



TBSI 清华-伯克利深圳学院
Tsinghua-Berkeley Shenzhen Institute



Tsinghua University

Tsinghua-Berkeley Shenzhen Institute
Department of Industrial Engineering

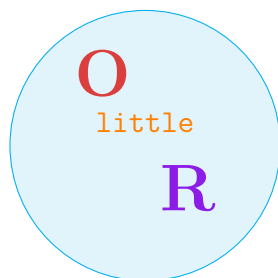
运小筹 OlittleRer

运小筹月刊 2020 年 11 月

本期作者：刘兴禄 段淇耀 夏旻 何彦东 王乃禹 曾文佳

本期主编：张祎 陈琰钰 王涵民 王基光

本期发布：周鹏翔 徐璐



目录

0.1 运小筹团队成员	1
0.2 运小筹公众号简介	2
第 1 章 2020-11-10: 论文代码复现	3
1.1 无人机与卡车联合配送 (Python+Gurobi)(The flying sidekick traveling sales- man problem)	3
1.1.1 无人机配送概述	3
1.1.2 文献笔记	4
1.1.3 无人机与卡车联合运输 (FSTSP) 的数学模型	7
1.1.4 Python 调用 Gurobi 求解 FSTSP	10
1.1.5 数值实验	23
1.1.6 结果展示与可视化	25
1.1.7 拓展	31
第 2 章 2020-11-13: 元启发式算法.	32
2.1 禁忌搜索 (Tabu Search) 解决 TSP 问题 (Python 代码实现)	32
2.1.1 Tabu Search 基本概念	32
2.1.2 Tabu Search 算法实现细节	32
2.1.3 问题与总结	40
第 3 章 2020-11-16:TSP 中两种不同消除子环路的方法及 callback 实现	42
3.1 TSP 问题的一般模型	42
3.2 TSP Model 1: subtour-elimination 消除子环路	44
3.2.1 TSP 整数规划模型	44
3.2.2 Python 调用 Gurobi 实现中的一些小问题	46
3.3 TSP Model 2 : MTZ 约束消除子环路	46
3.3.1 MTZ 约束消除子环路	46
3.3.2 为什么 MTZ 约束可以消除子环路?	48
3.4 Python+Gurobi: 用 callback 实现 TSP 的 subtour-elimination	49

3.4.1	callback 的工作逻辑: 王者荣耀版独家解读	49
3.5	使用 callback 的通用步骤	51
3.5.1	callback 实现 subtour-elimination 的详细代码	52
3.6	Python+Gurobi: 实现 TSP 的 MTZ 约束版	58
3.7	后记	63
3.8	参考文献	63
第 4 章	2020-11-23: Java 调用 cplex 求解运输问题	64
4.1	运输问题 (Transportation Problem) 描述	64
4.2	运输问题的数学模型	65
4.3	Java 调用 cplex 求解运输问题	66
4.3.1	transportation_node 类	66
4.3.2	transportation_relation 类	67
4.3.3	读取数据	67
4.3.4	在 cplex 中建立运输问题模型	69
4.3.5	主函数 main	71
4.4	求解结果	72
第 5 章	2020-11-26: 两阶段启发式算法求解带时间窗车辆路径问题 (附 Java 代码)	73
5.1	文档阅读说明	73
5.2	VRPTW 定义	73
5.3	Java 数据读取 (附 Java 代码)	74
5.4	两阶段启发式算法 (附 Java 代码)	74
5.5	邻域搜索策略 (附 Java 代码)	75
5.6	结果展示	76
第 6 章	2020-11-29: 手把手教你用 Python 调用 Gurobi 求解 VRPTW	77
6.1	带时间窗的车辆路径规划问题 (Vrptw)	77
6.2	python 调用 Gurobi 求解 Vrptw	79
6.2.1	首先我们定义一下需要用到的参数	79
6.2.2	定义一个读取数据的函数, 并对节点之间的距离进行计算	79
6.2.3	读取数据, 并定义一些参数	80
6.2.4	调用 gurobi 进行模型的建立与求解	80
第 7 章	2020-12-02: 单纯形法及其 Java 实现	85
7.1	单纯形法 (Simplex Method)	85
7.1.1	单纯形法伪代码	86
7.2	单纯形法的 Java 实现	87
7.2.1	变量符号定义	87

7.2.2 模型存储与读入 88

7.2.3 模型的标准型转化 88

7.2.4 单纯形法主体 88

7.2.5 输出结果 91

0.1 运小筹团队成员

运小筹编委：运小筹团队编委 (成员) 的单位及联系方式如下：

刘兴禄	清华大学，清华-伯克利深圳学院，邮箱： hsinglul@163.com
何彦东	清华大学，深圳国际研究生院 (博士后)，邮箱： ydhe602@163.com
段淇耀	清华大学，清华-伯克利深圳学院，邮箱： duanqy71@163.com
修宇璇	清华大学，清华-伯克利深圳学院，邮箱： yuxuanxiu@gmail.com
曾文佳	清华大学，工业工程系，邮箱： zwj19@mails.tsinghua.edu.cn
夏旻	清华大学，工业工程系，邮箱： xia970201@gmail.com
王梦彤	清华大学，工业工程系，邮箱： wmt15@mails.tsinghua.edu.cn
段宏达	清华大学，工业工程系，邮箱： dhd18@mails.tsinghua.edu.cn
王鑫	清华大学，工业工程系
王涵民	清华大学，清华-伯克利深圳学院，邮箱： humminwang@163.com
张祎	清华大学，清华-伯克利深圳学院，邮箱： zhangyihrrmm2015@163.com
王基光	清华大学，清华-伯克利深圳学院，邮箱： wangjg2020@163.com
陈琰钰	清华大学，清华-伯克利深圳学院，邮箱： yan_yu_chen98@163.com
周鹏翔	清华大学，清华-伯克利深圳学院，邮箱： zpx20@mails.tsinghua.edu.cn
徐璐	清华大学，清华-伯克利深圳学院，邮箱： xu-l20@mails.tsinghua.edu.cn

0.2 运筹小筹公众号简介

本公众号是致力于分享运筹优化 (LP、MIP、NLP、随机规划、鲁棒优化)、凸优化、强化学习等研究领域的内容以及涉及到的算法的代码实现。编程语言和工具包括 Java、Python、Matlab、CPLEX、Gurobi、SCIP 等。

Chapter 1

2020-11-10: 论文代码复现

作者: 刘兴禄, 清华大学, 清华伯克利深圳学院 (博士在读)

邮箱: hsinglul@163.com

CSDN 主页:

<https://blog.csdn.net/HsinglukLiu?spm=1010.2135.3001.5113>

1.1 无人机与卡车联合配送 (Python+Gurobi)(The flying sidekick traveling salesman problem)

1.1.1 无人机配送概述

随着无人机技术的不断发展, 无人机在工业界的应用场景也日益多样化。近几年, 很多物流企业开始将目光瞄向无人机配送。比如亚马逊、DHL、京东、顺丰等。渐渐的, 这种配送模式得到了更大的扩展, 现如今, 很多企业都在使用无人配送的提法, 在这种无人配送的场景中, 无人配送小车、无人机是重要的组成部分。本文主要关注无人机配送。

无人机配送的模式多种多样。

- 第一种就是在城市中选择若干个无人机起飞点, 在这个点的无人机从改点出发, 去配送周围的客户。
- 另一种是无人机和卡车联合配送的模式。这种模式也有很多不同的操作方法。
 1. 卡车司机带着无人机一起去配送 (flying sidekick TSP, FDTSP), 到一些比较难服务的点, 就让无人机去配送那个点, 然后自己去下一个客户点。无人机配送完后, 卡车司机再遥控无人机跟卡车司机会合。
 2. 卡车司机不带无人机 (parallel drone scheduling TSP, PDTSP)。我们分配任务的时候, 决策哪些任务交给无人机去服务, 哪些任务交给卡车去服务, 然后各自规划

自己的服务顺序，最终把所有顾客都服务完。期间，卡车和无人机没有交互。

- 当然还有其他服务方式。这里不做展开。

1.1.2 文献笔记

本文主要是发表在 Transportation Research Part C: Emerging Technologies 上的关于无人机和卡车的协同配送的文章的模型部分的代码复现 (Murray and Chu 2015)。文章题目为 The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery (参考文献 [1])。该问题简称 FSTSP。

论文摘要

Once limited to the military domain, unmanned aerial vehicles are now poised to gain widespread adoption in the commercial sector. One such application is to deploy these aircraft, also known as drones, for last-mile delivery in logistics operations. While significant research efforts are underway to improve the technology required to enable delivery by drone, less attention has been focused on the operational challenges associated with leveraging this technology. This paper provides two mathematical programming models aimed at optimal routing and scheduling of unmanned aircraft, and delivery trucks, in this new paradigm of parcel delivery. In particular, a unique variant of the classical vehicle routing problem is introduced, motivated by a scenario in which an unmanned aerial vehicle works in collaboration with a traditional delivery truck to distribute parcels. We present mixed integer linear programming formulations for two delivery-by-drone problems, along with two simple, yet effective, heuristic solution approaches to solve problems of practical size. Solutions to these problems will facilitate the adoption of unmanned aircraft for last-mile delivery. Such a delivery system is expected to provide faster receipt of customer orders at less cost to the distributor and with reduced environmental impacts. A numerical analysis demonstrates the effectiveness of the heuristics and investigates the tradeoffs between using drones with faster flight speeds versus longer endurance.

论文主要图表摘录

下面我们讲论文中的一些重要的图摘录下来，帮助读者快速理解文章的内容。首先就是亚马逊和 DHL 的无人机携带包裹的场景。



(a) Amazon's *Prime Air* UAV
(source: amazon.com)



(b) DHL's *Parcelcopter* (source: dhl.com)

Fig. 1. UAVs under evaluation for small parcel delivery.

无人机与卡车联合运输 - 联合但无交互的模式

下面几幅图中:

绿色方框: 在无人机飞行里程范围内的客户点, 这些点都可以被无人机服务;

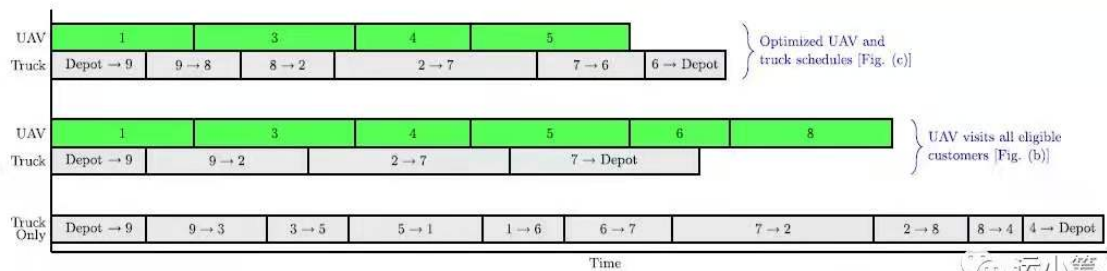
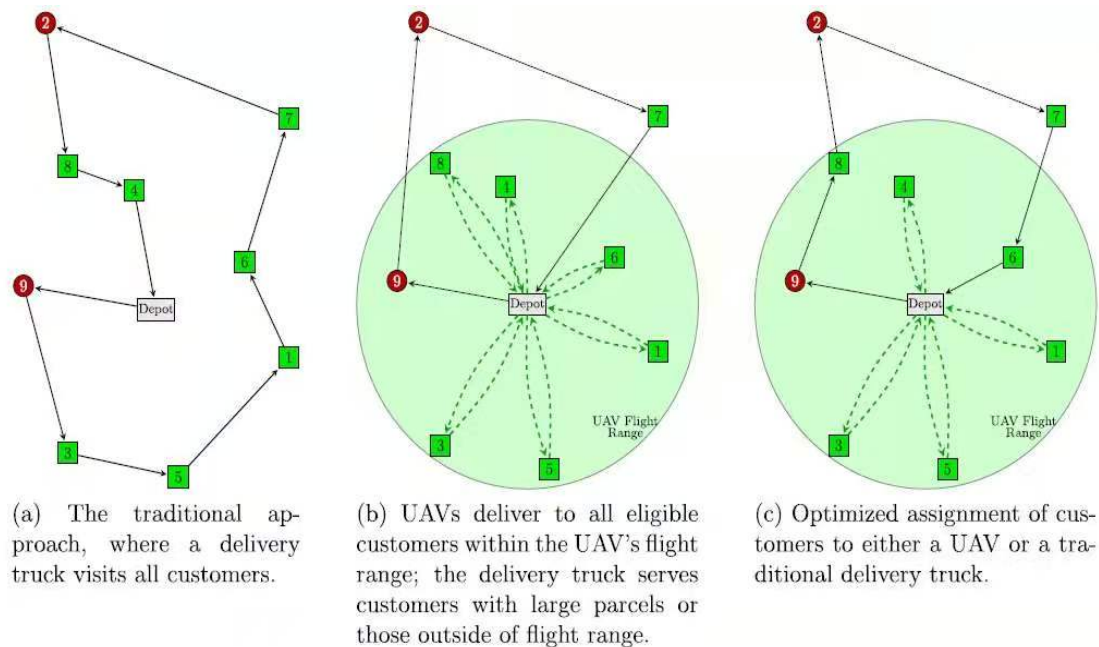
红色圆点: 在无人机里程范围之外的客户点, 这些点只能被卡车服务, 不能被无人机服务。

图 a 是传统的配送模式, 卡车按照访问顺序, 从出发点依次服务完所有的顾客点。

图 b 是卡车和无人机联合配送的场景, 但是卡车和无人既没有互动。可以看到, 顾客 ‘2’ 和顾客 ‘9’ 是无法被无人机服务的点, 而其余的点都是可以无人机服务的。一个可行解就是:

- 卡车的服务顺序 depot \rightarrow 9 \rightarrow 2 \rightarrow 7 \rightarrow depot
- 无人机的服务点为: 1、3、4、5、6、7、8。每个点无人机都往返依次完成服务。

图 c 是经过优化的最优配送方案。

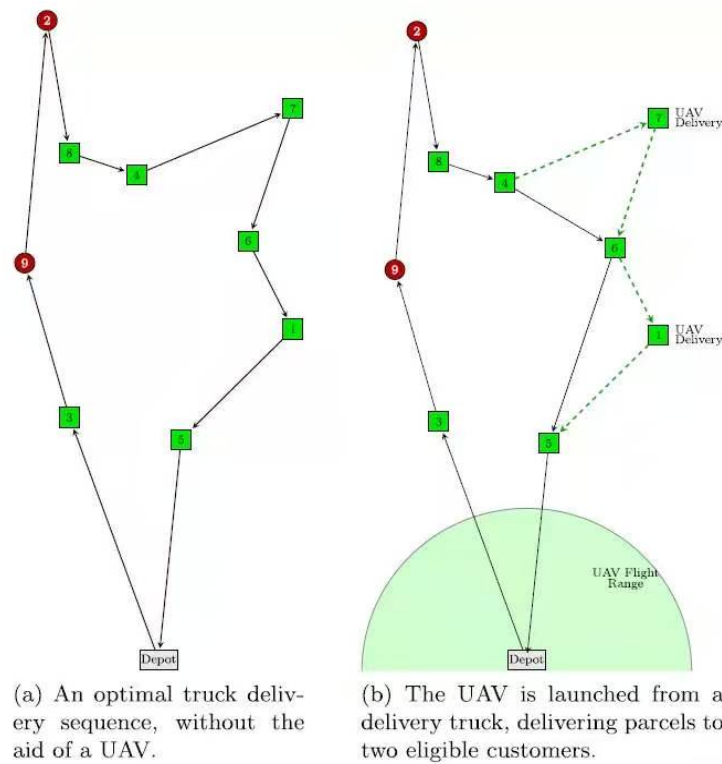


(d) A comparison of delivery schedules for the three systems depicted above.

无人机与卡车联合运输 - 联合有交互的模式

图 a 是卡车单独运输的最优解。

图 b 是无人机和卡车联合配送有交互的最优解。可以看到，卡车在配送到客户点 4 的时候，放飞了无人机，让无人机去服务客户点 7，之后又在客户点 6 放飞了无人机去服务客户点 1。分别在客户点 6 和客户点 5 回收无人机。(当然是同一架无人机)



1.1.3 无人机与卡车联合运输 (FSTSP) 的数学模型

其实就是上面讲过的第 1 种模式。我们再重复一遍。

3. The flying sidekick TSP

The FSTSP considers a set of c customers, each of whom must be served exactly once by either a driver-operated delivery truck or an unmanned aircraft operating in coordination with the truck. Because some customer requests may be infeasible to fulfill by the UAV (e.g., parcels that exceed the UAV's payload capacity, parcels requiring a signature, or customer locations not amenable to safely landing the UAV), these customers must be served by the truck only.

The truck and UAV must depart from, and return to, a single depot (distribution center) exactly once. The two vehicles may depart (or return) either in tandem or independently; while traveling in tandem the UAV is transported by the truck, thus conserving battery power.

卡车司机带着无人机一起去配送 (flying sidekick TSP, FDTSP), 到一些比较难服务的点, 就让无人机去配送那个点, 然后自己去下一个客户点。无人机配送完后, 卡车司机再遥控无

人机跟卡车司机会合。

这里最主要的决策变量有两个：

- x_{ij} : ‘0-1’变量。表示卡车是否从点 i 直接运行到点 j ;
- y_{ijk} : ‘0-1’变量。如果无人机从点 i 被发射，去服务客户点 j ，然后在客户点 k 被回收，则 $y_{ijk} = 1$ ，否则为 0。这里论文中强调，收发、访问点互不相同，也就是 i, j, k 互不相同。
- t_i : 到达顾客点 i 的时间;
- u_i : 为了消除子环路，也可以看做是第 i 个客户点的访问顺序，也就是他是第几个被访问的;

重要参数:

1. 无人机的单次发射时间;
2. 无人机的回收时间;

这些时间可以自己设置。另外，无人机的飞行速度、卡车的行驶速度等，都可以自己进行设置测试。

目标函数: 最小化服务完所有顾客的时间。

接下来，我们贴上该问题的数学模型。数学模型的具体解释请读者自行读论文，这里我们就不在赘述。

$$\min t_{c+1} \quad (1.1.1)$$

$$\text{s.t.} \quad \sum_{\substack{i \in N_0 \\ i \neq j}} x_{ij} + \sum_{i \in N_0} \sum_{\substack{k \in N_+ \\ i \neq j}} y_{ijk} = 1, \quad \forall j \in C \quad (1.1.2)$$

$$\sum_{j \in N_+} x_{0j} = 1, \quad (1.1.3)$$

$$\sum_{i \in N_0} x_{i,c+1} = 1, \quad (1.1.4)$$

$$u_i - u_j + 1 \leq (c + 2)(1 - x_{ij}), \quad \forall i \in C, j \in \{N_+ : j \neq i\} \quad (1.1.5)$$

$$\sum_{i \in N_0} x_{ij} = \sum_{\substack{k \in N_+ \\ i \neq j}} x_{jk}, \quad \forall j \in C \quad (1.1.6)$$

$$\sum_{j \in C} \sum_{\substack{k \in N_t \\ j \neq i(j,j) \in P}} y_{ijk} \leq 1, \quad \forall i \in N_0 \quad (1.1.7)$$

$$\sum_{\substack{i \in N_0 \\ i \neq k}} \sum_{\substack{j \in C \\ i \neq i(j,k) \in P}} y_{ijk} \leq 1, \quad \forall k \in N_+ \quad (1.1.8)$$

$$2y_{ijk} \leq \sum_{\substack{h \in N_0 \\ h \neq i}} x_{hi} + \sum_{\substack{l \in C \\ l \neq k}} x_{lk},$$

$$\forall i \in C, j \in \{C : j \neq i\}, k \in \{N_+ : \langle i, j, k \rangle \in P\} \quad (1.1.9)$$

$$y_{0jk} \leq \sum_{\substack{h \in N_0 \\ h \neq k}} x_{hk}, \quad \forall j \in C, k \in \{N_+ : \langle 0, j, k \rangle \in P\} \quad (1.1.10)$$

$$u_k - u_i \geq 1 - (c + 2) \left(1 - \sum_{j \in C} y_{ijk} \right),$$

$$\forall i \in C, k \in \{N_+ : k \neq i\} \quad (1.1.11)$$

$$t'_i \geq t_i - M \left(1 - \sum_{j \in C} \sum_{\substack{k \in N_+ \\ j \neq i}} y_{ijk} \right), \quad \forall i \in C \quad (1.1.12)$$

$$t'_i \leq t_i + M \left(1 - \sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{k \in N_+ \\ \langle i, j \rangle \in P}} y_{ijk} \right), \quad \forall i \in C \quad (1.1.13)$$

$$t'_k \geq t_k - M \left(1 - \sum_{i \in N_0} \sum_{\substack{j \in C \\ i \neq k}} y_{ijk} \right), \quad \forall k \in N_+ \quad (1.1.14)$$

$$t'_k \leq t_k + M \left(1 - \sum_{\substack{i \in N_0 \\ i \neq k(i, k) \in P}} y_{ijk} \right), \quad \forall k \in N_+ \quad (1.1.15)$$

$$t_k \geq t_h + \tau_{hk} + s_L \left(\sum_{l \in C} \sum_{\substack{m \in N_+ \\ l \neq k}} y_{klm} \right) + s_R \left(\sum_{i \in N_0} \sum_{j \in C} y_{ijk} \right) - M(1 - x_{hk}),$$

$$\forall h \in N_0, k \in \{N_+ : k \neq h\} \quad (1.1.16)$$

$$t'_j \geq t'_i + \tau'_{ij} - M \left(1 - \sum_{\substack{k \in N_+ \\ \langle i, j, k \rangle \in P}} y_{ijk} \right) \quad \forall j \in C', i \in \{N_0 : i \neq j\} \quad (1.1.17)$$

$$t'_k \geq t'_j + \tau'_{jk} + s_R - M \left(1 - \sum_{\substack{i \in N_0 \\ \langle i, j, k \rangle \in P}} y_{ijk} \right)$$

$$\forall j \in C', k \in \{N_+ : k \neq j\} \quad (1.1.18)$$

$$t'_k - (t'_j - \tau'_{ij}) \leq e + M(1 - y_{ijk})$$

$$\forall k \in N_+, j \in \{C : j \neq k\}, i \in \{N_0 : \langle i, j, k \rangle \in P\} \quad (1.1.19)$$

$$u_i - u_j \geq 1 - (c + 2)p_{ij} \forall i \in C, j \in \{C : j \neq i\} \quad (1.1.20)$$

$$u_i - u_j \leq -1 + (c + 2)(1 - p_{ij}) \forall i \in C, j \in \{C : j \neq i\} \quad (1.1.21)$$

$$p_{ij} + p_{ji} = 1 \quad \forall i \in C, j \in \{C : j \neq i\} \quad (1.1.22)$$

$$t'_l \geq t'_k - M(3 - \sum_{j \in C} y_{ijk} - \sum_{m \in C} \sum_{\substack{n \in N_+ \\ \{i,j,k\} \in P}} y_{lmn} - p_{il}) \quad (1.1.23)$$

$$t_0 = 0, \quad (1.1.24)$$

$$t'_0 = 0, \quad (1.1.25)$$

$$p_{0j} = 1, \quad \forall j \in C \quad (1.1.26)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in N_0, j \in \{N_+ : j \neq i\} \quad (1.1.27)$$

$$y_{ijk} \in \{0, 1\}, \quad \forall i \in N_0, j \in \{C : j \neq i\}, k \in \{N_+ : \langle i, j, k \rangle \in P\} \quad (1.1.28)$$

$$1 \leq u_i \leq c + 2, \quad \forall i \in N_+ \quad (1.1.29)$$

$$t_i \geq 0, \quad \forall i \in N \quad (1.1.30)$$

$$t'_i \geq 0 \quad \forall i \in N \quad (1.1.31)$$

$$p_{ij} \in \{0, 1\}, \quad \forall i \in N_0, j \in \{C : j \neq i\} \quad (1.1.32)$$

