

Codecraft2016 Linear Programming 解法小结

By 六院八队 小王

2016.11.16

1	前言	2
1.1	关于 LP 与 TSP.....	2
1.2	求解器.....	2
2	TSP 的 LP 解法.....	3
2.1	Gurobi 简介.....	3
2.2	LP 求解 TSP 数学模型	3
3	初赛（单条路）的 LP 解法.....	5
3.1	建模	5
3.2	一个例子.....	5
3	复赛/决赛（两条路）的 LP 解法.....	6
3.1	建模	6
3.2	一个例子.....	6
4	实验结果&结论.....	8

1 前言

1.1 关于 LP 与 TSP

TSP 有很多很多算法，不管是精确算法还是启发式算法。推荐看 William J. Cook 的《迷茫的旅行商 一个无处不在的计算机算法问题》一书，介绍了半个世纪以来各种各样的 TSP 算法。

书中第五章“线性规划”介绍了用 Linear Programming（简称 LP）方法解决 TSP 问题的历史进程，从 Simplex Algorithm（单纯形算法：20 世纪十大经典算法之一）到 Primal-Dual Algorithm（原始-对偶算法）等等。

在初赛时候就有好多人使用求解器，但惹来很多争议。虽然复赛时不让用了，但还是有研究的价值。

1.2 求解器

目前求解器已经越来越不局限于求解纯线性问题了，求解混合整数规划（MIP）和二次规划（QP）等也很棒。Gurobi7.0 版本甚至允许模型可以添加 And 和 Or 这种纯 non-linear 的限制条件（7.0 有 bug，官方立马改正后发布了 7.0.1），模型甚至还可以设置多目标（所以目标为线性）。

求解器分商业和开源，前者基本上完爆后者。具体评测数据请见：<http://plato.asu.edu/bench.html>。求解质量上：cplex \approx Gurobi>XPRESS>>其它。然而这三个求解器都是商业的，不过 cplex 和 Gurobi 都有学术完整版。

Cplex 申请：用校园邮箱在 <https://ibm.onthehub.com> 注册，验证邮箱，在网站上免费购买、下载、安装。

Gurobi 申请：在 Gurobi 官网用校园邮箱注册，验证邮箱，在网站上申请学术验证码（免费）。在 <http://www.gurobi.com/downloads/gurobi-optimizer> 下载软件，网络连接到校园网，打开软件，输入验证码、回车，验证校园网成功后即可使用。

2 TSP 的 LP 解法

2.1 Gurobi 简介

笔者主要使用的是 Gurobi，大家可也自行学习 cplex。学习 Gurobi 主要参考三个文件：Quick Start Guides（区分平台）、Example Tour、Reference Manual，下载地址：<http://www.Gurobi.com/documentation/7.0/>。

Quick Start Guides 中 8-14 章（C、C++、Java、.Net、Python、Matlab、R）分别用不同语言的接口讲解了两个例子：一个整数规划模型，一个网络流模型。如果没有任何使用求解器的经验，建议把其中某个语言的一章看完。

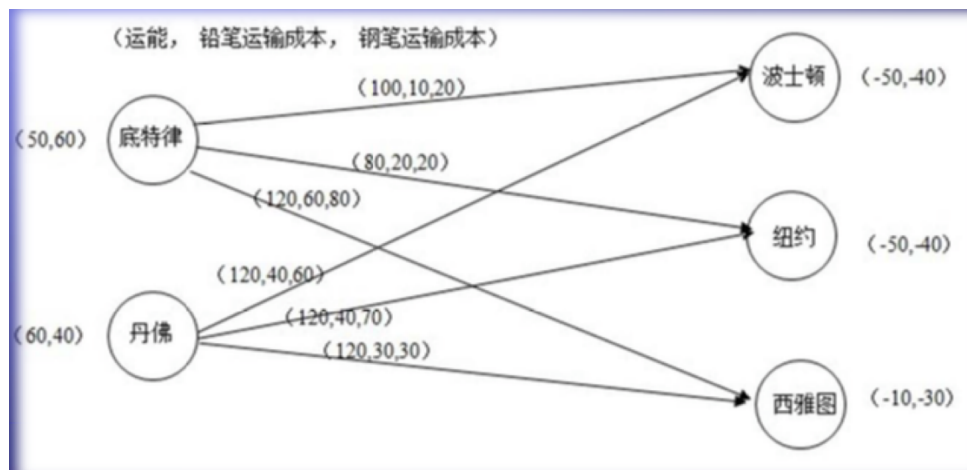
我使用的语言为 Python，简单易用、开发快，Gurobi 官方也推荐。

Quick Start Guides 中 12.1 Simple Python Example 讲解了一个最最简单的整数规划模型，如下图：

$$\begin{array}{llllll} \text{maximize} & x & + & y & + & 2z \\ \text{subject to} & x & + & 2y & + & 3z \leq 4 \\ & x & + & y & & \geq 1 \\ & & & & & x, y, z \text{ binary} \end{array}$$

