基于双目标规划模型的中小微型企业信贷决策分析

摘要

银行与中小微企业建立信用关系时，其信贷策略分析十分重要。然而，由于中小微企业具有体量小、抵押资产少的特点，银行往往需要通过对其企业实力、信誉等级等因素的多方面评估，并结合自身目标，确定信贷策略。本文通过对反映企业信誉、交易票据等相关数据进行分析处理，建立指标和模型，并提出解决算法，从而给出决策方案的讨论分析。

针对问题一决策方案的讨论，本文分两步进行分析。首先通过基于熵权处理评价矩阵的 TOPSIS-GC 综合分析评判模型，本文对 123 家企业进行客观综合信贷情况评判、排序和分类（三类）。其次考虑到银行的长短期目标取舍关系并未被明确，本文建立了以不同企业“信贷额度占比”、“年利率”为求解变量的“短期利益”和“潜在损失量”的双目标规划模型。然后本文通过蒙特卡洛思想对数据进行离散化处理和穷举遍历，求出 Pareto front 解集，共有 122 组。进一步对决策细化，两目标取相等权值时求得最优解为：综合排名划分的三类企业的信贷额度占比分别为 0.4，0.3，0.3；A,B,C 类企业的年利率分别设置为 0.04，0.0425，0.1500。针对双目标规划，本文进一步取不同权值对银行决策进行敏感性分析，并给出不同目标取向下的最优决策 (方案一至五)。

针对问题二信誉评级未知企业的信贷决策规划，本文首先通过建立随机森林分类模型对 123 家企业的信誉评级和企业发票信息进行数据提取和学习（准确率为 87.80%），再由此预测 302 家企业的信誉评级。结果显示 A,B,C,D 类企业分别有 59，140，39，64 家。其次对模型目标函数进行调整后，本文采用改进的 NSGA-II 算法高效地求出了精度更高的解集，并依次进行敏感性分析和不同权值的方案（六至十一）讨论。

针对问题三中提到的突发情况，本文首先建立了新冠疫情背景下的企业违约概率指标，并通过对短期收益目标函数添加硬违约概率惩罚项，对潜在损失目标函数添加软违约概率惩罚项，以更新规划模型中的目标函数。采用同问题二中的改进 NSGA-II 算法，本文对调整的规划模型进行了求解并给出决策方案（十二至十七）的讨论。

在模型优化方面，本文针对企业信誉评级分类预测方法的多样性问题，采用了不同的深度学习方法进行分类学习，分别为：随机森林分类、装袋分类、极端随机树分类，并对精确度和结果相似度进行了对比分析。

关键字： TOPSIS-GC　双目标规划　随机森林　蒙特卡洛思想　NSGA-II

一、 问题重述

1.1 问题背景

银行对中小微企业提供信用贷款时，往往由于后者在规模上的劣势和抵押资产的缺乏，需要对其进行信贷综合评估，从而确定一定期限内总信贷额度在中小微企业中的分配策略。而企业的发票信息记录、流水、实力规模、行业类型等因素均可以被认为是银行与该企业进行借贷关系建立的评估影响因素。因此，对这些因素的综合分析在银行信贷策略的短期、长期决定中十分重要。

1.2 问题相关的信息

附件 1，2，3: 123 家企业的信誉评级、发票信息；302 家企业的发票信息；年利率和各类企业流失率的关系数据。

附件 4: 表一、表二分别存储 123 家企业、302 家企业在本文中提到的各项指标计算值；表三存储了问题一的 Pareto Front 的自变量集及其目标函数值。

1.3 待解决问题

问题一: 对 123 家有信誉评级的企业进行信贷综合评估量化分析，并给出银行在固定信贷总额时的分配策略。

问题二: 对 302 家无信誉评级的企业进行信贷综合评估量化分析，并给出银行在 1 亿元年度信贷总额时的分配策略。

问题三: 考虑某种突发因素对不同类型、不同行业企业的影响，从而对问题二中策略进行调整。

二、 问题分析

2.1 问题一的分析

问题一将分两步骤进行——建立评价模型对企业进行综合信贷评价；通过信誉等级等因素对当年银行信贷策略进行制定。

针对企业综合信贷评价模型，本文首先建立分别表示企业商业信誉、供求关系稳定性和企业实力的指标。考虑到指标均为客观数据，不适合主观评价方法；且单一客观方法的数据代表性不强，因此本文建立基于熵权处理评价矩阵的 TOPSIS-GC 客观综合评判方法对企业的综合信贷情况进行评估，并根据结果将企业分为三个等级。

针对信贷策略制定，由于银行并未明确当年利益和长期利益的取舍关系，因此本文考虑建立双目标——短期利益和潜在损失规划模型。其次基于每家企业所属综合信贷评估等级和信誉评级，本文对变量信贷额度比例、年利率进行范围条件约束。双目标的 Pareto front 解集（不止一组解）则由蒙特卡洛思想离散化变量、穷举求得。进一步确定策略时，本文先假设银行对两目标的重视程度相同，取相等权重求出最优解；再通过敏感性分析，对不同目标取向的最优方案进行讨论。

2.2 问题二的分析

问题二与问题一的区别主要在于企业信誉等级信息的缺失。因此本文首先建立随机森林分类模型对 123 家企业的信誉评级和企业发票信息进行数据提取和学习，再由此预测 302 家企业的信誉评级。

接下来本文将修正问题一中双目标规划模型的参数。在考虑到蒙特卡洛离散可能存在的数据失真化问题的基础上，本文提出了基于本题特征的元启发式算法——改进 NSGA-II 算法。其求解结果较于蒙特卡洛求解更为精确，更适合于问题二数据特点的策略决定。此时求出最后一代种群后，本文对其进行不同权值的目标取向分析，并给出不同策略。

2.3 问题三的分析

问题三可以分为新冠疫情影响指标的建立和对目标模型的惩罚项修正两步。

针对新冠疫情影响指标的建立，本文基于企业类型因子、行业类别因子和企业综合评估权值建立了企业违约概率指标。

其次，本文对短期收益目标函数添加硬违约概率惩罚项，对潜在损失目标函数添加软违约概率惩罚项。调整后的双目标规划模型，仍通过问题二中的改进 NSGA-II 算法进行求解。最终决策方案也依此得到讨论。

三、 模型假设

1. 假设对于同一综合信贷评判等级的企业，银行提供的贷款的额度相同。
2. 假设对于同一信誉评级的企业，银行所收取的贷款年利率相同。
3. 假设问题一和问题二中，企业违约的概率完全由该企业的信誉评级决定，A,B,C 类企业不会违规；而问题三中则由信誉评级与企业违约概率指标共同决定。
4. 假设所有企业的发票信息完整，且可以真实反映企业运营情况。
5. 假设企业的性质和所属行业可以从其名称中判断。

四、 符号说明

符号	意义
$\boldsymbol{X_i}$	企业对应的各项发票数据，$i = 1, 2.....14$，具体含义见表 2。
$\boldsymbol{f_i}$	评级为 i 类的企业的流失率。
$\boldsymbol{BR}$	企业商业信誉；用于衡量其商业行为是否诚信。
$\boldsymbol{ST}$	供求关系稳定性；用于衡量其收支失衡及贸易面额波动性。
$\boldsymbol{ES}$	企业实力；进项销项总额之和，用于衡量其商业体量大小。
$\boldsymbol{PB}$	企业收支失衡情况；用于衡量其收入支出是否接近。
$\boldsymbol{IP}$	企业进销项无效交易占比；即企业的无效单数占总单数比值。
$\boldsymbol{MP}$	企业进项购置取消比；即企业进项负数单数占进项总单数比值。
$\boldsymbol{TV}$	企业贸易面额波动性；即不同贸易单数之间的金额大小变化程度。
$\boldsymbol{z_i}$	目标规划函数（问题一）；$i = 1$ 时为银行收益函数，$i = 2$ 时为银行潜在损失函数。
$\boldsymbol{z_i'}$	目标规划函数（问题二）；$i = 1$ 时为银行收益函数，$i = 2$ 时为银行潜在损失函数。
$\boldsymbol{z_i''}$	目标规划函数（问题三）；$i = 1$ 时为银行收益函数，$i = 2$ 时为银行潜在损失函数。
$\boldsymbol{Z}$	复合目标规划函数；为 z_1 与 z_2 的加权求和。
$\boldsymbol{m_i}$	综合评级为 i 的企业的信贷额度比例。
$\boldsymbol{r_i}$	信誉评级为 i 的企业的贷款年利率。
$\boldsymbol{M_t}$	银行可供贷款的总金额（元）。
$\boldsymbol{ET}$	企业所属的行业类别；即制造业、建筑业等。
$\boldsymbol{EI}$	企业的公司类型；即个体、股份有限公司等。
$\boldsymbol{Ind_i}$	第 i 个企业的公司类型影响因子。
$\boldsymbol{Type_i}$	第 i 个企业的行业类别影响因子。
$\boldsymbol{T/G/GT}$	TOPSIS/灰色关联/GC-TOPSIS 分别对应的理想程度 (评分)。

五、 模型的建立与求解

5.1 数据的处理与评价指标的建立

5.1.1 附件 1、附件 2 数据的处理

根据附件 1 中企业信息数据，本文首先对 123 家公司的信誉评级和是否违约进行统计，结果见表1。由表1可知，其信誉评级可以基本代表其违约情况：D 类企业违约概率近似看作 1，而 A,B,C 类企业违约概率可以近似看作 0。因此可以给出假设：银行仅考虑给 A,B,C 类企业放贷，并且 A,B,C 类企业均不会违约（迟交贷款）。

表 **1**　**123** 家企业的信誉评级和违约关系

信誉评级	企业数量	违约企业数量	违约企业比例
A	27	0	0.00%
B	38	0	0.00%
C	34	2	5.88%
D	24	24	100.00%

根据附件 1、附件 2 中进项发票信息和销项发票信息数据，本文对每家企业的数据进行统计处理——计算出每一家企业的进/销项总额标准差，进/销项无效发票数量及交易额总量；进/销项有效正数发票数量及交易额总量，进/销项有效负数发票数量及交易额总量，共计 14 个初步指标（其符号表示见下表2），14 个初步指标的计算结果见附件）。

表 **2**　**14** 个初步指标符号及含义

	标准差	无效发票		有效正数发票		有效负数发票	
	总额	总额	数量	总额	数量	总额	数量
进项	X_1	X_2	X_3	X_4	X_5	X_6	X_7
销项	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}

考虑到发票数量及面额在一定程度上可以代表企业的信誉程度、进出项交易稳定性和实力等因素，因此本文在5.1.3中将在以上数据的基础上，进一步建立指标体系。

5.1.2 附件 3 数据的处理

根据附件 3 中的数据，本文首先分别对 A,B,C 类企业的客户流失率和其年利率做出散点图（图1）。由图初步分析得，其客户流失率 ($f_{A,B,C}$) 和年利率 (r) 类似对数函数关系。因此本文考虑用 Matlab 对其进行对数函数拟合, 可以给出以下拟合函数 (1)。由于已知 $r = 0.04$ 时，f 均为 0，因此其中 $0.04a + b = 1$。

拟合结果如下表3（系数值后的括号内表示系数置信区间，取 $\alpha = 0.05$），因此给出拟合方程 (2)。可以看出，$SSE, RMSE$ 均较小，且系数置信区间均不包括 0，表明拟合整体效果，系数拟合效果均良好，模型可信。其拟合图2也表明，真实值散点与拟合函数十分逼近，拟合模型可以基本表示 f 和 r 的关系。

$$f = c \cdot ln(ar + b) \tag{1}$$

$$\begin{cases} f_A &= 0.42ln(73.26r - 1.93) \\ f_B &= 0.44ln(59.67r - 1.39) \\ f_C &= 0.48ln(49.46r - 0.98) \end{cases} \tag{2}$$

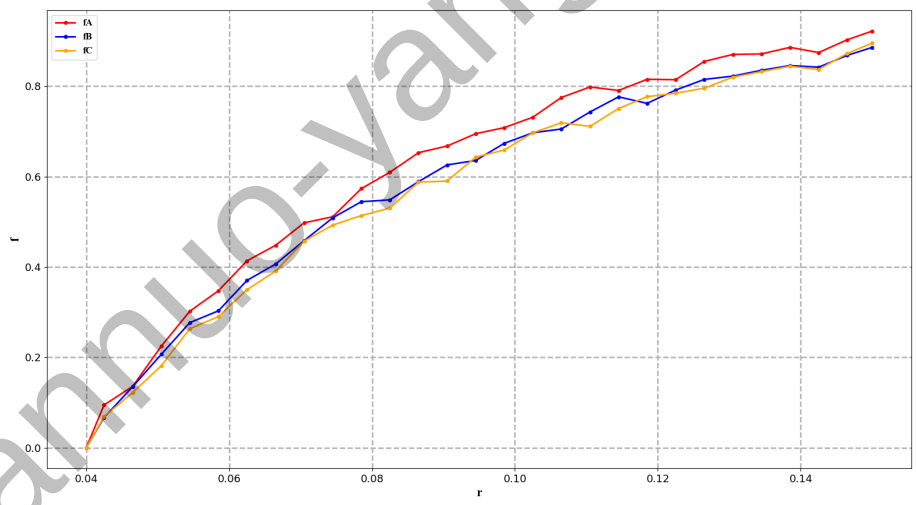


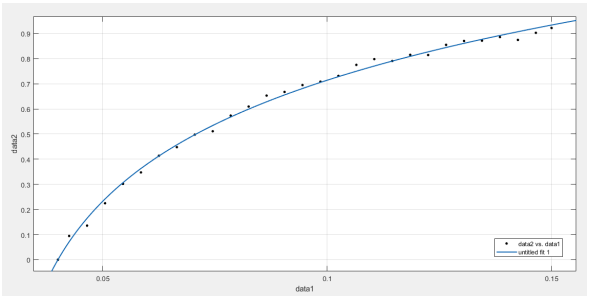
图 **1**　贷款年利率-信誉评级散点图

5.1.3 指标体系的建立

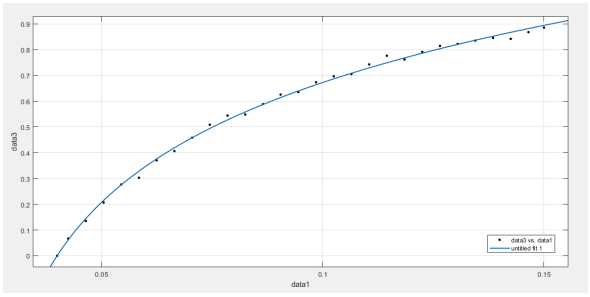
根据以上数据理解和处理，本文对中小微企业信贷综合评估，并建立如图3的二层指标体系 (最终计算得出的 BR, ST, ES 结果见附件 4)。

1. 商业信誉 (BR)

