中小微企业的信贷决策模型

摘　要

本文针对银行对中小微企业的信贷决策问题，通过衡量银行的信贷收益与风险，建立了贷款金额与贷款利率模型，在信贷总额和期限固定的条件下，给出了银行以利润最大化为目标时针对中小微企业的信贷策略，并讨论了在突发因素影响下，企业经营状况和银行信贷优惠政策的变化，并给出了调整后的银行信贷策略。

针对有信贷记录的企业贷款决策问题，本文首先以净销项收入、信誉评级、信贷优惠政策和客户流失系数作为变量，构建了贷款金额决策模型，对各企业的贷款金额相对系数进行评估，然后建立信贷违约概率模型，量化评估银行的信贷收益及预期的信贷损失，以贷款金额系数、违约系数和贷款利率为变量，建立了银行贷款利率决策目标函数，确定约束条件，最终求解出每个信誉评级所对应的贷款利率最优解，并根据固定的信贷总额和企业的贷款金额相对系数计算银行提供给各企业的最终贷款额。

针对无信贷记录的企业贷款金额与贷款利率决策问题，首先选取企业经营规模、供求关系稳定性、企业经营发展能力和对上下游企业的影响力四类因子，作为对企业信誉评级过程中考虑的主要因素，采用熵权法和层次分析法对各指标进行定权，然后利用 TOPSIS 法对企业进行综合信誉评级。将所得的信誉评级与财务数据信息代入第一问所建立的信贷决策模型中，求解出银行该批贷款的金额分配与定价策略，其中信誉评级为 A、B、C 的企业获得的信贷利率分别为 4.0622%、4.0622%、4.1189%，总贷款金额分别为 2688.64 万元、3761.49 万元、3498.30 万元，各等级企业所获得的贷款总金额比较接近。

针对影响企业的生产经营的突发因素可能改变银行的信贷策略这一问题，本文以新冠肺炎疫情为例，综合分析了该突发因素对企业经营和银行信贷优惠政策的影响。考虑到突发因素往往对不同行业的企业有不同的影响，因此本文将企业分为 19 个主要行业，分别衡量了该冲击对每个行业影响的严重程度和行业重要程度，结合冲击发生后的实际情况对企业的经营状况和信誉评级、银行的信贷政策系数进行调整，利用上述信贷决策模型给出了突发的外生冲击发生后该银行在年度信贷总额仍为 1 亿元时的信贷调整策略，并与冲击前的结果进行对比，发现冲击后银行实际提供贷款的企业数从 177 家减少至 146 家，信誉评级为 A、B、C 的企业获得的信贷利率分别为 4.2628%，4.2628%，4.2681%，总贷款金额分别为 1763.00 万元、3591.18 元、4581.72 万元。疫情发生后银行对各信誉等级企业的贷款额度分配更加均匀，贷款利率虽有所提高但各等级间的差异明显缩小，体现了信贷政策对社会资源的再分配作用。

关键词：决策模型；最优化；银行信贷；信誉评级

一、问题重述

中小微企业对我国的经济发展做出了重要贡献。随着金融环境的日渐复杂，中小微企业的融资问题日益凸显。中小微企业规模相对较小，往往缺少抵押资产，因此对商业银行而言，对中小微企业发放贷款需要建立较严格的风险管理机制。银行通常依据企业的信誉评级、经营状况、供求关系等因素决策是否对企业发放贷款以及发放贷款的金额，并对信誉较好的企业给予利率优惠。

基于上述背景，需要建立数学模型解决以下问题：

某银行对确定放贷的企业设置 10~100 万元的贷款额度，年利率为 4%~15%，贷款期限为 1 年。

(1)根据附件 1 中有信贷记录企业的相关数据，以及贷款利率与客户流失率关系的 2019 年统计数据，量化分析企业的信贷风险，给出该银行年度信贷总额固定时对这些企业的信贷策略。

(2)根据附件 2 中无信贷记录企业的相关数据，量化分析企业的信贷风险，给出该银行年度信贷总额为 1 亿元时对这些企业的信贷策略。

(3)考虑可能发生的、影响企业的生产经营和经济效益的突发因素，对不同行业、不同类别企业的不同影响。分析这些因素对附件 2 中无信贷记录企业的信贷风险的综合影响，由此调整上述信贷策略，给出该银行年度信贷总额为 1 亿元时调整后的策略。

二、问题分析

2.1 问题一

根据附件 1 的 123 家企业的信誉评级以及经营状况，首先需建立银行的贷款金额决策模型，对企业的贷款金额系数进行评估，然后建立信贷违约概率模型。由于贷款利率与信誉评级高度相关，因此假设对相同评级的企业采用同一贷款利率，量化评估银行的信贷收益及预期的信贷损失，建立银行贷款利率决策目标函数，求解出每个信誉评级所适用的贷款利率最优解，并根据固定的信贷总额和每

家企业的贷款金额系数来确定银行对每家企业的最终贷款额。

2.2 问题二

问题二将沿用第一问所建立的信贷策略决策模型。由于附件 2 中 302 家企业的信贷记录缺失，因此需要利用附件 2 中企业的财务数据和企业信息，从各方面综合衡量企业的经营状况，对企业进行模拟信誉评级。并根据信誉评级计算企业的贷款金额系数，根据银行贷款利率决策目标函数，求解适用于附件 2 企业的贷款利率最优解，并根据固定的信贷总额和每家企业的贷款金额系数来确定银行对每家企业的最终贷款额。

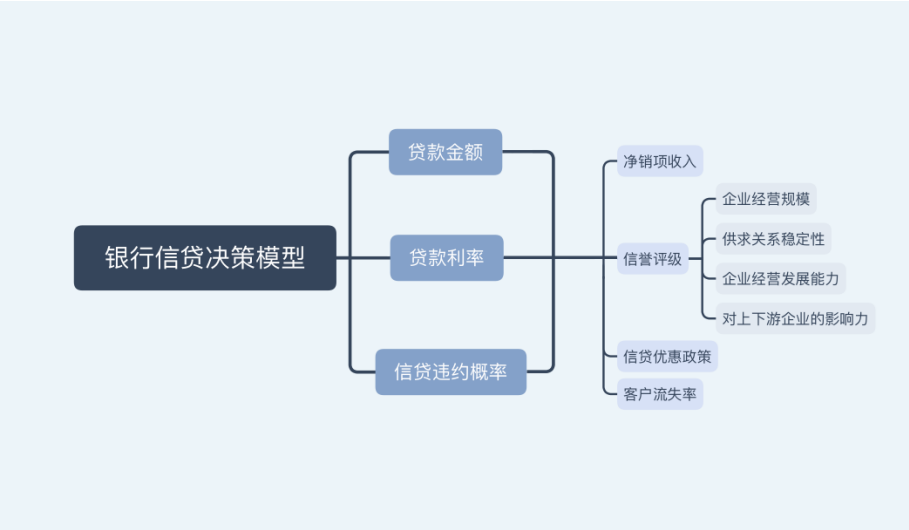

图 1 信贷决策分析

2.3 问题三

综合考虑可能发生的、影响企业生产经营和经济效益的突发因素，以新冠肺炎疫情为例，衡量该冲击对不同行业、不同类别的企业的影响。对企业的经营状况和信誉评级、银行的信贷政策系数进行调整，给出调整后的最优贷款金额和贷款利率。

三、模型假设

1.假设银行提供的信贷利率仅对客户流失率有影响，而不对每笔贷款金额产生影响。若客户流失，则银行给该客户的贷款金额为零，否则银行按照决策所定贷款额，一次性全额支付给该客户。

2.假设银行提供的贷款额与企业的经营规模有关。

3.银行提供的贷款利率与企业的信用相关，因此假设银行对同一信誉评级的企业实行无差别贷款利率。

4.假设企业的信誉评级与其经营能力、供求关系稳定性直接相关。

5.假设银行所贷企业还款仅有全额还款或者全部计入资产减值损失两种，因中小微企业缺少抵押资产，故不考虑部分还款的情况，即违约回收率为零。

6 假设不同行业的企业，其与上下游企业的交易频率以及交易额可以进行横向对比。

7.假设银行的信贷优惠政策与客户的信誉评级两者独立地影响贷款金额。

8.出于对信用风险的规避，假设银行不对信用评级为 D 的企业发放贷款。

四、变量说明

符号	说明
m_x_i	企业贷款额相对系数
m_c_i	企业i的贷款金额系数，表示贷款额占总贷款额的相对比例
m_i	银行提供给企业i的贷款金额
r_i	银行提供给企业i的贷款利率
K	银行该批一年期信贷总额
LR_i	银行对企业i的可贷率
l_i	银行对企业i的信誉评级
$L_i(l)$	信誉评级l对应的可贷额度比率
DC_i	银行对企业i的优惠政策
$ref(r,L)$	客户流失系数，表示是否流失该客户
p_i	利率r和信誉评级l决定的客户流失率
D_i	违约系数，表示预期企业i是否违约

NP	银行发放贷款的净本息
T_{Ii}	企业i的进项交易额
T_{Ei}	企业i的销项交易额
Br_i	企业i的有效发票率
G_i	企业i交易额的年增长率
F_i	企业i对产业链上下游的影响力

五、模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 信贷决策影响因素及其机制分析

本问要求分析 123 家企业的信贷风险并在银行年度信贷总额固定时，建立银行信贷决策模型，给出银行对样本企业的信贷策略。

我们将在附件 1 给出的数据的基础上，分别考虑银行的信贷策略，包括银行对各企业的贷款金额m_i以及每笔贷款的利率r_i。

5.1.1.1 贷款金额决策模型

1.净销项收入

本文主要基于以下原理计算银行对各企业的贷款金额：在实践中，银行发放的贷款金额通常以借款人的偿债能力为主要衡量指标，以降低该笔债务到期不能收回本息的风险。因此，企业最终得到的贷款数额取决于其贷款期末的偿债能力。中小微企业的资金流通渠道主要体现为主营产品/服务的支出成本和销售收入，融资渠道主要为银行借款，通过其他融资渠道筹得资金来归还银行借款的可能性较低。因此，可以忽略其他融资渠道对偿债能力的影响，通过分析中小微企业的进、销项发票信息，以其净销项收入作为单个企业获得银行审批贷款额的主要决

定因素较为合理。

净销项收入公式表示如下：

$$\overline{NT_i} = \bar{T}_{Ei} - \bar{T}_{Ii}$$

其中，$\bar{T}_{Ei}$为企业i的年均销项收入，$\bar{T}_{Ii}$企业i的年均进项支出。由于作废发票记录的交易未能真实发生，因此不计入其中；负向发票记录的交易发生退货或退款，因此予以扣除。

2.信誉评级与信贷优惠政策

除净销项收入外，银行对企业的信誉评级和信贷优惠政策也直接决定企业能够获批的贷款金额。信用等级越高的企业，越能申请到与收入高度匹配的贷款金额；银行对特定的行业或某个企业采取优惠政策，也将提高该企业的可贷额度上限。因此，我们建立可贷率这一指标，来衡量两者对贷款额的影响：

$$LR_i = L_i \times DC_i$$

其中，L_i为表示企业的信誉评级对应的可贷额度比率，与信誉评级相对应，DC_i为企业因符合银行的优惠政策条件而提高的贷款额度比例。四类信誉评级下L_i的具体数值如下：

信誉评级l	可贷额度比率L
A	1
B	0.8
C	0.5
D	0

假设正常情况下银行并没有对不同行业的企业实行差别待遇，因此这一问中企业并未因信贷优惠政策而提高贷款额度，$DC_i = 1$。当市场受到外生冲击时，企业正常的经营活动无法持续，不同行业由于经营特征和盈利模式的不同，其经营活动将受到不同程度的干扰，此时才需要银行实施有差别的信贷优惠政策，以重点扶持部分受到严重冲击的行业或国家重点支持的行业。

3.客户流失率

客户流失率是银行由于贷款利率过高而失去潜在的拥有特定信誉评级的客户的比例。一般情况下，利率越高，信用越好的客户流失率越大。因此，在制定针对不同信誉评级客户的利率时，银行必须在贷款利息收入和客户对信贷利率的

