



计算金融与仿真

课程论文

论文题目：计算金融与仿真

学生姓名：李晶晶 张璐 朱冯婧

指导老师：邓智斌

1. 投资组合选择介绍

P2P 借贷 (peer-to-peer lending) 是一种个人之间直接借贷的方式, 通过在线平台进行, 无需传统银行中介. 这种创新的融资方式提供了一个市场, 借款人可以向多个贷方申请贷款, 贷方可以选择为这些贷款提供全部或部分资金. 这一模式利用技术手段简化了借贷过程, 通常为借款人带来更低的利率, 同时为投资者提供更高的回报.

目前, 全球几大平台主导着 P2P 借贷领域, 主要包括 LendingClub, Prosper 和 Funding Circle 等. 这类平台通过评估贷款项目和借款人的 FICO 分数, 贷款金额和期限, 借款人的资产, 债务状况, 就业类型等数据, 为每笔贷款提供风险评级. 其中, LendingClub 是美国最大的 P2P 借贷平台之一, 提供个人贷款和投资机会. 本文使用来自 LendingClub 的公共数据集 (涵盖 2007–2018 年的贷款记录), 将贷款属性转化为违约概率作为后续建模依据.

为了实现贷款投资组合的多元化, 投资者可依据自身的风险偏好从不同风险等级的贷款中进行选择. 与传统金融机构贷款不同, P2P 贷款中的每位个人投资者其可用于投资的资金较为有限. 因此, 如何在资金约束下, 基于借款人的风险特征评估潜在回报与风险, 选择合适的贷款项目, 并进行最优资金配置, 是 P2P 投资者亟需解决的问题.

大量学者针对 P2P 网络贷款的投资组合优化问题开展了深入研究. Wan 等人将贷款投资组合决策转化为一个在特定时点下实现收益最大化与风险最小化的优化问题, 并引入混合治愈模型 (mixture cure model, MCM) 以提升投资效果, 构建实例驱动模型对投资者的组合决策进行优化 [?]. Guo 等人则提出一种基于实例的 P2P 投资组合决策模型, 从风险最小角度出发, 利用 LendingClub 和 Prosper 数据集实现投资组合配置优化 [?]. Ajay 等人则将该问题转化为多目标优化模型, 在计算贷款相似度时将期望值框架与传统核方法结合, 优化结果优于既有模型 [?].

贷款投资组合优化 (Loan Portfolio Optimization) 与传统投资组合优化具有相似性, 但更关注与贷款相关的特定风险, 例如借款人信用评分, 贷款金额, 借款用途等因素引发的借款人风险, 违约风险以及利率波动风险. 优化目标是通过合理配置不同借款人贷款项目, 在控制风险的同时实现收益最大化.

在风险量化方面, VaR (Value at Risk) 与 CVaR (Conditional Value at Risk) 被广泛用于衡量投资组合的风险水平. 其中, VaR 表示在置信水平 α 下, 金融资产或组合在未来特定持有期内的最大可能损失; 而 CVaR 则刻画了超过 VaR 截止点的极端损失的期望, 即对尾部风险的加权平均, 是更稳健的风险衡量指标. 在贷款组合优化中, CVaR 能够帮助投资者更有效地控制整体尾部损失风险, 从而实现收益与风险的权衡优化.

2. 模型建立

决策变量

$$x_i \in \{0, 1\}, \quad i = 1, 2, \dots, N$$

其中,

- $x_i = 1$ 表示选择资助第 i 个贷款对象；
- $x_i = 0$ 表示不选择。

参数说明

- A_i : 第 i 个贷款的金额；
- r_i : 第 i 个贷款的利率；
- P_i : 第 i 个贷款的违约概率；
- B : 总投资预算；
- R_{\max} : 允许的最大违约风险；
- G_k : 第 k 个信用等级的贷款集合；
- α_k : 第 k 个信用等级的最大投资比例；
- m : 最多选择的贷款数量 (Top- m) 。

