基于整数规划模型的网络金融交易问题研究

摘要

网络金融交易是适应电子商务发展需要而产生的网络时代的金融运行模式。本文针对网络金融交易的订单分配问题，通过建立整数规划模型，实现了对于平台交易费用最优化的求解。

针对问题一，考虑平台有最大允许交易量的限制，将其考虑为背包问题的改进，建立 0-1 整数规划模型，以订单交易费用除以交易量作为权衡值，使用贪心算法进行初解的计算，之后采用模拟退火算法，对初解进行优化，得到实例 1-4 的总交易费用最少的平台交易费 673、336、464、234。

针对问题二，本问要求每个平台的最大允许交易量都要满足，因此需要对问题一建立模型进行改进，以所有平台的总交易费用最小为优化目标，建立线性最优化模型。继续使用贪心算法和模拟退火算法继续求解，但其结果于最优解相差就多，之后使用分支界定法，获取最优值。得到实例 5-8 的总交易费用最小值分别为 1408、2039、4979、5070。

本文的特色在于将背包问题机理分析与订单交易问题相结合，并灵活采用贪心算法进行求解。此外，对于解空间较复杂的模型，设计了模拟退火算法算法，在保证了求解精度的同时，有效降低了运算的时间复杂程度，为 xxx 的设计和发展提供了参考依据。

关键词：整数规划模型；贪心算法；模拟退火算法；分支界定法

一、问题重述

1.1 问题背景

网络金融，是指在国际互联网(Internet)上开展的金融业务（包括网络银行、网络证券、网络保险, 网络股票、期权等金融服务及相关内容），它不同于传统的以物理形态存在的金融活动，是存在于电子空间中的金融活动，其存在形态是虚拟化的、运行方式是网络化的。它是信息技术特别是互联网技术飞速发展的产物，是适应电子商务发展需要而产生的网络时代的金融运行模式。

1.2 要解决的问题

在此，我们探讨的网络金融交易问题场景设定如下：某一时间段t内，共有 m 个网络支付平台（下简称平台）和 n 笔可能要发生的交易，每笔交易$T_i(1 \leq i \leq n)$都需经平台进行，每笔交易$T_i(1 \leq i \leq n)$都有其相应的交易量w_i和交易费用f_i（注：此项交易费用完全归处理交易T_i的平台P_j所有）。为了降低金融风险和维护现有 m 个平台的共同发展，金融监管部门依据历史数据对于每个平台都设置其最大允许交易量，即平台$P_j(1 \leq j \leq m)$的最大允许交易量为C_j，假设所有交易都必须由金融监管部门统一分配。请解答下面两个问题：

问题（1）：部分交易完成（最大化最小总交易费用）。由于平台有最大允许交易量的限制，故此时不能保证每笔交易都能成功进行，平台为了追求自身利益将会有选择性的进行交易，但是每笔交易T_i最多只能在其中某一个平台上进行（即单笔交易不能拆分）。为了维护现有 m 个平台的共同发展和重点扶持小微平台，请你们为金融监管部门设置可行交易分配方案使得总交易费用最少的平台其总交易费用最大。试建立此问题的一般数学模型并设计快速高效的求解算法，利用附件中的实例 1-4 进行检验，将计算结果具体信息以附件形式提交，并对计算结果的优劣进行分析。

问题（2）：全部交易完成（最小化总交易费用）。假设必须要完成所有 n 笔交易且有$\sum_{i=1}^{n} w_i = \sum_{j=1}^{m} C_j$成立，即每个平台的最大允许交易量都要满足。不同于问题（1），此时，交易T_i可能会在多个平台上进行（即单笔交易可拆分），不管交易T_i在平台P_j上进行交易的交易量大小如何，只要交易T_i在平台P_j上进行了交易或者部分交易，那么平台P_j都会收取相应的交易费用f_i。金融监管部门希望社会总成本最小，即总交易费用最小，也就是说尽量使得单笔交易不要拆分或者尽可能少的拆分，请你们为金融监管部门设置可行交易分配方案使得总交易费用最小。试建立此问题的一般数学模型并设计快速高效的求解算法，利用附件中的实例 5-8 进行检验，将计算结果具体信息以附件形式提交，并对计算结果的优劣进行分析。

