基于多阶段整数规划和鲁棒优化的生产过程决策方案

摘要

在电子产品生产过程中，由于零配件、半成品与成品均可能存在一定比例的次品，需要对其进行必要的检测，实施有效的质量控制，以确保成品满足市场销售要求。为了减少生产过程成本，提高企业收益，本文通过设计抽样检测方案、建立 0-1 整数规划模型进行生产过程决策优化，并基于贝叶斯后验进行多阶段鲁棒优化，结合遗传算法制定优化的生产决策方案，分析实际次品率对生产决策的影响。

针对问题一，我们设计了零配件抽样检测方案，根据检测结果及标称次品率决定是否在一定的置信水平下接收该批零件。我们认为零配件的次品率出现符合二项分布，基于中心极限定理，其可近似为正态分布，从而推导出在一定置信水平和容忍误差下对于一批零配件的最小抽样检测样本量。

针对问题二，我们通过建立 **0-1 整数规划模型**，对生产过程决策进行优化。以多个 0-1 变量作为决策变量，表示生产过程不同阶段的决策，推导出该过程中的购买、检测、拆解、调换成本以及企业售卖收入，并引入**零件利用提升率**参数用以描述成品拆解后带来的收益。以最大化单批零件条件下企业利润为目标函数，在次品率和价格不同的多种情况下，为企业制定最优生产过程决策。

针对问题三，在工序数量和生产规模进一步扩大的情况下，我们建立**多阶段 0-1 整数规划模型**，在问题二的基础上推导出普适情况下的生产过程中成本消耗及产品收益，从而求得单批零件条件下企业利润。在问题规模扩大情况下，为确保计算效率，我们采用启发式**遗传算法**对最优解进行全局搜索，从而得到在不同情况下的最优生产过程决策方案。

针对问题四，当题目中多给次品率均为抽样检测所得结果时，结合问题一所设计的抽样检测方案，我们采用**贝叶斯后验分布**估计实际次品率均值。基于一定的先验信息，我们认为次品率先验分布满足函数 $Beta(2,5)$，并根据检测所得次品率计算后验分布。由于真实次品率为估计值，为不确定性参数，我们建立**多阶段鲁棒优化**，根据置信区间设定**盒式不确定性集**，根据遗传算法在最劣情况下重新求解问题二、三的最优生产过程决策方案。

最后，我们对本文所提出的模型进行敏感性分析，通过验证可知本文所提出的数学模型具有良好的普适性，针对不同参数和情况均能求解出最优生产决策方案，且求解所得方案的鲁棒性良好。

关键字：抽样检测　0-1 整数规划　多阶段鲁棒优化　贝叶斯估计　遗传算法

目录

一、 问题重述

1.1 问题背景

当今世界市场的竞争本质上是科学技术的竞争、质量的竞争。产品质量反映的是产品能够满足消费者使用要求的程度[1]。某企业某种电子产品由多件零件经过多道工序制作完成，成品进入市场售卖。其生产过程即为电子产品展现电子产品质量的重要阶段[2]。由于零件及成品中可能存在一定比例的次品，为了确保产品的质量，需要对零部件进行必要的检测，以保证装配后的成品达到市场销售要求。现需要根据企业在生产中遇到的实际情况，设计零配件的抽样检测方案，并对零配件装配过程中的检测、拆解、调换等操作进行决策，实施有效的质量控制，减少生产过程成本，提高企业收益。进一步的研究需要针对生产过程中工序和零部件数量增加的情况，制定电子产品生产过程的决策方案，同时讨论抽样检测下实际次品率对决策方式的影响。

1.2 问题提出

基于问题背景和题设限制条件，本文需要解决以下问题：

问题一

企业需要通过抽样检测决定是否接收供应商提供的零配件（标称次品率不超过 10%），要求设计抽样检测方案，在抽样检测次数尽量小的情况下，对于 95% 和 90% 置信水平，分别判断零配件次品率是否超过标称值，以决定是否接收该批零配件。

问题二

已知企业在生产过程中两种零配件和成品的次品率，针对不同情况讨论是否进行零配件的检测、成品的检测以及不合格产品的拆解等决策，以减小生产过程成本，并提高企业收益。

问题三

进一步扩大生产规模，讨论在工序增加至 m 道、零配件数量增加至 n 个，且零配件、半成品和成品次品率已知情况下的决策方案，并针对具体情况进行计算，使企业利益最大化。

问题四

当问题二、三中所给出的次品率均为抽样检测所得时，结合问题一所设计的抽样检测方法，考虑实际次品率，为问题二、三重新设计决策方案。

二、 模型假设

为了构建出更加精确合理的数学模型，本文根据实际情况及模型需要作出以下的合理假设和条件约束，并给出假设理由：

1. **零配件是否为次品满足二项分布**。在抽样检测的过程中，为便于分析，认为抽取零配件的样本量足够大，根据中心极限定理，此时二项分布可近似为正态分布进行计算。

2. **初始各零配件数量相同**。由于在生产过程中，均为一套零配件装配为一件成品，故设初始各零配件数量与加工过程所用比例同为 1：1，便于后续决策依据判断。

3. **多余未装配零配件与半成品在后续不再考虑装配**。由于各零配件与半成品次品率不同，故当检测后，在装配阶段可能会存在多余零配件和半成品。该部分多余的配件由于缺少配件，无法进入下一道工序，故在后续不再考虑装配。

4. **当拆解后的零配件或半成品返回上一道工序时，其检测决策与原来相反**。当不合格品拆解后，返回上一工序检测。若此前已经历检测，说明零配件或半成品为正品，则不用重复检测；若此前未经历检测，则零配件或半成品可能为次品，此时需要检测。

5. **抽样检测成本不计入单套零配件生产过程成本**。由于抽样检测样本量相对于一整批零配件、半成品及成品总量较小，故而分摊到单套零配件时成本较小。故抽样检测成本仅在计算整批零配件生产过程时统一计入成本。

三、 符号说明

本文章使用模型所用到的主要变量符号及其含义如表 1所示：

表 1: **主要变量符号及含义**

符号	含义
p	零配件是次品的概率
$\hat{p}$	抽样样本的实际次品率
p_{lower}	实际次品率不确定集下界
p_{upper}	实际次品率不确定集上界
n	抽样检测样本量
p_0	零配件次品率的标称值
α	显著性水平
z_α	显著性水平为 α 时正态分布对应的上分位数
E	容忍误差
x_j	0-1 变量，是否检测零件
x_{prod}	0-1 变量，是否检测成品

表 1: **主要变量符号及含义（续表）**

符号	含义
$x_{disassemble}$	0-1 变量，是否进行拆解
n_j	购买零配件 j 的初始数量
n_{prod}	零配件装配得到的成品数量
n'_{prod}	经过拆解决策后成品数量修正值
$n_{disassemble}$	不合格成品拆解数量
c_j	零配件 j 的次品率
c'_{prod}	成品实际次品率
c_{prod}	当零配件均为正品时成品的次品率
p_j	零配件 j 的购买单价
p_{prod}	单件成品市场售价
d_j	单件零配件 j 的检测成本
d_{prod}	单件成品的检测成本
$p_{assemble}$	单件成品的装配成本
p_{change}	成品的调换损失
$p_{disassemble}$	成品的拆解费用
C_{init}	初始购入总零配件成本
$C_{assemble}$	总装配成本
C_{detect}	总检测成本
$C_{disassemble}$	总拆解成本
C_{change}	总调换损失
$C_{exdetect}$	拆解操作后新增检测成本
C	总生产成本
I	总收入
P	总利润
$\Delta\epsilon_j$	拆解操作后零配件 j 利用提升率
$\Delta\epsilon_{prod}$	拆解操作后成品数量提升率
N_0	初始购买零配件的数量向量
P_0	初始购买零配件的价格向量
S_i	第 i 道工序后零件状态
N_i	状态 S_i 下各零配件/半成品/成品的数量列向量

表 1: **主要变量符号及含义（续表）**

符号	含义
N_i'	状态 S_i 下拆解修正后各零配件/半成品/成品的数量列向量
N_{di}	状态 S_i 下拆解产品数量
$\Delta\epsilon_i$	状态 S_i 下拆解后的产品利用提升率
C_{ti}	状态 S_i 下各零配件/半成品/成品的检测费用
C_{di}	状态 S_i 下各零配件/半成品/成品的拆解费用
C_{ai}	状态 S_i 下各半成品/成品的装配费用
E_i	状态 S_i 下各零配件，半成品/成品（组件均为正品时）的次品率列向量
E_i'	状态 S_i 下半成品/成品的实际次品率列向量
X_i	状态 S_i 下是否检测各零配件/半成品/成品决策列向量
Y_i	状态 S_i 下是否拆解各半成品/成品决策列向量
C_i	状态 S_i 下总决策成本

四、模型的建立与求解

4.1 问题一的分析与求解

4.1.1 问题分析

在给定标称值为 10% 的情况下，现需要设计抽样检测方案，在尽可能降低企业检测成本的情况下，判断零配件次品率是否超过标称值，从而决定是否接收该批零配件。假设在零配件次品率一定的情况下，抽样检测得到的样品中的次品数量分布满足二项分布。当抽样样品量较大时，由中心极限定理，样品中次品数量可近似为正态分布。根据以下两种情形分别确定原假设和备择假设，通过假设检验判断是否接收这批零配件。

情形（1） 信度为 95% 情况下判定零配件次品率大于标称值，拒绝接收此批零配件。

情形（2） 信度为 90% 的情况下判定零配件次品率小于标称值，接收此批零配件。

4.1.2 模型的建立与求解

在抽样检测过程中，零配件次品出现次数满足二项分布，抽中零配件是次品的概率为 p，为合格品的概率为 $1-p$。中心极限定理认为, 无论研究的统计总体服从什么样的分布, 样本平均值的分布均接近一个正态分布, 正态分布的均值等于总体分布的均值, 标准偏差等于总体分布的标准偏差除以样本大小的平方根 [3]。设样品数量为 n，根据二项分布的特点，零配件中次

品数量的期望值为 np，方差 σ^2 为 $np(1-p)$ [4]。设样本中次品数量为 x，则样本的次品率测量值为 $\hat{p}$ 为 $\frac{x}{n}$（x 为样本中的次品数量），样本次品率的期望值为 p，方差为 $\frac{p(1-p)}{n}$。在样本数量足够大的情况下，样品中次品率的分布可近似为正态分布，其检验统计量 $Z=\frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$，如式 1所示：

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

情形（1） 根据情形（1），建立原假设 $H_0:p>10\%$，备择假设 $H_1:p\leq 10\%$，显著水平 α 为 5%，当原假设被拒绝时说明实际次品率小于 10%。由原假设可知（p_0 为标称值，即 $p_0=10\%$ ）：

$$Z=\frac{\hat{p}-p_0}{\sqrt{p_0(1-p_0)/n}}\geq\frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$$

根据公式 1，由于 $\frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$ 满足正态分布，有 $P(\frac{\hat{p}-p}{\sqrt{p(1-p)/n}}<-z_\alpha)=\alpha$ 成立。根据以上信息，可以推得式 2：

$$P(Z<-z_\alpha)\leq P(\frac{\hat{p}-p}{\sqrt{p(1-p)/n}}<-z_\alpha)=\alpha \tag{2}$$

设可接受的次品率误差为 E：

$$E=\hat{p}-p_0 \tag{3}$$

代入 $p_0=0.1$，计算可得拒绝域：$\hat{p}<p_0-\frac{0.3z_\alpha}{\sqrt{n}}$，则有

$$n>(\frac{-0.3z_\alpha}{E})^2 \tag{4}$$

查表可知，当 α 取 0.05 时，z_α 为 1.645。当 E 取不同的容忍误差时，代入 $(\frac{-0.3z_\alpha}{E})^2$ 可以得到 n 的最小取值，也即在给定置信水平和容忍误差范围内，抽样检测样本数量的最小值。

情形（2） 根据情形（2），建立原假设 $H_0:p<10\%$，备择假设 $H_1:p\geq 10\%$，显著水平 α 为 10%。当原假设被拒绝时说明实际次品率大于 10%，由原假设可知：

$$Z=\frac{\hat{p}-p_0}{\sqrt{p_0(1-p_0)/n}}\leq\frac{\hat{p}-p}{\sqrt{p(1-p)/n}}$$