模型形式

$$\begin{aligned}
 \max_{x_i \in \{0,1\}} \quad & \sum_{i=1}^N x_i A_i r_i (1 - P_i) \\
 \text{s.t.} \quad & \sum_{i=1}^N x_i A_i \leq B, \\
 & \sum_{i \in G_k} x_i A_i \leq \alpha_k B, \quad \forall k, \\
 & \sum_{i=1}^N x_i A_i P_i \leq R_{\max} \quad (\text{风险控制}) \\
 & \sum_{i=1}^N x_i \leq m,
 \end{aligned} \tag{P2P}$$

$$\begin{aligned}
 \max_{x_i \in \{0,1\}} \quad & \sum_{i=1}^N x_i A_i r_i (1 - P_i) \\
 \text{s.t.} \quad & \sum_{i=1}^N x_i A_i \leq B, \\
 & \sum_{i \in G_k} x_i A_i \leq \alpha_k B, \quad \forall k, \\
 & \sum_{i=1}^N x_i A_i \cdot \tilde{P}_i^{(s)} - \eta \leq \mathcal{M} z_s, \quad \forall s = 1, \dots, S, \\
 & \sum_{s=1}^S z_s \leq (1 - \beta) S \\
 & \sum_{i=1}^N x_i \leq m, \\
 & z_s \in \{0,1\}, \quad \eta \in \mathbb{R}
 \end{aligned} \tag{P2P-VaR}$$

$$\begin{aligned}
 & \max_{x_i \in \{0,1\}} \sum_{i=1}^N x_i A_i r_i (1 - P_i) \\
 & \text{s.t.} \quad \sum_{i=1}^N x_i A_i \leq B, \\
 & \quad \sum_{i \in G_k} x_i A_i \leq \alpha_k B, \quad \forall k, \\
 & \quad \xi_s \geq \sum_{i=1}^N x_i A_i \cdot \tilde{P}_i^{(s)} - \eta, \quad \forall s = 1, \dots, S, \\
 & \quad \eta + \frac{1}{S(1-\beta)} \sum_{s=1}^S \xi_s \leq \text{CVaR}_{\max}, \\
 & \quad \sum_{i=1}^N x_i \leq m, \\
 & \quad \xi_s \geq 0, \quad \eta \in \mathbb{R}
 \end{aligned} \tag{P2P-CVaR}$$

3. 算法设计

Algorithm 1: 启发式算法：贷款组合优化（含 CVaR 控制）

Input: 贷款数据 $\{A_i, r_i, P_i\}_{i=1}^N$; 预算 B ; Top- m 限制; CVaR 上限 CVaR_{\max} ; 置信水平 β ; 模拟场景矩阵 $\tilde{P}_i^{(s)} \in \{0, 1\}^{S \times N}$

Output: 最优选择向量 $x^* \in \{0, 1\}^N$

```

1  $x_{\text{best}} \leftarrow 0$ ; // 初始为空解
2 根据评分  $\text{score}_i = A_i[r_i(1 - P_i) - P_i]$  降序排列贷款
3  $\text{total\_budget} \leftarrow 0, \text{total\_selected} \leftarrow 0, x_{\text{current}} \leftarrow 0$ 
4 foreach  $i$  in 排序后的贷款列表 do
5   if  $\text{total\_budget} + A_i > B$  或  $\text{total\_selected} + 1 > m$  then
6     continue;
7    $x_{\text{current}}[i] \leftarrow 1$ ;
8    $\text{total\_budget} \leftarrow \text{total\_budget} + A_i$ ;
9    $\text{total\_selected} \leftarrow \text{total\_selected} + 1$ ;
10 if  $\text{Feasible}(x_{\text{current}})$  then
11    $x_{\text{best}} \leftarrow x_{\text{current}}$ ;
12 Function  $\text{Feasible}(x)$ :
13   for  $s \leftarrow 1$  to  $S$  do
14      $L_s \leftarrow \sum_{i=1}^N x_i A_i \cdot \tilde{P}_i^{(s)}$ ;
15    $\eta \leftarrow \beta$  分位点的  $\{L_s\}$ ;
16   for  $s \leftarrow 1$  to  $S$  do
17      $\xi_s \leftarrow \max(L_s - \eta, 0)$ ;
18    $\text{CVaR}_\beta(x) \leftarrow \eta + \frac{1}{S(1-\beta)} \sum_{s=1}^S \xi_s$ ;
19   return 是否满足  $\text{CVaR}_\beta(x) \leq \text{CVaR}_{\max}$  且满足其他约束;
20 return  $x_{\text{best}}$ ;

