队伍编号	MC2302848
题号	A

基于贪心策略的量子计算模型在信用评分卡组合优化中的应用研究

摘 要

本文的研究主要集中在金融领域中的信用评估问题，并探讨了将量子计算和贪心策略引入信用评分卡优化问题的潜在优势和应用前景。传统的计算方法存在计算复杂度高、求解效率低等问题，而量子计算作为一种新兴的计算模型，具有并行计算和量子态叠加特性，有望在信用评估中实现更准确、更高效的求解。

首先，介绍并分析了问题 1、问题 2 和问题 3，这些问题涉及在给定多个信用评分卡和对应阈值的情况下，选择最优的组合以最大化银行的收入。问题 1 考虑从 100 个信用评分卡中选择 1 个信用评分卡和对应的阈值；问题 2 考虑从 3 种已选定的信用评分卡中选择阈值；问题 3 考虑从 100 个信用评分卡中任选 3 种信用评分卡并选择对应的阈值。这些问题的目标是搜索阈值对应的通过率和坏账率，在满足约束条件的前提下，最大化银行的贷款利息收入。

接着，将这些问题转化为 QUBO（二次无约束二进制优化）形式，其中目标是最大化最终收入，约束条件包括评分卡组合和阈值之间的关系。通过将问题映射为 QUBO 形式，可以充分利用量子计算的并行计算和量子态叠加特性，从而在信用评分卡优化问题中实现更准确、高效的求解。

之后，探讨了如何利用模拟退火算法来求解这些 QUBO 问题。模拟退火是一种基于概率的优化算法，可以在搜索解空间时避免陷入局部最优解，并在较短的时间内找到全局最优解。介绍了模拟退火算法的基本原理和流程，并探讨了如何在信用评分卡组合优化问题中应用模拟退火算法进行求解，还对比了梯度下降法和 Tabu 算法的求解效果。

最后，为了进一步降低运算复杂度，我们设计了一种基于贪心策略的 QUBO 求解方案。该方案将贪心策略，模拟退火算法与 QUBO 结合起来，充分发挥其优势，极大地提升了问题三这种大样本数据下的信用评估精确性和效率，达到了普通移动设备下的分钟级别精准求解。

综上所述，本文通过量子计算和贪心策略求解了三个信用评分卡优化问题，并且探讨了其潜在优势和应用前景。假定信用评分卡和阈值编号从 1 开始，这里以（信用评分卡编号，阈值编号）为示例。得出**问题一最佳信用评分卡和阈值编号为（49，1）。问题二的三张信用评分卡最佳阈值组合编号依次为（1，8），（2，1），（3，2）。问题三的最佳信用评分卡和对应的最佳阈值组合编号依次为（8，2），（33，6），（49，3）。**

结果证明通过将问题转化为 QUBO 形式并应用贪心策略和模拟退火算法进行求解，可以实现更准确、更高效的信用卡评估决策。未来的研究可以进一步深入探讨量子计算在金融领域的应用，并解决相关的技术和实施挑战，以推动量子计算在金融风险管理和贷款决策中的应用。

关键字：组合优化、信用风险建模、QUBO、模拟退火算法，贪心策略

目录

目录 2

一、问题提出 1

1.1 背景 1

1.2 问题重述 1

二、基本假设 2

三、符号说明 2

四、问题分析 3

4.1 数据分析 3

4.2 问题分析 4

4.3 模型准备 5

五、问题一模型的建立与求解 6

5.1 基于 QUBO 的方法分析 6

5.1.1 QUBO 形式转换 7

5.1.2 模拟退火算法 8

5.2 模型验证与分析 9

5.2.1 模型的结果分析 9

六、问题二模型的建立与求解 9

6.1 基于 QUBO 的方法分析 9

6.1.1 QUBO 形式转换 11

6.2 贪心方法分析 11

6.3 结合 QUBO 的贪心方法分析 12

6.4 模型验证与分析 13

6.4.1 模型结果分析 13

七、问题三模型的建立与求解 14

7.1 基于 QUBO 的方法分析 14

7.1.1 QUBO 形式转换 16

7.2 贪心方法分析 16

7.3 结合 QUBO 的贪心方法分析 17

7.4 模型验证与分析 18

7.4.1 模型的结果分析 18

7.4.2 算法的对比分析 18

7.4.3 参数对模拟退火算法的影响 19

八、评价与改进 20

参考文献 20

附录 20

一、问题提出

1.1 背景

信用评分卡是在金融领域常用的风险评估工具，但是传统方法存在模型复杂度和数据处理等方面的局限性。因此，研究者开始探索将量子计算引入信用评分卡优化问题中，QUBO 模型作为一种数学模型被引入到金融领域的信用评分卡优化问题中，并结合模拟退火算法和贪心策略进行求解，以提高信用风险评估的精度和效果。充分利用量子计算的并行计算和量子态叠加特性，可以在信用评分卡的选择与阈值优化问题中实现更加准确和高效的求解。这种新颖的方法有望为金融机构提供更加高效和准确的信用评估工具，从而改进风险管理和提升业务利润。当然，在实际应用中，还需要进一步研究和验证其可行性和效果，并解决可能遇到的问题和挑战。

将量子计算引入金融领域的信用评分卡优化问题具有潜在的优势和应用前景。该方法通过 QUBO 模型、模拟退火算法和贪心策略的结合，可以在信用评分卡的选择与阈值优化问题中实现更加准确和高效的求解。随着量子计算技术的不断进步，这种新颖的方法有望为金融机构提供更加高效和准确的信用评估工具，从而改进风险管理并提升业务利润。

1.2 问题重述

问题 1：在 100 个信用评分卡中选择一个评分卡和对应的阈值，以最大化银行的收入。已知每个评分卡的通过率和坏账率，假设银行的贷款利息收入率为 8%。我们需要将该问题转化为 QUBO 形式，其中目标是最大化贷款利息收入，约束条件为选择的评分卡和阈值之间的关系。可以使用量子退火算法对该问题进行求解。

问题 2：在已经选定了 3 种信用评分卡的情况下，需要选择对应的阈值，以最大化银行的收入。已知每个评分卡的通过率和坏账率，假设银行的贷款利息收入率为 8%。我们需要将该问题转化为 QUBO 形式，其中目标是最大化贷款利息收入，约束条件为选择的评分卡和阈值之间的关系。可以使用量子退火算法对该问题进行求解。

问题 3：在 100 个信用评分卡中，任选三种信用评分卡，并选择对应的阈值，以最大化银行的收入。已知每个评分卡的通过率和坏账率，假设银行的贷款利息收入率为 8%。我们需要将该问题转化为 QUBO 形式，其中目标是最大化贷款利息收入，约束条件为选择的三种评分卡和对应阈值之间的关系。可以使用量子退火算法对该问题进行求解。

二、基本假设

在对该问题进行建模时，我们可以基于以下假设：

假设每个信用评分卡的通过率和坏账率是已知的，并且与银行的历史数据相关。通过分析历史数据，可以建立评估通过率和坏账率的模型，为信用评分卡的建模提供基础。

假设组合的收益和损失可以通过公式计算得出，包括贷款利息收入、通过率、坏账率和贷款金额等因素。通过定义合适的公式，可以对不同信用评分卡规则的组合进行评估，从而选择最优的组合。

假设银行贷款的金额固定为 1000000 元，并且银行贷款利息收入率为 8%。这为计算组合的收益和损失提供了基本参数，并且在优化信用评分卡规则时起到了参考作用。

假设信用评分卡的选择和阈值对应的通过率和坏账率之间是独立的，即每个阈值的选择不会影响其他阈值的结果，并且每个评分卡只能选择一个阈值作为其判断标准。这可以作为简化模型的假设，降低模型的复杂性。

假设通过率与收入成正比例关系，坏账率与收入成反比例关系。这为计算组合的收益和损失提供了基本假设，并在选择最优信用评分卡规则时起到了指导作用。

三、符号说明

这里只列出部分通用符号，个别模型单独使用符号或下标会在下文中具体说明。

符号	说明
C	贷款资金
r	贷款利息收入率
n	信用评分卡个数
m	每个信用评分卡的阈值个数
P	总通过率
p	单个阈值的通过率
Q	总坏账率
q	单个阈值的坏账率

