11 *
                                                    半成品检测成本 *T3 +x12 *成品检测成本 *T4
    Q3 =(T1 +T2 +T3 +T4) *8
    Q4 =(40 +64 +29)*(1 -x12) *T4 *0.10

    SS =N * 售价 -(Q1 +Q2 +Q3 +Q4)
    W =-(T1 +T2 +T3) *6 *0.1 -T4 *10 *0.1 +SS

    S =S +W
    return S



    # PSO算法

    # PSO参数
    num_particles =800 # 粒子数量
    num_dimensions =16 # 每个粒子的维度（每个粒子由 0 和 1 组成）
    max_iterations =1000 # 最大迭代次数
    inertia =0.5 # 惯性权重
    cognitive_constant =1.5 # 个体学习因子
    social_constant =1.5 # 社会学习因子

    # 初始化粒子的位置（0 或 1）和速度（可以初始化为较小的随机值）
```

```
position =np.random.randint(0, 2, (num_particles, num_dimensions))
velocity =np.random.uniform(low=-1, high=1, size=(num_particles, num_dimensions))

# 初始化粒子历史最佳位置和全局最佳位置
personal_best_position =np.copy(position)
personal_best_value =np.array([fitness_function(position[i]) for i in range(num_particles)])
global_best_position =position[np.argmax(personal_best_value)]
global_best_value =np.max(personal_best_value)

# PSO主循环
for iteration in range(max_iterations): # 迭代
for i in range(num_particles):
# 更新速度
r1, r2 =np.random.rand(num_dimensions), np.random.rand(num_dimensions)

cognitive_velocity =cognitive_constant *r1 *(personal_best_position[i] -position[i])
social_velocity =social_constant *r2 *(global_best_position -position[i])
velocity[i] =inertia *velocity[i] +cognitive_velocity +social_velocity

# 通过Sigmoid函数获取更新概率
velocity_prob =sigmoid(velocity[i])

# 更新位置：根据概率来决定每个维度的0或1是否翻转
for d in range(num_dimensions):
if np.random.rand() <velocity_prob[d]:
position[i, d] =1 -position[i, d] # 翻转0或1

# 计算新的适应度值
fitness_value =fitness_function(position[i])

# 更新个人最佳位置
if fitness_value >personal_best_value[i]:
personal_best_position[i] =np.copy(position[i])
personal_best_value[i] =fitness_value

# 更新全局最佳位置
if np.max(personal_best_value) >global_best_value:
global_best_value =np.max(personal_best_value)
global_best_position =np.copy(personal_best_position[np.argmax(personal_best_value)])
```

附录 E　问题四中对问题二进行次品率分析的 python 主要代码

```
# 参数定义
step_size =0.005 # 可根据需要添加更多步长

# 存储每种情况的结果
results =[[]for _ in range(len(data))]
errors =[[]for _ in range(len(data))]
```

```
# 遍历每种情况
for idx, params in enumerate(data):
# 更新 p 为每种情况的概率1
p =params[0]
lower_bound =p -0.0392
upper_bound =p +0.0392
points =np.arange(lower_bound, upper_bound +step_size, step_size)

for point in points:
def fitness_function(position, params):
# 解包每一行的数据作为函数的参数
概率1, 配件1单价, 成本1, 概率2, 配件2单价, 成本2, 概率3, 装配花费, 成本3, h, 调换损失, 拆卸成本 =params
x1, x2, x3, x4 =position
概率1 =point
概率2 =point
概率3 =point

# 重置初始值
t =100
T =int(min((t -x1 *t *概率1), (t -x2 *t *概率2)))
t_ =int(x3 *T *概率3) # 次品
N =T -t_ # 成品数
Q1 =22 *t
Q2 =x1 *成本1 *t +x2 *成本2 *t +x3 *T *成本3
Q3 =6 *T
Q4 =(1 -x3) *(22 +6 +成本1 +成本2 +成本3 +调换损失) *T *概率3

S =N *h -(Q1 +Q2 +Q3 +Q4)

t =T *概率3
T =min((t -x1 *t *概率1), (t -x2 *t *概率2))
t_ =int(x3 *T *概率3)
N =T -t_ # 成品数
Q1 =22 *t
Q2 =x1 *成本1 *t +x2 *成本2 *t +x3 *T *成本3
Q3 =6 *T
Q4 =(1 -x3) *(22 +6 +成本1 +成本2 +成本3 +调换损失) *T *概率3
SS =N *h -(Q1 +Q2 +Q3 +Q4)
W =-1 *T *成本3 *拆卸成本 +SS

S =S +W

return S
```

附录 F　问题四中对问题三进行次品率分析的 python 主要代码

```
# 参数定义
p =概率
lower_bound =p -0.0392
upper_bound =p +0.0392
```

```
# 遍历不同步长
step_size =0.005
points =np.arange(lower_bound, upper_bound +step_size, step_size)

# Sigmoid函数
def sigmoid(x):
return 1 /(1 +np.exp(-x))

# 定义适应度函数
def fitness_function(position, 概率):
t =100
x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16 =position

# 计算初始零配件数
T1 =min(t -x1 *t *概率, t -x2 *t *概率, t -x3 *t *概率)
T2 =min(t -x4 *t *概率, t -x5 *t *概率, t -x6 *t *概率)
T3 =min(t -x7 *t *概率, t -x8 *t *概率)
T4 =min(T1 -x9 *T1 *概率, T2 -x10 *T2 *概率, T3 -x11 *T3 *概率)
N =T4 -x12 *T4 *概率

Q1 =64 *t
Q2 =t *(x1 +x2 +x3 *2 +x4 +x5 +x6 *2 +x7 +x8 *2) +\
x9 *半成品检测成本 *T1 +x10 *半成品检测成本 *T2 +x11 *半成品检测成本 *T3 +x12 *成品检测成本 *T4
Q3 =(T1 +T2 +T3 +T4) *8
Q4 =(40 +64 +29) *(1 -x12) *T4 *概率
S =N *售价 -(Q1 +Q2 +Q3 +Q4)

# 拆解过程
t =(T1 +T2 +T3 +T4) *概率
T1 =min(t -x1 *t *概率, t -x2 *t *概率, t -x3 *t *概率)
T2 =min(t -x4 *t *概率, t -x5 *t *概率, t -x6 *t *概率)
T3 =min(t -x7 *t *概率, t -x8 *t *概率)
T4 =min(T1 -x9 *T1 *概率, T2 -x10 *T2 *概率, T3 -x11 *T3 *概率)
N =T4 -x12 *T4 *概率

Q1 =64 *t
Q2 =t *(x1 +x2 +x3 *2 +x4 +x5 +x6 *2 +x7 +x8 *2) +\
x9 *半成品检测成本 *T1 +x10 *半成品检测成本 *T2 +x11 *半成品检测成本 *T3 +x12 *成品检测成本 *T4
Q3 =(T1 +T2 +T3 +T4) *8
Q4 =(40 +64 +29) *(1 -x12) *T4 *概率

SS =N *售价 -(Q1 +Q2 +Q3 +Q4)
W =-(T1 +T2 +T3) *6 *概率 -T4 *10 *概率 +SS

return S +W
```