二、问题分析

2.1 问题一的分析

由于平台有最大允许交易量的限制，故此时不能保证每笔交易都能成功进行，平台为了追求自身利益将会有选择性的进行交易。本问为 0-1 背包问题的进一步模型，即为多个背包的背包问题。可以建立 0-1 整数规划模型，采用贪心算法进行初始解的求解，之后在使用模拟退火算法进行解的优化，获得最终结果。

2.2 问题二的分析

问题二要求所有交易全部完成，即每个平台的最大允许交易量都要满足。不同于问题（1），此时，交易可能会在多个平台上进行（即单笔交易可拆分）。因此需要对第一问的模型进行一定的修改，将优化目标和约束条件进行改进，同样采用贪心算法进行初解的求解，采用模拟退火算法进行解的优化，得到最终结果。

三、模型假设

1. 假设所有交易的交易量和交易费用都固定不变。
2. 假设所有平台没有不能处理的交易。
3. 假设每个交易只有交易量和交易费用的区别。

四、符号说明

符号	含义
T_i	第 i 个交易
w_i	第 i 个交易的交易量
f_i	第 i 个交易的交易费用
n	交易的个数
m	平台的个数
C_j	平台 j 的最大允许交易量
X_{ij}	平台交易矩阵

Y_j	平台收益矩阵
Z	最小平台交易费用
M_j	平台总交易量矩阵
P_{ij}	平台交易比例矩阵

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 模型的准备

在每一笔网络金融交易之后，平台的最大允许交易量均会减少，直至得到最大的允许交易量，因此，需要定义交易订单和交易平台的一些概念，以便于将具体的平台交易问题数字化，便于模型的建立于求解。

已知每笔交易$T_i(1 \le i \le n)$，对应的交易量w_i和交易费用f_i，平台记为P_j，现有 m 个平台，平台$P_j(1 \le j \le m)$的最大允许交易量为C_j

定义一（平台交易矩阵）

为了表示平台P_j和每笔交易T_i之间的关系，我们假设第 j 个平台处理第 i 个交易，记为 平台交易矩阵，记为X_{ij}，即

$$X_{ij} = \begin{cases} 1 & \text{第 j 个平台处理第 i 个交易} \\ 0 & \text{else} \end{cases} \tag{1}$$

（1）目标函数的确定

定义二（平台收益矩阵）

假设第 j 个平台的收益为Y_j，记为平台收益矩阵，则有：

$$Y_j = \sum_{i=1}^{n} X_{ij} \cdot f_i \ (1 \le j \le m) \tag{2}$$

令总交易费用最少的平台其总交易费用为 Z，即：

$$Z \le Y_j \ (1 \le j \le m) \tag{3}$$

问题要求总交易费用最少的平台其总交易费用最大，即：

$$max\ Z \tag{4}$$

（2）约束条件的确定

定义三（平台交易量矩阵）

每个平台都设置有其最大允许交易量，当有订单交易后，该平台剩余的交易总量将会减少，为了保障网络平台的正常交易，我们需要限制平台的最大交易数量，令每个平台的交易总量不得超过其最大交易量。用M_j来表示第 j 个平台所处理的总的交易量，即：

$$M_j = \sum_{i=1}^{n} X_{ij} \cdot w_i \ (1 \le j \le m) \tag{5}$$

由于第 j 个平台的最大允许交易量为C_j，该数据附件中均已给出，为已知量，即：

$$M_j \le C_j \tag{6}$$

而由于订单的唯一性和独立性，一个订单完成后，其无法再次被其他平台接受处理，因此有约束条件每个订单只能在一家平台处理，故有:

$$\sum_{j=1}^{m} X_{ij} = 1 \ (1 \le i \le n) \tag{7}$$

5.1.2 0-1 整数规划模型的建立

根据上述目标方程和约束条件，可以建立如下的 0-1 整数规划模型

目标函数为

$$max\ Z$$

约束条件为：

$$\begin{cases} M_j = \sum_{i=1}^{n} X_{ij} \cdot w_i \\ Y_j = \sum_{i=1}^{n} X_{ij} \cdot f_i \\ \sum_{j=1}^{m} X_{ij} \le 1 \\ Z \le Y_j \\ M_j \le C_j \\ X_{ij} \in \{0,1\} \\ 1 \le j \le m \end{cases} \tag{8}$$

