# 企业抽样检验与决策优化

# 摘要

随着电子制造业和自动化生产的发展,质量检测和产能优化问题成为工业生产中的重要研究课题。本文针对电子成品生产过程中质量检测和产能优化的问题，基于遗传算法和统计学思想,通过确定检测成本、装配成本、调换损失和拆解费用等指标，以最大化利润为目标建立了阶段质量检测优化模型，并使用遗传算法对模型进行求解，做出合理决策策略，以提高产品质量和经济效益。

**针对问题一，**针对供应商提供的零配件次品率声明，本文使用计数序贯抽样检验方法，计算了不同置信度下的最小样本量。并要求在一定信度下，如果检测结果表明零配件次品率超过标称值，企业将拒收该批零配件如果检测结果表明次品率不超过标称值，则企业将接收该批零配件。通过此标准，为后续决策提供数据支撑。

**针对问题二，**针对已知零配件和成品次品率的情况下，本文为企业生产过程的各个阶段提供了决策方案。且考虑了零配件检测、成品检测、不合格成品处理以及不合格品调换损失等多个因素。通过成本效益分析，我们为企业在不同情况下提供了最优的决策方案，包括是否进行检测、是否拆解不合格成品等，以最大化企业的利润。

**针对问题三，**针对多道工序和多个零配件的生产环境中，我们扩展了问题二的模型，考虑了更复杂的生产流程。我们为每个生产阶段提供了决策方案，包括零配件的检测、半成品的装配、成品的检测与市场投放，以及不合格成品的处理。这些决策方案帮助企业在保证产品质量的同时，优化生产成本。

**针对问题四，**我们考虑了抽样检测结果的不确定性对决策的影响。通过敏感性分析，我们评估了次品率估计误差对决策方案的影响，并据此调整了决策方案。这确保了即使在次品率估计存在误差的情况下，企业仍能做出有效的决策。

**关键词：**计数序贯抽样、次品率、工业生产、生产决策、遗传算法

# 问题重述

## 问题背景

本题聚焦于一个电子产品制造企业面临的生产决策问题。该企业生产一种市场需求量较大的电子产品，其生产过程依赖于两种关键零配件。这些零配件的质量直接决定了最终产品的性能和可靠性。在装配过程中，任何一个零配件的不合格都会导致整个产品的不合格，而即使两个零配件都合格，最终产品也可能因为其他因素而不达标。而企业在生产过程中需要做出多个关键决策，包括是否对零配件进行检测、如何处理不合格的成品、以及如何管理供应链中的风险。这些决策涉及到成本和效益的权衡，如购买成本、检测成本、装配成本、市场售价、调换损失和拆解费用等。企业需要在保证产品质量的同时，尽可能地降低成本和提高经济效益。

## 问题提出

**问题一：**已知次品率标称值为10%，采用抽样检测方法为在90%、95%两种不同置信度下决定是否接收从供应商购买的这批零配件，以减少不合格零配件对工业生产的影响，并为企业设计检测次数尽可能少的抽样检测方案。

**问题二：**已知两种零配件和成品的次品率的情况下，企业需要对生产过程的各个阶段作出决策。这包括是否对零配件进行检测，是否对装配好的成品进行检测，对检测出的不合格成品是否进行拆解，以及对用户购买的不合格品是否进行调换。并且需要根据这些决策，根据表格数据给出具体的决策方案，并提供决策的依据及相应的指标结果。

**问题三：**对于多道工序和多个零配件的生产过程，要求重复问题2的分析，给出生产过程的决策方案。已知零配件、半成品和成品的次品率，需要考虑这些信息，提出具体的决策方案和决策依据。

**问题四：**假设问题2和问题3中的零配件、半成品和成品的次品率是通过抽样检测方法得到的，需要重新审视这些决策方案。分析抽样检测结果的不确定性对决策的影响，并据此调整决策方案。

# 问题分析

**问题一：**题目要求设计一个抽样检测方案，以确定是否接收供应商提供的零配件。在这种情况下，序贯抽样是一种有效的策略，因为它允许在达到明确的决策标准之前连续进行抽样，从而减少所需的样本量。为解决这一问题，设定决策边界，即在累积到一定数量的样本后，根据观察到的次品率与标称值的比较结果，决定是否接收整批零配件。其次通过序贯抽样方案，计算在给定的置信度和错误接受率下所需的平均样本量，以最小化检测成本。最后通过模拟不同的抽样场景，验证序贯抽样方案的有效性，并调整参数以满足企业的具体需求。

**问题二：**题目要求在已知零配件和成品次品率的情况下，为生产过程的各个阶段制定决策方案。在此情况下，遗传算法是一种适用于解决此类优化问题的启发式搜索算法，它模仿自然选择的过程来迭代地改进解决方案。在本问题中，我们可以规定适应度函数可以定义为最大化利润或最小化成本，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，将决策变量（如是否检测零配件、是否拆解不合格成品等）编码为遗传算法可以处理的形式，如二进制字符串，然后实现遗传算法，包括初始化种群、选择操作、交叉和变异操作、适应度评估和新一代种群的生成，最后分析遗传算法的输出，确定最优的生产策略，并评估其在不同生产情况下的表现和稳健性。

**问题三：**题目要求在多道工序和多个零配件的生产环境中，制定决策方案。这涉及到更复杂的生产流程和更多的决策变量。遗传算法同样适用于这种复杂问题的优化。我们可以将生产过程分解为多个阶段，每个阶段都有其决策变量，如是否检测、是否拆解等，以此为基础扩展适应度函数以包括多阶段生产的所有成本和收益，确保算法能够全面评估每个决策方案的总体效益，并调整遗传算法的参数，如种群大小、交叉率和变异率，以适应更复杂的决策空间，并提高算法的搜索效率和解的质量，最后考虑可能需要进行多目标优化，比如同时最大化利润和最小化不合格品率，这可以通过设计多目标遗传算法来实现。

**问题四：**题目要求在考虑次品率数据不确定性的情况下，重新评估和调整决策方案。我们可以将次品率的不确定性纳入模型中，可能通过引入随机变量或模糊参数来模拟次品率的变化，在此基础上设计适应度函数以评估决策方案在不同次品率情况下的表现，优先选择那些在不确定性下表现稳定的方案。和前两题一样通过遗传算法搜索在各种可能的次品率情况下都能表现良好的生产策略，以协助工厂调整决策方案。

# 模型假设与符号说明

## 模型基本假设

1. 零配件质量一致性：假设供应商提供的每种零配件的次品率是恒定的，并且在整个生产周期内保持不变。
2. 检测准确性：假设检测过程是完全准确的，即检测结果能100%正确反映零配件的真实质量状态。
3. 成本价格稳定性：假设在考虑的时间段内，零件的购买成本、检测成本、装配成本、拆解费用、调换损失以及成品的市场售价都是固定不变的

## 符号说明

符号说明

|  |  |  |
| --- | --- | --- |
| **符号** | **含义** | **单位** |
| di | 零件i的数量 | 件 |
| μ | 成品数量 | 件 |
| latexmath | 成品的正品数量 | 件 |
| xi | 零件i是否检测 | — |
| X | 成品是否检测 | — |
| Ci | 零件i购买成本 | 元/件 |
| qi | 零件i的次品率 | — |
| Q | 成品次品率 | — |
| Chi | 零件i的检测成本 | 元/件 |
| Cg | 调换损失 | 元/件 |
| Cc | 拆解费用 | 元/件 |
| Cl | 装配成本 | 元/件 |
| sale | 成品售价 | 元/件 |
| Ce | 成品的检测成本 | 元/件 |

# 模型建立与求解

## 问题一模型建立与求解

### 问题一求解思路

问题一要求我们为企业设计一个抽样检测方案来决定是否从供应商购买一批零件。由于检测存在检测成本故该问题的核心是如何在保证检测准确的情况下尽可能减少样本容量来减少检验成本。在该问题中影响抽样数量的关键因素包括供应商声称的零件的标称值，企业所要求的置信水平，以及决策风险。

因此该问题可以使用假设检验的方法来解决，可以把供应商声称的标称值作为原假设，通过抽样检测的方式来判断是否有足够的证据来拒绝该假设。

考虑到在实际情况下供应商提供的零件次品率并不固定，因此固定样本量抽样的方法会导致无法动态考虑到实际次品率不同而造成浪费。因此我们考虑采用一种自适应的抽样检验方法序贯抽样检验来解决此问题。该方法可以在抽样过程中动态的决定是否继续进行抽样，而不是给定一个固定的样本量。该方法可以保证在所需的置信水平下平均而言减少所需样本量。

在建立该问题的求解模型时需要考虑两个重要的参数即：可接受合格质量水平，是抽样检测中对应一个可接受的次品率；极限质量水平，是抽样检测中对应较低接受概率的被认为不能接受的次品率。此外还需要考虑两类错误的风险：第一类错误(误拒)和第二类错误(误收)的概率。

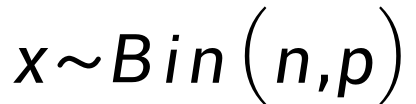
### 问题一模型建立

基于以上思路我们可以建立一个序贯概率比检验模型来解决这个问题，该模型的核心思想就是在每次抽样过后判断该累计结果在批不合格率为合格质量时出现的概率大还是在极限质量时出现的概率大，从而判断是否继续进行抽样或者做出接收/拒绝的决定。

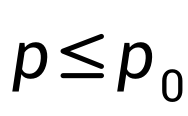
###### 二项分布模型建立

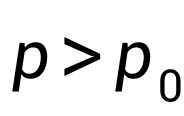
基于以上思路对零件进行抽样检测，其产品的抽样检测结果应符合二项分布。

设n为样本量，x为样本中次品的个数，则x服从参数为n和p的二项分布即：



###### 建立假设和检验目标

原假设H0：次品率 

备选假设H1：次品率

检验目标：在95%（情形一）或者90%（情形二）的置信水平下，尽可能少的样本量判断零配件的次品率是否超过标称值，从而决定是否拒收该批零配件。

###### 序贯检验参数设定

我们定义以下参数：

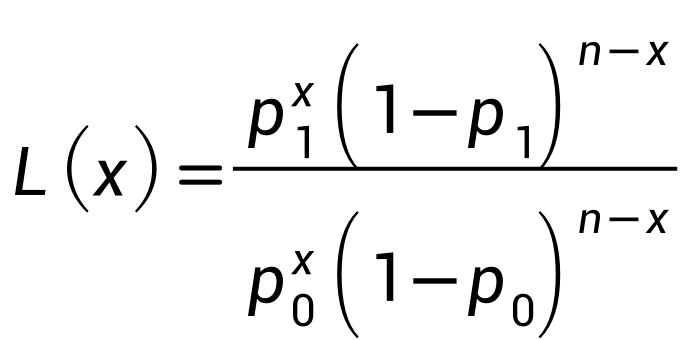
1. p0：可接受合格质量水平，设定为供应商声称的标称值。
2. p1：极限质量水平，可以根据实际需求设定。
3. α：第一类错误（误拒）的概率。
4. β：第二类错误（误收）的概率。