四、问题分析

4.1 数据分析

结合附件一中的数据，可以了解到其包含 100 个信用评分卡的数据。每个评分卡包含 10 个阈值，每个阈值对应一个通过率和坏账率，即包含 1000 个通过率和坏账率的组合。我们对其进行了分析如图 4.1.1 所示，可以看出通过率分布在[0.7,0.99]之间，在（0.874,0.932]的分布数量最多。而坏账率分布在[0.003,0.088]之间，在（0.02,0.932]的分布数量最多但与在(0.037,0.054]中的数量相当。同时我们对通过率和坏账率的散点对应分布进行了展示，如图 4.1.2 所示，可以看出大多数分布都集中在中间部分，而对于本文的分析我们更希望取得总通过率大且总坏账率小的值，即尽量靠近图例的右下角。

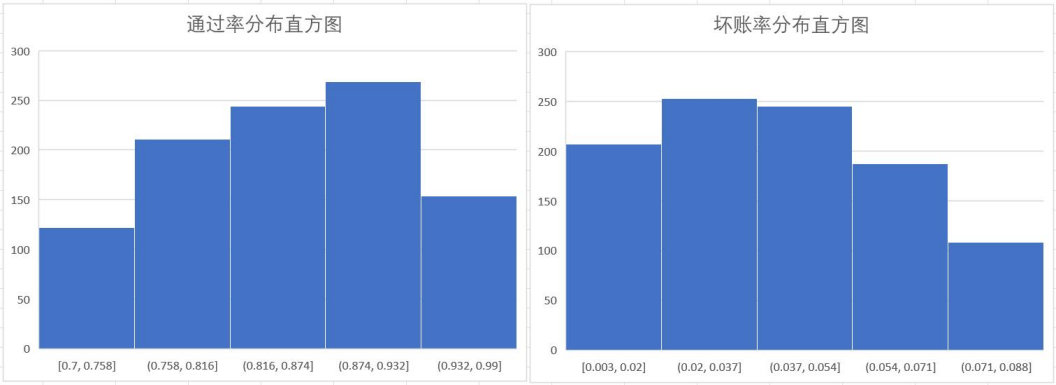

图 4.1.1 通过率和坏账率分布直方图

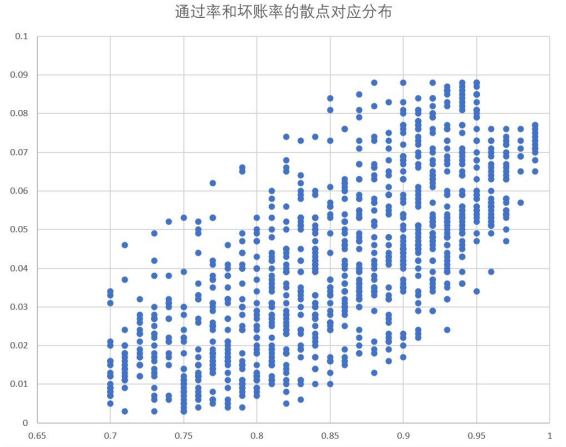

图 4.1.2 通过率和坏账率的散点对应分布图

4.2 问题分析

这三个问题都是针对银行场景下的信用评分卡组合问题进行求解。在实际应用中，通过选择最佳的信用评分卡和相应的阈值，帮助银行实现最佳的风险控制策略，从而最大化收益。这三个问题分别探讨了不同情况下的信用评分卡组合问题，包括信用评分卡的选择、阈值的优化以及最终银行收益的计算。图 4.2.1 展示了我们对问题的分析流程和解决方案的展示。

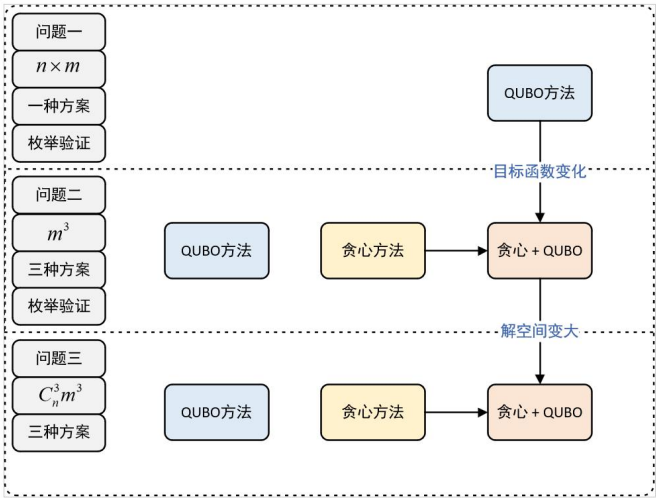

图 4.2.1 问题分析图

对于第一个问题，我们需要从 100 张信用评分卡中选择一张，并在其 10 个阈值中选择最优的一个，以最大化总收入。这个问题可以直接转化为 QUBO 形式，并使用模拟退火算法进行求解。在实际应用中，这种方法可用于快速优化大规模信用评分卡数据。

对于第二个问题，我们已经确定了三种信用评分卡的选择，并需要在每种信用评分卡中选择对应的阈值，以最大化总收入。这是一个组合优化问题，我们提出了三种解决思路：①将问题转换为 QUBO 形式并利用模拟退火算法直接求解；②结合贪心策略，选择当前对组合最优的解来达到最终组合最优；③使用贪心策略和 QUBO 减少问题规模，改变问题的目标函数形式，并通过模拟退火、梯度下降等算法选择当前最优解。

对于第三个问题，我们需要任选三种信用评分卡，并选择对应的阈值，以最大化银行的收益。由于需要在 100 张信用评分卡中选择三种，并且每种评分卡有 10 个不同的阈值可供选择，该问题的规模较大。我们提出了三种解决思路：①将问题转换为 QUBO 形式并直接求解，但由于问题规模太大，一般计算机难以实现；②结合贪心策略，在当前组合中选择最优解来达到最终的最优解；③使用贪心策略和 QUBO 相结合来减少问题规模，并通过模拟退火、梯度下降等算法选择当前最优解，从而使得问题在一般计算机上可以实现求解。

由于三个问题都需要对 QUBO 形式进行转换，我们真对本应用的 QUBO 模型的分析如图 4.3.1 所示，最主要的是我们需要将公式转为 QUBO 形式，并通过相应的量子算法来对其求解。我们使用的算法有模拟退火、梯度下降和 Tabu 方法，并对每种方法进行了分析和对比。

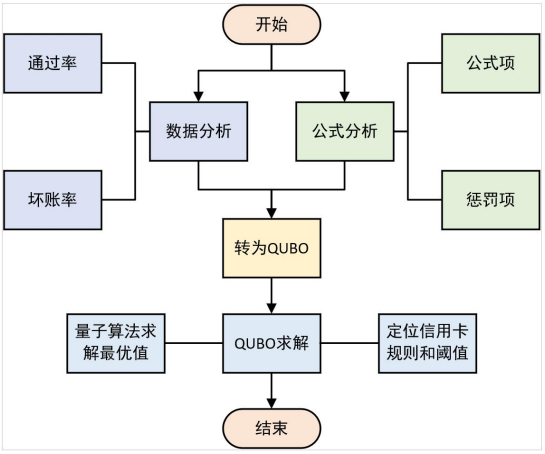

图 4.2.2 使用 QUBO 模型求解流程图

4.3 模型准备

根据以上对问题分析，我们假设贷款资金为 C，银行贷款利息收入率为 r，由于每一信用评分卡有且只可选择 1 个阈值，且每个阈值对应不同的通过率和坏账率。信用评分卡组合后的总通过率为 P,总坏账率为 Q。
因此本次贷款利息收入可以表述为：贷款资金×利息收入率×总通过率×（1－总坏账率），即：

$$CrP(1-Q)$$

由坏账带来的坏账损失为贷款资金×总通过率×总坏账率，即：

$$CPQ$$

那么可以计算出银行的最终收入为贷款利息收入－坏账损失，即：

$$R = CrP(1-Q) - CPQ$$

合并同类项可得：

$$R = CrP - (Cr + C)PQ$$

那么针对问题一到问题三其实都是对找到最优的 P 和 Q 来使得 R 取最大值。针对该问题，当 C=1000000 且 r=8%时，可以画出关于 R 的相对于 P 和 Q 变量的平面图。如图 4.3.1 所示，可以看出当总通过率越大，总坏账率越小最终收入越大。因此我们应该设计合理的算法尽量保证所选的阈值组合使得 P 尽量大 Q 尽量小。