接受程度之间做出权衡，以保证贷款整体利润最大化。

构建银行客户流失率数据与利率、客户信誉评级的函数关系，拟合附件 3 中给出的 2019 年该银行客户流失率数据，以确定特定贷款利率和信誉评级下的客户流失率p_i。基于模型求解的需要，使用线性模型对数据进行拟合，结果如下：

$$p_i = \begin{cases} 7.5240 \times r_i - 0.0979, & l_i = A \\ 7.3511 \times r_i - 0.1178, & l_i = B \\ 7.4684 \times r_i - 0.1379, & l_i = C \end{cases}$$

设ref_i为企业i的客户流失系数，若客户i流失，则银行不能对其进行贷款，贷款金额系数m_c_i应为 0，因此该情况下ref_i为 0，若客户未流失则ref_i为 1。

基于以上三方面因素的分析，最终确定贷款额决策模型如下：

$$\begin{aligned} m\_x_i(\boldsymbol{r}) &= (\bar{T}_{Ei} - \bar{T}_{Ii}) \times LR_i \times ref_i(\boldsymbol{r}) \\ &= (\bar{T}_{Ei} - \bar{T}_{Ii}) \times L_i \times DC_i \times ref_i(\boldsymbol{r}) \end{aligned}$$

其中，m_x_i为企业贷款额相对系数，其数值表示获得贷款额的相对大小，$m_x_i = 0$表示银行不对该企业提供贷款。由于此模型带有未求解的贷款利率，因此具体贷款数额将在利率$\boldsymbol{r}$求解出后再进行计算。

5.1.1.2 信贷违约概率模型

根据巴塞尔委员会的定义，信用风险为借款人因各种原因未能及时、足额偿还银行贷款而违约的可能性。若银行因追求利润目标而不能合理控制其贷款业务的信用风险，将面临投资收益率过低、现金流短缺、资产负债无法合理匹配甚至倒闭的风险。

因此，基于对风险控制的要求，企业信贷违约的可能性将是影响银行最终盈利的重要因素，银行的信贷决策模型中必须加入企业信贷违约率。

下面我们引入信用风险死亡率模型来估计不同信用等级与借款人违约概率之间的关系。该模型参考寿险精算中的死亡率模型，以死亡率类比借贷客户的违约率，通过附件 1 中提供的企业信用等级及其违约情况的信息，我们绘制了客户违约率与信用等级的散点图，根据两者明显的非线性关系，初步假定客户违约率关于信用等级的解析分布函数构成指数死亡率模型：

$$\mu_{li} = ae^{bn_l} \cdot \varepsilon_i (a > 0)$$

其中，μ_{li}为信用等级l对应的企业i的贷款违约率，a,b为参数，ε为随机误差项。用对附件 1 中各信誉评级的企业贷款违约率进行拟合，该拟合结果较为理想：

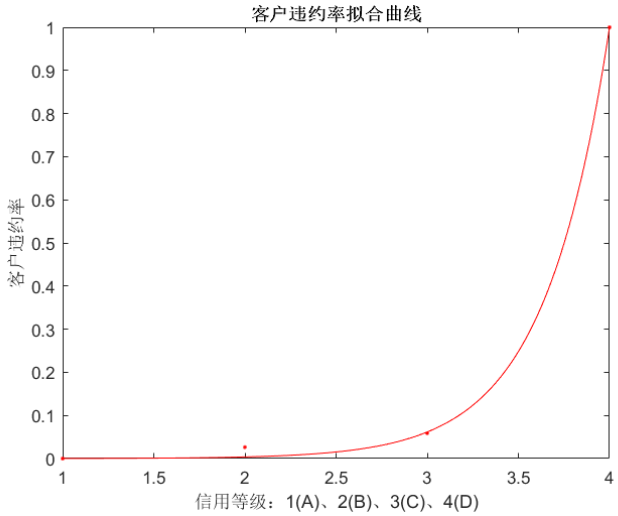

图 2 客户违约率拟合曲线

$$\mu_{li} = 0.000014 \times e^{2.7871 \times n_l} \cdot \varepsilon_i (a > 0)$$

$$\begin{cases} n_l = 1, l = A \\ n_l = 2, l = B \\ n_l = 3, l = C \\ n_l = 4, l = D \end{cases}$$

若企业违约，银行将面临该笔贷款全部本息的损失，且由于中小微企业缺少抵押资产，一旦违约，该笔贷款返还本息的概率非常小，因此假设违约返还率为 0。假设D_i为企业i取得贷款后是否发生违约的预期结果，若企业i按时归还本息则D_i为 1，若无法偿还该笔贷款的全部本息，则违约系数D_i为-1。

5.1.2 银行信贷利率决策模型的建立

通过对各影响因素及其影响机制的分析，可建立以下优化模型对银行信贷决策过程进行描述。

商业银行是以盈利为目的的金融企业，因此可以认为利润最大化是银行开展贷款业务的主要目的。将银行发放该批贷款的税前总现金流入净额（不包含员工薪酬、租金等费用）作为目标函数，表示为：

$$\max NP(\boldsymbol{r}) = E[P(\boldsymbol{r})]$$

$$E[P(\boldsymbol{r})] = \sum_{i=1}^{n} m\_x_i \times (1 + r_i) \times D_i$$

该目标函数的经济含义为：在固定信贷总额下，银行可以通过贷款额的分配以及贷款利率的调整，使期末可收回本息和与违约成本之差达到最大。其中，P为该批贷款期望取得的总现金流入净额；m_i为银行提供给企业i的贷款金额；r_i为银行提供给企业i的贷款利率，且只存在三个可能取值r_a, r_b, r_c；D_i表示企业i是否违约，n为发放该批贷款的企业总数。

基于上述分析，该目标函数中，可由已知信息得出m_i, D_i，而在客户信用等级已知的情况下$ref(r_i, L)$仅依赖于r_i，故通过该函数确定银行的信贷策略过程可简单表述为：在上述已知条件的基础上计算$\boldsymbol{r} = (r_a, r_b, r_c)$的取值，使$NP(\boldsymbol{r})$取得最大值。

根据题目要求，给出如下约束条件：

$$\begin{cases} r_l \in (0.04, 0.15), l = a, b, c \\ r_a \le r_b \le r_c \end{cases}$$

上述两个约束条件表示贷款利率在$4\% - 15\%$之间，且信誉评级高的客户享有更低的贷款利率。

下一步我们将确定在总贷款数额固定的情况下各企业的贷款金额表达式。

根据上述最优化模型结果$\boldsymbol{r}$确定企业i的客户流失系数$ref_i(\boldsymbol{r})$，若客户i流失，则贷款金额系数m_x_i取 0。

将全部非零的m_x_i标准化至$(0,1)$区间，记为m_c_i，表示银行提供给企业i的贷款金额系数，其含义为企业i获得的贷款额占总贷款额的相对比例，$m_c_i = 1$表示该企业获得银行提供给单个企业的贷款额上限，$m_c_i = 0$表示该企业获得银行提供给单个企业的贷款额下限。根据各企业的贷款金额相对比例系数，线性地将企业的贷款数额在 10~100 万元之间均匀分配，因此将企业的贷款金额系数m_c_i通过函数$f(m_c_i)$变换至 10~100 万元区间内（保证$f(m_c_i)$在$(0,1)$区间内为严格单调递增函数），建立模型满足约束条件：

$$s.t.\begin{cases}\sum_{i=1}^{n} m_i = \sum_{i=1}^{n} f(m\_c_i) = K \\ f(0) \geq l_m \\ f(1) \leq u_m \\ f(m\_c_i) = a + b \times m\_c_i \\ b > 0\end{cases}$$

其中，l_m为单个企业获得贷款数额的下限（在此题中为 10 万元），u_m为单个企业获得贷款数额的上限（在此题中为 100 万元），n为确定发放贷款的企业总数。通过以上约束条件可求解该函数的各项系数。因此，企业i最终将获得的贷款金额为m_i=$f(m_{c_i}) = a + b \times m\_c_i$。

5.1.3 模型的求解与分析

利用 Cplex 12.9 软件求解以上最优化模型的精确解，得到银行的信贷利率决策结果如下：

信誉评级	信贷利率（年化）
A	4.8615%
B	4.8615%
C	14.2510%

出于信贷风险控制的考虑，银行需对不同信誉评级的企业实行差别利率，对信誉高、信贷风险小的企业给予利率优惠，对信贷风险高的企业提高利率，即设置约束条件$r_a \leq r_b \leq r_c$。

在未设置约束条件$r_a \leq r_b \leq r_c$的情况下，求得最优解如下：

信誉评级	信贷利率（年化）
A	5.1099%
B	4.1528%
C	14.2510%

根据求解结果，发现所求利率并非无约束条件下的最优解，说明在 A 类企业贷款需求较大且客户流失率对利率变动的敏感度较小的情况下，提高该类企业的利率能使银行的贷款利润进一步扩大。在此情况下，银行牺牲部分利润，对信贷

风险高的企业提高利率，能够体现银行业强烈的风险管理意识，同时对信誉好的企业给予利率优惠，也有利于获取更多优质客户，进一步降低信贷风险，有利于银行的长远利益，因此该约束条件的设置较为合理。

我们也求得了各家企业的贷款金额系数m_c_i，经前述分析，可建立线性函数将该系数映射到 10~100 万元的单笔贷款限额区间内，使其总金额相加等于银行年均贷款限额K，且该映射函数的各项系数随K的变化而变化，由于此问中未给出K的具体数值，暂不对映射函数的各项系数进行计算。

5.2 问题二的模型建立与求解

本问要求对无信贷记录的 302 家中小微企业的信贷风险进行量化分析，并给出在信贷总额为 1 亿元时银行的信贷策略。

5.2.1 无信贷记录企业的信誉评级模型

5.2.1.1 信誉评级影响因素分析

附件 2 中的企业缺少信誉评级信息，无法直接利用第一问的决策模型，因此我们首先需要借助企业数据信息对 302 家目标企业进行信誉评级。

根据相关的信用理论，我们筛选出企业经营规模、供求关系稳定性、企业经营发展能力和对上下游企业的影响力，作为对企业信誉评级过程中考虑的主要因素。由此，我们对附件 2 中提供的进销项发票金额数据、企业名称信息进行财务分析处理，挖掘其中含有的价值信息，最终选取企业年均交易额作为衡量企业经营规模的变量，用有效发票比率（有效发票数与总发票数额之比）、企业与上下游交易最高频率的企业的资金流出与流入量分别占总流出与流入量的比例，作为衡量供求关系稳定性以及对上下游企业的影响力的代理变量，用交易额年增长率作为衡量企业经营发展能力的变量，建立如下的企业信誉评级模型：

$$S_i = w_1 \cdot \overline{T_i} + w_2 \cdot Br_i + w_3 \cdot F_i + w_4 \cdot G_i$$

其中，$\overline{T_i}$为企业年均交易额（去除不完整年份数据），Br_i为有效发票比率，F_i为该企业与上下游高频交易企业的交易额占比，G_i为交易额年增长率（去除不

完整年份数据）。

我们采用熵权法和层次分析法分析上述四个变量的重要程度，然后根据所得的各因素权重值，利用 TOPSIS 综合评价法对企业进行综合信用评价。

5.2.1.2 熵权法与层次分析法确定各因素权重

我们认为企业经营规模、供求关系的稳定、持续经营发展能力和对上下游企业的影响力四类因素对企业信誉评级的影响力度存在差异，故综合采用客观的熵权法及较主观的层次分析法给四个变量赋予客观权重，确定用于 TOPSIS 综合评价的权重系数。

熵权法利用了信息论原理，其基本假设为数据的变异程度越大，该指标蕴含的信息量越大，被赋予的权重就越大。

熵值计算方法如下：

$$e = -\frac{1}{\ln n} \times \sum_{i=1}^{n} p_i \times \ln p_i$$

其中，p_i为相应值在该组变量中所占比重，n为样本个数。信息的效用值即为$1 - e$，再对其归一化处理得到四个因子的权重。

利用层次分析法，根据相关文献及主观经验设置判断矩阵，对根据排序进行一致性检验并计算各指标权重。

综合以上两种方法，计算四种指标（每种指标分为进项和销项两类）的权重，所得结果为：

$$\boldsymbol{w} = 0.5 \times \boldsymbol{w}_{EWM} + 0.5 \times \boldsymbol{w}_{AHP}$$