1.1.4 Python 调用 Gurobi 求解 FSTSP

这里我们贴上 ‘Python’ 调用 ‘Gurobi’ 求解 ‘FSTSP’ 的完整代码。代码中包含了可视化最优解的部分，方便读者查看最优解的情况。

```

Code
1  # -*- coding: utf-8 -*-
2  from __future__ import print_function
3  from gurobipy import *
4  import re
5  import math
6  # from test.pickletester import BigmemPickleTests
7  import matplotlib.pyplot as plt
8  import numpy
9  import pandas as pd
10
11 class Data:
12     customerNum = 0
13     nodeNum = 0
14     range = 0
15     lunchingTime = 0
16     recoverTime = 0
17     cor_X = []
18     cor_Y = []

```

```

19     demand      = []
20     serviceTime = []
21     readyTime   = []
22     dueTime     = []
23     disMatrix    = [[]] # 读取数据
24
25     # function to read data from .txt files
26     def readData(data, path, customerNum):
27         data.customerNum = customerNum
28         data.nodeNum = customerNum + 2
29         f = open(path, 'r')
30         lines = f.readlines()
31         count = 0
32         # read the info
33         for line in lines:
34             count = count + 1
35             if(count == 2):
36                 line = line[:-1]
37                 str = re.split(r" +", line)
38                 data.range = float(str[0])
39             elif(count == 5):
40                 line = line[:-1]
41                 str = re.split(r" +", line)
42                 data.lunchingTime = float(str[0])
43                 data.recoverTime = float(str[1])
44             elif(count >= 9 and count <= 9 + customerNum): # (count >= 9 and count <= 9 + customerNum)
45                 line = line[:-1]
46                 str = re.split(r" +", line)
47                 data.cor_X.append(float(str[2]))
48                 data.cor_Y.append(float(str[3]))
49                 data.demand.append(float(str[4]))
50                 data.readyTime.append(float(str[5]))
51                 data.dueTime.append(float(str[6]))
52                 data.serviceTime.append(float(str[7]))
53
54         data.cor_X.append(data.cor_X[0])
55         data.cor_Y.append(data.cor_Y[0])
56         data.demand.append(data.demand[0])
57         data.readyTime.append(data.readyTime[0])
58         data.dueTime.append(data.dueTime[0])
59         data.serviceTime.append(data.serviceTime[0])
60
61
62         # compute the distance matrix
63         data.disMatrix = [[0] * data.nodeNum for p in range(data.nodeNum)] # 初始化距离矩阵的维度, 防
        ↪ 止浅拷贝
64         # data.disMatrix = [[0] * nodeNum] * nodeNum 这个是浅拷贝, 容易重复
65         for i in range(0, data.nodeNum):
66             for j in range(0, data.nodeNum):
67                 temp = (data.cor_X[i] - data.cor_X[j])**2 + (data.cor_Y[i] - data.cor_Y[j])**2

```

```

68         data.disMatrix[i][j] = math.sqrt(temp)
69     #         if(i == j):
70     #             data.disMatrix[i][j] = 0
71     #     print("%6.2f" % (math.sqrt(temp)), end = " ")
72     temp = 0
73
74     return data
75
76
77 def printData(data, customerNum):
78     print(" 下面打印数据\n")
79     print("UAV range = %4d" % data.range)
80     print("UAV lunching time = %4d" % data.lunchingTime)
81     print("UAV recover time = %4d" % data.recoverTime)
82     for i in range(len(data.demand)):
83         print('{0}\t{1}\t{2}\t{3}'.format(data.demand[i], data.readyTime[i], data.dueTime[i],
84         ↪ data.serviceTime[i]))
85
86     print("-----距离矩阵-----\n")
87     for i in range(data.nodeNum):
88         for j in range(data.nodeNum):
89             #print("%d  %d" % (i, j))
90             print("%6.2f" % (data.disMatrix[i][j]), end = " ")
91         print()
92
93 class Solution:
94     ObjVal = 0
95     X = [[]]
96     Y = [[]]
97     U = []
98     P = []
99     T = []
100     Tt = []
101     route_Truck = []
102     route_UAV = []
103
104     def __init__(self):
105         solution = Solution()
106         # X_ij
107         solution.X = [[] for i in range(data.nodeNum)] for j in range(data.nodeNum)]
108         # Y_ijk
109         solution.Y = [[] for k in range(data.nodeNum)] for j in range(data.nodeNum)] for i in
110         ↪ range(data.nodeNum)]
111         # U_i
112         solution.U = [[] for i in range(data.nodeNum)]
113         # P_ij
114         solution.P = [[] for j in range(data.nodeNum)] for i in range(data.nodeNum)]
115         # T_i, T_i'
116         solution.T = [[] for i in range(data.nodeNum)]
117         solution.Tt = [[] for i in range(data.nodeNum)]

```



```

116 #         return solution
117
118 def getSolution(self, data, model):
119     solution = Solution()
120     solution.ObjVal = model.ObjVal
121     # X_ij
122     solution.X = [[0] * data.nodeNum for j in range(data.nodeNum)]
123     # Y_ijk
124     solution.Y = [[[0] * data.nodeNum for j in range(data.nodeNum)] for i in
        ↪ range(data.nodeNum)]
125     # U_i
126     solution.U = [[0] for i in range(data.nodeNum)]
127     # P_ij
128     solution.P = [[[0] for j in range(data.nodeNum)] for i in range(data.nodeNum)]
129     # T_i, T_i'
130     solution.T = [[0] for i in range(data.nodeNum)]
131     solution.Tt = [[0] for i in range(data.nodeNum)]
132
133     a = U[0].x
134     for m in model.getVars():
135         str = re.split(r"_", m.VarName)
136         if(str[0] == "X" and m.x == 1):
137             solution.X[int(str[1])][int(str[2])] = m.x
138             print(str, end = "")
139             print(" = %d" % m.x)
140         elif(str[0] == "Y" and m.x == 1):
141             solution.Y[int(str[1])][int(str[2])][int(str[3])] = m.x
142         elif(str[0] == "U" and m.x > 0):
143             solution.U[int(str[1])] = m.x
144         elif(str[0] == "T" and m.x > 0):
145             solution.T[int(str[1])] = m.x
146         elif(str[0] == "Tt" and m.x > 0):
147             solution.Tt[int(str[1])] = m.x
148         elif(str[0] == "P" and m.x > 0):
149             solution.P[int(str[1])][int(str[2])] = m.x
150
151     # get the route of truck and UAV
152     j = 0
153     for i in range(data.nodeNum):
154         i = j # note that the variable is whether is a local variable or a global variable
155         # print("i = %d, j = %d" % (i, j), end = " ")
156         for j in range(data.nodeNum):
157             if(solution.X[i][j] == 1):
158                 solution.route_Truck.append(i)
159                 print(" %d -" % i, end = " ")
160                 # print(" i = %d, j = %d" % (i, j))
161                 break
162     print(" 0")
163     solution.route_Truck.append(0)
164

```

```

165     print("\n\n -----Route of UAV ----- ")
166     count = 0
167     for i in range(data.nodeNum):
168         for j in range(data.nodeNum):
169             for k in range(data.nodeNum):
170                 if(solution.Y[i][j][k] == 1):
171                     count = count + 1
172                     #print("UAV %d : %d - %d - %d" % (count, i, j, k))
173                     temp = [i, j, k]
174                     solution.route_UAV.append(temp)
175
176     for i in range(len(solution.route_Truck)):
177         print(" %d " % solution.route_Truck[i], end = " ")
178     print()
179
180     print("\n\n -----Route of UAV ----- ")
181     for i in range(len(solution.route_UAV)):
182         for j in range(len(solution.route_UAV[0])):
183             print("UAV %d : %d - %d - %d" % (i, solution.route_UAV[i][0],
184                 ↪ solution.route_UAV[i][1], solution.route_UAV[i][2]))
185
186     # print(solution.route_UAV)
187
188     return solution
189
190 # reading data
191 data = Data()
192 # path = r'C:\Users\hsingluLiu\eclipse-workspace\PythonCallGurobi_Applications\FSTSP\c101.txt'
193 path = 'c101.txt'
194
195 customerNum = 10
196 readData(data, path, customerNum)
197 printData(data, customerNum)
198
199
200 # =====build the model=====
201 big_M = 10000
202 # construct the model object
203 model = Model("FSTSP")
204
205 # Initialize variables
206 # create variables: Muilti-dimension vector: from inner to outer
207 # X_ij
208 X = [[[ for i in range(data.nodeNum)] for j in range(data.nodeNum)]
209
210 # Y_ijk
211 Y = [[[[ for k in range(data.nodeNum)] for j in range(data.nodeNum)] for i in range(data.nodeNum)]
212
213 # U_i

```

```

214 U = [[] for i in range(data.nodeNum)]
215
216 # P_ij
217 P = [[] for j in range(data.nodeNum)] for i in range(data.nodeNum)]
218
219 # T_i, T_i'
220 T = [[] for i in range(data.nodeNum)]
221 Tt = [[] for i in range(data.nodeNum)]
222
223 for i in range(data.nodeNum):
224     name1 = 'U_' + str(i)
225     name2 = 'T_' + str(i)
226     name3 = 'Tt_' + str(i)
227     U[i] = model.addVar(0, data.nodeNum, vtype = GRB.CONTINUOUS, name = name1)
228     T[i] = model.addVar(0, big_M, vtype = GRB.CONTINUOUS, name = name2)
229     Tt[i] = model.addVar(0, big_M, vtype = GRB.CONTINUOUS, name = name3)
230     for j in range(data.nodeNum):
231         name4 = 'X_' + str(i) + "_" + str(j)
232         name5 = 'P_' + str(i) + "_" + str(j)
233         X[i][j] = model.addVar(0, 1, vtype = GRB.BINARY, name = name4)
234         P[i][j] = model.addVar(0, 1, vtype = GRB.BINARY, name = name5)
235         for k in range(data.nodeNum):
236             name6 = 'Y_' + str(i) + "_" + str(j) + "_" + str(k)
237             Y[i][j][k] = model.addVar(0, 1, vtype = GRB.BINARY, name = name6)
238
239 # Add constraints
240 # create the objective expression(1)
241 obj = LinExpr(0)
242
243 # add the objective function into the model
244 model.setObjective(T[data.nodeNum - 1], GRB.MINIMIZE)
245
246 # constraint (2)
247 for j in range(1, data.nodeNum - 1): # 这里需要注意, i 的取值范围, 否则可能会加入空约束
248     expr = LinExpr(0)
249     for i in range(0, data.nodeNum - 1): # i -- NO
250         if(i != j):
251             expr.addTerms(1, X[i][j])
252             for k in range(1, data.nodeNum): # k -- N+
253                 if(i != k and j != k):
254                     expr.addTerms(1, Y[i][j][k])
255
256     model.addConstr(expr == 1, "c1")
257     expr.clear()
258
259
260 # constraint (3)
261 expr = LinExpr(0)
262 for j in range(1, data.nodeNum):
263     expr.addTerms(1, X[0][j])

```

```

264 model.addConstr(expr == 1, "c2")
265 expr.clear()
266
267 # constraint (4)
268 expr = LinExpr(0)
269 for i in range(data.nodeNum - 1):
270     expr.addTerms(1, X[i][data.nodeNum - 1])
271 model.addConstr(expr == 1.0, "c3")
272 expr.clear()
273
274 # constraint (5)
275 for i in range(1, data.nodeNum - 1):
276     for j in range(1, data.nodeNum):
277         if(i != j):
278             model.addConstr(U[i] - U[j] + 1 <= big_M - big_M * X[i][j], 'c5')
279
280
281 # constraint (6)
282 for j in range(1, data.nodeNum - 1):
283     expr1 = LinExpr(0)
284     expr2 = LinExpr(0)
285     for i in range(0, data.nodeNum - 1):
286         if(j != i):
287             expr1.addTerms(1, X[i][j])
288
289     for k in range(1, data.nodeNum):
290         if(j != k):
291             expr2.addTerms(1, X[j][k])
292
293     model.addConstr(expr1 == expr2, "c6")
294     expr1.clear()
295     expr2.clear()
296
297 # constraint (7)
298 for i in range(data.nodeNum - 1):
299     expr = LinExpr(0)
300     for j in range(1, data.nodeNum - 1):
301         if(i != j):
302             for k in range(1, data.nodeNum):
303                 if(i != k and j != k):
304                     expr.addTerms(1, Y[i][j][k])
305     model.addConstr(expr <= 1, 'c7')
306     expr.clear()
307
308 # constraint (8)
309 for k in range(1, data.nodeNum):
310     expr = LinExpr(0)
311     for i in range(0, data.nodeNum - 1):
312         if(i != k):
313             for j in range(1, data.nodeNum - 1):

```

```

314         if(j != i and j != k):
315             expr.addTerms(1, Y[i][j][k])
316     model.addConstr(expr <= 1, 'c8')
317     expr.clear()
318
319     # constraint (9)
320     for i in range(1, data.nodeNum - 1):
321         for j in range(1, data.nodeNum):
322             for k in range(1, data.nodeNum):
323                 if(i != j and i != k and j != k):
324                     expr1 = LinExpr(0)
325                     expr2 = LinExpr(0)
326                     for h in range(data.nodeNum - 1):
327                         if(h != i):
328                             expr1.addTerms(1, X[h][i])
329                     for l in range(1, data.nodeNum - 1):
330                         if(l != k):
331                             expr2.addTerms(1, X[l][k])
332                     model.addConstr(2 * Y[i][j][k] <= expr1 + expr2, "c9")
333                     expr1.clear()
334                     expr2.clear()
335
336     # constraint (10)
337     for j in range(1, data.nodeNum - 1):
338         for k in range(1, data.nodeNum):
339             if(j != k):
340                 expr = LinExpr(0)
341                 for h in range(1, data.nodeNum - 1):
342                     expr.addTerms(1, X[h][k])
343                 model.addConstr(Y[0][j][k] <= expr, "c10")
344                 expr.clear()
345
346     # constraint (11)
347     for i in range(1, data.nodeNum - 1):
348         for k in range(1, data.nodeNum):
349             if(k != i):
350                 expr = LinExpr(0)
351                 for j in range(1, data.nodeNum - 1):
352                     if(i != j and j != k):
353                         expr.addTerms(big_M, Y[i][j][k])
354                 model.addConstr(U[k] - U[i] >= 1 - big_M + expr, "c11")
355                 expr.clear()
356
357     # constraint (12)
358     for i in range(1, data.nodeNum - 1):
359         expr = LinExpr(0)
360         for j in range(1, data.nodeNum - 1):
361             for k in range(1, data.nodeNum):
362                 if(j != i and i != k and j != k):
363                     expr.addTerms(big_M, Y[i][j][k])

```

```

364     model.addConstr(Tt[i] >= T[i] - big_M + expr, "c12")
365     expr.clear()
366
367     # constraint (13)
368     for i in range(1, data.nodeNum - 1):
369         expr = LinExpr(0)
370         for j in range(1, data.nodeNum - 1):
371             for k in range(1, data.nodeNum):
372                 if(j != i and i != k and j != k):
373                     expr.addTerms(big_M, Y[i][j][k])
374             model.addConstr(Tt[i] <= T[i] + big_M - expr, "c13")
375             expr.clear()
376
377     # constraint (14)
378     for k in range(1, data.nodeNum):
379         expr = LinExpr(0)
380         for i in range(0, data.nodeNum - 1):
381             for j in range(1, data.nodeNum - 1):
382                 if(j != i and i != k and j != k):
383                     expr.addTerms(big_M, Y[i][j][k])
384             model.addConstr(Tt[k] >= T[k] - big_M + expr, "c14")
385             expr.clear()
386
387     # constraint (15)
388     for k in range(1, data.nodeNum):
389         expr = LinExpr(0)
390         for i in range(0, data.nodeNum - 1):
391             for j in range(1, data.nodeNum - 1):
392                 if(j != i and i != k and j != k):
393                     expr.addTerms(big_M, Y[i][j][k])
394             model.addConstr(Tt[k] <= T[k] + big_M - expr, "c15")
395             expr.clear()
396
397     # constraint (16)
398     for h in range(data.nodeNum - 1):
399         for k in range(1, data.nodeNum):
400             if(h != k):
401                 expr1 = LinExpr(0)
402                 expr2 = LinExpr(0)
403                 for l in range(1, data.nodeNum - 1):
404                     for m in range(1, data.nodeNum):
405                         if(k != l and k != m and l != m):
406                             expr1.addTerms(data.lunchingTime, Y[k][l][m])
407
408                 for i in range(data.nodeNum - 1):
409                     for j in range(1, data.nodeNum - 1):
410                         if(i != j and i != k and j != k):
411                             expr2.addTerms(data.recoverTime, Y[i][j][k])
412             model.addConstr(T[k] >= T[h] + data.disMatrix[h][k] + expr1 + expr2 - big_M + big_M *
413                 ↪ X[h][k], "c16")

```

```

413         expr1.clear()
414         expr2.clear()
415
416     # constraint (17)
417     for j in range(1, data.nodeNum - 1):
418         for i in range(data.nodeNum - 1):
419             if(i != j):
420                 expr = LinExpr(0)
421                 for k in range(1, data.nodeNum):
422                     if(i != k and j != k):
423                         expr.addTerms(big_M, Y[i][j][k])
424                 model.addConstr(Tt[j] >= Tt[i] + data.disMatrix[i][j] - big_M + expr, "c17")
425                 expr.clear()
426
427     # constraint (18)
428     for j in range(1, data.nodeNum - 1):
429         for k in range(1, data.nodeNum):
430             if(k != j):
431                 expr = LinExpr(0)
432                 for i in range(data.nodeNum - 1):
433                     if(i != k and i != j):
434                         expr.addTerms(big_M, Y[i][j][k])
435                 model.addConstr(Tt[k] >= Tt[j] + data.disMatrix[j][k] + data.recoverTime - big_M + expr,
436                               ↪ "c18")
437                 expr.clear()
438
439     # constraint (19)
440     for k in range(1, data.nodeNum):
441         for j in range(1, data.nodeNum - 1):
442             for i in range(data.nodeNum - 1):
443                 if(i != j and i != k and j != k):
444                     model.addConstr(Tt[k] - Tt[j] + data.disMatrix[i][j] <= data.range + big_M - big_M *
445                                     ↪ Y[i][j][k], "c19")
446
447     # constraint (20)
448     for i in range(1, data.nodeNum - 1):
449         for j in range(1, data.nodeNum - 1):
450             if(i != j):
451                 model.addConstr(U[i] - U[j] >= 1 - big_M * P[i][j], "c20")
452
453     # constraint (21)
454     for i in range(1, data.nodeNum - 1):
455         for j in range(1, data.nodeNum - 1):
456             if(i != j):
457                 model.addConstr(U[i] - U[j] <= -1 + big_M - big_M * P[i][j], "c21")
458
459     # constraint (22)
460     for i in range(1, data.nodeNum - 1):
461         for j in range(1, data.nodeNum - 1):
462             if(i != j):

```

```

461         model.addConstr(P[i][j] + P[j][i] == 1, "c22")
462
463     # constraint (23)
464     for i in range(data.nodeNum - 1):
465         for k in range(1, data.nodeNum):
466             for l in range(1, data.nodeNum - 1):
467                 if(k != i and l != i and l != k):
468                     expr1 = LinExpr(0)
469                     expr2 = LinExpr(0)
470                     for j in range(1, data.nodeNum - 1):
471                         if(k != j and i != j):
472                             expr1.addTerms(big_M, Y[i][j][k])
473                     for m in range(1, data.nodeNum - 1):
474                         for n in range(1, data.nodeNum):
475                             if(l != m and l != n and m != n):
476                                 expr2.addTerms(big_M, Y[l][m][n])
477                     model.addConstr(Tt[l] >= Tt[k] - 3*big_M + expr1 + expr2 + big_M * P[i][l], "c23")
478                     expr1.clear()
479                     expr2.clear()
480
481     # constraint (24)
482     model.addConstr(T[0] == 0, "c24")
483
484     # constraint (25)
485     model.addConstr(Tt[0] == 0, "c25")
486
487     # constraint (26)
488     for j in range(1, data.nodeNum - 1):
489         model.addConstr(P[0][j] == 1, "c26")
490
491     # constraint (27)
492     for i in range(data.nodeNum):
493         for j in range(data.nodeNum):
494             if(i == j):
495                 model.addConstr(X[i][j] == 0, "c27")
496         for k in range(data.nodeNum):
497             if(i == j or i == k or k == j):
498                 model.addConstr(Y[i][j][k] == 0, "c28")
499
500
501     # solve the problem
502     model.write('a.lp')
503     model.Params.timelimit = 3600
504     model.optimize()
505
506
507     # get the solution info
508     solution = Solution()
509     solution = solution.getSolution(data, model)
510     print("\n\n\n\n----optimal value----")

```



```

511 print("Obj: %g" % solution.ObjVal)
512 print("\n\n -----Route of truck-----")
513 # print("Truck: ", end = " ")
514 j = 0
515 for i in range(data.nodeNum):
516     i = j # note that the variable is whether is a local variable or a global variable
517     # print("i = %d, j = %d" % (i, j), end = " ")
518     for j in range(data.nodeNum):
519         if(solution.X[i][j] == 1):
520             print(" %d -" % i, end = " ")
521             # print(" i = %d, j = %d" % (i, j))
522             break
523 print(" 0")
524
525 print("\n\n -----Route of UAV ----- ")
526 count = 0
527 for i in range(data.nodeNum):
528     for j in range(data.nodeNum):
529         for k in range(data.nodeNum):
530             if(solution.Y[i][j][k] == 1):
531                 count = count + 1
532                 print("UAV %d : %d - %d - %d" % (count, i, j, k))
533
534
535 # draw the route graph
536 # draw all the nodes first
537 # data1 = Data()
538 # readData(data1, path, 100)
539 fig = plt.figure(figsize=(15,10))
540 font_dict = {'family': 'Arial', # serif
541             'style': 'normal', # 'italic',
542             'weight': 'normal',
543             'color': 'darkred',
544             'size': 30,
545             }
546 font_dict2 = {'family': 'Arial', # serif
547             'style': 'normal', # 'italic',
548             'weight': 'normal',
549             'color': 'darkred',
550             'size': 24,
551             }
552 plt.xlabel('x', font_dict)
553 plt.ylabel('y', font_dict)
554 plt.title('Optimal Solution for FSTSP (5 customers)', font_dict)
555 plt.xticks(fontsize=22)
556 plt.yticks(fontsize=22) # plt.yticks(fontsize=30)
557 plt.grid(True, color='r', linestyle='-', linewidth=2)
558
559
560 '''

```

```

561     marker='o'
562     marker=', '
563     marker='.'
564     marker=(9, 3, 30)
565     marker='+'
566     marker='v'
567     marker='^'
568     marker='<'
569     marker='>'
570     marker='1'
571     marker='2'
572     marker='3'
573     red         blue         green
574     '''
575     plt.scatter(data.cor_X[0], data.cor_Y[0], c='blue', alpha=1, marker=',', linewidths=5,
576 ↪     label='depot')
577
578     plt.scatter(data.cor_X[1:-1], data.cor_Y[1:-1], c='magenta', alpha=1, marker='o', linewidths=5,
579 ↪     label='customer') # c='red' 定义为红色, alpha 是透明度, marker 是画的样式
580
581     # draw the route
582     for i in range(data.nodeNum):
583         for j in range(data.nodeNum):
584             if(solution.X[i][j] == 1):
585                 x = [data.cor_X[i], data.cor_X[j]]
586                 y = [data.cor_Y[i], data.cor_Y[j]]
587                 plt.plot(x, y, 'b', linewidth = 3)
588
589                 # plt.text(data.cor_X[i]-1, data.cor_Y[i], str(i), fontsize=15, color = 'black')
590                 # plt.text(coverage50index*0.98, 4, coverage50index, fontsize=10, color = 'red')
591                 plt.text(data.cor_X[i]-0.2, data.cor_Y[i], str(i), fontdict = font_dict2)
592
593     for i in range(data.nodeNum):
594         for j in range(data.nodeNum):
595             for k in range(data.nodeNum):
596                 if(solution.Y[i][j][k] == 1):
597                     x = [data.cor_X[i], data.cor_X[j], data.cor_X[k]]
598                     y = [data.cor_Y[i], data.cor_Y[j], data.cor_Y[k]]
599                     plt.plot(x, y, 'r--', linewidth = 3)
600                     plt.text(data.cor_X[j]-0.2, data.cor_Y[j], str(j), fontdict = font_dict2)
601                     #plt.plot(x, y, 'r--', label = "UAV", linewidth = 3)
602
603     # plt.grid(True)
604     plt.grid(False)
605     plt.legend(loc='best', fontsize = 20)
606     plt.show()