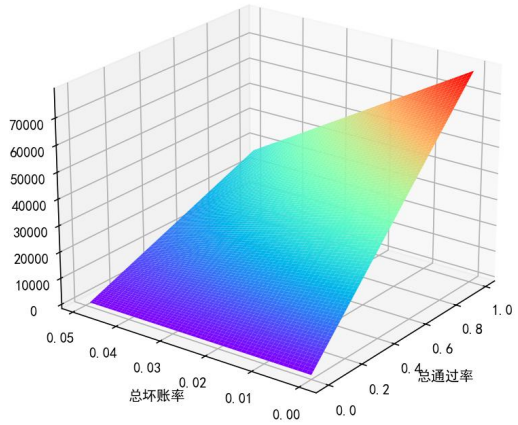

图 4.3.1 总通过率和总坏账率对结果的影响

针对以上分析我们对问题一到问题三的模型建立和求解做出了详细的介绍。

五、问题一模型的建立与求解

5.1 基于 QUBO 的方法分析

问题一是在所有的阈值组合中找到一个最优解。假设有$n = 100$ 个信用评分卡，每个评分卡有 $m = 10$ 个阈值。则其可行解的数量为：

$$n \times m = 1000$$

针对该规模的问题我们设计了如图 5.1.1 的模型分析思路。

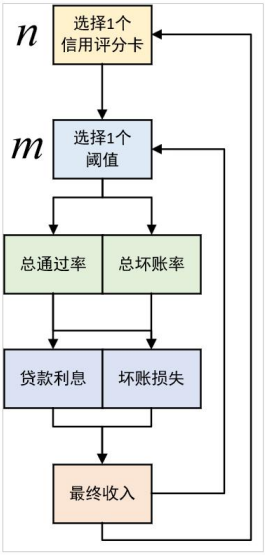

图 5.1.1 问题一的模型分析思路

在对模型的建模中我们引入了 0,1 变量x_i，表示第 $\lfloor i/m \rfloor$+1 个评分卡，第 $i\ mod\ n$ (表示取余数)个阈值是否被选。则总通过率和总坏账率可以表示为：

$$P = \sum_{i=1}^{n\times m} p_i x_i \ , Q = \sum_{i=1}^{n\times m} q_i x_i$$

问题一是在$n \times m$个阈值中找到一个最优的阈值，即变量x需要满足以下条件：

$$\sum_{i=1}^{n\times m} x_i = 1$$

将总通过率和总坏账率带入总收入公式可以得 R 的表示为:

$$R = \sum_{i=0}^{n\times m} \left(Crp_i x_i - (Cr + C)p_i q_i x_i^2\right)$$

5.1.1 QUBO 形式转换

QUBO（二次无约束二值优化）是一种 NP 问题的数学表述方式，其目标是将一个约束条件简单的二次方程多项式转换为只包含 0 和 1 的二进制变量的多项式，通常情况下，这个多项式可以被表示为二进制变量的线性组合和二次项的组合。其一般形式为：

$$\min_{x\in\{0,1\}^n} \left(\sum_{i=1}^{n}\sum_{j=1}^{n} Q_{i,j} x_i x_j + \sum_{i=1}^{n} c_i x_i \right)$$

为了将第一问转换为 QUBO 形式进行求解，需要将最大问题转为求最小问题，即可我们直接取其相反数来进行表示。由于x_i的取值为 0,1 则$x_i = x_i^2$，将公式进行替换并合并同类项可得到目标函数的表达式为：

$$-R = -\sum_{i=0}^{n\times m} (Cr - (Cr + C)q_i)p_i\, x_i^2$$

模型需要满足在 n×m 中选择一个阈值，即约束条件为：

$$H_1 = \left(\sum_{i=1}^{n\times m} x_i - 1\right)^2$$

将目标函数和约束条件带入可以转换为最终的 QUBO 形式$H = min(-R + \lambda \mathrm{H}_1)$，其中 λ 是惩罚项的系数，需要根据具体情况进行调整。即最终的 QUBO 完整表达式为：

$$H = \min_{x}\left(-\sum_{i=0}^{n\times m}(Cr-(Cr+C)q_i)p_i x_i^2 + \lambda\left(\sum_{i=1}^{n\times m}x_i - 1\right)^2\right)$$

在实际求解中，λ可以取一个比较大的值，比如 100000，以确保惩罚项的作用。由于问题中需要求解最优的阈值，可以将 p_i, q_i看作是已知的，而x_i则是未知的，直接对 x_i进行求解。本文使用的 dwave-samplers 库，将所推出的目标函数和约束条件即可转为其 Q 矩阵进行求解。

5.1.2 模拟退火算法

问题一对于 QUBO 形式的求解算法采用了模拟退火算法。模拟退火是一种全局启发式搜索优化算法，用于在搜索空间中找到最优解。它的基本原理是模拟物理学中的退火过程，通过温度参数控制搜索过程中接受次优解的概率，从而避免陷入局部最优解，如图 5.1.2 所示。其思想是将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。

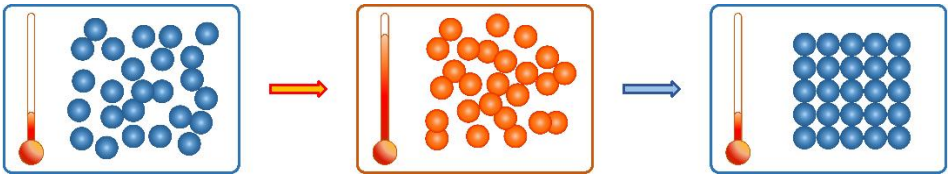

图 5.1.2 模拟退火物理过程图解

其算法流程如下：

1. 初始化一个解 x_0 和一个初始温度 T_0 。
2. 在该温度下，进行迭代，每次迭代进行以下操作：

   a. 从当前解 x 的邻域中随机选择一个新解 $x^{'}$ 。

   b. 计算 $x^{'}$ 的目标函数值 $f(x^{'})$ 和 x 的目标函数值 $f(x)$ 。

   c. 根据 Metropolis 准则判断是否接受新的解，即如果 $f(x^{'}) < f(x)$ ，则以概率 1 接受 $x^{'}$作为新的解；否则以概率 $e^{-\Delta f/T}$ 接受 $x^{'}$ ， 其中 $\Delta f = f(x^{'}) - f(x)$ 。

3. 更新温度 T ，并重复步骤 2 直到温度满足停止条件

具体算法流程如图 5.1.3 所示。

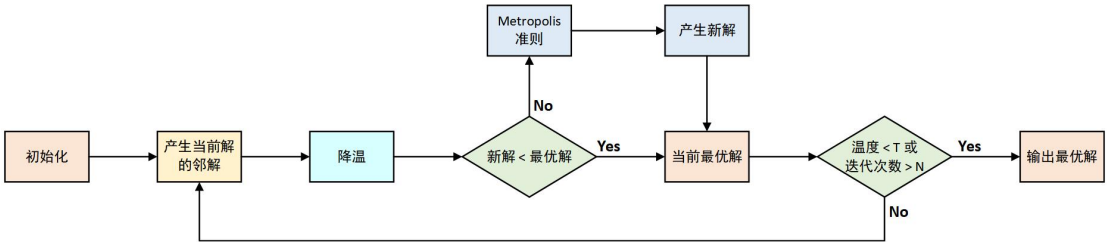

图 5.1.3 模拟退火算法的流程图

由上述算法可知，其迭代效果受到初始温度、降温速率和终止温度三个参数的影响。初始温度通常设置为一个较高的值，以保证在搜索的早期能够接受一些劣解，从而有更大的概率跳出局部最优解。终止温度通常设置为一个较小的值，当温度降到终止温度时，搜索停止，此时得到的解就是算法得到的最优解。而降温速率通常设置为一个小于 1 的常数，降温速率越慢，则搜索的时间越长，但是搜索范围也就越广，有更大的概率找到全局最优解。因此需要合理设置三个参数以便能够达到快速达到最优解。