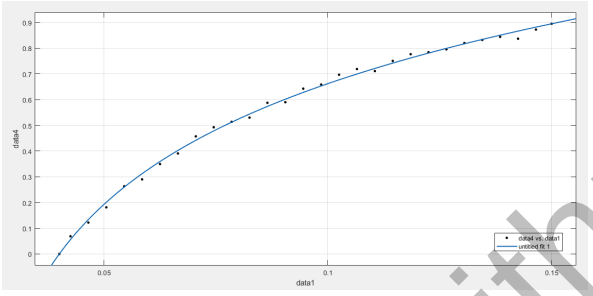
本文通过计算该企业的商业活动中“无效交易占比”(IP) 与由于其自身原因导致的



(a) A 类 **f-r** 拟合图



(b) B 类 **f-r** 拟合图



(c) C 类 **f-r** 拟合图

图 **2**　**A,B,C** 类企业 **f-r** 拟合图

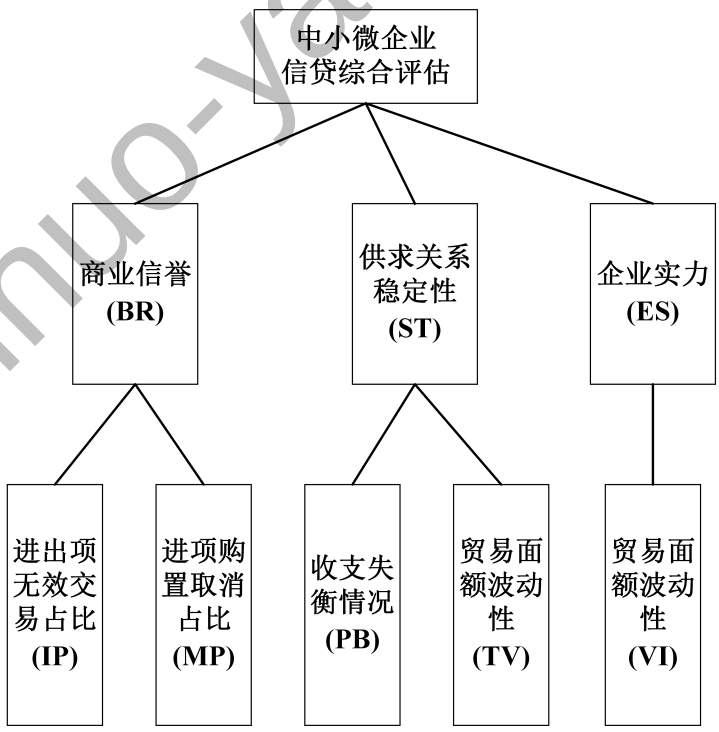


图 **3**　指标体系图

表 **3**　**A,B,C** 类公司 **f-r** 拟合结果

信誉评级	系数 a	系数 b(=1-0.04a)	系数 c	SSE	RMSE
A	73.26 (63.22, 83.29)	-1.9304	0.42 (0.3963, 0.451)	0.0071	0.0162
B	59.67 (53.14, 66.21)	-1.3868	0.44 (0.4179, 0.4661)	0.0041	0.0123
C	49.46 (43.51, 55.42)	-0.9784	0.48 (0.4502, 0.5108)	0.0048	0.0134

“进项购置项的取消占进项总数比例”(MP)，判断该企业在商业行为上的诚信度，进而判断对其信贷可能遭受的风险。经过数据合理化处理后公式如下：

$$IP = \frac{X_3}{X_3 + X_5 + X_7} \tag{3}$$

$$MP = \frac{X_{10}}{X_{10} + X_{12} + X_{14}} \tag{4}$$

$$BR = \frac{1}{IP^2 + MP} \tag{5}$$

2. 供求关系稳定性 (ST)

本文通过计算“收支平衡情况”(PB)，“贸易面额波动性”(TV) 来量化衡量该企业的稳定性进而判断对其信贷可能的损失。经过分析得，收支比例过高或过低都表示收支存在失衡，因此本文考虑用对数函数的绝对值表示收支平衡情况 PB；而 TV 则可以用每笔交易的标准差计算。保证 ST 为极大型指标，则公式如下：

$$PB = \frac{1}{\left| ln\frac{X_4+X_6}{X_{11}+X_{13}} \right|} \tag{6}$$

$$TV = X_1 + X_8 \tag{7}$$

$$ST = \frac{PB}{log_{10}^{TV}} \tag{8}$$

3. 企业实力 (ES)

本文根据计算其“进项与销项有效发票的总面额大小”，估计此企业的规模大小。公式如下：

$$ES = VI = X_4 + X_6 + X_{11} + X_{13} \tag{9}$$

5.2 问题一模型的建立与求解

5.2.1 基于熵权法权重矩阵处理的 TOPSIS-GC 综合分析评判模型

根据第5.1.3节指标体系的建立，BR, ST, ES 三个指标分别从商业信誉，供求关系稳定性，企业体量考虑信贷综合评估，三指标均为客观性指标。因此，如层次分析法等主观方法在此问题中被认为无法准确地分析评判企业综合信贷情况。

而单一的客观评价方法都有其局限性，如原始 TOPSIS 法并未考虑每一项指标的权重，仅简单以 1：1 的指标权重进行距离计算。因此本文考虑建立客观综合评价模型对每家企业进行评判，从而决定第一步贷款策略。

熵权法利用信息熵，计算出各个指标的权重，为多指标综合评价提供依据，具有数据客观性；TOPSIS 法根据有限个评价对象与理想化目标的接近程度进行排序，是在现有的对象中进行相对优劣的评价方法；灰色关联通过确定参考数据列和若干个比较数据列的几何形状相似程度来判断其联系是否紧密，反映了曲线间的关联程度。

根据文献 [1]，基于熵权法处理权重后的 GC-TOPSIS 综合评价方法较于单独的 GC 或 TOPSIS 评价法具有评价结果更稳定的特性，评价更贴合实际情况。基于此评价系统的方法，流程图可绘出见图4：

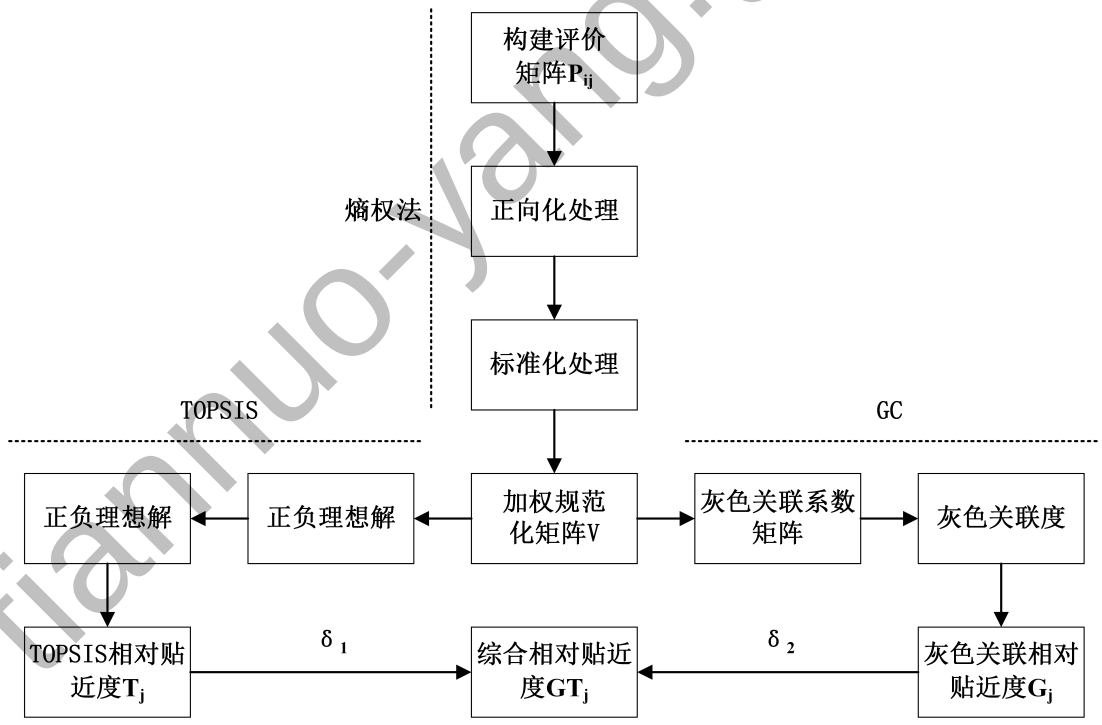


图 **4**　基于熵权的 **TOPSIS-GC** 评价方法流程图

依据以上流程图，具体步骤如下：

Step1 构建评价矩阵 (1), 其中，i 为不同评价指标，j 为不同评价个体，其个数分别用 m, n

表示。将其正向无量纲归一化处理后得到新矩阵 (2):

$$y = (C_{ij})_{mn} \quad i = 1.2 \ldots m \quad j = 1, 2 \ldots n \tag{10}$$

$$y^{'} = \left(C_{ij}^{'}\right)_{mn} \tag{11}$$

Step2 正向化处理，对于非及大型指标（极小型指标、中间型指标、区间型指标），需对其进行正向化处理，方可进行后续标准化处理。以极小型指标为例，其正向化过程为 (3) 式。

$$C_{ij}^{'} = \frac{max(C_{ij}) - C_{ij}}{max(C_{ij}) - min(C_{ij})} \tag{12}$$

Step3 标准化处理得到 $C_{ij}^{'}$，计算熵值 e_i 及权重 ω_i，进而建立基于熵权的规范化评价矩阵 V。

$$C_{ij}^{'} = C_{ij} / \sqrt{\sum_{i=1}^{n} C_{ij}^{2}} \tag{13}$$

$$e_i = -\frac{1}{\ln n} \sum_{j=1}^{n} C_{ij}^{'} \ln C_{ij}^{'}, \quad e_i \geq 0 \tag{14}$$

$$\omega_i = \frac{1 - e_i}{m - \sum_{i=1}^{m} e_i} \tag{15}$$

$$V = v_{ij} = \omega_i \left(C_{ij}^{'}\right)_{mn} \tag{16}$$

Step4 求解正负理想解 (R^+ 和 R^-) 和灰色关联系数矩阵 (Z^+ 和 Z^-), 式中 ρ 为分辨率，此处取 0.5，r_{ij} 为评价矩阵中对应数据，而 r_{ij}^{+} 为对应第 i 行的最大值。$\min \left|r_{ij}^{+/-} - r_{ij}\right|$ 与 $\max \left|r_{ij}^{+/-} - r_{ij}\right|$ 分别为两极最大差以及两级最小差，即从所有 i, j 取值中计算所得的最大值与最小值。

$$\begin{cases} R^{+} = \{\max v_{ij} \mid i = 1, 2, \ldots, m\} = \left\{v_1^{+}, v_2^{+}, \ldots, v_m^{+}\right\} \\ R^{-} = \{\min v_{ij} \mid i = 1, 2, \ldots, m\} = \left\{v_1^{-}, v_2^{-}, \ldots, v_m^{-}\right\} \end{cases} \tag{17}$$

$$\begin{cases} z_{ij}^{+} = \frac{\min\left|r_{ij}^{+} - r_{ij}\right| + \rho \max\left|r_{ij}^{+} - r_{ij}\right|}{\left|r_{ij}^{+} - r_{ij}\right| + \rho \max\left|r_{ij}^{+} - r_{ij}\right|}, \quad Z^{+} = \left(z_{ij}^{+}\right)_{mn} \\ z_{ij}^{-} = \frac{\min\left|r_{ij}^{-} - r_{ij}\right| + \rho \max\left|r_{ij}^{-} - r_{ij}\right|}{\left|r_{ij}^{-} - r_{ij}\right| + \rho \max\left|r_{ij}^{-} - r_{ij}\right|}, \quad Z^{-} = \left(z_{ij}^{-}\right)_{mn} \end{cases} \tag{18}$$

Step5 求解各评价单元与正负理想解的距离 D_j^+, D_j^- 和灰色关联度 S_j^+, S_j^-。

$$D_j^+ = \sqrt{\sum_{i=1}^{m} \left(R_j^+ - v_{ij}\right)^2}, j = 1, 2, \ldots n \tag{19}$$

$$D_j^- = \sqrt{\sum_{i=1}^{m} \left(R_j^- - v_{ij}\right)^2}, j = 1, 2, \ldots n \tag{20}$$

$$S_j^+ = \frac{1}{m}\sum_{i=1}^{m} z_{ij}^+, j = 1, 2, \ldots n \tag{21}$$

$$S_j^- = \frac{1}{m}\sum_{i=1}^{m} z_{ij}^-, j = 1, 2, \ldots n \tag{22}$$

Step6 求解相对贴近度和综合贴近度，分别由 D_j^+, D_j^- 计算出 TOPSIS 相对贴近度；S_j^+, S_j^- 计算出 GC 相对贴近度。由于 D_j^- 和 S_j^+ 分别表示 TOPSIS 中负理想解距离和与最大值距离的灰色关联度，均与每个评价单元的权重占比正相关，因此均可在一定程度上表示当前评价单元的“理想程度”（D_j^+ 和 S_j^- 同理，表示“偏离理想程度”）。对其按一定比例 (δ_1, δ_2) 对两种贴近度的中间变量 $(D_j^+, D_j^-; S_j^+, S_j^-)$ 加权可计算出综合贴近度 GT_j。

$$T_j = \frac{D_j^-}{D_j^- + D_j^+}, G_j = \frac{S_j^+}{S_j^+ + S_j^-}, GT_j = \frac{GT_j^+}{GT_j^+ + GT_j^-} \tag{23}$$

其中，GT_j^+, GT_j^- 分别计算如下。$\delta_1 + \delta_2 = 1$, 这里取 $\delta_1 = \delta_2 = 0.5$。

$$\begin{cases} GT_j^+ = \delta_1 D_j^- + \delta_2 S_j^+ \\ GT_j^- = \delta_1 D_j^+ + \delta_2 S_j^- \end{cases} \tag{24}$$

5.2.2 综合评判模型的求解

将本文中第5.1.3节中的 BR, ST, ES 作为指标，每家企业作为评价单元进行基于熵权处理矩阵的 TOPSIS-GC 结合分析评判，得出前十家结果如表4（所有企业结果见附录 4）。结合图5（图中 x,y,z 轴表示每家企业的三种排名基本相同，呈 $x = y = z$ 图象）可知，TOPSIS,GC,TOPSIS-GC 排名的结果基本相同，但综合评价法有更客观的实际代表性。