```

1.1.5 数值实验

算例

算例我们就采用 Solomon 的 VRP benchmark 的算例数据，只是抽出其中的一些点作为我们算例的点。这里我们需要对 Solomon benchmark 的算例做一些小改动，就是设置无人机的飞行里程 RANGE，以及设置无人机的发射时间和回收时间 LUNCTING 和 RECOVER。

如下，我们设置 RANGE=100, LUNCTING=1, RECOVER=1 其余的数据不做改动。设置无人机的里程为 100 修改虽然感觉不合理，但是这样在小规模的情况下能够让无人机被发射出去服务一个顾客点。

举 Solomon benchmark 的 c101 为例，修改后的算理数据如下：

Code							
1	RANGE						
2	100						
3							
4	LUNCTING		RECOVER				
5	1	1					
6							
7	CUSTOMER						
8	CUST NO.	XCOORD.	YCOORD.	DEMAND	READY TIME	DUE DATE	SERVICE TIME
9	0	40	50	0	0	1236	0
10	1	45	68	10	912	967	90
11	2	45	70	30	825	870	90
12	3	42	66	10	65	146	90
13	4	42	68	10	727	782	90
14	5	42	65	10	15	67	90
15	6	40	69	20	621	702	90
16	7	40	66	20	170	225	90
17	8	38	68	20	255	324	90
18	9	38	70	10	534	605	90
19	10	35	66	10	357	410	90
20	11	35	69	10	448	505	90
21	12	25	85	20	652	721	90
22	13	22	75	30	30	92	90
23	14	22	85	10	567	620	90
24	15	20	80	40	384	429	90
25	16	20	85	40	475	528	90
26	17	18	75	20	99	148	90
27	18	15	75	20	179	254	90
28	19	15	80	10	278	345	90
29	20	30	50	10	10	73	90
30	21	30	52	20	914	965	90
31	22	28	52	20	812	883	90
32	23	28	55	10	732	777	90
33	24	25	50	10	65	144	90
34	25	25	52	40	169	224	90
35	26	25	55	10	622	701	90

36	27	23	52	10	261	316	90
37	28	23	55	20	546	593	90
38	29	20	50	10	358	405	90
39	30	20	55	10	449	504	90
40	31	10	35	20	200	237	90
41	32	10	40	30	31	100	90
42	33	8	40	40	87	158	90
43	34	8	45	20	751	816	90
44	35	5	35	10	283	344	90
45	36	5	45	10	665	716	90
46	37	2	40	20	383	434	90
47	38	0	40	30	479	522	90
48	39	0	45	20	567	624	90
49	40	35	30	10	264	321	90
50	41	35	32	10	166	235	90
51	42	33	32	20	68	149	90
52	43	33	35	10	16	80	90
53	44	32	30	10	359	412	90
54	45	30	30	10	541	600	90
55	46	30	32	30	448	509	90
56	47	30	35	10	1054	1127	90
57	48	28	30	10	632	693	90
58	49	28	35	10	1001	1066	90
59	50	26	32	10	815	880	90
60	51	25	30	10	725	786	90
61	52	25	35	10	912	969	90
62	53	44	5	20	286	347	90
63	54	42	10	40	186	257	90
64	55	42	15	10	95	158	90
65	56	40	5	30	385	436	90
66	57	40	15	40	35	87	90
67	58	38	5	30	471	534	90
68	59	38	15	10	651	740	90
69	60	35	5	20	562	629	90
70	61	50	30	10	531	610	90
71	62	50	35	20	262	317	90
72	63	50	40	50	171	218	90
73	64	48	30	10	632	693	90
74	65	48	40	10	76	129	90
75	66	47	35	10	826	875	90
76	67	47	40	10	12	77	90
77	68	45	30	10	734	777	90
78	69	45	35	10	916	969	90
79	70	95	30	30	387	456	90
80	71	95	35	20	293	360	90
81	72	53	30	10	450	505	90
82	73	92	30	10	478	551	90
83	74	53	35	50	353	412	90
84	75	45	65	20	997	1068	90
85	76	90	35	10	203	260	90

86	77	88	30	10	574	643	90
87	78	88	35	20	109	170	90
88	79	87	30	10	668	731	90
89	80	85	25	10	769	820	90
90	81	85	35	30	47	124	90
91	82	75	55	20	369	420	90
92	83	72	55	10	265	338	90
93	84	70	58	20	458	523	90
94	85	68	60	30	555	612	90
95	86	66	55	10	173	238	90
96	87	65	55	20	85	144	90
97	88	65	60	30	645	708	90
98	89	63	58	10	737	802	90
99	90	60	55	10	20	84	90
100	91	60	60	10	836	889	90
101	92	67	85	20	368	441	90
102	93	65	85	40	475	518	90
103	94	65	82	10	285	336	90
104	95	62	80	30	196	239	90
105	96	60	80	10	95	156	90
106	97	60	85	30	561	622	90
107	98	58	75	20	30	84	90
108	99	55	80	10	743	820	90
109	100	55	85	20	647	726	90

1.1.6 结果展示与可视化

由于该问题比较难求解，我们先来一个小算例。

设置 `nodeNum = 5`

注意，设置有 5 个顾客点的时候，点 0 和点 6 就代表 depot。其余的以此类推。

我们设置 `nodeNum = 5`

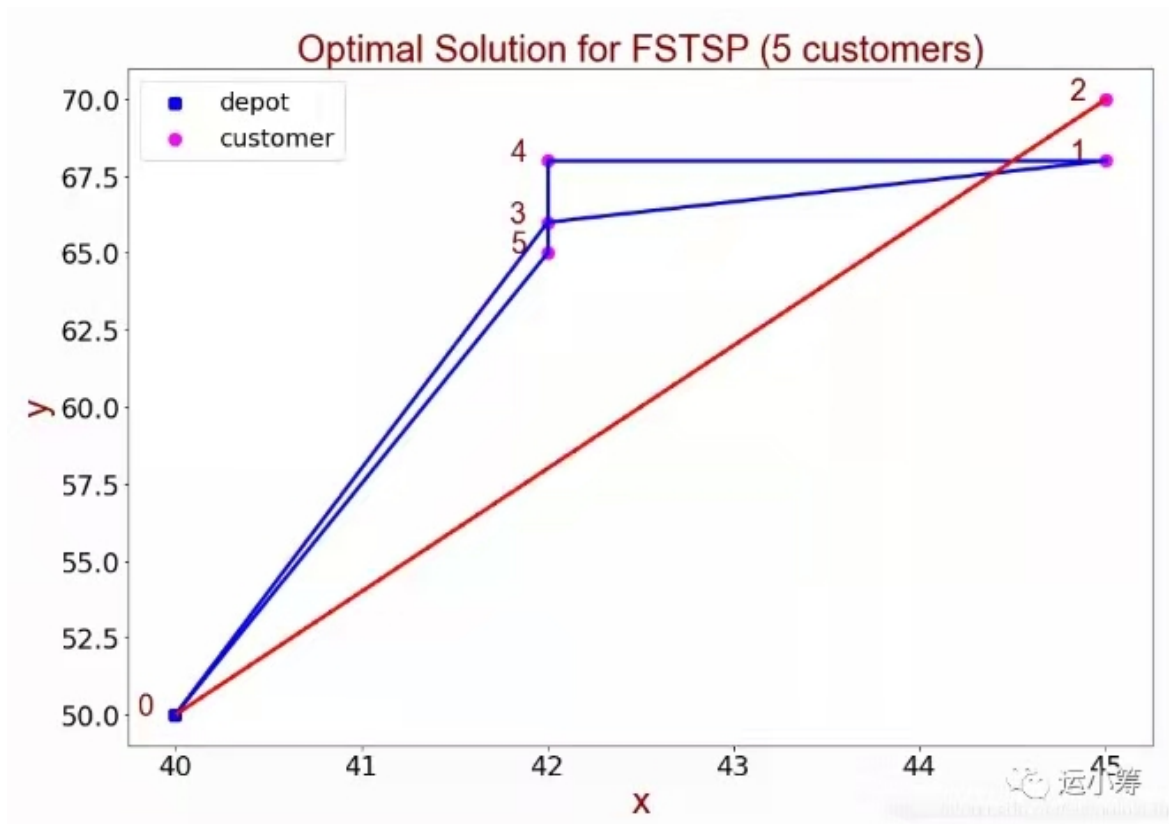
```

Code
1  Explored 3528 nodes (35235 simplex iterations) in 1.15 seconds
2  Thread count was 8 (of 8 available processors)
3
4  Solution count 4: 42.2311 42.2311 43.3318 47.0835
5
6  Optimal solution found (tolerance 1.00e-04)
7  Best objective 4.223105625185e+01, best bound 4.223105625185e+01, gap 0.0000%
8  ['X', '0', '5'] = 1
9  ['X', '1', '3'] = 1
10 ['X', '3', '6'] = 1
11 ['X', '4', '1'] = 1
12 ['X', '5', '4'] = 1
13 0 - 5 - 4 - 1 - 3 - 0

```

```
14
15
16 -----Route of UAV -----
17 0   5   4   1   3   0
18
19
20 -----Route of UAV -----
21 UAV 0 : 0 - 2 - 6
22 UAV 0 : 0 - 2 - 6
23 UAV 0 : 0 - 2 - 6
24
25
26
27
28 -----optimal value-----
29 Obj: 42.2311
30
31
32 -----Route of truck-----
33 0 - 5 - 4 - 1 - 3 - 0
34
35
36 -----Route of UAV -----
37 UAV 1 : 0 - 2 - 6
```

结果为:



蓝色的路径：代表卡车的路径；

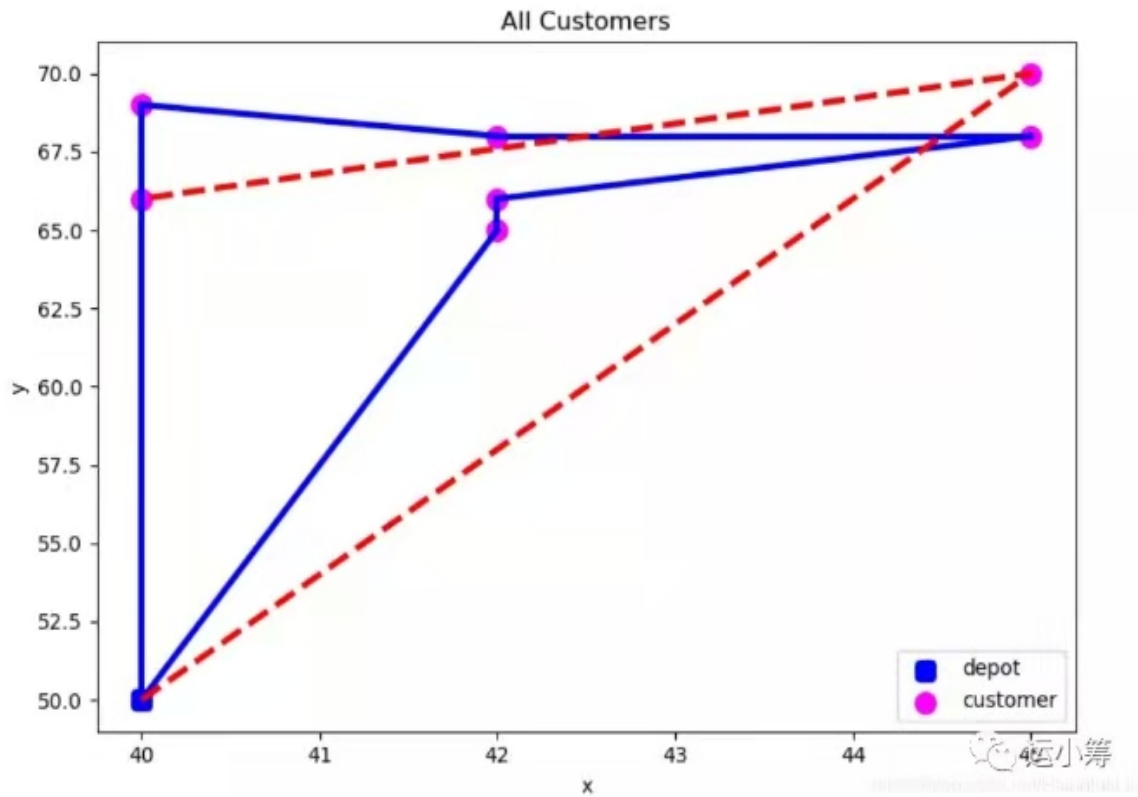
红色的路径：代表无人机的路径。

可以看到：

- 卡车的服务顺序 $0 \rightarrow 5 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 0$
- 无人机的服务任务为： $0 \rightarrow 2 \rightarrow 0$ 最终服务完所有顾客的时间为 42.23。

设置 `nodeNum = 7`

运行结果如下：



设置 `nodeNum = 8`

`nodeNum` 设置成 8。求解时间为 4m 26s。

```

Code
1  Explored 163568 nodes (11107937 simplex iterations) in 265.29 seconds
2  Thread count was 16 (of 16 available processors)
3
4  Solution count 10: 47.0389 47.0389 47.0389 ... 49.1723
5
6  Optimal solution found (tolerance 1.00e-04)
7  Best objective 4.703886030563e+01, best bound 4.703886030563e+01, gap 0.0000%
8  ['X', '0', '7'] = 1
9  ['X', '1', '3'] = 1
10 ['X', '3', '5'] = 1
11 ['X', '4', '1'] = 1
12 ['X', '5', '9'] = 1
13 ['X', '6', '4'] = 1
14 ['X', '7', '8'] = 1
15 ['X', '8', '6'] = 1
16 0 - 7 - 8 - 6 - 4 - 1 - 3 - 5 - 0
17
18
19 -----Route of UAV -----

```

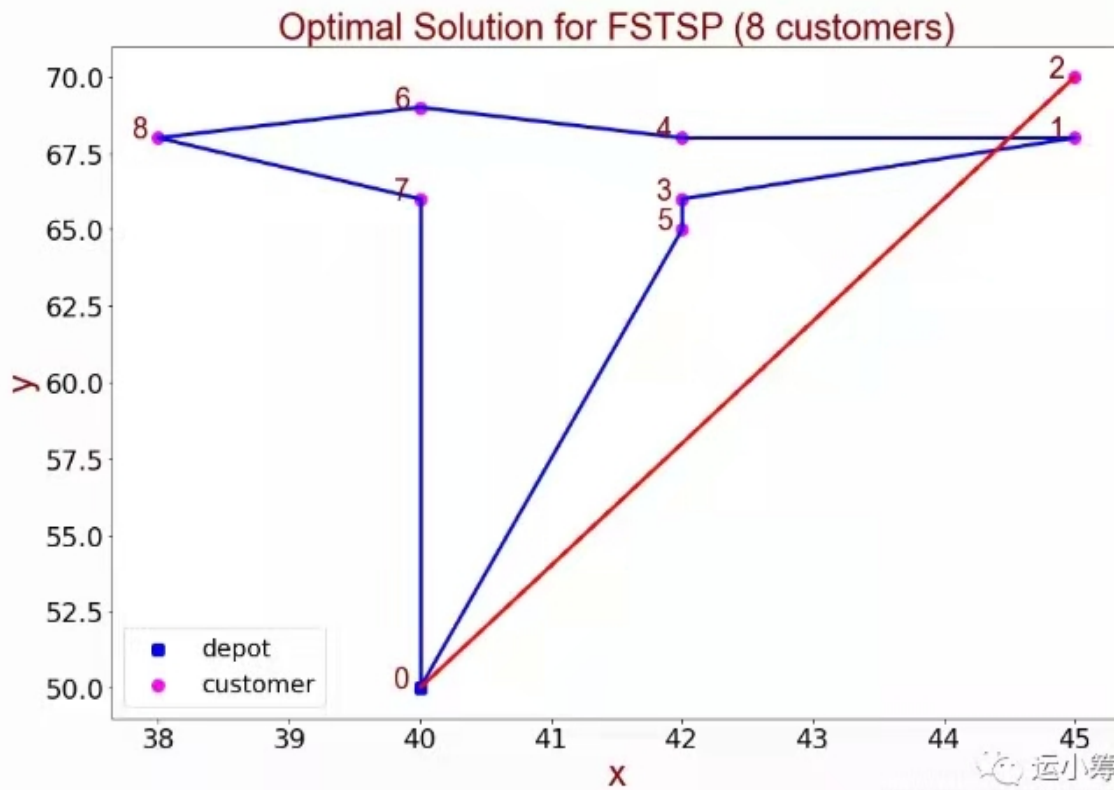


```

20 0 7 8 6 4 1 3 5 0
21
22
23 -----Route of UAV -----
24 UAV 0 : 0 - 2 - 9
25 UAV 0 : 0 - 2 - 9
26 UAV 0 : 0 - 2 - 9
27
28
29
30
31 -----optimal value-----
32 Obj: 47.0389
33
34
35 -----Route of truck-----
36 0 - 7 - 8 - 6 - 4 - 1 - 3 - 5 - 0
37
38
39 -----Route of UAV -----
40 UAV 1 : 0 - 2 - 9

```

结果为:



设置 `nodeNum = 10`

运行 3600 秒，还是没有找到可行解。如下图所示。

```

913449 66176 0.00000 54 36 51.38701 0.00000 100% 111 3537s
914737 66212 infeasible 58 51.38701 0.00000 100% 111 3542s
915935 66198 infeasible 64 51.38701 0.00000 100% 111 3547s
916768 66234 infeasible 59 51.38701 0.00000 100% 111 3553s
917937 67660 6.15385 48 36 51.38701 0.00000 100% 112 3558s
924808 67674 39.14170 70 28 51.38701 0.00000 100% 111 3562s
925309 67676 infeasible 53 51.38701 0.00000 100% 111 3565s

926138 67706 infeasible 55 51.38701 0.00000 100% 111 3571s
927121 67702 0.18614 58 56 51.38701 0.00000 100% 111 3577s
927670 67717 16.12548 61 35 51.38701 0.00000 100% 111 3580s
928964 67709 16.76589 53 36 51.38701 0.00000 100% 111 3587s
929643 67726 15.13673 53 42 51.38701 0.00000 100% 111 3591s
930402 67785 0.00000 40 25 51.38701 0.00000 100% 111 3595s
931646 67795 16.00028 60 42 51.38701 0.00000 100% 111 3600s

```

Cutting planes:

```

Learned: 15
Gomory: 6
Implied bound: 20
Projected implied bound: 7
Clique: 8
MIR: 11
StrongCG: 2
Flow cover: 176
Inf proof: 8
Zero half: 4
RLT: 5
Relax-and-lift: 15

```

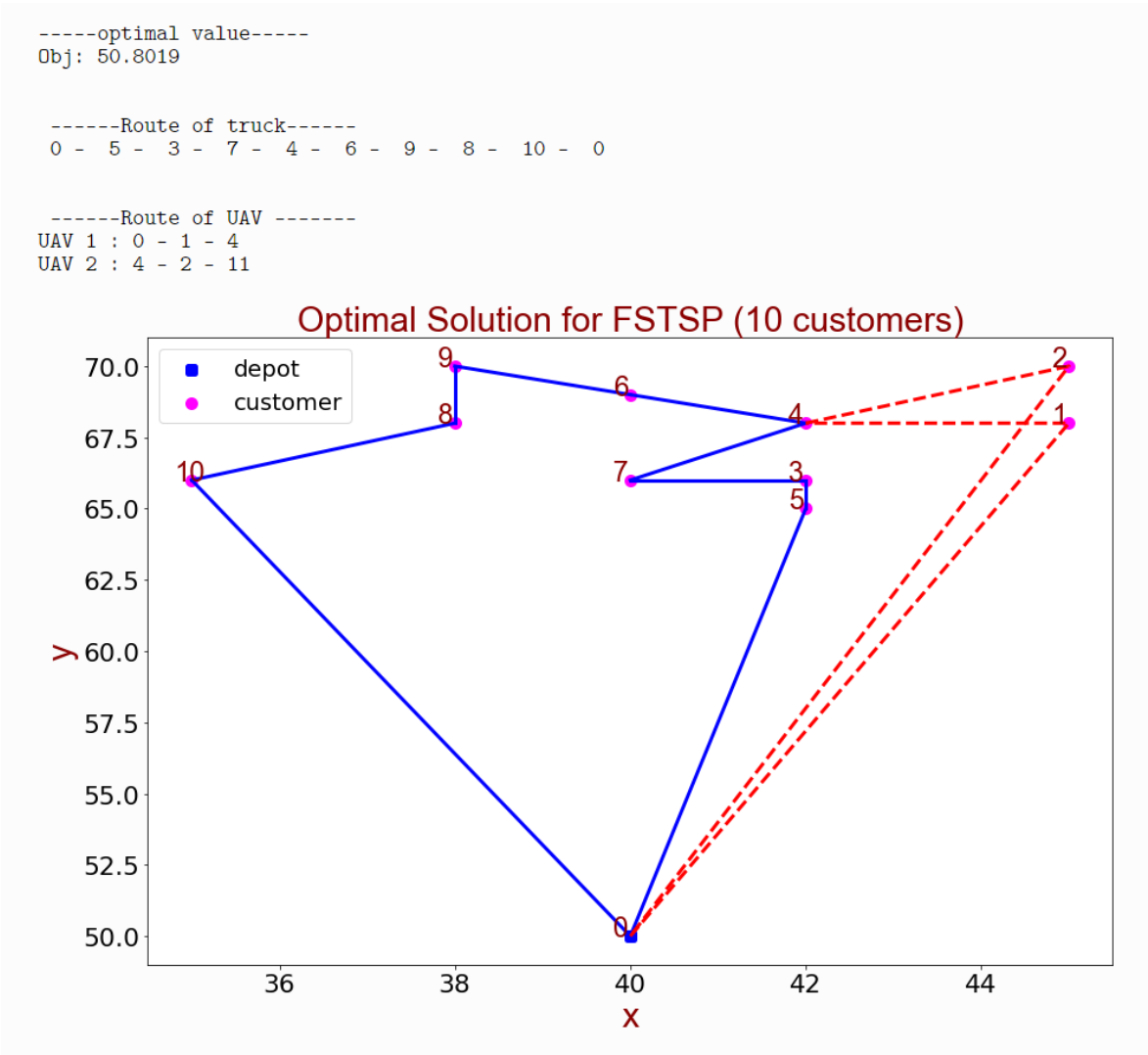
```

Explored 931886 nodes (103808224 simplex iterations) in 3600.06 seconds
Thread count was 16 (of 16 available processors)

```

运小筹

最后，我们也得到了 10 个点的最优解，如下图：



1.1.7 拓展

FSTSP 求解起来比较难，本推文介绍的这篇，作者在论文中提出了一些启发式的求解方法。如果后续有机会，我们可以继续拓展。

当然，还有一些论文提出了动态规划的方法。见文献Agatz et al. 2018和Bouman et al. 2018。

这里我们也附上几篇其他相关文献 (Ham 2018, Chang and Lee 2018)。

Chapter 2

2020-11-13: 元启发式算法

作者：段淇耀，清华大学，清华伯克利深圳学院 (博士在读)