5.2 模型验证与分析

5.2.1 模型的结果分析

方案	选取结果	最终收入	时间花费
QUBO 模型+模拟退火	(49，1)	61172	13s
枚举（验证）	(49，1)	61172	<0.1s

注：选取结果中(i, j)表示选取第 i 张卡的第 j 个阈值

为了验证问题一的 QUBO 模型的正确性，我们采用了循环枚举所有单张卡和单个阈值组合的方法，计算出每种组合对应的最终收入，并记录其中的最大值。同时，我们还比较了枚举法和模拟退火算法实现的结果。由于问题规模较小，循环枚举方法在时间效率上表现更好。

这种验证方法能够确保我们得到的 QUBO 模型的解是准确的，并验证了该模型的可行性。同时，由于循环枚举方法的实现简单，它也可以作为我们后续考虑更复杂问题时的一个基准对比。但对于规模更大的问题，循环枚举方法会面临组合数爆炸的问题，导致计算时间和空间成本快速增加。

六、问题二模型的建立与求解

6.1 基于 QUBO 的方法分析

第二问已经选定了数据集中给出的信用评分卡 1、信用评分卡 2、信用评分卡 3 这三种规则，来找到最优的阈值组合。假设有$n = 100$ 张信用评分卡，每个信用评分卡有 $m = 10$ 个阈值可以知道该模型的可行解的数量为:

$$C_m^1 C_m^1 C_m^1 = m^3 = 1000$$

针对该规模的问题我们设计了如图 6.1.1 的模型分析思路。

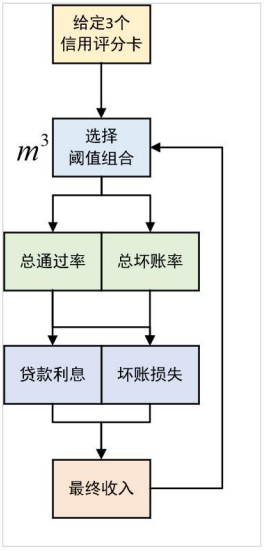

图 6.1.1 问题二的模型分析思路

第二问是从m^3个阈值组合中找到最终收入最高的一个组合，我们引入 0,1 变量x_i 来表示第i个阈值组合是否被选择，所有的阈值组合都是通过给定的三个信用评分卡的阈值排列组合得到的，因此是确定的。其总体通过率可以表示为：

$$P = \sum_{i=1}^{m^3} P_i x_i,\ P_i = p_i^1 p_i^2 p_i^3$$

其中P_i表示第i个阈值组合对应的总通过率，p_i^1, p_i^2, p_i^3表示第i个阈值组合对应的三个通过率。总坏账率可以表示为：

$$Q = \sum_{i=1}^{m^3} Q_i x_i,\ Q_i = \frac{1}{3}(q_i^1 + q_i^2 + q_i^3)$$

其中Q_i表示第i个阈值组合对应的总坏账率，q_i^1, q_i^2, q_i^3表示第i个阈值组合对应的三个坏账率。由于我们是在m^3个阈值组合中找到一个适合的阈值组合，即我们的变量x需要满足以下条件：

$$\sum_{i=1}^{m^3} x_i = 1$$

将总通过率和总坏账率带入总收入公式可以得R为:

$$R = \sum_{i=0}^{m^3} \left(CrP_i x_i - (Cr + C)P_i Q_i x_i^2\right)$$

6.1.1 QUBO 形式转换

将以上表达式转换为 QUBO 形式进行求解，其目标函数的表达式为：

$$-R = -\sum_{i=0}^{m^3} (Cr - (Cr + C)Q_i)P_i x_i^2$$

模型需要满足在m^3中选择一个阈值组合，即约束条件为：

$$H_1 = \left(\sum_{i=1}^{m^3} x_i - 1\right)^2$$

可以转换为最终的 QUBO 形式为$H = min(-R + \lambda \mathrm{H}_1)$，其中 λ 是惩罚项的系数。即最终的 QUBO 完整表达式为：

$$H = \min_x \left(-\sum_{i=0}^{m^3} (Cr - (Cr + C)Q_i)P_i x_i^2 + \lambda \left(\sum_{i=1}^{m^3} x_i - 1\right)^2 \right)$$

通过使用 dwave-samplers 库，将所推出的目标函数和约束条件即可转为其 Q 矩阵进行算法求解。并使用模拟退火算法来选择最优阈值组合。

6.2 贪心方法分析

首先我们对其进行相关数学分析。假设任选了满足条件的三个阈值，则其三个通过率分别为p_1, p_2, p_3，三个坏账率分别为q_1, q_2, q_3，则可求得其总通过率和总坏账率为：

$$P = p_1 p_2 p_3$$
$$Q = \frac{1}{3}(q_1 + q_2 + q_3)$$

将其带入公式$R = CrP - (Cr + C)PQ$可以得到组合阈值的最终收入展开式：

$$R = Crp_1 p_2 p_3 - \frac{1}{3}(Cr + C)p_1 p_2 p_3 (q_1 + q_2 + q_3)$$

从公式可以看出R受到p_1, p_2, p_3和q_1, q_2, q_3的组合影响。现我们做出如下假设，当已经确定了两个阈值对应的通过率p_1, p_2和坏账率q_1, q_2时，只需对最后一个阈值的通过率p_3和坏账率q_3进行分析来找到目标函数的最优解。即我们可以将第二问拆分为解决找到单个最优的子问题来求解。且我们的子问题的最优解可以堆导出整个问题的最优解。这正好满足了贪心思想。

于是通过以上数学分析，设计了基于贪心策略的思路具体如下，①程序初始化将三个信用评分卡的第一个阈值作为最优阈值组合；②固定第一个和第二

个的最优阈值并在第三个信用评分卡中找到符合目标函数的最优解得到第三个最优阈值；③固定第三个和第一个阈值的最优解并在第二个信用评分卡中找到符合目标函数的最优解得到第二个最优阈值；④固定第二个和第三个阈值的最优解并在第一个信用评分卡中找到符合目标函数的最优解得到第一个最优阈值；⑤通过多轮迭代确定最优阈值组合。具体如下图所示：