```

4. 案例研究

4.1. 数据集描述

本研究使用的数据集来自 Lending Club 平台, 原始数据由 Kaggle 网站<https://www.kaggle.com/datasets/wordsforthewise/lending-club>公开提供. Lending Club 是美国最大的网络借贷平台之一, 提供了详尽的个人借款申请及其还款情况的数据, 广泛应用于学术界和工业界进行信贷风险评估, 违约预测及投资组合优化等研究.

该数据集包含了从 2007 年至 2018 年的借款记录, 共计数百万条样本. 每条记录对应一笔贷款申请, 涵盖了包括贷款金额, 利率, 借款人信用等级, 债务收入比, 贷款期限, 还款状态, 就业年限, 收入, 地址状态, 房屋所有权, FICO 评分区间等在内的多维度信息.

在本研究中, 我们主要筛选并保留以下变量用于建模分析:

- **loan_amnt**: 借款人申请的贷款金额, 作为 A_i ;
- **int_rate**: 借款合同中约定的年利率, 用于计算收益率 r_i ;
- **grade**: 借款人的信用等级 (A 至 G), 用于分组限制;
- **loan_status**: 实际贷款的还款状态 (如 Fully Paid, Charged Off), 用于推断违约情况;
- **annual_inc**: 借款人年收入;
- **dti**: 债务收入比, 用于辅助风险刻画;
- **term**: 贷款期限 (如 36 months 或 60 months);
- **emp_length**: 借款人工作年限;
- **addr_state**: 借款人所在州;
- **fico_range_high, fico_range_low**: 借款人 FICO 信用评分区间;

为了满足模型中对违约概率 P_i 的需求, 我们将 “loan_status” 字段中状态为 “Charged Off” 的贷款视为违约样本, 其余如 “Fully Paid”, “Current” 等状态作为非违约样本, 并基于历史频率法估算每一类贷款的违约概率。

此外, 为模拟贷款违约的风险场景, 我们以借款人的信用等级, FICO 评分和历史违约频率为依据, 构建了 S 个 Monte Carlo 风险场景, 用于后续 CVaR 优化模型的风险评估。

通过上述数据处理步骤, 最终形成了一个结构规范, 信息完备, 适用于组合优化问题的数据集, 为后续实证分析与建模提供了坚实的数据基础。

4.2. 使用机器学习预测违约概率 P_i

尽管本研究使用的数据集中所有贷款均已获得实际资助, 但在资金有限, 需进行优选配置的情境下, 我们仍需要对这些已发放贷款的还款风险进行再评估。为此, 我们引入机器学习方法, 对每笔贷款的未来违约概率 P_i 进行预测建模, 以作为后续优化模型中的风险输入参数。

建模目标 预测函数的目标是: 对于每笔已发放贷款 i , 根据其已知特征向量 \mathbf{x}_i , 估计其在未来发生违约的概率 $P_i = \mathbb{P}(y_i = 1 | \mathbf{x}_i)$ 。其中, $y_i = 1$ 表示贷款最终发生违约 (如状态为 Charged Off), $y_i = 0$ 表示贷款最终还清 (如状态为 Fully Paid)。

特征构造 我们基于借款人基本属性, 贷款合同信息以及信用评级等信息构建预测特征集, 涵盖但不限于以下变量:

- **loan_amnt**: 贷款金额;
- **term**: 贷款期限;
- **int_rate**: 贷款利率;
- **grade** 和 **sub_grade**: Lending Club 信用评级;
- **emp_length**: 工作年限;
- **home_ownership**: 住房类型;
- **annual_inc**: 年收入;
- **dti**: 债务收入比;

- **purpose:** 贷款用途;
- **fico_range_high / low:** 信用评分;

建模方法 考虑到目标变量仍是二元状态 (违约 / 未违约), 我们采用监督学习的二分类方法进行建模. 尝试的模型包括逻辑回归 (Logistic Regression), 随机森林 (Random Forest), 梯度提升树 (GBDT), 极端梯度提升 (XGBoost) 等.