	$\overline{T_i}$	Br_i	G_i	F_i
进项	0.2091	0.0950	0.1121	0.1150
销项	0.2016	0.0950	0.1143	0.0579

5.2.1.3 TOPSIS 法综合评价企业信用

TOPSIS 法在确定最优解的基础上，计算每个样本到最优解、最劣解的距离，衡量各评价对象与最优方案的贴近程度，进而利用上述指标权重值对评价对象计

算综合得分并进行排序，给出评价结果。

模型中我们采用的指标均与信用水平呈正相关，故信用评价得分高的企业相应的信誉更好。根据评价得分，我们参考第一问中各信誉水平的企业数量和所得的评分情况，对 302 家企业进行信用等级划分，确认信誉评级为 A、B、C、D 分别为 66、94、83、59 家企业，具体结果如下：

企业编号	信用评分	信用评级
E125	0.5265	A
E124	0.4789	A
E168	0.3712	A
…	…	…
E312	0.1956	B
E143	0.1954	B
…	…	…
E244	0.1890	C
E268	0.1890	C
…	…	…
E187	0.0520	D
E217	0.0463	D
E264	0.0235	D

5.2.2 对无信贷记录企业的信贷策略

通过上述方法得到无信贷记录企业的信誉评级，下面即可将包含信誉评级的企业各项数据代入第一问所建立的信贷决策模型中（这一问同样取$DC_i = 1$，假设银行对不同行业依然没有差别的优惠政策，同理，其他系数也假设不变）：

$$m\_x_i(\boldsymbol{r}) = (\bar{T}_{Ei} - \bar{T}_{Ii}) \times L_i \times DC_i \times ref_i(\boldsymbol{r})$$

$$\max NP(\boldsymbol{r}) = \sum_{i=1}^{n} m\_x_i(\boldsymbol{r}) \times (1 + r_i) \times D_i$$

$$s.t.\begin{cases} r_l \in (0.04,0.15), l = a,b,c \\ r_a \le r_b \le r_c \end{cases}$$

$$s.t.\begin{cases} \sum_{i=1}^{n} m_i = \sum_{i=1}^{n} f(m\_c_i) = K \\ f(0) \ge l_m \\ f(1) \le u_m \\ f(m\_c_i) = a + b \times m\_c_i \\ b > 0 \end{cases}$$

给定年度信贷总额K为 1 亿元，可通过以上模型求解银行该批贷款的分配和定价策略，给附件 2 企业提供的贷款金额和相应的贷款利率。

5.2.3 模型的求解与分析

利用 Cplex 软件求解以上最优化模型的精确解，得到最终银行的信贷决策结果如下：

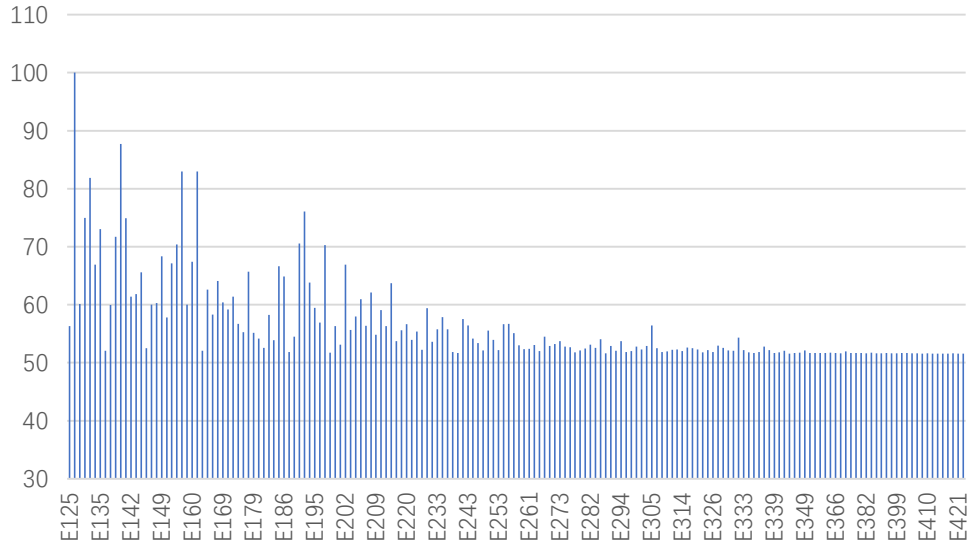

图 3 银行信贷金额分配

结果显示，在银行以利润最大化为目标的信贷决策中，信誉评级为 A、B、C 的企业获得的信贷利率分别为 4.0622%、4.0622%、4.1189%，每个等级的总贷款金额分别为 2688.64 万元、3761.49 万元、3498.30 万元，可见各等级企业所获得的贷款总金额比较接近。

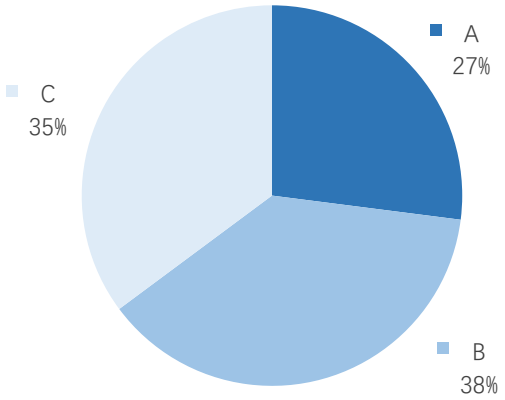

图 4 各信誉等级企业获得贷款总金额

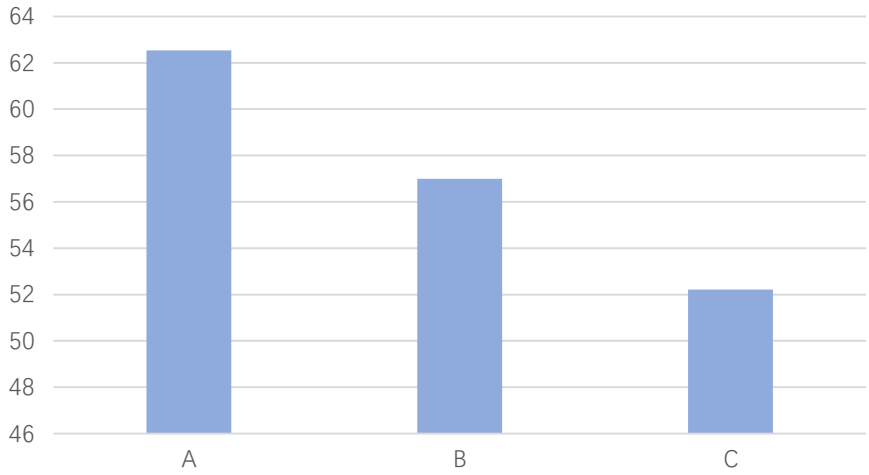

图 5 各信誉等级企业平均获得贷款金额

在最终获得银行信贷的企业中，信誉评级为 A、B、C 的企业数分别为 43 家、66 家、67 家，平均单个企业获得的贷款金额为 62.5 万元、57.0 万元、52.2 万元，其分配较为均衡。

5.3 问题三的模型建立与求解

下面，我们以新冠肺炎疫情为例，分析突发的外生冲击对企业的生产经营和经济效益的影响，进而分析对银行信贷决策的影响。

5.3.1.1.新冠肺炎疫情对企业经营的影响分析

2020 年年初新冠疫情的突然爆发，使国内外物流、人流的跨区域流通受到严格限制，由于各地复工复产的一再推迟，经济出现系统性停滞，中小微企业的经营受到重创，流动性短缺问题十分突出，部分甚至陷入严重的财务危机。

疫情对企业的冲击是多方面的，这里我们仅考虑其中受冲击最为直接且严重的企业主营业务交易额。该影响既广泛遍布于各行业间，同时又因各行业的经营特质而具有一定的差异性。根据 Wind 数据库宏观统计数据显示，疫情期间主营业务受冲击较为严重的行业主要有：旅游食宿服务、运输与储存、制造、金融等行业，旅游食宿服务、运输与储存、制造业受到严重冲击主要是由于这些企业的盈利模式与人员流动率或物流效率高度相关，金融行业受到严重影响则是因为与整体经济情况高度相关，系统性风险 beta 值较高。疫情期间主营业务受冲击较小的行业主要有：公共管理、水电供应、信息通信、建筑业等行业，主要是由于这些企业的盈利模式与人员流动率或物流效率相关性较弱，这些行业中的部分是固定资产占总资产比例较高（经营杠杆高），部分是在疫情爆发过程中承担了一些的社会防疫责任，必须保证正常生产经营。由于根据样本企业所属行业的不同，我们将企业受疫情影响的轻重程度分为严重、较重、一般、轻微 4 类，其交易额减损程度依次递减，具体分类如下表所示：

受疫情影响程度	交易额减损率	所属行业
严重	-60%	食宿服务、运输存储、文娱教育
较重	-40%	制造业、金融、行政辅助
一般	-20%	建筑业、个体经营、农林业
轻微	-10%	公共管理、水电供应、信息通信、批零业

5.3.1.2.信贷政策

根据相关的经济理论，突发的外生冲击，如突发的公共卫生事件、自然灾害等，往往会导致短期的市场失灵，此时及时且恰当的政府调节措施能够通过调整资源的分配来改善该局面，在一定程度上维持企业的正常经营活动，促进经济增

长，避免失业率的突然上升给社会各方面带来更大的冲击。

而根据前文分析，新冠疫情导致大批中小微企业面临经营危机和财务危机，这将直接导致其贷款违约率的提高，商业银行出于对自身利润的考量，往往选择严控风险，减少信贷发放或提高对中小微企业的贷款利率，这将对资金短缺的企业带来第二次冲击。此时信贷政策的出台能够保证受疫情冲击的企业以更加合理的资金成本得到充足的贷款额以渡过危机，减弱该事件对社会稳定的冲击力，加快冲击后经济复苏的速度。

我们将信贷政策处理为影响银行可贷率L的因子DC_i，对银行在疫情期间的信贷资金分配具有强制效力。根据行业重要性和行业受疫情影响的严重程度，信贷政策对银行可贷率的上浮调整力度可分为 4 类，行业分类如下图所示：

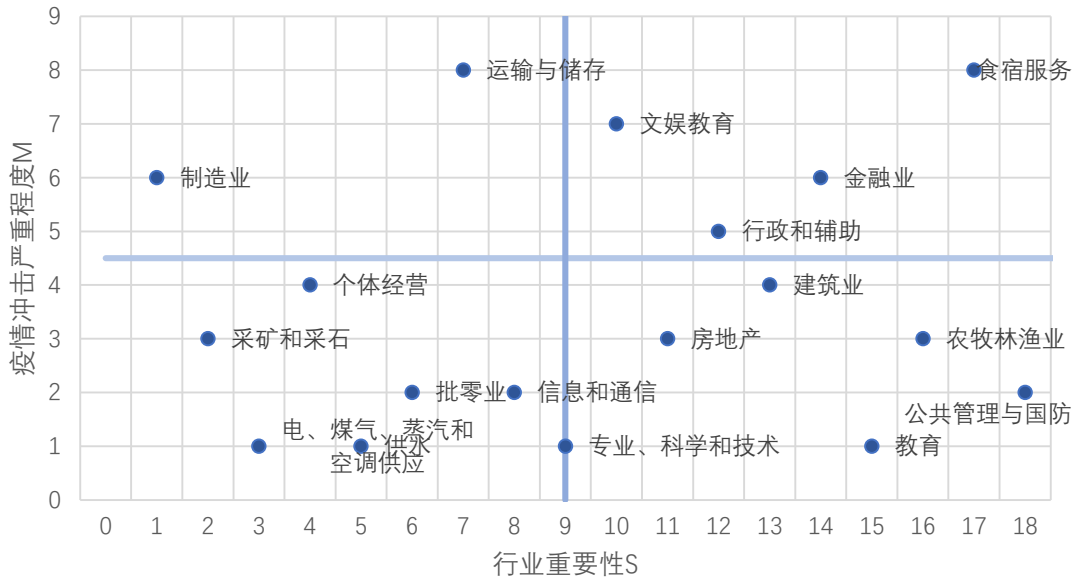

图 6 信贷政策行业分析图

其中，横坐标数值越大，表明该行业重要程度越高，政府对其发展的支撑力度越大，这些行业往往是国家的支柱行业、经济发展引擎，或具有重要的战略发展意义，不能放任这类企业由于受到严重冲击而大规模倒闭；纵坐标数值越大，表明该行业受疫情冲击越大，这是由行业的经营特征和盈利模式决定的。设置信贷政策对可贷率的 4 类调整情况如下：