5.1.3 模型的求解

本问的问题可以转化为多背包问题，其本身是一个组合优化问题，也是一个典型的 NP 难问题。如果使用枚举的方法,我们需要找到 n 个物品的所有子集，然后在那些满足约束条件的子集中比较物品的总价值,找到总价值最大的子集，

也就是问题的最优解。但是我们知道,大小为 n 的集合的子集数目为 2^n ，所以当背包问题的规模变大(n 变大)的时候，要找出所有的子集是一个不现实的做法，因为计算复杂度的指数级增长已经使得问题在规模稍大的时候就无法在可以接受的时间内得到解决。

因此背包问题需要采用一些计算复杂度较低的，但是能够提供令人满意的解的算法，而模拟退火算法是解决背包问题的重要手段。大量的实验证明，模拟退火算法能够处理规模较大的背包问题,而且能够鲁棒地得到满意的解。

为了更快的获得一个较好的初始解，我们贪心算法快速的进行初解的求值，它是寻找最优解问题的常用方法，这种方法模式一般将求解过程分成若干个步骤，但每个步骤都应用贪心原则，选取当前状态下最好/最优的选择（局部最有利的选择），并以此希望最后堆叠出的结果也是最优的解。

按照贪心算法的思想，按一定的规则进行遍历，步骤如下：

Step1：将每个交易订单的交易费用除以交易量，计算订单的性价比。

Step2：将每个订单，按照订单的性价比从小到大进行排序。

Step3：计算平台个数 i，从性价比高的订单开始，按照订单顺序依次将其置入平台中进行交易，每三个为一轮。

Step4：之后将其每三个为一组，尽可能保证每个平台的性价比总和均衡，以此保证每个交易平台的总交易费用相同。

Step5：当交易总量达到平台的最大交易量时，停止交易

Step6：获取剩余的未交易订单的交易量，判断其是否满足各个交易平台的交易量，若满足则置入。

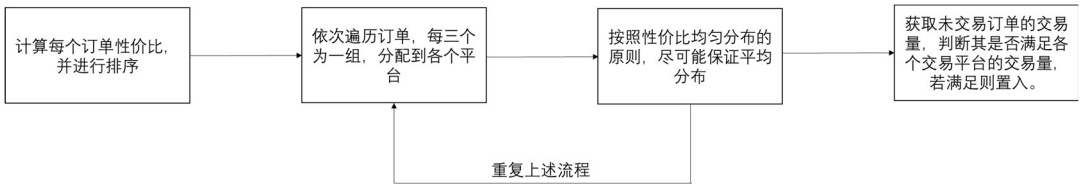

图 1：贪心流程图

使用贪心算法进行求解后，只能得到局部最优解，而采用一般的分类优化算法计算时间长，次数多，难以进行大量，多元数据的求解，因此采用模拟退火算法进行求解。

模拟退火算法最早的思想是由 N.Metropolis 等人于 1953 年提出。它是基于迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。该算法的优点为求解速度快，可达到全局最优解。