因此本文据 GT 排名对 99 家企业 (排除 D 类, 剩余 99 家) 进行排名分类，第一类为 TOPSIS-GC 排名在 1 至 33 名，第二类在 34 至 66 名，第三类在 67 至 99 名。定义每一类的信贷额度占银行本年总信贷额度的比例为 $m_I, m_{II}, m_{III}(m_I + m_{II} + m_{III} = 1)$。而由于假设对 A,B,C 类企业的信贷利率分别为 r_A, r_B, r_C, 因此可将前十家企业的三种排名方法的名次，以及企业对应的年利率，信贷额度记入表4中。

表 **4**　前十家企业的信贷综合评估排名，信贷额度，年利率

企业代号	TOPSIS 排名	GC 排名	GT 排名	信贷额度比例	信贷利率
1	2	2	2	m_I	r_A
2	6	6	6	m_I	r_A
3	7	7	7	m_I	r_C
4	5	5	5	m_I	r_C
5	15	14	16	m_I	r_B
6	8	8	8	m_I	r_A
7	9	10	9	m_I	r_A
8	13	16	14	m_I	r_A
9	19	19	19	m_I	r_A
10	22	24	22	m_I	r_B

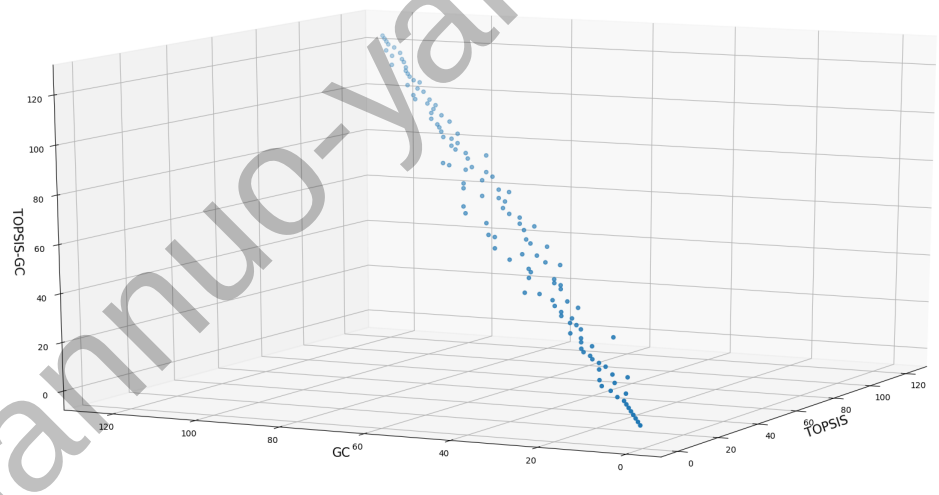


图 **5**　**G,T,GT** 排名三维散点图

5.2.3 双目标-短期收益和潜在损失规划模型

若银行的信贷利率过小，则会导致银行今年收益降低；若银行的信贷利率过高，则会导致客户流失率提高，造成潜在损失，进而导致未来收益降低。且由于银行信贷决策的目标并不绝对唯一，因此本文考虑建立双目标规划模型，对银行的短期收益和潜在损

失进行分析，模型如下：

$$min \quad z_1 = \frac{1}{\sum_{i=1}^{99} r_i m_i / n} \tag{25}$$

$$min \quad z_2 = \sum_{i=1}^{99} f(i) ES_i \tag{26}$$

目标函数中，$m_i \in \{m_I, m_{II}, m_{III}\}, r_i \in \{r_A, r_B, r_C\}$,$n$ 为按照综合评判模型求解排名分类的每一类企业个数，则 z_1 表示银行今年收益的倒数，目标一则是最大化银行当年的利益；$f(i)$ 表示当前企业的流失率，ES_i 为企业实力指标，则 z_2 表示银行潜在损失量，目标二则是最小化银行未来的潜在损失。

$$s.t. \begin{cases} m_I \geq m_{II} \geq m_{III} \\ r_A < r_B < r_C \\ 0.04 \leq r_A, r_B, r_C \leq 0.15 \\ 100000/M_t \leq m_I, m_{II}, m_{III} \leq 1000000/M_t \\ m_I + m_{II} + m_{III} = 1 \end{cases} \tag{27}$$

约束条件中，M_t 表示当年银行总的信贷额度。对综合信贷评判给分高的企业（I 类得分最大），应该给予更多或相等的信贷额度比例 m_i；对于企业信誉等级高的企业（A 类最高），给予更高的利率优惠 (根据题意不取等，保证利率优惠政策实施)；并且利率、贷款应该在题中规定范围取值。

5.2.4 双目标规划模型的求解

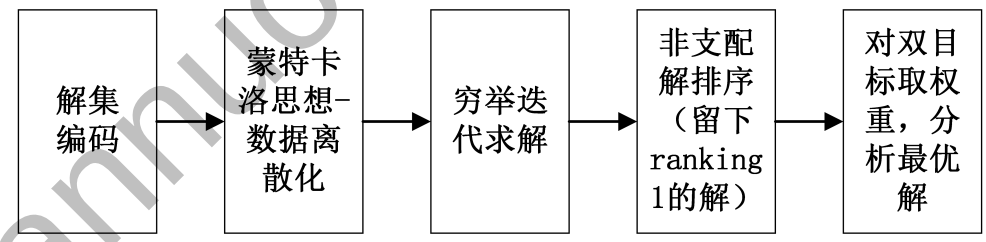


图 **6**　问题一双目标规划求解思路

针对5.2.3中的模型，本文考虑简化求解过程，使用蒙特卡洛思想对解集 $(m_I, m_{II}, m_{III}, r_A, r_B, r_C)$ 进行离散化。r 取附件 3 中给出的 29 种利率，m 取步长为 0.1，范围 (0,1) 的散点集，据此对所有解集进行穷举求解。对每一解集的两个目标函数 z_1, z_2 值，本文首先使用 non-dominated sorting(NS) 对其进行排序，求出 Pareto ranking 为 1 的解集（穷举法，NS 算法，及5.2.5中敏感性分析代码见附录A）。图7为所有解的 Pareto ranking 散点图（颜色按等级划分）。由图8可知， ranking=1 的解集总共有 122 个元素，即所有解中，共有 122 个未被支配解 (ranking1 解集及对应的 z_1, z_2 值见附件 4)。

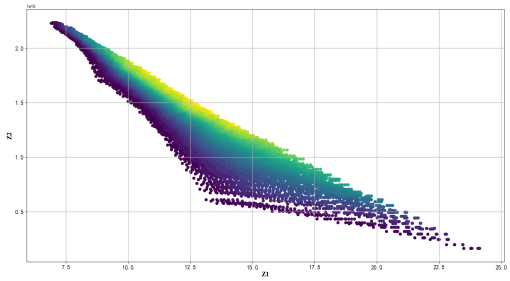


图 7　**ranking** 解集散点图

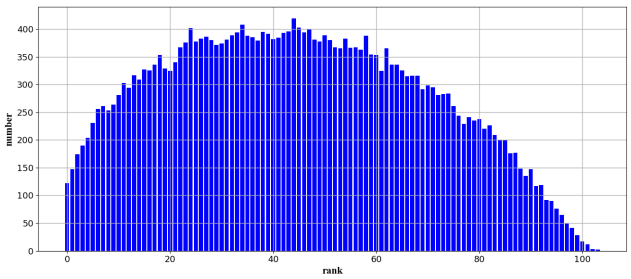


图 **8**　不同 **ranking** 等级对应解的数量

然而对于银行来说，显然要寻求更明确的方案，因此本文进一步对两个目标赋权值，并考虑不同权值比重下，银行选取方案对应的解集。此时需要对 z_1, z_2 进行量纲统一化。依题意，企业总数为 99 时，大致的贷款额度为 50000000 元（取近似中间值），因此我们将未来潜在企业流失的损失值 z_2 除以此参考值来消除量纲影响，即对于复合目标 Z 考虑 $(\omega_1 + \omega_2 = 1)$：

$$min \quad Z = \omega_1 z_1 + \omega_2 \frac{z_2}{50000000} \tag{28}$$

此处取权重组合 $(\omega_1 = \omega_2 = 0.5)$ 取进行分析：要使 Z 最小，根据附件 4 中 122 个 ranking1 的解计算，则排序得到最优方案解集 $(m_I, m_{II}, m_{III}, r_A, r_B, r_C)$ 的取值为 (0.4,0.3,0.3,0.04,0.0425,0.15)。因此说明，当目标一短期收益和目标二潜在损失取相等权值时，银行应该对信贷综合排名分类 I, II, III 类中的每家企业，分别给予当年银行总信贷额度的 $0.4/n, 0.3/n, 0.3/n, (n = 33)$。

而由于假设，问题一中银行当年信贷总额为 50000000，因此，I, II, III 每类中的每家企业分别获得信贷额度约 606060，454545，454545 元；且银行对 (A, B, C) 类企业分别设置年利率为 0.04，0.0425，0.15。

此时当年最大收益化目标 $z_1 = 13.3590$，潜在损失最小化目标 $z_2 = 357270000$, 即表示：当银行对两目标重视程度相等时，银行获得的最大利益为当年信贷总额度的 0.0749（若以当年总额度 50000000 元计算，则约为 3742795 元），潜在损失量约为 357270000 元。

5.2.5 问题一决策方案的敏感性分析

由于5.2.4中，假定了银行对短期收益和潜在损失量的重视程度相同（权重均为 0.5），不够全面客观，因此本文考虑对银行的信贷决策进行敏感性分析。如图9, 通过取步长为 0.01，不同的 $w_1(0 < w_1 < 1, w_2 = 1 - w_1)$，可以分别计算出总目标 Z 最优的解的序号。根据附录A，图中的方案一、方案二、、方案三、方案四、方案五的解 $(m_I, m_{II}, m_{III}, r_A, r_B, r_C)$ 见表5.

图9中，方案一、方案二、方案三、方案四、方案五对应的 ω_1 最大取值分别为：0.26，0.4 0，0.88，0.91，1. 通过查看曲线走势可以得出根据银行对当年利益与潜在损失量的

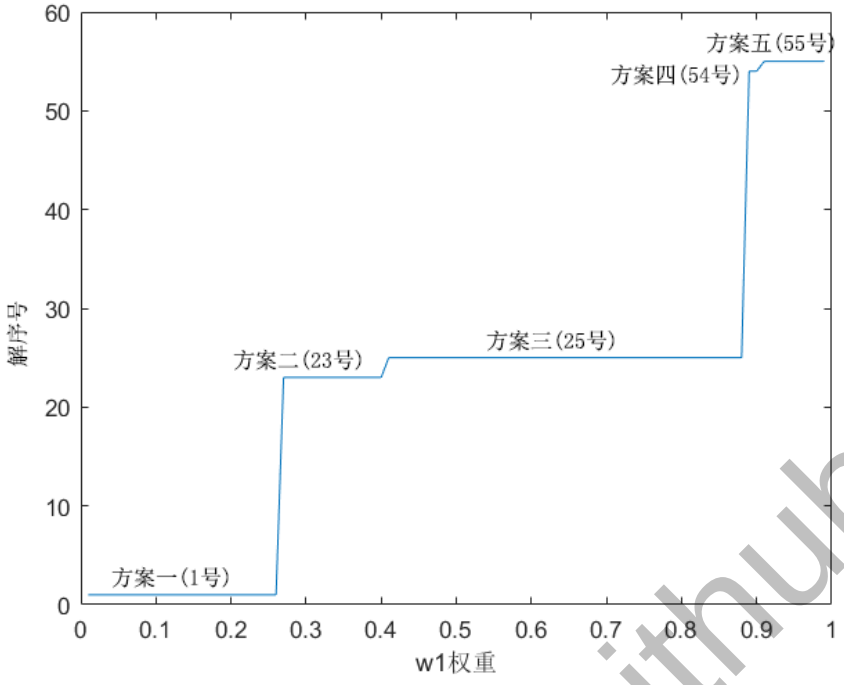


图 **9**　问题一敏感性分析

表 **5**　问题一五种方案对应解

方案	对应解 $(m_I, m_{II}, m_{III}, r_A, r_B, r_C)$
方案一	(0.4,0.3,0.3,0.04,0.0425,0.0465)
方案二	(0.4,0.3,0.3,0.04,0.0425,0.1425)
方案三	(0.4,0.3,0.3,0.04,0.0425,0.1500)
方案四	(0.4,0.3,0.3,0.04,0.1425,0.1500)
方案五	(0.4,0.3,0.3,0.04,0.1465,0.1500)

不同的重视程度下，银行有不同的信贷决策最优解。

- 若银行由于长期可持续化发展需求，对客户的长期合作关系十分重视，则应考虑**方案一**；
- 若银行较为看重潜在损失量，则应考虑**方案二**；
- 若银行对当年收益和潜在损失量的重视程度相近，则可以考虑**方案三**；
- 若银行较为看重当年收益，则应考虑**方案四**；
- 若银行在实行信贷后由于资金周转等问题，需要快速获得收益，则应考虑**方案五**。

5.3 问题二模型的建立与求解

5.3.1 随机森林分类模型的建立与求解

对比附件 1，2 中的数据可知，企业数据的区别在于是否有信贷记录和信贷评级。前 123 家企业有信贷评级和大量的发票数据，而后 302 家企业只有大量的发票数据。

据此本文考虑使用 sklearn 中的随机森林分类器，学习前 123 家企业在数据表2中 14 项初始指标（$X_{1,2,3...14}$，具体数值见附件 4）对于企业信誉等级 (A、B、C、D) 的分类关系，学习分类特征并对决策树数量进行迭代并寻找最优参数，对后 302 家企业的信誉等级进行同样标准的分类预测，操作见图10(代码见附录B)。

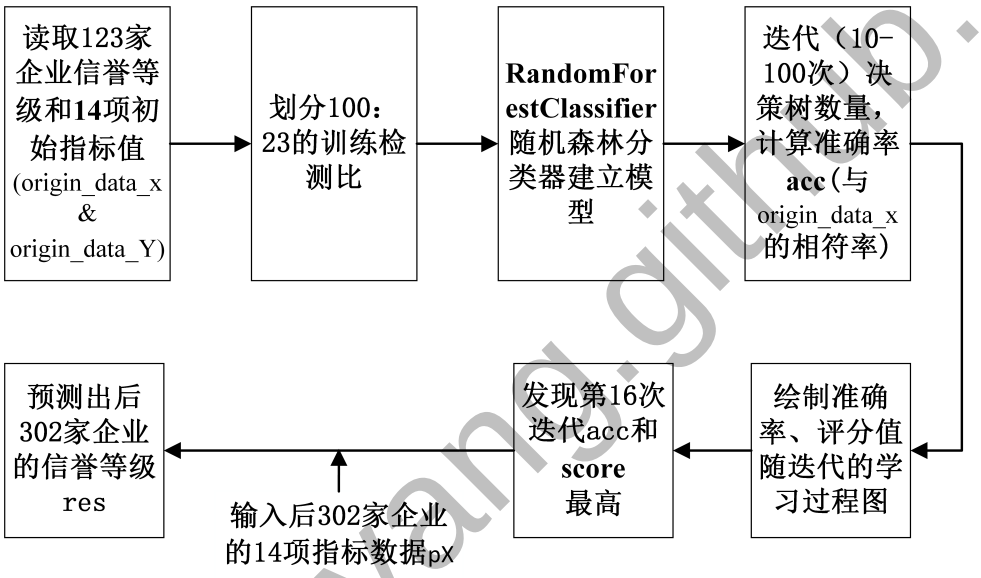


图 **10**　随机森林求解过程

运行后得到结果：图11中，横坐标为迭代次数（决策树数量-10），纵坐标为准确率和 sklearn 模型打分函数得分 scs（为便于观察，此处做放大处理 ($score = 10scs + 0.1$)。第 16 次（决策树数量为 26）得到最大准确率 $\boldsymbol{acc = 87.80\%}$。$acc > 80\%$，因此分类模型较为成功。依此对后 302 家企业进行结果预测记入附件 4，302 家企业的信誉评级分类情况见表6。