###### 定义似然比（概率比）函数

·可接受合格质量水平p0 = 0.01，即供应商声称的标称值。

·备选假设次品率p1 = 0.15，取略大于标称值的次品率用来代表极限质量水平。

则似然比函数L(x)为：

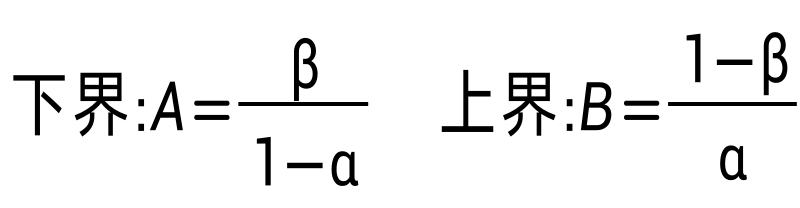


其中n为当前的样本量，x为目前观察到的次品数。

###### 定义判断界限

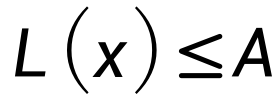
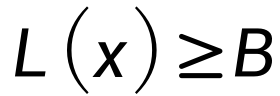
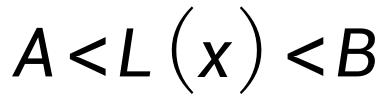
由序贯抽样的核心思想可知，需要设定似然比函数的判断界限来做出是否继续进行抽样或者接收/拒收的决定。因此判断上界和下界的选取决定了抽样数量的大小。

基于瓦尔特(Wald)的证明

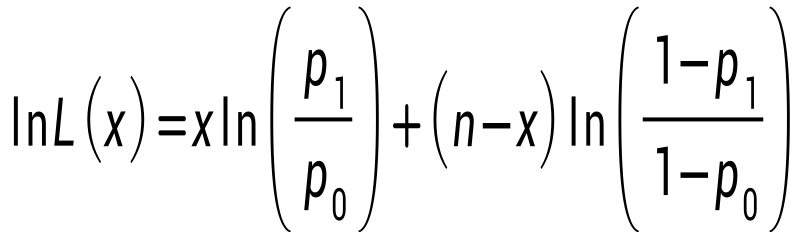


在下界与上界取A和B时，在一切其犯2类错误的概率分布不超过α和β的检验类中，以序贯抽样检测的平均抽样样品个数最少。

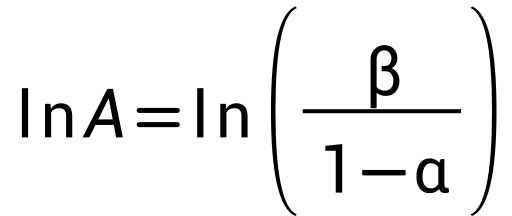
###### 序贯抽样的决策规则及计算简化

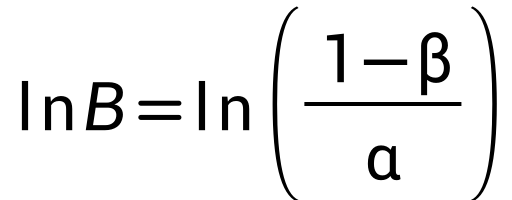
1. 如果 ，则接收这批零件。
2. 如果 ，则拒收这批零件。
3. 如果，则继续抽样。

为了简化计算，我们可以对释然比函数取对数，得到：



相应的，判断界限变为：

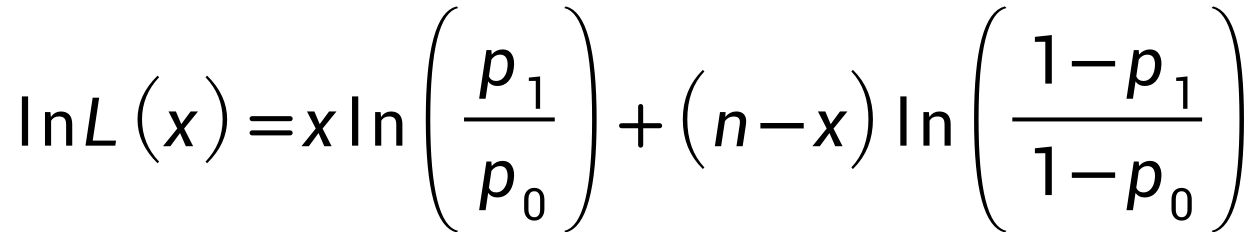




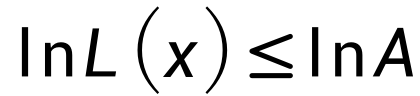
### 问题一模型求解与分析

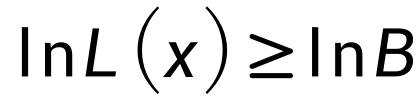
###### 基于序贯抽样检测的自适应动态抽样算法步骤

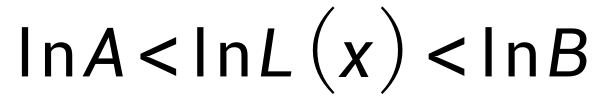
1. 初始化参数：设定p0 、p1、α、β，同时计算lnA和lnB的值。
2. 开始抽样：设定起始样本量n0，并从批次中随机抽取该样本量进行检测。
3. 更新相应变量：更新当前样本量n0以及次品数x。
4. 计算似然比函数：



1. 根据当前似然比数值做出决定：

如果，则接收这批零部件。

如果，则拒收这批零部件。

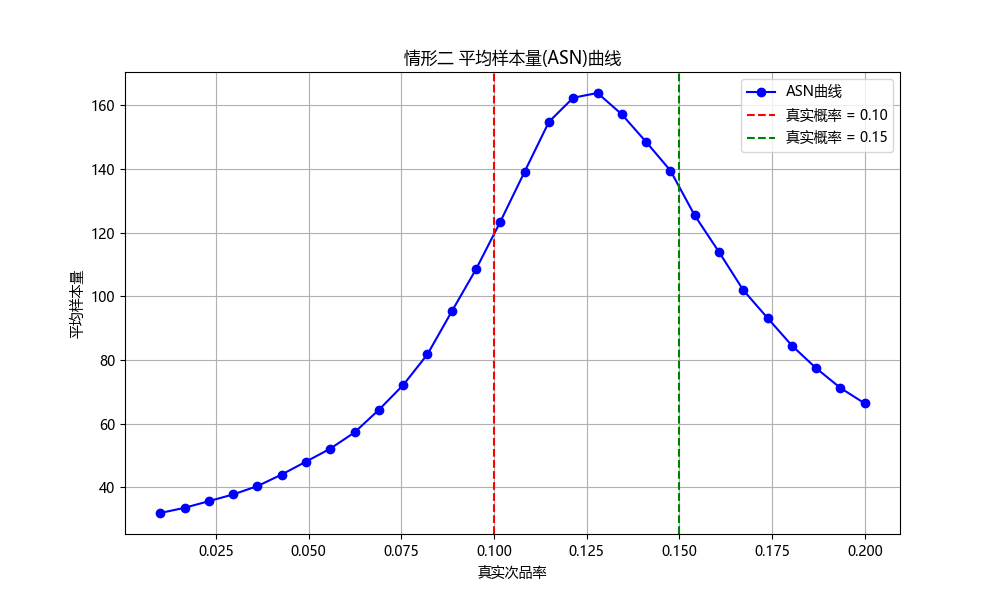
如果，则继续抽样。

1. 重复步骤2-5，直到做出最终决定。

###### 问题一模型的求解

由于该模型的样本数量为自适应动态变化，所以样本数量的大小于实际情况有关。我们可以使用蒙洛卡特模拟对不同实际次品率下的平均检验件数(ASN)。对于情况一和情况二有以下结果：

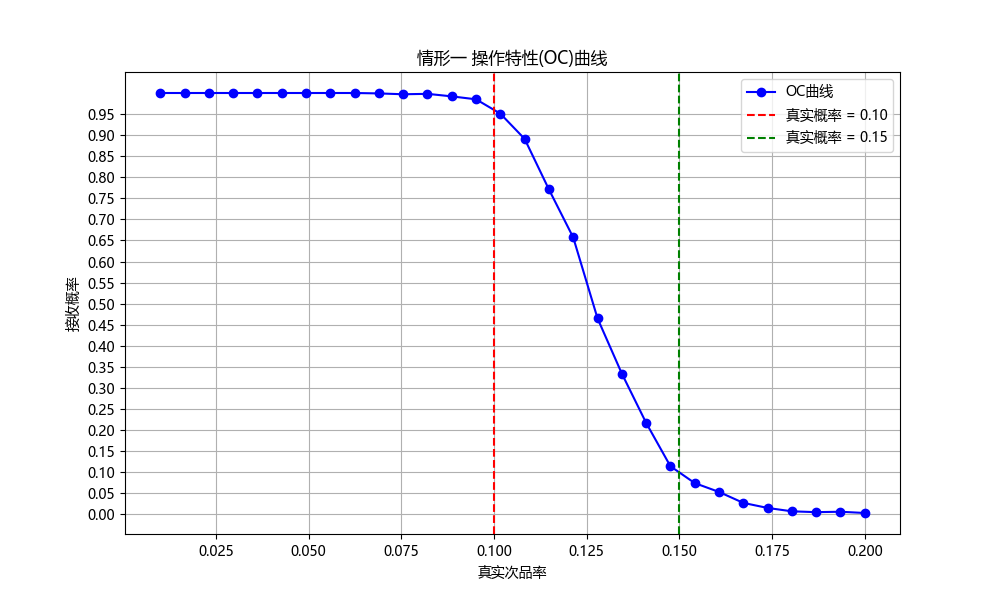
###### 

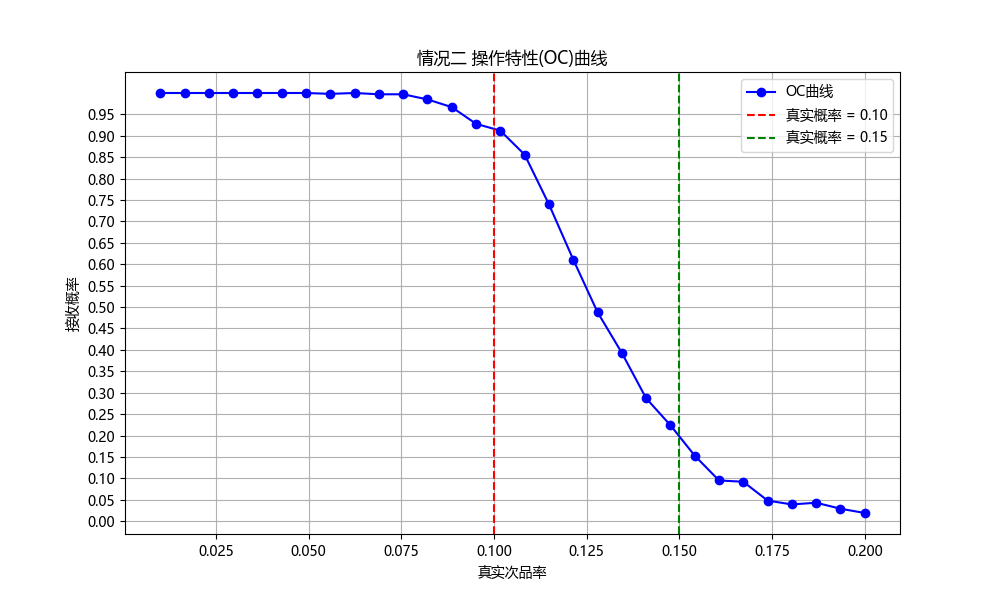


###### 问题一的求解结果分析

下面进行问题一的结果分析和性能评价：

OC曲线分析：OC曲线显示了在不同真实次品率的情况下接收该批次的概率。如图可见，当次品率接近或低于p0(0.10)时，接收的概率非常高；但是当次品率接近或高于p1(0.15)时，接收的概率急剧下降并且在高于p1(0.15)时接收的概率很低。