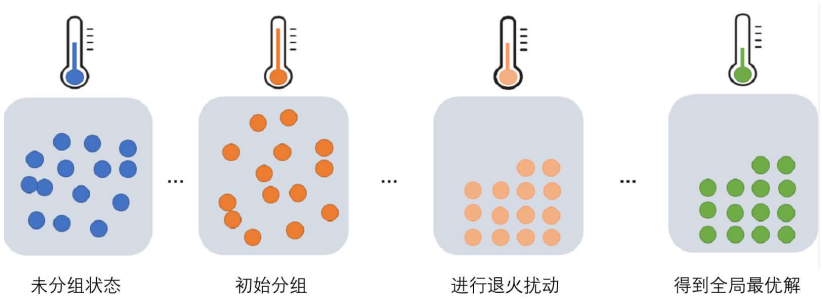

图 2：退火算法示意图

模拟退火算法包括三函数两准则，即状态产生函数，状态接受函数，温度更新函数；内循环终止和外循环终止准则。这些环节之间影响模拟退火算法的优化性能。

求解模拟退火算法描述如下：

(1)解空间。解空间 S 可表示为数据的所有分组形式的排列组合。

(2)目标函数。目标函数为交易总量。要求

$$\boldsymbol{max\ min}(Y_j)$$

(3)新解的产生。利用上一步迭代分好的组，每次选取两个组，每组内各挑选一个数据进行交换，得到新的分组。

选组要求为：1.每次选取两个不同的组；2.三个小组两两之间交易的个数不能完全相同。

(4)代价函数差。假设原先的分组数据最小交易费用组的费用为 T1，新的分组数据最小交易费用组的费用为 T2。隐匿量差可表示为：

$$\Delta T = T2 - T1$$

(5)接受准则

$$P = \begin{cases} 1, \Delta T < 0 \\ exp(-\Delta T/T), \Delta T \geq 0 \end{cases} \qquad (13)$$

如果$\Delta T < 0$，则接受新的分组；否则，以概率$exp(-\Delta T/T)$接受新的分组，即用计算机产生一个[0,1]区间上均匀分布的随机数 rand，若 rand <= $exp(-\Delta T/T)$则接受。

(6)降温。利用选定的降温系数 α 进行降温，取新的温度 T 为 AT(这里 T 为上一步迭代的温度)，这里选 α=0.999。

(7)结束条件。用选定的终止温度$e = 10^{-30}$，判断退火过程是否结束。若 T<e，则算发结束，输出当前状态。

其算法的步骤具体为：

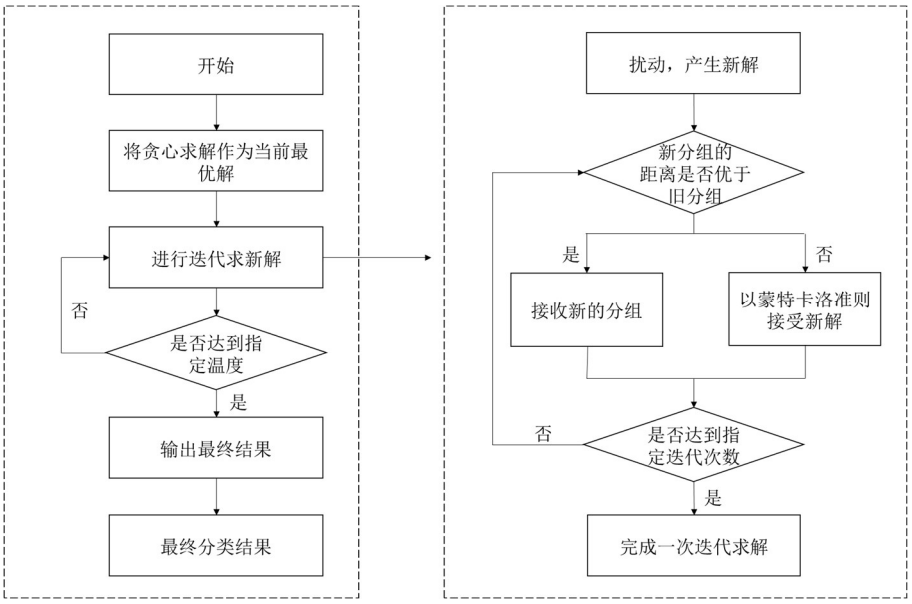

图 3 模拟退火算法流程图

经过求解后得到每个平台的交易结果如以下表格所示：

表 1：实例 1 求解值

	平台 1	平台 2
总交易量	1999	1991
总交易费用	673	674

表 2：实例 1 求解值

	平台 1	平台 2	平台 3	平台 4
总交易量	991	993	998	989
总交易费用	336	336	336	337

表 3：实例 1 求解值

	平台 1	平台 2	平台 3	平台 4	平台 5	平台 6	…
总交易量	1500	1494	1498	1497	1499	1499	…
总交易费用	464	465	464	464	464	464	…