由于样本中违约样本占比相对较小, 我们在训练过程中采用类别加权, 欠采样等方式处理类别不平衡问题.

训练与评估 我们将全部已发放贷款随机划分为训练集 (70%) 与测试集 (30%), 使用交叉验证调优参数, 并基于测试集评估模型表现. 评价指标包括准确率 (Accuracy), AUC 值 (Area Under the ROC Curve), F1 分数等.

最终, 我们选择 AUC 表现最优的模型用于对所有贷款样本生成违约概率预测值 \hat{P}_i , 作为后续优化模型中的输入.

说明 虽然原始数据中每笔贷款都已实际放款, 但我们的建模任务是为现实中的“再选择”提供依据. 即在预算受限, 资源不足时, 如何在这些真实已放款的贷款中优先挑选违约概率低, 预期收益高的子集, 构建一个更稳健的投资组合. 因此, \hat{P}_i 的预测并非用于决定放款与否, 而是作为组合优化的“风险估计量”, 用于构建期望收益与 CVaR 等风险指标.

4.3. 使用算法求解原始模型, VaR 模型与 CVaR 模型

在本节中, 我们分别构建并求解三类贷款筛选优化模型: 原始模型, VaR (Value at Risk) 模型和 CVaR (Conditional Value at Risk) 模型. 三者均以最大化投资收益为目标, 并在此基础上逐步引入风险控制手段, 以模拟实际投资场景中对风险的不同管控需求.

原始模型 (期望损失约束) 。

为求解最大化期望回款的贷款投资组合问题, 本文设计了如算法 ?? 所示的混合整数规划模型. 在贷款数据集中, 每条记录包含金额 A_i 、年利率 r_i 、还款概率 P_i 以及信用等级信息. 目标是在预算、风险和信用等级约束下, 从中筛选出若干笔贷款以最大化组合的期望净收益.

模型特点如下: 目标函数最大化 $\sum x_i A_i r_i (1 - P_i)$, 即各笔贷款在预期条件下的净收益;

约束条件包括总预算限制、信用等级分布比例控制、风险容忍上限 (以期望损失衡量), 以及最多投资 m 笔贷款的限制;

决策变量 $x_i \in 0, 1$ 表示是否选中贷款 i .

在设置参数方面, 本文采用如下配置:

求解结果概述: Gurobi Optimizer 在约 1.3 秒内成功收敛到全局最优解. 最终选中的贷款子集包含 3293 笔贷款, 总投资金额为 \$58,676,750, 未触及预算上限, 表明优化模型具备有效控制风险和冗余空间的能力. 组合的期望净收益为 \$3,538,325.75.

按信用等级统计结果:

Algorithm 2: 最大化期望回款的投资组合优化

Input: 贷款数据集：每条数据包含 A_i （金额）、 r_i （利率）、 P_i （还款概率）、信用等级

Output: 最优贷款子集 $x_i \in \{0, 1\}$ 及其组合统计

1 数据预处理：计算 $A_i = \text{loan_amnt}_i$, $r_i = \text{int_rate}_i/100$, $P_i = \text{default_prob}_i$;

2 清洗缺失值与无效值；按等级分组为 G_k ;

3 模型构建：;

4 定义决策变量 $x_i \in \{0, 1\}$ ，表示是否选择贷款 i ;

5 设定目标函数：最大化 $\sum x_i A_i r_i (1 - P_i)$;

6 添加约束：

- $\sum x_i A_i \leq B$ // 预算约束
- $\sum_{i \in G_k} x_i A_i \leq \alpha_k B$ // 信用等级比例约束
- $\sum x_i A_i P_i \leq R_{\max}$ // 风险约束
- $\sum x_i \leq m$ // Top-m 限制

调用 Gurobi 优化器求解模型;

if 模型可解 **then**

输出最优贷款组合 $x_i = 1$ 的子集;
统计并保存结果至 CSV 文件;

else

输出模型求解失败与错误类型（不可行/无界）;