邮箱：duanqy71@163.com

CSDN 主页：

<https://blog.csdn.net/DCXY71?spm=1001.2100.3001.5113>

2.1 禁忌搜索 (Tabu Search) 解决 TSP 问题 (Python 代码实现)

2.1.1 Tabu Search 基本概念

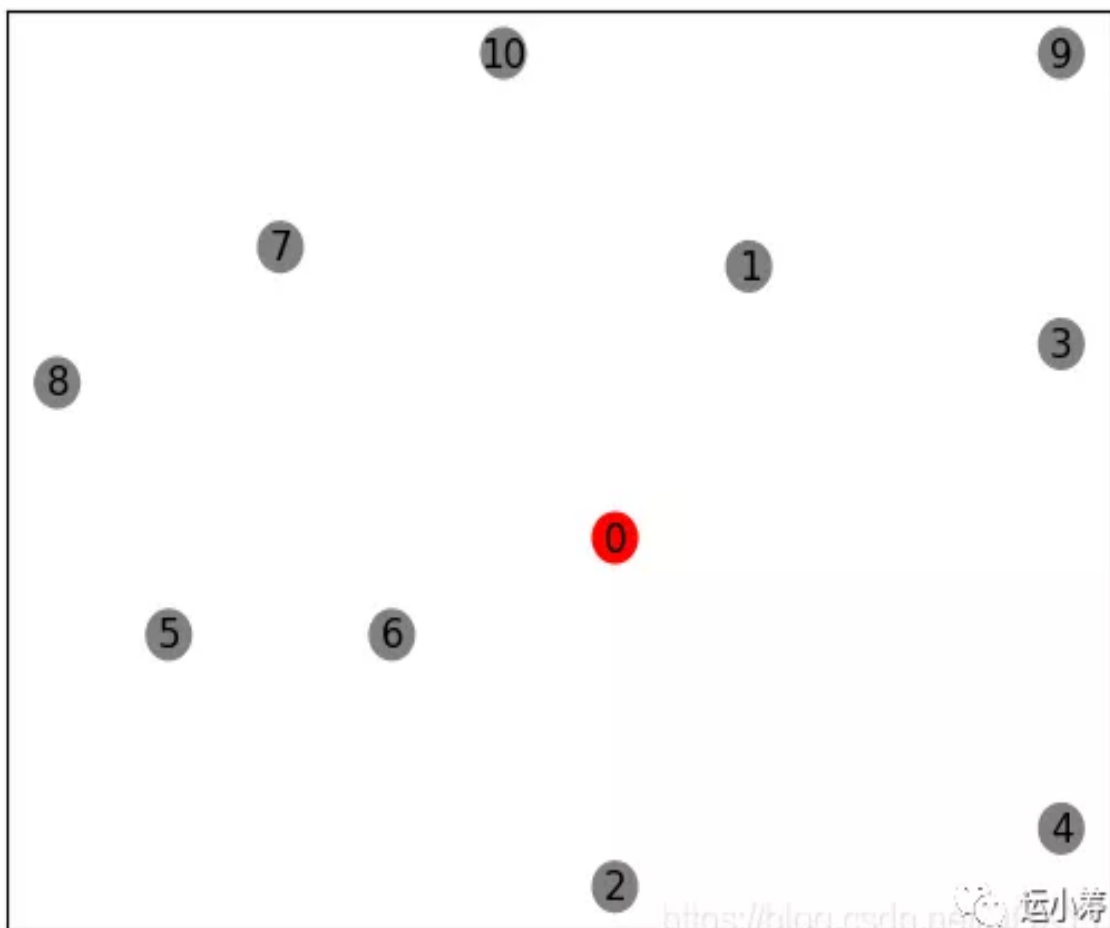
禁忌搜索 (Tabu Search, TS, 以下简称 TS) 是一种基于邻域搜索策略的元启发式算法, 由 Fred W. Glover 在 1986 提出 (Glover 1986), 并于 1989 构建 Glover 1989, Glover 1990。应用于各类组合优化问题。

禁忌搜索的核心思想：从一个初始解出发，按一系列规则对邻域进行探索，对已搜索过的途径和局部最优解进行记录（建立禁忌表，tabu list），并在进一步的迭代搜索中尽量避开这些对象（不是绝对禁止循环），减少重复搜索，从而保证对不同的有效搜索途径的探索，提高搜索效率。

禁忌搜索涉及到的几个概念。邻域 (neighborhood)、禁忌表 (tabu list)、禁忌长度 (tabu length)、候选解 (candidate)、藐视准则 (aspiration criterion) (有的也叫特赦规则)

2.1.2 Tabu Search 算法实现细节

以 TSP (Traveling Salesman Problem) 问题为例，0 号为起始点，需要依次拜访 1,2,...,10 号，最终返回 0 号，求最短路径。已知各个点坐标位置，各个点的位置如下图所示。解的形式表示为一个列表，如：route=[1,2,3,4,5,6,7,8,9,10].



（这里用的是 Solomon 标准算例，R101.txt，文件可留言获取。当然，也可以随机生成一系列坐标点。）

邻域 (neighborhood)：邻域一般定义为由给定转化规则对给定的问题域上每结点进行转化所得到的问题域上结点的集合。那么在这里即 1-10 号点的排列组合。

候选解 (candidate)：从邻域中选择若干个目标值或评价值最佳的邻居作为候选解。候选解集合的生成规则一定程度上决定了搜索的方向和邻域大小，十分关键。候选集过大增加计算内存和计算时间，过小容易过早陷入局部最优。候选集的选择一般由邻域中的邻居组成，可以选择所有邻居，也可以选择表现较好的邻居，还可以随机选择几个邻居。这里我们采用两两交换法则生成候选解集合。

禁忌表 (tabu list)：记录最优候选解和对应元素，这些元素在下次搜索时将不会被考虑。

禁忌长度 (tabu length)：禁忌表的最大长度。

藐视准则 (aspiration criterion)：禁忌搜索算法中，迭代的某一步会出现候选集的某一个元素被禁止搜索，但是若解禁该元素，则会使评价函数有所改善，因此我们需要设置一

个特赦规则，当满足该条件时该元素从禁忌表中跳出。

评价函数 (evaluation)：评价解的好坏。在这里即路径距离。

禁忌搜索算法核心问题

禁忌对象：禁掉谁？根据受禁对象的不同选择，可行解是一禁禁一个；还是一禁禁掉一大片。

主要对禁忌范围，及搜索范围有影响

禁忌长度：禁多久？禁掉的东西什么时候放出来？禁忌长度过短，会导致循环；禁忌长度过长，会导致计算时间过长

候选集：邻域中可行解的选取？候选集的大小，过大增加计算内存和计算时间，过小过早陷入局部最优。

python 代码实现

```

Code
1  from itertools import combinations
2  import os,sys,copy
3  import numpy as np
4  import time
5  from Datareader import Data
6  import matplotlib.pyplot as plt
7
8  class Tabu():
9      def __init__(self,disMatrix,max_iters=50,maxTabuSize=10):
10         """parameters definition"""
11         self.disMatrix = disMatrix
12         self.maxTabuSize = maxTabuSize
13         self.max_iters = max_iters
14         self.tabu_list=[]
15
16     def get_route_distance(self,route):
17         '''
18         Description: function to calculate total distance of a route. evaluate function.
19         parameters: route : list
20         return : total distance : float
21         '''
22         routes = [0] + route + [0]    # add the start and end point
23         total_distance = 0
24         for i,n in enumerate(routes):
25             if i != 0 :
26                 total_distance = total_distance + self.disMatrix[last_pos][n]
27                 last_pos = n
28         return total_distance
29
30     def exchange(self,s1,s2,arr):
31         """
32         function to Swap positions of two elements in an arr

```

```

33     Args: int,int,list
34           s1 : target 1
35           s2 : target 2
36           arr : target array
37     Duput: list
38           current_list : target array
39     """
40     current_list = copy.deepcopy(arr)
41     index1 , index2 = current_list.index(s1) , current_list.index(s2) # get index
42     current_list[index1], current_list[index2]= arr[index2] , arr[index1]
43     return current_list
44
45 def generate_initial_solution(self,num=10,mode='greedy'):
46     """
47     function to get the initial solution,there two different way to generate route_init.
48     Args:
49         num : int
50             the number of points
51         mode : string
52             "greedy" : advance step by choosing optimal one
53             "random" : randomly generate a series number
54     Duput: list
55         s_init : initial solution route_init
56     """
57     if mode == 'greedy':
58         route_init=[0]
59         for i in range(num):
60             best_distance = 10000000
61             for j in range(num+1):
62                 if self.disMatrix[i][j] < best_distance and j not in route_init:
63                     best_distance = self.disMatrix[i][j]
64                     best_candidate = j
65             route_init.append(best_candidate)
66             route_init.remove(0)
67
68     if mode == 'random':
69         route_init = np.arange(1,num+1) #init solution from 1 to num
70         np.random.shuffle(route_init) #shuffle the list randomly
71
72     return list(route_init)
73
74 def tabu_search(self,s_init):
75     """tabu search"""
76     s_best = s_init
77     bestCandidate = copy.deepcopy(s_best)
78     routes , temp_tabu = [] , [] # init
79     routes.append(s_best)
80     while(self.max_iters):
81         self.max_iters -= 1 # Number of iterations
82         neighbors = copy.deepcopy(s_best)

```

```

83         for s in combinations(neighbors, 2):
84             sCandidate = self.exchange(s[0],s[1],neighbors) # exchange number to generate
                        ↪ candidates
85             if s not in self.tabu_list and self.get_route_distance(sCandidate) <
                        ↪ self.get_route_distance(bestCandidate):
86                 bestCandidate = sCandidate
87                 temp_tabu = s
88             if self.get_route_distance(bestCandidate) < self.get_route_distance(s_best): # record
                        ↪ the best solution
89                 s_best = bestCandidate
90             if temp_tabu not in self.tabu_list:
91                 self.tabu_list.append(temp_tabu)
92             if len(self.tabu_list) > self.maxTabuSize :
93                 self.tabu_list.pop(0)
94             routes.append(bestCandidate)
95         return s_best, routes
96
97 if __name__ == "__main__":
98
99     data = Data()
100     data.read_data(path='R101.txt',customerNum=100,depotNum=1) # change the path
101
102     """ Tabu :
103         disMatrix : the distance matrix from 0 to X , 0 represents starting and stopping point.
104         for example: disMatrix = [[0,3,4,...
105                             1,0,5,...
106                             3,5,0,...]]
107         that means the distance from 0 to 0 is 0, from 0 to 1 is 3,... from 1 to 3 is 5....
108         max_iters : maximum iterations
109         maxTabuSize : maximum iterations
110     """
111     tsp = Tabu(disMatrix=data.disMatrix,max_iters=10,maxTabuSize=10)
112     # two different way to generate initial solution
113     # num : the number of points
114     s_init = tsp.generate_initial_solution(num=10,mode='greedy') # mode = "greedy" or "random"
115     print('init route : ' , s_init)
116     print('init distance : ' , tsp.get_route_distance(s_init))
117
118     start = time.time()
119     best_route , routes = tsp.tabu_search(s_init) # tabu search
120     end = time.time()
121
122     print('best route : ' , best_route)
123     print('best best_distance : ' , tsp.get_route_distance(best_route))
124     print('the time cost : ',end - start )
125
126     # plot the result changes with iterations
127     results=[]
128     for i in routes:
129         results.append(tsp.get_route_distance(i))

```



```

130 plt.plot(np.arange(len(results)) , results)
131 plt.show()
132 # plot the route
133 data.plot_route(best_route)

```

如使用 Solomon 标准算例，只需更改路径即可。如随机生成距离矩阵，只需要 `disMatrix=data.disMatrix`（改成你的距离矩阵）

Code

```

1 tsp = Tabu(disMatrix=data.disMatrix,max_iters=10,maxTabuSize=10)

```

`Datareader.py` 读取 Solomon 标准算例，生成距离矩阵，还有画图。如果随机生成距离矩阵那就不用不到了。

Code

```

1 import numpy as np
2 import networkx as nx
3 import matplotlib.pyplot as plt
4 import copy
5 import re
6 import math
7
8 class Data():
9     '''
10     the format of solomon dataset
11     '''
12     def __init__(self):
13         self.customerNum = 0 # the number of customers
14         self.nodeNum = 0 # the sum of customers and depots
15         self.vehicleNum = 0
16         self.capacity = 0
17         self.cor_X = []
18         self.cor_Y = []
19         self.demand = []
20         self.readyTime = []
21         self.dueTime = []
22         self.serviceTime = []
23         self.disMatrix = {}
24
25     def read_data(self, path, customerNum, depotNum):
26         '''
27         function to read solomon data from .txt files, notice that it must be solomon dataset
28         INPUT
29             @ data : class Data
30             @ path : Data path
31             @ customerNum : the number of customer
32         OutPut : none
33         '''

```

```

34     self.customerNum = customerNum
35     self.nodeNum = customerNum + depotNum
36     f = open(path, 'r')
37     lines = f.readlines()
38     count = 0
39     for line in lines:
40         count = count + 1
41         if(count == 5):
42             line = line[:-1].strip()
43             str = re.split(r" +", line)
44             self.vehicleNum = int(str[0])
45             self.capacity = float(str[1])
46         elif(count >= 10 and count <= 10 + customerNum):
47             line = line[:-1]
48             str = re.split(r" +", line)
49             self.cor_X.append(float(str[2]))
50             self.cor_Y.append(float(str[3]))
51             self.demand.append(float(str[4]))
52             self.readyTime.append(float(str[5]))
53             self.dueTime.append(float(str[6]))
54             self.serviceTime.append(float(str[7]))
55
56     # compute the distance matrix
57     self.disMatrix = {}
58     for i in range(0, self.nodeNum):
59         dis_temp={}
60         for j in range(0, self.nodeNum):
61             dis_temp[j] = int(math.hypot(self.cor_X[i] - self.cor_X[j],self.cor_Y[i] -
62             ↪ self.cor_Y[j]))
63         self.disMatrix[i] = dis_temp
64
65     def plot_nodes(self):
66         '''
67         Description: function to plot
68         '''
69         Graph = nx.DiGraph()
70         nodes_name = [str(x) for x in list(range(self.nodeNum))]
71         Graph.add_nodes_from(nodes_name)
72         cor_xy = np.array([self.cor_X,self.cor_Y]).T.astype(int)
73         pos_location = {nodes_name[i]:x for i,x in enumerate(cor_xy)}
74         nodes_color_dict = ['r'] + ['gray'] * (self.nodeNum-1)
75
76         nx.draw_networkx(Graph,pos_location,node_size=200,node_color=nodes_color_dict,labels=None)
77         plt.show(Graph)
78
79     def plot_route(self,route,color='k'):
80         Graph = nx.DiGraph()
81         nodes_name = [0]
82         cor_xy=[[self.cor_X[0] , self.cor_Y[0]]]

```

```

83     edge = []
84     edges = [[0,route[0]]]
85     for i in route :
86         nodes_name.append(i)
87         cor_xy.append([self.cor_X[i] , self.cor_Y[i]])
88         edge.append(i)
89         if len(edge) == 2 :
90             edges.append(copy.deepcopy(edge))
91             edge.pop(0)
92     edges.append([route[-1],0])
93
94     Graph.add_nodes_from(nodes_name)
95     Graph.add_edges_from(edges)
96
97     pos_location = {nodes_name[i]:x for i,x in enumerate(cor_xy)}
98     nodes_color_dict = ['r'] + ['gray'] * (len(route))
99
100     ↪ nx.draw_networkx(Graph,pos_location,node_size=200,node_color=nodes_color_dict,edge_color=color,
101     ↪ labels=None)
102     plt.show(Graph)

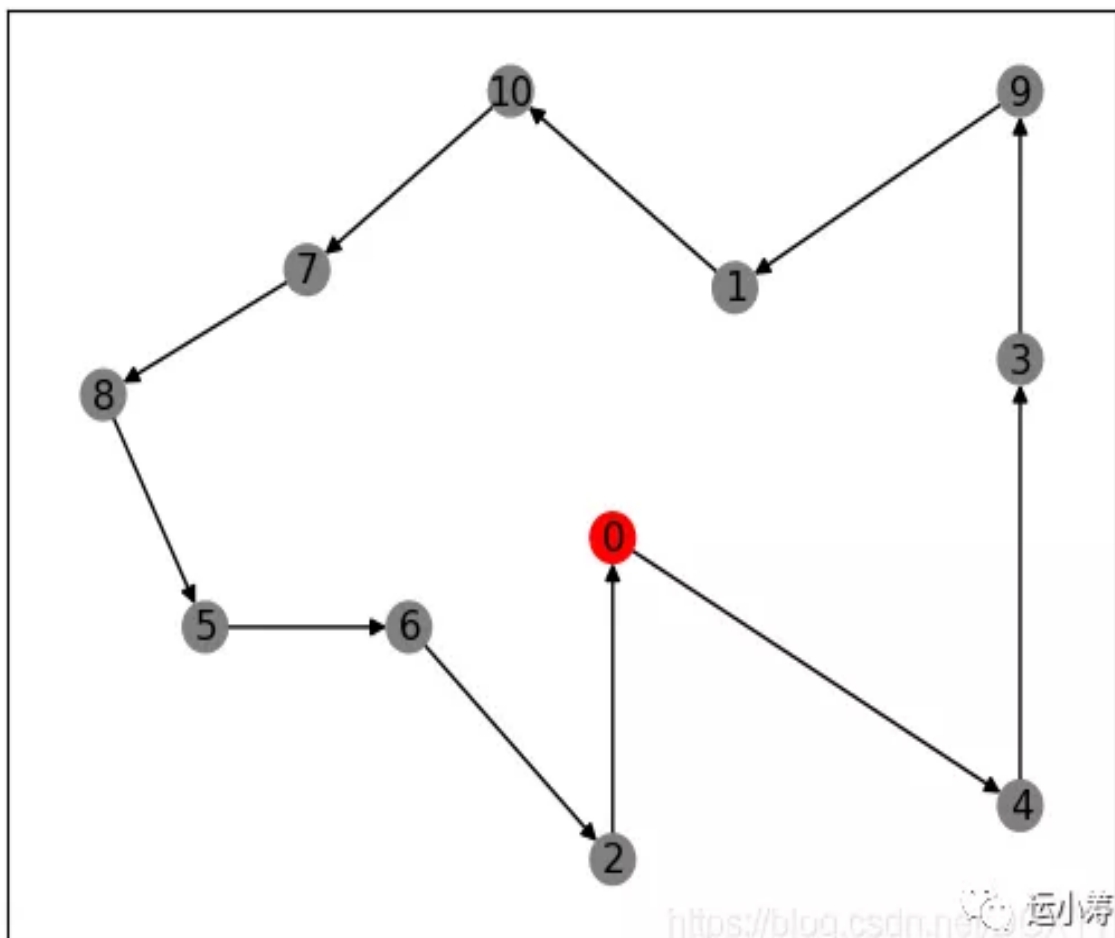
```

最后求解可以得到:

```

Code
1 solution=[4,3,9,1,10,7,8,5,6,2], total\_distance = 180.

```



2.1.3 问题与总结

对于中小规模 TSP 的求解，求解速度和解的质量都不错，禁忌表长度似乎对求解速度影响不大。但随着点数增加，求解规模增大，所用时间明显增加，解的质量出现明显下降，且解的质量与给定初始解密切相关。

1. 求解时间增加

这里我们采用的是 `combinations()` 函数，对列表元素两两组合，将会产生 C_n^2 种组合，当 n 变大时，组合数将非常可观，一般可采用随机组合的方式，随交换元素或是随机选择组合，没必要全部遍历。

Code

```

1      self.max_iters -= 1 # Number of iterations
2      neighbors = copy.deepcopy(s_best)
3      for s in combinations(neighbors, 2): # 这里将所有元素两两组合，若点数增加至 1000 点，
        ↳ 每次将循环 499500 次

```

```
4 | sCandidate = self.exchange(s[0],s[1],neighbors)
```

2. 随着规模增加，解的质量出现下降

我们的候选解集合是通过两两交换规则产生，其实还是针对局部邻域进行贪婪的搜索，搜索范围有限，虽然算法通用易实现，且容易理解，但搜索性能完全依赖于邻域结构和初解，随着规模的增加尤其会陷入局部极小而无法保证全局优化型。为了实现全局优化，可尝试结合其他算法思想：例如以可控性概率接受劣解来跳出局部极小，如模拟退火算法；扩大邻域搜索结构，如 TSP 的 2-opt 扩展到 k-opt。

3. 禁忌长度禁忌长度

禁忌长度是禁忌搜索非常关键的参数，其大小直接影响到整个算法的搜索进程和的解的质量。在这里我们设置禁忌长度为一常数，采用先进先出 (FIFO) 的形式，但当问题比较复杂，规模比较大时，可以考虑让禁忌长度动态的自适应变化。

4. 一些细节

设定合适的终止条件：（在这里我们设置了最大迭代次数）

(1) 设置最大迭代次数。

(2) 采用频率控制，当某个解、目标值或元素序列出现的频率高于给定的阈值，停止算法。

(3) 如果在给定的迭代次数内，历史最优值未发生变化，可以停止算法。

还需要注意的是：

- 不要仅仅局限于将解作为禁忌对象，禁忌对象可以是其他任何合理的元素。例如，在前面代码中，我们禁忌的对象是交换的元素 `tabu_list=[(2, 9), (6, 10), (5, 7), (10, 4), (1, 3)]`，但其实还可以将距离（价值）也作为禁忌的对象，当一个解的价值已经位于禁忌表，那么就强迫算法选择不同价值的解，从而达到跳出局部极值的目的。
- 在实际搜索中，可能出现所有候选解都处于禁止选中的状态，为了能够让算法能够正常继续下去，就需要引入特赦准则（在我们这个算例中不明显）。当走投无路的时候（候选集全部位于禁忌表）如果违反禁忌表可获得更优解，那么特赦准则允许候选集违反禁忌规则，对象可以重新被选中。从这个角度来看，这相当于加强了对某个较优局部的搜索，以希望发现更好的解。

Chapter 3

2020-11-16:TSP 中两种不同消除子环路的方法及 callback 实现

作者：刘兴禄，清华大学，清华伯克利深圳学院 (博士在读)