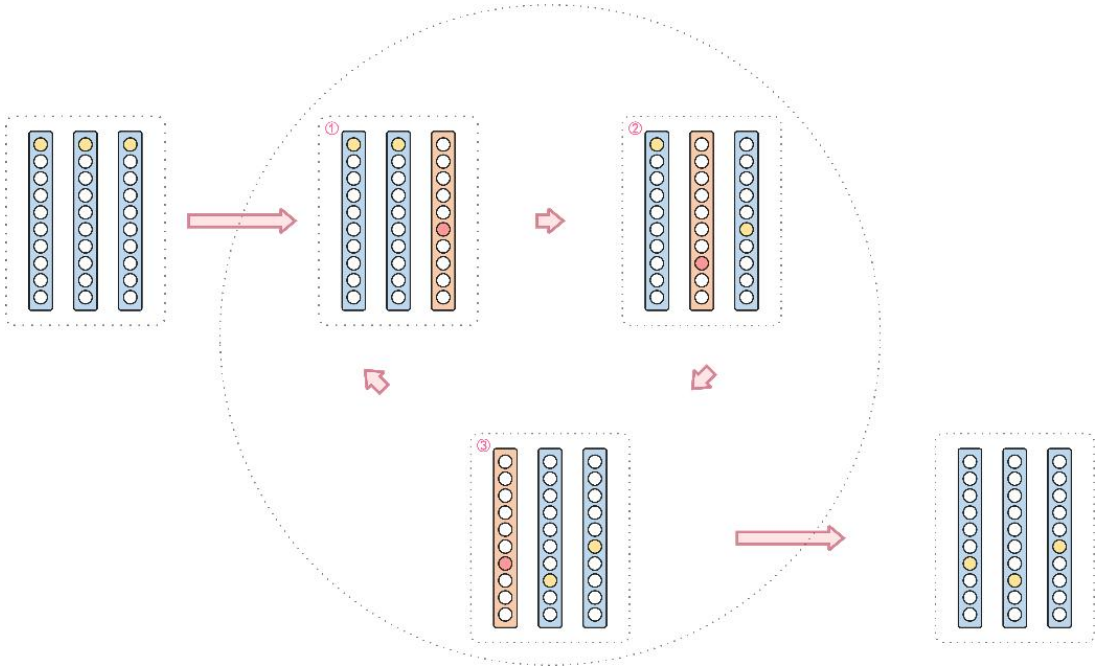

图 6.2.1 问题二的贪心策略流程

基于贪心方法的求解目标为固定两个阈值找第三个阈值的最优。可列出如下公式：

$$max(R) = Crp_1p_2p_3 - \frac{1}{3}(Cr + C)p_1p_2p_3(q_1 + q_2 + q_3)$$

即将通过率p_1, p_2和坏账率q_1, q_2视为已知的常数即可，基于纯贪心方法我们只需在剩下的一个信用评分卡中找到一个最优阈值对应的通过率p_3和坏账率q_3即可，再依次迭代。由于在本问中的每个信用评分卡对应的阈值个数为 10，即我们可以通过枚举来实现找到 10 个的最优阈值。

6.3 结合 QUBO 的贪心方法分析

当然随着问题规模的增大，也可以采用 QUBO 的形式并结合贪心策略和相关算法进行求解，但与直接使用 QUBO 来分析三个信用评分卡的阈值组合方式来说得到了大大的简化。因为基于贪心的方法只需要分析单个信用评分卡的最优阈值即可。同时我们结合了第一问的 QUBO 转换方法将$n \times m$个可行解空间变为第二题的m个可行解空间，即将第一问的在 n 个信用评分卡变为本问的单个评分卡中找最优解，同时我们替换了其目标函数为组合最优的表示。因此我们引入 0,1 变量x_i 来表示未选信用卡第i个阈值是否被选。其总体通过率可以表示为：

$$P = \sum_{i=1}^{m} P_1 P_2 P_3^i x_i$$

其中P_3^i表示未选信用卡第i个阈值的通过率，P_1, P_2表示选中的两个信用卡的最优阈值的通过率。总坏账率可以表示为：

$$Q = \frac{1}{3}\sum_{i=1}^{m} (Q_1 + Q_2 + Q_3^i)\, x_i$$

其中Q_3^i表示未选信用卡第i个阈值对应的坏账率，Q_1, Q_2表示选中的两个信用卡的最优阈值的坏账率。由于我们是在m个阈值中找到一个适合的阈值，即我们的变量x需要满足以下条件：

$$\sum_{i=1}^{m} x_i = 1$$

将总通过率和总坏账率带入总收入公式R可得到最终的 QUBO 完整表达式为：

$$H = \min_x \left(-\sum_{i=0}^{m} \left(Cr - \frac{1}{3}(Cr + C)(Q_1 + Q_2 + Q_3^i) \right) P_1 P_2 P_3^i\, x_i^2 + \lambda \left(\sum_{i=1}^{m} x_i - 1 \right)^2 \right)$$

将得到的 QUBO 表达式后可将其转换为 Q 矩阵并通过 dwave-samplers 库来进行建模，我们使用了模拟退火算法并结合贪心策略来选择最优的阈值组合。

6.4 模型验证与分析

6.4.1 模型结果分析

在求解问题二过程中， 对于 QUBO 方法进行模拟退火算法实验，最终发现在 1000 个阈值组合中，索引为 811 的阈值组合对应的x取值为 1，其他为 0，映射回原有的阈值选择是第一张卡选第 8 个，第二张卡选第 1 个，第三张卡选第 2 个。对于贪心+QUBO 算法，其迭代结果如下所示：

迭代次数	模拟退火算法	
	选取结果	最终收入
1	(1，1) (2，1) (3，2)*	26527.38
2	(1，1) (2，1) (3，2)	26527.38
3	(1，8)*	27914.8

	(2，1) (3，2)	2
4	(1，8) (2，1) (3，2)	27914.82
...	...	...

从表中可以发现其收敛于第一张卡选第 8 个，第二张卡选第 1 个，第三张卡选第 2 个。为了验证这几种方法的准确性，我们通过循环去遍历了这三张卡所有阈值的组合，并获得了最大值，所有的结果如下表所示：

	卡 1 阈值	卡 2 阈值	卡 3 阈值	最大值
QUBO	8	1	2	27914.82
贪心	8	1	2	27914.82
贪心+QUBO	8	1	2	27914.82
枚举（验证）	8	1	2	27914.82

通过枚举验证，可以发现三种方法都能准确找到最优，也证明了我们模型的可行性。针对不同的问题规模，我们可以采用不同的方法，以获得最好的性能。

七、问题三模型的建立与求解

7.1 基于 QUBO 的方法分析

第三问是一个组合优化问题。即在 100 个信用评分卡中找到三张信用评分卡，并设置合理的阈值，使得最终收入最多。假设有$n = 100$ 张信用评分卡，每个评分卡有 $m = 10$ 个阈值，可以知道该模型的可行解的数量为:

$$C_n^3 C_m^1 C_m^1 C_m^1 = C_n^3 m^3 = 161700000$$

针对该数量级别的任务，普通计算机很难实现，因此我们只对其进行了 QUBO 的形式转换和分析，最终 QUBO 形式可结合模拟退火算法、梯度下降等算法在更好的计算机实现。针对该规模的问题我们设计了如图 7.1.1 的模型分析思路。

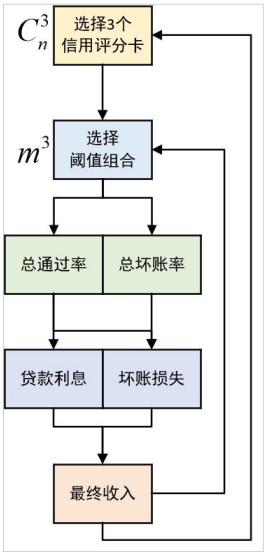

图 7.1.1 问题三的模型分析思路

首先可以将该问题转换为第二问的阈值组合方法，从$C_n^3 m^3$个阈值组合中找到最终收入最高的一个组合，我们引入 0,1 变量$x_{i,j,k,l}$ 来表示当选择第i,j,k的信用评分卡的第l种组合是否被选择。所有的阈值组合都是通过i,j,k三个信用评分卡的阈值排列组合得到的，因此是确定的。其总体通过率可以表示为：

$$P = \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n}\sum_{l=1}^{m^3} P_{i,j,k,l}^1 P_{i,j,k,l}^2 P_{i,j,k,l}^3 x_{i,j,k,l}$$

其中$P_{i,j,k,l}^3, P_{i,j,k,l}^2, P_{i,j,k,l}^3$表示$i,j,k,l$对应的阈值组合的三个通过率，总坏账率可以表示为：

$$Q = \frac{1}{3}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n}\sum_{l=1}^{m^3}\left(Q_{i,j,k,l}^1 + Q_{i,j,k,l}^2 + Q_{i,j,k,l}^3\right)x_{i,j,k,l}$$