Table 1 优化模型参数设置

参数	数值说明
总预算 B	\$100,000,000
最大风险容忍度 R_{\max}	\$10,000,000
最大投资笔数 m	5000
信用等级比例上限 α_k	A: 40%, B: 30%, C: 20%, D: 10%, E: 5%, F: 2%, G: 1%

Table 2 最优组合中各等级贷款统计

信用等级	贷款数量（笔）	金额合计（美元）	平均违约概率
A	2180	40,000,000	12.65%
B	1113	18,676,750	26.52%

结果显示，等级为 A 的贷款在数量和金额上占据主导地位，表明模型倾向于优先选择高评级、低风险贷款；等级 B 的贷款次之，作为补充以提高收益。在严格的风险与比例控制下，模型实现了回报与稳健的双重目标，具有良好的现实应用价值。

Algorithm 3: 基于 VaR 约束的贷款组合优化 (PSO + Gurobi)

Input: 贷款数据 (A_i, r_i, P_i) ，预算 B ，个数上限 m ，VaR 置信水平 β

Output: 最大期望收益的贷款子集 $x_i \in \{0, 1\}$

- 1 **1. 数据处理与场景生成:**
 - 2 计算 $profit_i = A_i \cdot r_i \cdot (1 - P_i)$;
 - 3 生成 S 个场景损失 $L_{i,s} \sim \text{Bernoulli}(P_i) \cdot A_i$;
 - 4 **2. 粒子群初始化:**
 - 5 启发式生成初始粒子，设置速度与参数;
 - 6 **3. PSO 迭代:**
 - 7 **for** $t = 1$ **to** T **do**
 - 8 计算每个粒子的 VaR 值 η ;
 - 9 若 η 超过上限或约束不满足，则适应度为 $-\infty$;
 - 10 否则计算期望收益作为适应度;
 - 11 更新个体/全局最优，调整速度与位置;
 - 12 **4. Gurobi 精解 (Warm-start):**
 - 13 构建模型:
 - 14 最大化 $\sum x_i profit_i$ ，约束包括:
 - 15 预算 $\sum x_i A_i \leq B$; 个数 $\sum x_i \leq m$;
 - 16 VaR 置信限制 $\sum_s z_s \leq (1 - \beta)S$;
 - 17 损失 $> \eta$ 的标记变量 z_s ;
 - 18 使用 PSO 解 warm-start，求解最优组合;
-

为验证模型有效性，本文基于 VaR (Value at Risk) 约束构建贷款投资组合优化模型，结合粒子群算法 (PSO) 与混合整数规划 (Gurobi) 进行双阶段求解。结果如表 ?? 所示，PSO 能在较短时间内提供满足约束的可行解，其投资组合期望收益为约 701.66 万元，对应 VaR 上界为 3669.81 万元，满足置信水平为 95% 的风险容忍范围。进一步使用 Gurobi 以 PSO 解为 warm-start 进行精确优化后，期望收益提升至 838.91 万元，VaR 上界略升至 4755.68 万元。两种方法结果一致性良好，验证了所构建模型在兼顾收益与风险控制方面的有效性与实用性。

在本模型中，我们采用蒙特卡洛模拟构造投资损失的随机场景，基于条件风险价值 (CVaR) 准则对投资组合进行优化，目标为在限定风险水平下最大化期望收益。首先利用粒子群优化算法 (PSO) 快速获取近似可行解，随后通过 Gurobi 求解器进行精确优化。参数设置如表 ?? 所示。

Table 3 VaR 模型主要参数设置

符号	参数含义	数值/设置
B	投资总预算	1×10^8 元
m	最多选择的贷款数量	5000
S	蒙特卡洛场景数	1000
β	VaR 置信水平	0.95
η	VaR 上界	动态计算
pop_size	粒子种群规模	30
max_iter	PSO 最大迭代次数	100
w, c_1, c_2	PSO 参数（惯性 + 学习因子）	$w = 0.7, c_1 = 1.5, c_2 = 1.5$

Table 4 基于 VaR 约束的优化结果对比

方法	选中贷款数	投资组合期望收益（元）	VaR 上界 η （元）
PSO 粒子群算法	5000	7,016,573.73	36,698,075.00
Gurobi 精确求解	5000	8,389,092.16	47,556,800.00