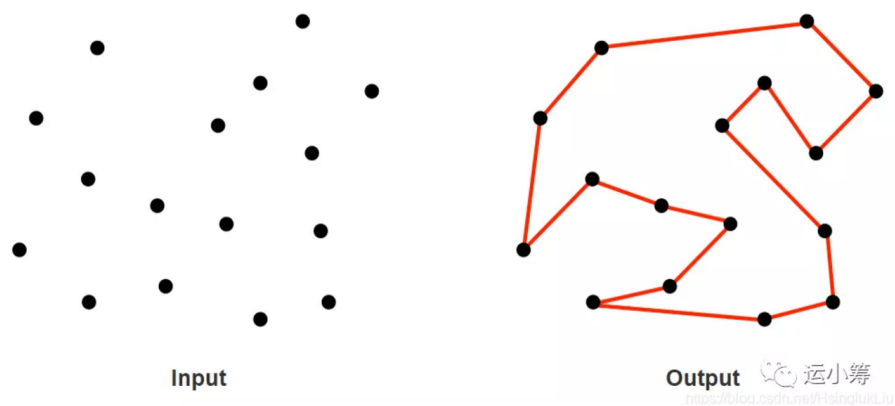
邮箱：hsinglul@163.com

CSDN 主页：

<https://blog.csdn.net/HsinglukLiu?spm=1010.2135.3001.5113>

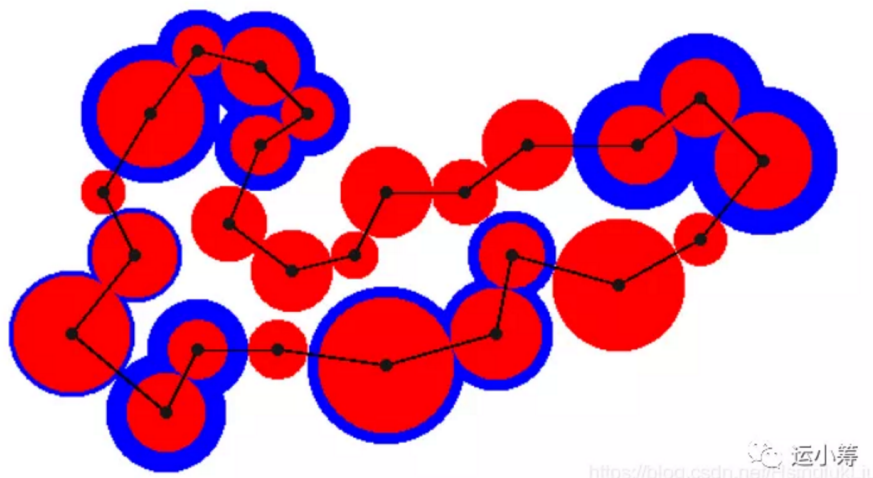
3.1 TSP 问题的一般模型

Traveling Salesman Problem(TSP)，中文名叫旅行商问题，货郎担问题 (前者更常见)。TSP 的描述如下：给定一系列的结点集合 V ($|V| = N$)，找到一条从该节点出发，依次不重复的经过所有其他节点，最终返回到出发点的最短路径。



图片来自 <http://algorist.com/problems/Traveling Salesman Problem.html>

上述例子中的节点可以广义化成一系列的区域，如下图。但是本质是一样的问题。



图片来自<http://www.math.uwaterloo.ca/tsp/methods/opt/subtour.html>

该问题是计算机领域和应用数学领域一个非常经典和重要的问题。下面我们来从运筹学的角度，来详细了解一下 TSP。

直观上来讲，我们可以将问题建模为：

$$\begin{aligned} \min \quad & \sum_i \sum_j c_{ij} x_{ij} \\ & \sum_{i \in V} x_{ij} = 1, \quad \forall j \in V, i \neq j \\ & \sum_{j \in V} x_{ij} = 1, \quad \forall i \in V, i \neq j \\ & x_{ij} \in \{0,1\}, \quad \forall i, j \in V \end{aligned}$$

上面的模型：

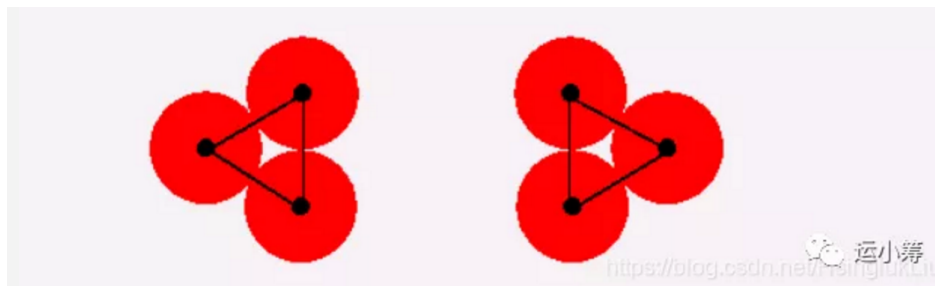
1. 约束 1：每个点都被离开一次；
2. 约束 2：每个点都被到达一次。

也就是说，上面的两个约束联合起来，可以保证，获得的解一定满足：每一个点都被访问一次，并且，经过一个点就会离开一个点。看上去貌似是没错的吧，直觉上是可以得到一条上图中展示的路径的，依次不重复经过所有节点的封闭的完美路径。但是不然。

也就是说，上述模型并不正确，会导致一个叫做子环路 subtour 的东东出现。如下面的简单解释。

子环路 (subtour)：没有包含所有节点的一条闭环。子环路首先是一个封闭的环；其次，这个环中被访问的节点集合 (假设为 S) 是所有节点集合 V 的一个真子集，也就是 $S \subseteq V$ 。或者说 $S \subset V, \text{and } |V| < N$ 。如果上述模型的解出现了子环路，那么为了满足模型的约束 1

和约束 2，解中必然至少存一个其他环路。这就导致与 TSP 想要得到的单环解矛盾，如下图所示的情况，图中出现了两个子环路。



图片来自 <http://www.math.uwaterloo.ca/tsp/methods/opt/subtour.htm>

可以看到，子环路也完美的满足

- (i) 每个点只被访问一次
- (ii) 经过一个点就离开那个点

但是这样的解会导致解中含有多个相离的环，也就是 subtour。而我们需要的是一个单个的经过所有点的大环。为了得到一个大环，我们就要添加消除子环路的约束，来完善 TSP 的模型。

这里比较常见的消除子环路的办法有两种：

加入 subtour-elimination 约束

加入 Miller-Tucker-Zemlin(MTZ) 约束

当然，之前我还钻研过 TSP 的另外一种建模思路，就是用 1-tree 的定义，结合纯图论的理论来支持建模的，日后我再专门写一个 1-tree 建模 +column generation 求解的文章。

3.2 TSP Model 1: subtour-elimination 消除子环路

3.2.1 TSP 整数规划模型

之前，我们的问题描述中提到，图中点的个数为 N , ($|V| = N$)。subtour-elimination 的思路比较直观，主要想法就是，根据子环路的特点，在模型中添加相应的约束，将其破开，用大白话说，就是破圈。

举个例子，假如考虑一个由图中点 3 个点 A, B, C 组成的子点集， $S = \{A, B, C\}$ ，假定他们仨构成了一个环 $A \rightarrow B \rightarrow C \rightarrow A$ 。那么这个环出现，就导致问题的解就必须要满足：

$$x_{AB} = x_{BC} = x_{CA} = 1$$

换句话说, 对于点 A, B, C 组成的结点子集合 $S = \{A, B, C\}$ 而言, 必须要有

$$x_{AB} + x_{BC} + x_{CA} \geq 3$$

也就是, 只有上面这个条件成立, 才会导致 subtour 的出现。那么这个子环路不存在的条件就是 (即 ‘破圈的方法’) 加入下面的约束

$$x_{AB} + x_{BC} + x_{CA} < 3$$

也就是说, 我们只允许所有点都被包含进来的环存在, 即包含点的个数为 N 的环, 删除其余所有的环。那怎么做呢, 一个简单的想法就是枚举, 也就是我们在 TSP 中经常看到的约束:

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad 2 \leq |S| \leq N - 1, S \subset V \quad (3.2.1)$$

可以看到 $2 \leq |S| \leq N - 1$ 就是枚举了所有可能导致子环路出现的结点集合 V 的集合 (枚举个数的复杂度为 2^N)。即我们只保留了 $|S| = N$ 的环。因此, 最终的 TSP 模型可以建模为下面的整数规划问题, 该问题是一个经典的 NP-hard 问题。目前也没有特别好的精确算法。

$$\min \sum_i \sum_j c_{ij} x_{ij} \quad (3.2.2)$$

$$\sum_{i \in V} x_{ij} = 1, \quad \forall j \in V, i \neq j \quad (3.2.3)$$

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in V, i \neq j \quad (3.2.4)$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad 2 \leq |S| \leq n - 1, S \subset V \quad (3.2.5)$$

$$x_{ij} \in \{0,1\}, \quad \forall i, j \in V \quad (3.2.6)$$

上面的模型中, 是假设了网络是全连接的情况, 也就是任何两个点都是可以直接到达的, 这也是为什么前两条约束加了 $i \neq j$ 的限制条件。不加这个条件, 会使得解变成 $x_{11} = 1, x_{22} = 1, \dots$ 你会发现它也满足约束。这个小坑可是一定要注意一下, 由于太小了, 大佬们在论文中是不会拿出来说的, 甚至在他们的模型中也没加上这个强调, 因为他们觉得这都是常识, 懒得跟你说, 说了显得自己很没水平, 哈哈。(我就不在意这个了, 我本来就是个小菜。) 很多

人也没有明确的提出来这个问题。我在这里提一下，免得大家踩坑了又浪费很久去 debug。

3.2.2 Python 调用 Gurobi 实现中的一些小问题

这个 subtour-elimination 的约束，是一个枚举的约束，我们不能在建模的时候就直接全枚举，这样的话有 2^N 复杂度的情况。等到把这些约束枚举完，黄花菜都凉了。

啰嗦几句，subtour-elimination 的思路就是相当于 cutting plane。在原来前两个约束的基础上，加上这个约束。但是如果你要在求解步骤 `model.optimize()` 之前就想全枚举，把 subtour-elimination 所有可能的 2^N 个约束全加上去，其他的不论，就只是加约束所耗费的时间，别人 TSP 早都解完去写 Paper 了，你这边约束还没加完。得不偿失，因此不能硬钢去枚举。

那怎么办呢？业内一般采用 Gurobi 或者 CPLEX 求解器中提供的 callback(回调函数) 的方法来动态的添加 subtour-elimination 约束。总的来讲，就是在 branch and bound tree 迭代的过程中，根据当前结点的松弛后的线性规划模型 (relaxed LP) 的解，来检查该解是否有存在子环路 subtour，如果有，我们就把执行 subtour-elimination 时候产生的破圈约束加到正在求解的模型中去；如果没有，我们就直接接着迭代算法。当然这个 check 的过程和 branch and bound tree 的过程是并行的。具体实现在下面展示。这里由于篇幅原因和为了保证可读性。我们先把这小节结束了。

3.3 TSP Model 2 : MTZ 约束消除子环路

3.3.1 MTZ 约束消除子环路

另外一种消除子环路的方法是加入 Miller-Tucker-Zemlin(MTZ) 约束。(本人认为这个方法的思想真的非常巧妙，做这个的时候就非常佩服前辈们的奇思妙想)。具体方法是：

对每个结点，引入一个决策变量 μ_i

利用 μ_i 构造 Miller-Tucker-Zemlin(MTZ) 约束

这样就可以完美的解决子环路的问题。

也就是引入决策变 $\mu_i, \forall i \in V, \mu_i \geq 0$ 量，然后假如下面的 MTZ 约束

$$\mu_i - \mu_j + Mx_{ij} \leq M - 1, \quad \forall i, j \in V, i, j \neq 0, i \neq j \quad (3.3.1)$$

【小思考 1】 μ_i 可以理解为点 $i \in V$ 的访问次序。比如 $\mu_1 = 5$ ，可以理解为点 1 是从出发点开始，第 5 个被访问到的点。很多最近的论文里也是这么解释的，再次验证了我的理解是和其他学者相近的。

【小思考 2】 μ_i 的取值范围一般设置成 $\mu_i \geq 0$ ，如果设置成无约束就会导致原问题不可行。

其中 M 是一个很大的数,也是运筹学中非常常见的基本操作-逻辑约束(就是 if..., then...)。参见课本 [3]

理论上讲,根据课本中的说法, M 应当是 $\mu_i - \mu_j + 1$ 的一个上界就可以,有论文指出,取最紧的上界,效果会好一些,因此我们取 $M = N$ 。参见文献 [4]。这样一来,上述约束可以拉紧为:

$$\mu_i - \mu_j + Nx_{ij} \leq N - 1, \quad \forall i, j \in V, i, j \neq 0, i \neq j \quad (3.3.2)$$

我们还是整理成逻辑约束的形式吧,上面的看着费劲

$$\mu_i - \mu_j + 1 - N(1 - x_{ij}) \leq 0 \quad \forall i, j \in V, i, j \neq 0, i \neq j \quad (3.3.3)$$

其中 N 为节点的个数,也就是算例的大小。

这样, TSP 问题的第二种最终版模型可以表示为

$$\min \sum_i \sum_j c_{ij} x_{ij} \quad (3.3.4)$$

$$\sum_{i \in V} x_{ij} = 1, \quad \forall j \in V, i \neq j \quad (3.3.5)$$

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in V, i \neq j \quad (3.3.6)$$

$$\mu_i - \mu_j + Nx_{ij} \leq N - 1, \quad \forall i, j \in V, i, j \neq 0, i \neq j \quad (3.3.7)$$

$$x_{ij} \in \{0,1\}, \mu_i \geq 0, \mu_i \in \mathbf{R}^1 \quad \forall i \in V \quad (3.3.8)$$

MTZ 约束的加入,使得原问题增加了 N 个连续变量和 N^2 复杂度个的逻辑约束,从代码实现上来讲,是非常方便的,比起 subtour-elimination 的实现要容易得多。

并且,就我看过的论文来讲,大家还是用 MTZ 约束多一些。像 TRB, TRC, TRE, TS, EJOR 上的文章,很多都是用 MTZ 约束,当然他们也不会论文中指出这些约束是 MTZ 约束,他们只是说这是消除子环路的,毕竟 MTZ 也是常识了。具体论文我不在这里举例了,之后再找时间贴过来。

接下来还是列几个小坑在这里,把前面说过的坑也一并再强调一下。

【小坑 2】 注意,实现这个的时候需要注意,由于 mu 表示访问顺序,由于 TSP 的起点和终点是一致的,如果不做一些处理,就会出现 infeasible 的情况。为此,我们假如一个虚拟点,也就是将起始点 s copy 一份,作为终止点,其实他俩位置是一样的。这样的话,就不会

存在问题了。这个坑相信很多人都踩过，我在这里给还没踩过的小伙伴提个醒。另外，就是实现的时候，一定记住，添加决策变量的时候，要判断 $i \neq j$ ，否则也会出问题。也就 x_{ij} 中， $i \neq j$ 必须成立才能添加变量，否则肯定会出错。

我们将模型修正一下，变成点集为 $V' = \{1, 2, \dots, N, N+1\}$ ，一共 $N+1$ 个点，其中点 1 和点 $N+1$ 是同一个点，点 1 表示起点，点 $N+1$ 表示终点。模型修正为

$$\min \sum_i \sum_j c_{ij} x_{ij} \quad (3.3.9)$$

$$\sum_{i \in V} x_{ij} = 1, \quad \forall j \in \{2, \dots, N+1\}, i \neq j \quad (3.3.10)$$

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in \{1, \dots, N\}, i \neq j \quad (3.3.11)$$

$$\mu_i - \mu_j + N x_{ij} \leq N - 1, \quad \forall i \in \{1, \dots, N\}, j \in \{2, \dots, N+1\}, i \neq j \quad (3.3.12)$$

$$x_{ij} \in \{0, 1\}, \mu_i \geq 0, \mu_i \in \mathbf{R}^1 \quad \forall i \in V \quad (3.3.13)$$

请仔细琢磨上面约束 1，约束 2 和约束 3，不等式后面的 comment 部分的细微变化，这些都是为之后写代码的时候做基础，为了避免栽跟头。

接下来，我们解释一下为什么 MTZ 约束会 work 吧。

3.3.2 为什么 MTZ 约束可以消除子环路？

MTZ 这个约束为什么能够消除子环路呢？

我们将 MTZ 约束、做一个变换，得到：

$$\mu_i - \mu_j + 1 + N(x_{ij} - 1) \leq 0 \quad \forall i \in \{1, \dots, n\}, j \in \{2, \dots, N+1\}, i \neq j \quad (3.3.14)$$

在上式中， $(x_{ij} - 1)$ 并不是 0-1 变量，而 $(1 - x_{ij})$ 才是 0-1 变量，因此该式变化成：

$$\mu_i - \mu_j + 1 - N(1 - x_{ij}) \leq 0 \quad \forall i \in \{1, \dots, n\}, j \in \{2, \dots, N+1\}, i \neq j \quad (3.3.15)$$

这个约束保证了，当 $x_{ij} = 1$ 时， $\mu_i - \mu_j + 1 \leq 0$

我们任取 $n, (n \leq N)$ 个点，他们之间的被选择的总数小于等于 $N - 1$ 即是消除了子环路。举例来说，任取 i, j, k 3 个点，如果出现子环路，则有

$$x_{ij} = x_{jk} = x_{ki} = 1 \quad (3.3.16)$$

$$x_{ij} + x_{jk} + x_{ki} = 3 \quad (3.3.17)$$

也就是说, 根据 MTZ 约束, 如果上述情况成立, 则必有:

$$\mu_i - \mu_j + 1 \leq 0 \quad (3.3.18)$$

$$\mu_j - \mu_k + 1 \leq 0 \quad (3.3.19)$$

$$\mu_k - \mu_i + 1 \leq 0 \quad (3.3.20)$$

将以上相加, 我们得到

$$3 \leq 0$$

上面不等式显然不成立, 这说明, 这个子环路不可能出现, 这也就用反证法证明了, 任一满足 MTZ 的点集, 都不存在环路。而注意, 我们的约束后的 comment 是 $\forall i \in \{1, \dots, N\}, j \in \{2, \dots, N+1\}, i \neq j$ 。这个在代码实现部分还是挺重要的。

其他情况, 任意取 n 个点, 都是同样的道理。这个约束成功的避免了子环路。

下面我们来讲一下, Python 调用 Gurobi, 如何用 callback 实现 subtour-elimination, 以及如何实现 MTZ 版的 TSP 求解吧。

3.4 Python+Gurobi: 用 callback 实现 TSP 的 subtour-elimination

首先, 我们还是以 VRP 上古大神 solomon [1] 的 benchmark 为算例, 来进行今天代码的数值实验部分。算例下载地址:<https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/100-customers/>

在这之前, 我想先说一下 Gurobi 和 CPLEX 里面的 callback 是怎么个逻辑:

3.4.1 callback 的工作逻辑: 王者荣耀版独家解读

下面我夹杂王者荣耀的角度来轻松解释 callback 是怎么起作用的, 打农药的小伙伴应该秒懂。(有些地方不严谨, 但是大概是这么个意思, 这段本来就是辅助理解, 不严谨的地方可以私信我, 我再修改)

1. 之前说过, subtour-elimination 的想法, 是想把所有的子集列举出来, 为每一个子集添加破圈约束, 但是这么做太慢。

2. 于是我们想, 先不加 subtour-elimination 的约束, 我们先把只含有 ‘前两组约束的 IP 输入给 Gurobi 求解, Gurobi 当然会先把 IP 的整数约束松弛掉, 把模型变成 LP, 然后调用 branch and bound 算法, 并将 IP 松弛后的 relaxed LP 作为根节, 进行 branch and bound tree 的迭代。

下面高能解释来了

我们把这个算法的迭代比作一场王者双排排位赛。假设我们准备开始玩游戏, 我方打野选了野王 Gurobi, OK, 我见势立马一手奶妈蔡文姬, 死跟打野, 只干两件事

1. 探视野 (识别 subtour) 和
2. 给助攻 (根据 subtour 构建破圈约束)。

Gurobi 大佬构建模型, 并且加入了前两组约束。(铭文带的不够呀, 我有点慌, 大佬却说, 躺好看我 carry, 帮我看蓝探视野), 而我们也不示弱, 选了 subtour-elimination 的辅助装出装策略。(也就是 subtour-elimination 的 callback 函数, 用于添加通过 callback 的方式添加 subtour-elimination 约束的)

3. 游戏开始, Gurobi 打着前两组约束, 并且设置 `model.Params.lazyConstraints = 1` 也就是给我 (软辅蔡文姬) 发出跟着我的信号。然后拉着我一起双排开始了游戏, 代码中就是 `model.optimize(subtourelim)`。

OK, 算法开始迭代。野王 Gurobi 还是基本操作, 熟练的 branch and bound 在野区以最适合的刷野路径刷野。在刷野 (迭代) 的过程中, 在每个 branch and bound tree 的结点处, Gurobi 会去调用各种变式的 simplex 算法得到该节点的解。如果得到了一个整数解 (也可以是得到小数解就操作), 我们可以在这个地方, 人为的插一脚, 相当于 Gurobi 老哥正在屁颠屁颠刷野 (跑算法) 呢, 我们在旁边探视野 (做监工), 一直就这么直勾的看着。我们看到老哥得到了一个整数解 (或者一个小数可行解), 一机灵, 激动地过去拍拍 Gurobi 老哥的肩膀说, 大佬, 我拿一下这个节点的 LP 解哈, 您继续。

4.OK, 拿到了 LP 的解以后, 我们自己来看看这个解中存不存在 subtour, 如果我们检测完后发现不存在。尴尬, 我们假装啥也没发生。继续静静地看着 Gurobi 老哥刷野 (算法迭代)。下一次, Gurobi 老哥又得到了一个整数解 (或者一个小数可行解), 我们再厚脸皮去拿过来检测, 结果发现有 2-5-8-2 这个子环路混在里面, 这次可被我逮个正着哈, 小伙儿, 你子环路了么。我们激动地大声告诉 Gurobi 老哥: ”老哥, 老哥, 子环路 2-5-8-2 在这儿挑衅你, 你去 GUNK 它, 把 2-5-8-2 这个混子送回家”, 顺便我们还根据这个子环路 ‘2-5-8-2 的特点, 快速给 Gurobi 老哥想出了一套连招: 老哥, 你 1433223 就可以秒他!!! 哈哈, 在实际中, 把这个子环路踢出去的方法 (也就是刚刚的连招) 就是加下面这个约束:

$$x_{25} + x_{58} + x_{82} \leq 2$$

这里还有个坑, 虽然你也可以写成等价的

$$x_{25} + x_{58} + x_{82} < 3$$

, 但是求解器是不接受 $<$, $>$ 这样的约束的, 你要硬加, 那就报错。很多人其实并不知道这一点, 我在这里提一下。

5. 接上面, Gurobi 老哥非常强, 手脚麻利动作快, 脑子很好使能同时处理多个信息, 听到了我们奶妈蔡文姬的报视野—前面草丛有个 2-5-8-2 的 ADC 很浪, 落单了, 还 1433223 连招可以秒他, 回头敬个礼说: 好嘞, 知道了, 放心吧, 瞧好了您呐, 看我把他怼出去哈。看这老哥如此稳的操作, 我心中的默默点赞: 同九义, 何汝秀。然后继续监工。之后我就再也没见过 2-5-8-2 这个子环路混子。之后的监工中, 我这奶妈蔡文姬又陆续把 1-5-8-1, 3-4-7-9-3 等一众混子报给 Gurobi 老哥, 老哥一一将他们 1433223 送回家。

6. 由于我方打野位 Gurobi 老哥刚开始只加入了前两组约束, 轻车简从, 走路带风, 身为奶妈蔡文姬的我紧跟打野, 时刻监视打野行为, 并不断为打野探视野, 找敌方落单 ADC, 并每次都及时给个助攻 subtour-elimination constraints ($x_{25} + x_{58} + x_{82} \leq 2$), 抢个人头, 最终 Gurobi 老哥轻松 carry 全场, 拿到 2-5-8-2, 3-4-7-9-3, 1-5-8-1 等 3 个人头。直击敌方水晶, 获得最优解 $[0, 4, 3, 7, 1, 2, 5, 8, 9, 6, 0]$ 。评分 127, 夺得胜方 MVP。起立, 鼓掌!!! 是的, 每次都是这样。

上面的描述并不完美, 但是我想, 应该能给你一些辅助理解 callback 工作逻辑的帮助。

3.5 使用 callback 的通用步骤

其实总结一下, 使用 callback 的方法分为下面几步 (只针对本问题)

第一步: 利用 Gurobi 构建数学模型, 只加入前两组约束;

第二步: 构建一个用来识别 subtour 并返回消除子环路约束的函数 subtourelim(model, where)(注意, 这个函数的参数 model, where 是固定的, 求解器规定的)。这个函数用于: 拿到整数规划分支定界迭代过程中当前结点的解的信息, 并根据当前节点的解, 识别子环路, 如有则返回消除子环路的约束, 否则不作操作。