表 **6**　**302** 家企业的信誉评级预测

E124-E425	**A 类企业**	**B 类企业**	**C 类企业**	**D 类企业**
个数	59	140	39	64

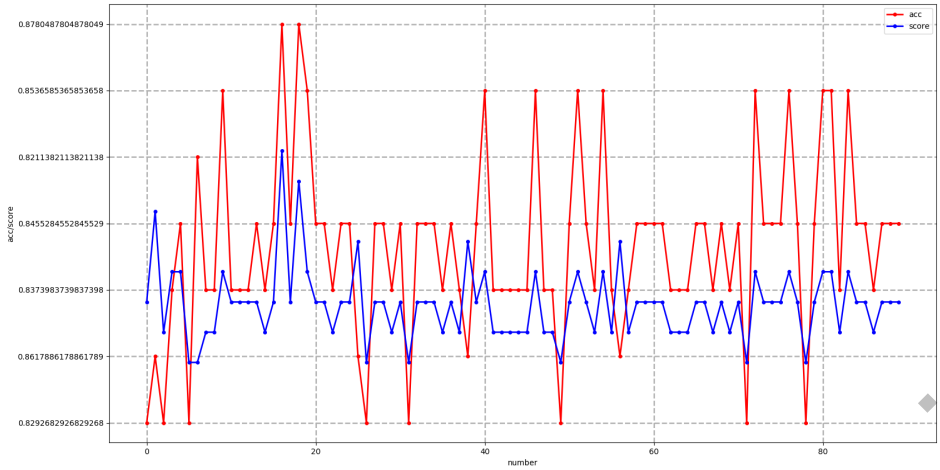


图 **11**　随机森林迭代图

5.3.2 综合评判模型和双目标规划模型的修正

由于5.3.1中已经求解出 302 家企业的信誉估计评级，因此本文进一步建立与问题一相同或类似的综合评判模型和双目标规划模型。

综合评判模型求解同问题一，计算结果记入附件 4。据此本文对后 302 家企业也进行如问题一解决思路的类别区分：由信誉等级区分年利率优惠 (r_A, r_B, r_C)；由综合评判得分区分信贷额度比例。由于问题二企业总数较问题一更多（由表6可知，留下 A,B,C 类企业共 238 家），因此考虑将综合信贷评分结果分为七类，第一类为 TOPSIS-GC 排名在 1 至 34 名，第二类在 35 至 68 名... 第七类在 205 至 238 名（记入附件 4）。

双目标规划模型中，本文首先修正模型的目标函数和约束函数：银行总贷款额度为一亿元，因此对 z_2 除以参考值 100000000, 再与 z_1 进行加和以构造 Z'。企业个数 i 总数被修正为 238，信贷综合评判分类每组 n 个数被修正为 34，$m_{I,II...VII}$ 顺次减小，且：$m_i \in \{m_I, m_{II}, m_{III}, m_{IV}, m_V, m_{VI}, m_{VII}\}$.

$$min \quad Z' = \omega_1 z_1' + \omega_2 \frac{z_2'}{100000000} \tag{29}$$

$$min \quad z_1' = \frac{1}{\sum_{i=1}^{238} r_i m_i / n} \tag{30}$$

$$min \quad z_2' = \sum_{i=1}^{238} f(i) ES_i \tag{31}$$

$$
s.t.\begin{cases} m_I \geq m_{II} \geq ... m_{VII} \\ r_A < r_B < r_C \\ 0.04 \leq r_A, r_B, r_C \leq 0.15 \\ 100000/M_t \leq m_i \leq 1000000/M_t \\ m_I + m_{II} + ... + m_{VII} = 1 \\ M_t = 100000000 \end{cases} \tag{32}
$$

5.3.3 双目标规划模型的求解

本文考虑到双目标规划模型求解中：

1. 问题一使用蒙特卡洛思想求解，在解集精确度方面存在一定局限性。尤其是对于 m_I, m_{II}, m_{III}, 取步长 0.1 尽管缩小了求解范围，但令解集却过于限定化——观察最优方案一至五的解集发现：m_I, m_{II}, m_{III} 均相等，分析认为，离散化后的数据精度有待提高，导致五种偏向情景下最优解取得等值 m_i；
2. 问题二对所有企业的综合信贷评判分组分成了 7 组，若仍使用蒙特卡洛思想离散化遍历求解，解集数据量过大。

因此，针对问题二中双目标规划问题的求解，本文考虑使用智能优化算法处理，以解决过度离散化带来的解集精度缺乏问题，$f(i)$ 则取公式(2)中的拟合函数进行计算。

由文献 [2, 3, 4, 5]，为了克服非支配排序遗传算法在多目标规划问题中的缺点，带有非支配排序快速算法和选择算子的 NSGA-II 算法被提出，且 NSGA-II 在寻找不同解集和收敛真正的 Pareto front 方面优于强度 Pareto 进化算法和 Pareto 存档进化策略算法。

如图12, 本文对 NSGA-II 进行修改，通过贪心算法的初始筛选加快收敛速度，并对交叉算子、变异算子进行当前解集形式的适配变化，以期有效、高效地解决问题二中短期收益和潜在损失量的双目标规划求解问题。

Step1 种群初始化，首先对解集进行编码，根据模型解特性，设置编码如图13, 共计 10 位数，前七位为 $m_i, i \in \{I, II, III...VII\}$, 后三位为 $r_i, i \in \{A, B, C\}$，据限定条件，随机生成初始种群。

本文对传统 NSGA-II 算法的种群初始化进行改进，加入贪心算法对初始种群中的个体进行初步筛选，以避免过多无效解的迭代，加快收敛速度，即对产生的种群个体序列使用贪婪策略进行检验，若符合要求加入到种群种，否则被抛弃。对阈值 cutoff，检验序列 $J = [j_1, j_2, ..., j_7]$，原始序列 $M = [m_1, m_2, ..., m_7]$，有：

$$
\sum_{i=1}^{7} m_i \times j_i > cutoff, m_i \in M, j_i \in J \tag{33}
$$

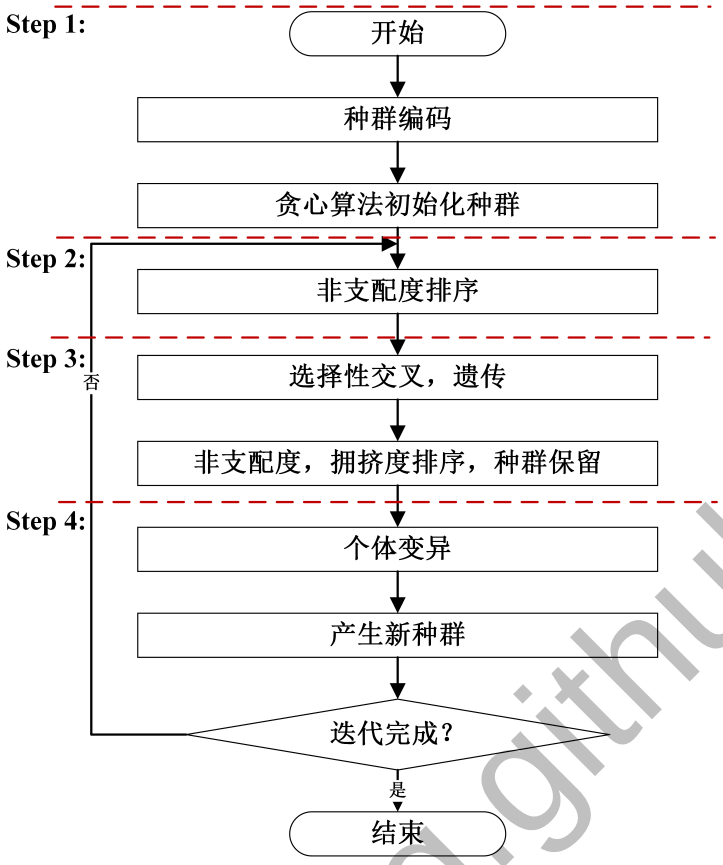


图 **12**　改进 **NSGA-II** 流程图

Step2 **非支配排序**，非支配排序方法同5.2.4，令排序结果中的 Pareto front　(ranking 1) 集合作为下一步被选择交叉的对象（若为奇数则弃掉 rangking 1 中最后一个个体）。

Step3 **选择性交叉遗传**，依据此处编码特性，对被选择个体两两采取如图13所示交叉方法：将父代 A 的 m,r 部分和父代 B 的 m,r 部分进行交换，每组得到两个新的子代个体。**排序，种群保留**，采用 Pareto ranking 对种群进行 ranking 分级，拥挤度排序对每一个 ranking 中的个体排序，据此抛弃种群排名低的个体，按种群个体数量不变进行保留。

Step4 **个体变异**, 种群中按一定概率 $P_{mutation}$ 每个个体采取如图13所示变异方法：随机选取两个 m_i, 进行加和，重分配；随机选取一个 r_i，进行一定范围的随机变化。

变异概率 $P_{mutation}$ 如下，其中 P_0 为初始概率设置，r_i 为第 i 个个体的 ranking 级别，m 为当前种群的 ranking 级别数量。按式(34)设置能使变异概率随着当前个体的 ranking 和迭代次数增加，从两个角度规避了解集陷入局部最优。

$$P_{mutation}(i)=p_0\times\frac{\max\left(r_1,r_2,\cdots,r_m\right)-r_i+1}{\max\left(r_1,r_2,\cdots,r_m\right)}\quad i\in(1,2,\cdots,m) \tag{34}$$

一次变异完成后，需判断基因是否违反范围约束, 若违反则需要进行基因片段修复（多次迭代排除异常片段）并重新判断；直到基因满足范围约束后，再对其进行一次基因序列修复（按约束条件排序），流程见图14。

经 Matlab 编写代码（代码见附录D）计算，取种群数量 1000，遗传 3000 代，初始

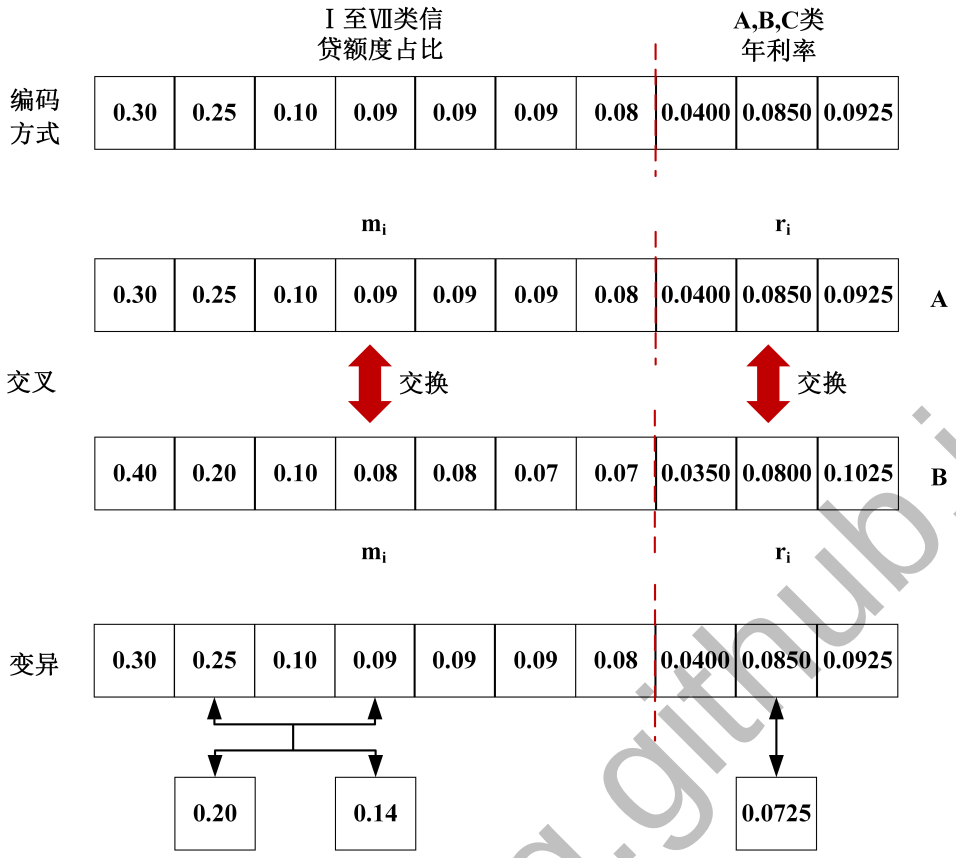


图 **13**　编码、交叉和变异

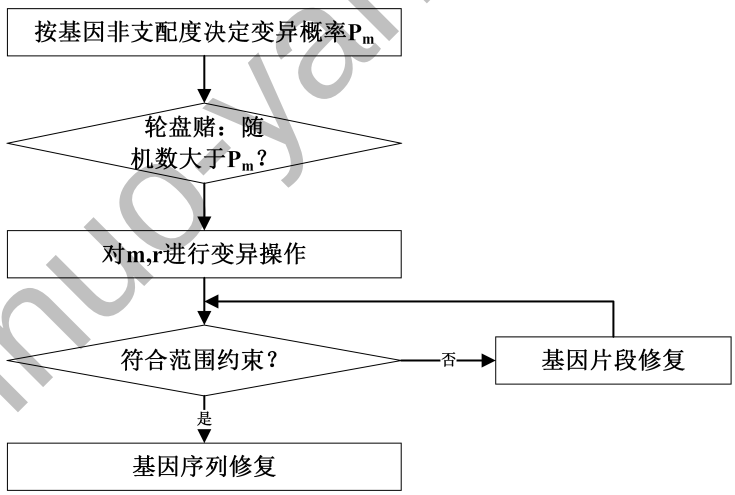


图 **14**　基因变异及修复

变异概率 P_0 取 0.5%. 迭代完成后，所有个体的 Pareto ranking 都为 1，因此取最后一次迭代的所有个体作为候选决策解集，同第5.2.5节进行银行不同目标取向情况下的方案决策，得到结果记入附录E，据其可得到问题二决策方案：