性能指标：在p0(0.10)处接收的概率应接近于1-α(0.95)/(0.90)。在p1(0.15)处接收的概率应接近于β(0.10)/(0.20)。实际情况与这些值的近似程度反映了模型的准确性。

检测结果作用：接收的概率反映了在模拟生产的环境中被拒收的批次数量，从而可以用来判断平均需要的最少样本数量。从而帮助企业减少成本提高生产效益。

## 问题二模型建立与求解

### 问题二问题背景

在企业生产销售过程中，需要分别购买零配件1和零配件2，并在企业将两个零配件装配成成品。且在装配的成品中，只要其中一个零配件不合格，则成品一定不合格；如果两个零配件均合格，装配出的成品也不一定合格。对于不合格成品，企业可以选择报废，或者对其进行拆解，拆解过程不会对零配件造成损坏，但需要花费拆解费用。在此条件下要求我们在已知两种零配件和成品次品率的情况下，为企业在生产中遇到的情况的各个阶段做出相应的决策。

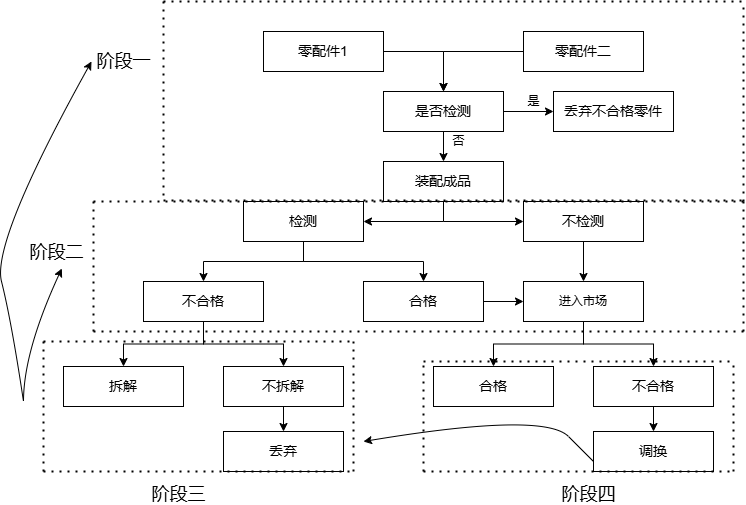
### 问题二求解思路

由于该问题要求我们在不同情况做出决策，所以我们需要目标函数对我们的决策进行判断是否为最优决策，根据企业实际生产情况，我们将最大利润作为我们的目标函数，为了求解最大利润的决策策略，我们考虑使用遗传算法解决此问题。首先我们定义一个适应度函数，该函数基于利润最大化原则，考虑销售额、成本（包括零件成本、检测成本、组装成本、拆解费用和调换损失）等因素。然后，将生产策略的决策变量（如是否检测零件、是否拆解次品等）编码二进制形式，以方便遗传算法处理。接下来，初始化一组随机解作为种群，每个解代表一种可能的生产策略。通过选择、交叉和变异操作迭代地改进这些解。选择操作根据适应度函数筛选出适应度较高的解，交叉操作通过组合两个解的基因来生成新的解，变异操作则通过随机改变某些基因来增加种群的多样性。在每一代中，计算每个个体的适应度，并根据这些适应度来生成新一代种群。这个过程重复进行，直到达到预定的迭代次数或适应度收敛。最后，从最终的种群中选择适应度最高的个体，即最优的生产策略，并对其进行稳健性分析，以评估其在不同生产情况下的表现。

### 问题二模型建立

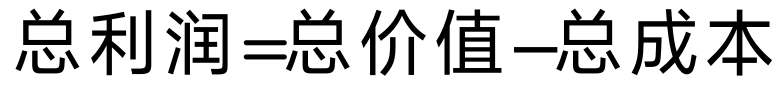
基于前面的思路，我们决定利用遗传算法计算出目标函数与决策变量之间的关系，在解决问题前，我们明确了企业在生产过程中需要面临三大决策，且绘画了企业生产流程，如下图所示：

1. 是否检测零配件1和/或2：如果检测，可以减少进入装配过程的不合格零件数，从而减少成品次品率，但会产生检测费用；反之如果不检测，装配结束后成品中次品比例增加，流入市场会产生额外的调换损失和拆解费用，但不需要检测费用。
2. 是否检测成品：若果检测，可以减少不合格品进入市场产生的额外的调换损失和拆解费用，但会产生检测费用；反之如果不检测成品中次品流入市场比例增加，流入市场会产生额外的调换损失和拆解费用，但不需要检测费用。
3. 是否拆解不合格品：如果拆解，虽然会产生拆解费用，但可能拆解出合格零件重复使用，以回收成本；反之如果不拆解，虽然没有拆解费用，但本产品的购买与检测等成本无法挽回。

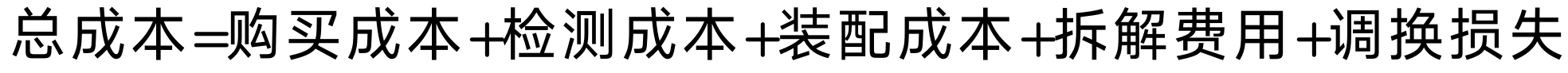


**目标函数：**

企业生产的目标是在成本与利润之间达到最佳平衡，使得利润达到最大值，即目标函数可表示为：



总成本的计算公式为：

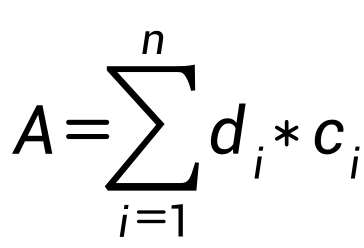


·总价值表示在给定零件1和零件2数量后装配后生成的成品数全部销售所赚的总金额

·总成本表示包含零配件购买成本A、零配件和成品检测成本B、装配成本C、拆解费用D和调换损失E

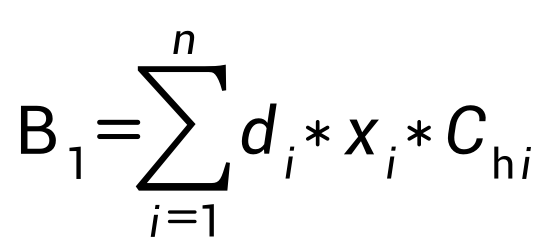
###### 阶段一：零件购买与检测成本

零件购买成本计算公式：



·零件购买成本为购买零件1的花费和购买零件2的花费之和

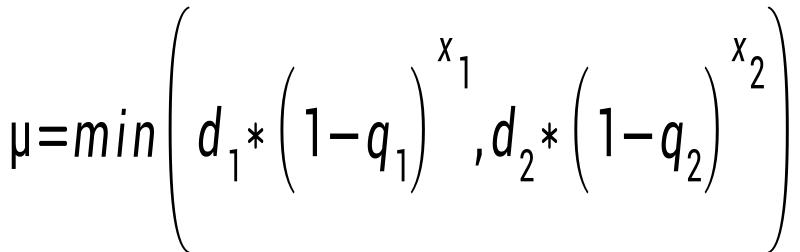
零件检测成本计算公式：



·零件检测成本为零件1、2的检测费用之和，xi取0或1,0表示不检测零件，1表示检测零件

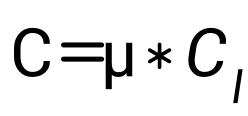
###### 阶段二：成品装配与检测成本

装配成品数计算公式：

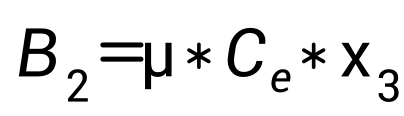


·装配成品数表示在给定零件1和零件2数量后装配生成的成品数

成品装配成本计算公式：



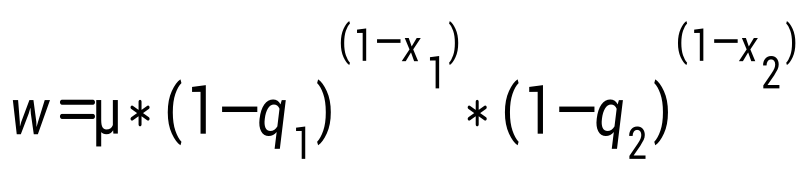
成品检测成本计算公式：



·成品检测成本为成品检测费用，x3取0或1,0表示不检测成品，1表示检测成品

###### 阶段三：不合格品拆解成本

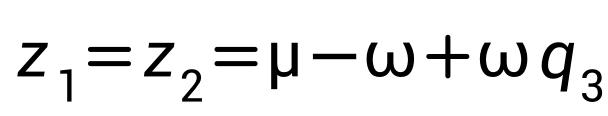
·正品数量计算公式：



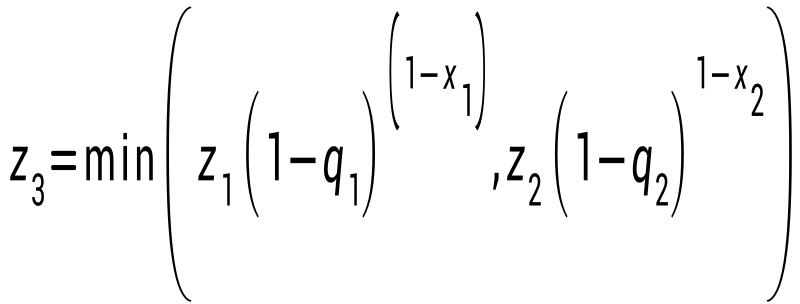
·不合格品拆解带来的收益

考虑到企业在得到拆解后的不合格品，为了防止再次组装后产生不合格品从而影响企业多方面问题。我们做出合理假设：企业将对零件一和零件二在本次流程中未检测的零件进行检测同时对成品一定进行检测从而为企业带来一定的收益。