第三步: 设置使用 lazyConstraints, 并启动优化算法求解模型, 也就是 model.optimize(subtourelim), 而且必须以 callback 函数 subtourelim(model, where) 为参数, 具体代码为:

```
Code
1  model._vars = X
2  model.Params.lazyConstraints = 1
3  model.optimize(subtourelim)
```

【Remark】 这里, subtourelim(model, where) 中, 添加子环路消除约束是以 lazyConstraints 的形式添加的, lazyConstraints 就是不在建模一开始就加入, 而是在算法迭代的过程

中，动态的在 branch and bound 的分支节点处才加入的约束。可以通过 callback 函数，控制在节点的解满足什么样的条件下，我们去构建特定形式的约束，这个约束以 lazyConstraints 的形式构建并添加到求解的函数 model.optimize() 中，然后 Gurobi 就可以自动的识别，且调用 callback 函数，按照你的要求在求解过程中把约束加进去。这一招 branch and cut, benders decomposition, row generation 的时候用的非常多。想要进阶的小伙伴这招儿还是必须要攻克的。

3.5.1 callback 实现 subtour-elimination 的详细代码

这部分代码有点长，待我明天再放出来把。算了，还是直接放上来把。首先定义一些读取数据的函数：

readData(path, nodeNum): 读取.txt 文档中的算例数据;

reportMIP(model, Routes): 获得并打印最优解信息;

getValue(var dict,nodeNum): 获得决策变量的值，并存储返回一个 np.array() 数组;

getRoute(x value): 根据解 x value 得到该解对应的路径。

```

Code
1  # -*- coding:utf-8 -*-
2  '''
3  @author: Hsinglu Liu
4  @version: 1.0
5  @Date: 2019.5.5
6  '''
7
8  from __future__ import print_function
9  from __future__ import division, print_function
10 from gurobipy import *
11 import re;
12 import math;
13 import matplotlib.pyplot as plt
14 import numpy as np
15 import pandas as pd
16 import copy
17 from matplotlib.lines import lineStyles
18 import time
19
20 starttime = time.time()
21
22 # function to read data from .txt files
23 def readData(path, nodeNum):
24     nodeNum = nodeNum;
25     cor_X = []
26     cor_Y = []
27
28     f = open(path, 'r');
29     lines = f.readlines();

```



```

30     count = 0;
31     # read the info
32     for line in lines:
33         count = count + 1;
34         if(count >= 10 and count <= 10 + nodeNum):
35             line = line[:-1]
36             str = re.split(r" +", line)
37             cor_X.append(float(str[2]))
38             cor_Y.append(float(str[3]))
39
40     # compute the distance matrix
41     disMatrix = [[0] * nodeNum for p in range(nodeNum)]; # 初始化距离矩阵的维度, 防止浅拷贝
42     # data.disMatrix = [[0] * nodeNum] * nodeNum; 这个是浅拷贝, 容易重复
43     for i in range(0, nodeNum):
44         for j in range(0, nodeNum):
45             temp = (cor_X[i] - cor_X[j])**2 + (cor_Y[i] - cor_Y[j])**2;
46             disMatrix[i][j] = (int)(math.sqrt(temp));
47             # disMatrix[i][j] = 0.1 * (int)(10 * math.sqrt(temp));
48             # if(i == j):
49             #     data.disMatrix[i][j] = 0;
50             # print("%6.0f" % (math.sqrt(temp)), end = " ");
51             temp = 0;
52
53     return disMatrix;
54
55 def printData(disMatrix):
56     print("-----cost matrix-----\n");
57     for i in range(len(disMatrix)):
58         for j in range(len(disMatrix)):
59             #print("%d %d" % (i, j));
60             print("%6.1f" % (disMatrix[i][j]), end = " ");
61             # print(disMatrix[i][j], end = " ");
62         print();
63
64 def reportMIP(model, Routes):
65     if model.status == GRB.OPTIMAL:
66         print("Best MIP Solution: ", model.objVal, "\n")
67         var = model.getVars()
68         for i in range(model.numVars):
69             if(var[i].x > 0):
70                 print(var[i].varName, " = ", var[i].x)
71                 print("Optimal route:", Routes[i])
72
73 def getValue(var_dict, nodeNum):
74     x_value = np.zeros([nodeNum + 1, nodeNum + 1])
75     for key in var_dict.keys():
76         a = key[0]
77         b = key[1]
78         x_value[a][b] = var_dict[key].x
79

```

```

80     return x_value
81
82 def getRoute(x_value):
83     # 假如是 5 个点的算例，我们的路径会是 1-4-2-3-5-6 这样的，因为我们加入了一个虚拟点
84     # 也就是当路径长度为 6 的时候，我们就停止，这个长度和 x_value 的长度相同
85     x = copy.deepcopy(x_value)
86     # route_temp.append(0)
87     previousPoint = 0
88     arcs = []
89     route_temp = [previousPoint]
90     count = 0
91     while(len(route_temp) < len(x) and count < len(x)):
92         print('previousPoint: ', previousPoint, 'count: ', count)
93         if(x[previousPoint][count] > 0):
94             previousPoint = count
95             route_temp.append(previousPoint)
96             count = 0
97             continue
98         else:
99             count += 1
100     return route_temp
101
102 # cost = [[0, 7, 2, 1, 5],
103 #          [7, 0, 3, 6, 8],
104 #          [2, 3, 0, 4, 2],
105 #          [1, 6, 4, 0, 9],
106 #          [5, 8, 2, 9, 0]]

```

然后定义几个非常关键的用于添加 subtour-elimination 约束的函数：

subtourelim(model, where): callback 函数,用于为 model 对象动态添加 subtour-elimination 约束；

computeDegree(graph): 给定一个 graph(二维数组形式)，也就是给定一个邻接矩阵，计算出每个结点的 degree.(degree= 每个结点被进入次数 + 被离开的次数)；

findEdges(graph): 给定一个 graph(二维数组形式)，也就是给定一个邻接矩阵，找到该图中所有的边，例如 [(1, 2), (2, 4), (2, 5)]；

subtour(graph): 给定一个 graph(二维数组形式)，也就是给定一个邻接矩阵，找到该图中包含结点数目最少的子环路，例如 [2, 3, 5]。其中，函数 subtourelim(model,where) 中，调用了函数 computeDegree(graph)、findEdges(graph) 和 subtour(graph)。

Code

```

1  # Callback - use lazy constraints to eliminate sub-tours
2
3  # Callback - use lazy constraints to eliminate sub-tours
4
5  def subtourelim(model, where):

```

```

6     if(where == GRB.Callback.MIPSOL):
7         # make a list of edges selected in the solution
8         print('model._vars', model._vars)
9         #         vals = model.cbGetSolution(model._vars)
10        x_value = np.zeros([nodeNum + 1, nodeNum + 1])
11        for m in model.getVars():
12            if(m.varName.startswith('x')):
13                #         print(var[i].varName)
14                #         print(var[i].varName.split('_'))
15                a = (int)(m.varName.split('_')[1])
16                b = (int)(m.varName.split('_')[2])
17                x_value[a][b] = model.cbGetSolution(m)
18        print("solution = ", x_value)
19        #         print('key = ', model._vars.keys())
20        #         selected = []
21        #         for i in range(nodeNum):
22        #             for j in range(nodeNum):
23        #                 if(i != j and x_value[i][j] > 0.5):
24        #                     selected.append((i, j))
25        #                 selected = tuplelist(selected)
26        # #         selected = tuplelist((i,j) for i in range(nodeNum), for if x_value[i][j] > 0.5)
27        #         print('selected = ', selected)
28        # find the shortest cycle in the selected edge list
29        tour = subtour(x_value)
30        print('tour = ', tour)
31        if(len(tour) < nodeNum + 1):
32            # add subtour elimination constraint for every pair of cities in tour
33            print("---add sub tour elimination constraint---")
34            #         model.cbLazy(quicksum(model._vars[i][j]
35            #                                 for i in tour
36            #                                 for j in tour
37            #                                 if i != j)
38            #                         <= len(tour)-1)
39            #         LinExpr = quicksum(model._vars[i][j]
40            #                                 for i in tour
41            #                                 for j in tour
42            #                                 if i != j)
43            for i,j in itertools.combinations(tour, 2):
44                print(i,j)
45
46            model.cbLazy(quicksum(model._vars[i, j]
47                                for i,j in itertools.combinations(tour, 2))
48                            <= len(tour)-1)
49            LinExpr = quicksum(model._vars[i, j]
50                                for i,j in itertools.combinations(tour, 2))
51            print('LinExpr = ', LinExpr)
52            print('RHS = ', len(tour)-1)
53
54        # compute the degree of each node in given graph
55        def computeDegree(graph):

```

```

56     degree = np.zeros(len(graph))
57     for i in range(len(graph)):
58         for j in range(len(graph)):
59             if graph[i][j] > 0.5:
60                 degree[i] = degree[i] + 1
61                 degree[j] = degree[j] + 1
62     print('degree', degree)
63     return degree
64
65     # given a graph, get the edges of this graph
66     def findEdges(graph):
67         edges = []
68         for i in range(1, len(graph)):
69             for j in range(1, len(graph)):
70                 if graph[i][j] > 0.5:
71                     edges.append((i, j))
72
73         return edges
74
75
76
77     # Given a tuplelist of edges, find the shortest subtour
78     def subtour(graph):
79         # compute degree of each node
80         degree = computeDegree(graph)
81         unvisited = []
82         for i in range(1, len(degree)):
83             if degree[i] >= 2:
84                 unvisited.append(i)
85         cycle = range(0, nodeNum + 1) # initial length has 1 more city
86
87         edges = findEdges(graph)
88         edges = tuplelist(edges)
89         print(edges)
90         while unvisited: # true if list is non-empty
91             thiscycle = []
92             neighbors = unvisited
93             while neighbors: # true if neighbors is non-empty
94                 current = neighbors[0]
95                 thiscycle.append(current)
96                 unvisited.remove(current)
97                 neighbors = [j for i, j in edges.select(current, '*') if j in unvisited]
98                 neighbors2 = [i for i, j in edges.select('*', current) if i in unvisited]
99                 if neighbors2:
100                     neighbors.extend(neighbors2)
101             #         print('current:', current, '\n neighbors', neighbors)
102
103         isLink = ((thiscycle[0], thiscycle[-1]) in edges) or ((thiscycle[-1], thiscycle[0]) in edges)
104         if len(cycle) > len(thiscycle) and len(thiscycle) >= 3 and isLink:

```

```

105 #             print('in = ', ((thiscycle[0], thiscycle[-1]) in edges) or ((thiscycle[-1],
↪   thiscycle[0]) in edges))
106         cycle = thiscycle
107         return cycle
108     return cycle

```

然后是建模部分的代码，建模部分相比学运筹的人比较熟悉，这里比较特殊的就是求解时候的几行代码：

model.Params.lazyConstraints = 1 : set lazy constraints Parameter

model.optimize(subtourelim) : use callback function when executing branch and bound algorithm

```

Code
1  # nodeNum = 5
2  nodeNum = 10
3  # # path = 'C:\Users\hsingluLiu\eclipse-workspace\PythonCallGurobi_Applications\VRPTW\R101.txt';
4  path = 'solomon-100/in/c201.txt';
5  cost = readData(path, nodeNum)
6  printData(cost)
7
8  model = Model('TSP')
9
10 # creat decision variables
11 X = {}
12 mu = {}
13 for i in range(nodeNum + 1):
14     mu[i] = model.addVar(lb = 0.0
15                          , ub = 100 #GRB.INFINITY
16                          # , obj = distance_initial
17                          , vtype = GRB.CONTINUOUS
18                          , name = "mu_" + str(i)
19                          )
20
21 for j in range(nodeNum + 1):
22     if(i != j):
23         X[i, j] = model.addVar(vtype = GRB.BINARY
24                                , name = 'x_' + str(i) + '_' + str(j)
25                                )
26
27 # set objective function
28 obj = LinExpr(0)
29 for key in X.keys():
30     i = key[0]
31     j = key[1]
32     if(i < nodeNum and j < nodeNum):
33         obj.addTerms(cost[key[0]][key[1]], X[key])
34     elif(i == nodeNum):
35         obj.addTerms(cost[0][key[1]], X[key])

```

```

36     elif(j == nodeNum):
37         obj.addTerms(cost[key[0]][0], X[key])
38
39 model.setObjective(obj, GRB.MINIMIZE)
40
41 # add constraints 1
42 for j in range(1, nodeNum + 1):
43     lhs = LinExpr(0)
44     for i in range(0, nodeNum):
45         if(i != j):
46             lhs.addTerms(1, X[i, j])
47     model.addConstr(lhs == 1, name = 'visit_' + str(j))
48
49 # add constraints 2
50 for i in range(0, nodeNum):
51     lhs = LinExpr(0)
52     for j in range(1, nodeNum + 1):
53         if(i != j):
54             lhs.addTerms(1, X[i, j])
55     model.addConstr(lhs == 1, name = 'visit_' + str(j))
56
57 # model.addConstr(X[0, nodeNum] == 0, name = 'visit_' + str(0) + ',' + str(nodeNum))
58
59 # set lazy constraints
60 model._vars = X
61 model.Params.lazyConstraints = 1
62 model.optimize(subtourelim)
63 # subProblem.optimize()
64 x_value = getValue(X, nodeNum)
65 # route = getRoute(x_value)
66 # print('optimal route:', route)

```

搞定。再重复一下，关键的地方就是 `subtourelim()` 这个函数和 `subtour(graph)` 这两个关键函数。还有就是求解的时候，别忘了 `model.optimize(subtourelim)`。就可以了。

Ok, 我们将 `solomon100` 个点的 VRP 算例中的 `c201.txt` 拿出来，取前 10 个点运行一下，结果为：

	Code
1	<code>obj : 127</code>
2	<code>optimal route: [0, 4, 3, 7, 1, 2, 5, 8, 9, 6, 0]</code>

完美！是不是觉得世界都又好了。哈哈，若干年前，我就是这种感觉。

3.6 Python+Gurobi: 实现 TSP 的 MTZ 约束版

首先定义一些读取数据的函数什么的，同上。

Code

```

1  # -*- coding:utf-8 -*-
2  '''
3  @author: Hsinglu Liu
4  @version: 1.0
5  @Date: 2019.5.5
6  '''
7
8  # -*- coding:utf-8 -*-
9  '''
10 @author: Hsinglu Liu
11 @version: 1.0
12 @Date: 2019.5.5
13 '''
14
15 from __future__ import print_function
16 from __future__ import division, print_function
17 from gurobipy import *
18 import re;
19 import math;
20 import matplotlib.pyplot as plt
21 import numpy as np
22 import pandas as pd
23 import copy
24 from matplotlib.lines import lineStyles
25 import time
26
27 starttime = time.time()
28
29 # function to read data from .txt files
30 def readData(path, nodeNum):
31     nodeNum = nodeNum;
32     cor_X = []
33     cor_Y = []
34
35     f = open(path, 'r');
36     lines = f.readlines();
37     count = 0;
38     # read the info
39     for line in lines:
40         count = count + 1;
41         if(count >= 10 and count <= 10 + nodeNum):
42             line = line[:-1]
43             str = re.split(r" +", line)
44             cor_X.append(float(str[2]))
45             cor_Y.append(float(str[3]))
46
47     # compute the distance matrix
48     disMatrix = [[([0] * nodeNum) for p in range(nodeNum)]]; # 初始化距离矩阵的维度, 防止浅拷贝
49     # data.disMatrix = [[0] * nodeNum] * nodeNum]; 这个是浅拷贝, 容易重复

```

```

50     for i in range(0, nodeNum):
51         for j in range(0, nodeNum):
52             temp = (cor_X[i] - cor_X[j])**2 + (cor_Y[i] - cor_Y[j])**2;
53             disMatrix[i][j] = (int)(math.sqrt(temp));
54         #         disMatrix[i][j] = 0.1 * (int)(10 * math.sqrt(temp));
55         #         if(i == j):
56         #             data.disMatrix[i][j] = 0;
57         #         print("%6.0f" % (math.sqrt(temp)), end = " ");
58         temp = 0;
59
60     return disMatrix;
61
62 def printData(disMatrix):
63     print("-----cost matrix-----\n");
64     for i in range(len(disMatrix)):
65         for j in range(len(disMatrix)):
66             #print("%d %d" % (i, j));
67             print("%6.1f" % (disMatrix[i][j]), end = " ");
68         #         print(disMatrix[i][j], end = " ");
69     print();
70
71 def reportMIP(model, Routes):
72     if model.status == GRB.OPTIMAL:
73         print("Best MIP Solution: ", model.objVal, "\n")
74         var = model.getVars()
75         for i in range(model.numVars):
76             if(var[i].x > 0):
77                 print(var[i].varName, " = ", var[i].x)
78                 print("Optimal route:", Routes[i])
79
80 def getValue(var_dict, nodeNum):
81     x_value = np.zeros([nodeNum + 1, nodeNum + 1])
82     for key in var_dict.keys():
83         a = key[0]
84         b = key[1]
85         x_value[a][b] = var_dict[key].x
86
87     return x_value
88
89 def getRoute(x_value):
90     '''
91     input: x_value 的矩阵
92     output: 一条路径, [0, 4, 3, 7, 1, 2, 5, 8, 9, 6, 0], 像这样
93     '''
94     # 假如是 5 个点的算例, 我们的路径会是 1-4-2-3-5-6 这样的, 因为我们加入了一个虚拟点
95     # 也就是当路径长度为 6 的时候, 我们就停止, 这个长度和 x_value 的长度相同
96     x = copy.deepcopy(x_value)
97     #     route_temp.append(0)
98     previousPoint = 0
99     route_temp = [previousPoint]

```



```

100     count = 0
101     while(len(route_temp) < len(x)):
102         #print('previousPoint: ', previousPoint )
103         if(x[previousPoint][count] > 0):
104             previousPoint = count
105             route_temp.append(previousPoint)
106             count = 0
107             continue
108         else:
109             count += 1
110     return route_temp
111 '''
112 # toy example
113 cost = [[0, 7, 2, 1, 5],
114         [7, 0, 3, 6, 8],
115         [2, 3, 0, 4, 2],
116         [1, 6, 4, 0, 9],
117         [5, 8, 2, 9, 0]]
118 '''

```

然后就是 Python 调用 Gurobi 求解 TSP 的代码了 (MTZ 约束消除子环路)。MTZ 的实现还是比较简单的。

```

Code
1  # nodeNum = 5
2  nodeNum = 10
3  # # path = 'C:\Users\hsingluLiu\eclipse-workspace\PythonCallGurobi_Applications\VRPTW\R101.txt';
4  path = 'solomon-100/in/c201.txt';
5  cost = readData(path, nodeNum)
6  printData(cost)
7
8
9  model = Model('TSP')
10
11  # creat decision variables
12  X = {}
13  mu = {}
14  for i in range(nodeNum + 1):
15      mu[i] = model.addVar(lb = 0.0
16                          , ub = 100 #GRB.INFINITY
17                          # , obj = distance_initial
18                          , vtype = GRB.CONTINUOUS
19                          , name = "mu_" + str(i)
20                          )
21
22  for j in range(nodeNum + 1):
23      if(i != j):
24          X[i, j] = model.addVar(vtype = GRB.BINARY
25                                , name = 'x_' + str(i) + '_' + str(j)

```

```

26         )
27
28     # set objective function
29     obj = LinExpr(0)
30     for key in X.keys():
31         i = key[0]
32         j = key[1]
33         if(i < nodeNum and j < nodeNum):
34             obj.addTerms(cost[key[0]][key[1]], X[key])
35         elif(i == nodeNum):
36             obj.addTerms(cost[0][key[1]], X[key])
37         elif(j == nodeNum):
38             obj.addTerms(cost[key[0]][0], X[key])
39
40     model.setObjective(obj, GRB.MINIMIZE)
41
42     # add constraints 1
43     for j in range(1, nodeNum + 1):
44         lhs = LinExpr(0)
45         for i in range(0, nodeNum):
46             if(i != j):
47                 lhs.addTerms(1, X[i, j])
48         model.addConstr(lhs == 1, name = 'visit_' + str(j))
49
50     # add constraints 2
51     for i in range(0, nodeNum):
52         lhs = LinExpr(0)
53         for j in range(1, nodeNum + 1):
54             if(i != j):
55                 lhs.addTerms(1, X[i, j])
56         model.addConstr(lhs == 1, name = 'visit_' + str(j))
57
58     # add MTZ constraints
59     # for key in X.keys():
60     #     org = key[0]
61     #     des = key[1]
62     #     if(org != 0 or des != 0):
63     #         pass
64     #         model.addConstr(mu[org] - mu[des] + 100 * X[key] <= 100 - 1)
65     for i in range(0, nodeNum):
66         for j in range(1, nodeNum + 1):
67             if(i != j):
68                 model.addConstr(mu[i] - mu[j] + 100 * X[i, j] <= 100 - 1)
69
70     model.write('model.lp')
71     model.optimize()
72
73     x_value = getValue(X, nodeNum)
74     route = getRoute(x_value)

```

```
75 print('optimal route:', route)
```

设置 10 个点跑一个 toy example 试试, 结果为

```
Code
1 Explored 683 nodes (4011 simplex iterations) in 0.19 seconds
2 Thread count was 8 (of 8 available processors)
3
4 Solution count 5: 127 137 150 ... 230
5
6 Optimal solution found (tolerance 1.00e-04)
7 Best objective 1.270000000000e+02, best bound 1.270000000000e+02, gap 0.0000%
8 optimal route: [0, 4, 3, 7, 1, 2, 5, 8, 9, 6, 10]
```

由于我们把起始点 copy 了一下, 因此最优解为

```
Code
1 obj : 127
2 optimal route: [0, 4, 3, 7, 1, 2, 5, 8, 9, 6, 0]
```

3.7 后记

运筹修炼真是个非常磨人的事情, 需要理论与实战结合才能理解更深入。理论已经门槛够高了, 再加上编程实现, 可真要了命了。另外有非常多的小细节, 大佬们在论文中并不会讲, 但是又非常关键, 只有自己实际一个一个去踩坑, 或者多请教前辈, 毕竟行万里路不如高人指路。这里我把我的笔记和心得放在这里, 供大家参考, 互相交流学习, 进步。也是为我自己整理、复习一下之前的知识。以后我自己复习的时候回来看也非常方便。

3.8 参考文献

1. : Desrochers, M., Desrosiers, J., Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. Operations research, 40(2), 342-354. <https://doi.org/10.1287/opre.40.2.342>
2. :Gurobi documents <https://www.gurobi.com/documentation/>
3. :Winston, W. L., Goldberg, J. B. (2004). Operations research: applications and algorithms (Vol. 3). Belmont eCalif Calif: Thomson/Brooks/Cole.
4. :Desaulniers, G., Desrosiers, J., Solomon, M. M. (Eds.). (2006). Column generation (Vol. 5). Springer Science Business Media.