- 若银行更看重潜在损失量，则应对 I 到 VII 类企业分配 0.2946, 0.1821, 0.1548, 0.1054, 0.0952, 0.0840, 0.0589 的信贷额度；对 A,B,C 类企业分别设定 0.1482,0.1130,0.0720 的

年利率。此时银行当年收益为 4318340.253 元，潜在损失量为 1364678103 元 (方案六)。

- 若银行更看重当年收益，则可以按目标取向考虑附录E中的方案七、八、九、十。如方案七中银行当年收益为 5538684.348 元，较方案六有较大提高，但潜在损失量也提升到了 2701281655 元。（详细方案见附录E）

5.4 问题三模型的建立与求解

5.4.1 新冠疫情对企业违约概率的影响指标

2020 年新冠疫情的爆发，对不同类型、不同行业企业的生产经营和经济效益会产生不同程度的影响。因此本文首先考虑对 302 家企业进行“行业类别”(ET)、“企业类型”(EI) 的统计处理，分类结果见表7。根据文献 [6, 7], 本文对 ET, EI 进行分级，并将具体类别与分级对应记入表7。

其中，设置 $ET = \{Ind_i | i = 1, 2, 3, 4, 5\}$,1 表示新冠疫情对其 ET 有最大积极影响，而 3 表示新冠疫情影响可近似看作无，5 表示有最大消极影响；$EI = \{Type_i | i = 0, 1\}$,0 表示新冠疫情对此类企业无影响，1 表示有影响。

表 **7** **302** 家企业类型、行业类别分类统计

类别	个体 (1)	有限责任公司 (0)	股份有限公司 (0)	总计
建筑业 (4)	2	49	0	51
IT 技术服务行业 (3)	0	28	0	28
制造业 (4)	2	37	0	39
食品行业 (4)	1	5	0	6
交通运输业 (4)	0	13	0	13
人力资源服务业 (4)	0	35	0	35
文娱业 (5)	0	14	2	16
医药业 (1)	0	6	0	6
销售业 (4)	0	21	0	21
其他 (4)	79	8	0	87
总计	84	216	2	302

由于新冠疫情在“银行对企业的信贷方案决策”事件中，其影响主要体现在“企业是否倒闭，拖欠款项”。而此情况主要由企业综合实力、企业类别、行业类别决定。因此本文考虑按照式(35)对“企业违约概率”$P_{default}(i)$ 进行估计（值记入附件 4）：

$$P_{default}(i) = 100\left(\max\{k_i\} - k_i\right) \times \left(\frac{\text{Ind}_i}{3} + 0.1Type_i\right) \tag{35}$$

其中，i 为企业数，取 $i = 1, 2, 3, ..., 302$,k_i 为问题二中，进行归一化后的信贷综合评判权值 (数据见附件 4)，$(\max\{k_i\} - k_i)$ 将极大型指标转换为极小型指标，Ind_i 为第 i 个企业的企业类型影响因子，$Type_i$ 为第 i 个企业的行业类别影响因子。

5.4.2 双目标规划模型的违约概率惩罚项

对于短期收益目标，本文考虑使用硬违约概率惩罚项 $\lambda_1 P'_{default}(i)$ 对其进行修正。其中，(36)对初始违约概率作 0-1 化处理（银行当年利益的获得只需考虑企业是否能够付清信贷账款），而(37)的取值则保证此惩罚项为硬违约概率惩罚项——只要企业在 0.5 以上的概率违约，就对该企业收益进行巨大惩罚项 $(-\infty)$ 的加和。

$$P'_{default}(i) = \begin{cases} 1 & ,0.5 \le P_{default}(i) \le 1 \\ 0 & ,0 \le P_{default}(i) < 0.5 \end{cases} \tag{36}$$

$$min \quad z_1'' = \frac{1}{\sum_{i=1}^{238}(r_i m_i/n + \lambda_1 P'_{default}(i))} \tag{37}$$

$$\lambda_1 = -\infty \tag{38}$$

对于潜在损失量目标，本文考虑使用软违约概率惩罚项 $\lambda_2 P_{default}(i)$ 对其进行修正。其中,λ_2 的取值为定值 $ES_i(1 - f(i))$（软违约概率惩罚项），其实际意义则为：本来不会流失的企业以一定概率违约，则银行潜在损失多出的部分为 $ES_i(1 - f(i))P_{default}(i)$。

$$min \quad z_2'' = \sum_{i=1}^{238} f(i)ES_i + \lambda_2 P_{default}(i) \tag{39}$$

$$\lambda_2 = ES_i(1 - f(i)) \tag{40}$$

5.4.3 双目标规划模型的求解

根据5.4.2中经过惩罚项修改的目标函数，本文仍然使用问题二中的改进 NSGA-II 算法进行求解。则可得到新的调整策略为（具体方案数据见附录E）：

- 若银行对目标 z_2 取更多权重 $(\omega_2 > 0.31)$，则应考虑此时的方案十二：
- 若银行对当年收益的目标权重更大，则可考虑此时的方案十三、十四、十五、十六、十七。

六、 模型的评估与优化

6.1 模型的评估

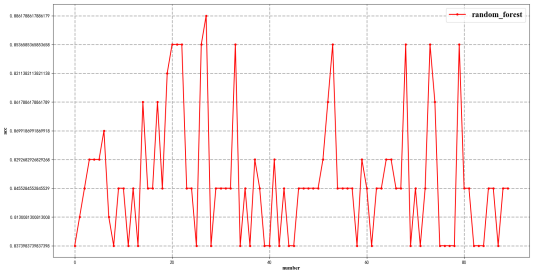
1. 本文在熵权法处理指标数据的基础上，使用 TOPSIS-GC 综合评判方法，既避免了主观方法的臆断性，也规避了单一客观评价方法的数据利用局限性；
2. 本文在对银行的信贷方案决策时，依据银行的不同目标取向给出了多种解决方案，具有较广的实用性；
3. 本文采用随机森林分类模型对企业信誉分级进行预测，模型准确率达到 87.80%。；
4. 本文采用离散化的穷举法、改进 NSGA-II 算法分别对问题一、问题二和三进行求解，其结果具有更高的精确性，可以推广至其他有多目标优化问题的求解中；
5. 本文在考虑银行信贷方案决策时，限定条件“每组企业中的每一家均具有相同信贷额度”，不够灵活，后续可以考虑对决策方案进一步详细化或企业定制化。

6.2 模型的优化

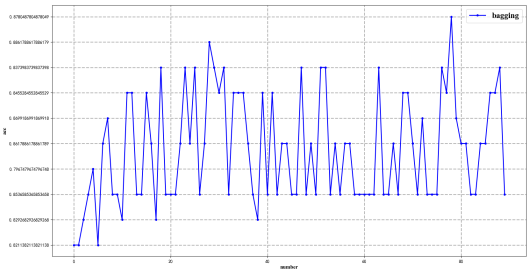
本文在确定问题二中 302 家企业的信誉评级时，仅采用了随机森林一种算法，在方法多样性方面有待提高。因此本文考虑使用另外两种深度学习的算法对问题一中的信誉评级和十四项初始指标进行学习，并分别对问题二中的企业信誉评级进行分类预测。如图15, 随机森林分类、装袋分类、极端随机树分类中最优精度分别为 88.62%, 87.80%, 85.37%。三种方法的预测结果相似度见表8（代码见附录B）。

表 **8**　三种方法分类预测结果相似度

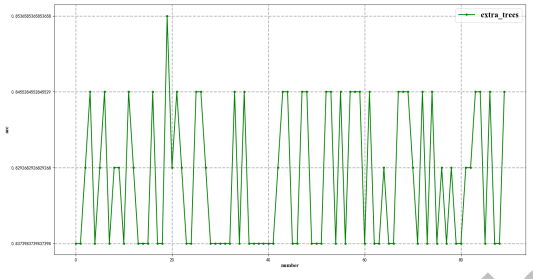
相似度	随机森林分类	袋装分类	极端树分类
随机森林分类	1	66.89%	62.95%
袋装分类	66.89%	1	59.93%
极端树分类	62.95%	59.93%	1



(a) 随机森林分类



(b) 装袋分类



(c) 极端随机树分类

图 **15**　三种深度学习算法分类器结果

参考文献

[1] 曹佳蕾 and 李停. 基于熵权 gc-topsis 的区域科技创新能力评价与实证. 统计与决策, 36(15):171–174, 2020. 42-1009/C.

[2] Zsofia Toth, Christoph Thiesbrummel, Stephan C Henneberg, and Peter Naude. Understanding configurations of relational attractiveness of the customer firm using fuzzy set qca. *Journal of Business Research*, 68(3):723–734, 2015.

[3] M Afshar-Bakeshloo, A Mehrabi, H Safari, M Maleki, and F Jolai. A green vehicle routing problem with customer satisfaction criteria. *Journal of Industrial Engineering International*, 12(4):529–544, 2016.

[4] Tasnim Ibn Faiz, Chrysafis Vogiatzis, and Md Noor-E-Alam. A column generation algorithm for vehicle scheduling and routing problems. *Computers & Industrial Engineering*, 130(APR.):222–236, 2019.

[5] Zhitao Xu, Adel Elomri, Shaligram Pokharel, and Fatih Mutlu. A model for capacitated green vehicle routing problem with the time- varying vehicle speed and soft time windows. *Computers & Industrial Engineering*, 137(Nov.):106011.1–106011.14, 2019.

[6] 何诚颖, 闻岳春, 常雅丽, and 耿晓旭. 新冠病毒肺炎疫情对中国经济影响的测度分析. 数量经济技术经济研究, 037(005):3–22, 2020.

[7] 李江文. 新冠肺炎疫情对服务类中小企业的影响及对策分析. 服务科学和管理, 009(003):P.126–138, 2020.

附录 A　穷举,NS, 敏感性分析 Matlab 代码

```
%domain.m
%% 穷举、NS、敏感性分析主函数代码
clc;
%% 蒙特卡洛主执行方法,最终结果为评级结果
mi=[];%对三种企业有不同的贷款大小
%对企业有不同的贷款利率
%不同贷款利率对不同评级企业也有不同概率流失客户
f=zeros(8,30000);%测验发现只需要记录26209个数据
k=1;
i=1;
while i<9
r1=1;
while r1<30
r2=r1+1;%r1,r2,r3从小到大所以如此赋初值
while r2<30
r3=r2+1;
while r3<30
c=1;
x=0;
y=0;
while c<99%每次要遍历99个企业
rt=0;mt=0;
if(custom(1,c)==1)%custom第一行是ABC评级来看应该对应r1r2r3哪个
rt=r1;
end
if(custom(1,c)==2)
rt = r2;
end
if(custom(1,c)==3)
rt=r3;
end
x=x+r(rt)*mi(i,custom(2,c))/33;%custom第二行是综合分析一二三评级判断获得贷款比例
y=y+perCons(custom(1,c),rt)*custom(3,c);
c=c+1;
end
f(1,k)=r(r1);
f(2,k)=r(r2);
f(3,k)=r(r3);
f(4,k)=mi(i,1);
f(5,k)=mi(i,2);
f(6,k)=mi(i,3);
f(7,k)=x;
f(8,k)=y;%记录收益与风险
r3=r3+1;
```

```
k=k+1;%k为总下标来记录
end
r2=r2+1;
end
r1=r1+1;
end
i=i+1;
end
% 初始数据
chromo(1,:) = f(7,:);
chromo(2,:) = f(8,:);
chromo = chromo';
% population初始种群数量
pop = size(chromo)(0);
% 目标函数数量
f_num = 2;
% 目标数为0
x_num = 0;
% 求倒数
f(7,:)=1./f(7,:);
% 赋变量名后，需要进行转置
% non_domination_sort( pop,chromo,f_num,x_num );
% 之后进行NSGA多目标规划即可

% 进行多目标规划
[mainF,chromo] = non_domination_sort( 29232,chromo,2,0);
% 拥挤度计算
chromo = crowding_distance_sort( mainF,chromo,2,0);

% 进行敏感性分析
c = [];
d = [];
for i=1:1000
[c(i),d(i)] = solve2(chromo(i,[1:7]),chromo(i,[8:10]));
end



% crowding_distance_sort.m
% NSGA多目标规划计算拥挤度代码
function chromo = crowding_distance_sort( F,chromo,f_num,x_num )
%计算拥挤度
%%%按照Pareto等级对种群中的个体进行排序
[~,index]=sort(chromo(:,f_num+x_num+1));
[~,mm1]=size(chromo);
temp=zeros(length(index),mm1);
for i=1:length(index)%=pop
```

```
temp(i,:)=chromo(index(i),:);%按照Pareto等级排序后种群
end

%%%对于每个等级的个体开始计算拥挤度
current_index = 0;
for Pareto_rank=1:(length(F)-1)%计算F的循环时多了一次空，所以减掉
%%拥挤度初始化为0
nd=[];
nd(:,1)=zeros(length(F(Pareto_rank).ss),1);
%y=[];%储存当前处理的等级的个体
[~,mm2]=size(temp);
y=zeros(length(F(Pareto_rank).ss),mm2);%储存当前处理的等级的个体
previous_index=current_index + 1;
for i=1:length(F(Pareto_rank).ss)
y(i,:)=temp(current_index + i,:);
end
current_index=current_index + i;
%%对于每一个目标函数fm
for i=1:f_num
%%根据该目标函数值对该等级的个体进行排序
[~,index_objective]=sort(y(:,x_num+i));
%objective_sort=[];%通过目标函数排序后的个体
[~,mm3]=size(y);
%通过目标函数排序后的个体
objective_sort=zeros(length(index_objective),mm3);
for j=1:length(index_objective)
objective_sort(j,:)=y(index_objective(j),:);
end
%%记fmax为最大值，fmin为最小值
fmin=objective_sort(1,x_num+i);
fmax=objective_sort(length(index_objective),x_num+i);
%%对排序后的两个边界拥挤度设为1d和nd设为无穷
y(index_objective(1),f_num+x_num+1+i)=Inf;
y(index_objective(length(index_objective)),f_num+x_num+1+i)=Inf;
%%计算nd=nd+(fm(i+1)-fm(i-1))/(fmax-fmin)
for j=2:(length(index_objective)-1)
pre_f=objective_sort(j-1,x_num+i);
next_f=objective_sort(j+1,x_num+i);
if (fmax-fmin==0)
y(index_objective(j),f_num+x_num+1+i)=Inf;
else
y(index_objective(j),f_num+x_num+1+i)=(next_f-pre_f)/(fmax-fmin);
end
end
end
%多个目标函数拥挤度求和
for i=1:f_num
```

```
nd(:,1)=nd(:,1)+y(:,f_num+x_num+1+i);
end
%第2列保存拥挤度，其他的覆盖掉
y(:,f_num+x_num+2)=nd;
y=y(:,1:(f_num+x_num+2));
temp_two(previous_index:current_index,:) = y;
end
chromo=temp_two;
end



% non_domination_sort.m
% non_domination_sort 初始种群的非支配排序代码
function [mainF,chromo] = non_domination_sort(pop,chromo,f_num,x_num)
%% 变量定义
%初始化Pareto等级为1
rank_Pareto=1;
mainF(rank_Pareto).ss=[];%Pareto等级为Pareto_rank的集合
perCons=[];%每个个体p的集合
%% 主循环
for i=1:pop
%% 计算被支配个数n和支配的解的集合s
%被支配个体数目n
perCons(i).n=0;
;%支配解的集合s
perCons(i).s=[];
%% 计算数值数目
for j=1:pop
less=0;% 目标函数值小于个体的目标函数值数目
equal=0;% 目标函数值等于个体的目标函数值数目
greater=0;% 目标函数值大于个体的目标函数值数目
for k=1:f_num
if(chromo(i,x_num+k)<chromo(j,x_num+k))
less=less+1;
elseif(chromo(i,x_num+k)==chromo(j,x_num+k))
equal=equal+1;
else
greater=greater+1;
end
end
if(less==0 && equal~=f_num)
%被支配个体数目n+1
perCons(i).n=perCons(i).n+1;
elseif(greater==0 && equal~=f_num)
perCons(i).s=[perCons(i).s j];
end
```

```
end
%种群中参数为n的个体放入集合F(1)中
if(perCons(i).n==0)
chromo(i,f_num+x_num+1)=1;%储存个体的等级信息
mainF(rank_Pareto).ss=[mainF(rank_Pareto).ss i];
end
end
%% 求Pareto等级为Pareto_rank+1的个体
while ~isempty(mainF(rank_Pareto).ss)
temp=[];
for i=1:length(mainF(rank_Pareto).ss)
if ~isempty(perCons(mainF(rank_Pareto).ss(i)).s)
for j=1:length(perCons(mainF(rank_Pareto).ss(i)).s)
perCons(perCons(mainF(rank_Pareto).ss(i)).s(j)).n
=perCons(perCons(mainF(rank_Pareto).ss(i)).s(j)).n - 1;%nl=nl-1
if perCons(perCons(mainF(rank_Pareto).ss(i)).s(j)).n==0
chromo(perCons(mainF(rank_Pareto).ss(i)).s(j),
f_num+x_num+1)=rank_Pareto+1;%储存个体的等级信息
temp=[temp perCons(mainF(rank_Pareto).ss(i)).s(j)];
end
end
end
end
rank_Pareto=rank_Pareto+1;
mainF(rank_Pareto).ss=temp;
end
end



% sensitive_analysis.m
% 敏感性分析代码,输入数据需要有包含f1、f2列的种群数据集
clc;
% data是种群数据集
x1=data(:,11);%x1与x2分别是两个目标函数解，在蒙特卡洛模拟结果中取得
x2=data(:,12);
i=1;%以0.01为步长进行敏感性分析，在计算时会除100
j=1;
index=0;%记录每次循环中的最优解下标
INDEX=[];%记录所有情况下最优解集合
FMIN=[];
while i<100
fmin=inf;
while j<1001
F=i/100*x1(j)+(100-i)/100*x2(j)/100000000;%F的计算方法
if(F < fmin)
fmin = F;
```

```
index = j;
end
j=j+1;
end
j=1;
INDEX(i)=index;
i=i+1;
end
X=0.01:0.01:0.99;
plot(X,INDEX);%作图进行敏感性分析图解
% xlabel("w1权重","fontsize",15);
% ylabel("解","fontsize",15);
data(8,:)*50000000
```