表 4：实例 1 求解值

	平台 1	平台 2	平台 3	平台 4	平台 5	平台 6	…
总交易量	797	780	789	799	789	790	…
总交易费用	235	234	237	234	236	234	…

得到实例 1-4 的总交易费用最少的平台交易费 673、336、464、234。具体结果见表格附件。

5.2 问题二模型的建立与求解

5.2.1 模型的准备

（1）目标函数的确定

本题要求在满足所有交易的情况下，使所有平台的总台的总交易费用最小，记 Y 为总交易费用，则有：

由于第 j 个平台的最大允许交易量为C_j，该数据附件中均已给出，为已知量，即：

$$Y = \sum_{i=1}^{m} y_i \tag{9}$$

所有平台的总台的总交易费用最小，则有：

$$min\, Y \tag{10}$$

（2）约束条件的确定

定义四（平台交易比例矩阵）

由于第二问的交易可以在不同的平台处理，所以每个交易就不一定只在一个平台进行交易，为了表示表示平台P_j和每笔交易T_i之间的关系，我们将第 i 个交易在第 j 个平台处理的量与第 i 个交易总量的比值记为平台交易量矩阵，记为P_{ij}

每个交易在所有平台交易比值总和为 1，固有如下约束：

$$\sum_{j=1}^{m} P_{ij} = 1 \tag{11}$$

由于每个平台处理的交易有上限，所以每个平台处理的交易总量固定，为C_i，即：

$$\sum_{i=1}^{n} P_{ij} \cdot w_i = C_i \tag{12}$$

一个交易只要在一个平台有交易量，就说明这个交易在这个平台处理，故平台交易矩阵可以和平台交易量矩阵建议等价关系，为：

$$X_{ij} = \begin{cases} 1, & P_{ij} > 0 \\ 0, & P_{ij} = 0 \end{cases} \tag{13}$$

上式可以化简为：

$$0 \le P_{ij} \le X_{ij} \tag{14}$$

5.2.2 线性规划模型的建立

目标函数为

$$min\, Y$$

约束条件为：

$$\begin{cases} Y = \sum_{i=1}^{m} y_i \\ Y_j = \sum_{i=1}^{n} X_{ij} \cdot f_i \\ X_{ij} \in \{0,1\} \\ \sum_{j=1}^{m} P_{ij} = 1 \\ \sum_{i=1}^{n} P_{ij} \cdot w_i = C_i \\ 0 \le P_{ij} \le X_{ij} \\ 1 \le j \le m \\ 1 \le i \le n \end{cases}$$