Chapter 4

2020-11-23: Java 调用 cplex 求解运输问题

作者:夏旻,清华大学,工业工程系/深圳国际研究生院 (硕士在读) 邮箱:xia970201@gmail.com

本文中的课件来自清华大学深圳国际研究生院,物流与交通学部张灿荣教授《生产管理》课程。

4.1 运输问题 (Transportation Problem) 描述

运输问题是一种特殊的最小费用网络流问题,其中每个节点都是一个纯供给节点或一个纯需求节点。即所有的流都从供给它的某个源头节点出发,而后直接到达一个需要它的需求节点,中间不经过其它中间步骤或转运环节。一个运输问题中的商品流可能是任何东西,唯一必要的条件就是需要从供给源头节点流入需求节点。总结一下,运输问题具有以下的特点:

The Transportation Problem

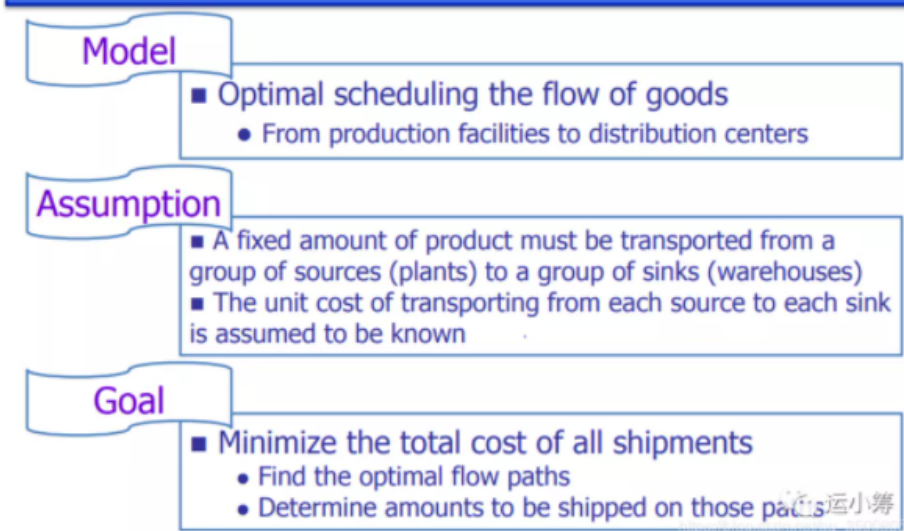


Figure 4.1.1: 运输问题的特点

4.2 运输问题的数学模型

首先定义运输问题的参数与决策变量:

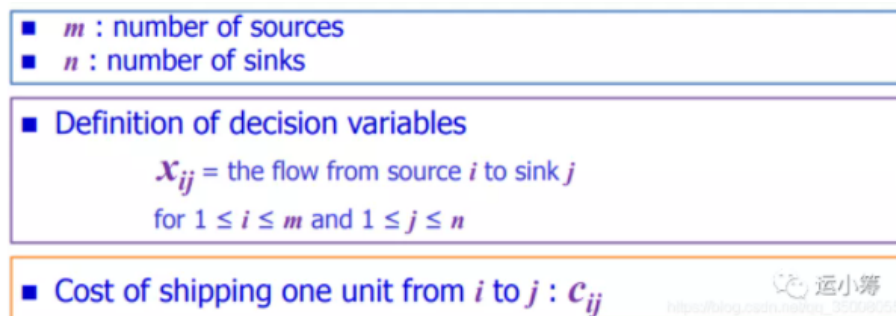


Figure 4.2.1: 运输问题的参数与决策变量

目标函数为最小化运输成本:

$$\min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

约束一为从每个供给节点运出的商品总和等于该节点可用商品的数量:

$$\sum_j^n x_{ij} = a_i, \quad \forall 1 \leq i \leq m$$

约束二为每个需求节点流入的商品总和等于该节点需求商品的数量：

$$\sum_i^m x_{ij} = b_j, \quad \forall 1 \leq j \leq n$$

最后决策变量的取值范围：

$$x_{ij} \geq 0, \quad \forall 1 \leq i \leq m \text{ and } 1 \leq j \leq n$$

综上，(4.2)-(4.2) 式即为运输问题的数学模型，且为一个线性规划模型。

4.3 Java 调用 cplex 求解运输问题

本节介绍如果使用 Java 语言调用 cplex 求解器对运输问题进行求解。问题相关参数如下：

■ Shipping cost per 1000 units in \$

		TO			
		Amarillo	Teaneck	Chicago	Sioux Falls
FROM	Sunnyvale	250	420	380	280
	Dublin	1280	990	1440	1520
	Bangkok	1550	1420	1660	730

Figure 4.3.1: 问题相关参数如下

Java 是一种面向对象的编程语言，故编写过程中主要涉及以下几个类：

4.3.1 transportation_node 类

```

Code
1  package transportation_problem;
2
3  /**
4   * This class is to define the transportation_node
5   * @param Property of node
6   * @return a node object
7   */

```

```

8
9 public class transportation_node {
10     String nodeName;
11     int quantity;
12     int isSource;
13     int id;
14     public transportation_node(String nodeName,int quantity,int isSource,int id) {
15         this.nodeName = nodeName;
16         this.quantity = quantity;
17         this.isSource = isSource;
18         this.id = id;
19     }
20
21     public boolean isSource() {
22         if(this.isSource == 1) {
23             return true;
24         }
25         return false;
26     }
27 }

```

4.3.2 transportation_relation 类

Code

```

1 package transportation_problem;
2
3 /**
4  * This class is to define the transportation_relation
5  * @param Property of node
6  * @return a relation object
7  */
8
9 public class transportation_relation {
10     int [] distance ;
11     public transportation_relation(int[] distance) {
12         this.distance = distance;
13     }
14 }

```

4.3.3 读取数据

Code

```

1 package transportation_problem;
2
3 import java.io.*;
4 import java.util.*;
5
6 /**

```

```

7  * This class is to read data from txt file
8  * @param path: the file path to be read
9  * @return return a data object
10 * @throws IOException
11 */
12
13 public class Readfile {
14     List<transportation_node> transportationNodeList = new ArrayList<transportation_node>();
15     transportation_relation transportationRelation;
16
17     public Readfile(String path1,String path2) throws IOException {
18
19         readtransportation_node(path1);
20         transportation_relation(path2);
21     }
22
23     public void readtransportation_node(String path) throws IOException {
24         BufferedReader br = new BufferedReader(new FileReader(path));
25         String line = new String();
26
27         //读取变量
28         br.readLine();
29         line = br.readLine();
30         String[] tokens = line.split("\\s+");
31         while(tokens.length > 0) {
32             transportation_node Temp = new
33                 ↪ transportation_node(tokens[0],Integer.parseInt(tokens[1]),Integer.parseInt(tokens[2]),Integer.parseInt(to
34             transportationNodeList.add(Temp);
35             line = br.readLine();
36             tokens = line.split("\\s+");
37         }
38         br.close();
39     }
40
41     public void transportation_relation(String path) throws IOException {
42         BufferedReader br = new BufferedReader(new FileReader(path));
43         String line[] = new String[100];
44         int row = 0;
45         line[row] = br.readLine();
46         while(line[row] != null) {
47             row++;
48             line[row] = br.readLine();
49         }
50
51         String[] tokens = line[0].split("\\s+");
52         int column = tokens.length;
53         int [] dis = new int[(row-1)*column];
54
55         for (int i = 1; i < row; i++) {

```



```

56         tokens = line[i].split("\\s+");
57         for (int j = 1; j <= column; j++) {
58             dis[(column)*(i-1)+j-1] = Integer.parseInt(tokens[j]);
59         }
60     }
61     transportationRelation = new transportation_relation(dis);
62     br.close();
63 }
64 }

```

4.3.4 在 cplex 中建立运输问题模型

```

Code
1  package transportation_problem;
2
3  import java.util.List;
4  import ilog.concert.IloException;
5  import ilog.concert.IloNumExpr;
6  import ilog.concert.IloNumVar;
7  import ilog.cplex.IloCplex;
8
9  /**
10   * This class is to conduct a model in cplex and solve
11   * @param problem parameter
12   * @return return solve result
13   * @throws IloException
14   */
15
16  public class model_transportation {
17      IloCplex cplex ;
18      double objectiveValue;
19      IloNumVar x[];
20      int numOfPlants = 0;
21      int numOfdestination = 0;
22      /**
23       * This method is to bulidModel
24       * @param transportation_problem's data
25       * @return transportation_problem's model
26       * @throws IloException
27       */
28      public void bulidModel(List<transportation_node> transportationNodeList,transportation_relation
29      ↪ transportationRelation)throws IloException {
30          this.cplex = new IloCplex();
31          for(transportation_node tsNode : transportationNodeList){
32              if (tsNode.isSource()) {
33                  numOfPlants++;
34              }else {
35                  numOfdestination++;
36              }
37          }
38      }
39  }

```

```

36     }
37     CreatDecisionVariab();
38     CreatObjFunc(transportationRelation);
39     CreatConstraints(transportationNodeList);
40     Solve(transportationNodeList);
41 }
42 /**
43  * This method is to create decision variables
44  * @throws IloException
45  */
46 public void CreatDecisionVariab() throws IloException{
47     x = new IloNumVar[numOfPlants*numOfdestination];
48     for (int i = 0; i < numOfPlants; i++) {
49         for (int j = 0; j < numOfdestination; j++) {
50             x[i*numOfdestination+j] = cplex.numVar(0, Double.MAX_VALUE,"x"+(i+1)+(j+1));
51         }
52     }
53 }
54 /**
55  * This method is to create objective function
56  * @throws IloException
57  */
58 public void CreatObjFunc(transportation_relation transportationRelation) throws IloException{
59     cplex.addMinimize(cplex.scalProd(x,transportationRelation.distance));
60 }
61 /**
62  * This method is to add constraints
63  * @throws IloException
64  */
65 public void CreatConstraints(List<transportation_node> transportationNodeList) throws
66 ↪ IloException {
67     for(transportation_node tsNode : transportationNodeList){
68         IloNumExpr left = cplex.linearNumExpr();
69         if (tsNode.isSource()) {
70             for(int i = 0;i<numOfdestination;i++) {
71                 left = cplex.sum(left,cplex.prod(1,x[tsNode.id*numOfdestination+i]));
72             }
73             cplex.addEq(left,tsNode.quantity);
74         }else {
75             for(int i = 0;i<numOfPlants;i++) {
76                 left = cplex.sum(left,cplex.prod(1,x[tsNode.id+i*numOfdestination]));
77             }
78             cplex.addEq(left,tsNode.quantity);
79         }
80     }
81 }
82 /**
83  * This method is to solve model
84  * @return values of objective function and decision variables
85  * @throws IloException

```

```

85  */
86  public void Solve(List<transportation_node> transportationNodeList) throws IloException{
87      cplex.setOut(null);
88      if (cplex.solve()) {
89          cplex.exportModel("111.lp");
90          objectiveValue = cplex.getObjValue();
91          System.out.println(" 最小运费为: " + objectiveValue);
92          for (int i = 0; i < numOfPlants; i++) {
93              for (int j = 0; j < numOfdestination; j++) {
94                  System.out.print(transportationNodeList.get(i).nodeName+" to
95                  ↵ "+transportationNodeList.get(numOfPlants+j).nodeName+" : ");
96                  System.out.println(cplex.getValue(x[i*numOfdestination+j]));
97              }
98          }
99      }
100  }

```

4.3.5 主函数 main

```

Code
1  package transportation_problem;
2
3  import java.io.IOException;
4  import java.util.List;
5  import ilog.concert.IloException;
6
7  public class Main {
8      public static void main(String[] args) throws IOException, IloException {
9          Readfile file = new Readfile("transportation_node.txt","transportation_relation.txt");
10         List<transportation_node> transportationNodeList = file.transportationNodeList;
11         transportation_relation transportationRelation = file.transportationRelation;
12
13         model_transportation model = new model_transportation();
14         model.bulidModel(transportationNodeList, transportationRelation);
15     }
16 }

```

4.4 求解结果

```
最小运费为: 297800.0
Sunnyvale to Amarillo : 0.0
Sunnyvale to Teaneck : 0.0
Sunnyvale to Chicago : 0.0
Sunnyvale to Sioux Falls : 45.0
Dublin to Amarillo : 42.0
Dublin to Teaneck : 78.0
Dublin to Chicago : 0.0
Dublin to Sioux Falls : 0.0
Bangkok to Amarillo : 38.0
Bangkok to Teaneck : 0.0
Bangkok to Chicago : 47.0
Bangkok to Sioux Falls : 10.0
```

Minimize
obj: 250 x11 + 420 x12 + 380 x13 + 280 x14 + 1280 x21 + 990 x22 + 1440 x23
+ 1520 x24 + 1550 x31 + 1420 x32 + 1660 x33 + 1730 x34
Subject To
c1: x11 + x12 + x13 + x14 = 45
c2: x21 + x22 + x23 + x24 = 120
c3: x31 + x32 + x33 + x34 = 95
c4: x11 + x21 + x31 = 80
c5: x12 + x22 + x32 = 78
c6: x13 + x23 + x33 = 47
c7: x14 + x24 + x34 = 55
End

Figure 4.4.1: 求解结果

这样就完成了，感谢大家的阅读。

完整 Project 文件请关注运小筹公众号，回复【运输问题】获取。

Chapter 5

2020-11-26: 两阶段启发式算法求解带 时间窗车辆路径问题（附 Java 代码）

作者：何彦东，清华大学，深圳国际研究生院 (博士后) 邮箱：ydhe602@163.com

CSDN 主页：

<https://blog.csdn.net/dongpangpang88>

5.1 文档阅读说明

本文档是作者利用 Java 编程语言编写的启发式算法求解带时间窗车辆路径问题。包括两大部分：一是带时间窗车辆路径问题描述；二是编程算法求解代码及说明（并附录 Solomon 经典案例求解结果）。

5.2 VRPTW 定义

标准的带时间窗车辆路径问题定义如下：

1. 车辆从某一车场出发，服务后返回该车场；
2. 每个顾客有且被一辆车服务一次；
3. 服务有时间窗限制，车辆可早到，但不允许晚到；
4. 车辆有最大旅行时间和容量限制；
5. 所使用的车辆数不超过最大可使用车辆数量。

基于上述问题，本文档在 IntelliJ IDEA 平台利用 Java 语言进行编程，求解该车辆路径问题。主要利用贪心策略求解。核心程序的各个部分如下所示：

5.3 Java 数据读取（附 Java 代码）

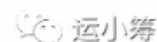
原始数据格式：以 Solomon 数据案例 C101_25 为例：

```
C101

VEHICLE
NUMBER  CAPACITY
 25      200

CUSTOMER
CUST NO. XCOORD. YCOORD. DEMAND  READY TIME DUE DATE  SERVICE  TIME

 0   40    50    0    0   1236    0
 1   45    68   10   912   967   90
 2   45    70   30   825   870   90
 3   42    66   10    65   146   90
 4   42    68   10   727   782   90
```



上述是 Solomon 经典案例的某一个案例的原始数据，各行代表的意思如下：** 第一行 **：案例编号 ** 第二行 **：空行（注意读取数据时需要跳过，下同）** 第三行 **：车辆信息 ** 第四行 **：车辆数量每辆车装载能力 ** 第五行 **：空行 ** 第六行 **：客户信息 ** 第七行 **：客户编码 X 坐标 Y 坐标需求量开始服务时间结束服务时间服务时间 ** 第八行之后 **：按照第七行顺序所列的具体客户信息

读取该数据的说明如下：

算法运行的前提是需要输入客户、车辆等的的数据，那么如何读取这些数据呢？下面以读取 Solomon 数据案例 C101_25 为例进行说明。代码及其代码说明如下

注意：在上面的读取数据代码中，`String [] parts = line.split(" s+");` 语句需要注意你的数据间隔是“空格”还是 ** “Tab” ** 键。这两个视觉效果是一样的。不过计算机能分辨差异。当初小编初学时曾在这坑里蹲了一天。没发现错在哪里。

5.4 两阶段启发式算法（附 Java 代码）

基于随机贪心策略的初始解生成（附 Java 代码）

在读取数据之后，就可以根据这些数据进行初始解的产生，本文档利用随机贪心策略生成初始解。在介绍随机贪心策略之前，先介绍下贪心策略生成初始解方法。

在贪心策略中，一般将未被访问的客户逐个插入路径中，并对每个插入成本进行排序，选择引起成本增加最少的客户，将其插入路径中。

但是我们在实验中发现，上述策略得到的初始解较差，主要是贪心策略是一种短视的策略。因此我们利用随机贪心策略。即在排序后，保留 N 个最小成本的插入策略，从这 N 个插入策略中随机选择一个策略，并将对应客户插入路径中。利用此方法得到的初始解，经过验证比单纯的贪心策略好很多，甚至对一些 30 个客户的案例都能直接得到最优解。

具体 Java 代码如下：

5.5 邻域搜索策略（附 Java 代码）

在进行迭代局部搜索时，可以根据问题特性设计各种邻域搜索策略。这是由于仅仅利用一种策略，算法很容易陷入局部最优，不能得到更好的解。比如最简单的贪心搜索策略等，容易在迭代一定次数后陷入局部最优。因此，在局部搜索时，一边会设计多种邻域策略进行搜索。比如随机地交换两条路径中某个（也可以 2 个或 3 个等）客户，得到 2 个新的路径，等都是些有效的邻域搜索策略。

当然，还有许多复杂的邻域搜索策略，在这里就不一一详细介绍，感兴趣的朋友们可以参考近些年来应用较多的自适应大规模邻域搜索算法（简称：ALNS），这个算法里面一般会设计较多的邻域搜索策略。当然还有一些比较经典的算法，如遗传算法（GA）、变邻域搜索算法（VNS）、** 迭代局部搜索（ILS）** 等等，里面都会介绍一些邻域搜索策略。当然具体利用哪一种策略，除了参考已有的邻域搜索策略外，自己也需要根据自己研究问题的特点，设计一些针对性的搜索策略，只有这样，设计的算法求解效果才会更好。

不说废话了，下面介绍一种 K-随机贪心局部搜索策略对 Section 4 得到的初始解进行优化。该策略的具体步骤如下（以一代 N 个客户为例）

：(1) 针对某个解，随机选择一个客户，将该客户从这个解中删除；(2) 将该客户逐次插入解中每条路径的每个位置，每一个位置都会产生一个插入成本，计算该插入成本；(3) 将上述插入成本递增排序，选择前 K 个插入位置；(4) 从这 K 个中随机选择一个（并不是选择插入成本最少的），将客户插入这个位置。(5) 重复 (1) - (4) N 次；完成一次迭代过程。

下面针对每一步的具体 Java 代码如下：(1) 随机选择一组客户：

(5) 至此结束一次迭代过程。

备注：上面介绍的是根据贪心策略改进的一个简单策略，与单纯的贪心策略相比，加入随机因素后，使得算法避免过早的陷入局部最优。当然，仅靠这一种局部搜索策略，对小规模问题或许求解效果还可以。但是对于大规模问题，求解效果或许就不那么友好了。需要针对问题设计更多局部搜索策略。而对于算法的终止规则，也不能单纯的以时间或者代数来规定。可以使用模拟退火原则、延迟爬山策略等进行规定算法的终止。

5.6 结果展示

根据上述代码, 求解 Solomon 一个有 25 个客户的经典算例 C_101_25, 其解(具体路径)如下所示: 总的距离为: 764.69

```
[[5, 18, 6, 13], [12, 9, 3, 1], [2, 21, 22, 4, 25], [14, 7, 8, 17], [11, 19, 10], [16, 24], [15, 20], [23]]
```

在上述解的表示中, 每个 [] 里面包含车辆访问客户的顺序, 不包含车场序号。

上述代码中只包含了初始解的部分, 此外, 还可以利用各种邻域搜索来改善它, 由于篇幅所限及庞大的代码量, 在这里并未全部列出, 仅给出最终结果: 改善后的解总距离为: 618.25 (这个应该是最优解的值, 对于小规模 VRP 问题, 好的启发式算法一般也都会得到最优解), 解为:

```
[[11, 19, 10], [14, 15, 13], [7, 8, 17], [5, 16, 6], [12, 9, 20, 1], [2, 21, 3, 24, 25], [23, 22, 4], [18]]
```

下次有时间小编将详细说明下各种常用的邻域搜索策略以及使用 Java 调用 CPLEX 求解 VRPTW 问题。

Chapter 6

2020-11-29: 手把手教你用 Python 调用 Gurobi 求解 VRPTW

作者：夏旻，清华大学，工业工程系/深圳国际研究生院（硕士在读）
刘兴禄，清华大学，清华伯克利深圳学院（博士在读）

在接触 Gurobi 之前，一直使用 Java 语言调用 cplex 求解数学模型，这段时间在 [师兄] (<https://blog.csdn.net/HsinglukLiu>) 的指点下，学习了使用 python 调用 Gurobi 的一些基础操作，感叹实在是太简易了。在此分享一个求解 Vrptw 问题的小例子。

6.1 带时间窗的车辆路径规划问题 (Vrptw)

对于 Vrptw 问题来说，数学模型主要由以下部分组成。首先我们定义一些相关参数，一个图可以表示为 $G(V, A)$ ，其中 $V = \{0, 1, \dots, n, n+1\}$ 为图中所有点的集合， A 为图中所有弧的集合，有 $(i, j) \in A, \forall i, j \in V, i \neq j$ 。弧 (i, j) 的单位运输费用为 c_{ij} ，运输时间为 t_{ij} ，每个客户点的需求为 q_{ij} ，可服务的时间窗为 $[e_i, l_i]$ ，服务时长为 $serv_i$ 。令车辆的集合为 K ，每辆车的最大载重为 $Q_k, \forall k \in K$ 。决策变量为 x_{ij}^k ，代表第 k 辆车是否服务了弧 (i, j) ； s_i 为客户点 i 开始被服务的时间。

接下来构建数学模型。目标函数为最小化运输成本：

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \quad (6.1.1)$$

约束一让车辆驶出仓库 (depot):

$$\sum_{(0,j) \in A} x_{0j}^k = 1, \quad \forall k \in K \quad (6.1.2)$$

约束二为流平衡 (除去 depot 之外的点):

$$\sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = 0, \quad \forall k \in K, i \in V \setminus \{s, t\} \quad (6.1.3)$$

约束三让车辆驶回 depot:

$$\sum_{(i,n+1) \in A} x_{i,n+1}^k = 1, \quad \forall k \in K \quad (6.1.4)$$

约束四保证每个客户点都被服务:

$$\sum_{k \in K} \sum_{(i,j) \in A} x_{ij}^k = 1, \quad \forall i \in V \setminus \{s, t\} \quad (6.1.5)$$

约束五保证被服务的相邻节点开始服务时间的大小关系 (去回路):

$$s_i + t_{ij} + serv_i - M(1 - x_{ij}) \leq s_j, \quad \forall (i, j) \in A \quad (6.1.6)$$

约束六保证不违反客户的时间窗:

$$e_i \leq s_i \leq l_i, \quad \forall i \in V \setminus \{s, t\} \quad (6.1.7)$$

约束七保证不违反车辆的载重约束:

$$\sum_{(i,j) \in A} x_{ij}^k q_i \leq Q_k, \quad \forall k \in K \quad (6.1.8)$$

最后是决策变量的取值约束:

$$x_{ij}^k \in \{0, 1\} \quad \forall (i, j) \in A, k \in K \quad (6.1.9)$$

$$s_i \geq 0, \quad \forall i \in V \setminus \{s, t\} \quad (6.1.10)$$

那么 (6.1.1)-(6.1.10) 式就组成了 Vrptw 问题的数学模型。

6.2 python 调用 Gurobi 求解 Vrpw

6.2.1 首先我们定义一下需要用到的参数

Code

```
1 class Data:
2     customerNum = 0
3     nodeNum     = 0
4     vehicleNum  = 0
5     capacity    = 0
6     cor_X       = []
7     cor_Y       = []
8     demand      = []
9     serviceTime = []
10    readyTime    = []
11    dueTime      = []
12    disMatrix    = [[]]
```