行业分类	可贷率调整力度
(S_1, M_1)	+50bp

(S_0, M_1)	+30bp
(S_1, M_0)	+20bp
(S_0, M_0)	+10bp

其中：S表示行业重要性，$S_1 > S_0$；M表示受疫情冲击程度，$M_1 > M_0$。

5.3.2.银行信贷策略的调整与灵敏度分析

根据上述突发因素（新冠肺炎疫情）对企业经营及信贷政策的影响，我们将该冲击发生后的信贷决策模型输入变量进行调整，来分析银行信贷策略的变动情况。

无信贷记录企业的受到冲击后，通过上述方法计算影响信誉评级的指标数值变化（仅调整年均交易额）与权重变化，进而得到新的企业信用评分，假设中小微企业受到冲击后的信誉评级普遍下滑，设置信誉等级为 A、B、C、D 的企业数分别变为原来的 0.6、0.8、1.2、1.4，得到新的企业信誉评级。

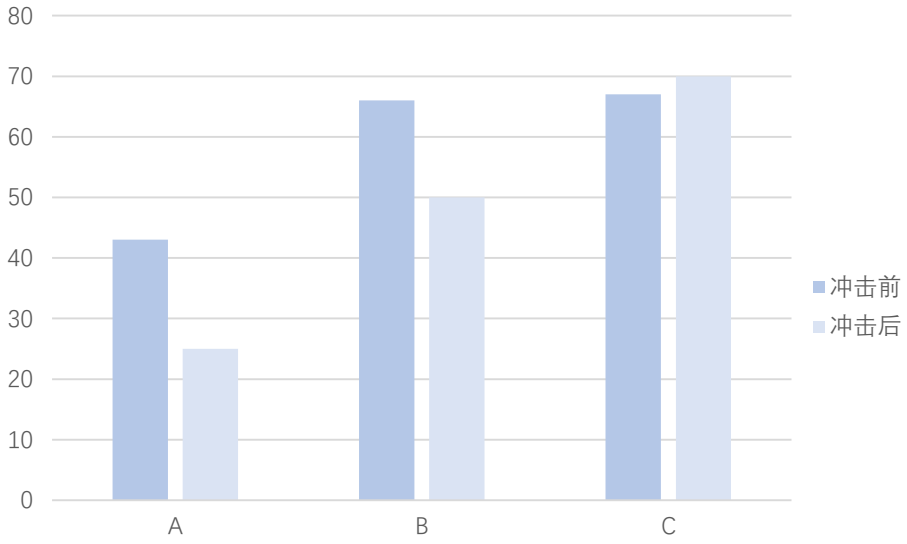

图 7 疫情前后各信誉评级企业数对比图

将包含调整后的信誉评级和交易额的企业各项数据代入第一问所建立的信贷决策模型中：

求解模型，可得到该冲击发生后银行给该批无信贷记录企业发放一年期固定贷款总额（K=1 亿元）的贷款金额分配和利率定价策略。

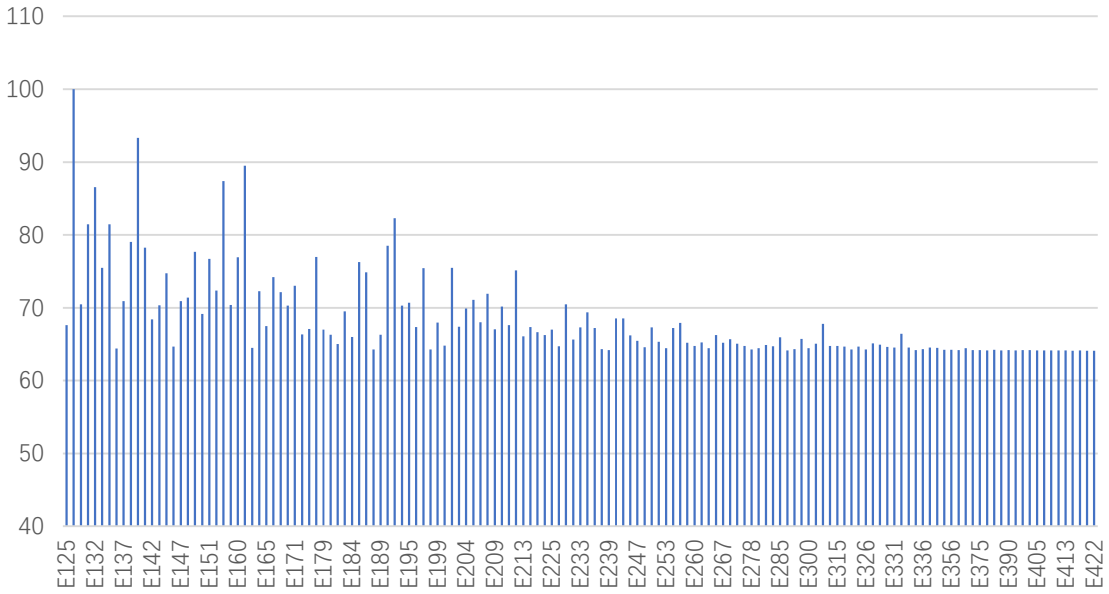

图 8 疫情后银行信贷金额分配

其中，每个等级的总贷款金额分别为 1763.00 万元、3591.18 元、4581.72 万元，占信贷总额的 17.7%，36.1%、46.1%，这是主要是由于各企业的信誉评级水平整体下滑，高信誉企业数量普遍减少所导致的。

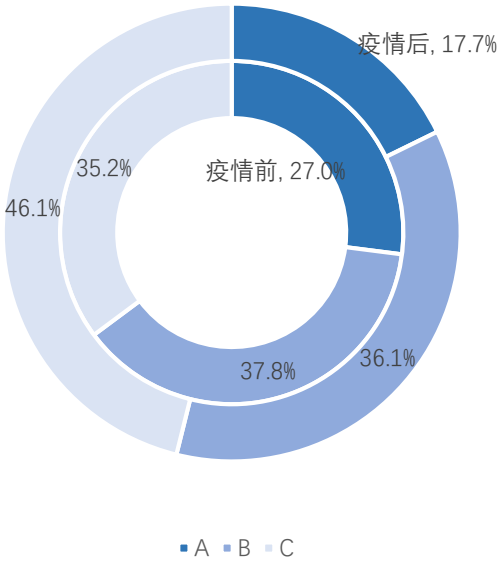

图 9 疫情前后各信誉评级企业贷款总金额占比

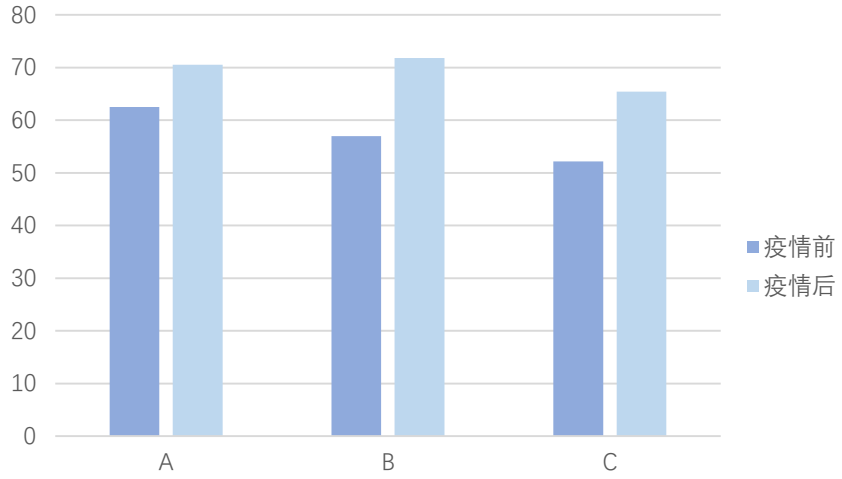

图 10 疫情前后各信誉评级企业平均获得贷款金额

疫情前后各信誉评级企业平均获得贷款金额均有所上升，其中 B、C 等级的企业获得的平均贷款额增加幅度较大。体现了银行政策对信贷资源分配的有效调节。

信誉评级为 A、B、C 的企业获得的信贷利率分别为 4.2628%、4.2628%、4.2681%。

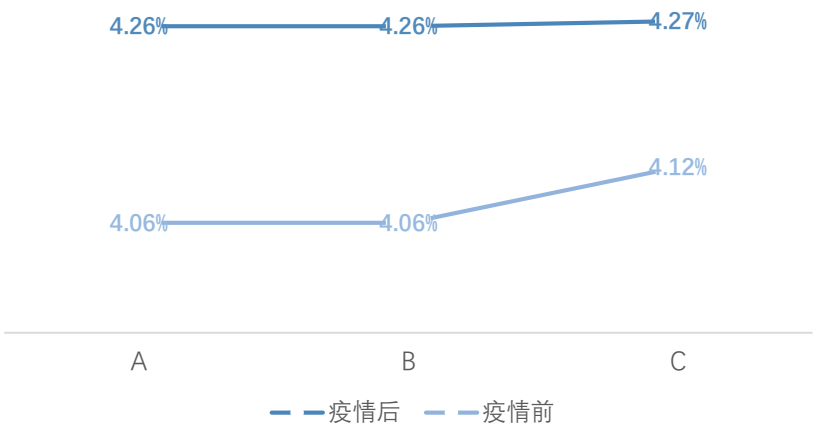

图 11 疫情前后各信誉评级企业贷款利率

与疫情发生前的银行信贷决策相比，虽然各企业的交易额受疫情影响发生不同程度的下降，信誉评级水平普遍下滑，但由于政策的及时调整，我们发现疫情发生后银行对各信誉等级企业的贷款额度分配更加均匀，贷款利率虽有所提高但各等级间的差异明显缩小。

上述对模型的灵敏度分析结果体现了政策对社会资源的再分配作用。在突发因素的影响下，银行对各企业的贷款利率及贷款金额分配策略没有发生异常性的

变化，说明我们建立的银行信贷决策模型具有良好的稳健性。

六、模型的评价与推广

6.1 模型的优点

- 模型的构建比较符合实际情况，综合考虑了多种影响银行信贷决策的因素，对各类影响因素的考虑比较全面，对各因素对信贷决策的影响机制与冲击对各因素的影响分析均有经济理论依据，较为科学合理。
- 采用了较科学的财务分析方法，代理变量的选取较为合理，且均有相关的理论依据。
- 模型框架的适用性较广。信贷金额决策模型的构建不受单笔贷款限额或总贷款额度的限制，通用性强。若数据充足，可进一步增加贷款金额与利率决策模型、信誉评级模型中的变量，而无需修改模型的结构，因此具有较强的实用性。
- 采用了严谨的最优化问题解决算法，有效地避免了启发式算法结果不稳定的问题。

6.2 模型的缺点

- 模型的求解结果比较依赖数据的准确性。
- 考虑到解决问题的简便性，对模型设置了较多假设，如，对中小微企业贷款只有全额收回或全额损失两种情况，与实际情况相比稍显简化。

6.3 模型的改进与推广

- 考虑放松部分假设，如：假设未偿还贷款能够以一定概率收回，且该偿还机制受贷款利率等因素的影响；假设银行对同一信誉评级但不同规模的企业采用不同的贷款利率；假设客户有权根据贷款利率提出理想的贷款金额，在银行决策的过程中考虑该信息等。

- 搜集更多对象的更多数据指标，拓宽变量的选取范围，对企业信誉评级、银行的信贷决策过程的衡量将更加准确。
- 除所属行业外，对企业的特征（如经营规模、资产负债情况）进行进一步分类，将外生冲击对企业经营的影响衡量地更加精确。
- 若拓展贷款的时限，可综合衡量企业的融资需求和经营状况，对贷款时限进行决策。由于在长期中企业的信誉评级、主营业务收入等变量都将变化，因此需要通过更多的数据和较准确的预测模型，对银行信贷未来的收益与损失进行评估。

参考文献

[1]林建华.一种用于确定小微贷款额的盲数模型及其应用[J].海南金融,2013(04):45-49.