5.2.2 模型的求解

对于此问的求解，我们继续使用第一问当中的方法求解，即先使用贪心算法获取初始解，之后使用模拟退火算法继续优化，从而获取更优的解。

对于此问题，我们考虑在不同平台交易的订单价值越小越好，所以我们的贪心策略是优先在一个平台上交易费用较大的订单。直到所有订单完成结束程序。

贪心步骤如下：

a）对于初始情况首先对所有的订单按照订单费用排序，首先交易所有订单中费用高的订单。

b）从剩下所有订单中选取费用最大订单进行交易，如果此平台因为订单量的限制无法交易，则换一个平台进行交易。

c）重复 a）,b）贪心过程，当所有的平台都无法完整的交易此订单时，选择交易量空余大的平台，将此订单进行拆分处理，完成交易。

d）重复上述策略，直到所有的订单完成。

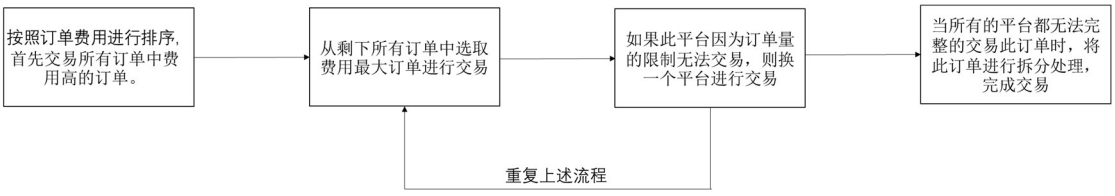

图 3：贪心流程图

采用贪心算法后获得初解，继续使用问题一中模拟退火算法的思路进行解的优化得到实例 5-8 的总交易费用最小值分别为 1456、2179、5012、5152。

其距离最优解的差距较大，因此采用 gurobi 中的分支定界法进行求解。

求解结果如下：

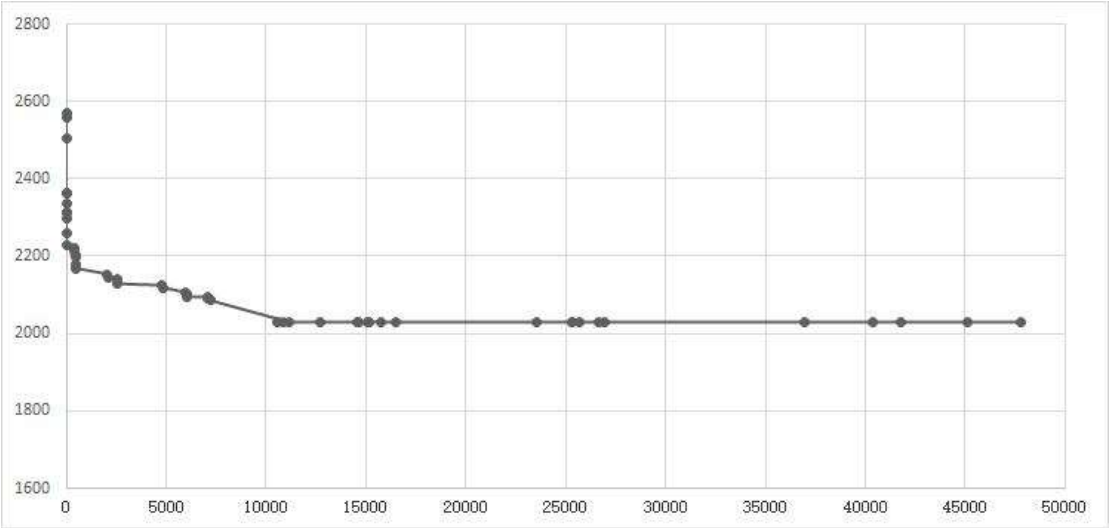

图 4：实例 6 迭代图

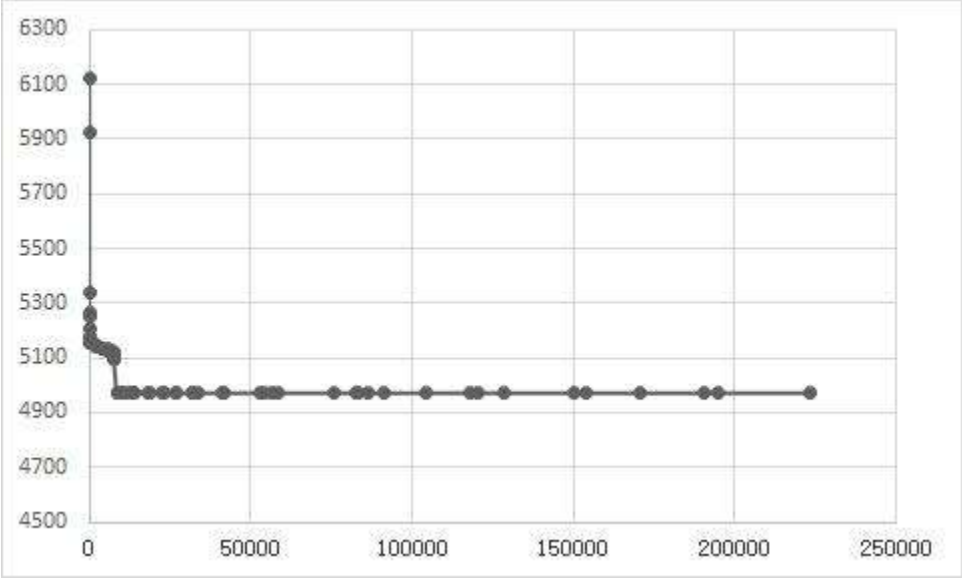

图 5：实例 7 迭代图

最终得到实例 5-8 的总交易费用最小值分别为 1408、2039、4979、5070。具体结果见表格附件。

六、灵敏度分析

采用实例 1 中的数据，改变订单交易总数的值，观察所求得的总交易费用最少的平台交易费，如图所示：

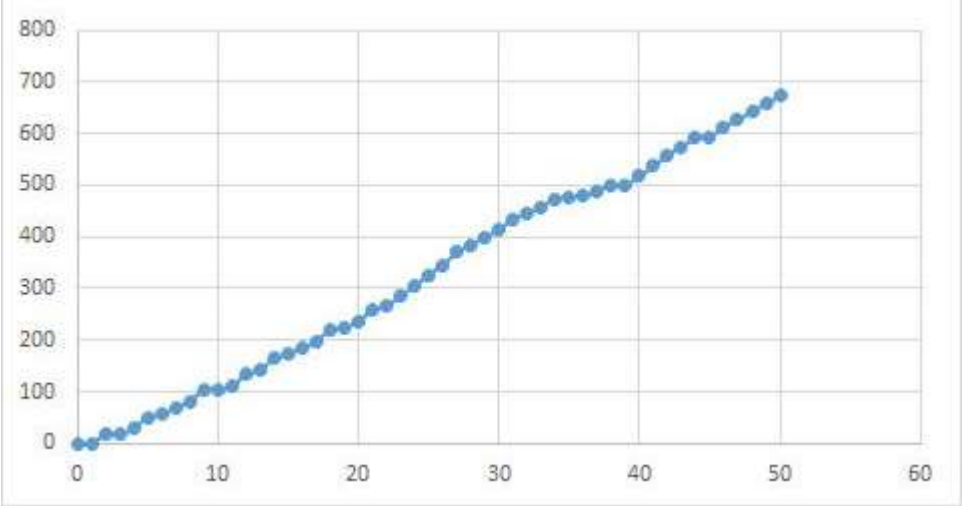