6.2.2 定义一个读取数据的函数，并对节点之间的距离进行计算

Code

```
1 # function to read data from .txt files
2 def readData(data, path, customerNum):
3     data.customerNum = customerNum
4     data.nodeNum = customerNum + 2
5     f = open(path, 'r')
6     lines = f.readlines()
7     count = 0
8     # read the info
9     for line in lines:
10         count = count + 1
11         if (count == 5):
12             line = line[:-1].strip()
13             str = re.split(r" +", line)
14             data.vehicleNum = int(str[0])
15             data.capacity = float(str[1])
16         elif (count >= 10 and count <= 10 + customerNum):
17             line = line[:-1]
18             str = re.split(r" +", line)
19             data.cor_X.append(float(str[2]))
20             data.cor_Y.append(float(str[3]))
21             data.demand.append(float(str[4]))
22             data.readyTime.append(float(str[5]))
23             data.dueTime.append(float(str[6]))
24             data.serviceTime.append(float(str[7]))
25
26     data.cor_X.append(data.cor_X[0])
27     data.cor_Y.append(data.cor_Y[0])
```

```

28     data.demand.append(data.demand[0])
29     data.readyTime.append(data.readyTime[0])
30     data.dueTime.append(data.dueTime[0])
31     data.serviceTime.append(data.serviceTime[0])
32
33     # compute the distance matrix
34     data.disMatrix = [[0] * data.nodeNum for p in range(data.nodeNum)] # 初始化距离矩阵的维度, 防
    ↪ 止浅拷贝
35     # data.disMatrix = [[0] * nodeNum] * nodeNum]; 这个是浅拷贝, 容易重复
36     for i in range(0, data.nodeNum):
37         for j in range(0, data.nodeNum):
38             temp = (data.cor_X[i] - data.cor_X[j]) ** 2 + (data.cor_Y[i] - data.cor_Y[j]) ** 2
39             data.disMatrix[i][j] = math.sqrt(temp)
40             temp = 0
41
42     return data

```

6.2.3 读取数据, 并定义一些参数

Code

```

1     data = Data()
2     path = 'c101.txt' //读取 Solomon 数据集
3     customerNum = 20 //设置客户数量
4     readData(data, path, customerNum)
5     data.vehicleNum = 10 //设置车辆数
6     printData(data, customerNum)
7     BigM = 100000 //定义一个极大值

```

6.2.4 调用 gurobi 进行模型的建立与求解

Code

```

1     x = {}
2     s = {} //定义字典用来存放决策变量
3
4     ##### <font size=4> 根据式 (9) 定义决策变量, 并加入模型当中:
5
6     for i in range(data.nodeNum):
7         for k in range(data.vehicleNum):
8             name = 's_' + str(i) + '_' + str(k)
9             s[i,k] = model.addVar(0
10                                     , 1500
11                                     , vtype= GRB.CONTINUOUS
12                                     , name= name) //定义访问时间为连续变量
13
14     for j in range(data.nodeNum):
15         if(i != j):
16             name = 'x_' + str(i) + '_' + str(j) + '_' + str(k)
17             x[i,j,k] = model.addVar(0

```

```

18                                     , vtype= GRB.BINARY
19                                     , name= name) //定义是否服务为 0-1变量
20
21 ##### <font size=4> 根据式 (1) 定义目标函数, 并加入模型当中:
22
23
24 # 首先定义一个线性表达式
25 obj = LinExpr(0)
26 for i in range(data.nodeNum):
27     for k in range(data.vehicleNum):
28         for j in range(data.nodeNum):
29             if(i != j):
30                 //将目标函数系数与决策变量相乘, 并进行连加
31                 obj.addTerms(data.disMatrix[i][j], x[i,j,k])
32 # 将表示目标函数的线性表达式加入模型, 并定义为求解最小化问题
33 model.setObjective(obj, GRB.MINIMIZE)
34
35 ##### <font size=4> 根据式 (2)~(8) 定义决策变量, 并加入模型当中:
36 # 约束一 (vehicle_depart):
37 for k in range(data.vehicleNum):
38     //同样先定义一个线性表达式
39     lhs = LinExpr(0)
40     for j in range(data.nodeNum):
41         if(j != 0):
42             //约束系数与决策变量相乘
43             lhs.addTerms(1, x[0,j,k])
44     //将约束加入模型
45     model.addConstr(lhs == 1, name= 'vehicle_depart_' + str(k))
46
47 # 约束二 (flow_conservation):
48 for k in range(data.vehicleNum):
49     for h in range(1, data.nodeNum - 1):
50         expr1 = LinExpr(0)
51         expr2 = LinExpr(0)
52         for i in range(data.nodeNum):
53             if (h != i):
54                 expr1.addTerms(1, x[i,h,k])
55
56         for j in range(data.nodeNum):
57             if (h != j):
58                 expr2.addTerms(1, x[h,j,k])
59
60         model.addConstr(expr1 == expr2, name= 'flow_conservation_' + str(i))
61         expr1.clear()
62         expr2.clear()
63
64 # 约束三 (vehicle_enter):
65
66 for k in range(data.vehicleNum):
67     lhs = LinExpr(0)

```

```

68     for j in range(data.nodeNum - 1):
69         if(j != 0):
70             lhs.addTerms(1, x[j, data.nodeNum-1, k])
71     model.addConstr(lhs == 1, name= 'vehicle_enter_' + str(k))
72
73     # 约束四 (customer_visit):
74
75     for i in range(1, data.nodeNum - 1):
76         lhs = LinExpr(0)
77         for k in range(data.vehicleNum):
78             for j in range(1, data.nodeNum):
79                 if(i != j):
80                     lhs.addTerms(1, x[i,j,k])
81         model.addConstr(lhs == 1, name= 'customer_visit_' + str(i))
82
83     # 约束五 (time_windows):
84
85     for k in range(data.vehicleNum):
86         for i in range(data.nodeNum):
87             for j in range(data.nodeNum):
88                 if(i != j):
89                     model.addConstr(s[i,k] + data.disMatrix[i][j] + data.serviceTime[i] - s[j,k]- BigM +
90                                     ↪ BigM * x[i,j,k] <= 0 , name= 'time_windows_')
91
92     # 约束六 (ready_time and due_time):
93
94     for i in range(1,data.nodeNum-1):
95         for k in range(data.vehicleNum):
96             model.addConstr(data.readyTime[i] <= s[i,k], name= 'ready_time')
97             model.addConstr(s[i,k] <= data.dueTime[i], name= 'due_time')
98
99     # 约束七 (capacity_vehicle):
100
101     for k in range(data.vehicleNum):
102         lhs = LinExpr(0)
103         for i in range(1, data.nodeNum - 1):
104             for j in range(data.nodeNum):
105                 if(i != j):
106                     lhs.addTerms(data.demand[i], x[i,j,k])
107             model.addConstr(lhs <= data.capacity, name= 'capacity_vehicle' + str(k))
108
109     # 求解模型 (solve) 并输出解:
110
111     model.optimize()
112     print("\n\n-----optimal value-----")
113     print(model.ObjVal)
114
115     for key in x.keys():
116         if(x[key].x > 0 ):
117             print(x[key].VarName + ' = ', x[key].x)

```

```
117  
118 # 导出模型:  
119  
120 model.write('VRPTW.lp')
```

求解结果（部分）:

```
Solution count 1: 235.58  
  
Optimal solution found (tolerance 1.00e-04)  
Best objective 2.355802911741e+02, best bound 2.355802911741e+02, gap 0.0000%  
  
-----optimal value-----  
235.58029117413327  
x_0_20_0 = 1.0  
x_0_13_1 = 1.0  
x_0_7_2 = 1.0  
x_0_5_3 = 1.0  
x_0_10_4 = 1.0  
x_1_21_2 = 1.0  
x_2_1_2 = 1.0  
x_3_21_3 = 1.0  
x_4_2_2 = 1.0
```

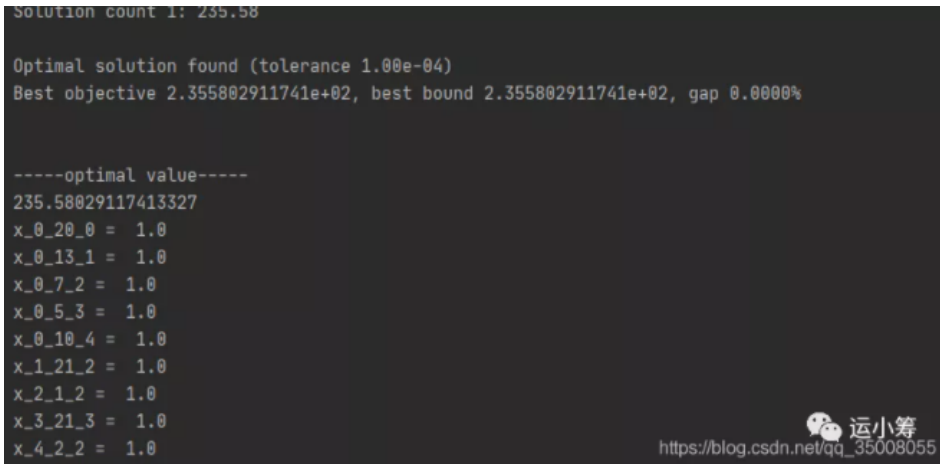


Figure 6.2.1: 求解结果

这样就完成了，感谢大家的阅读。

完整 Project 文件请关注运小筹公众号，回复【Vrptw】获取。

@[TOC](优化 |)

Chapter 7

2020-12-02: 单纯形法及其 Java 实现

作者:

王乃禹, 清华大学, 深圳国际研究生院 (硕士在读);

曾文佳, 清华大学, 工业工程系/深圳国际研究生院 (硕士在读)

本文中的课件来自清华大学深圳国际研究生院, 物流与交通学部张灿荣教授《高级运筹学》课程。张灿荣教授个人主页: <https://www.sigs.tsinghua.edu.cn/zcr/list.htm>

7.1 单纯形法 (Simplex Method)

单纯形法是求解线性规划问题的一个经典算法, 是一种直接、快速的搜索最小值方法, 其优点是收敛速度快, 适用面较广。很多同学肯定对使用单纯形表解决线性规划问题并不陌生, 但是每次迭代时对系数矩阵进行变换, 其中某些系数的变换是不必要的。通过矩阵变换的形式表达单纯形法的过程可以使计算过程更加简洁, 编程实现也更加方便。

Linear programming (review)

Important formulas

$$\begin{aligned}
 \max z &= (c_B, c_N)(x_B, x_N)^T \\
 \text{s.t.} \quad (B, N)(x_B, x_N)^T &= b^T \\
 (B, N)(x_B, x_N)^T &= b^T \\
 Bx_B^T + Nx_N^T &= b^T \\
 B^{-1}Bx_B^T + B^{-1}Nx_N^T &= B^{-1}b^T \\
 \Rightarrow x_B^T &= B^{-1}b^T - B^{-1}Nx_N^T \\
 z &= (c_B, c_N)(x_B, x_N)^T \\
 &= c_Bx_B^T + c_Nx_N^T \\
 &= c_B(B^{-1}b^T - B^{-1}Nx_N^T) + c_Nx_N^T \\
 &= c_BB^{-1}b^T + (c_N - c_BB^{-1}N)x_N^T
 \end{aligned}$$

Figure 7.1.1: Simplex

该方法涉及的重要公式推导过程如上图所示。其中 x_B 表示基变量， x_N 表示非基变量， B 和 N 分别代表基变量与非基变量的约束系数，而 c_B 和 c_N 则分别表示其目标函数的系数。由目标函数 z 的表达式可以看出，由于基变量最终取值为零，若其系数 $c_N - c_BB^{-1}N$ 大于零，则目标函数值仍有提升的空间未得到最优解。‘故 $c_N - c_BB^{-1}N$ 正是检验数。’

7.1.1 单纯形法伪代码

下图给出了单纯形法实现的伪代码。注意，此处解决的是一个最大化问题。

Algorithm	Revised Simplex Method
1:	Initialization:
2:	初始化 $N, B, c_N, c_B, x_N, x_B, b$
3:	while True do
4:	检验数 $= c_N - c_B B^{-1} N$
5:	if 存在检验数 > 0 then :
6:	选取非正检验数中最小检验数对应的非基变量作为进基变量。
7:	计算 $(B^{-1}b^T)/(B^{-1}N^{xj})$
8:	if 不存在大于 0 的比值 then
9:	存在无界解, 输出
10:	Break
11:	else if 存在两个以上相同的最小比值 then
12:	退化问题: 在所有检验数 > 0 的非基变量中, 选下标最小的作为进基变量
13:	进行出基、进基操作, 更新 N, B, c_N, c_B, x_N, x_B
14:	else
15:	选择最大比值者作为进基变,
16:	进行出基、进基操作, 更新 N, B, c_N, c_B, x_N, x_B
17:	end if
18:	else if 存在检验数 $= 0$ 且其它检验数 < 0 then
19:	该问题具有无穷多最优解, 输出
20:	Break
21:	else
22:	该问题有唯一可行解, 计算得到最优解 $x^* = B^{-1}b^T$ 和最优目标函数值 $z^* = c_B x^*$
23:	Break
24:	end if
25:	end while

 运小筹
https://blog.csdn.net/weixin_44591773

Figure 7.1.2: Simplex Algorithm

7.2 单纯形法的 Java 实现

接下来, 将详细介绍单纯形法的 Java 实现。

7.2.1 变量符号定义

计算中用到的变量符号定义, 具体代码为:

```
Code
1  double[] c = new double[c_.size()+1]; //定义目标函数的系数矩阵 c
2  double[][] A = new double[A_.size()][A_.get(0).size()+1]; //定义约束的系数矩阵 A
3  double[] b = new double[b_.size()]; //定义约束值的矩阵 b
4  int per = 0; // 需要添加人工变量的约束个数
5  double M = 1000000; // 大 M 法中的“M”
6  int[] x_B_Num = new int[A.length]; //定义存储基变量标号的一维数组
7  int[] x_N_Num = new int[A[0].length - A.length]; //定义存储非基变量的一维数组
8
9  double[] x_B = new double[A.length]; //定义基变量值的一维数组
10 double[] enter_Jud = new double[x_N_Num.length]; //定义存储进基变量的判断数
11 double[] exit_Jud = new double[x_B_Num.length]; //定义存储出基变量的判断数
12
13 double[] c_B = new double[x_B_Num.length]; //定义存储基变量的目标函数系数的向量
14 double[] c_N = new double[x_N_Num.length]; //定义存储非基变量的目标函数系数的向量
```

```

15 double[] Y = new double[c_B.length]; //定义 c_B*B 逆
16 double[] F = new double[x_B_Num.length]; //定义 B 逆 *P
17 double[][] B = new double[A.length][A.length]; // 定义存储基变量在约束表达式系数的矩阵
18
19 int u = 0; //定义循环次数
20 int en = 0; //定义当前的进基序号
21 int ex = 0; //定义当前的出基序号
22 int[] ee = new int[2]; //定义当前的出基序号和判断是否退化

```

7.2.2 模型存储与读入

用 txt 文件存储模型。以一个简单的例子，进行说明。如下图，图左为 txt 文件截图，每部分数据用空格隔开，第一部分表示 c 中每个元素的取值，即目标函数中变量的系数，以逗号隔开；第二部分表示 A 中每个元素的取值，每行表示每条约束左边项的变量系数，以逗号隔开；第三部分表示每条约束的符号，以逗号隔开，0 对应等于，-1 对应小于等于，1 对应大于等于；第四部分表示 b 中每个元素的取值，即每条约束右端项的值。该 txt 文件表示的模型如图右所示。将 txt 文件中的数据读入，并用 ArrayList 分别进行存储。

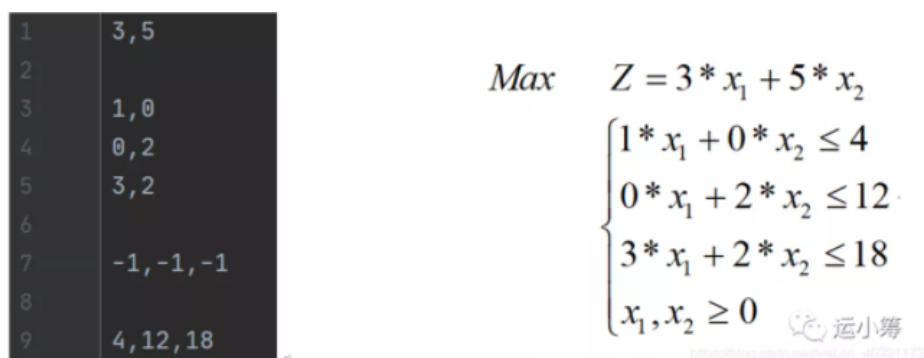


Figure 7.2.1: Simplex Algorithm

7.2.3 模型的标准型转化

对于小于等于的约束，每条约束的左端需加上一个非负的松弛变量。对于大于等于的约束和等于的约束，每条约束的左端需减去一个非负的剩余变量，加上一个非负的人工变量。用大 M 法求解带人工变量的线性规划问题。初始给定一个较大的值 M ，更新标准化模型中的 A, c, b 。

7.2.4 单纯形法主体

首先计算 $Y = c_B * B^{-1}$ ，然后计算每个非基变量的检验数 $c_N - c_B B^{-1} N$ 。

```

Code
1  try {
2      x_B = MatrixOperation.matrix_Multiplication(MatrixOperation.matrix_Inverse(B), b);
3      Y = MatrixOperation.matrix_Multiplication(c_B, MatrixOperation.matrix_Inverse(B));
4  } catch (Exception e) {
5      e.printStackTrace();
6  }
7  for (int i = 0; i < enter_Jud.length; i++) {
8      try {
9          enter_Jud[i] = MatrixOperation.oneD_Matrix_Multiplication(Y,
10             ↪ MatrixOperation.get_Matrix_Col(A, x_N_Num[i] - 1)) - c_N[i];
11      } catch (Exception e) {
12          e.printStackTrace();
13      }
14  }

```

根据每个非基变量的检验数进行判断。

- 情况 1: 所有检验数均大于等于 0, 并且其中存在一个非基变量的检验数为 0。那么此问题有无穷多最优解, 退出循环。
- 情况 2: 所有检验数均大于等于 0, 并且基变量中还有某个非零的人工变量。那么此问题无解, 退出循环。
- 情况 3: 所有检验数均大于等于 0, 且不满足上述情况。说明以求得最优解, 退出循环。
- 情况 4: 如果以上条件均不满足, 则选取非正检验数中最小检验数对应的非基变量作为‘进基变量’。

```

Code
1  //判断是否有无穷多解
2  if (jud_If_Inf(enter_Jud)){
3      System.out.println(" 无穷多最优解");
4      type_Solution = 1;
5      break;
6  }
7
8  //判断是否无解
9  int noS = 0;
10 for(int i = 0; i < enter_Jud.length; i++){
11     if(enter_Jud[i] >= 0){
12         noS = noS + 1;
13     }
14 }
15 if(noS == enter_Jud.length){ //如果检验数全部大于等于 0
16     for(int i = 0; i < x_B_Num.length; i++){ //检验基变量是否还有非 0 的人工变量
17         for(int j = 0; j < per; j++){
18             if(x_B_Num[i] == c.length-j*2){
19                 System.out.println(" 无解");
20                 type_Solution = 3;

```

```

21         break;
22     }
23 }
24 }
25
26 }
27
28 en = get_Minus_Min(enter_Jud); //得到进基变量在 x_N_Num 中的序号。如果返回-1 则达到最优解
29 //判断是否达到循环停止条件
30 if (en == -1) {
31     break;
32 }

```

然后计算每个基变量的判别数： $(B^{-1}b^T)/(B^{-1}N^{xj})$ 。

Code

```

1  try {
2      double[] ii = MatrixOperation.get_Matrix_Col(A, x_N_Num[en] - 1);
3      F = MatrixOperation.matrix_Multiplication(MatrixOperation.matrix_Inverse(B),
4          ↪ MatrixOperation.get_Matrix_Col(A, x_N_Num[en] - 1));
5  } catch (Exception e) {
6      e.printStackTrace();
7  }
8  for (int i = 0; i < exit_Jud.length; i++) {
9      if (F[i] == 0) {
10         exit_Jud[i] = -1;
11     } else {
12         if(F[i] == 0){
13             exit_Jud[i] = 1000;
14         }
15         else {
16             if(x_B[i] == 0 && F[i] < 0){ //处理特殊情况，当出现此情况时，i 不可作为出基变量
17                 exit_Jud[i] = 1000;
18             }
19             else {
20                 exit_Jud[i] = x_B[i] / F[i];
21             }
22         }
23     }
24 }

```

根据每个基变量的判别数进行判断。

- 情况 1: 如果满足每个基变量判别数均小于 0: 那么此问题有无界解, 退出循环。
- 情况 2: 若此时存在两个以上相同的非负最小判别数, 则出现退化现象, 用勃兰特规则重新确定 ‘进基变量和出基变量’。
- 情况 3: 如果不满足上述情况, 则选取非负最小判别数对应的基变量作为 ‘出基变量’。

Code

```

1 //判断是否有无界解
2 if(jud>If_Endless(enter_Jud,exit_Jud)){
3     System.out.println(" 无界解");
4     type_Solution = 2;
5     break;
6 }
7
8 ee = get_Positive_Min(exit_Jud); //ee[0] 出基变量在 x_B_Num 中的序号。ee[1] 若不等于 0, 则出现退化现象
9 ex = ee[0];
10
11 //判断是否出现退化现象
12 if(ee[1] == 0){ //如果没有退化现象
13 }
14 else{ //如果出现退化现象 重新确定进基和出基变量
15     en = get_Re_Minus_Min(enter_Jud, x_N_Num);
16     ex = get_Re_Positive_Min(exit_Jud, x_B_Num);
17 }

```

‘更新基变量与非基变量’。

Code

```

1 int exchange = 0;
2 exchange = x_B_Num[ex];
3 x_B_Num[ex] = x_N_Num[en];
4 x_N_Num[en] = exchange;
5 for(int i = 0; i < B.length; i++){
6     for(int j = 0; j < B[0].length; j++){
7         double oo = A[i][x_B_Num[j]-1];
8         B[i][j] = A[i][x_B_Num[j]-1];
9     }
10 }
11 for(int i = 0; i < c_B.length; i++){
12     c_B[i] = c[x_B_Num[i]-1];
13 }
14 for(int i = 0; i < c_N.length; i++){
15     c_N[i] = c[x_N_Num[i]-1];
16 }

```

7.2.5 输出结果

最后, 根据解的类型, 输出结果。

Code

```

1 if(type_Solution == 0){ //若为唯一最优解, 则输出最优解和目标函数
2     //输出最终结果
3     u = u + 1;
4     System.out.printf(" 第%d 次迭代结果 (最终结果): ",u);

```

```
5      System.out.println("");
6      System.out.println("[基变量]");
7      for(int i = 0; i < x_B.length; i++){
8          System.out.print(x_B_Num[i]+" ");
9          System.out.println(x_B[i]);
10     }
11     System.out.print("[目标函数] ");
12     try{
13         System.out.println(MatrixOperation.oneD_Matrix_Multiplication(c_B,x_B));
14     }catch (Exception e){
15         e.printStackTrace();
16     }
17 }
18 else if(type_Solution == 1){ //若为无穷最优解，则只输出目标函数
19     System.out.print("[目标函数] ");
20     try{
21         System.out.println(MatrixOperation.oneD_Matrix_Multiplication(c_B,x_B));
22     }catch (Exception e){
23         e.printStackTrace();
24     }
25 }
```

完整 Project 文件请在后台回复【单纯形法 Java】获取。

参考文献

- [1] Niels Agatz, Paul Bouman, and Marie Schmidt. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981, 2018.
- [2] Paul Bouman, Niels Agatz, and Marie Schmidt. Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4):528–542, 2018.
- [3] Yong Sik Chang and Hyun Jung Lee. Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, 104:307–317, 2018.
- [4] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.
- [5] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.
- [6] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533 – 549, 1986. ISSN 0305-0548. doi: [https://doi.org/10.1016/0305-0548\(86\)90048-1](https://doi.org/10.1016/0305-0548(86)90048-1). URL <http://www.sciencedirect.com/science/article/pii/0305054886900481>. Applications of Integer Programming.
- [7] Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.
- [8] Fred Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990.
- [9] Andy M Ham. Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, 91:1–14, 2018.
- [10] Chase C Murray and Amanda G Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015.

- [11] Wayne L Winston and Jeffrey B Goldberg. *Operations research: applications and algorithms*, volume 3. Thomson/Brooks/Cole Belmont^ eCalif Calif, 2004.