[2]周四军,袁鹏,冯岑.商业银行信用风险测度的死亡率模型[J].统计与决策,2009(12):33-35.

[3]朱武祥,张平,李鹏飞,王子阳.疫情冲击下中小微企业困境与政策效率提升——基于两次全国问卷调查的分析[J].管理世界,2020,36(04):13-26.

[4]王伟,宋西圣.以“大数据思维强化小微企业信贷风险管理[J].金融发展研究,2014(04):83-84.

[5]贾生华,史煜筠.商业银行的中小企业信贷风险因素及其管理对策[J].浙江大学学报(人文社会科学版),2003(02):57-64.

[6]孙雪梅.浅析商业银行中小企业信贷风险与管理[J].财经界(学术版),2010(24):27.

[7]尹竹青. 信用度量模型在中小企业信贷风险管理中的应用研究[D].中央财经大学,2008.

[8]糜仲春,申义,张学农.我国商业银行中小企业信贷风险评估体系的构建[J].金融论坛,2007(03):21-25.

附 录

Matlab R2015b 代码：

```
%dieline_fitness.m 客户违约率拟合
close all;
clear all;
clc;
x=[1,2,3,4];
y=[0,1/38,2/34,1];
a=zeros(2,1);
fun=@(a,x)(a(1)*exp(a(2)*x));
fit_a=nlinfit(x,y,fun,[0.005,2.8]);
figure
x_0=[1:0.01:4];
plot(x,y,'r.',x_0,fit_a(1)*exp(fit_a(2)*x_0),'r-');
y_0=fit_a(1)*exp(fit_a(2)*x);
xlabel('信用等级：1(A)、2(B)、3(C)、4(D)');
ylabel('客户违约率');
title('客户违约率拟合曲线');
```

```
%fitness.m 客户流失率拟合
close all;
clear all;
clc;
data=xlsread('rate_refuse_data.xlsx');
x=data(:,1);
a=zeros(3,1);
y_0=data(:,2:4);
for i=1:3
    y=y_0(:,i);
   fun=@(a,x)(a(1)*x+a(2));
   fit_a=nlinfit(x,y,fun,[-0.5,-0.1 1]);
   figure
   plot(x,y,'r.',x,fit_a(1)*x+fit_a(2),'r-');
   if(i==1)
       title('level A 客户流失率拟合曲线');
   end
```

```
    if(i==2)
        title('level B 客户流失率拟合曲线');
    end
    if(i==3)
        title('level C 客户流失率拟合曲线');
    end
    total_fit_a(:,i)=fit_a';
end
```

```
%chain_count.m
clear all;
clc;
%销项
data1=xlsread('chain1.xlsx');
total_result_out=zeros(302,3);
en=1;
count_sum=1;
I=0;
n=1;
for i=124:425
    count_num=sum(data1(:,1)==i);
    data_0=data1(count_sum:count_sum+count_num-1,:);
    table=tabulate(data_0(:,2));%计算上下游企业出现的频数和频率
    max_t=max(table(:,2));%求交易频率最高的企业出现的频数
    F=find(table(:,2)==max_t);%求交易频率最高的企业的编号
    I=table(F,1:2);%[编号，出现频数]
    total_result_out(n,1:2)=I(1,:);%记录每个企业
    count_sum=count_sum+count_num;
    n=n+1;
end
for j=1:302
    n=find(data1(:,2)==total_result_out(j,1));%在 data 中寻找该上下游企业位置
    total_result_out(j,3)=sum(data1(n,3));%将其与上下游企业的业务现金流相加
end

%进项
data2=xlsread('chain2.xlsx');
total_result_in=zeros(302,3);
en=1;
count_sum=1;
```

```
I=0;
n=1;
for i=124:425
    count_num=sum(data2(:,1)==i);
    data_0=data2(count_sum:count_sum+count_num-1,:);
    table=tabulate(data_0(:,2));%计算上下游企业出现的频数和频率
    max_t=max(table(:,2));%求交易频率最高的企业出现的频数
    F=find(table(:,2)==max_t);%求交易频率最高的企业的编号
    I=table(F,1:2);%[编号，出现频数]
    total_result_in(n,1:2)=I(1,:);%记录每个企业
    count_sum=count_sum+count_num;
    n=n+1;
end
for j=1:302
    n=find(data2(:,2)==total_result_in(j,1));%在 data 中寻找该上下游企业位置
    total_result_in(j,3)=sum(data2(n,3));%将其与上下游企业的业务现金流相加
end
```

```
%AHP.m
function [weight_ahp]=AHP()
%Q 为权值，B 为对比矩阵
B=[1,2,5,8;1/2,1,3,4;1/5,1/3,1,2;1/8,1/4,1/2,1];%【请修改此处，输入判别矩阵 B】
[rows,cols]=size(B);
%判别矩阵具有完全一致性
for i=1:rows
    for j=1:cols
        if B(i,j)*B(j,i)~=1
          fprintf('i=%d,j=%d,B(i,j)=%d,B(j,i)=%d\n',i,j,B(i,j),B(j,i))
        end
    end
end
%求特征值特征向量,找到最大特征值对应的特征向量
[V,D]=eig(B);
tz=max(D);
tzz=max(tz);%最大特征值
c1=find(D(1,:)==max(tz));%最大特征值所在的位置
tzx=V(:,c1);%对应特征向量
%求权值
quan=zeros(rows,1);
for i=1:rows
```

```
quan(i,1)=tzx(i,1)/sum(tzx);%特征向量标准化
end
Q=quan;%结果 权值向量 Q
%一致性检验
CI=(tzz-rows)/(rows-1);%计算一致性检验指标 CI
RI=[0,0,0.58,0.9,1.12,1.24,1.32,1.41,1.45,1.49,1.52,1.54,1.56,1.58,1.59];% 修 正因子取值（定值）
%判断是否通过一致性检验
CR=CI/RI(1,rows);
if CR>=0.1
  fprintf('对比矩阵 B 没有通过一致性检验\n');
else
  fprintf('对比矩阵 B 通过一致性检验\n');
end
weight_ahp=[Q(1) Q(1) Q(2) Q(2) Q(3) Q(3) Q(4) Q(4)];
%end
```

```
%level_analysis.m 附件 2 企业信用综合评级
%R=xlsread('level_analyse2.xlsx');%第二问数据输入
R=xlsread('level_analyse3.xlsx');%第三问数据输入
[rows,cols]=size(R);   % rows 为对象个数，cols 为指标个数
k=1/log(rows);         % 求 k
Rmin = min(R);
Rmax = max(R);
A = max(R) - min(R);
y = R - repmat(Rmin,rows,1);
%y(i,j) = (R - repmat(Rmin,rows,1))/(repmat(A,rows,1));
for j = 1 : size(y,2)
     y(:,j) = y(:,j)/A(j);
end%标准化为 0-1 的数
%求 Y（i,j）
S = sum(y,1);
Y = zeros(rows,cols);
for i = 1 : size(Y,2)
    Y(:,i) = y(:,i)/S(i);
end%定义标准化
%计算指标信息熵值 e 和信息效用值 d
lnYij=zeros(rows,cols);
for i=1:rows
    for j=1:cols
```

```
        if Y(i,j)==0
            lnYij(i,j)=0;
        else
            lnYij(i,j)=log(Y(i,j));
        end
    end
end
ej=-k*(sum(Y.*lnYij,1)); %计算熵值 Hj
weights=(1-ej)/(cols-sum(ej));%熵权法评价指标的权重 weights
F = zeros(rows,cols);%综合评价矩阵 F

%AHP
weight_ahp=AHP();
weights=0.5*weights+0.5*weight_ahp;%熵权法+层次分析法综合权重 weights
weights=weights./sum(weights);

%TOPSIS
[rows,cols]=size(R);
c=sqrt(sum(R.*R));
for i=1:rows
  for j=1:cols
    d(i,j)=R(i,j)/c(j);% d 为规范化决策矩阵
  end
end

for i=1:rows
  for j=1:cols
    c(i,j)=d(i,j)*weights(j);% c 为加权规范矩阵,weights 为权重向量
  end
end
cmax=max(c);
cmin=min(c);
for i=1:rows
  c1=c(i,:)-cmax;
  s1(i)=norm(c1);%各方案到正理想解的距离
  c2=c(i,:)-cmin;
  s2(i)=norm(c2);%各方案到负理想解的距离
  T(i)=s2(i)/(s1(i)+s2(i));%计算各方案的综合评价指数
end
T=T';
disp('各企业的综合评价指数 T 为：')
T
```

Cplex 12.9 **代码：**

源程序 1：

```
/*********************************************
 * OPL 12.9.0.0 Model
 * Author: asus
 * Creation Date: 12 Sep 2020 at 22:45:47
 *********************************************/
 //第一问 银行信贷决策模型--非线性规划求最优解
 //企业信誉评级信息
int n=123;
float
account[1..n]=[0,140069703.3,98375596.87,315328700.4,1229090.504,23973283.4
8,177821167,66742001.62,106749832.2,87226437.07,3311860.025,31999926.29,468
43852.2,19442821.19,62050554.5,67189214.43,7389070.76,19739350.04,0,4162021
.312,1768418.827,16207956.04,2483695.992,20991732.6,5467557.398,48365.74667
,0,12129434.95,7798293.39,14371633.45,16033022.04,11448714.02,1200265.808,1
1393771.1,1495186.973,0,0,9853564.005,5424039.37,4749977.757,4208545.712,77
38230.537,4218377.971,530027.77,268686.5653,1548248.738,1482042.83,13793538
.86,2594082.255,3041747.07,2094482.773,0,530668.1933,9226092.11,2613263.465
,8166.643333,1322470.483,4528404.072,3141728.033,588164.3813,4672377.659,25
05411.875,4144643.733,2929935.993,788058.7387,0,1205056.68,992723.0333,5593
26.605,1183189.312,592647.928,436163.1233,1045272.607,887669.5333,650890.91
83,897372.8507,633323.665,1048349.747,33706.09867,147034.82,542522.8767,0,0
,1313869.683,535545.0613,92951.295,414093.12,584200.93,0,424125.3717,390092
.6167,316930.0983,312206.4453,178074.9817,632420.3867,0,245060.3733,320615.
4187,0,0,0,0,0,45041.66667,147261.9817,139444.4693,0,0,0,33606.25167,0,0,0,
0,0,0,0,0,0,0,0,0,0];
int
level_A[1..n]=[1,1,0,0,0,1,1,1,1,0,0,0,1,0,1,1,1,1,1,0,0,1,0,1,0,1,1,0,0,0,
1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
int
level_B[1..n]=[0,0,0,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,1,0,1,
0,1,1,1,1,0,1,1,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1,1,0,1,1,1,1,0,1,1,1,0
,0,1,1,0,0,1,0,1,0,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,0,1,1,0,0,0,0,0,0,0,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
int
level_C[1..n]=[0,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,
0,0,0,0,0,0,0,0,1,1,1,0,0,1,0,1,1,0,1,1,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,1
,1,0,0,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1,1,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,1,1,
```

```
0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0];
    int
level_D[1..n]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,
0,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1];
    int d[1..n]=[1,1,1,1,1,1,1,1,1,-1,-
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-1,1,1,1,1,1,1,-1,1,1,-1,1,1,1,1,1,1,-
1,1,1,-1,1,1,-1,1,1,1,-1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-
1,1,1,1,1,1,1,1,-1,1,1,1,1,1,1,1,1,1,1,1,-1,1,1,-1,1,1,-1,-1,-1,-1,-1,1,1,1,-
1,-1,-1,1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1];
    float
loss_rand[1..n]=[0.155705435,0.389965514,0.959959716,0.927426984,0.00601214
6,0.287545396,0.009430219,0.288460952,0.18530839,0.589403974,0.70393994,0.0
47242653,0.454176458,0.930112613,0.890316477,0.911801508,0.443159276,0.6549
8825,0.321909238,0.753318888,0.644337291,0.469862972,0.834986419,0.70229194
,0.24897,0.553331095,0.136387219,0.410016175,0.487014374,0.1633961,0.108890
042,0.758079775,0.624713889,0.269417402,0.040314951,0.795556505,0.276741844
,0.240607929,0.682729575,0.186712241,0.189123203,0.122684408,0.547929319,0.
949369793,0.148930326,0.533463546,0.34562212,0.765343181,0.17731254,0.70870
0827,0.970336009,0.644947661,0.661976989,0.730033265,0.943449202,0.22571489
6,0.585253456,0.465285195,0.716513565,0.48750267,0.504257332,0.253364666,0.
74221015,0.164555803,0.439252907,0.864192633,0.718985565,0.636494034,0.3262
73385,0.765465255,0.916135136,0.578600421,0.190649129,0.227301859,0.9846186
71,0.153416547,0.697500534,0.29606006,0.161076693,0.024658956,0.235175634,0
.168248543,0.270332957,0.974364452,0.188512833,0.996642964,0.432905057,0.40
5011139,0.557512131,0.658619953,0.884090701,0.507705924,0.578112125,0.51924
192,0.810480056,0.37736137,0.500991852,0.719382305,0.610400708,0.856776635,
0.185125278,0.02047792,0.484206671,0.390087588,0.661824396,0.737479781,0.76
1314737,0.667531358,0.325693533,0.790398877,0.869228187,0.342356639,0.90896
3286,0.406964324,0.352061525,0.601184118,0.895138401,0.506302072,0.84365367
6,0.869289224,0.528366955,0.295693838,0.402050844];
    dvar float+ rate[1..n];
    dvar float+ ra;
    dvar float+ rb;
    dvar float+ rc;
    dvar float+ p_loss[1..n];
    dvar boolean refuse[1..n];

    //定义目标函数
    maximize
      sum (i in 1..n)
account[i]*(1+(level_A[i]+level_B[i]+level_C[i])*rate[i]+level_D[i]*(-
```

```
1))*d[i]*refuse[i];

    //输入约束条件
    subject to {
    forall (i in 1..n) (level_A[i]==1)=>rate[i]==ra;
    forall (i in 1..n) (level_B[i]==1)=>rate[i]==rb;
    forall (i in 1..n) (level_C[i]==1)=>rate[i]==rc;

    forall (i in 1..n)
(level_A[i]==1)=>(p_loss[i]==7.52406404059649*rate[i]-0.0979300813722221);
    forall (i in 1..n)
(level_B[i]==1)=>(p_loss[i]==7.35107535431523*rate[i]-0.117765769021589);
    forall (i in 1..n)
(level_C[i]==1)=>(p_loss[i]==7.46841647701602*rate[i]-0.137910844288652);

    forall (i in 1..n)
(level_A[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;
    forall (i in 1..n)
(level_B[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;
    forall (i in 1..n)
(level_C[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;

    ra>=0.04&&ra<=0.15;
    rb>=0.04&&rb<=0.15;
    rc>=0.04&&rc<=0.15;

    ra<=rb;
    rb<=rc;
    }
```