图 6：贪心流程图

可以看到当订单交易数量逐渐上升时，总交易费用也在逐渐上升，总体呈现线性，说明模型的灵敏度较好。

七、模型的评价与改进

7.1 模型优点

1. 在第一问中，建立了 0-1 整数规划模型，可以较好的描述具体化的订单交易问题。
2. 使用了贪心算法和模拟退火算法，具有较高的算法性能，算法的时间复杂度较小。
3. 使用 gurobi 软件中的分支定界法进行了解的验证，从而将解的取值更加优化。

7.2 模型缺点

1. 使用贪心算法和模拟退火算法所求得的值距离最优解的差距较大，可能算法流程和思想有待改进。
2. 整数规划模型的求解过程较长

7.3 模型改进

后续可以对整数规划模型进行优化，改进最优化目标和约束条件。此外，还可以尝试动态规划法，图论法等其他方法进行求解，优化求解的过程和结果。

八、参考文献

[1] 谢平, 尹龙. 网络经济下的金融理论与金融治理[J]. 经济研究, 2001, 4(6): 25.

[2] 贺毅朝, 刘坤起, 张翠军, 等. 求解背包问题的贪心遗传算法及其应用[J]. 计算机工程与设计, 2007, 28(11): 2655-2657.

[3] 金慧敏, 马良. 遗传退火进化算法在背包问题中的应用[J]. 上海理工大学学报, 2004, 26(6): 561-564.

[4] 冯静生. 网络金融: 优势, 问题及策略[J]. 金融纵横, 2003 (10): 49-51.

[5] Pedroso J P. Optimization with gurobi and python[J]. INESC Porto and Universidade do Porto,, Porto, Portugal, 2011, 1.