Algorithm 4: 基于 CVaR 约束的贷款组合优化（PSO + Gurobi）

Input: 贷款数据 (A_i, r_i, P_i) , 预算 B , 个数上限 m , 置信水平 β , CVaR 上限 R_{\max}

Output: 最大期望收益的贷款子集 $x_i \in \{0, 1\}$

- 1 **1. 数据处理与场景生成:**
- 2 计算 $profit_i = A_i \cdot r_i \cdot (1 - P_i)$;
- 3 生成 S 个损失场景 $L_{i,s}$;
- 4 **2. 粒子群初始化与适应度:**
- 5 启发式初始化粒子 x ;
- 6 适应度为 $\sum x_i profit_i$ 加上风险利用率奖励;
- 7 其中 $CVaR = \eta + \frac{1}{S(1-\beta)} \sum_s \max(l_s - \eta, 0)$;
- 8 **3. PSO 优化迭代:**
- 9 更新粒子位置与速度, 保存最优组合;
- 10 **4. Gurobi 精解:**
- 11 构建模型:
- 12 最大化 $\sum x_i profit_i + \lambda \cdot \eta / R_{\max}$;
- 13 约束包括: 预算、个数、CVaR 构造与 CVaR 上限限制;
- 14 使用 PSO 解 warm-start, 求解精确解;

Table 5 模型参数设置

参数	取值说明
B	总预算限制, 1×10^8 元
m	投资项目上限, 5000 个
β	CVaR 置信水平, 0.95
R_{\max}	风险容忍值上限, 1.5×10^7 元
S	蒙特卡洛场景数量, 1000
pop_size	粒子群规模, 30
max_iter	粒子群最大迭代次数, 100
(w, c_1, c_2)	PSO 参数: $w = 0.7, c_1 = c_2 = 1.5$

Table 6 PSO 与 Gurobi 优化结果比较

指标	PSO 解	Gurobi 解
期望回款 (元)	€7,016,573.73	€5,753,233.48
CVaR 风险值 (元)	€39,172,172.50	€14,997,698.50
是否满足 CVaR 约束	否	是
选中项目数量	若干 (≤ 5000)	若干 (≤ 5000)

结果表明 (见表 ??), PSO 能在较短时间内获得收益较高的组合, 但其解未能严格满足 CVaR 风险约束; 而 Gurobi 求解得到的解在收益上略低, 但能严格控制组合风险, 满足预设风险容忍上限。两者结合可有效提升模型的求解效率与解的可行性, 为后续实际投资提供策略参考

4.4. 结果分析

5. 结论

小组分工

姓名	学号	分工
李晶晶	202428016443014	论文撰写, 模型设计
张璐	202428016446002	数据处理, 模型求解
朱冯婧	202428016443008	论文撰写, 算法设计

参考文献

- [1] 万洁 and 郝俊. 基于混合治愈模型的网络借贷投资组合决策模型与实证研究. *系统科学与数学*, 43(5):1138–1156, 2023.
- [2] Yiyang Guo, Wei Zhou, Chen Luo, and et al. Instance-based credit risk assessment for investment decisions in p2p lending. *European Journal of Operational Research*, 249(2):417–426, 2016.
- [3] A Byanjankar, J Mezei, and M Heikkila. Data-driven optimization of peer-to-peer lending portfolios based on the expected value framework. *Intelligent Systems in Accounting, Finance and Management*, 28(2):119–129, 2021.