源程序 2：

```
    /*********************************************
     * OPL 12.9.0.0 Model
     * Author: asus
     * Creation Date: 13 Sep 2020 at 08:44:38
     *********************************************/

     //第二问 银行信贷决策模型--信贷利率
     //企业评级信息
    int n=302;
```

```
    float
account[1..n]=[0,7879767.117,136396217.6,223247538.9,80442906.28,96273755.1
6,14245815.34,38885642.38,50284782.31,25468544.27,31801113.86,35690174.99,8
18405.25,13951398.88,33422096.81,65271432.2,60052462.54,38736833.28,1635229
5.48,17021930.56,23320369.6,1450638.49,1604636.883,14000828.57,14525569.34,
27897238.11,10413386.06,25875199.11,31245899.55,84858008.43,0,1606218.39,52
138174.13,4720648.579,6633885.873,14033878.47,26318012.44,52121909.95,81007
3.29,19256041.62,18323341.41,11153086.69,20795322.87,0,30489629.23,14670067
.94,12693497.56,16296627.03,8537725.752,0,6120255.513,0,23504057.16,9503416
.033,0,5925160.848,4350644.63,0,1637587.077,11109941.33,3887400.229,2501710
2.3,22097450.87,0,495065.3033,4905446.955,31536669.77,0,5876246.523,4074899
7.34,20340677.7,13084660.35,8861147.211,31058379.32,324662.2587,7936946.215
,0,2608795.928,25451918.11,6749138.893,10616891.33,15603972.45,7984733.193,
17485520.14,0,5382111.992,12446446.62,7849881.365,20161391.62,3577955.397,8
031455.059,18373895.03,6691235.179,0,0,124547.9307,8416333.867,3913675.928,
3420979.669,0,0,6319317.837,0,1134982.107,7930063.488,0,13049437.42,0,33791
14.835,6996946.157,10461949,7986983.008,7005342.378,494968.135,11399633.07,
218325.8533,0,9887755.035,0,8105034.541,3888210.562,4296306.593,4210667.148
,3024721.395,903085.74,6608996.704,4478396.105,0,3978249.656,1002582.05,847
0110.125,8510194.896,5850774.488,2422620.592,14703905.32,3571177.472,128079
2.656,1394448.315,2508826.512,2186071.908,0,721615.6133,4854281.17,2225905.
945,3497350.82,5565005.212,2728258.742,0,0,3554093.96,2008152.253,0,0,18205
39.267,334003.2767,948757.328,1501372.421,0,2543191.341,1313996.355,1649945
.253,4167835.368,0,126910.8483,564423.7333,2292446.625,2207256.452,0,826501
.2317,37096.19667,3603823.405,499325.1533,0,583324.1667,1647324.085,0,75377
8.7493,2073534.64,1187975.848,2238487.093,0,8089642.717,1574594.043,463275.
1467,642015.0067,1107771.55,1205143.243,0,0,978688.155,763190.9433,1756329.
205,1598655.655,0,0,1220150.803,381774.765,1041861.977,0,0,0,0,451592.0733,
2275231.763,0,1682676.517,917589.835,855594.055,4596770.077,976557.455,3797
76.6567,200486.5233,432416.0183,2058233.357,1017202.707,190892.6933,0,37359
8.46,0,855974.9333,0,0,20983.65333,219339.5233,246937.8767,912399.9867,1730
98.4017,0,0,0,0,213558.6033,240559.475,200709.9233,0,322060.8053,0,311633.9
,0,0,310339.9033,0,186945.115,0,0,148952.7867,654052.35,190389.8183,0,0,201
421.0133,167525.75,0,0,0,0,0,0,87087.83333,0,0,0,0,282840.3333,0,78764.5,13
5104.47,211243.5333,0,0,0,0,0,0,92706.42667,144817.8667,205070.4133,0,0,187
415.0333,0,86306.14667,77068.38933,0,0,57321.02333,104233.1867,0,43364.1066
7,40923.335,0,22645.06667,27717.088,0,65271.29,0,0,36816.97167,4239.435,0,0
,0];
    int
level_A[1..n]=[1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,0,1,1,1,0,0,1,0,0,1,
1,1,1,0,1,1,1,1,1,1,1,1,0,1,1,1,0,0,0,1,1,1,0,1,1,0,1,1,1,0,0,0,0,0,1,0,0,1
,0,1,0,1,0,1,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,
0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,
0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0
,0,0,0,1,0,0,0,0,0];
    int
level_B[1..n]=[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,1,1,0,
0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,0,0,0,1,0,0,1,0,0,0,1,1,1,1,0,0,1,1,0
,1,0,1,0,1,0,1,0,1,1,1,1,0,1,0,0,0,1,1,1,1,1,1,1,0,1,1,0,1,1,1,0,1,1,1,1,
1,1,1,1,0,1,1,0,0,1,1,0,1,0,1,0,0,0,1,0,1,0,0,1,0,1,1,1,1,1,1,1,0,1,0,0,1,0
,0,0,0,0,0,0,1,0,0,0,1,0,1,1,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,
1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,1
,0,0,0,0,0,0,0,0,0];
    int
level_C[1..n]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1,1,0,1,0,1,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,1
,1,1,1,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,1,1,1,0,1,1,1,1,0,1,1,0,0,1,1,1,
0,0,1,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,0,0,0,0,1,0,0,1,1,1,0,1,1,1,1,1,1,1,0,1
,0,0,0,0,1,1,1,0,1,0,1,0,0,0,1,1,0,0,0,1,0,0,1,0,1,0,0,1,0,1,0,0,1,1,0,0,0,
0,0,0,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,0,0,0,0,0,1,1,0,0,0
,1,1,1,0,1,1,0,0,0];
    int
level_D[1..n]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0
,0,0,0,0,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0
,1,0,1,1,0,0,0,0,0,1,0,1,1,0,0,0,1,0,1,0,1,1,0,1,0,1,1,0,0,0,1,0,0,0,1,1,1,
1,1,0,0,1,1,1,1,0,1,0,0,0,1,1,1,1,1,1,0,0,0,1,1,0,1,0,0,1,1,0,0,1,0,0,1,0,0
,0,0,0,0,0,0,1,1,1];
    int
d[1..n]=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-1,1,-1,1,1,1,1,1,1,1,1,-
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-1,1,1,1,-1,1,1,-1,-1,1,1,-1,-
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-
1,1,1,1,1,1,1,-1,-1,1,1,1,-1,-1,1,1,1,1,-1,1,1,1,1,1,1,1,1,1,1,1,1,-1,1,-1,1,-
```

```
1,-1,1,1,1,1,1,-1,1,-1,-1,1,1,1,-1,1,-1,1,-1,-1,1,-1,1,-1,-1,1,1,1,-
1,1,1,1,-1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,1,-1,1,1,1,1,-1,-1,-1,-1,-1,-1,1,1,1,-
1,-1,1,-1,1,1,-1,-1,1,1,-1,1,1,-1,1,1,1,1,1,1,1,1,-1,-1,-1];
    float
loss_rand[1..n]=[0.814723686,0.905791937,0.126986816,0.000091337,0.63235924
6,0.097540405,0.278498219,0.546881519,0.957506835,0.964888535,0.157613082,0
.970592782,0.957166948,0.485375649,0.800280469,0.141886339,0.421761283,0.91
5735525,0.79220733,0.959492426,0.655740699,0.035711679,0.849129306,0.933993
248,0.678735155,0.757740131,0.743132468,0.39222702,0.65547789,0.171186688,0
.706046088,0.031832846,0.276922985,0.046171391,0.097131781,0.823457828,0.69
4828623,0.31709948,0.950222049,0.034446081,0.43874436,0.381558457,0.7655167
88,0.795199901,0.186872605,0.489764396,0.445586201,0.64631301,0.709364831,0
.754686682,0.276025077,0.679702677,0.655098004,0.162611735,0.118997682,0.49
8364052,0.959743959,0.340385727,0.585267751,0.223811939,0.751267059,0.25509
5115,0.505957052,0.699076723,0.890903253,0.959291425,0.54721553,0.138624443
,0.149294006,0.257508254,0.840717256,0.254282179,0.814284826,0.243524969,0.
929263623,0.349983766,0.19659525,0.251083858,0.616044676,0.473288849,0.3516
59507,0.830828628,0.585264091,0.549723608,0.917193664,0.285839019,0.7572002
29,0.753729094,0.380445847,0.567821641,0.07585429,0.053950119,0.530797553,0
.77916723,0.934010684,0.129906208,0.568823661,0.469390641,0.01190207,0.3371
22644,0.162182308,0.794284541,0.311215042,0.528533136,0.165648729,0.6019819
41,0.262971285,0.654079098,0.689214503,0.748151593,0.450541599,0.083821378,
0.228976969,0.913337362,0.152378019,0.825816977,0.538342435,0.996134717,0.0
78175529,0.44267827,0.10665277,0.961898081,0.004634224,0.774910465,0.817303
221,0.868694705,0.084435846,0.399782649,0.259870403,0.80006848,0.431413827,
0.910647594,0.181847028,0.263802917,0.14553898,0.136068559,0.869292208,0.57
9704587,0.549860202,0.144954798,0.853031118,0.622055131,0.350952381,0.51324
954,0.401808034,0.075966692,0.239916154,0.123318935,0.183907788,0.239952526
,0.417267069,0.04965443,0.90271611,0.94478719,0.490864092,0.489252638,0.337
71941,0.900053846,0.369246781,0.111202755,0.780252068,0.389738837,0.2416912
86,0.403912146,0.096454525,0.131973293,0.942050591,0.95613454,0.575208595,0
.059779543,0.234779913,0.353158571,0.82119404,0.015403438,0.043023802,0.168
990029,0.649115475,0.731722386,0.647745963,0.450923706,0.547008892,0.296320
806,0.744692807,0.188955015,0.686775433,0.183511156,0.368484596,0.625618561
,0.780227435,0.081125769,0.929385971,0.775712679,0.486791632,0.435858589,0.
446783749,0.306349472,0.508508655,0.510771564,0.817627708,0.794831417,0.644
31813,0.378609383,0.811580458,0.532825589,0.350727104,0.939001562,0.8759428
11,0.550156343,0.622475086,0.587044705,0.207742293,0.30124633,0.470923349,0
.23048816,0.844308793,0.19476429,0.225921781,0.170708047,0.227664298,0.4356
98684,0.311102287,0.923379642,0.430207391,0.18481632,0.904880969,0.97974837
8,0.438869973,0.111119223,0.258064696,0.408719846,0.594896074,0.262211748,0
.602843089,0.71121578,0.221746734,0.117417651,0.296675873,0.318778302,0.424
16676,0.507858285,0.085515797,0.262482235,0.801014623,0.029220278,0.9288541
```

```
39,0.730330863,0.488608974,0.578525061,0.23728358,0.458848828,0.963088539,0
.546805719,0.521135831,0.231594387,0.488897744,0.624060088,0.679135541,0.39
5515216,0.367436649,0.987982003,0.037738866,0.885168008,0.913286828,0.79618
3874,0.098712279,0.261871184,0.33535684,0.679727951,0.136553137,0.721227499
,0.106761862,0.653757349,0.494173937,0.779051723,0.715037078,0.903720561,0.
890922504,0.334163053,0.698745832,0.197809827,0.030540946,0.74407426,0.5000
22436,0.479922141,0.904722238,0.609866648,0.61766639,0.859442306,0.80548942
5,0.576721516,0.182922469,0.239932011,0.886511933,0.028674152,0.489901389,0
.167927146,0.97868065,0.712694472,0.500471624,0.471088375,0.059618868,0.681
971904];
    dvar float+ rate[1..n];
    dvar float+ ra;
    dvar float+ rb;
    dvar float+ rc;
    dvar float+ p_loss[1..n];
    dvar boolean refuse[1..n];

    //定义目标函数
    maximize
       sum (i in 1..n)
account[i]*(1+(level_A[i]+level_B[i]+level_C[i])*rate[i]+level_D[i]*(-
1))*d[i]*refuse[i];

    //输入约束条件
    subject to {
    forall (i in 1..n) (level_A[i]==1)=>rate[i]==ra;
    forall (i in 1..n) (level_B[i]==1)=>rate[i]==rb;
    forall (i in 1..n) (level_C[i]==1)=>rate[i]==rc;

    forall (i in 1..n)
(level_A[i]==1)=>(p_loss[i]==7.52406404059649*rate[i]-0.0979300813722221);
    forall (i in 1..n)
(level_B[i]==1)=>(p_loss[i]==7.35107535431523*rate[i]-0.117765769021589);
    forall (i in 1..n)
(level_C[i]==1)=>(p_loss[i]==7.46841647701602*rate[i]-0.137910844288652);

    forall (i in 1..n)
(level_A[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;
    forall (i in 1..n)
(level_B[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;
    forall (i in 1..n)
(level_C[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;
```

```
forall (i in 1..n) (ra>=0.04&&ra<=0.15);
forall (i in 1..n) (rb>=0.04&&rb<=0.15);
forall (i in 1..n) (rc>=0.04&&rc<=0.15);
ra>=0.04&&ra<=0.15;
rb>=0.04&&rb<=0.15;
rc>=0.04&&rc<=0.15;
ra<=rb;
rb<=rc;
}
```