近似视为正态分布，故可以推得式 5：

$$P(Z>z_\alpha)\leq P(\frac{\hat{p}-p}{\sqrt{p(1-p)/n}}>z_\alpha)=\alpha \tag{5}$$

代入 $p_0=0.1$，计算可得拒绝域 $\hat{p}>p_0+\frac{0.3z_\alpha}{\sqrt{n}}$。查表可知，当 α 取 0.1 时，z_α 为 1.28。

同情形（1），当样本次品率可接受误差为 E 时，样本数量满足

$$n > (\frac{0.3z_{\alpha}}{E})^2 \tag{6}$$

据此可以得到 10% 信度下，单边备择假设下不同允许误差取值对应样本数量 n 的最小取值。

4.1.3 模型结果

当 $|E|$ 取 0.01、0.015、0.02、0.025、0.03、0.035、0.04、0.045、0.05 值时，分别计算情形（1）、情形（2）下样本数量最小取值，结果如图 1所示（不同允许误差 E 对应最小样本容量具体值表格见附录）：

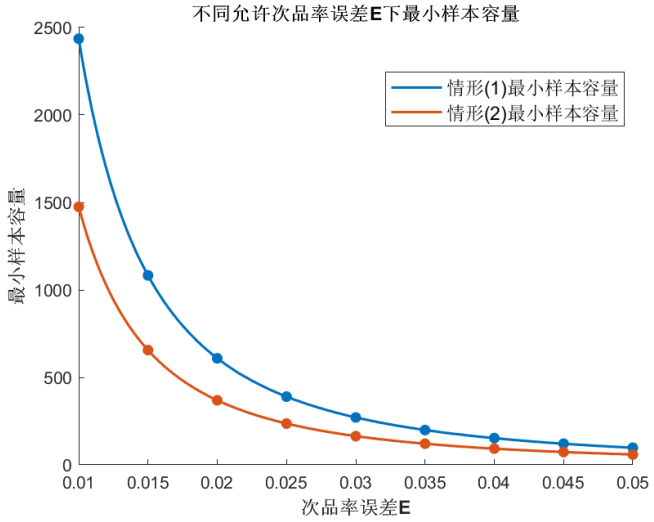

图 1: 不同允许次品率误差 E 下情形（1）和情形（2）的最小样本容量

由图 1可知，允许次品率误差取值 E 越小，即次品率检测越精确，最小抽样检测样本容量越大；对于标称值判断的置信区间，置信区间越大，最小抽样检测样本容量越小。

常见容忍误差 E 取值有 0.05、0.03 与 0.01，其中 0.05 是最常用的容忍误差之一，广泛用于商业调查、政策研究、社会科学和市场调查等。其在样本量和准确性之间提供了一个良好的平衡，能够在保持合理准确度的同时控制样本量和成本。当 $E = 0.05$ 时，情形（1）对应最小样本容量为 98，情形（2）对应最小样本容量为 59。

4.2 问题二的分析与求解

4.2.1 问题分析

已知企业生产该电子产品时，由两种零配件装配为成品，再进入市场售卖。已知两种零配件的次品率、购买单价以及成品的次品率、装配成本、市场售价等信息，现需对产品生产过程中的各阶段作出决策。产品的生产流程如图 2所示：

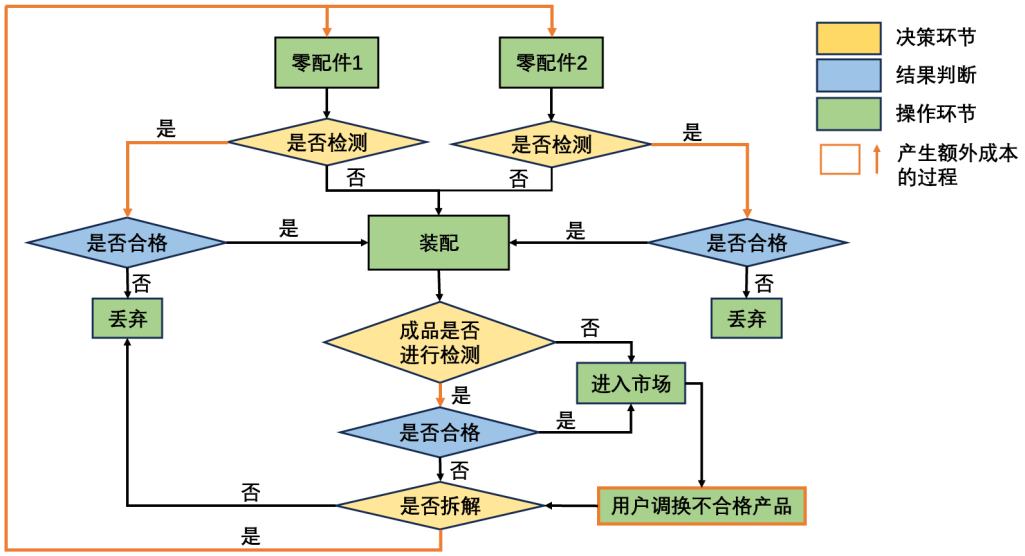

图 2: 产品生产流程图

在该电子产品的生产过程中，共包含 4 次决策过程，分别为是否对零配件 1 进行检测、是否对零配件 2 进行检测、是否对成品进行检测以及是否对不合格成品进行拆解。为设计最佳决策方案，将四个决策过程表示为 4 个取值范围为 0-1 的决策变量 x_1、x_2、x_{prod} 和 $x_{disassemble}$，分别表述是否对零件一、零件二和成品进行检测，以及是否对不合格成品进行拆解。选择进行操作，相应决策变量取值为 1；选择不进行操作，相应决策变量取值为 0。利用 4 个决策变量写出企业总利润表达式，通过 0-1 整数规划求解得到企业总利润最大化的决策方案。

4.2.2 模型的建立与求解

结合题目所给条件，已知零件一与零件二的次品率、单件购买成本和单件检测成本；成品的理想次品率（即成品均由正品零件组装而成时的次品率）、单件检测成本、单件拆解成本和售卖价格，以及不合格成品的调换损失。基于上述已知数据，构建如下 0-1 整数规划模型：

初始购入成本 假设零配件 1 的购买数量为 n_1，购买单价为 p_1，零配件 2 的购买数量为 n_2，购买单价为 p_2，可以求得初始购入成本 C_{init}，如式 7所示：

$$C_{init} = n_1 p_1 + n_2 p_2 \tag{7}$$

检测费用 当选择对零配件 1 进行检测时，x_1 取值为 1，否则取 0；当选择对零配件 2 进行检测时，x_2 取值为 1，否则取 0；当选择对成品进行检测时，x_{prod} 取值为 1，否则取 0。已知单件零配件 1 的检测成本为 d_1，单件零配件 2 的检测成本为 d_2，单件成品的检测成本为 d_{prod}，且零配件 1、零配件 2 及成品的数量分别为 n_1，n_2 和 n_{prod}，由此可以推得总检测费用 C_{detect} 如式 8所示：

$$C_{detect} = n_1 x_1 d_1 + n_2 x_2 d_2 + n_{prod} x_{prod} d_{prod} \tag{8}$$

装配费用 由于组装的成品总数目为 n_{prod}，且单件成品的装配费用为 $p_{assemble}$，因此总装配费用 $C_{assemble}$ 如式 9所示：

$$C_{assemble} = n_{prod}p_{assemble} \tag{9}$$

成品数量 由于成品是由零配件 1 和零配件 2 以 1:1 的比例装配组成，因此成品数量取决于零配件 1 数量和零配件 2 数量中的最小值。若对零配件 1 进行检测，仅有合格零配件参与后续装配过程，不合格零配件将直接丢弃；若不对零配件 1 进行检测，则所有零配件 1 均会参与到装配过程，零配件 2 同理。已知零配件 1 和零配件 2 的次品率分别为 c_1、c_2，若进行检测，则零配件 1 和零配件 2 次品总数量分别为 c_1n_1、c_2n_2。综上，成品数量 n_{prod} 可以表示为式 10：

$$n_{prod} = min\{n_1 - n_1c_1x_1, n_2 - n_2c_2x_2\} \tag{10}$$

成品次品率 根据企业提供条件，当组成成品的零配件均为合格品时，成品的理想次品率已知，记为 c_{prod}。由产品的生产流程可知，当且仅当零配件 1 和零配件 2 均选择检测时，才能保证生产得到的成品组件零配件均为合格品，而当成品组件无法确定是否合格时，无法直接得到成品次品率的取值。因此，作如下推导：

若选择进行零配件检测，零配件 1、零配件 2 中合格零配件所占比例为 1；若不选择进行零配件检测，零配件 1 中合格零配件所占比例为 $1-c_1$，零配件 2 中合格零配件所占比例为 $1-c_2$。因此，经过零配件检测决策后，零配件 1 和零配件 2 中合格零配件所占比例可以分别表示为 $1-c_1(1-x_1)$、$1-c_2(1-x_2)$，组装得到的成品中由两个合格零配件组成的成品比例为 $(1-c_1(1-x_1))(1-c_2(1-x_2))$，不合格成品比例为 $1-(1-c_1(1-x_1))(1-c_2(1-x_2))$。由于只有由两个合格零配件组装得到的成品有一定概率检验合格，且已知由两个合格零配件组装得到的成品次品率为 c_{prod}，因此得到的成品中实际次品所占总比例可表示为式 11形式。

$$c_{prod}^{'} = 1-(1-c_1(1-x_1))(1-c_2(1-x_2)) + c_{prod}(1-c_1(1-x_1))(1-c_2(1-x_2)) \tag{11}$$

由于零配件 1、零配件 2 的次品率取值范围在 $5\% \sim 20\%$ 之间，次品率二次项的取值范围为 $0.25\% \sim 4\%$，远远小于 c_1、c_2，因此在 $c_{prod}^{'}$ 的计算过程中，可以忽略零配件 1 和零配件 2 次品率的二次项。对成品次品率表达式 11进行化简并省略次品率二次项可得表达式 12。

$$\begin{aligned} c_{prod}^{'} &= 1-(1-c_1(1-x_1))(1-c_2(1-x_2)) + c_{prod}(1-c_1(1-x_1))(1-c_2(1-x_2)) \\ &\approx c_{prod} + (1-c_{prod})(c_1(1-x_1) + c_2(1-x_2)) \end{aligned} \tag{12}$$

调换费用 当成品不经检测进入市场，用户有一定概率购买到不合格成品。对于购买到不

合格成品的用户，企业需要提供无条件调换。调换过程中，除成品自身成本外，同时会产生如物流成本、企业信誉等调换损失。已知单件成品市场售价为 p_{prod}，调换过程中单件不合格成品导致的调换损失为 p_{change}，故单件不合格成品产生的总调换成本为 $p_{prod}+p_{change}$，即认为损失了一个合格成品的售卖价格以及调换过程中产生的额外损失。

已知成品的实际次品率为 $c_{prod}^{'}$。由于当且仅当成品不经检测（也即 $x_{prod}=0$ 时）进入市场会导致调换事件的发生，因此可推到进入市场的不合格成品总数为 $(1-x_{prod})n_{prod}c_{prod}^{'}$。