6. 附录

6.1. VaR 模型求解结果

```

1  PS E:\github\homeworkofcalculatefinance> python -u "e:\github\homeworkofcalculatefinance\5-25-模型求解\pso_gurobi_VaR.py"
2  正在运行 PSO 寻优...
3
4  --- 粒子群优化 (PSO) 结果 ---
5  选中借款人数 = 5000
6  PSO 投资组合期望收益 = 7016573.73
7  PSO 使用的 VaR 上界 = 36698075.00
8  Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 11+.0 (26100.2))
9
10 CPU model: Intel(R) Core(TM) Ultra 9 185H, instruction set [SSE2|AVX|AVX2]
11 Thread count: 16 physical cores, 22 logical processors, using up to 22 threads
12
13 Optimize a model with 1003 rows, 30370 columns and 10302786 nonzeros
14 Model fingerprint: 0xa4fbee09
15 Variable types: 1 continuous, 30369 integer (30369 binary)
16 Coefficient statistics:
17   Matrix range      [1e+00, 2e+08]
18   Objective range   [1e+00, 3e+03]
19   Bounds range      [1e+00, 1e+00]
20   RHS range         [5e+01, 1e+08]
21 Warning: Model contains large matrix coefficients
22   Consider reformulating model or setting NumericFocus parameter
23   to avoid numerical issues.
24
25 User MIP start produced solution with objective 7.01657e+06 (2.25s)
26 Loaded user MIP start with objective 7.01657e+06
27 Processed MIP start in 2.23 seconds (0.95 work units)
28
29 Presolve removed 1001 rows and 1001 columns
30 Presolve time: 1.88s
31 Presolved: 2 rows, 29369 columns, 58738 nonzeros
32 Variable types: 0 continuous, 29369 integer (29369 binary)
33 Found heuristic solution: objective 7283784.2164
34
35 Starting sifting (using dual simplex for sub-problems)...
36
37   Iter    Pivots    Primal Obj    Dual Obj    Time
38     0         0    infinity    -8.5954234e+06    4s
39     1         1  -1.0939240e+05  -8.5461998e+06    4s
40     2         3  -1.3438183e+05  -8.5297172e+06    4s
41     3         5  -1.8822504e+05  -8.5227263e+06    4s
42     4         7  -2.1502998e+05  -8.5162448e+06    4s
43     5         9  -2.6694422e+05  -8.5105148e+06    4s
44     6        11  -3.1011424e+05  -8.5063219e+06    4s
45     7        13  -3.4598473e+05  -8.5027436e+06    4s
46     8        15  -3.7983946e+05  -8.4994864e+06    4s
47     9        17  -4.1301854e+05  -8.4971532e+06    4s

```

```

48      10      19 -4.4834645e+05 -8.4944963e+06 4s
49      11      21 -4.8769815e+05 -8.4924081e+06 4s
50      12      23 -5.2353089e+05 -8.4903439e+06 4s
51      13      25 -5.6559651e+05 -8.4875771e+06 4s
52      14      27 -6.2374219e+05 -8.4851169e+06 4s
53      15      29 -6.9229155e+05 -8.4833902e+06 4s
54      16      31 -7.4067974e+05 -8.4817202e+06 4s
55      17      33 -7.9618193e+05 -8.4802313e+06 4s
56      18      35 -8.3944338e+05 -8.4781641e+06 4s
57      19      37 -9.0488706e+05 -8.4757665e+06 4s
58      20      39 -9.6948627e+05 -8.4735416e+06 4s
59      21      41 -1.0366669e+06 -8.4713711e+06 4s
60      22      43 -1.1075983e+06 -8.4689240e+06 4s
61      23      45 -1.1876166e+06 -8.4668946e+06 4s
62      24      47 -1.2650869e+06 -8.4644744e+06 4s
63      25      49 -1.3537151e+06 -8.4623352e+06 4s
64      26      51 -1.4380471e+06 -8.4602487e+06 4s
65      27      53 -1.5182690e+06 -8.4583143e+06 4s
66      28      55 -1.6024285e+06 -8.4563366e+06 4s
67      29      57 -1.6864300e+06 -8.4543667e+06 4s
68      30      59 -1.7730984e+06 -8.4524107e+06 4s
69      31      61 -1.8757160e+06 -8.4505185e+06 4s
70      32      63 -1.9832589e+06 -8.4489934e+06 4s
71      33      65 -2.0609758e+06 -8.4470407e+06 4s
72      34      67 -2.1849239e+06 -8.4453954e+06 4s
73      35      69 -2.2734543e+06 -8.4435278e+06 4s
74      36      71 -2.3904212e+06 -8.4417966e+06 4s
75      37      73 -2.5010673e+06 -8.4402299e+06 4s
76      38      75 -2.6152965e+06 -8.4387511e+06 4s
77      39      77 -2.7081939e+06 -8.4372583e+06 4s
78      40      79 -2.8321013e+06 -8.4356705e+06 4s
79      41      81 -2.9482273e+06 -8.4340800e+06 4s
80      42      83 -3.0612090e+06 -8.4325333e+06 4s
81      43      85 -3.1986347e+06 -8.4310381e+06 4s
82      44      87 -3.3213768e+06 -8.4295375e+06 4s
83      45      89 -3.4644936e+06 -8.4281239e+06 4s
84      46      91 -3.6011453e+06 -8.4268796e+06 4s
85      47      93 -3.7285862e+06 -8.4254399e+06 4s
86      48      95 -3.8623207e+06 -8.4239676e+06 4s
87      49      97 -3.9861846e+06 -8.4226384e+06 4s
88
89 Sifting complete
90
91
92 Root relaxation: objective 8.389622e+06, 106 iterations, 0.22 seconds (0.14 work units)
93
94      Nodes      |      Current Node      |      Objective Bounds      |      Work
95      Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
96
97      48      95 -3.8623207e+06 -8.4239676e+06 4s
98      49      97 -3.9861846e+06 -8.4226384e+06 4s
99
100 Sifting complete
101
102
103 Root relaxation: objective 8.389622e+06, 106 iterations, 0.22 seconds (0.14 work units)
104
105      Nodes      |      Current Node      |      Objective Bounds      |      Work
106      Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
107
108      49      97 -3.9861846e+06 -8.4226384e+06 4s
109
110 Sifting complete
111
112
113 Root relaxation: objective 8.389622e+06, 106 iterations, 0.22 seconds (0.14 work units)
114
115      Nodes      |      Current Node      |      Objective Bounds      |      Work
116      Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
117
118