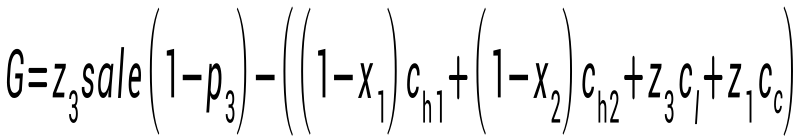
首先我们计算拆解成品后得到的零件一/二数量：

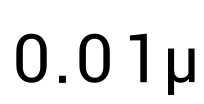


以及检测后组成成品的数量：



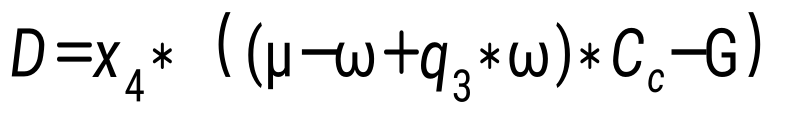
然后可以计算检测成本以及拆解装配成本以及该次卖出的金额，从而估算带来的收益：



其中sale为成品销售单价。其中经过估算改成剩余的不合格品数量大概为，比例太小不再考虑其带来的收益。

·正品数表示在给定零件1和零件2数量后装配生成的成品中的合格品

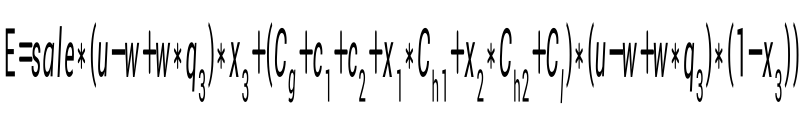
拆解费用：



·拆解费用表示检测成品时筛选出的不合格品和用户退回的不合格品的拆解费与拆解得到合格零件返还的收益之差，x4取0或1,0表示不拆解，1表示拆解

###### 阶段四：调换不合格产品成本

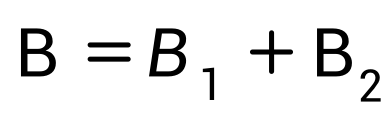
调换损失：



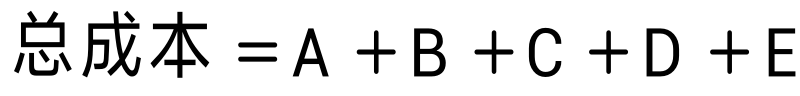
·调换损失表示因为调换产品而丧失的利润

###### 阶段五：整合总成本

检测成本计算公式：

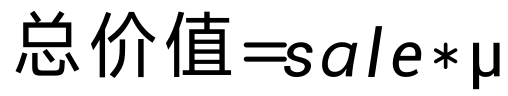


总成本计算公式：

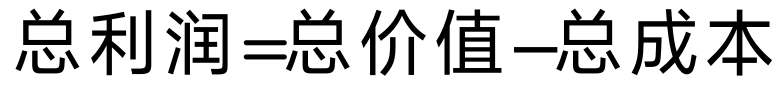


###### 阶段六：计算总利润

总价值计算公式：



总利润计算公式：



### 问题二模型求解与分析

本题我们开始想到最简单的算法，枚举法，计算每种策略组合下的总利润，并选取总利润最大的策略为最优解，且因为变量数较少（共4\*2\*2=16种组合），组合策略也较少，我们认定枚举法策略是可行的。但在后续实现过程中，我们发现用遗传算法可以很好优化我们的数学模型，故我们重写了测试代码，发现测试结果和枚举法结果相同。故遗传算法也是可行且有效的方法。

###### 基于遗传算法的求解步骤

‌初始化‌：随机生成一组初始解，这组解被称为种群。每个解在遗传算法中通常被表示为一个‌染色体，对应问题的一个可能答案。‌

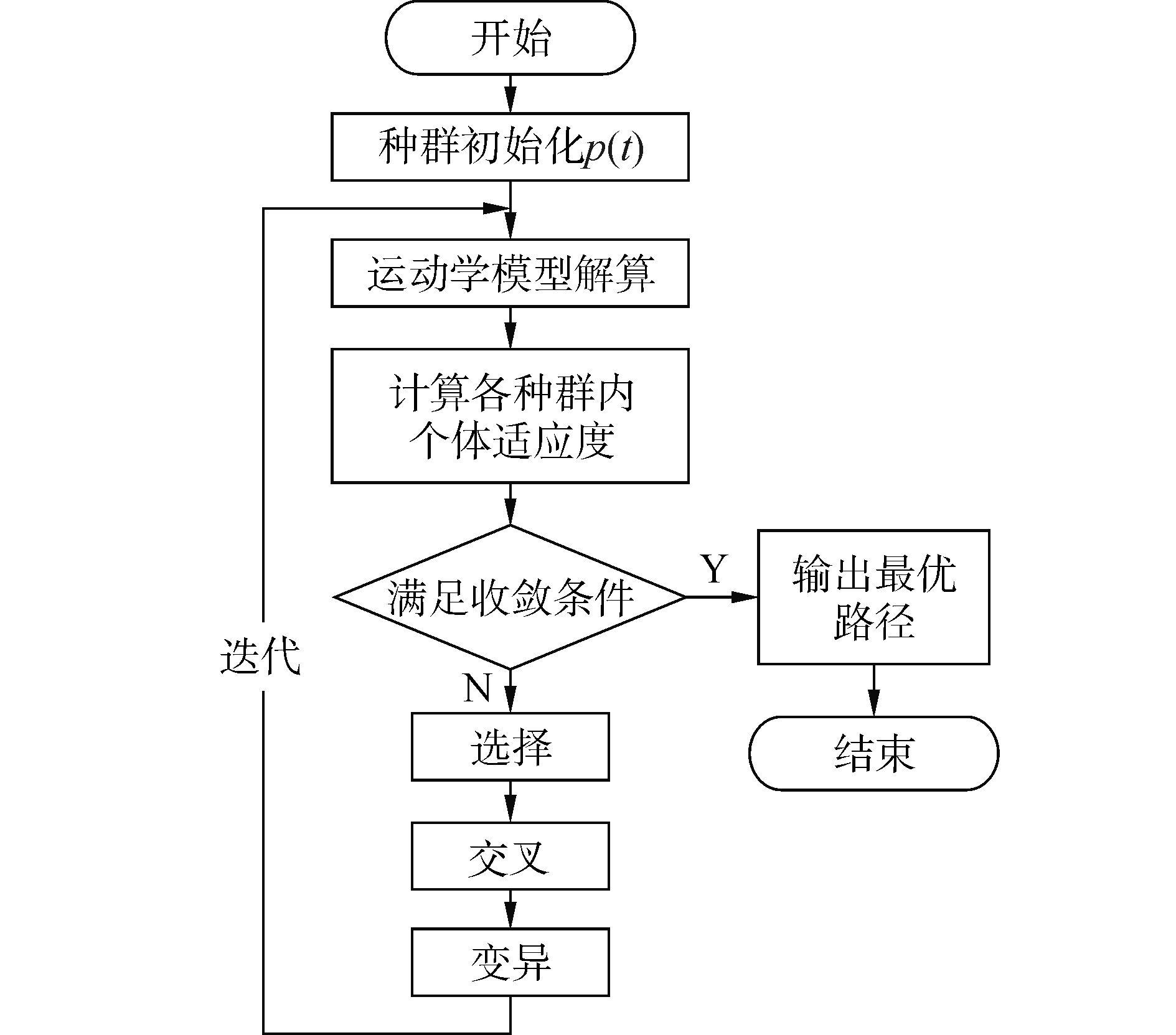
评估适应度‌：对种群中的每个个体（解）进行适应度评估。这通常涉及解码个体并根据问题的目标函数计算其适应值，该值反映了个体对环境的适应程度或解决问题的优劣。‌

选择‌：根据适应度评估的结果，选择适应度高的个体进行繁殖。选择操作确保了优秀的基因能够有更多的机会传递给下一代。‌

交叉（杂交）‌：通过交叉操作生成新的个体。这通常涉及在两个父代染色体的特定位置进行切割和交换，以产生两个新的子代染色体。交叉操作有助于在搜索空间中探索新的可能解。‌

变异‌：以较小的概率对个体进行变异操作。变异涉及随机改变染色体中的某些基因值，以模拟自然界中的基因突变。变异操作有助于保持种群的多样性，并防止算法过早收敛到局部最优解。‌

迭代‌：重复上述评估、选择、交叉和变异的过程，直到满足终止条件。终止条件可以是达到预定的迭代次数、适应度达到某个阈值或找到满足问题要求的解。



###### 问题二求解结果与性能分析

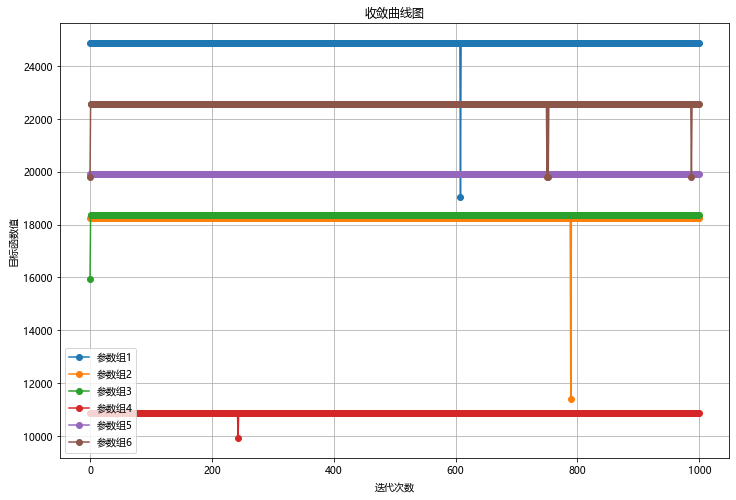
下面给出每种情况对应的最优决策以及在零件一和零件二的个数为1000个的情况下对应的最大利润：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 情况 | 零件一是否检测 | 零件二是否检测 | 成品是否检测 | 不合格品是否拆解 | 最大利润 |
| 1 | 0 | 0 | 0 | 1 | 24883.5 |
| 2 | 0 | 0 | 0 | 1 | 18240.0 |
| 3 | 0 | 0 | 0 | 1 | 18379.5 |
| 4 | 1 | 1 | 0 | 1 | 10880.0 |
| 5 | 0 | 0 | 0 | 1 | 19904.0 |
| 6 | 0 | 0 | 0 | 0 | 22580.25 |

其中0代表当前决策不选择，1代表当前决策选择。

下面对该模型算法的性能进行分析：

收敛曲线图的分析：该曲线图展示了对利润的目标函数进行最大化优化时，最优目标函数的值随着迭代次数的变化。由图中信息可知，该模型在进行优化时，收敛较快且最终目标函数的最优值趋于稳定没有振荡的过程，说明了求解的最优值可靠稳定。