综上不合格成品进入市场产生的调换总成本 C_{change} 可表示为式 13：

$$C_{change}=(p_{prod}+p_{change})(1-x_{prod})n_{prod}c_{prod}^{'} \tag{13}$$

拆解成本 若成品进行检测，直接进入不合格成品是否拆解决策；若成品未进行检测，不合格成品流入市场后被调回仍将进入是否拆解决策，因此不对不合格成品的来源进行区分，所有不合格成品都需进行是否拆解决策。当选择对成品进行拆解时，$x_{disassemble}$ 取值为 1，否则取值为 0。已知组装成品总数为 n_{prod}，不考虑组成成品零配件合格与否的情况下，成品的实际次品率为 $c_{prod}^{'}$，则拆解的不合格成品数量 $n_{disassemble}=x_{disassemble}n_{prod}c_{prod}^{'}$。拆解单个成品的成本为 $p_{disassemble}$，可以得到总拆解成本 $C_{disassemble}$ 如式 14所示：

$$C_{disassemble}=x_{disassemble}n_{prod}c_{prod}^{'}p_{disassemble} \tag{14}$$

拆解成品操作后处理 前文拆解成本部分仅考虑了拆解操作本身带来的额外成本，此部分将对拆解后得到的零配件进行具体分析。对不合格成品进行拆解的主要目的为提高零配件的利用率，因此在以下分析中将零配件利用率作为分析指标。在生产流程中，若选择对不合格成品进行拆解，则零配件 1、零配件 2 数量增加值均为 $c_{prod}^{'}n_{prod}$。分析得到的成品检测不合格的原因有两种，一是组成成品的零配件存在检测不合格组件，二是在两个零配件均合格的情况下由于其他原因导致的不合格。因此，若此前零配件检测环节已对零配件进行过检测操作，不再对得到的零配件进行检测操作；若此前零配件未经过检测操作，则检测得到的零配件是否合格。经过检测决策后，零配件 1 和零配件 2 的新增数量分别为 $c_{prod}^{'}n_{prod}-(1-x_1)c_{prod}^{'}n_{prod}c_1$、$c_{prod}^{'}n_{prod}-(1-x_1)c_{prod}^{'}n_{prod}c_2$。由前文可知，经过初始零配件检测决策环节，零配件 1 和零配件 2 的原始数量分别为 $n_1-n_1x_1c_1$、$n_2-n_2x_2c_2$。由此可得拆解操作后，零配件 1 和零配件 2 元件利用率的提高值，如式 15所示：

$$\begin{cases}\Delta\epsilon_1=\dfrac{c_{prod}^{'}n_{prod}-(1-x_1)c_{prod}^{'}n_{prod}c_1}{n_1-n_1x_1c_1}\\ \Delta\epsilon_2=\dfrac{c_{prod}^{'}n_{prod}-(1-x_1)c_{prod}^{'}n_{prod}c_2}{n_2-n_2x_2c_2}\end{cases} \tag{15}$$

拆解操作后得到的新增零配件 1、零配件 2 导致的新增检测成本 $C_{exdetect}$ 可表示为式 16。

$$C_{exdetect} = x_{disassemble} n_{prod} c_{prod}((1-x_1)d_1 + (1-x_2)d_2) \tag{16}$$

成本修正 拆解操作后得到的零配件 1 和零配件 2 经过检测后再次进入装配环节，产生新的成品。为了简化计算，结合题意合理假设，认为组装成品数量提升率取两个零件利用提升率的均值（$\frac{\Delta\epsilon_1+\Delta\epsilon_2}{2}$），即利用零件利用提升率来表述拆解决策对整体生产流程所带来的增益 $\Delta\epsilon$。已知初始组装成品数量 $n_{prod} = min\{n_1 - n_1x_1c_1, n_2 - n_2x_2c_2\}$，经过拆解决策后，组装成品数量修正为 $n_{prod}^{'}$，表达式如式 17:

$$n_{prod}^{'} = (n_{prod} - n_{disassemble})(1 + \Delta\epsilon_{prod} x_{disassemble}) \tag{17}$$

组装成品数量修正操作对初始购入成本、零配件 1 和零配件 2 检测成本、拆解成本无影响，而成品检测成本、装配成本以及调换成本中的 n_{prod} 均需替换为 $n_{prod}^{'}$，修正后各成本值如式 18所示。

$$\begin{cases} C_{init} = n_1p_1 + n_2p_2 \\ C_{detect} = n_1x_1d_1 + n_2x_2d_2 + n_{prod}^{'}x_{prod}d_{prod} \\ C_{assemble} = n_{prod}^{'}p_{assemble} \\ C_{disassemble} = x_{disassemble}n_{prod}p_{disassemble}c_{prod}^{'} \\ C_{change} = (p_{prod} + p_{change})(1 - x_{prod})n_{prod}^{'}c_{proc}^{'} \\ C_{exdetect} = x_{disassemble}n_{prod}c_{prod}((1-x_1)d_1 + (1-x_2)d_2) \end{cases} \tag{18}$$

综上，当零配件 1、2 数量分别为 n_1、n_2 时，该生产流程产生总生产成本 C 可表达为式 19。

$$C = C_{init} + C_{detect} + C_{assemble} + C_{disassemble} + C_{change} + C_{exdetect} \tag{19}$$

企业总收入 最终组装成品总数量为 $n_{prod}^{'}$，考虑是否对成品进行检测：若不检测，此时 $x_{prod} = 0$，成品全部售出；若对成品进行检测，此时 $x_{prod} = 1$，只有合格成品售出。单个成品市场售价为 p_{prod}，则企业出售所有组装成品的总收入 I 表达式为式 20。

$$I = p_{prod}n_{prod}^{'}(1 - x_{prod}c_{prod}^{'}) \tag{20}$$

企业的总利润 P 可表达为 $P = I - C$，将上述式(7)–(20)带入可得企业总利润的完整表达式。

决策优化的目标函数 P_{max} 为:

$$P_{max} = (I - C)_{max} \tag{21}$$

可知该目标函数为 0-1 整数规划问题，故利用 Matlab 按上述步骤编写利润计算函数，函数输入为 0-1 编码的决策方案序列，输出为该方案下单位零件数所产生的利润值，也即决策优化目标函数值。所有输出值中的最大值及其对应决策方案即为所求。

4.2.3 模型结果

题目中给定 6 种情况条件信息如表 2所示:

表 2: 问题 2 给定 6 种情况下生产成本相关信息

情况	零配件 1			零配件 2			成品				不合格成品	
	次品率	购买单价	检测成本	次品率	购买单价	检测成本	次品率	装配成本	检测成本	市场售价	调换成本	拆解成本
1	10%	4	2	10%	18	3	10%	6	3	56	6	5
2	20%	4	2	20%	18	3	20%	6	3	56	6	5
3	10%	4	2	10%	18	3	10%	6	3	56	30	5
4	20%	4	1	20%	18	1	20%	6	2	56	10	5
5	10%	4	8	20%	18	1	10%	6	2	56	10	5
6	5%	4	2	5%	18	3	5%	6	3	56	10	40

由于零配件 1、零配件 2 按 1:1 的比例组装成成品，因此假设购入零配件 1 和零配件 2 数量相同，也即 $n_1 = n_2$。由于 n_1、n_2 取任何值均等效于 $n_1 = n_2 = 1$ 情况的等效放大，因此计算过程中，将 n_1、n_2 设置为 1。将上述 6 中情况各参数带入模型求解，得到各情况下最优方案如表 3所示（企业总利润均保留两位小数）。

表 3: 6 种情况下企业最优决策方案以及总利润最大值

情形	是否对零配件 1 进行检测	是否对零配件 2 进行检测	是否对成品进行检测	是否对不合格成品进行拆解	企业总利润最大值
情形 1	✗	✗	✗	✓	16.07
情形 2	✓	✓	✗	✓	8.30
情形 3	✗	✗	✓	✓	14.41
情形 4	✓	✓	✓	✓	10.53
情形 5	✗	✓	✗	✓	9.89
情形 6	✗	✗	✗	✗	18.43

(1) 情形 1： 根据表 2已知条件，情形 1 中零配件 1、零配件 2 以及成品次品率取值均为 10%。在此条件下，装配所得的成品不合格概率较低，检测零配件、成品带来的成本损失较大。选择不对零配件、成品进行检测，能够节省检测过程产生的成本，提高企业盈利。情形 1 下每种不合格成品拆解成本为 5，选择对不合格成品进行拆解对成本的增加作用不大，同时能够提高 2 种零配件的利用率。因此预期最优决策应为不对零配件 1、2 及成品进行检测，对不合格成品进行拆解。对比表 3可知，情形 1 下，模型得到的企业最大利润为 16.07，对应的

最优决策方案为不对零配件 1、零配件 2 及成品进行检测，对不合格成品进行拆解，符合理论预期。

（2）情形 2： 根据表 2已知条件，情形 2 中零配件 1、零配件 2 以及成品的次品率分别为 20%、20%、20%，不合格成品拆解费用为 5。由于零配件 1、零配件 2 以及成品的次品率均较大，若不对零配件及成品进行检测，进入市场中的不合格成品比例较高，产生的调换损失也会随之增大。因此，预期最优决策应该对零配件 1、零配件 2 及成品进行检测。同时在拆解成本较低的情况下，为了提高零配件利用率，应当选择对不合格成品进行拆解操作。对比表 3可知，情形 2 下，模型得到的企业总利润最大值为 8.30，最优决策方案为对零配件 1、零配件 2 及成品进行检测，对不合格成品进行拆解，与预期最优方案一致。

（3）情形 3： 根据表 2已知条件，情形 3 与情形 1 相比，仅不合格成品调换损失明显增大，情形 1 条件下不合格成品调换损失为 6，而情形 3 下不合格成品调换损失为 30。调换损失增大则不合格成品进入市场带来的损失增大，因此应当在成品进入市场前对成品进行检测，避免不合格成品进入市场。在其他条件与情形 1 一致的情况下，情形 3 预期最优决策方案应当为不对零配件 1、2 进行检测，对成品进行检测，对不合格成品进行拆解。对比表 3可知，情形 3 下，模型所得企业总利润最大值为 14.41，对应的决策方案与预期决策方案相同。

（4）情形 4： 根据表 2已知条件，情形 4 与情形 2 相比，零配件 1、零配件 2 以及成品的单件检测成本减少，调换损失明显增大，情形 2 中不合格成品调换损失为 6，而情形 4 中不合格成品调换损失为 30。因此情形 4 预期最优决策方案应当在情形 2 最优决策方案基础上增加对成品进行检测的操作，也即对零配件 1、2 及成品进行检测，对不合格成品进行拆解。对比表 3可知，情形 4 下，模型所得企业总利润为 10.53，最优决策方案与预期一致。

（5）情形 5： 根据表 2已知条件，情形 5 对应的零配件 1、零配件 2 以及成品的次品率分别为 10%、20%、10%，检测成本分别为 8、1、2。由于零配件 1 次品率较低且检测成本较高，不选择对零配件 1 进行检测；零配件 2 次品率较高且检测成本较低，应当对零配件 2 进行检测；成品次品率较低，在零配件 1 次品率较低且零配件 2 进行检测情况下，不选择对成品进行检测；拆解成本较低，可选择对不合格成品进行拆解操作。综上，预期最优决策方案为不对零配件 1 进行检测，对零配件 2 进行检测，不对成品进行检测，对不合格成品进行拆解。对比表 3可知，情形 5 的情况下，模型所得企业利润最大为 9.89，对应决策方案与预期一致。

（6）情形 6： 根据表 2已知条件，情形 6 零配件 1、零配件 2 及成品的次品率取值均为 5%，不选择对两种零配件及成品进行检测。拆解费用为 40，明显高于其余 5 种情况，不选择对不合格成品进行拆解操作。因此预期最优方案为不对两种零配件及成品进行检测，不对不合格成品进行拆解。对比表 3可知，情形 6 的情况下，模型所得企业利润最大值为 18.43，对应决策方案与预期一致。

根据求解结果，在以上六种情形中，该整数规划模型在面对不同的次品率、检测费用、拆解费用以及调换费用等条件时，计算出的最优决策方案均能有效地模拟实际情况，并高效地

求解最优决策方案。由此可知该模型在求解生产过程决策问题时具有很强的实用性和可靠性，能够在实际应用中为决策者提供有效的科学依据。

4.3 问题三的分析与求解

4.3.1 问题分析

在问题二的基础上，零配件数量增加为 n 个，工序增加为 m 道。在已知所有零配件、半成品以及成品的次品率的情况下，现需求解使企业总利润最大化的决策方案。相比于问题二，问题三的决策总数明显增加，问题规模扩大。为便于求解，在原模型的基础上引入多阶段决策模型分析企业总收益，并使用遗传算法搜索最优决策方案。多工序成品生产流程如图 3所示。