附录 B　随机森林分类模型 Pyhton 代码

```
# data_execute.py
# 文件操作主函数，需要的函数均封装好，目录下有对应文件即可调用
import pandas as pd
import time
import numpy as np
from sklearn import svm

# 附件路径
fileNames = ['./datas/附件1: 123家有信贷记录企业的相关数据.xlsx',
             './datas/附件2: 302家无信贷记录企业的相关数据.xlsx',
              './datas/附件3: 银行贷款年利率与客户流失率关系的统计数据.xlsx',
              './datas/f1_origin_data.xlsx', './datas/f2_origin_data.xlsx']


# .xlsx文件操作类

class FileExe:
    def __init__(self):
        pass

    # 获取表头
    def getSheetName(dataName):
        data_xlsx = pd.ExcelFile(dataName)
        return data_xlsx.sheet_names

    # 读取表内容
    @staticmethod
    def getSheetData(dataName, flag):
        data_xlsx = pd.ExcelFile(dataName)
```

```
        sheetName = data_xlsx.sheet_names[flag]
        # usecols 使用指定列，例[0,1,3]
        # index_col 索引列，例0
        # header 指定作为列名的行，例0
        df = data_xlsx.parse(sheet_name=sheetName, usecols=[0, 4])
        print(df)

    # 第一问计算6(共12)项指标
    @staticmethod
    def domain1_1(dataName, flag):
        start = int(time.time())
        data_xlsx = pd.ExcelFile(dataName)
        sheetName = data_xlsx.sheet_names[flag]
        # usecols 使用指定列，例[0,1,3]
        # index_col 索引列，例0
        # header 指定作为列名的行，例0
        df = data_xlsx.parse(sheet_name=sheetName, usecols=[0, 6, 7])
        print('文件读取完成')
        Bus = [] # 初始化企业列表
        for i in range(0, 123):
            # 下标0,1,2,3,4,5分别为
            # 进项无效（正数），进项有效（正数），进项有效（负数），
            进项无效正数数量，进项有效正数数量，进项有效负数数量
            # 或(销项无效（正数），销项有效（正数），销项有效（负数），
            销项无效正数数量，销项有效正数数量，销项有效负数数量)
            # 均初始化为0
            Bus.append([0, 0, 0, 0, 0, 0])

        for i in df.itertuples(): # 遍历
            # print(i[0], ' - ', i[1], " - ", i[2], " - ", i[3])
            if i[3] == '作废发票':
                Bus[int(i[1][1:]) - 1][0] += i[2] # 无效（正数）
                Bus[int(i[1][1:]) - 1][3] += 1 # 无效正数数量
            elif i[3] == '有效发票':
                if i[2] >= 0:
                    Bus[int(i[1][1:]) - 1][1] += i[2] # 有效（正数）
                    Bus[int(i[1][1:]) - 1][4] += 1 # 有效正数数量
                else:
                    Bus[int(i[1][1:]) - 1][2] += i[2] # 有效（负数）
                    Bus[int(i[1][1:]) - 1][5] += 1 # 有效负数数量

        print('列表行数' + str(len(Bus)))
        # 写入文件
        # temp1_1.xlsx存储进项数据
        # temp1_2.xlsx存储销项数据
        writer = pd.ExcelWriter('../BS/datas/temp1_2.xlsx')
        # Bus.insert(0, ['进项无效（正数）', '进项有效（正数）', '进项有效（负数）',
```

```
        '进项无效正数数量', '进项有效正数数量', '进项有效负数数量'])
    Bus.insert(0, ['销项无效（正数）', '销项有效（正数）', '销项有效（负数）',
    '销项无效正数数量', '销项有效正数数量', '销项有效负数数量'])
    df2 = pd.DataFrame(data=Bus)
    #  不写index会输出索引
    df2.to_excel(writer, 'Sheet', index=False)
    writer.save()
    print('文件写入完成')
    end = int(time.time())
    print('耗时: ' + str(end - start) + 's')

# 第二问计算6(共12)项指标
@staticmethod
def domain2_1(dataName, flag):
    start = int(time.time())
    data_xlsx = pd.ExcelFile(dataName)
    sheetName = data_xlsx.sheet_names[flag]
    # usecols 使用指定列, 例[0,1,3]
    # index_col 索引列, 例0
    # header 指定作为列名的行, 例0
    df = data_xlsx.parse(sheet_name=sheetName, usecols=[0, 6, 7])
    print('文件读取完成')
    Bus = [] # 初始化企业列表
    for i in range(0, 302):
        # 下标0,1,2,3,4,5分别为
        # 进项无效（正数）, 进项有效（正数）, 进项有效（负数）,
        进项无效正数数量, 进项有效正数数量, 进项有效负数数量
        # 或(销项无效（正数）, 销项有效（正数）, 销项有效（负数）,
        销项无效正数数量, 销项有效正数数量, 销项有效负数数量)
        # 均初始化为0
        Bus.append([0, 0, 0, 0, 0, 0])

    for i in df.itertuples(): # 遍历
        # print(i[0], ' - ', i[1], " - ", i[2], " - ", i[3])
        if i[3] == '作废发票':
            Bus[int(i[1][1:]) - 124][0] += i[2] # 无效（正数）
            Bus[int(i[1][1:]) - 124][3] += 1 # 无效正数数量
        elif i[3] == '有效发票':
            if i[2] >= 0:
                Bus[int(i[1][1:]) - 124][1] += i[2] # 有效（正数）
                Bus[int(i[1][1:]) - 124][4] += 1 # 有效正数数量
            else:
                Bus[int(i[1][1:]) - 124][2] += i[2] # 有效（负数）
                Bus[int(i[1][1:]) - 124][5] += 1 # 有效负数数量

    print('列表行数' + str(len(Bus)))
    # 写入文件
```

```
        # temp2_1.xlsx存储进项数据
        # temp2_2.xlsx存储销项数据
        writer = pd.ExcelWriter('../BS/datas/temp2_2.xlsx')
        # Bus.insert(0, ['进项无效（正数）', '进项有效（正数）', '进项有效（负数）',
         '进项无效正数数量', '进项有效正数数量', '进项有效负数数量'])
        Bus.insert(0, ['销项无效（正数）', '销项有效（正数）', '销项有效（负数）',
        '销项无效正数数量', '销项有效正数数量', '销项有效负数数量'])
        df2 = pd.DataFrame(data=Bus)
        #  不写index会输出索引
        df2.to_excel(writer, 'Sheet', index=False)
        writer.save()
        print('文件写入完成')
        end = int(time.time())
        print('耗时: ' + str(end - start) + 's')

    # 第一问计算有效正数标准差
    @staticmethod
    def domain1_2(dataName, flag):
        start = int(time.time())
        data_xlsx = pd.ExcelFile(dataName)
        sheetName = data_xlsx.sheet_names[flag]
        # usecols 使用指定列，例[0,1,3]
        # index_col 索引列，例0
        # header 指定作为列名的行，例0
        df = data_xlsx.parse(sheet_name=sheetName, usecols=[0, 6, 7])
        print('文件读取完成')
        Bus = [] # 初始化企业列表
        for i in range(0, 123):
            Bus.append([])

        for i in df.itertuples(): # 遍历
            # print(i[0], ' - ', i[1], " - ", i[2], " - ", i[3])
            if i[3] == '有效发票' and i[2] >= 0:
                Bus[int(i[1][1:]) - 1].append(i[2]) # 有效（负数）
        res = []
        for i in Bus:
            res.append(np.std(i, ddof=1).tolist()) # ddof=1参数可选，是否使用贝塞斯修正
        print('列表行数' + str(len(res)))
        print(type(res))
        # 写入文件
        # temp1_1.xlsx存储进项数据
        # temp1_2.xlsx存储销项数据
        writer = pd.ExcelWriter('../BS/datas/temp1_3.xlsx')
        df2 = pd.DataFrame(data=res)
        #  不写index会输出索引
        df2.to_excel(writer, 'Sheet', index=False)
        writer.save()
```

```
        print('文件写入完成')
        end = int(time.time())
        print('耗时: ' + str(end - start) + 's')

    # 第二问计算有效正数标准差
    @staticmethod
    def domain2_2(dataName, flag):
        start = int(time.time())
        data_xlsx = pd.ExcelFile(dataName)
        sheetName = data_xlsx.sheet_names[flag]
        # usecols 使用指定列, 例[0,1,3]
        # index_col 索引列, 例0
        # header 指定作为列名的行, 例0
        df = data_xlsx.parse(sheet_name=sheetName, usecols=[0, 6, 7])
        print('文件读取完成')
        Bus = [] # 初始化企业列表
        for i in range(0, 302):
            Bus.append([])

        for i in df.itertuples(): # 遍历
            # print(i[0], ' - ', i[1], " - ", i[2], " - ", i[3])
            if i[3] == '有效发票' and i[2] >= 0:
                Bus[int(i[1][1:]) - 124].append(i[2]) # 有效（负数）
        res = []
        for i in Bus:
            res.append(np.std(i, ddof=1).tolist())
        print('列表行数' + str(len(res)))
        print(type(res))
        # 写入文件
        # temp1_1.xlsx存储进项数据
        # temp1_2.xlsx存储销项数据
        writer = pd.ExcelWriter('../BS/datas/temp2_4.xlsx')
        df2 = pd.DataFrame(data=res)
        #  不写index会输出索引
        df2.to_excel(writer, 'Sheet', index=False)
        writer.save()
        print('文件写入完成')
        end = int(time.time())
        print('耗时: ' + str(end - start) + 's')

    # 第一问学习分类
    @staticmethod
    def domain1_3(dataName, flag):
        start = int(time.time())
        data_xlsx = pd.ExcelFile(dataName)
        sheetName = data_xlsx.sheet_names[flag]
        # usecols 使用指定列, 例[0,1,3]
```

```
        # index_col 索引列，例0
        # header 指定作为列名的行，例0
        df = data_xlsx.parse(sheet_name=sheetName,
        usecols=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13])
        print('文件读取完成')
        # Score = [] # 评级
        datas = [] # 数据集
        for i in range(0, 302):
            datas.append([])
        for i, j in zip(df.itertuples(), range(0, 302)): # 遍历
            # Score.append(i[1])
            for f in range(1, 15):
                datas[j].append(i[f])
        # print(Score)
        print(datas)
        print(len(datas))
        # clf = svm.SVC()
        # clf.fit(Score, datas) # 训练ＳＶＣ模型
        # result = clf.predict(Score) # 预测  测试样本
        # print(result) # 得出预测值

        # print('列表行数' + str(len(res)))
        # print(type(res))
        end = int(time.time())
        print('耗时: ' + str(end - start) + 's')


class FileExeStatic:
    def __init__(self):
        pass

    # 附件2预测信誉评级存入excle文件
    @staticmethod
    def writePre():
        # 此数据手动运行data_pre_1.py获得，为提高性能独立运行
    # 此处目前为空数组，需输入结果集来运行
        res = []

        writer = pd.ExcelWriter('./datas/temp2_5_1.xlsx')
        res1 = []
        for i in range(0, len(res)):
            res1.append([])
        for i, j in zip(res, range(0, len(res))):
            res1[j].append(i)
        print(res1)
        df2 = pd.DataFrame(data=res1)
        #  不写index会输出索引
```

```
            df2.to_excel(writer, 'Sheet', index=False)
            writer.save()
            print('文件写入完成')


    # 主方法
    if __name__ == '__main__':
        pass
        # todo
        # 第一问初始化附件1 12项指标
        # fe = FileExe()
        # fe.domain1_1(fileNames[0], 2) # 1或2
        # todo
        # 第二问初始化附件2 12项指标
        # fe = FileExe()
        # fe.domain2_1(fileNames[1], 2) # 1或2
        # todo
        # 第一问计算有效数标准差
        # fe = FileExe()
        # fe.domain1_2(fileNames[0], 2) # 1或2
        # todo
        # 第二问计算有效数标准差
        # fe = FileExe()
        # fe.domain2_2(fileNames[1], 2) # 1或2
        # todo
        # 第一问学习评价方法获取数据
        # fe = FileExe()
        # fe.domain1_3(fileNames[4], 0)
        # todo
        # 第二问预测企业信誉等级
        fes = FileExeStatic()
        fes.writePre()
        # todo
        #


    else:
        print('error')



    # random_forest_predict.pyxx
    # 随机森林分类器分类及预测算法
        （包含模型评估中的装袋分类和极端随机树分类，以及可视化展现、数据分析）
    from sklearn.ensemble import RandomForestClassifier, BaggingClassifier,
        ExtraTreesClassifier
```

```
from matplotlib import pyplot as plt
# 使用随机森林分类器和袋装分类器
import numpy as np
import pandas as pd

# 设置plt显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 导入data_ex_1数据列表，为提高性能独立运行
# 为节省篇幅，此处为空数组，需导入数据来运行代码
origin_data_x = []

# 为节省篇幅，此处为空数组，需导入数据来运行代码
origin_data_y = []

# 训练数据用于训练继承学习,测试数据用于计算随机森林评分
x_train = origin_data_x[0:100]
# 123个数据，选择前100个为训练数据
y_train = origin_data_y[0:100]
# 选择100个数据作为训练数据
x_test = origin_data_x[101:]
# 123个数据，选择前100个为训练数据，剩下的是测试数据
y_test = origin_data_y[101:]
# 前100个为训练数据，剩下的是测试数据

# 附件2 12项数据，来源：data_ex_1.py运行
# 为节省篇幅，此处为空数组，需导入数据来运行代码
pX = []


# 构建分类器，训练样本，预测得分
def try_different_method(clf):
    # sklearn随机森林评分,用做参考
    clf.fit(x_train, y_train)
    score = clf.score(x_test, y_test)
    res1 = clf.predict(origin_data_x)
    res2 = clf.predict(pX)
    sum = 0
    for i, j in zip(origin_data_y, res1):
        if i == j:
            sum += 1
    # print('原数据集正确率：' + str(sum / 123))
    return str(sum / 123), score, res2
    # print(res1)
    # print(res2)
```

```
accs1, accs2, accs3 = [], [], [] # 准确率
indexs = [] # 下标
scs1, scs2, scs3 = [], [], [] # 得分
res1, res2, res3 = [], [], [] # 结果集

for i in range(10, 100):
    indexs.append(i)
    clfs = {'random_forest': RandomForestClassifier(n_estimators=i),
    'bagger': BaggingClassifier(n_estimators=i),
           'extra': ExtraTreesClassifier(n_estimators=i)}
    for clf_method in clfs.keys():
        # print('分类器是:', clf_method)
        clf = clfs[clf_method]
        a, b, c = try_different_method(clf)
        if clf_method == 'random_forest':
            accs1.append(a)
            scs1.append(b)
            res1.append(c)
        if clf_method == 'bagger':
            accs2.append(a)
            scs2.append(b)
            res2.append(c)
        if clf_method == 'extra':
            accs3.append(a)
            scs3.append(b)
            res3.append(c)
print('随机森林分类准确率最大值: ' + str(max(accs1)))
print('n_estimators = ' + str(10 + accs1.index(max(accs1))))
print('此时附件2预测结果: ')
print(res1[accs1.index(max(accs1))])

print('袋装分类准确率最大值: ' + str(max(accs2)))
print('n_estimators = ' + str(10 + accs2.index(max(accs2))))
print('此时附件2预测结果: ')
print(res2[accs2.index(max(accs2))])

print('极端树分类准确率最大值: ' + str(max(accs3)))
print('n_estimators = ' + str(10 + accs3.index(max(accs3))))
print('此时附件2预测结果: ')
print(res3[accs3.index(max(accs3))])

# 画图可视化显示学习过程随参数变化的变化
# plt.plot(accs1, color='red', linewidth=2.0, marker='.', markersize=8)
# plt.plot(accs2, color='blue', linewidth=2.0, marker='.', markersize=8)
# plt.plot(accs3, color='green', linewidth=2.0, marker='.', markersize=8)
plt.scatter = 10000
```

```
    # 设置字体
    font1 = {'family': 'Times New Roman',
            'weight': 'normal',
            'size': 12,
            }
    font2 = {'family': 'Times New Roman',
            'weight': 'normal',
            'size': 18,
            }
    # # plt.legend(labels=['random_forest', 'bagging', 'extra_trees'], prop=font2)
    plt.xlabel('number', font1)
    plt.ylabel('acc', font1)
    plt.grid(linewidth=1.8, linestyle='--')
    # # plt.show()
    #
    plt.plot(accs1, color='red', linewidth=2.0, marker='.', markersize=8)
    plt.xlabel('number', font1)
    plt.ylabel('acc', font1)
    plt.grid(linewidth=1.8, linestyle='--')
    plt.legend(labels=['random_forest'], prop=font2)
    plt.show()
    plt.plot(accs2, color='blue', linewidth=2.0, marker='.', markersize=8)
    plt.xlabel('number', font1)
    plt.ylabel('acc', font1)
    plt.grid(linewidth=1.8, linestyle='--')
    plt.legend(labels=['bagging'], prop=font2)
    plt.show()
    plt.plot(accs3, color='green', linewidth=2.0, marker='.', markersize=8)
    plt.xlabel('number', font1)
    plt.ylabel('acc', font1)
    plt.grid(linewidth=1.8, linestyle='--')
    plt.legend(labels=['extra_trees'], prop=font2)
    plt.show()

    d1, d2, d3 = 0, 0, 0
    for aa, bb, cc in zip(res1[accs1.index(max(accs1))],
     res2[accs2.index(max(accs2))], res3[accs3.index(max(accs3))]):
        d1 += 1 if aa == bb else 0
        d2 += 1 if aa == cc else 0
        d3 += 1 if bb == cc else 0
    print(len(res1[accs1.index(max(accs1))]))
    print(d1 / 302)
    print(d2 / 302)
    print(d3 / 302)
```