其中$Q_{i,j,k,l}^1, Q_{i,j,k,l}^2, Q_{i,j,k,l}^3$表示$i,j,k,l$对应的阈值组合的三个坏账率。由于模型是在$C_n^3 m^3$个阈值组合中找到一个最优的阈值组合，即我们的变量$x$需要满足以下条件：

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n}\sum_{l=1}^{m^3} x_{i,j,k,l} = 1$$

将总通过率和总坏账率带入总收入公式可以得R为:

$$R = \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n}\sum_{l=1}^{m^3}\left(Cr - (Cr + C)\left(Q_{i,j,k,l}^1 + Q_{i,j,k,l}^2 + Q_{i,j,k,l}^3\right)\right) P_{i,j,k,l}^1 P_{i,j,k,l}^2 P_{i,j,k,l}^3 x_{i,j,k,l}^2$$

7.1.1 QUBO 形式转换

将以上表达式转换为 QUBO 形式进行求解，取目标函数的相反数来转为求最小形式，其表达式为：

$$-R = -\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n}\sum_{l=1}^{m^3}\left(Cr-(Cr+C)\left(Q_{i,j,k,l}^{1}+Q_{i,j,k,l}^{2}+Q_{i,j,k,l}^{3}\right)\right)P_{i,j,k,l}^{1}P_{i,j,k,l}^{2}P_{i,j,k,l}^{3}x_{i,j,k,l}^{2}$$

模型需要满足在$C_n^3 m^3$个解空间中选择一个阈值组合，即约束条件为：

$$H_1 = \left(\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n}\sum_{l=1}^{m^3}x_{i,j,k,l}-1\right)^2$$

可以转换为最终的 QUBO 形式$H = min(-R + \lambda \mathrm{H}_1)$，其中 λ 是惩罚项的系数，需要根据具体情况进行调整。即最终的 QUBO 完整表达式为：

$$H = \min_x\left(-\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n}\sum_{l=1}^{m^3}\left(Cr-(Cr+C)\left(Q_{i,j,k,l}^{1}+Q_{i,j,k,l}^{2}+Q_{i,j,k,l}^{3}\right)\right)P_{i,j,k,l}^{1}P_{i,j,k,l}^{2}P_{i,j,k,l}^{3}x_{i,j,k,l}^{2} + \lambda\left(\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=j+1}^{n}\sum_{l=1}^{m^3}x_{i,j,k,l}-1\right)^2\right)$$

通过使用 dwave-samplers 库，将所推出的目标函数和约束条件即可转为其 Q 矩阵进行算法求解。但是我们在求解的过程中，计算机并不能负载该规模的问题。在已知其 QUBO 表达式的前提下可以使用更高性能的计算机来结合相关算法进行求解。因此我们需要寻找一种高效的算法来搜索最优解，将问题进行简化。一种可行的思路是根据第二问的数学分析，将模型转换为贪心优化问题并结合 QUBO 形式来进行最终求解。因此我们提出了基于贪心策略的两种方案。

7.2 贪心方法分析

与第二问的想法类似当已经确定了两个阈值对应的通过率p_1, p_2和坏账率q_1, q_2时，只需对最后一个阈值的通过率p_3和坏账率q_3进行分析来找到目标函数的最优解。但是不同的是，需要在剩下的$n-2$ 个信用评分卡中找到组合最优值。数学分析与第二问相同。

针对第三问具体思路如下，①程序初始化将任选三个信用评分卡的第一个阈值作为最优阈值组合；②固定第一个和第二个的最优阈值并在剩下的信用评分卡中找到符合目标函数的最优解得到第三个最优阈值；③固定第三个和第一个阈值的最优解并在剩下的信用评分卡中找到符合目标函数的最优解得到第二个最优阈值；④固定第二个和第三个阈值的最优解并在剩下的信用评分卡中找到符合目标函数的最优解得到第一个最优阈值；⑤通过多轮迭代确定最优阈值组合。具体如下图所示：

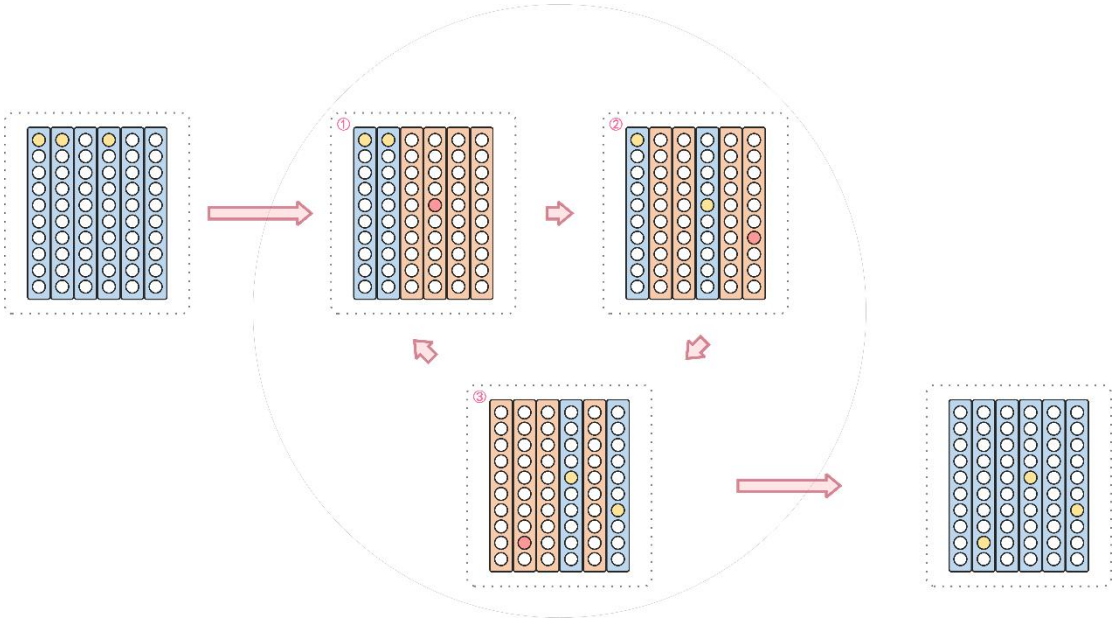

图 7.2.1 问题三的贪心策略流程

问题三的求解目标为固定两个阈值在剩下的信用评分卡中找第三个阈值的最优。可列出如下公式：

$$max(R) = Crp_1p_2p_3 - \frac{1}{3}(Cr + C)p_1p_2p_3(q_1 + q_2 + q_3)$$

即将通过率p_1, p_2和坏账率q_1, q_2视为已知的常数即可，基于纯贪心方法我们只需要在剩下$n-2$个信用评分卡中找到一个最优阈值对应的通过率p_3和坏账率q_3即可，再依次迭代。该方法在每一步的可行解数量为$(n-2)m$，针对本问题即98×10=980。即依旧可以通过枚举法来实现对980个可行解的选择。

7.3 结合 QUBO 的贪心方法分析

在该方法中我们结合了问题一关于QUBO建立的思路和第二题与贪心结合方法的思路来进行求解。具体来说将第一题的$n \times m$个可行解空间和第二题的m个可行解空间转为本问的$(n-2)m$个可行解空间，而目标函数使用的第二问的组合函数。通过在解空间中不断替换并迭代找到最优组合阈值。因此我们引入0,1变量x_i 来表示在$(n-2)m$的可行阈值是否被选。其总通过率、总坏账率和约束可以表示为：

$$P = \sum_{i=1}^{(n-2)m} P_1P_2P_3^i x_i$$

$$Q = \frac{1}{3}\sum_{i=1}^{(n-2)m} (Q_1 + Q_2 + Q_3^i)\, x_i$$

$$\sum_{i=1}^{(n-2)m} x_i = 1$$

将总通过率、总坏账率和约束带入总收入公式R可得到最终的 QUBO 完整表达式为：

$$H = \min_{x}\left(-\sum_{i=0}^{(n-2)m}\left(Cr-\frac{1}{3}(Cr+C)(Q_1+Q_2+Q_3^i)\right)P_1P_2P_3^i x_i^2+\lambda\left(\sum_{i=1}^{(n-2)m}x_i-1\right)^2\right)$$