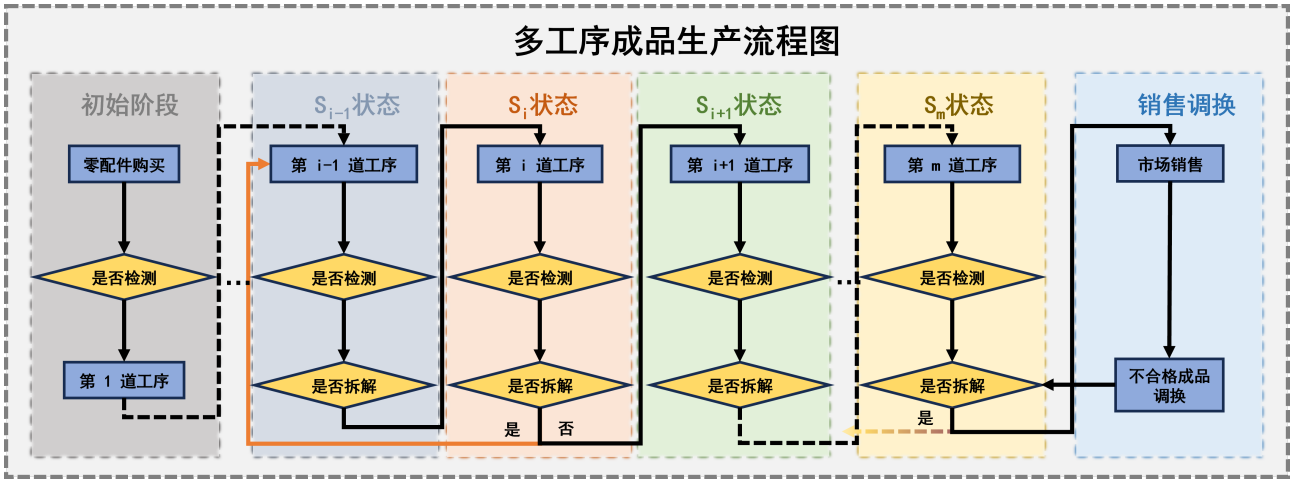

图 3: 多工序成品生产流程图

在初始阶段，企业购入零配件，并决策是否检测零配件，之后进入第一道装配工序。在装配工序过程中，零配件组装为半成品，再由半成品装配为下一阶段半成品。若在工序 i 决定对不合格半成品进行拆解，则拆解后得到的零件/半成品返回上一工序进行检测。当进入最后一道工序，半成品组装为成品后（即状态 S_m），对成品进行是否检测决策，并对不合格成品进行是否拆解决策，最后进入市场销售和售后调换环节。基于该多工序成品生产流程，构建 0-1 整数规划模型求解最优决策方案。

4.3.2 模型的建立与求解

问题三成品组装为多工序过程，每道工序的每个零件都涉及检测与拆解决策，为便于表述，用 n 维向量表示各工序零件的次品率、检测价格、拆解价格、装配价格等，建立多阶段 0-1 整数规划模型如下。

工序 i 后工作状态记作 S_i。对于每道工序，分析其生产过程所耗成本。在每道工序 i 中，对于零件是否检测，用 0-1 变量向量 $X_i = [x_1 \quad x_2 \dots x_n]^T$ 表示；对于检测后的不合格品是否

拆解，用 0-1 变量向量 $Y_i = [y_1 \quad y_2 \dots y_n]^T$ 表示。

状态转移方程 根据式 10，状态 S_i 到状态 S_{i+1} 的半成品/成品的数量 n_i 状态转移方程可表示为式 22 23。

$$E_{i+1}^{'} = \left[(1 - X_i)^T E_i^{'}\right] \circ (1 - E_{i+1}) + E_{i+1} \tag{22}$$

$$N_{(i+1)} = min\{N_i - N_i \circ E_i^{'} \circ X_i^{'}\} \circ (1 + \Delta\epsilon_i Y_i) \tag{23}$$

式 22是实际次品率转移方程，即经过前一工序装配后，所得到的半成品/成品的实际次品率向量。式 23为数量状态转移方程，及经过前一道工序检测、拆解和装配后所得到的下一工序的产品数量，其中 $\Delta\epsilon_i$ 为第 i 道工序拆解决策后所带来的零件利用提升率。由于拆解后的零件要返回上一工序重新进行检测和装配，从而会带来产品数量的提升，因此定义 $\Delta\epsilon_i$ 如式 24：

$$\Delta\epsilon_i = \frac{N_{di} - N_{di} \circ (1 - X_{i-1}) \circ E_{i-1}^{'}}{N_{i-1} - N_{i-1} \circ E_{i-1}^{'} \circ X_{i-1}} \tag{24}$$

式 24中 N_{di} 为第 i 道工序的拆解零件数量，可表示为式 25：

$$N_{di} = N_i \circ E_i^{'} \circ Y_i \tag{25}$$

由于拆解会导致零件利用率提升，从而提高得到的产品数量，故对每一阶段的产品数量进行修正：

$$N_i^{'} = N_i \circ (1 + \Delta\epsilon \circ Y_i) \tag{26}$$

式 26表示当在工序 i 实行拆解决策时，会对产品数量有所增益，故进行产品数量修正。

状态 S_i 决策成本 状态 S_i 下决策成本包括检测成本、拆解成本以及装配成本。对问题二中检测成本、拆解成本及装配成本进行状态转移扩写。已知状态 S_i 下的各半成品/成品的装配费用向量为 C_{ai}，装配成本可表示为以下形式：

$$C_{ai}^T N_i^{'}$$

已知状态 S_i 下的各半成品/成品的检测费用向量为 C_{ti}，检测决策向量为 X_i，检测成本可表达为以下形式：

$$C_{ti}^T (N_i^{'} \circ X_i)$$

已知状态 S_i 下的各半成品/成品的拆解成本向量为 C_{di}，拆解决策向量为 Y_i，拆解成本可表达为以下形式：

$$C_{di}^T (N_i^{'} \circ E_i \circ Y_i)$$

状态 S_i 下决策产生的总成本 C_i 表达式如式 27所示：

$$C_i = C_{ai}^T N_{'} + C_{ti}^T (N_i^{'} \circ X_i) + C_{di}^T (N_i^{'} \circ E_i \circ Y_i) \tag{27}$$

购买零配件初始成本 已知初始购买零配件价格向量为 P_0，初始购买零配件数量向量为 N_0，购买成本表达式为式 28所示：

$$C_{init} = P_0^T N_0 \tag{28}$$

调换成本 已知成品调换损失为 P_{change}，调换损失可表达为式 29。

$$C_{change} = (P_{change} + P_{prod})^T (1 - X_m) \circ E_m^{'} \circ N_{m+1} \tag{29}$$

企业总利润 结合上述分析，企业总利润可表示为式 30。

$$P = (P_{prod})^T N_{m+1} - C_{init} - C_{change} - \sum_{i=0}^{m} C_i \tag{30}$$

故该多阶段整数规划的目标函数为：

$$\max P = (P_{prod})^T N_{m+1} - C_{init} - C_{change} - \sum_{i=0}^{m} C_i \tag{31}$$

应用遗传算法搜索最优决策方案 由于问题三中问题规模扩大，故考虑采用遗传算法求解。遗传算法是一种仿照达尔文进化论，基于自然选择和遗传变异等生物机制的迭代搜索算法，包括自然选择、杂交和突变等基本进化过程 [5]。其通过模拟自然选择的过程，能够在广泛的搜索空间中找到全局最优解，而不仅仅是局部最优解，适合解决复杂性高和高维问题。

- **初代种群** 遗传算法中，每一种方案为一个物种，一组方案的集合构成种群。每一种方案由组成方案的所有 0-1 决策变量组成，每一个决策变量构成一个基因。初始总群规模为 N，种群中个体通过随机函数随机生成。
- **适应度函数** 物种适应度函数以物种种类（也即具体方案）为自变量，是衡量物种适应能力强弱的指标。在自然选择过程中，适应度越高的物种参与下一代繁殖的概率越高。应用遗传算法时，需确定方案的优化目标，根据方案的优化目标设置合适的适应度函数。方案所得结果越接近优化目标，对应适应度函数的取值越大。
- **自然选择** 利用设置的适应度函数评估初代种群中个体的适应能力，选择其中适应能力强的个体参与繁殖下一代。
- **交叉配对** 选中的个体模拟自然界中染色体交叉过程，互换部分基因组成。
- **基因变异** 参与繁殖下一代的物种的基因有一定的概率发生突变。随机选择变异概率比

例的基因进行取反操作，得到新一代种群。

- **迭代** 新一代种群仿照初代种群继续迭代过程，达到目标迭代次数后停止迭代，此时种群中适应能力最强物种所对应的决策方案即为最优决策方案。

应用遗传算法搜索最优决策方案迭代流程图

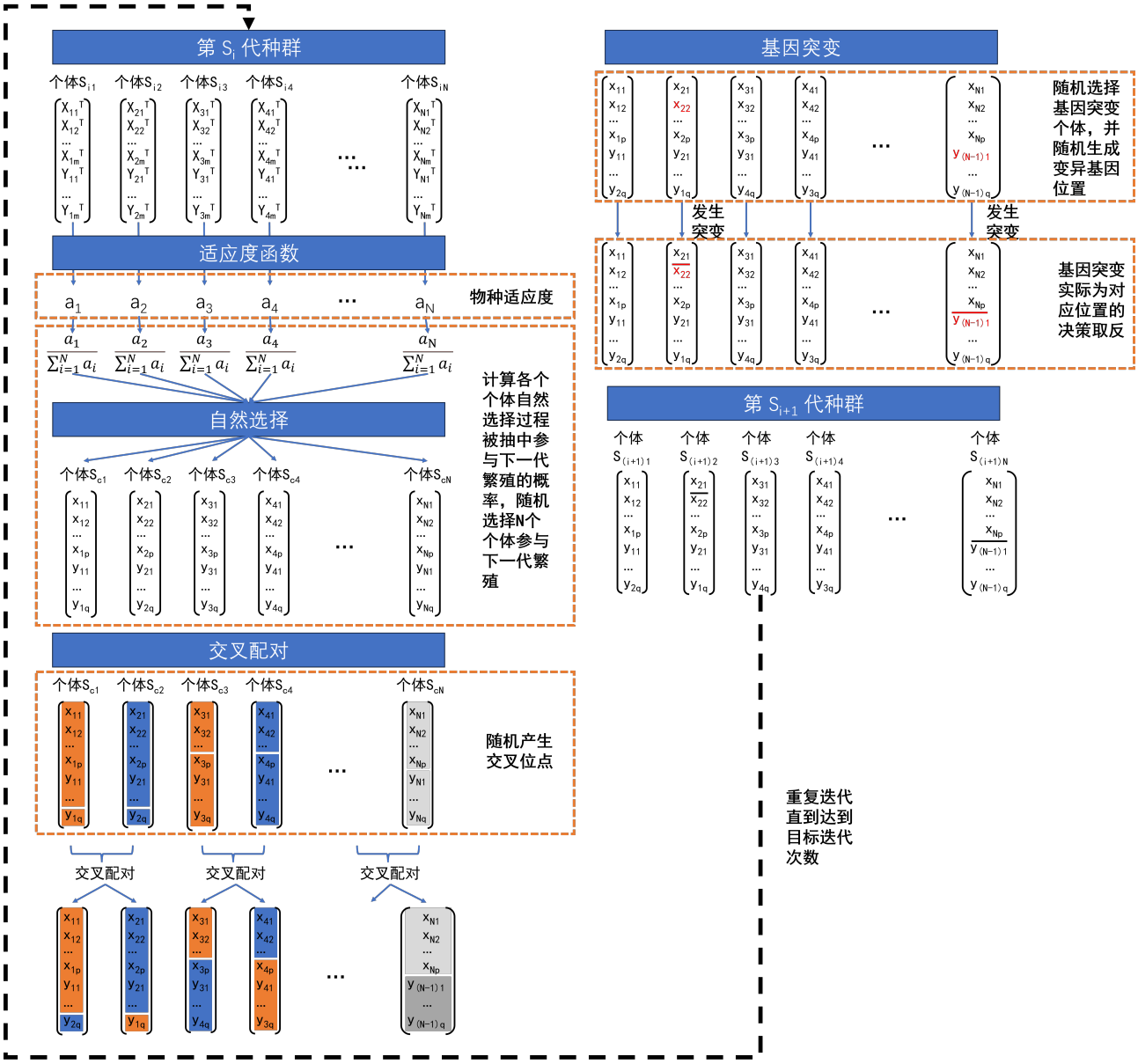

图 4: 应用遗传算法搜索最优决策方案迭代流程图

将已知的零配件、半成品及成品信息带入式 30，得到仅含 X_0、$X_1...X_m$、Y_0、$Y_1...Y_m$ 决策变量的企业总利润表达式。将全部决策变量组合得到一个总决策向量 D_{xy}，作为遗传算法中的一个物种。设置种群规模为 N，初始种群中个体由随机函数随机生成。决策的优化目标为提高企业总利润，决策方案对应企业总利润越大，被选择的概率越大，因此将企业总利润表达式作为物种适应度函数。