源程序 3：

```
/*********************************************
 * OPL 12.9.0.0 Model
 * Author: asus
 * Creation Date: 13 Sep 2020 at 12:25:15
 *********************************************/
//第二问 求解信贷金额系数与信贷金额映射函数
//企业贷款金额系数
int n=177;
int K=10000;
float
m_c[1..n]=[0.097907238,1,0.176877538,0.483259508,0.625001206,0.316425668,0.
443525722,0.009915475,0.17321664,0.415323485,0.746456649,0.481409153,0.2030
70398,0.211396922,0.289714361,0.01969181,0.17383127,0.180356117,0.346625069
,0.129223501,0.321482186,0.388263733,0.648047059,0.174242226,0.326988313,0.
647844824,0.009811872,0.227579212,0.13842125,0.258316869,0.182152873,0.1575
75366,0.202378193,0.105900752,0.075840911,0.29199841,0.073415022,0.05383682
5,0.020101527,0.137884762,0.048076649,0.310812249,0.274508076,0.005894931,0
.06073547,0.391879292,0.506429247,0.252663619,0.162439249,0.109922311,0.385
932028,0.003776067,0.098430407,0.032177946,0.316218932,0.083660711,0.131753
971,0.193765435,0.09902461,0.21716139,0.066662524,0.154503431,0.097347806,0
.250434301,0.044228882,0.082940712,0.104391314,0.048403372,0.078316135,0.01
3851925,0.162001272,0.041756415,0.086742051,0.129827353,0.086846453,0.00589
3722,0.002453836,0.122687584,0.100520488,0.053161163,0.037349738,0.01096842
8,0.081918124,0.049206309,0.012205607,0.105059991,0.105558422,0.07249007,0.
029862963,0.015664995,0.017078237,0.030934884,0.008711952,0.060099254,0.027
416931,0.033663397,0.043932179,0.02470929,0.022376431,0.003892218,0.0115363
28,0.018407776,0.031362191,0.020255193,0.051563698,0.001317143,0.027185035,
0.010016144,0.044550536,0.005947899,0.009111882,0.025522282,0.014510872,0.0
2757337,0.100329099,0.019318245,0.005499638,0.007722165,0.013513578,0.01472
```

```
4338,0.009228918,0.021578016,0.019617437,0.014910949,0.004486228,0.01269403
,0.005354366,0.028030269,0.020662188,0.011148778,0.010377897,0.056897252,0.
011882006,0.004461383,0.002232014,0.005115923,0.025332019,0.012387407,0.002
11272,0.004384561,0.010382633,0,0.00246644,0.00280961,0.011084245,0.0018914
59,0.002394558,0.002730298,0.002234792,0.003614067,0.002063635,0.001591222,
0.007871842,0.002106467,0.002243634,0.001822166,0.000821967,0.003256036,0.0
00718471,0.001419026,0.002365771,0.000891831,0.001539807,0.002289012,0.0020
69478,0.000812247,0.000697381,0.000451834,0.00103516,0.000278288,0.00024793
8,2.06587E-05,8.37264E-05,0.000550691,0.000196878,-0.000208205];
    dvar float f_mc[1..n];
    dvar float a;
    dvar float b;
    dvar float c;
    //约束条件
    subject to{
      (sum(i in 1..n) f_mc[i])==K;
      forall(i in 1..n) m_c[i]==0=>f_mc[i]>=10;
      forall(i in 1..n) m_c[i]==1=>f_mc[i]<=100;
      forall(i in 1..n) forall (m in 1..n) m_c[m]<=m_c[i]=>f_mc[m]<=f_mc[i];
      forall(i in 1..n) f_mc[i]==a+b*m_c[i]+c*m_c[i]^2;

    }
```

源程序 4：

```
    /*********************************************
     * OPL 12.9.0.0 Model
     * Author: asus
     * Creation Date: 13 Sep 2020 at 08:53:55
     *********************************************/
    //第三问 银行信贷决策模型
     //企业信誉评级信息
    int n=302;
    float
account[1..n]=[0,6934195.063,120028671.5,196457834.2,70789757.53,84720904.5
4,12536317.5,34219365.29,44250608.44,22412318.96,34345202.97,34262567.99,58
9251.78,13393342.92,29411445.2,57438860.34,57650364.04,27890519.96,8503193.
648,12255790,20988332.64,1436132.105,1155338.556,13440795.42,14380313.65,26
781348.58,9996850.614,24840191.14,16247867.77,74675047.42,0,1413472.183,458
81593.23,4531822.636,6368530.438,12349813.05,25265291.95,50037033.55,801972
.5571,10013141.64,16124540.44,6691852.012,19963509.96,0,29270044.06,1584367
```

```
3.38,12185757.66,17600357.2,4439617.391,0,5875445.293,0,25384381.74,9408381
.873,0,5688154.414,4307138.184,0,1768594.043,10665543.68,3731904.22,2401641
8.21,21213552.83,0,356447.0184,4316793.32,28383002.79,0,5641196.662,3585911
7.66,12204406.62,12953813.74,6380025.992,22362033.11,321415.6361,7619468.36
6,0,1356573.883,22397687.94,6479173.338,11466242.64,13731495.76,7665343.866
,15387257.72,0,5812680.951,11948588.76,6907895.601,21774302.95,3864191.829,
7710196.856,17638939.23,6423585.772,0,0,123302.4514,5049800.32,4226770.002,
3694658.043,0,0,5687386.054,0,1225780.675,6978455.869,0,12527459.92,0,30412
03.351,6297251.541,10357329.51,7028545.047,6164701.293,435571.9588,10031677
.1,192126.7509,0,8701224.431,0,8753437.305,3421625.294,4124454.33,3789600.4
34,2661754.827,975332.5992,6344636.836,3224445.196,0,2386949.794,721859.076
,6098479.29,7488971.508,3042402.734,2180358.533,7646030.765,2142706.483,126
7984.729,1004002.787,2207767.331,2164211.189,0,714399.4572,4271767.43,21368
69.707,3357456.787,5509355.16,2400867.693,0,0,3127602.685,1927826.163,0,0,1
310788.272,330663.2439,683105.2762,1321207.731,0,1525914.805,946077.3754,11
87960.582,3667695.124,0,91375.8108,496692.8853,1192072.245,1589224.645,0,42
9780.6405,19290.02227,3171364.596,359514.1104,0,524991.75,1482591.677,0,723
627.5994,1866181.176,855342.6108,1343092.256,0,7280678.446,1385642.758,4586
42.3952,385209.004,974838.964,1301554.703,0,0,508917.8406,686871.849,126455
7.028,1406816.976,0,0,1073732.707,366503.7744,1125210.935,0,0,0,0,325146.29
28,2002203.951,0,1615369.456,990997.0218,847038.1145,4550802.377,878901.709
5,273439.1928,180437.871,415119.3776,1811245.354,1007030.68,137442.7392,0,2
68990.8912,0,847415.184,0,0,18465.61493,157924.4568,266692.9068,802911.9883
,152326.5935,0,0,0,0,211423.0173,230937.096,198702.8241,0,318840.1973,0,336
564.612,0,0,307236.5043,0,179467.3104,0,0,160869.0096,706376.538,137080.669
2,0,0,199406.8032,165850.4925,0,0,0,0,0,0,94054.86,0,0,0,0,280011.93,0,7088
8.05,129700.2912,209131.098,0,0,0,0,0,0,100122.9408,139025.152,203019.7092,
0,0,134938.824,0,93210.6384,69361.5504,0,0,41271.1368,112571.8416,0,42930.4
656,40514.10165,0,22418.616,29934.45504,0,39162.774,0,0,26508.2196,4197.040
65,0,0,0];
    int
level_A[1..n]=[1,1,1,1,1,1,1,1,1,0,0,0,1,1,0,1,0,0,0,0,0,1,1,1,0,0,1,0,0,1,
1,0,0,0,0,0,0,1,0,1,0,1,0,0,0,1,0,0,0,0,1,1,0,0,1,1,0,0,0,1,0,0,0,0,0,0,0,1
,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0
,0,0,0,1,0,0,0,0,0];
    int
level_B[1..n]=[0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,0,1,1,1,1,1,0,0,0,1,1,0,1,1,0,
0,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1,1,1,1,0,1,1,1,0
```

```
,0,1,0,0,1,0,1,0,1,0,1,1,1,1,1,0,1,1,0,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,
1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
,0,0,0,0,0,0,0,0,0];
    int
level_C[1..n]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,1,0,1,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
0,1,0,1,0,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,0,1,1,0,1,1
,1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1,1,1,1,0,1,0,1,0,1,1,0,1,0,0,1,0,1,1,1,0,0,
1,1,0,1,0,1,1,0,0,1,0,1,0,0,0,1,1,1,0,0,0,1,1,1,0,1,1,1,0,1,0,1,1,0,0,0,0,0
,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,0,0,0,
0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,0,0,0,1,0,0,1,0,0,0
,1,1,1,0,1,1,0,0,0];
    int
level_D[1..n]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0
,0,0,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,1,0,1,1,0,1,0,0,0,1,1,
0,0,1,0,1,0,0,1,0,0,1,0,1,1,1,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,1,0,0,1,1,1,1,1
,1,0,1,1,1,0,1,0,1,1,0,1,1,0,0,1,1,0,1,1,1,1,1,1,0,1,1,1,0,1,1,0,0,0,1,1,1,
1,1,0,0,1,1,1,1,1,0,1,0,0,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1,0,0,1,1,0,0,1,1,0,1,0,0
,0,0,0,0,0,0,1,1,1];
    int
d[1..n]=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-
1,1,1,1,1,1,1,1,1,1,1,1,-1,1,1,1,1,1,1,1,1,1,1,1,-1,1,-1,1,1,1,1,1,1,1,1,-
1,1,1,1,1,1,1,1,1,1,1,-1,1,1,-1,1,1,1,-1,1,-1,-1,-1,1,1,-1,-1,1,1,1,-
1,1,1,1,1,1,1,-1,1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,1,-1,-1,1,1,-1,1,-1,1,1,-
1,1,1,-1,1,-1,-1,-1,1,1,1,1,-1,-1,1,1,1,1,1,-1,1,1,1,1,1,1,-1,1,1,-1,-1,-1,-1,-
1,-1,1,-1,-1,-1,1,-1,1,-1,-1,1,-1,-1,1,1,-1,-1,1,-1,-1,-1,-1,-1,-1,1,-1,-
1,-1,1,-1,-1,1,1,1,-1,-1,-1,-1,-1,1,1,-1,-1,-1,-1,1,-1,1,1,-1,-1,-1,-1,-1,-
1,-1,1,1,1,-1,-1,-1,-1,1,1,1,-1,-1,1,1,1,-1,-1,1,-1,1,1,1,1,1,1,1,1,-1,-1,-1];
    float
loss_rand[1..n]=[0.814723686,0.905791937,0.126986816,0.000091337,0.63235924
6,0.097540405,0.278498219,0.546881519,0.957506835,0.964888535,0.157613082,0
.970592782,0.957166948,0.485375649,0.800280469,0.141886339,0.421761283,0.91
5735525,0.79220733,0.959492426,0.655740699,0.035711679,0.849129306,0.933993
```