可以看出与问题二的公式相识，仅替换了其可行解大小。将其公式表示带入库中进行不同算法对比来分析其性能。

7.4 模型验证与分析

7.4.1 模型的结果分析

方案	选取结果	最终收入
QUBO 模型	\	\
贪心方法	(8，2)(33，6)(49，3)	43880.97
贪心+QUBO 模型	(8，2)(33，6)(49，3)	43880.97

注：选取结果中(i, j)表示选取第 i 张卡的第 j 个阈值

由于枚举规模较大，我们并未进行相关枚举实验。因此我们比较了贪心方法和贪心+QUBO 模型的实验结果。可以发现所提出的两种不同方法都能得到相同的结果，则可以证明其一定的准确性。

7.4.2 算法的对比分析

对基于 QUBO 的三种算法，我们做了如下的对比分析：

迭代次数	TUBO		梯度下降		模拟退火	
	选取结果	最终收入	选取结果	最终收入	选取结果	最终收入
1	(2，1) (3，1) **(49，1)***	29453.87	(2，1) (3，1) **(49，2)***	29640.96	(2，1) (3，1) **(33，3)***	29384.52
2	(3，1) (49，1) **(73，2)***	38727.09	(3，1) (49，2) **(54，2)***	37331.67	(3，1) **(8，2)*** (33，3)	40200.09
3	**(4，1)*** (49，1) (73，2)	39810.16	**(16，2)*** (49，2) (54，2)	37670.91	(8，2) **(19，6)*** (33，3)	42905.26
4	(4，1)	40251.32	(16，2)	37670.91	(8，2)	43182.34

	(49，3)*		(49，2)		(19，6)	
	(73，2)		(54，2)		**(49，3)***	
	(4，1)		**(4，1)***		(8，2)	
5	(49，3)	40251.32	(16，2)	38153.87	(19，6)	43182.34
	(73，2)		(49，2)		(49，3)	
	(4，1)		(4，1)		(8，2)	
6	(49，3)	40251.32	(49，2)	39529.79	**(33，6)***	43880.97
	(73，2)		**(73，1)***		(49，3)	
	(4，1)		(4，1)		(8，2)	
7	(49，3)	40251.32	(49，2)	39529.79	(33，6)	43880.97
	(73，2)		(73，1)		(49，3)	
…	**…**		**…**		**…**	

注：*表示该次迭代后选取的最优值，选取结果中(i, j)表示选取第 i 张卡的第 j 个阈值

根据给出的表格，我们可以看到三种不同的优化器在迭代过程中的表现以及最终的收入情况。在第 4 轮贪心迭代后，TUBO 优化器的最优值不再增加，即达到收敛。而梯度下降优化器在第 6 轮贪心迭代后收敛，模拟退火优化器在第 6 轮贪心迭代后收敛。然而，最终的收益表明 TUBO 优化器和梯度下降优化器都陷入了局部最小值。相比之下，模拟退火优化器具有跳出局部最优解的可能性，因为它会以一定的概率接受比当前解更差的解，这样就可以避免陷入局部最优解。因此，我们认为其收敛的值很大可能是最大值。

7.4.3 参数对模拟退火算法的影响

在实验中使用 python 的 dwave 库实现，以下是对模拟退火的参数影响分析。通过波尔兹常量将初始温度和终止温度转换为 β_1 和 β_0，且通过采集区间 $[\beta_1, \beta_0]$ 在对数尺度上均匀分布的数据用于赋值更新温度参数来模拟降温过程，采集个数由参数 num_sweeps 参数决定。基于此，初始温度、降温速率和终止温度三个参数的确定可以转换为 β_1、β_0 和 num_sweeps 参数的设定。为了确定最优参数，做了实验如下：

超参数			收敛的迭代次数	最终收入最优值
β_0	β_1	num_sweeps		
10e-11	10	1000	9	43430.58
10e-11	30	1000	5	43427.38
10e-11	50	1000	6	43880.97
10e-11	100	1000	8	43880.97
10e-10	50	1000	4	42698.65
10e-9	50	1000	4	43853.77
10e-12	50	1000	9	43434.86
10e-10	50	100	6	42950.16
10e-10	50	10000	3	43376.79

在本次实验中，设置参数中，选取 $\beta_0 = 10e-11, \beta_1 = 50, num\_sweeps = 1000$ 时收敛比较快，并且能获得的最优值为 43880.97。

八、评价与改进

优点：

- 利用贪心思想对 QUBO 模型中指数级增长的二进制变量进行压缩，可以在有限的计算资源下对可行解进行搜索。同时，采用模拟退火算法可以有效避免局部最优解陷阱，提高全局搜索能力
- 算法的时间复杂度较低，可以在较短时间内得到较好的结果。
- 算法的空间复杂度较低，不需要大量的存储空间。

缺点：

- 算法的性能高度依赖于温度参数和降温策略的设置，需要进行大量的实验和调试。
- 对于无法使用贪心策略的问题，使用 QUBO 模型构建可能会导致指数级别的二进制变量，从而导致对于复杂问题难以求解。

参考文献

[1] Glover F, Kochenberger G, Hennig R, et al. Quantum bridge analytics I: a tutorial on formulating and using QUBO models[J]. Annals of Operations Research, 2022, 314(1): 141-183.

[2] Morstyn T. Annealing-based Quantum Computing for Combinatorial Optimal Power Flow[J]. IEEE Transactions on Smart Grid, 2022.

[3] Papalitsas C, Andronikos T, Giannakis K, et al. A QUBO model for the traveling salesman problem with time windows[J]. Algorithms, 2019, 12(11): 224.

[4] Zaman M, Tanahashi K, Tanaka S. PyQUBO: Python library for mapping combinatorial optimization problems to QUBO form[J]. IEEE Transactions on Computers, 2021, 71(4): 838-850.

[5] Mandal A, Roy A, Upadhyay S, et al. Compressed quadratization of higher order binary optimization problems[C]//Proceedings of the 17th ACM International Conference on Computing Frontiers. 2020: 126-131.

[6]王宝楠,水恒华,王苏敏等.量子退火理论及其应用综述[J].中国科学:物理学 力学 天文学,2021,51(08):5-17.

[7] 文凯,马寅,王鹏等.基于光量子计算的信用评分特征筛选研究报告[J].网络安全与数据治理,2022,41(09):13-18.DOI:10.19358/j.issn.2097-1788.2022.03.002.