反映到遗传算法中，决策方案下所得企业利润越大，对应方案物种对应适应度函数取值

越大，参与繁殖下一代的概率越大。根据适应度函数计算初始群落中各个个体的适应度，将个体适应度与群体适应度的比值作为该个体被抽中下一代繁殖的概率，所有个体被抽中概率之和为 1。若将各个个体被抽中概率依次放置在数轴上，得到的线段总长度为 1。利用随机函数生成一个 $[0,1]$ 之间的随机数，随机数所在个体区段对应个体即为自然选择中被抽中个体，抽取操作重复 N 次，得到规模为 N 的高适应度个体集合。

为了简化染色体交叉过程，默认被抽中的第一个个体与抽中的第二个个体进行交叉配对，抽中的第三个个体与抽中的第四个个体进行交叉配对，以此类推。设决策向量 D_{xy} 中共包括 p 次检测决策以及 q 次拆解决策。利用随机函数为每一对交叉组合生成一个交叉位点，每对中的两个个体交换基因交叉位点到基因末尾的全部基因，得到规模为 N 的交叉后个体集合。

基因变异概率设置为 α，在交叉后群体所有基因中以 α 的概率抽取相应数量的基因进行变异，对应决策方案中操作为对应位置决策取反。完成所有选中基因变异后，得到新一代种群。新一代种群重复上述过程，直到迭代次数达到目标迭代次数，选择最后一代中种群中适应度最大个体对应的决策方案作为最终的最优决策方案，上述流程如图 4所示。

4.3.3 模型结果

当零配件数目 n 为 8，工序数量 m 为 2 时，已知企业生产过程中的成本相关信息如表4所示。

表 4: 零配件数目为 8，工序数量为 2 时企业生产过程中成本相关信息

零配件	次品率	购买单价	检测成本	半成品	次品率	装配成本	检测费用	拆解费用
1	10%	2	1	1	10%	8	4	6
2	10%	8	1	2	10%	8	4	6
3	10%	12	2	3	10%	8	4	6
4	10%	2	1					
5	10%	8	1	成品	10%	8	6	10
6	10%	12	2					
7	10%	8	1		市场售价		调换损失	
8	10%	12	2	成品	200		40	

将上述已知信息带入企业总收益函数中，可得仅含检测决策向量 X 和拆解决策向量 Y 的总利润表达式。将决策向量 X、Y 整合成为一个向量 D_{xy}，整理总利润表达式，使之成为以 D_{xy} 为自变量的函数，所得函数为遗传算法的适应度函数。遗传算法中种群规模 N 取为 100，迭代次数取 20。由于在以上参数条件下最终种群中物种适应度已趋于稳定，不考虑进一步扩大种群规模及迭代次数。在以上条件下，模型所得最优决策方案如表 5（企业总利润保留 2 位小数）。

表 5: 问题 3 已知条件下最优决策方案

决策	是否进行决策	决策	是否进行决策
是否对零件 1 进行检测	✓	是否对零件 2 进行检测	✓
是否对零件 3 进行检测	✓	是否对零件 4 进行检测	✓
是否对零件 5 进行检测	✓	是否对零件 6 进行检测	✓
是否对零件 7 进行检测	✓	是否对零件 8 进行检测	✓
是否对半成品 1 进行检测	✓	是否对半成品 2 进行检测	✓
是否对半成品 3 进行检测	✓	是否对半成品 1 进行拆解	✓
是否对半成品 2 进行拆解	✓	是否对半成品 3 进行拆解	✓
是否对成品进行检测	✓	是否对成品进行拆解	✓
企业利润最大值		50.76	

根据表 4，8 种零配件，3 种半成品及成品的次品率均为 10%，单件检测成本分别为 1、1、2、1、1、2、1、2、4、4、4、6，单件成品的市场售价为 200，单件不合格成品的调换损失为 40。相比于不合格成品调换产生的成本，零配件、半成品及成品的检测成本较低，因此选择对零配件、半成品及成品进行检测，以避免不合格成品进入市场。3 种半成品及成品的单件拆解费用分别为 6、6、6、10，单件装配成本均为 8。若不拆解直接将不合格半成品及成品丢弃，将损失半成品/成品中的零配件购买费用以及半成品/成品装配费用，损失费用大于拆解操作带来的成本。与此同时，拆解操作能够提高零配件的利用率，有利于最大化企业受益，因此预期最优决策应选择对不合格半成品/成品进行拆解。综上，预期最优决策方案应检测 8 种零配件、3 种半成品以及成品，拆解所有不合格半成品及成品。对比表 5，企业最大总利润为 50.76，对应最优决策方案与预期方案一致，表明模型能够很好地模拟实际情况并求得最优决策方案。

4.4 问题四的分析与求解

4.4.1 问题分析

前文问题二、问题三中零配件、半成品和成品次品率默认为所有零配件、半成品和成品是次品的概率。现假设前文给出的次品率均为利用问题一中抽样检测方案所得，要求重新求解问题二、三条件下的最优决策方案。

4.4.2 模型的建立与求解

在问题四中，零配件、半成品和成品的次品率均由抽样检测得到，故采用贝叶斯后验进行真实次品率的估计。假设零配件、半成品及成品的实际分布符合 $Beta$ 分布，$Beta$ 分布的

概率密度函数如式 32所示，不同 α、β 取值下概率密度函数曲线如图 5所示。

$$f(x;\alpha,\beta)=\frac{x^{\alpha-1}(1-x)^{\beta-1)}}{\int_0^1 u^{\alpha-1}(1-u)^{\beta-1}du} \tag{32}$$

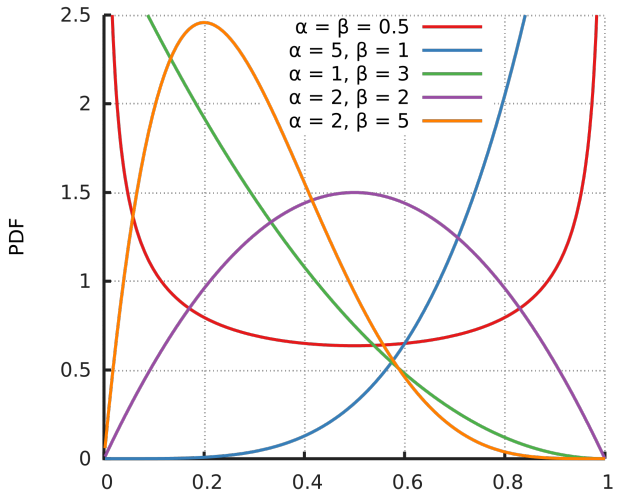

图 5: 不同 α、β 取值下 $Beta$ 分布概率密度函数曲线图

$Beta$ 分布的峰值对应 x 取值为 $\frac{\alpha-1}{\alpha+\beta-2}$，数学期望 μ 为 $\frac{\alpha}{\alpha+\beta}$。根据问题一到三的已知信息，认为零配件、半成品及成品的次品率的先验分布在 $0\%\sim20\%$ 之间，且由于次品率倾向于取低值，故 α 取值为 2，β 取值为 5，得到零配件、半成品及成品的先验概率分布为 $Beta(x;\alpha,\beta)$。

对零配件、半成品及成品进行抽样检测。根据问题 1，在次品率标称值、容忍误差已知的情况下，能够求解出 95% 信度下接收此批零配件所需的最小抽样检测样本容量。假设求解所得的样本容量为 k，样本的次品率为 e，抽中样品中次品数量可表示为 ek，合格品数量可表示为 $(1-e)k$。根据抽样数据对先验假设进行修正，将 α 修正为 $\alpha+ek$，β 修正为 $\beta+(1-e)k$。得到修正后的零配件/半成品/成品的次品率分布为 $Beta(x;\alpha+ek,\beta+(1-e)k$，修正后的次品数学期望 $\hat{p}$ 如式 33所示：

$$\hat{p}=\frac{\alpha+ek}{\alpha+\beta+k} \tag{33}$$

置信水平取 95%，利用已知的修正后次品率分布函数，求解得到 95% 信度下零配件/半成品/成品次品率的置信区间 $[p_{lower},p_{upper}]=[F^{-1}(0.025),F^{-1}(0.075)]$。

由于此时零配件、半成品及成品的次品率均由抽样检测得到，其实际次品率具有不确定性，因此考虑使用鲁棒优化方法，通过构建多阶段决策鲁棒优化模型搜索最优决策方案。当使用连续的区间范围描述不确定性参数时，需要寻求一个近似最优解。近似最优解对不确定性参数观测值变化不敏感。鲁棒优化的最大特点在于考虑了不确定性参数数值实现后，不同目标函数值之间的差异[6]，具有较强的稳定性和实用性。

鲁棒优化在处理时的关键问题是不确定集合 u 的确定，以及如何在给定 u 的情况下对复

杂模型的化简 [7]。采用区间上下限描述次品率的不确定性称为盒式不确定性，又称区间集，该类不确定集可几乎涵盖所有次品率不确定性情况，是普遍使用的一种不确定集，适合在各个维度上独立地限定不确定参数的范围。

盒式不确定集 U 的形式通常是一个多维矩形区域，可以定义为：

$$\mathcal{U} = \{u \in \mathbb{R}^n : l_i \leq u_i \leq u_i, \forall i = 1, \ldots, n\} \tag{34}$$

其中 $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$ 是不确定参数的向量（维度为 $n \times 1$），l_i 和 u_i 是不确定参数 u 的每个分量 u_i 在第 i 个维度上的下届和上界，即 $l_i = p_{lower}$，$u_i = p_{upper}$。

得到盒不确定集后，鲁棒优化问题可以被表示为式 35：

$$\max_{x} \max_{\mathbf{u} \in \mathcal{U}} f(\mathbf{x}, \mathbf{u}) \tag{35}$$

最坏情况出现在 U 的边界点。因此可以通过比较在不确定集边界点的目标值来找到最坏情况。代入问题二与问题三数据，得到实际次品率均值如表 6（可容忍误差 E 取值为 0.05。根据所给信息，抽样检测所得样本次品率共有有 5%、10%、20% 三种情况，因此表中给出可容忍误差为 0.05 时，3 种抽样检测次品率情况下的实际次品率预测值以及 95% 的置信区间，均保留 2 位小数）。

表 6: 问题 2 及问题 3 中零配件/半成品/成品实际次品率

抽样检测所得次品率	实际次品率预测值	95% 置信区间下限值	95% 置信区间上限值
5%	7.80%	2.47%	15.80%
10%	11.24%	5.96%	17.91%
20%	20.33%	14.81%	26.48%

4.4.3 模型结果

应用鲁棒优化方法，将 95% 置信区间上限值（即次品率取最大值）作为实际次品率带入问题 2 和问题 3 中重新求解，得到的方案分别如表 7、表 8所示（结果均保留 2 位小数）。

表 7: 最大次品率情况下问题 2 中 6 种情形对应最优决策方案

情形	是否对零配件 1 进行检测	是否对零配件 2 进行检测	是否对成品进行检测	是否对不合格成品进行拆解	企业利润最大值
情形 1	✓	✓	✗	✓	9.92
情形 2	✓	✓	✗	✓	3.26
情形 3	✓	✓	✓	✓	8.05
情形 4	✓	✓	✓	✓	5.87
情形 5	✗	✓	✓	✓	1.47
情形 6	✓	✓	✗	✗	6.32

表 8: 最大次品率情况下问题 3 对应最优决策方案

决策	是否进行决策	决策	是否进行决策
是否对零件 1 进行检测	✓	是否对零件 2 进行检测	✓
是否对零件 3 进行检测	✓	是否对零件 4 进行检测	✓
是否对零件 5 进行检测	✓	是否对零件 6 进行检测	✓
是否对零件 7 进行检测	✓	是否对零件 8 进行检测	✓
是否对半成品 1 进行检测	✓	是否对半成品 2 进行检测	✓
是否对半成品 3 进行检测	✓	是否对半成品 1 进行拆解	✓
是否对半成品 2 进行拆解	✓	是否对半成品 3 进行拆解	✓
是否对成品进行检测	✓	是否对成品进行拆解	✓
企业利润最大值		29.85	