```

```

119 Sifting complete
120
121
122 Root relaxation: objective 8.389622e+06, 106 iterations, 0.22 seconds (0.14 work units)
123
124 Nodes | Current Node | Objective Bounds | Work
125 Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
126
127 Sifting complete
128
129
130 Root relaxation: objective 8.389622e+06, 106 iterations, 0.22 seconds (0.14 work units)
131
132 Nodes | Current Node | Objective Bounds | Work
133 Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
134
135
136 Root relaxation: objective 8.389622e+06, 106 iterations, 0.22 seconds (0.14 work units)
137
138 Nodes | Current Node | Objective Bounds | Work
139 Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
140
141
142 Nodes | Current Node | Objective Bounds | Work
143 Expl Unexpl | Obj Depth IntInf | Incumbent BestBd Gap | It/Node Time
144
145
146 0 0 8389621.93 0 2 7283784.22 8389621.93 15.2% - 4s
147 H 0 0 8389092.1600 8389621.93 0.01% - 4s
148
149 0 0 8389621.93 0 2 7283784.22 8389621.93 15.2% - 4s
150 H 0 0 8389092.1600 8389621.93 0.01% - 4s
151
152 Explored 1 nodes (106 simplex iterations) in 4.55 seconds (2.12 work units)
153 Thread count was 22 (of 22 available processors)
154
155 Solution count 3: 8.38909e+06 7.28378e+06 7.01657e+06
156
157 Explored 1 nodes (106 simplex iterations) in 4.55 seconds (2.12 work units)
158 Thread count was 22 (of 22 available processors)
159
160 Solution count 3: 8.38909e+06 7.28378e+06 7.01657e+06
161 Explored 1 nodes (106 simplex iterations) in 4.55 seconds (2.12 work units)
162 Thread count was 22 (of 22 available processors)
163
164 Solution count 3: 8.38909e+06 7.28378e+06 7.01657e+06
165
166 Optimal solution found (tolerance 1.00e-04)
167
168 Solution count 3: 8.38909e+06 7.28378e+06 7.01657e+06
169
170 Optimal solution found (tolerance 1.00e-04)
171
172 Optimal solution found (tolerance 1.00e-04)
173 Best objective 8.389092160022e+06, best bound 8.389621934455e+06, gap 0.0063%
174
175 --- Gurobi 精确求解结果 ---
176 Gurobi 选中借款人数量 = 5000
177 Gurobi 投资组合期望收益 = 8389092.16
178 Gurobi 计算的 VaR 上界 = 47556800.00

```