## 问题三模型建立与求解

### 问题三问题背景

问题三的背景涉及一个更为复杂的多工序、多零配件的生产流程。在这种情境下，企业不仅要管理单个零配件的采购、检测、装配和次品处理，还需要协调多个工序中涉及的零配件和半成品的动态生产决策。与问题二相比，问题三引入了更多的决策变量和成本因素，要求企业从系统的角度对整个生产链进行优化管理。根据实际生产情况，生产一个成品需要半层品，可能会有m道工序，n个零配件，而零件，半成品以及成品会尤其自身的次品率，如果组成半成品的零件有次品则半成品一定是次品，同理如果组成成品的半成品有次品，则成品一定为次品，在该情境中，生产过程包括多个零配件，这些零配件经过多个工序的加工，逐步形成半成品和成品。每个零配件、半成品和成品都存在一定的次品率，而企业则需要决定是否对这些零配件、半成品和成品进行检测。在检测过程中，如果发现次品，企业可以选择直接丢弃、拆解或者继续装配，这一决策将直接影响企业的生产效率和最终利润。同时，成品的市场售价、退货后的调换损失以及拆解不合格成品的费用等也是影响企业决策的重要因素。此外，不同的工序中涉及不同的检测成本、装配成本和拆解费用，因此，企业需要在每个阶段仔细权衡这些成本与收益。

### 问题三求解思路

问题三的求解思路涉及对多道工序、多零配件的生产流程进行系统性分析和优化。通过综合考虑各个阶段的次品率、检测成本、装配成本、拆解费用等因素，可以确定最优的决策路径，以最大化企业的整体利润。首先，需要对生产过程中涉及的各个零配件和工序进行建模。零配件具有不同的次品率和采购成本，而每道工序中，企业需决定是否对零配件进行检测。若进行检测，检测不合格的零配件可以被丢弃，从而避免次品进入后续装配阶段。然而，检测成本会增加，因此检测与否的决策应基于成本与次品率的权衡。如果检测成本高于次品对生产后续环节带来的潜在损失，企业可以选择跳过检测。对于装配过程，企业需要考虑是否对半成品和成品进行检测。半成品检测可以避免不合格品进入成品阶段，而成品检测则能降低不合格品进入市场的风险。如果不进行检测，可能导致次品进入市场，从而增加退货和调换的成本。成品的市场售价和调换损失也是关键因素，尤其是当成品的次品率较高或市场调换损失较大时，检测成品的必要性就更加明显。在处理不合格成品时，企业有两个选择：拆解或丢弃。如果选择拆解，企业可以回收部分零配件，并重新进入生产流程。但拆解费用需要与丢弃的直接损失进行比较。如果拆解费用较高，直接丢弃次品成品可能更经济；如果拆解费用较低，回收可用零配件再利用可能更加划算。对于退回的不合格品，企业需要承担调换的物流成本和潜在的信誉损失，这部分损失应与成品检测的成本进行比较。如果检测成品的成本低于后续调换损失，进行成品检测是合理的选择；反之则可跳过检测。

### 问题三模型建立

对于m道工序、n个零配件且已知零配件、半成品和成品的次品率的情况：

通过分析可知该种情况中组成原件组成一原件一共可划分为三类情况：

·组成原件为零配件，生产原件为半成品

·组成原件为半成品，生产原件为半成品

·组成原件为半成品，生产原件为成品

首先我们做出以下合理假设以及参数说明：

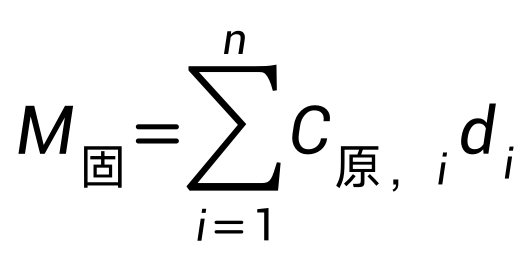
###### 半成品在作为生产下一道工序的配件时其成本等于组成该半成品的配件成本之和（包括检验成本，组装成本）加上自己成本D

###### 由于m，n未知故设不合格的半成品其拆解后单个效益为G1，不合格的成品其拆解后单个效益为G2。

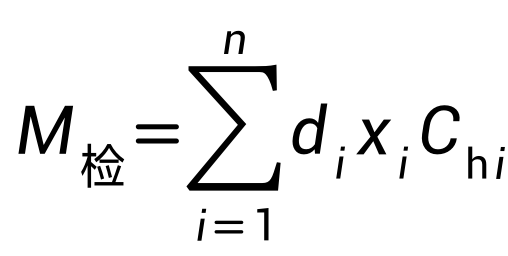
###### 每类情况的组成原件数量为n个，Cli是该生产工序装配的成本，原件i是否被检测为xi（0不检测），原件i的检测成本为Chi, 原件i的数量为di，原件i成本为C原，i，原件i的次品率为pi。

然后我们进行第一类情况进行讨论：

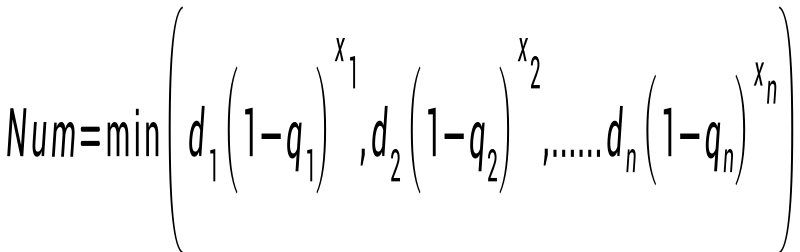
固定成本：



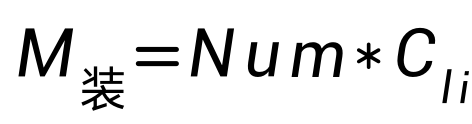
检测成本：



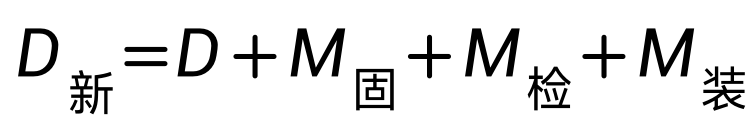
生产的原件数量：



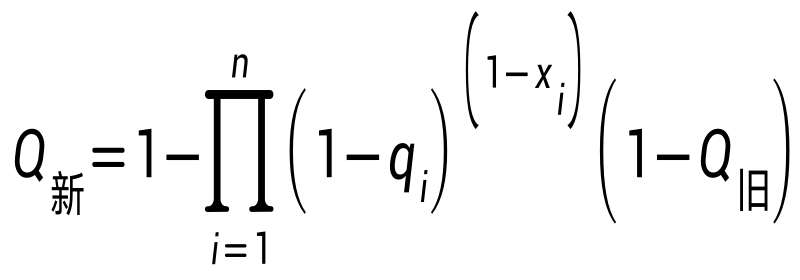
装配成本：



生产的原件的新成本：



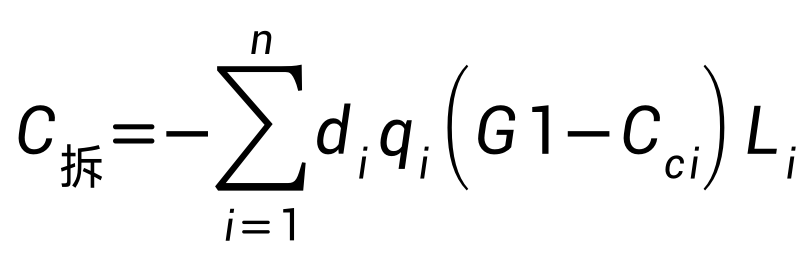
生产的原件的新次品率：



第二类情况进行讨论：

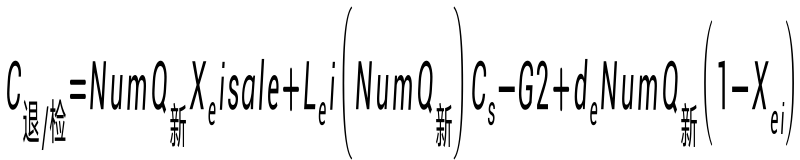
该种情况与第一种情况相比多了一个抉择：是否对检测不合格的半成品进行拆解我们设第i个半成品原件检测不合格拆解为Li。

即多了一个拆解成本：



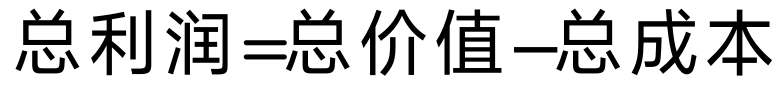
第三类情况进行讨论：

该种情况与第二种情况相比也多了二个抉择：是否对生产的成品进行检测Xei以及是否对检测不合格的产品进行拆解Lei。同时考虑到第二问所说对用户购买的不合格品无条件退换，设退还损失为Cs以及当前的成品成本，考虑到m和n未知设当前成品成本为de。由此多了一个退还/检测成本