由于鲁棒优化是一种考虑次品率不确定性最劣场景下的优化方法，且在上述条件下，次品率最劣场景发生的概率不超过 5% [8]。因此采用该类不确定集的鲁棒优化结果过于保守，会降低生产系统运转的经济性。

取次品率为实际次品率预测值，重新带入问题 2、问题 3 中模型进行求解，得到的最优决策方案如表 9、表 10所示。

表 9: 预测次品率情况下问题 2 中 6 种情形对应最优决策方案

情形	是否对零配件 1 进行检测	是否对零配件 2 进行检测	是否对成品进行检测	是否对不合格成品进行拆解	企业利润最大值
情形 1	✓	✓	✗	✓	14.99
情形 2	✓	✓	✗	✓	8.04
情形 3	✗	✗	✓	✓	13.21
情形 4	✓	✓	✓	✓	10.29
情形 5	✗	✓	✓	✓	8.79
情形 6	✓	✓	✗	✓	14.71

表 10: 预测次品率情况下问题 3 对应最优决策方案

决策	是否进行决策	决策	是否进行决策
是否对零件 1 进行检测	✓	是否对零件 2 进行检测	✓
是否对零件 3 进行检测	✓	是否对零件 4 进行检测	✓
是否对零件 5 进行检测	✓	是否对零件 6 进行检测	✓
是否对零件 7 进行检测	✓	是否对零件 8 进行检测	✓
是否对半成品 1 进行检测	✓	是否对半成品 2 进行检测	✓
是否对半成品 3 进行检测	✓	是否对半成品 1 进行拆解	✓
是否对半成品 2 进行拆解	✓	是否对半成品 3 进行拆解	✓
是否对成品进行检测	✓	是否对成品进行拆解	✓
企业利润最大值		47.58	

与次品率取 95% 置信区间上限值 p_{upper} 相比，当次品率取真实次品率取贝叶斯后验均值时（$p = \hat{p}$）时，仅问题 2 中情形 3、情形 6 决策方案发生变化；其他情况下最优决策方案不变，但企业最优决策方案下企业所得总利润最大值明显下降。

情形 3 中，当零配件 1、零配件 2 及成品次品率均取最大值时，若不对 2 种零配件进行检测，组装得到的成品中不合格成品的比例将大大增加，拆解所需成本大幅提高。且根据已知信息，零配件 1、2 的检测成本较低，选择对零配件进行检测增加成本对总成本影响较小，同时有助于降低不合格成品的拆解成本。因此当次品率取最大值时，选择对 2 种零配件进行检测。而当次品率取实际次品率预测值时，由于零配件的次品率较低，组装得到的不合格成品数量较少，拆解不合格成品增加的费用小于检测所需费用，因此不对 2 种零配件进行检测。

情形 6 中，当次品率取最大值时，由于成品未进行检测，进入市场中的不合格成品数量较多，且根据已知信息，情形 6 下不合格成品的拆解费用远大于其他情形。若选择拆解不合格成品，则拆解操作导致的新增成本将大幅增加，因此不对不合格成品进行拆解。当次品率取实际次品率预测值时，由于不合格成品数量较少，拆解不合格成品有助于提高零配件利用率，因此选择拆解不合格成品。

在其余情况下，次品率取最大值与实际次品率预测值时最优决策方案一致。但利润明显增加，主要是因为次品率提升导致最终所得的合格成品数量较少，企业收入降低，同时不合格成品带来的损失增加，因此企业总利润显著降低。实际结果与理论分析一致，说明问题 1 中设计的抽取检测方式与实际情况较为符合，且能够与问题 2、3 中建立模型共同作用，有效提高企业总收益。

五、 敏感性分析

综合考虑模型的普遍性、代表性，以及敏感性分析中数据的直观性，我们以问题三中所给出的两道工序、8 个零配件的实际生产数据情况为基准进行模型的敏感性分析实验。

实验方案：

依次分别对问题三中零件次品率、半成品次品率、成品次品率、半成品检测费、成品检测费、半成品拆解费、成品拆解费，共计 7 组变量，以问题三中对应数据为基准，独立地乘以相同的放大倍率 f，每次实验中各组变量内部的不同数据放大倍率相同；

f 按照固定步长递增，记录 7 组变量在不同 f 下的最大利润及对应方案。

实验结果：

各组变量在倍率 f 作用下，最大利润不断变化，决策方案在一定的范围内保持稳定；当 f 达到某一临界值时，决策方案发生改变。节选决策方案发生改变前后两组实验数据，如表 11 所示。绘制不同零部件次品率下的最大利润曲线关系图，如图 6所示。

表 11：决策方案发生改变前后两组实验数据

相对问题3变化参数	f	最大利润	方案变量															
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
零件次品率	0.7	55.00	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1
	0.8	53.56	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
半成品次品率	0.8	52.10	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1
	0.9	51.45	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
成品次品率	0.5	52.62	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1
	0.6	52.27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
半成品检测费	1.5	44.82	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1.7	43.51	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1
成品检测费	3	38.90	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	3.5	37.27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
半成品拆解费	6.5	40.96	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	7.5	39.62	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1
成品拆解费	14.5	37.41	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	16.5	37.27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

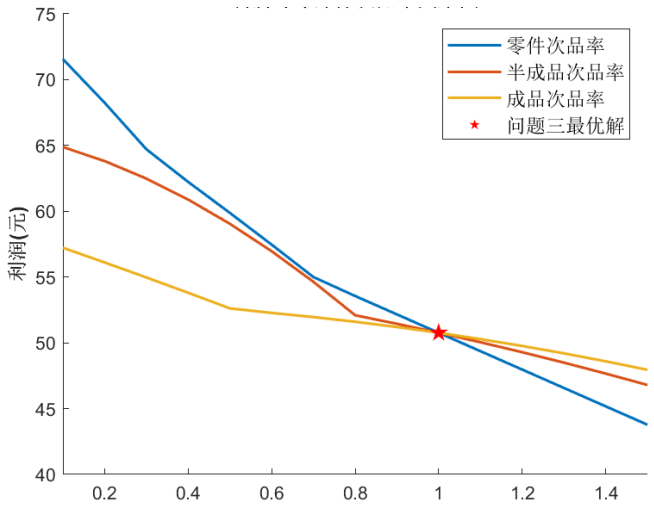

图 6: 企业利润与零配件、半成品及成品关系图

结果分析：

模型对不同环节的次品率、检测费、拆解费等生产数据均具有敏感性：当生产数据分别发生变化时，最优决策方案及利润值均发生改变，模型运算结果很好地反映了不同生产数据对决策方案和收益情况的影响：

a) 零件次品率增加至 0.8 倍时，需增加对部分零件的检测；且由图 6，可知零件次品率对利润的影响在三种次品率中最为显著；

b) 半成品次品率增加至 0.9 倍，或成品次品率增加至 0.6 倍时，需增加对所有半成品的检测；

c) 半成品检测费增加至 1.7 倍时，对半成品不再进行检测；

d) 半成品拆解费增加至 7.5 倍时，对不合格半成品不再拆解；

e) 成品检测费增加至 3.5 倍，或成品拆解费增加至 16.5 倍时，对成品不再检测，且对退回成品不再拆解。

六、 模型的评价与改进

6.1 模型的优缺点分析

我们构建的生产过程决策模型不仅有效提出了最优化决策方案，同时具有以下优点：

- **模型综合考虑多个因素，提出全面且系统的生产过程优化方案**。本文模型结合统计学、运筹学优化理论以及启发式算法，全面考虑了购买、检测、拆解、调换成本及收入等因素，对于企业收益，综合考虑了成品率及其利润，能够应对复杂的生产决策问题。
- **模型具有普适性，实用性较高**。文章提出在生产规模扩大情况下的多阶段整数规划普适公式，并采用遗传算法进行全局优化搜索求解模型，能够适应大规模决策优化问题。并结合贝叶斯后验分布和多阶段鲁棒优化处理次品率的不确定性，提升了模型的鲁棒性。
- **引入零件利用提升率参数，考虑拆解后所带来的额外成本和收益**。成品或半成品进行拆解后，得到的零配件重新回到前一工序进行检测和组装，能够有效提升零件利用率。本文综合考虑了该因素，使得模型更加完善。

但同样我们需要注意到，当前模型仍存在部分提升空间：

- 模型复杂度较高。在生产规模和工序数量扩大的情况下，0-1 整数规划模型和多阶段鲁棒优化模型可能导致计算复杂度高。虽然遗传算法能有效解决大规模问题，但其计算开销和收敛速度可能影响实际应用效果。
- 模型结果对实际数据的依赖性较强，需要较为精确的次品率数据和成本信息。在样本量较小或次品率极端值的情况下，二项分布近似为正态分布可能不够准确，且不同的贝叶斯具体先验选择可能会影响优化决策的效果。

6.2 未来展望与推广

生产过程决策优化在各行业中有着广泛的应用需求。本文所建立的基于多阶段整数规划和鲁棒优化的生产过程决策优化模型能够进一步推广应用于更多领域，如食品加工、机械制造等领域，以验证其在不同生产环境当中的效果。

此外，我们将进一步探索在实时生产数据下，能否对生产决策实现动态调整和实时优化决策方案，并对模型和优化算法进行简化，以提高其计算效率和解决大规模问题的能力，助力模型在未来得到进一步的推广和发展，使其能够信息化地服务于各类生产过程，提升企业的生产效率和经济效益。

参考文献

[1] 亓四华. 制造质量零废品控制理论与技术的研究. PhD thesis, 合肥工业大学, 2001.

[2] 刘喜玲 and 黄程华. 电子产品生产工序质量控制与管理的研究. 科技展望, 26(24):212, 2016.

[3] 丁伟程光, 龚俭. 基于抽样测量的高速网络实时异常检测模型. 软件学报, (03):594–599, 2003.

[4] 同济大学数学系. 概率论与数理统计. 人民邮电出版社, 2017.

[5] 李敏强, 寇纪淞, 林丹, 李书全, et al. 遗传算法的基本理论与应用. Ke xue chu ban she, 2002.

[6] 田俊峰. 不确定性条件下供应链管理优化模型及算法研究. PhD thesis, 西南交通大学, 2005.

[7] 李斯, 周任军, 童小娇, 彭莎, 赵邈, and 姚龙华. 基于盒式集合鲁棒优化的风电并网最大装机容量. 电网技术, 35(12):208–213, 2011.

[8] 杜刚, 赵冬梅, and 刘鑫. 计及风电不确定性优化调度研究综述. 中国电机工程学报, 43(07):2608–2627, 2023.

[9] Mohammad Saber Fallahnezhad and Muhammad Aslam. A new economical design of acceptance sampling models using bayesian inference. *Accreditation and Quality Assurance*, 18:187–195, 2013.

[10] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming*, 88:411–424, 2000.

[11] 维基百科. B 分布, 2023.