附录

```
1. import gurobipy
2. import pandas as pd
3. import matplotlib
4. import matplotlib.pyplot as plt
5. from matplotlib import style
6.
7. nn = 50
8. c = [2000, 2000]
9. JJ = range(nn)
10.II = range(len(c))
11.ff = []
12.ww = []
13.
14.list = []
15.
16.def readread():
17.    global ff
18.    global ww
19.    df = pd.read_excel(r'D:\Desktop\model5\one.xls',sheet_n
   ame='实例 1')
20.    data = df.values.tolist()
```

```
    ff = [data[j][2] for j in JJ]
    ww = [data[j][1] for j in JJ]

def model(n):

    f = ff[:n]
    w = ww[:n]
    m = len(c)

    J = range(n)
    I = range(m)


    # 模型建立
    MODEL = gurobipy.Model()

    # 定义变量
    x = MODEL.addVars(2, n, vtype=gurobipy.GRB.BINARY, name="x")
    # z = MODEL.addVars(1, vtype=gurobipy.GRB.INTEGER, name="CC")
    z = MODEL.addVar(vtype=gurobipy.GRB.INTEGER, name='z')
    # 更新模型
    MODEL.update()

    # 目标函数
    MODEL.setObjective(z, gurobipy.GRB.MAXIMIZE)

    # 约束条件
    MODEL.addConstrs((c[i] >= gurobipy.quicksum([w[j] * x[i, j] for j in J]) for i in I), name="one")
    MODEL.addConstrs((1 >= gurobipy.quicksum(x[i, j] for i in I) for j in J), name="two")
    MODEL.addConstrs((z <= gurobipy.quicksum(x[i, j] * f[j] for j in J) for i in I), name="three")

    MODEL.Params.LogToConsole = False  # 显示求解过程
    # MODEL.Params.MIPGap = 0.0001  # 百分比界差
    # MODEL.Params.TimeLimit = 30  # 限制求解时间为 100s
    MODEL.optimize()

    list.append(MODEL.getVars()[-1].x)

if __name__ == "__main__":
```

```
    #读数据
    readread()

    for k in range(51):
        model(k)
    for k in list:
        print(k)

    plt.plot([i for i in range(51)], list, '-p', color='grey',
             marker='o',
             markersize=8, linewidth=2,
             markerfacecolor='red',
             markeredgecolor='grey',
             markeredgewidth=2)
    plt.show()


import gurobipy
import pandas as pd
import csv

n = 200##############################################
c = [416 for i in range(50)]#################################################
J = range(n)
I = range(m)

def readread():
    global f
    global w
    df = pd.read_excel(r'D:\Desktop\model5\one.xls',sheet_name='实例7')#########################################################
    data = df.values.tolist()
    f = [data[j][2] for j in J]
    w = [data[j][1] for j in J]
```

```
    print(f)
        print(w)

if __name__ == "__main__":
        #读数据
        readread()

        #模型建立
        MODEL = gurobipy.Model()

        #定义变量
        x = MODEL.addVars(m, n, lb=0.0, ub=1, vtype=gurobipy.GRB.CONTINUOUS, name="x")
        p = MODEL.addVars(m, n, vtype=gurobipy.GRB.BINARY, name="p")

        #更新模型
        MODEL.update()

        #目标函数
        MODEL.setObjective(sum([sum([f[j] * p[i,j] for j in J]) for i in I]), gurobipy.GRB.MINIMIZE)

        #约束条件
        MODEL.addConstrs((c[i] == gurobipy.quicksum([w[j] * x[i,j] for j in J]) for i in I),name="one")
        MODEL.addConstrs((1 == gurobipy.quicksum(x[i,j] for i in I) for j in J),name="two")
        MODEL.addConstrs((x[i,j] <= p[i,j] for i in I for j in J),name="three")


        # MODEL.Params.LogToConsole = True  # 显示求解过程
        # MODEL.Params.MIPGap = 0.0001  # 百分比界差
        MODEL.Params.TimeLimit = 500  # 限制求解时间为 100s
        MODEL.optimize()

        list = []
        for v in MODEL.getVars():
            list.append([v.varName, v.x])
        print(list)
        with open("D:\Desktop\\three.csv", "w", newline="") as csvfile:
```

```
136.          writer = csv.writer(csvfile)
137.          writer.writerows(list)
```