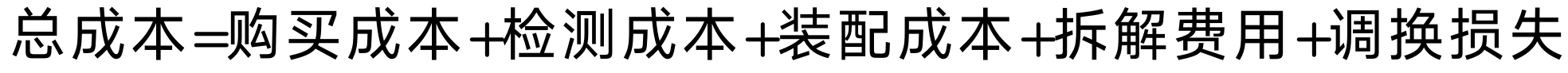


目标函数的建立：

企业生产的目标是在成本与利润之间达到最佳平衡，使得利润达到最大值，即目标函数可表示为：



总成本的计算公式：

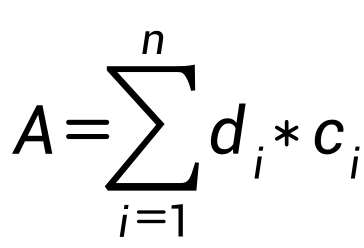


·总价值表示在给定零件1和零件2数量后装配后生成的成品数全部销售所赚的总金额

·总成本表示包含零配件购买成本A、零配件和成品检测成本B、装配成本C、拆解费用D和调换损失E

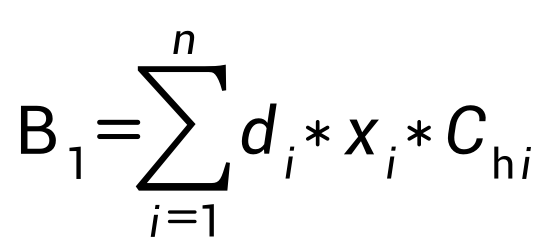
###### 阶段一：零件购买与检测成本

零件购买成本计算公式：



·零件购买成本为购买八种零件的花费之和

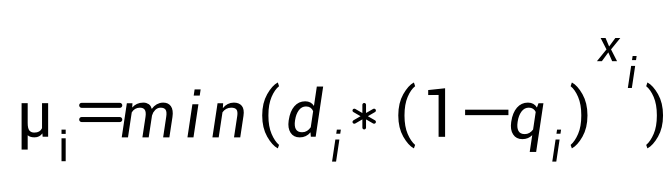
零件检测成本计算公式：



·零件检测成本为八种零件的检测费用之和，xi取0或1,0表示不检测零件，1表示检测零件

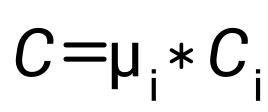
###### 阶段二：半成品装配与检测成本

装配半成品数计算公式：

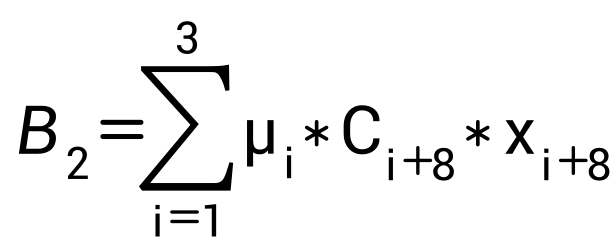


·装配半成品数表示在给定八种零件数量后装配生成的半成品数

半成品装配成本计算公式：



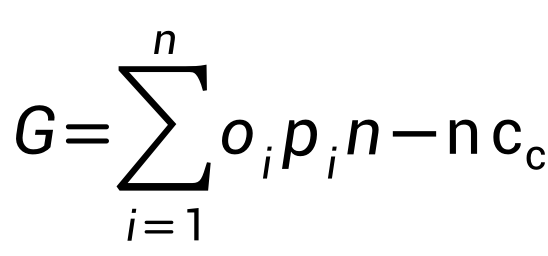
半成品检测成本计算公式：



·半成品检测成本为半成品检测费用，xi+8取0或1,0表示不检测半成品，1表示检测半成品。

###### 阶段三：本文中拆解成品/半成品的收益

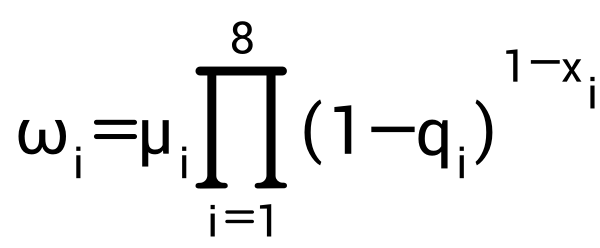
为了简便计算，本文做出合理假设即将合格的组成原件的成本作为收益的来源，收益计算公式为：



其中，oi是该成品/半成品的组成原件i的成本，包括其本身购买成本以及检测成本，pi为拆解后组成原件i的合格率，n为该组成原件个数。

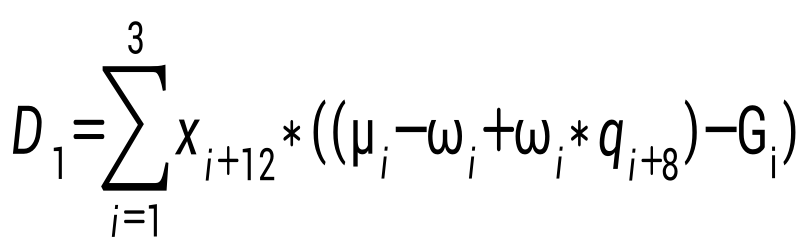
###### 阶段三：不合格半成品拆解成本

半成品正品数计算公式:



·半成品正品数表示在给定八种零件数量后装配生成的半成品中的合格品

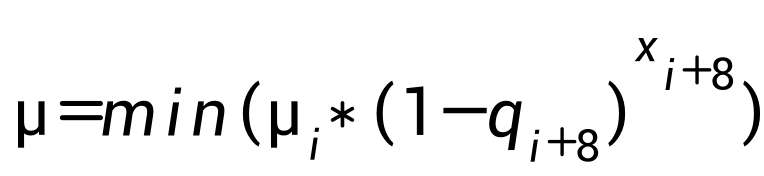
半成品拆解费用计算公式：



·拆解费用表示检测成品时筛选出的不合格品，x4取0或1,0表示不拆解，1表示拆解

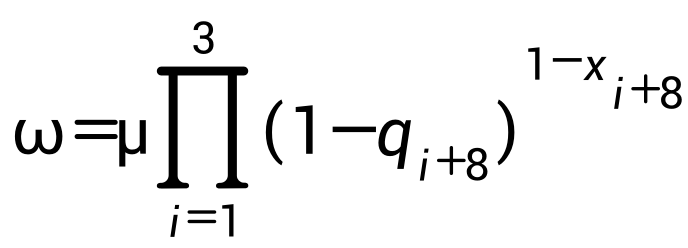
###### 阶段四：不合格成品拆解成本

装配成品数计算公式：



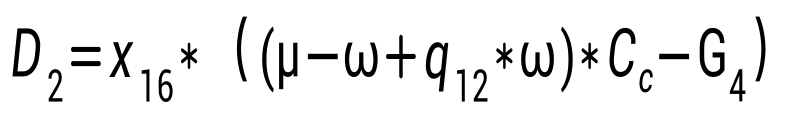
·装配成品数表示在给定八种零件数量后装配生成的成品数

成品正品数计算公式:



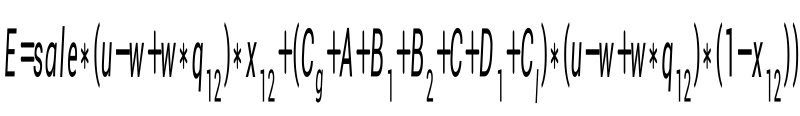
·成品正品数表示在给定八种零件数量后装配生成的成品中的合格品

成品拆解费用计算公式：



###### 阶段五：调换不合格产品成本

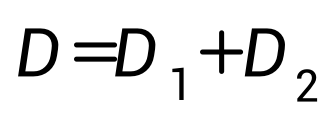
调换损失函数：



·调换损失表示为调换产品而丧失的利润

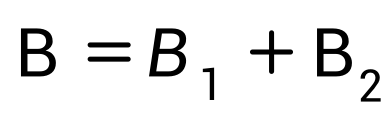
###### 阶段六：整合总拆解费用

拆解费用计算公式：

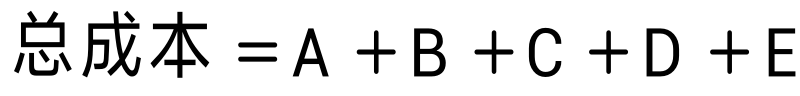


###### 阶段七：整合总成本

检测成本计算公式：

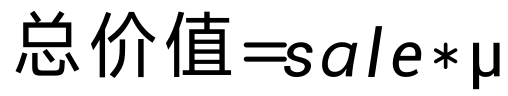


总成本计算公式：

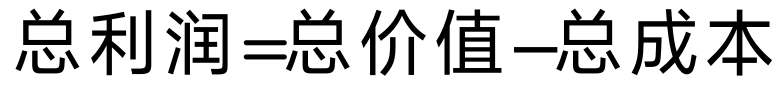


###### 阶段八：计算总利润

总价值计算公式：



总利润计算公式：



### 问题三模型求解与分析

从问题二开始类推，本题我们开始仍然想尝试枚举法，计算每种策略组合下的总利润，并选取总利润最大的策略为最优解，但因变量数过多，组合策略也过多，我们认定枚举法策略是不可行的。故在后续实现过程中，我们发现用遗传算法还是可以很好优化我们的数学模型，故我们重写了测试代码，发现测试结果和枚举法结果相同。故遗传算法是可行且有效的方法。

###### 基于遗传算法的求解步骤

‌初始化‌：随机生成一组初始解，这组解被称为种群。每个解在遗传算法中通常被表示为一个‌染色体，对应问题的一个可能答案。‌

评估适应度‌：对种群中的每个个体（解）进行适应度评估。这通常涉及解码个体并根据问题的目标函数计算其适应值，该值反映了个体对环境的适应程度或解决问题的优劣。‌

选择‌：根据适应度评估的结果，选择适应度高的个体进行繁殖。选择操作确保了优秀的基因能够有更多的机会传递给下一代。‌

交叉（杂交）‌：通过交叉操作生成新的个体。这通常涉及在两个父代染色体的特定位置进行切割和交换，以产生两个新的子代染色体。交叉操作有助于在搜索空间中探索新的可能解。‌

变异‌：以较小的概率对个体进行变异操作。变异涉及随机改变染色体中的某些基因值，以模拟自然界中的基因突变。变异操作有助于保持种群的多样性，并防止算法过早收敛到局部最优解。‌

迭代‌：重复上述评估、选择、交叉和变异的过程，直到满足终止条件。终止条件可以是达到预定的迭代次数、适应度达到某个阈值或找到满足问题要求的解。

###### 问题二求解结果与性能分析

下面在零件一，零件二，零件三，零件四，零件五，零件六，零件七，零件八的个数为1000个的情况下对应的最大利润以及最优决策：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 零件一 | 零件二 | 零件三 | 零件四 | 零件五 | 零件六 | 零件七 | 零件八 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

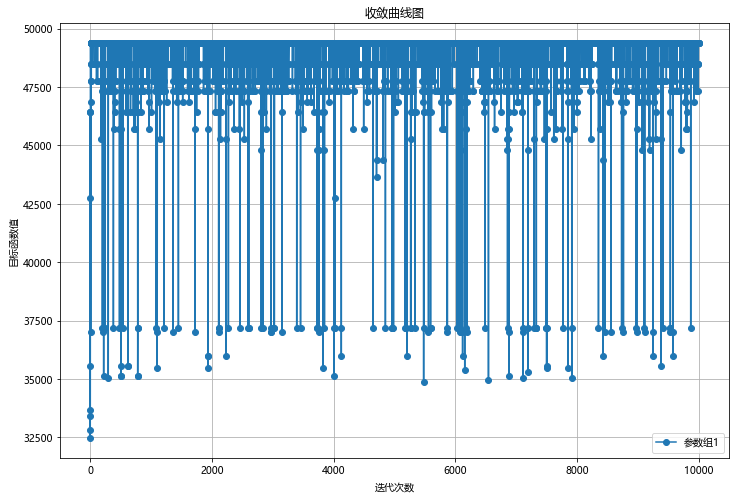
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 半成品一 | 半成品二 | 半成品三 | 成品 | 半成品一 | 半成品二 | 半成品三 | 成品 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

|  |
| --- |
| 最大利润 |
| 49400.0 |

前十二个决策代表是否，后面四个决策代表是否拆解，检测其中0代表当前决策不选择，1代表当前决策选择。

下面进行性能分析：

收敛曲线图分析：该曲线图展示了对利润的目标函数进行最大化优化时，最优目标函数的值随着迭代次数的变化。由图中信息可知，该模型在进行优化时，收敛较快，说明求解最优解的速度较快但是随着迭代次数的增加，目标函数的最优解大概率收敛基本收敛于一条直线，故仍能认为该模型可以求解一个可靠的最优决策以及目标值。