附录

```
import time
import re
from pyqubo import Array, Constraint, Binary
```

```
# from neal import SimulatedAnnealingSampler  # 使用 pyqubo 里面的采样器
from dwave.samplers import SimulatedAnnealingSampler
from dwave.samplers import TabuSampler
from dwave.samplers import SteepestDescentSolver
from dwave.samplers import RandomSampler
import numpy as np
from pyqubo import Placeholder

# 在字典中找到 value 的 key
def get_key(dict, value):
    return [k for k, v in dict.items() if v == value]

# q1 问题主函数,问题一的主要求解函数
def q1_main():
    # 定义一些变量
    L = 1000000  # 贷款资金
    I = 0.08  # 利息收入率

    # 读取数据
    data = np.genfromtxt('../data/附件 1：data_100.csv', delimiter=',', skip_header=1)

    # 获取 T 矩阵和 H 矩阵
    T = data[:, ::2]
    H = data[:, 1::2]

    # 定义二进制决策变量,10 * 100
    x = Array.create('x', shape=(10, 100), vartype='BINARY')

    # 定义惩罚项 M
    M = Placeholder('M')
    M = 100000

    # 定义哈密顿量,也就是对应的目标函数
    # 注意这里不是总通过率和总坏账率，是中间变量，将最终决策目标的连加符号放到里面整出来的
    P = np.sum(np.multiply(x, T))
    Q = np.sum(np.multiply(x, H))
    H = - (L * I * P * (1 - Q) - L * P * Q) + M * Constraint((np.sum(x) - 1) ** 2, label='sum(x_i_j) = 1')

    # 编译哈密顿量得到一个模型
    model = H.compile()

    # 将 Qubo 模型输出为 BinaryQuadraticModel，BQM 来求解
```

```
    bqm = model.to_bqm()

    # 记录开始退火时间
    start = time.time()

    # 模拟退火
    sa = SimulatedAnnealingSampler()
    sampleset = sa.sample(bqm, seed=666, beta_range=[10e-10, 50], num_sweeps=10000, beta_schedule_type='geometric',
                          num_reads=10)


    # 对数据进行筛选，对选取的数据进行选取最优的
    decoded_samples = model.decode_sampleset(sampleset) # 将上述采样最好的 num_reads 组数据变为模型可读的样本数据
    best_sample = min(decoded_samples, key=lambda x: x.energy) # 将能量值最低的样本统计出来，表示 BQM 的最优解

    end = time.time()
    print(f"退火时间花费：{end - start} s")

    # 统计决策变量为 1 的所有数据
    data_1_list = get_key(best_sample.sample, 1)

    print(f"对应的取 1 的决策变量有：{data_1_list}(注意是索引，从 0 开始),对应的能量为(最大最终收入):{- best_sample.energy}")


# q2 求解主函数,问题二的主要求解函数
def q2_main():
    # 定义一些变量
    L = 1000000 # 贷款资金
    I = 0.08 # 利息收入率

    # 表示选取的卡号
    card1, card2, card3 = 1, 2, 3

    # 读取数据
    data = np.genfromtxt('../data/附件 1：data_100.csv', delimiter=',', skip_header=1)

    # 获取 T 矩阵和 H 矩阵
    T = data[:, ::2]
    H = data[:, 1::2]
```

```
    # 定义二进制决策变量,10 * 100
    x = Array.create('x', shape=(10, 10, 10), vartype='BINARY')

    # 定义惩罚项 M
    M = Placeholder('M')
    M = 30000

    # 定义哈密顿量,也就是对应的目标函数
    T1 = T[:, card1 - 1]
    T2 = T[:, card2 - 1]
    T3 = T[:, card3 - 1]

    H1 = H[:, card1 - 1]
    H2 = H[:, card2 - 1]
    H3 = H[:, card3 - 1]

    # 计算三种组合后的总通过率和总坏账率,通过广播机制来实现
    T = T1[:, None, None] * T2[None, :, None] * T3[None, None, :]
    H = (H1[:, None, None] + H2[None, :, None] + H3[None, None, :]) / 3

    P = np.sum(np.multiply(x, T))
    Q = np.sum(np.multiply(x, H))
    # H = - (L * I * P * (1 - Q) - L * P * Q) + M * (np.sum(x) - 1) ** 2
    H = - (L * I * P * (1 - Q) - L * P * Q) + M * Constraint((np.sum(x) - 1) ** 2, label='sum(x_i_j)
= 1')

    # 编译哈密顿量得到一个模型
    model = H.compile()

    # 将 Qubo 模型输出为 BinaryQuadraticModel，BQM 来求解
    bqm = model.to_bqm()

    print("开始模拟退火")

    start = time.time()
    sa = SimulatedAnnealingSampler()
    sampleset = sa.sample(bqm, seed=888, beta_range=[10e-12, 60],
beta_schedule_type='geometric',
                          num_reads=50)

    # 对数据进行筛选
    decoded_samples = model.decode_sampleset(sampleset)  # 将上述采样最好的 num_reads 组
数据变为模型可读的样本数据
    best_sample = min(decoded_samples, key=lambda x: x.energy)  # 将能量值最低的样本统计
```

出来，表示 BQM 的最优解

```
    # print(f"验证约束条件 M：{best_sample.constraints()}")
    end = time.time()
    print(f"退火时间：{end - start} s")

    # print(best_sample.sample)

    # 统计决策变量为 1 的所有数据
    data_1_list = get_key(best_sample.sample, 1)

    print(f"对应的取 1 的决策变量有：{data_1_list}(表示[第一张卡的阈值][第二张卡的阈值][第三张卡的阈值]，注意是索引，从 0 开始),对应的能量为{-best_sample.energy}")
```

 # q3 求解主函数,问题三的主要求解函数

```
def q3_main():
    # 定义一些变量
    L = 1000000  # 贷款资金
    I = 0.08  # 利息收入率

    # 迭代次数
    Iter = 10

    # 选取第几列的信用卡,初始化最开始选择的数据
    card1, threshold1, card2, threshold2, card3, threshold3 = 1, 0, 2, 0, 3, 0

    # 定义最大的数
    max_value = -1
    # 读取数据
    df = pd.read_csv('../data/附件 1：data_100.csv', header=0)

    for iter in range(Iter): # 迭代次数，其实这里还可以根据几次迭代算出的总值来提前结束迭代
        # 设置更新迭代,不使用临时变量
        card1, threshold1, card2, threshold2, card3, threshold3 = card2, threshold2, card3, threshold3, card1, threshold1

        # 获取第一张卡和第二张卡的通过率和坏账率
        t1 = df[f"t_{card1}"].iloc[threshold1]
        t2 = df[f"t_{card2}"].iloc[threshold2]
        h1 = df[f"h_{card1}"].iloc[threshold1]
        h2 = df[f"h_{card2}"].iloc[threshold2]
```

```
    # 获取第三张卡能够选取的通过率与坏账率的矩阵

    choose_index = [i for i in range(1, 101) if i not in (card1, card2)]
    card3_data = df.drop([f"t_{card1}", f"h_{card1}", f"t_{card2}", f"h_{card2}"], axis=1).values

    # 获取 T 矩阵和 H 矩阵
    T = card3_data[:, ::2]
    H = card3_data[:, 1::2]

    # 定义二进制决策变量,10 * 98
    x = Array.create('x', shape=(10, 98), vartype='BINARY')
    # 定义惩罚项 M
    M = Placeholder('M')
    M = 50000

    # 定义哈密顿量,也就是对应的目标函数
    P = t1 * t2 * np.sum(np.multiply(x, T))
    Q = (h1 + h2 + np.sum(np.multiply(x, H))) / 3
    H = - (L * I * P * (1 - Q) - L * P * Q) + M * Constraint((np.sum(x) - 1) ** 2, label='sum(x_i_j) = 1')

    # 编译哈密顿量得到一个模型
    model = H.compile()
    bqm = model.to_bqm()
    # 记录开始退火时间
    start = time.time()

    #
    sa = SimulatedAnnealingSampler()

    sampleset = sa.sample(bqm, seed=666, beta_range=[10e-10, 50], num_sweeps=999, beta_schedule_type='geometric',
                          num_reads=50)

    # 对数据进行筛选，对选取的数据进行选取最优的
    decoded_samples = model.decode_sampleset(sampleset)  # 将上述采样最好的 num_reads 组数据变为模型可读的样本数据
    best_sample = min(decoded_samples, key=lambda x: x.energy)  # 将能量值最低的样本统计出来，表示 BQM 的最优解

    end = time.time()
    # print(f"退火时间花费：{end - start} s")
```

```
    # 统计决策变量为 1 的所有数据,并对第一个数据进行拆解，获得对应的下标
    data_1_list = get_key(best_sample.sample, 1)
    index_i_j = re.findall(r'\d+', data_1_list[0])
    index_i = int(index_i_j[0])

    if max_value < - best_sample.energy:
        card3 = choose_index[int(index_i_j[1])]
        threshold3 = index_i
        max_value = - best_sample.energy

    print(
        f" 第 {iter + 1} 次 迭 代 ， 最 大 值 ： {max_value}, 卡 选 择 :{card1}_{threshold1 + 1},{card2}_{threshold2 + 1},{card3}_{threshold3 + 1}")

    print(f" 最 大 值 ： {max_value}, 卡 选 择 :{card1}_{threshold1 + 1},{card2}_{threshold2 + 1},{card3}_{threshold3 + 1}")
```