附录 C　贪心算法 Python 代码

```
#greedy_strategy.py
# 贪心算法代码，用于第二问计算NSGA输入数据（选取输入数据集和）
import numpy as np
from matplotlib import pyplot as ply
import pandas as pd
import random


# 生成贪心随机数组
def spawnRandom():
res = [random.uniform(0.25, 0.34)]
i = 0
while i < 6:
r = random.uniform(0.034, 0.34)
if r <= min(res) and (sum(res) + r < 1) and i != 5:
res.append(r)
i += 1
elif i == 5:
res.append(1 - sum(res))
if max(res) <= 0.34 and min(res) >= 0.034:
res.sort(reverse=True)
i += 1
else:
return None
else:
return None
res1 = []
i = 0
r = random.uniform(0.04, 0.15)
res1.append(r)
while i < 3:
r = random.uniform(0.04, 0.15)
if r < min(res1):
res1.append(r)
i += 1
res.sort(reverse=True)
res1.sort(reverse=True)
res.append(res1[0])
res.append(res1[1])
res.append(res1[2])
return res


# 初始化贪心数组
```

```
data = []
test = [0.3, 0.25, 0.2, 0.15, 0.1, 0.05, 0.02]
while len(data) < 1000:
info = spawnRandom()
if info is not None:
data.append(info)
writer = pd.ExcelWriter('./tempdatas/t1.xlsx')
df2 = pd.DataFrame(data=data)
#  不写index会输出索引
df2.to_excel(writer, 'Sheet', index=False)
writer.save()
print('文件写入完成')
```