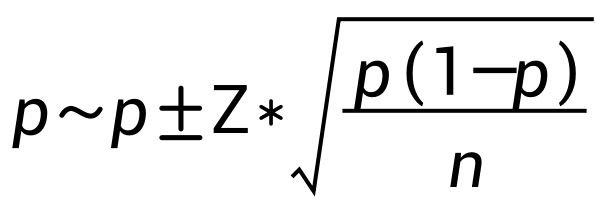


## 问题四模型建立与求解

### 问题四重写问题二模型建立

和问题二相同，问题四和问题二建立的模型相同，只是化静为动，将次品率纳入变量考量范围，引入不确定性，以下是模型建立过程。

1. 基于二项分布次品率的执行区间：



·p是样本次品率，n是抽样检测量，Z是对应置信度正态分布临界值

1. 基于抽样检测估计次品率

在问题二基础上，我们对零件1、零件2、成品使用二项分布进行次品率估计，并确定样本量，并在问题一的基础上，求出在95%和90%置信区间下的次品率置信区间。

1. 根据次品率做出决策

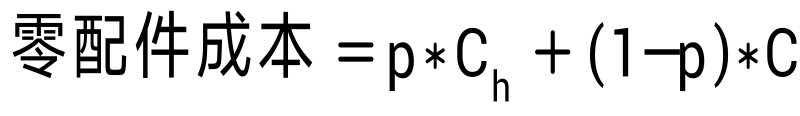
我们规定零配件和成品的次品率在5%—15%之间，若超过15%这个阈值，我们认定零件次品率高，需要检测，若远小于阈值，我们认定零件次品率低，我们可以少检验甚至不检验。

### 问题四重写问题二模型求解与分析

根据问题，零配件和成品的次品率均是通过抽样检测方法得到的，且我们规定零配件和成品的次品率在5%—15%之间，若超过15%这个阈值，我们认定零件次品率高，需要检测，若远小于阈值，我们认定零件次品率低，我们可以少检验甚至不检验，据此得出企业在生产过程中各决策的平衡，以实现最优决策。

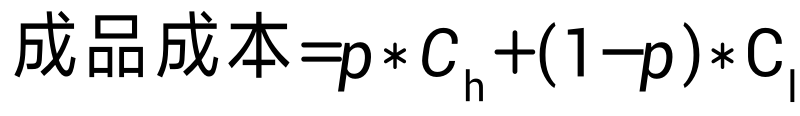
1. 成本分析

零配件成本：

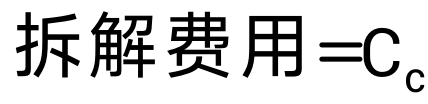


·在选择配件时，当零件次品率为p时，我们在p概率下认定其为次品，不购买，1-p概率认定为合格品，购买。

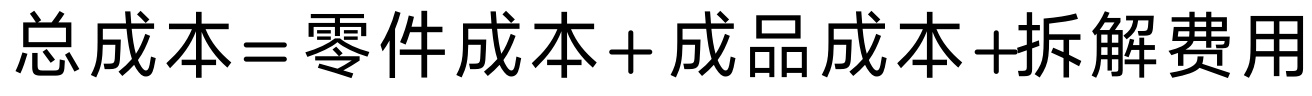
成品成本：



拆解成本：



总成本：



### 问题四重写问题三模型建立

和问题三相同，问题四和问题三建立的模型相同，只是化静为动，将次品率纳入变量考量范围，首先在次品率的不确定情形下，我们可以通过正态分布或者二项分布进行模拟次品率的分布，而根据题目中所说例如题目一中的方法，其分布是一个置信区间，而在考虑是否进行检测时，既要考虑检测成本还要考虑抽样结果的准确性。

(1)次品率的不确定性建模:

使用正态分布或者二项分布来模拟次品率的分布。在引入置信区间的基础上，使用贝叶斯更新的方法，随着生产过程的推进，不断更新对次品率的估计。

(2)成本优化模型:

模型中引入了不同次品率下的决策成本，使用动态规划优化整个生产过程中每个决策点的成本。将每个步骤的成本比如检测成本、装配成本、拆解成本等结合动态规划算法来寻找全局最优解。

### 问题四重写问题三模型求解与分析

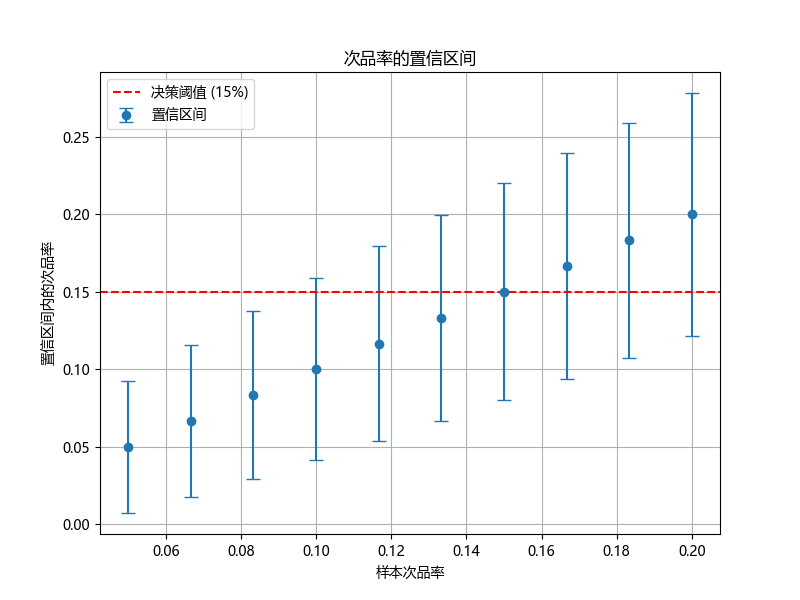
问题四在第三题的基础上引进了抽样检测，所以零配件，半成品以及成品的次品率不再是固定值，而是有一定误差的范围，所以引入了置信区间来进行处理这个误差，所以在每个决策点，既要考虑固定的次品率，还要考虑其上下限，基于置信区间的估计值动态调整决策，通过动态规划和统计推断相结合，优化在不同的生产过程中进行最优化决策。

1. 次品率估计：

假设每个零配件的次品率都是通过抽样检测得到，使用正态分布进行估计，使用问题一中的公式进行计算次品率的置信空间。

1. 在这个次品率估计区间内做出决策：

遇到置信区间的上界表明次品多于预期则对零部件或者半成品进行检测，减少次品卖出从而减少成本，反之则证明低于预期则不对零部件或者半成品进行检测，从而降低检测次数进而降低检测成本。



该图表明了**不同样本次品率的置信区间**，能够直观说明我模型中关于次品率不确定性的处理。问题四中，次品率并不是固定的，而是有误差范围。这正是通过**置信区间**的方式来表示的。除此之外，图中的红色虚线表示**企业设定的次品率决策阈值比**如15%。该图表明了在不同次品率下，置信区间是否会跨过这一阈值，这可以帮助企业决定是否需要检测或拒收某批次的零件。最后，通过图中的置信区间，可以看到某些样本次品率的置信区间上下限较宽，说明在这些情况下对次品率的估计不够准确，建议企业进行检测。而当置信区间较窄时，企业可以减少检测次数，从而降低检测成本。

# 模型评价与推广

## 模型的优点

1. 模型充分结合实际生产，考虑了诸多重要因素后以最大利润为决策的目的，得到合理的目标函数和模型，我们考虑了大量因素，如零件两者的数量不一定相等，考虑决策全面，成品的次品拆解后再次进行组装。这样得到的模型贴合实际，具有较高的应用价值，可以推广到其他工业生产领域。
2. 模型运用遗传和优化思想，抓住影响生产利润问题的重要因素，将复杂的决策问题转化为简单的优化问题，合理设置不同参数，模型的输出结果符合题目要求，能解决实际问题；
3. 本文得到的决策方案具有准确和合理特点，基本不存在模糊和遗漏信息等问题，在现有条件下能有效做出最优的决策。

## 模型的不足

1. 实际应用中，考虑到零件的数量过大和过小可能也是重要的因素，但本文未能考虑到这些因素的影响，一定程度上影响了模型的准确性；
2. 由于使用的统计学的相关公式过于冗杂和复杂，导致目标函数过于繁琐，从而导致编写代码难度增加。

## 模型的推广

在模型推广方面，可以将次品率参数替换为动态次品率模型，通过引入随时间变化的次品率，从而解决在长期生产过程中产品质量波动的问题。改进方面，可以结合参考文献中关于多阶段供应链的研究，进一步考虑供应商变更对次品率和成本的影响，从而得到更合理且具备适应性的生产决策模型。此类扩展有助于在多变的市场环境中优化生产效率与成本控制。

# 参考文献

1. 杨本朝,石雅男,段乾恒,李光松,于刚.大学生数学建模竞赛开展全周期教学实践探究[J].大学教育,2023(04):44-46.
2. 管宇.关于序贯抽样检验[J].浙江林学院学报,2002,(1): 90-95
3. 崔岩，崔迪.计数序贯抽样检验方法在元器件验收中的应用[J].长春光学精密机械学院学报,1996,(4): 66-70
4. 周俊文，刘界鹏.基于多种群遗传算法的钢框架结构优化设计[J].土木与环境工程学报(中英文),2024,第46卷(1): 71-81
5. 黄新林，张隆飛，唐小伟.基于改进遗传算法的家电回收车辆路径规划方法[J].同济大学学报(自然科学版),2024,第52卷(1): 27-34
6. 张文涛.数学建模核心素养测评的实践研究①[J].数学通报,2024,第63卷(2): 35-41