附录

1、beta_figure

```
%第四问贝叶斯概率分析图
clear all
x=0.005:0.005:0.995;
[real,low,up]=get_defecrate(x);
figure
hold on
plot(x,real,LineWidth=1.5);
plot(x,low,LineWidth=1.5);
plot(x,up,LineWidth=1.5);
plot(x,x,'--',LineWidth=1);
title('次品率预测值与抽样检测值关系图','FontWeight','bold')
legend('真实值预测','95%下限预测','95%上限预测','测量值','FontSize',11)
xlabel('次品率测量值','FontWeight','bold')
ylabel('次品率预测值','FontWeight','bold')
hold off
```

2、calculate_income_new

```
function income=calculate_income_new(decision)
%输入决策组合矩阵，输出利润结果列向量
%%plus计算未扣除版

%%变量定义
%%一级零件
%%%次品率
part_error=ones(8,1)*0.1;
%%%购入单价
part_price=[2,8,12,2,8,12,8,12];
part_price=part_price';
%%%检测成本
part_test=[1,1,2,1,1,2,1,2];
part_test=part_test';
%%半成品
%%%次品率
half_error=[0.1,0.1,0.1];
half_error=half_error';
%%%装配成本
half_assemble=[8,8,8];
```

```
half_assemble=half_assemble';
%%%检测成本
half_test=[4,4,4];
half_test=half_test';
%%%拆解费用
half_depart=[6,6,6];
half_depart=half_depart';
%%成品数据
all_error=0.1;%e3次品率
all_assemble=8;%装配成本
all_test=6;%检测成本
all_depart=10;%拆解费用
exchange=40;%调换
market_price=200;%售价

%开始计算-------------round 1
%%初始购买成本
initial_buy=sum(part_price);%一个数
%%原始零件检测消耗
part_test_origin_cost=decision(:,1:8)*part_test;%列向量
%%半成品次品率
gg=1-part_error'.*(1-decision(:,1:8));
G=[prod(gg(:,1:3),2),prod(gg(:,4:6),2),prod(gg(:,7:8),2)];
half_E=1-G+G.*(half_error');
% group=[(1-decision(:,1:3))*part_error(1:3),(1-decision(:,4:6))*part_error(4:6)
    ,(1-decision(:,7:8))*part_error(7:8)];
% half_E=half_error' + (1-half_error)'.*group;%每一行代表一种决策下的真实半成品
    次品率
%%半成品原始数量
half_N1=min(1-part_error(1:3,:)'.*decision(:,1:3),[],2);
half_N2=min(1-part_error(4:6,:)'.*decision(:,4:6),[],2);
half_N3=min(1-part_error(7:8,:)'.*decision(:,7:8),[],2);
half_N=[half_N1,half_N2,half_N3];%每一行代表一种决策下的***检测前成形***半成品数
    量
%半成品增益计算
half_depart_N=half_N.*(half_E).*decision(:,12:14);%每一行代表一种决策下的半成品
    拆解数量
%!!!这个应该用修正后的E
tmp1=repmat(half_depart_N(:,1),1,3);
tmp2=repmat(half_depart_N(:,2),1,3);
```

```
tmp3=repmat(half_depart_N(:,3),1,2);
part_N_from_depart=[tmp1,tmp2,tmp3];%每一行代表一种决策半成品拆解所得零件数
delta1=(part_N_from_depart-part_N_from_depart.*(1-decision(:,1:8)).*(part_error
    '))...
    ./(1-decision(:,1:8).*(part_error'));
elpsilon1=[mean(delta1(:,1:3),2),mean(delta1(:,4:6),2),mean(delta1(:,7:8),2)];%
    是一个列向量，每一列是一个方案的半成品增益率
%half_N_plus=(half_N-half_depart_N).*(1+elpsilon.*decision(:,12:14));%每一行代表
    一种决策下的修正半成品数量%!!!!!!!!!!!!!!!!!!
%half_N_plus=(half_N).*(1+elpsilon.*decision(:,12:14));
half_N_plus=(half_N).*(1+elpsilon1.*decision(:,12:14));
%半成品成本
half_cost=half_N_plus*half_assemble+(half_N_plus.*decision(:,9:11))*half_test...
    +half_N_plus.*(half_E).*decision(:,12:14)*half_depart;

%%拆解零件检测消耗
half_is_assemble=[repmat(decision(:,12),1,3),repmat(decision(:,13),1,3),repmat(
    decision(:,14),1,2)];
part_test_plus_cost=(part_N_from_depart.*(1-decision(:,1:8)).*half_is_assemble)*
    part_test;%可能有问题

part_cost=part_test_plus_cost+part_test_origin_cost;

%-----------------round2
%%成品真实次品率
gg1=1-half_E.*(1-decision(:,9:11));
G1=prod(gg1,2);
all_E=1-G1+G1.*all_error;
%all_E=all_error + (1-all_error)*sum((1-decision(:,9:11)).*half_E,2);%一个列向量
%%成品原始数量
all_N=min(half_N-half_N.*half_E.*decision(:,9:11),[],2).*mean(1+elpsilon1.*
    decision(:,12:14),2);%一个列向量
%%成品增益
all_depart_N=all_N.*all_E.*decision(:,16);%一个列向量
half_N_from_depart=repmat(all_depart_N,1,3);
delta2=(half_N_from_depart-half_N_from_depart.*(1-decision(:,9:11)).*half_E)
    ./...
    (half_N-half_N.*half_E.*decision(:,9:11));
elpsilon2=mean(delta2,2);
%all_N_plus=(all_N-all_depart_N).*(1+elpsilon1.*decision(:,16));%!!!!!!!!!
```

```
%all_N_plus=(all_N).*(1+elpsilon1.*decision(:,16));
all_N_plus=(all_N).*(1+elpsilon2.*decision(:,16));
%%成品成本
all_cost=all_N_plus*all_assemble+(all_N_plus.*decision(:,15))*all_test...
    +all_N_plus.*all_E.*decision(:,16)*all_depart;
%%拆解成品检测消耗
half_test_plus_cost=(half_N_from_depart.*(1-decision(:,9:11).*repmat(decision
    (:,16),1,3)))*half_test;
half_cost=half_cost+half_test_plus_cost;

%第三轮？
%%!!!!!!!!!!!!!!!!!!!!!!!!
end_N=min(all_N-all_N.*all_E.*decision(:,15),[],2).*mean(1+elpsilon2.*decision
    (:,16).*decision(:,15),2);

%%调换和收入
exchange_cost=all_E.*(1-decision(:,15)).*end_N.*(exchange+market_price);
sell=market_price*end_N;

%汇总
income=sell-initial_buy-part_cost-half_cost-all_cost-exchange_cost;
end
```

3、calculate_income2

```
function income=calculate_income2(decision,case_num)
%输入决策组合矩阵，输出利润结果列向量

%%变量定义
%%%次品率************************************
part_error0=[0.1,0.1; 0.2,0.2; 0.1,0.1; 0.2,0.2; 0.1,0.2; 0.05,0.05];
[part_error,~,~]=get_defecrate(part_error0);
%%%购入单价
part_price=[4,18];
%%%检测成本
part_test=[2,3; 2,3; 2,3; 1,1; 8,1; 2,3];
%%%成品数据
all_error0=[0.1;0.2;0.1;0.2;0.1;0.05];%e3次品率**********************
[all_error,~,~]=get_defecrate(all_error0);
all_assemble_price=6;%装配成本
all_test=[3;3;3;2;2;3];%检测成本
```

```
all_price=56;%售价
%%%不合格产品
exchange=[6;6;30;30;10;10];%调换
depart=[5;5;5;5;5;40];%拆解花费

%开始计算
e3=(all_error(case_num)-1)*(1-part_error(case_num,1)*(1-decision(:,1)))...
    .*(1-part_error(case_num,2)*(1-decision(:,2)))+1;
% e3=all_error(case_num)+(1-all_error(case_num))*...
%     (part_error(case_num,1)*(1-decision(:,1))+part_error(case_num,2)*(1-
    decision(:,2)));
%初始购买
    initial_buy=sum(part_price);

%检测消耗
parttest_cost=sum(part_test(case_num,:).*decision(:,1:2),2);
    n3=min(1-part_error(case_num,:).*decision(:,1:2),[],2);
    %recycletest_cost=sum(part_test(case_num,:)).*e3.*decision(:,4).*n3.*(1-
    decision(:,1));%回收部分的零件检测
    recycletest_cost=sum(part_test(case_num,:).*(1-decision(:,1:2)),2).*e3.*
    decision(:,4).*n3;%回收部分的零件检测
    %新n3计算
    delta1=(e3.*n3-e3.*n3.*(1-decision(:,1))*part_error(case_num,1))./(1-
    part_error(case_num,1)*decision(:,1));
    delta2=(e3.*n3-e3.*n3.*(1-decision(:,2))*part_error(case_num,2))./(1-
    part_error(case_num,2)*decision(:,2));
    n3_new=n3.*(1+decision(:,4).*(delta1+delta2)/2);

    alltest_cost=all_test(case_num)*decision(:,3).*n3_new;%成品检测
    test_cost=alltest_cost+parttest_cost+recycletest_cost;%检测总消耗

%装配成本
    assemble_cost=all_assemble_price*n3_new;
%拆解费用
    depart_cost=depart(case_num)*e3.*n3.*decision(:,4);
%调换费用
    %exchange_cost=e3.*n3_new.*(1-decision(:,3))*(all_price+exchange(case_num)+
    sum(part_price)+all_assemble_price);
    exchange_cost=e3.*n3_new.*(1-decision(:,3))*(all_price+exchange(case_num));
%销售额
```

```
    sell=all_price*(n3_new-e3.*n3_new.*decision(:,3));

%汇总
income=sell-initial_buy-test_cost-assemble_cost-depart_cost-exchange_cost;
end
```

4、calculate_income3

```
function income=calculate_income3(decision)
%输入决策组合矩阵，输出利润结果列向量
%%plus计算未扣除版
%%变量定义
%%一级零件
%%%次品率
part_error0=ones(8,1)*0.1;
[~,~,part_error]=get_defecrate(part_error0);
%%%购入单价
part_price=[2,8,12,2,8,12,8,12];
part_price=part_price';
%%%检测成本
part_test=[1,1,2,1,1,2,1,2];
part_test=part_test';
%%半成品
%%%次品率
half_error0=[0.1,0.1,0.1];
[~,~,half_error]=get_defecrate(half_error0);
half_error=half_error';
%%%装配成本
half_assemble=[8,8,8];
half_assemble=half_assemble';
%%%检测成本
half_test=[4,4,4];
half_test=half_test';
%%%拆解费用
half_depart=[6,6,6];
half_depart=half_depart';
%%成品数据
all_error0=0.1;%e3次品率
[~,~,all_error]=get_defecrate(all_error0);
all_assemble=8;%装配成本
all_test=6;%检测成本
```

```
all_depart=10;%拆解费用
exchange=40;%调换
market_price=200;%售价

%开始计算-------------round 1
%开始计算-------------round 1
%%初始购买成本
initial_buy=sum(part_price);%一个数
%%原始零件检测消耗
part_test_origin_cost=decision(:,1:8)*part_test;%列向量
%%半成品次品率
gg=1-part_error'.*(1-decision(:,1:8));
G=[prod(gg(:,1:3),2),prod(gg(:,4:6),2),prod(gg(:,7:8),2)];
half_E=1-G+G.*(half_error');
% group=[(1-decision(:,1:3))*part_error(1:3),(1-decision(:,4:6))*part_error(4:6)
    ,(1-decision(:,7:8))*part_error(7:8)];
% half_E=half_error' + (1-half_error)'.*group;%每一行代表一种决策下的真实半成品
    次品率
%%半成品原始数量
half_N1=min(1-part_error(1:3,:)'.*decision(:,1:3),[],2);
half_N2=min(1-part_error(4:6,:)'.*decision(:,4:6),[],2);
half_N3=min(1-part_error(7:8,:)'.*decision(:,7:8),[],2);
half_N=[half_N1,half_N2,half_N3];%每一行代表一种决策下的***检测前成形***半成品数
    量
%半成品增益计算
half_depart_N=half_N.*(half_E).*decision(:,12:14);%每一行代表一种决策下的半成品
    拆解数量
%!!!这个应该用修正后的E
tmp1=repmat(half_depart_N(:,1),1,3);
tmp2=repmat(half_depart_N(:,2),1,3);
tmp3=repmat(half_depart_N(:,3),1,2);
part_N_from_depart=[tmp1,tmp2,tmp3];%每一行代表一种决策半成品拆解所得零件数
delta1=(part_N_from_depart-part_N_from_depart.*(1-decision(:,1:8)).*(part_error
    ')).*...
     ./(1-decision(:,1:8).*(part_error'));
elpsilon1=[mean(delta1(:,1:3),2),mean(delta1(:,4:6),2),mean(delta1(:,7:8),2)];%
    是一个列向量，每一列是一个方案的半成品增益率
%half_N_plus=(half_N-half_depart_N).*(1+elpsilon.*decision(:,12:14));%每一行代表
    一种决策下的修正半成品数量%!!!!!!!!!!!!!!!!!!!
%half_N_plus=(half_N).*(1+elpsilon.*decision(:,12:14));
```

```
half_N_plus=(half_N).*(1+elpsilon1.*decision(:,12:14));
%半成品成本
half_cost=half_N_plus*half_assemble+(half_N_plus.*decision(:,9:11))*half_test...
    +half_N_plus.*(half_E).*decision(:,12:14)*half_depart;

%%拆解零件检测消耗
half_is_assemble=[repmat(decision(:,12),1,3),repmat(decision(:,13),1,3),repmat(
    decision(:,14),1,2)];
part_test_plus_cost=(part_N_from_depart.*(1-decision(:,1:8)).*half_is_assemble)*
    part_test;%可能有问题

part_cost=part_test_plus_cost+part_test_origin_cost;

%-----------------round2
%%成品真实次品率
gg1=1-half_E.*(1-decision(:,9:11));
G1=prod(gg1,2);
all_E=1-G1+G1.*all_error;
%all_E=all_error + (1-all_error)*sum((1-decision(:,9:11)).*half_E,2);%一个列向量
%%成品原始数量
all_N=min(half_N-half_N.*half_E.*decision(:,9:11),[],2).*mean(1+elpsilon1.*
    decision(:,12:14),2);%一个列向量
%%成品增益
all_depart_N=all_N.*all_E.*decision(:,16);%一个列向量
half_N_from_depart=repmat(all_depart_N,1,3);
delta2=(half_N_from_depart-half_N_from_depart.*(1-decision(:,9:11)).*half_E)
    ./...
    (half_N-half_N.*half_E.*decision(:,9:11));
elpsilon2=mean(delta2,2);
%all_N_plus=(all_N-all_depart_N).*(1+elpsilon1.*decision(:,16));%!!!!!!!!!
%all_N_plus=(all_N).*(1+elpsilon1.*decision(:,16));
all_N_plus=(all_N).*(1+elpsilon2.*decision(:,16));
%%成品成本
all_cost=all_N_plus*all_assemble+(all_N_plus.*decision(:,15))*all_test...
    +all_N_plus.*all_E.*decision(:,16)*all_depart;
%%拆解成品检测消耗
half_test_plus_cost=(half_N_from_depart.*(1-decision(:,9:11).*repmat(decision
    (:,16),1,3)))*half_test;
half_cost=half_cost+half_test_plus_cost;
```

```
%第三轮？
%%!!!!!!!!!!!!!!!!!!!!!!!!
end_N=min(all_N-all_N.*all_E.*decision(:,15),[],2).*mean(1+elpsilon2.*decision
    (:,16).*decision(:,15),2);

%%调换和收入
exchange_cost=all_E.*(1-decision(:,15)).*end_N.*(exchange+market_price);
sell=market_price*end_N;

%汇总
income=sell-initial_buy-part_cost-half_cost-all_cost-exchange_cost;
end
```