附录 D　NSGA-II　Matlab 代码

```
greedy_strategy.py
# 贪心算法代码，用于第二问计算NSGA输入数据（选取输入数据集和）
import numpy as np
from matplotlib import pyplot as ply
import pandas as pd
import random


# 生成贪心随机数组
def spawnRandom():
    res = [random.uniform(0.25, 0.34)]
    i = 0
    while i < 6:
        r = random.uniform(0.034, 0.34)
        if r <= min(res) and (sum(res) + r < 1) and i != 5:
            res.append(r)
            i += 1
        elif i == 5:
            res.append(1 - sum(res))
            if max(res) <= 0.34 and min(res) >= 0.034:
                res.sort(reverse=True)
                i += 1
            else:
                return None
        else:
            return None
    res1 = []
    i = 0
    r = random.uniform(0.04, 0.15)
    res1.append(r)
```

```
        while i < 3:
            r = random.uniform(0.04, 0.15)
            if r < min(res1):
                res1.append(r)
                i += 1
        res.sort(reverse=True)
        res1.sort(reverse=True)
        res.append(res1[0])
        res.append(res1[1])
        res.append(res1[2])
        return res


    # 初始化贪心数组
    data = []
    test = [0.3, 0.25, 0.2, 0.15, 0.1, 0.05, 0.02]
    while len(data) < 1000:
        info = spawnRandom()
        if info is not None:
            data.append(info)
    writer = pd.ExcelWriter('./tempdatas/t1.xlsx')
    df2 = pd.DataFrame(data=data)
    #  不写index会输出索引
    df2.to_excel(writer, 'Sheet', index=False)
    writer.save()
    print('文件写入完成')
```

```
    % domain.m
    % NSGA迭代主函数
    %% 初始过程
    %% 种群开始迭代并进化
    % data为初始数据集(经过贪心算法计算生成的)
    for i=1:500
        %非支配排序
        [mainF,data] = non_domination_sort(1000,data,2,10);
        %交叉生成第一代子群
        data = crossover(data);
        %非支配排序
        [mainF,data] = non_domination_sort(1000,data,2,10);
        %计算拥挤度
        data = crowding_distance_sort(mainF,data,2,10);
        %保留前100行种群(遗传)
        data = data([1:1000],:);
        %随机变异10个个体序列
```

```
    data = mutate(data);
%    %非支配排序
%    [mainF,data] = non_domination_sort(100,data,2,10);
%    %计算拥挤度
%    data = crowding_distance_sort(mainF,data,2,10);
end



%domain_solve.m
% 目标规划调用主函数
% data_origin是种群数据集，根据自己需要修改变量名
c = [];
d = [];
for i=1:1000
%    [c(i),d(i)] = solve(data_origin(i,[1:7]),data_origin(i,[8:10]));
    [c(i),d(i)] = solve2(data_origin(i,[1:7]),data_origin(i,[8:10]));
end



%non_domination_sort.m
% non_domination_sort 初始种群的非支配排序代码
function [mainF,chromo] = non_domination_sort(pop,chromo,f_num,x_num)
%% 变量定义
%初始化Pareto等级为1
rank_Pareto=1;
mainF(rank_Pareto).ss=[];%Pareto等级为Pareto_rank的集合
perCons=[];%每个个体p的集合
%% 主循环
for i=1:pop
    %% 计算被支配个数n和支配的解的集合s
    %被支配个体数目n
    perCons(i).n=0;
    ;%支配解的集合s
    perCons(i).s=[];
    %% 计算数值数目
    for j=1:pop
        less=0;% 目标函数值小于个体的目标函数值数目
        equal=0;% 目标函数值等于个体的目标函数值数目
        greater=0;% 目标函数值大于个体的目标函数值数目
        for k=1:f_num
            if(chromo(i,x_num+k)<chromo(j,x_num+k))
                less=less+1;
            elseif(chromo(i,x_num+k)==chromo(j,x_num+k))
                equal=equal+1;
            else
```

```
                greater=greater+1;
            end
        end
        if(less==0 && equal~=f_num)
            %被支配个体数目n+1
            perCons(i).n=perCons(i).n+1;
        elseif(greater==0 && equal~=f_num)
            perCons(i).s=[perCons(i).s j];
        end
    end
    %种群中参数为n的个体放入集合F(1)中
    if(perCons(i).n==0)
        chromo(i,f_num+x_num+1)=1;%储存个体的等级信息
        mainF(rank_Pareto).ss=[mainF(rank_Pareto).ss i];
    end
end
%% 求Pareto等级为Pareto_rank+1的个体
while ~isempty(mainF(rank_Pareto).ss)
    temp=[];
    for i=1:length(mainF(rank_Pareto).ss)
        if ~isempty(perCons(mainF(rank_Pareto).ss(i)).s)
            for j=1:length(perCons(mainF(rank_Pareto).ss(i)).s)
                perCons(perCons(mainF(rank_Pareto).ss(i)).s(j)).n
                =perCons(perCons(mainF(rank_Pareto).ss(i)).s(j)).n - 1;%nl=nl-1
                if perCons(perCons(mainF(rank_Pareto).ss(i)).s(j)).n==0
                    chromo(perCons(mainF(rank_Pareto).ss(i)).s(j),f_num+x_num+1)
                    =rank_Pareto+1;%储存个体的等级信息
                    temp=[temp perCons(mainF(rank_Pareto).ss(i)).s(j)];
                end
            end
        end
    end
    rank_Pareto=rank_Pareto+1;
    mainF(rank_Pareto).ss=temp;
end
end


%crowding_distance_sort.m
% NSGA多目标规划计算拥挤度代码
function chromo = crowding_distance_sort( F,chromo,f_num,x_num )
%计算拥挤度
%%%按照Pareto等级对种群中的个体进行排序
[~,index]=sort(chromo(:,f_num+x_num+1));
[~,mm1]=size(chromo);
temp=zeros(length(index),mm1);
```

```
for i=1:length(index)%=pop
    temp(i,:)=chromo(index(i),:);%按照Pareto等级排序后种群
end

%%%对于每个等级的个体开始计算拥挤度
current_index = 0;
for Pareto_rank=1:(length(F)-1)%计算F的循环时多了一次空，所以减掉
    %%拥挤度初始化为0
    nd=[];
    nd(:,1)=zeros(length(F(Pareto_rank).ss),1);
    %y=[];%储存当前处理的等级的个体
    [~,mm2]=size(temp);
    y=zeros(length(F(Pareto_rank).ss),mm2);%储存当前处理的等级的个体
    previous_index=current_index + 1;
    for i=1:length(F(Pareto_rank).ss)
        y(i,:)=temp(current_index + i,:);
    end
    current_index=current_index + i;
    %%对于每一个目标函数fm
    for i=1:f_num
        %%根据该目标函数值对该等级的个体进行排序
        [~,index_objective]=sort(y(:,x_num+i));
        %objective_sort=[];%通过目标函数排序后的个体
        [~,mm3]=size(y);
        objective_sort=zeros(length(index_objective),mm3);%通过目标函数排序后的个体
        for j=1:length(index_objective)
            objective_sort(j,:)=y(index_objective(j),:);
        end
        %%记fmax为最大值，fmin为最小值
        fmin=objective_sort(1,x_num+i);
        fmax=objective_sort(length(index_objective),x_num+i);
        %%对排序后的两个边界拥挤度设为1d和nd设为无穷
        y(index_objective(1),f_num+x_num+1+i)=Inf;
        y(index_objective(length(index_objective)),f_num+x_num+1+i)=Inf;
        %%计算nd=nd+(fm(i+1)-fm(i-1))/(fmax-fmin)
        for j=2:(length(index_objective)-1)
            pre_f=objective_sort(j-1,x_num+i);
            next_f=objective_sort(j+1,x_num+i);
            if (fmax-fmin==0)
                y(index_objective(j),f_num+x_num+1+i)=Inf;
            else
                y(index_objective(j),f_num+x_num+1+i)=(next_f-pre_f)/(fmax-fmin);
            end
        end
    end
    %多个目标函数拥挤度求和
    for i=1:f_num
```

```
            nd(:,1)=nd(:,1)+y(:,f_num+x_num+1+i);
        end
        %第2列保存拥挤度，其他的覆盖掉
        y(:,f_num+x_num+2)=nd;
        y=y(:,1:(f_num+x_num+2));
        temp_two(previous_index:current_index,:) = y;
    end
    chromo=temp_two;
    end



    %elitism.m
    % 精英保留函数，可以调用此函数筛选存留种群，也可以直接截取
    function chromo = elitism( pop,combine_chromo2,f_num,x_num )
    %精英保留策略
    [pop2,temp]=size(combine_chromo2);
    chromo_rank=zeros(pop2,temp);
    chromo=zeros(pop,(f_num+x_num+2));
    %根据Pareto等级从高到低进行排序
    [~,index_rank]=sort(combine_chromo2(:,f_num+x_num+1));
    for i=1:pop2
        chromo_rank(i,:)=combine_chromo2(index_rank(i),:);
    end
    %求出最高的Pareto等级
    max_rank=max(combine_chromo2(:,f_num+x_num+1));
    %根据排序后的顺序，将等级相同的种群整个放入父代种群中，直到某一层不能全部放下为止
    prev_index=0;
    for i=1:max_rank
        %寻找当前等级i个体里的最大索引
        current_index=max(find(chromo_rank(:,(f_num+x_num+1))==i));
        %不能放下为止
        if(current_index>pop)
            %剩余群体数
            remain_pop=pop-prev_index;
            %等级为i的个体
            temp=chromo_rank(((prev_index+1):current_index),:);
            %根据拥挤度从大到小排序
            [~,index_crowd]=sort(temp(:,(f_num+x_num+2)),'descend');
            %填满父代
            for j=1:remain_pop
                chromo(prev_index+j,:)=temp(index_crowd(j),:);
            end
            return;
        elseif(current_index<pop)
            % 将所有等级为i的个体直接放入父代种群
            chromo(((prev_index+1):current_index),:)
```

```
            =chromo_rank(((prev_index+1):current_index),:);
        else
            % 将所有等级为i的个体直接放入父代种群
            chromo(((prev_index+1):current_index),:)
            =chromo_rank(((prev_index+1):current_index),:);
            return;
        end
        prev_index = current_index;
    end
    end



    %mutate.m
    %% 变异函数，随机交换两个序列的两个m元素
    function temp = bianyi(chromo)
        %% 初始化变量
        temp = chromo;
        mutate_pro_max = max(temp(:,13));
        %% 计算变异几率
        % 开始随机变异,变异机制为:
        % 每个个体根据自己的排序等级，通过公式由基础变异率计算出自身变异率，
        % 然后迭代每个个体并生成随机数模拟是否发生变异
        for j=1:1000%循环开始
            mutate_rand = rand();
            if mutate_rand < 0.005*(mutate_pro_max-temp(j,13)+1)/mutate_pro_max%发生变异
                %% m变异
                index_m = floor(rand(1,2).*7+1);%变异随机m下标
                t1 = temp(j,index_m(1));
                t2 = temp(j,index_m(2));
                sum = t1+t2;
                if sum<0.34
                    t3 = (sum-0.034)*rand()+0.034;
                    % 基因片段修复，若基因值超出限制范围，则重新分配基因，直至符合要求
                    while sum-t3<=0.034
                        t3 = (sum-0.034)*rand()+0.034;
                    end
                    t4 = sum-t3;
                else
                    t3 = 0.306*rand()+0.034;
                    % 基因片段修复，若基因值超出限制范围，则重新分配基因，直至符合要求
                    while sum-t3>=0.34||sum-t3<=0.034
                        t3 = 0.306*rand()+0.034;
                    end
                    t4 = sum-t3;
                end
                temp(j,index_m(1)) = t3;
```

```
            temp(j,index_m(2)) = t4;
            % 进行基因序列修复，即对基因序列进行倒序排列后返回，防止返回错误基因
            temp(j,[1:7]) = sort(temp(j,[1:7]),'descend');% 基因序列倒序排列修复
            % r变异
            index_r = floor(rand().*3+8);%变异随机r下标
            temp(j,index_r) = 0.11*rand()+0.04;
            % 进行基因序列修复，即对基因序列进行倒序排列后返回，防止返回错误基因
            temp(j,[8:10]) = sort(temp(j,[8:10]),'descend');% 基因序列倒序排列修复
        end
    end
end



%crossover.m
%% 交叉函数，交叉（子）项的生成

function temp = jiaocha(chromo)
    temp = chromo;
    %% 开始随机交叉生成100个子种群
    for j=1:100
        index2_2 = floor(rand(1,2).*54+1);%交叉随机数组下标
        t1 = temp(index2_2(1),:);
        t2 = temp(index2_2(2),:);
        for k=1:7
            tem = t1(k);
            t1(k)=t2(k);
            t2(k)=tem;
        end
        temp=[temp;t1];
        temp=[temp;t2];
    end
end



%solve.m
% 目标规划函数代码
% 本函数是为了计算f1与f2而封装的
function [f1,f2] = solve(m,r)
% 传入mr都是数组，f1f2为两个目标函数值
% 为节省篇幅，此处为空数组，需导入数据来运行代码
guke=[];

    j=1;
    x=0;
    y=0;
```

```
    while j<239
    %第一次这里忘了把100个不同对象使用即k没有用所以导致结果都一样了
        x=x+r(guke(j,1))*m(guke(j,2))/34;
        y=y+dd(guke(j,1),r(guke(j,1)))*guke(j,3);
        j=j+1;
    end
    f1=1/x;%第一行为z1即利率
    f2=y;%第二行为z2即潜在风险
end

function y=dd(ra,r)
    if(ra==1)
        y=0.42*log(73.26*r-1.93);
    end
    if(ra==2)
         y=0.44*log(59.67*r-1.39);
    end
    if(ra==3)
         y=0.48*log(49.46*r-0.98);
    end
end



%solve2.m
% 经过改进的目标规划函数代码
% 本函数是为了计算f1与f2而封装的
function [f1,f2] = solve2(m,r)
% 传入mr都是数组，f1f2为两个目标函数值
    guke=[]
    j=1;
    x=0;
    y=0;
    while j<239
    %这里增加了惩罚项是与第二问函数中不同的地方
        x=x+r(guke(j,1))*m(guke(j,2))/34*abs(1-round(guke(j,4)));
        %这里增加了惩罚项是与第二问函数中不同的地方
        y=y+(dd(guke(j,1),r(guke(j,1)))+
        (1-dd(guke(j,1),r(guke(j,1))))*guke(j,4))*guke(j,3);
        j=j+1;
    end
    f1=1/x;%第一行为z1即疫情中的利率
    f2=y;%第二行为z2即疫情中的潜在风险
end

function y=dd(ra,r)
    if(ra==1)
```

```
            y=0.42*log(73.26*r-1.93);
        end
        if(ra==2)
            y=0.44*log(59.67*r-1.39);
        end
        if(ra==3)
            y=0.48*log(49.46*r-0.98);
        end
    end
```

附录 E　问题二、问题三求解结果

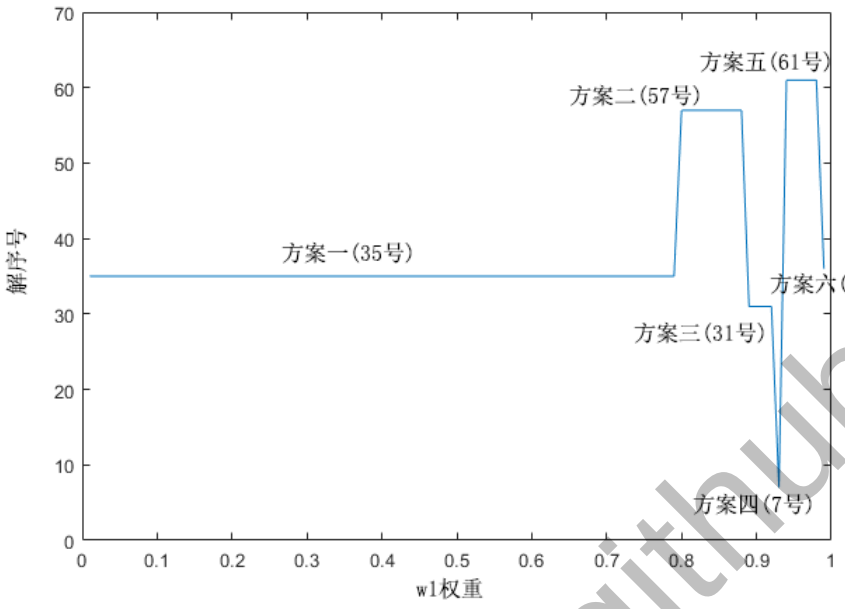


图 **16**　问题二敏感性分析

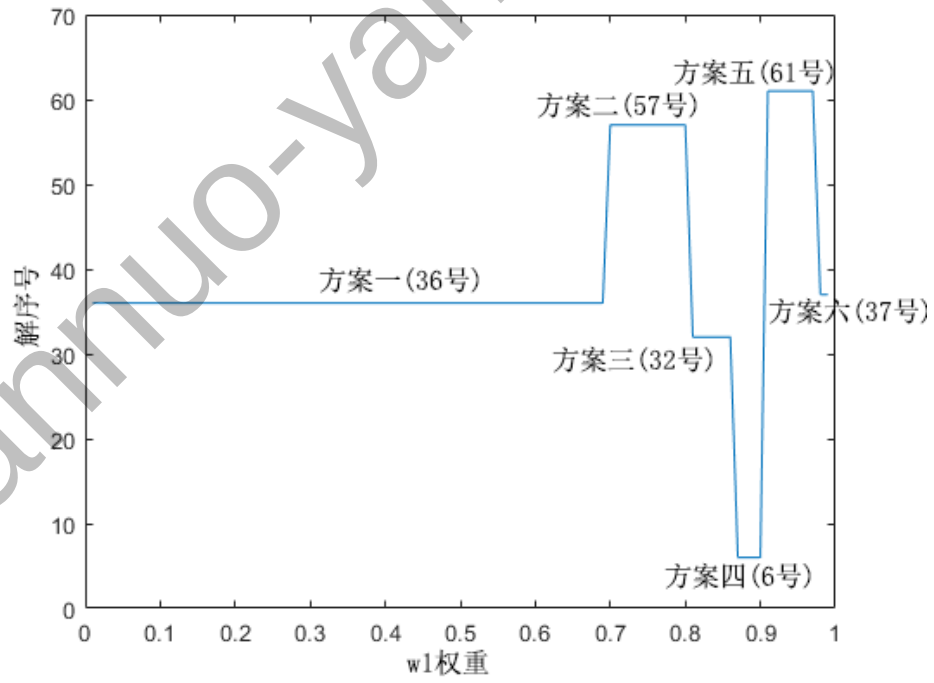


图 **17**　问题三敏感性分析

表 **9**　问题二六种方案

方案	ω_1 取值范围	对应解	当年收益	潜在损失
六	0.01-0.79	(0.2946,0.1821,0.1548,0.1054,0.0952, 0.0840,0.0589,0.1482,0.1130,0.0720)	4318340.253	1364678103
七	0.80-0.88	(0.3364,0.3027,0.1255,0.0597,0.0597, 0.0530,0.0473,0.0845,0.0758,0.0719)	5538684.348	2701281655
八	0.89-0.92	(0.3092,0.1880,0.1731,0.1661,0.1450, 0.0873,0.0627,0.1106,0.0686,0.0584)	5694738.511	3094056818
九	0.93	(0.2171,0.2131,0.2121,0.1769,0.1011, 0.0526,0.0501,0.1153,0.0754,0.0596)	9596362.097	11381294619
十	0.94-0.98	(0.2319,0.2014,0.1735,0.1227,0.1205, 0.0930,0.0570,0.0990,0.0698,0.0418)	11722070.48	14248880746
十一	0.99	(0.2424,0.2197,0.1679,0.1484,0.1190, 0.0679,0.0661,0.1271,0.0750,0.0749)	14102331.59	22321386868

表 **10**　问题三六种方案对应及解

方案	**w1** 取值范围	对应解	当年收益	潜在损失
十二	0.01-0.69	(0.3283,0.2128,0.1683,0.1654,0.0458, 0.0419,0.0376,0.0697,0.0400,0.0400)	3811420.746	9523582396
十三	0.70-0.80	(0.2581,0.2117,0.1733,0.1117,0.1073, 0.0750,0.0629,0.1362,0.1090,0.0602)	5288243.587	11218011247
十四	0.81-0.86	(0.3148,0.1585,0.1365,0.1256,0.1234, 0.0956,0.0423,0.1422,0.0770,0.0695)	5458788.913	11461888556
十五	0.87-0.90	(0.2646,0.2438,0.2080,0.0963,0.0820, 0.0684,0.0369,0.1145,0.0943,0.0866)	9233282.492	16292174060
十六	0.91-0.97	(0.3246,0.2701,0.1529,0.1296,0.0685, 0.0581,0.0350,0.1498,0.0957,0.0478)	11297488.47	18080855441
十七	0.98-0.99	(0.3031,0.2339,0.1291,0.1106,0.1065, 0.0811,0.0357,0.1462,0.1431,0.1422)	13457541.38	23560383330