# 附录

|  |
| --- |
| 附录1 |
| 基于序贯抽样检测的第一问求解代码 |
| #求解ASN曲线代码  import numpy as np  import matplotlib.pyplot as plt  import math  def sprt(p, p0, p1, alpha, beta, n0, path, num\_s):  sample\_sizes = []  for \_ in range(num\_s):  sample\_size = n0  #蒙洛卡特模拟  num\_defective = np.random.binomial(n0, p)  while True:  # 计算似然比函数  lx = math.log((p1 / p0) \*\* num\_defective \* ((1 - p1) / (1 - p0)) \*\* (sample\_size - num\_defective),math.e)  # Decision thresholds  A = math.log(beta / (1 - alpha) ,math.e) # 下界  B = math.log((1 - beta) / alpha, math.e )# 上界  if lx <= A:  sample\_sizes.append(sample\_size)  break  elif lx >= B:  sample\_sizes.append(sample\_size)  break  else:  sample\_size += path  num\_defective += np.random.binomial(path, p)  return np.mean(sample\_sizes)  p0 = 0.10 # 标称值（可接受合格质量水平）  p1 = 0.15 # 极限质量水平  alpha = 0.05 # 第一类错误的风险概率（情形一（0.05）/情形二（0.10））  beta = 0.10 # 第二类错误的风险概率（情形一（0.10）/情形二（0.20））  n0 = 10 # 初始样本数量  path = 5 # 每次抽样的数量  num\_s = 10000 #模拟次数  defect\_rates = np.linspace(0.01, 0.20, 30)  sample\_sizes = [sprt(p, p0, p1, alpha, beta, n0, path, num\_s) for p in defect\_rates]  # 画图  plt.figure(figsize=(10, 6))  plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  plt.plot(defect\_rates, sample\_sizes, marker='o', label='ASN曲线', color='blue')  plt.axvline(x=0.10, color='red', linestyle='--', label='真实概率 = 0.10')  plt.axvline(x=0.15, color='green', linestyle='--', label='真实概率 = 0.15')  plt.xlabel('真实次品率')  plt.ylabel('平均样本量')  plt.title('情形一 平均样本量(ASN)曲线')  plt.grid(True)  plt.legend(loc='upper right')  plt.show() |
|  |
| #OC曲线求解算法  import numpy as np  import matplotlib.pyplot as plt  import math  def sprt(p, p0, p1, alpha, beta, n0, path, num\_s):  acceptance\_p = []  for \_ in range(num\_s):  sample\_size = n0  #蒙洛卡特模拟  num\_defective = np.random.binomial(n0, p)  accepted = False  while True:  # 计算似然函数  lx = math.log((p1 / p0) \*\* num\_defective \* ((1 - p1) / (1 - p0)) \*\* (sample\_size - num\_defective), math.e)  # Decision thresholds  A = math.log(beta / (1 - alpha), math.e) # 下界  B = math.log((1 - beta) / alpha, math.e) # 上界  if lx <= A:  accepted = True  break  elif lx >= B:  break  else:  sample\_size += path  num\_defective += np.random.binomial(path, p)  acceptance\_p.append(accepted)  return np.mean(acceptance\_p)  # Parameters  p0 = 0.10 # 标称值（可接受合格质量水平）  p1 = 0.15 # 极限质量水平  alpha = 0.05 # 第一类错误的风险概率（情形一（0.05）/情形二（0.10））  beta = 0.10 # 第二类错误的风险概率（情形一（0.10）/情形二（0.20））  n0 = 1 # 初始样本数量  path = 1 # 每次抽样的数量  num\_s = 1000  defect\_rates = np.linspace(0.01, 0.20, 30)  acceptance\_p = [sprt(p, p0, p1, alpha, beta, n0, path, num\_s)  for p in defect\_rates]  #做出oc曲线图  plt.figure(figsize=(10, 6))  plt.plot(defect\_rates, acceptance\_p, marker='o', label='OC曲线', color='blue')  plt.axvline(x=0.10, color='red', linestyle='--', label='真实概率 = 0.10')  plt.axvline(x=0.15, color='green', linestyle='--', label='真实概率 = 0.15')  plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  plt.xlabel('真实次品率')  plt.ylabel('接收概率')  plt.title('情形一 操作特性(OC)曲线')  plt.yticks(np.arange(0, 1, 0.05))  plt.grid(True)  plt.legend(loc='upper right')  plt.show() |
| 基于遗传算法的问题二的代码 |
| import geatpy as ea  import numpy as np  import matplotlib.pyplot as plt  # 定义目标函数  def goal\_function(chrom, params):  x1, x2, x3, x4 = chrom  u = min(params['d1'] \* ((1 - params['q1']) \*\* x1), params['d2'] \* ((1 - params['q2']) \*\* x2))  w = u \* (1 - params['q1']) \*\* (1 - x1) \* (1 - params['q2']) \*\* (1 - x2)  n = (u - w + w \* params['q3']) \* min((1 - params['q1']) \*\* (1 - x1), (1 - params['q2']) \*\* (1 - x2))  good = n \* (1 - params['q3']) \* (params['sale'] - params['cl']) - (u - w + w \* params['q3']) \* (params['cc'] + params['ch3'] + (1 - x1) \* params['ch1'] + (1 - x2) \* params['ch2'])  return u \* params['sale'] - (params['d1'] \* params['c1'] + params['d2'] \* params['c2'] + params['d1'] \* x1 \* params['ch1'] + params['d2'] \* x2 \* params['ch2'] + u \* x3 \* params['ch3'] + u \* params['cl'] + x4 \* ((u - w + w \* params['q3']) \* (params['cc']) - good) + params['sale'] \* (u - w + w \* params['q3']) \* x3 + (params['cg'] + params['c1'] + params['c2'] + x1 \* params['ch1'] + x2 \* params['ch2'] + params['cl']) \* (u - w + w \* params['q3']) \* (1 - x3))  # 定义自定义问题类  class MyFunction(ea.Problem):  def \_\_init\_\_(self, params):  name = 'MyFunction'  M = 1  maxormins = [-1]  Dim = 4  varTypes = [1] \* Dim  lb = [0] \* Dim  ub = [1] \* Dim  lbin = [1] \* Dim  ubin = [1] \* Dim  ea.Problem.\_\_init\_\_(self, name, M, maxormins, Dim, varTypes, lb, ub, lbin, ubin)  self.params = params  self.log = []  def aimFunc(self, pop):  X = pop.Phen  PopObj = np.apply\_along\_axis(lambda chrom: goal\_function(chrom, self.params), 1, X)  pop.ObjV = PopObj.reshape(-1, 1)  self.log.append(pop.ObjV.max()) # 记录每一代的最小目标函数值  # 定义参数集  parameter\_sets = [  {'q1': 0.1, 'q2': 0.1, 'q3': 0.1, 'sale': 56, 'ch1': 2, 'ch2': 3, 'ch3': 3, 'cl': 6, 'cc': 5, 'cg': 6, 'c1': 4, 'c2': 18, 'd1': 1000, 'd2': 1000},  {'q1': 0.2, 'q2': 0.2, 'q3': 0.2, 'sale': 56, 'ch1': 2, 'ch2': 3, 'ch3': 3, 'cl': 6, 'cc': 5, 'cg': 6, 'c1': 4, 'c2': 18, 'd1': 1000, 'd2': 1000},  {'q1': 0.1, 'q2': 0.1, 'q3': 0.1, 'sale': 56, 'ch1': 2, 'ch2': 3, 'ch3': 3, 'cl': 6, 'cc': 5, 'cg': 30, 'c1': 4, 'c2': 18, 'd1': 1000, 'd2': 1000},  {'q1': 0.2, 'q2': 0.2, 'q3': 0.2, 'sale': 56, 'ch1': 1, 'ch2': 1, 'ch3': 2, 'cl': 6, 'cc': 5, 'cg': 30, 'c1': 4, 'c2': 18, 'd1': 1000, 'd2': 1000},  {'q1': 0.1, 'q2': 0.2, 'q3': 0.1, 'sale': 56, 'ch1': 8, 'ch2': 1, 'ch3': 2, 'cl': 6, 'cc': 5, 'cg': 10, 'c1': 4, 'c2': 18, 'd1': 1000, 'd2': 1000},  {'q1': 0.05, 'q2': 0.05, 'q3': 0.05, 'sale': 56, 'ch1': 2, 'ch2': 3, 'ch3': 3, 'cl': 6, 'cc': 40, 'cg': 10, 'c1': 4, 'c2': 18, 'd1': 1000, 'd2': 1000},  ]  # 收集结果  results = []  convergence\_data = []  for params in parameter\_sets:  problem = MyFunction(params)    algorithm = ea.soea\_EGA\_templet(  problem,  ea.Population(Encoding='BG', NIND=50),  MAXGEN=1000,  logTras=0,  Pc=0.8,  Pm=0.02  )    result = ea.optimize(algorithm)    # 收集每组数据的收敛曲线数据  generations = np.arange(len(problem.log))  convergence\_data.append(problem.log)    optimal\_solution = result['Vars'][0]  optimal\_value = result['ObjV'][0][0]  results.append({  'x1': optimal\_solution[0],  'x2': optimal\_solution[1],  'x3': optimal\_solution[2],  'x4': optimal\_solution[3],  'Optimal Value': optimal\_value  })  # 绘制所有参数组的收敛曲线图  plt.figure(figsize=(12, 8))  labels = [f'参数组{i+1}' for i in range(len(parameter\_sets))]  for i, log in enumerate(convergence\_data):  plt.plot(np.arange(len(log)), log, marker='o', label=labels[i])  plt.xlabel('迭代次数')  plt.ylabel('目标函数值')  plt.title('收敛曲线图')  plt.legend()  plt.grid(True)  plt.show()  # 创建结果表格  fig, ax = plt.subplots(figsize=(10, 4))  ax.axis('off')  plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  data = []  headers = ['x1', 'x2', 'x3', 'x4', '最大利润']  for result in results:  data.append([result['x1'], result['x2'], result['x3'], result['x4'], result['Optimal Value']])  table = ax.table(cellText=data, colLabels=headers, cellLoc='center', loc='center', bbox=[0, 0, 1, 1])  table.auto\_set\_font\_size(False)  table.set\_fontsize(10)  table.auto\_set\_column\_width(range(len(headers)))  plt.title('优化结果表格')  plt.show() |
| 基于遗传算法的第三问代码 |
|  |
| 基于遗传算法的第四问代码 |
| import numpy as np  from scipy.stats import norm  import matplotlib.pyplot as plt  def calculate\_confidence\_interval(p, n, confidence\_level):  z = norm.ppf(1 - (1 - confidence\_level) / 2)  margin\_of\_error = z \* np.sqrt((p \* (1 - p)) / n)  lower\_bound = max(0, p - margin\_of\_error)  upper\_bound = min(1, p + margin\_of\_error)  return lower\_bound, upper\_bound  def plot\_confidence\_interval():  p\_values = np.linspace(0.05, 0.2, 10)  n\_sample = 100  confidence\_level = 0.95  lower\_bounds = []  upper\_bounds = []  for p in p\_values:  lower, upper = calculate\_confidence\_interval(p, n\_sample, confidence\_level)  lower\_bounds.append(lower)  upper\_bounds.append(upper)  plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  plt.figure(figsize=(8, 6))  plt.errorbar(p\_values, p\_values,  yerr=[np.array(p\_values) - np.array(lower\_bounds), np.array(upper\_bounds) - np.array(p\_values)],  fmt='o', capsize=5, label='置信区间')  plt.axhline(y=0.15, color='r', linestyle='--', label='决策阈值 (15%)')  plt.xlabel('样本次品率')  plt.ylabel('置信区间内的次品率')  plt.title('次品率的置信区间')  plt.legend()  plt.grid(True)  plt.show()  if \_\_name\_\_ == "\_\_main\_\_":  plot\_confidence\_interval() |