5、get_defecrate

```
function [realRate,lowerRate,upperRate] = get_defecrate(ptest)
%UNTITLED3 此处显示有关此函数的摘要
%    此处显示详细说明
k=get_sample_num(ptest);
beta_alfa=2;
beta_beta=5;
num1=beta_alfa+ptest.*k;
num2=beta_beta+(1-ptest).*k;
realRate=num1./(beta_alfa+beta_beta+k);
lowerRate=betainv(0.025,num1,num2);
upperRate=betainv(0.975,num1,num2);
end
```

6、get_sample_num

```
function n95= get_sample_num(p0)
%UNTITLED2 此处显示有关此函数的摘要
%    p0标称值
E=0.05;
Z95=1.645;
n95=ceil(Z95^2.*(p0.*(1-p0))./E.^2);
end
```

7、Q_1

```
%第一问求解代码
p0=0.1;%标称值
E=0.01:0.005:0.05;
```

```
Esmooth=0.01:0.0001:0.05;
Z95=1.645;
Z90=1.280;
n95=ceil(Z95^2*(p0*(1-p0))./E.^2);
n95smooth=ceil(Z95^2*(p0*(1-p0))./Esmooth.^2);
n90=ceil(Z90^2*(p0*(1-p0))./E.^2);
n90smooth=ceil(Z90^2*(p0*(1-p0))./Esmooth.^2);
figure
hold on
plot(Esmooth,n95smooth,LineWidth=1.5);
plot(Esmooth,n90smooth,LineWidth=1.5);
scatter(E,n95,40,[0,0.447,0.741],'filled')
scatter(E,n90,40,[0.85,0.325,0.098],'filled')
legend({'情形(1)最小样本容量','情形(2)最小样本容量'},'FontSize',11);
title('不同允许次品率误差E下最小样本容量','FontWeight','bold')
xlabel('次品率误差E','FontWeight','bold')
ylabel('最小样本容量','FontWeight','bold')

hold off
```

8、Q_2

```
%问题2求解
clear all
%%所有决策变量组合的确定，每一行代表一种决策组合
varialbe_num=4;%决策变量数
decide=dec2bin(0:2^varialbe_num-1)-'0'; %所用决策组合
decide_num=size(decide,1);
complete_data=zeros(decide_num,6);
%利润计算函数定义
function income=calculate_income(decision,case_num)
%输入决策组合矩阵，输出利润结果列向量

%%变量定义
%%%次品率
part_error=[0.1,0.1; 0.2,0.2; 0.1,0.1; 0.2,0.2; 0.1,0.2; 0.05,0.05];
%%%购入单价
part_price=[4,18];
%%%检测成本
part_test=[2,3; 2,3; 2,3; 1,1; 8,1; 2,3];
%%%成品数据
```

```
all_error=[0.1;0.2;0.1;0.2;0.1;0.05];%e3次品率
all_assemble_price=6;%装配成本
all_test=[3;3;3;2;2;3];%检测成本
all_price=56;%售价
%%%不合格产品
exchange=[6;6;30;30;10;10];%调换
depart=[5;5;5;5;5;40];%拆解花费

%开始计算
e3=(all_error(case_num)-1)*(1-part_error(case_num,1)*(1-decision(:,1)))...
    .*(1-part_error(case_num,2)*(1-decision(:,2)))+1;
% e3=all_error(case_num)+(1-all_error(case_num))*...
%     (part_error(case_num,1)*(1-decision(:,1))+part_error(case_num,2)*(1-
    decision(:,2)));
%初始购买
    initial_buy=sum(part_price);

%检测消耗
parttest_cost=sum(part_test(case_num,:).*decision(:,1:2),2);
    n3=min(1-part_error(case_num,:).*decision(:,1:2),[],2);
    %recycletest_cost=sum(part_test(case_num,:)).*e3.*decision(:,4).*n3.*(1-
    decision(:,1));%回收部分的零件检测
    recycletest_cost=sum(part_test(case_num,:).*(1-decision(:,1:2)),2).*e3.*
    decision(:,4).*n3;%回收部分的零件检测
    %新n3计算
    delta1=(e3.*n3-e3.*n3.*(1-decision(:,1))*part_error(case_num,1))./(1-
    part_error(case_num,1)*decision(:,1));
    delta2=(e3.*n3-e3.*n3.*(1-decision(:,2))*part_error(case_num,2))./(1-
    part_error(case_num,2)*decision(:,2));
    n3_new=n3.*(1+decision(:,4).*(delta1+delta2)/2);

    alltest_cost=all_test(case_num)*decision(:,3).*n3_new;%成品检测
    test_cost=alltest_cost+parttest_cost+recycletest_cost;%检测总消耗

%装配成本
    assemble_cost=all_assemble_price*n3_new;
%拆解费用
    depart_cost=depart(case_num)*e3.*n3.*decision(:,4);
%调换费用
    %exchange_cost=e3.*n3_new.*(1-decision(:,3))*(all_price+exchange(case_num)+
```

```
    sum(part_price)+all_assemble_price);
    exchange_cost=e3.*n3_new.*(1-decision(:,3))*(all_price+exchange(case_num));
%销售额
    sell=all_price*(n3_new-e3.*n3_new.*decision(:,3));

%汇总
income=sell-initial_buy-test_cost-assemble_cost-depart_cost-exchange_cost;
end

for ii=1:6
    result=calculate_income(decide,ii);%收益计算结果
    complete_data(:,ii)=result;
    [max_income,method_num]=max(result);%找到最大收益和对应方案编号
    max_method=decide(method_num,:);%最大方案

    disp('当前情形：')
    disp(ii)
    disp('单件最大利润为：')
    disp(max_income);
    disp('决策方案为：')
    disp(max_method);
end
```

9、Q_3_ga_new

```
%第三问遗传算法求解
function [best_decision, max_income] = optimize_income()

    % 决策变量个数，每一行是16个决策变量
    num_vars = 16;

    options = optimoptions('ga', 'PopulationSize', 100, 'MaxGenerations', 20,
    ...
        'Display', 'iter', 'UseParallel', false, 'FunctionTolerance', 1e-6);

    [best_decision, max_income] = ga(@calculate_income_ga, num_vars, [], [], [],
    [], ...
        zeros(1, num_vars), ones(1, num_vars), [], 1:num_vars, options);

    % 打印结果
    disp('最优决策方案为:');
```

```
    disp(best_decision);
    disp(['对应的最大利润为: ', num2str(-max_income)]);
end

% 目标函数，遗传算法会最小化目标函数值，所以取负数
function income = calculate_income_ga(decision)
    % 遗传算法优化器返回的是一个行向量，将其调整为适合 calculate_income 的矩阵
    decision = round(decision); % 将浮点数四舍五入为0或1，确保是0/1决策
    decision = reshape(decision, 1, []); % 将决策调整为1行

    % 调用原来的利润计算函数
    income = -calculate_income_new(decision); % 求最大化利润，使用负数以适应遗传
    算法的最小化机制
end

optimize_income();
```

10、Q_3_main_new

```
%第三问求解
clear all
varialbe_num=16;%决策变量数
decide=dec2bin(0:2^varialbe_num-1)-'0'; %所用决策组合
%1-8:零件是否检测
%9-11：半成品是否检测
%12-14：半成品是否拆解
%15：成品是否检测
%16：成品是否拆解

result=calculate_income_new(decide);
[max_income,method_num]=max(result);%找到最大收益和对应方案编号
max_method=decide(method_num,:);%最大方案
disp('单件最大利润为：')
disp(max_income);
disp('决策方案为：')
disp(max_method);
```

11、Q_4_Q2main

```
%问题四-重解第二问
%问题2求解
clear all
```

```
%%所有决策变量组合的确定，每一行代表一种决策组合
varialbe_num=4;%决策变量数
decide=dec2bin(0:2^varialbe_num-1)-'0'; %所用决策组合
decide_num=size(decide,1);
complete_data=zeros(decide_num,6);
%利润计算函数定义


for ii=1:6
    result=calculate_income2(decide,ii);%收益计算结果
    complete_data(:,ii)=result;
    [max_income,method_num]=max(result);%找到最大收益和对应方案编号
    max_method=decide(method_num,:);%最大方案

    disp('当前情形：')
    disp(ii)
    disp('单件最大利润为：')
    disp(max_income);
    disp('决策方案为：')
    disp(max_method);
end
```

12、Q4_Q3_ga_new

```
%第三问遗传算法求解
function [best_decision, max_income] = optimize_income()

    % 决策变量个数，每一行是16个决策变量
    num_vars = 16;

    options = optimoptions('ga', 'PopulationSize', 100, 'MaxGenerations', 20,
   ...
        'Display', 'iter', 'UseParallel', false, 'FunctionTolerance', 1e-6);

    [best_decision, max_income] = ga(@calculate_income_ga, num_vars, [], [], [],
    [], ...
        zeros(1, num_vars), ones(1, num_vars), [], 1:num_vars, options);

    % 打印结果
    disp('最优决策方案为:');
    disp(best_decision);
```

```
    disp(['对应的最大利润为: ', num2str(-max_income)]);
end

% 目标函数，遗传算法会最小化目标函数值，所以取负数
function income = calculate_income_ga(decision)
    % 遗传算法优化器返回的是一个行向量，将其调整为适合 calculate_income 的矩阵
    decision = round(decision); % 将浮点数四舍五入为0或1，确保是0/1决策
    decision = reshape(decision, 1, []); % 将决策调整为1行

    % 调用原来的利润计算函数
    income = -calculate_income3(decision); % 求最大化利润，使用负数以适应遗传算
    法的最小化机制
end

optimize_income();
```

13、Q4_Q3main

```
%问题四-重解第三问
%第三问求解
clear all
varialbe_num=16;%决策变量数
decide=dec2bin(0:2^varialbe_num-1)-'0'; %所用决策组合
%1-8:零件是否检测
%9-11：半成品是否检测
%12-14：半成品是否拆解
%15：成品是否检测
%16：成品是否拆解

result=calculate_income3(decide);
[max_income,method_num]=max(result);%找到最大收益和对应方案编号
max_method=decide(method_num,:);%最大方案
disp('单件最大利润为：')
disp(max_income);
disp('决策方案为：')
disp(max_method);
```