248,0.678735155,0.757740131,0.743132468,0.39222702,0.65547789,0.171186688,0
.706046088,0.031832846,0.276922985,0.046171391,0.097131781,0.823457828,0.69
4828623,0.31709948,0.950222049,0.034446081,0.43874436,0.381558457,0.7655167
88,0.795199901,0.186872605,0.489764396,0.445586201,0.64631301,0.709364831,0
.754686682,0.276025077,0.679702677,0.655098004,0.162611735,0.118997682,0.49
8364052,0.959743959,0.340385727,0.585267751,0.223811939,0.751267059,0.25509
5115,0.505957052,0.699076723,0.890903253,0.959291425,0.54721553,0.138624443
,0.149294006,0.257508254,0.840717256,0.254282179,0.814284826,0.243524969,0.
929263623,0.349983766,0.19659525,0.251083858,0.616044676,0.473288849,0.3516
59507,0.830828628,0.585264091,0.549723608,0.917193664,0.285839019,0.7572002
29,0.753729094,0.380445847,0.567821641,0.07585429,0.053950119,0.530797553,0
.77916723,0.934010684,0.129906208,0.568823661,0.469390641,0.01190207,0.3371
22644,0.162182308,0.794284541,0.311215042,0.528533136,0.165648729,0.6019819
41,0.262971285,0.654079098,0.689214503,0.748151593,0.450541599,0.083821378,
0.228976969,0.913337362,0.152378019,0.825816977,0.538342435,0.996134717,0.0
78175529,0.44267827,0.10665277,0.961898081,0.004634224,0.774910465,0.817303
221,0.868694705,0.084435846,0.399782649,0.259870403,0.80006848,0.431413827,
0.910647594,0.181847028,0.263802917,0.14553898,0.136068559,0.869292208,0.57
9704587,0.549860202,0.144954798,0.853031118,0.622055131,0.350952381,0.51324
954,0.401808034,0.075966692,0.239916154,0.123318935,0.183907788,0.239952526
,0.417267069,0.04965443,0.90271611,0.94478719,0.490864092,0.489252638,0.337
71941,0.900053846,0.369246781,0.111202755,0.780252068,0.389738837,0.2416912
86,0.403912146,0.096454525,0.131973293,0.942050591,0.95613454,0.575208595,0
.059779543,0.234779913,0.353158571,0.82119404,0.015403438,0.043023802,0.168
990029,0.649115475,0.731722386,0.647745963,0.450923706,0.547008892,0.296320
806,0.744692807,0.188955015,0.686775433,0.183511156,0.368484596,0.625618561
,0.780227435,0.081125769,0.929385971,0.775712679,0.486791632,0.435858589,0.
446783749,0.306349472,0.508508655,0.510771564,0.817627708,0.794831417,0.644
31813,0.378609383,0.811580458,0.532825589,0.350727104,0.939001562,0.8759428
11,0.550156343,0.622475086,0.587044705,0.207742293,0.30124633,0.470923349,0
.23048816,0.844308793,0.19476429,0.225921781,0.170708047,0.227664298,0.4356
98684,0.311102287,0.923379642,0.430207391,0.18481632,0.904880969,0.97974837
8,0.438869973,0.111119223,0.258064696,0.408719846,0.594896074,0.262211748,0
.602843089,0.71121578,0.221746734,0.117417651,0.296675873,0.318778302,0.424
16676,0.507858285,0.085515797,0.262482235,0.801014623,0.029220278,0.9288541
39,0.730330863,0.488608974,0.578525061,0.23728358,0.458848828,0.963088539,0
.546805719,0.521135831,0.231594387,0.488897744,0.624060088,0.679135541,0.39
5515216,0.367436649,0.987982003,0.037738866,0.885168008,0.913286828,0.79618
3874,0.098712279,0.261871184,0.33535684,0.679727951,0.136553137,0.721227499
,0.106761862,0.653757349,0.494173937,0.779051723,0.715037078,0.903720561,0.
890922504,0.334163053,0.698745832,0.197809827,0.030540946,0.74407426,0.5000
22436,0.479922141,0.904722238,0.609866648,0.61766639,0.859442306,0.80548942
5,0.576721516,0.182922469,0.239932011,0.886511933,0.028674152,0.489901389,0

```
.167927146,0.97868065,0.712694472,0.500471624,0.471088375,0.059618868,0.681
971904];
    dvar float+ rate[1..n];
    dvar float+ ra;
    dvar float+ rb;
    dvar float+ rc;
    dvar float+ p_loss[1..n];
    dvar boolean refuse[1..n];
    //定义目标函数
    maximize
       sum (i in 1..n)
account[i]*(1+(level_A[i]+level_B[i]+level_C[i])*rate[i]+level_D[i]*(-
1))*d[i]*refuse[i];
    //输入约束条件
    subject to {
    forall (i in 1..n) (level_A[i]==1)=>rate[i]==ra;
    forall (i in 1..n) (level_B[i]==1)=>rate[i]==rb;
    forall (i in 1..n) (level_C[i]==1)=>rate[i]==rc;


    forall (i in 1..n)
(level_A[i]==1)=>(p_loss[i]==7.52406404059649*rate[i]-0.0979300813722221);
    forall (i in 1..n)
(level_B[i]==1)=>(p_loss[i]==7.35107535431523*rate[i]-0.117765769021589);
    forall (i in 1..n)
(level_C[i]==1)=>(p_loss[i]==7.46841647701602*rate[i]-0.137910844288652);

    forall (i in 1..n)
(level_A[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;
    forall (i in 1..n)
(level_B[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;
    forall (i in 1..n)
(level_C[i]==1&&(loss_rand[i]<=p_loss[i]))=>refuse[i]==0;

    ra>=0.04;
    rc<=0.15;
    ra<=rb;
    rb<=rc;
    }
```

源程序 5：

```
/*********************************************
 * OPL 12.9.0.0 Model
 * Author: asus
 * Creation Date: 13 Sep 2020 at 12:37:48
 *********************************************/
 //第三问 求解信贷金额系数与信贷金额映射函数
////企业贷款金额系数 银行信贷限额
int n=146;
int K=10000;
float
m_c[1..n]=[0.097901295,1,0.177043459,0.483363669,0.625076797,0.31656346,0.4
83974001,0.008265171,0.189150807,0.415441341,0.814377489,0.393954964,0.1200
66812,0.173080398,0.296446556,0.016262378,0.189821177,0.203093915,0.3782855
06,0.141167966,0.350862435,0.229477179,0.648118004,0.174408678,0.356867908,
0.706822637,0.011270314,0.227734912,0.094477672,0.281968706,0.223767054,0.1
72091039,0.248584034,0.062659959,0.082944151,0.358550311,0.08029826,0.06078
8403,0.024925945,0.15061471,0.052661972,0.33922485,0.299628281,0.004976297,
0.060924802,0.400912355,0.506528738,0.172354496,0.182941501,0.090072451,0.3
15853063,0.004481403,0.107582271,0.019105264,0.316356765,0.091473123,0.1619
26324,0.193927951,0.108230362,0.21731919,0.082057469,0.168740512,0.09752975
8,0.307550096,0.054530822,0.090687828,0.07128012,0.059653027,0.080287406,0.
017257526,0.176918326,0.04290432,0.088903082,0.146260514,0.087030522,0.0060
94109,0.002654916,0.122864428,0.123602048,0.058207596,0.037543784,0.0137194
02,0.089572503,0.033661565,0.010138537,0.086094992,0.105738719,0.030743014,
0.01785375,0.031130223,0.010033154,0.060288714,0.03012864,0.044124898,0.027
175445,0.018458443,0.004612045,0.009591055,0.021497573,0.016723235,0.051754
879,0.00123159,0.006012294,0.04474313,0.010163521,0.026304576,0.102796126,0
.018327999,0.017805326,0.015109518,0.005118371,0.015836759,0.004534106,0.02
8226193,0.022761315,0.013940696,0.011906963,0.064230689,0.012357106,0.00248
9785,0.005805172,0.01191229,0.011283586,0.002927518,0.003203196,0.002476074
,0.009919813,0.002757762,0.002283707,0.001269437,0.003896485,0.000942156,0.
001773006,0.001355162,0.00190474,0.002808803,0.001257511,0.00092059,0.00052
3752,0.00153103,0.000513057,0.000363597,0.000493967,0.000315194,0];
dvar float f_mc[1..n];
dvar float a;
dvar float b;
dvar float c;
//约束条件
subject to{
  (sum(i in 1..n) f_mc[i])==K;
  forall(i in 1..n) m_c[i]==0=>f_mc[i]>=10;
  forall(i in 1..n) m_c[i]==1=>f_mc[i]<=100;
```

```
  forall(i in 1..n) forall (m in 1..n) m_c[m]<=m_c[i]=>f_mc[m]<=f_mc[i];
  forall(i in 1..n) f_mc[i]==a+b*m_c[i]+c*m_c[i]^2;

}
```