Quick Start Guides 中 12.2 Python Dictionary Example 讲解了一个简单的网络流模型，如下图。该例子可参考中文 pdf: [Gurobi \(数学规划优化引擎\) 可视化建模环境—python 编程](#)，下载目录为：

https://github.com/Victoriayhk/future_net/tree/master/Linear-Programming。



2.2 LP 求解 TSP 数学模型

TSP 可以被规范化为一个整数规划问题：

minimize:

$$\sum_{i=1}^n \sum_{j \neq i, j=1}^n c_{ij} \cdot x_{ij}$$

subject to:

$$x_{ij} \in \{0, 1\}$$

$$i, j = 1, \dots, n$$

$$\sum_{i=1, i \neq j}^n x_{ij} = 1 \quad j = 1, \dots, n \quad (1)$$

$$\sum_{j=1, j \neq i}^n x_{ij} = 1 \quad i = 1, \dots, n \quad (2)$$

$$u_i \in Z \quad i = 1, \dots, n$$

$$u_i - u_j + n \cdot x_{ij} \leq n - 1 \quad 2 \leq i \neq j \leq n \quad (3)$$

用数字 $1, \dots, n$ 来标记 n 个城市。

u_i 是辅助变量：记录城市先后顺序。

c_{ij} 表示城市 i 到 j 的距离。

$x_{ij}=1$ 表示城市 i 到 j 相连， $x_{ij}=0$ 表示城市 i 到 j 不相连。

第一组等式①：out degrees=1

第二组等式②：in degrees=1

第三组约束③：只有等式组①和②的约束是不够的，因为会产生多个环（subtour）。而不等式组③就保证了只有一个环，证明请往下看。为了保证只有一个 tour，用到了辅助变量 u_i 。

证明不等式组③保证只有一个环：

如果某个 subtour 不包含城市 1，设这个 subtour 中城市为 $i, \dots, i + k - 1$ ，

令： $x_{i,i+1} = x_{i+1,i+2} = \dots = x_{i+k-1,i} = 1$

则有 k 个满足③的不等式：

$$\begin{aligned} u_i - u_{i+1} &\leq n - 1 \\ u_{i+1} - u_{i+2} &\leq n - 1 \\ &\dots \\ u_{i+k-1} - u_i &\leq n - 1 \end{aligned}$$

叠加后，得：

$$nk \leq (n - 1)k$$

产生矛盾，因此所有 subtour 必须包含城市 1，这些 subtour 又因为重合了城市 1 又合成为一个 tour。因此满足③的可行解只有一个 tour，得证。

3 初赛（单条路）的 LP 解法

3.1 建模

初赛题目区别于 TSP:

- (1) 有起点和终点，求的是一条路而不是环路
- (2) 某些节点为非必须节点，可以不经过

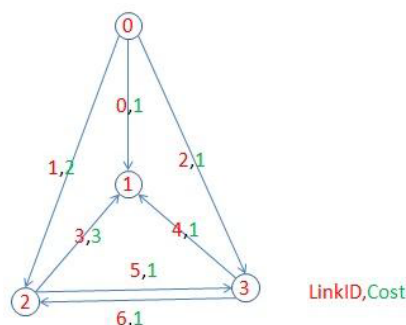
对上一章 TSP 模型做的修改:

- (1) 起点 s 和终点 t 单独做度的约束， s 只约束出度， t 只约束入度
- (2) 对于必须节点，约束不变：入度==出度==1

对于非必须节点，约束改成：入度==出度 ≤ 1

- (3) 对于辅助变量 u : $u_s = 1, u_t = n$ ，其它关于 u 的约束不变

3.2 一个例子



以官方提供的 demo 为例，如上图，下面是该模型示例代 future_net1.py，在终端输入 Gurobi.sh future_net1.py 即可。这个代码只是为了说明问题，真正的求解代码需要读入文件中的节点和边的信息，生成目标函数和约束条件。

```
future_net1.py
1 # coding:utf-8
2 # future_net1.py
3
4 from gurobipy import *
5
6 m = Model("TSP")
7
8 # cost/distance of node
9 c01 = 1
10 c02 = 2
11 c03 = 1
12 c21 = 3
13 c23 = 1
14 c31 = 1
15 c32 = 1
16
17 # create variables
18 x01 = m.addVar(vtype=GRB.BINARY)
19 x02 = m.addVar(vtype=GRB.BINARY)
20 x03 = m.addVar(vtype=GRB.BINARY)
21 x21 = m.addVar(vtype=GRB.BINARY)
22 x23 = m.addVar(vtype=GRB.BINARY)
23 x31 = m.addVar(vtype=GRB.BINARY)
24 x32 = m.addVar(vtype=GRB.BINARY)
25
26 # set objective
27 obj = c01*x01 + c02*x02 + c03*x03 + c21*x21 + c23*x23 + c31*x31 + c32*x32
28 m.setObjective(obj, GRB.MINIMIZE)
29
30 # degree constraint
31 m.addConstr(x01 + x02 + x03 == 1) # out degree of start node s
32 m.addConstr(x21 + x31 == 1) # in degree of end node t
33 m.addConstr(x02 + x32 == x21 + x23) # must node 2: out degree == in degree
34 m.addConstr(x21 + x23 == 1) # must node 2: out degree == 1
35 m.addConstr(x03 + x23 == x31 + x32) # must node 3: out degree == in degree
36 m.addConstr(x31 + x32 == 1) # must node 3: out degree == 1
37
38 # variable u constraint
39 n = 4
40 u0 = m.addVar(vtype=GRB.INTEGER, lb=1, ub=n)
41 u1 = m.addVar(vtype=GRB.INTEGER, lb=1, ub=n)
42 u2 = m.addVar(vtype=GRB.INTEGER, lb=1, ub=n)
43 u3 = m.addVar(vtype=GRB.INTEGER, lb=1, ub=n)
44 m.addConstr(u0 == 1)
45 m.addConstr(u1 == n)
46 m.addConstr(u0 - u1 + x01 * n <= n - 1)
47 m.addConstr(u0 - u2 + x02 * n <= n - 1)
48 m.addConstr(u0 - u3 + x03 * n <= n - 1)
49 m.addConstr(u2 - u1 + x21 * n <= n - 1)
50 m.addConstr(u2 - u3 + x23 * n <= n - 1)
51 m.addConstr(u3 - u1 + x31 * n <= n - 1)
52 m.addConstr(u3 - u2 + x32 * n <= n - 1)
53
54 # Compute optimal solution
55 m.optimize()
56
57 #print result
58 print "cost sum =", int(m.objVal), "\n"
59
60 print "x01 =", x01.x
61 print "x02 =", x02.x
62 print "x03 =", x03.x
63 print "x21 =", x21.x
64 print "x23 =", x23.x
65 print "x31 =", x31.x
66 print "x32 =", x32.x, "\n"
67
68 print "u0 =", u0.x
69 print "u1 =", u1.x
70 print "u2 =", u2.x
71 print "u3 =", u3.x
```

3 复赛/决赛（两条路）的 LP 解法

3.1 建模

复赛是求两条路径（主/备），而且有 2 个目标：

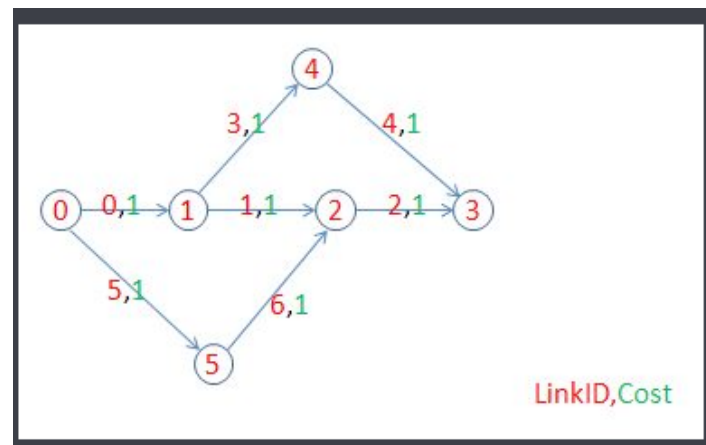
（1）主要目标：主备路径重复边个数最小

（2）次要目标：主备路径权值和最小

思路：单独求解主备路，图中每条边对应两个变量，分别是主备路径的使用与否（值为 0 或 1），相乘这两个变量，再相加所有乘积，就得到主要目标。先让模型计算出最优的主要目标，然后带入最优的主要目标值作为约束，再求次要目标。

3.2 一个例子

上面文字比较繁琐，直接看官方这个例子。



设图中六条边对应主路径是否使用的变量为： $x_{01}, x_{05}, x_{12}, x_{14}, x_{23}, x_{43}$ 。相应地，备选路径： $y_{01}, y_{05}, y_{12}, y_{14}, y_{23}, y_{43}$ 。这 12 个变量均为二进制变量。

主路径的必经节点为 0,3,1，非必须节点为 2,4,5。

备选路的必经节点为 0,3,2，非必须节点为 1,4,5。

模型如下：

minimize:

$$obj1 = x_{01} * y_{01} + x_{05} * y_{05} + x_{12} * y_{12} + x_{14} * y_{14} + x_{23} * y_{23} + x_{43} * y_{43}$$

$$obj2 = (x_{01} + x_{05} + x_{12} + x_{14} + x_{23} + x_{43}) + (y_{01} + y_{05} + y_{12} + y_{14} + y_{23} + y_{43})$$

subject to:

主路径：

$$\text{节点 0 出度: } x_{01} + x_{05} = 1$$

$$\text{节点 3 入度: } x_{23} + x_{43} = 1$$

$$\text{节点 1 入度和出度: } x_{01} = x_{12} + x_{14} = 1$$

$$\text{节点 2 入度和出度: } x_{12} + x_{52} = x_{23} \leq 1$$

$$\text{节点 4 入度和出度: } x_{14} = x_{43} \leq 1$$

节点 5 入度和出度: $x_{05} = x_{52} \leq 1$

备选路径:

节点 0 出度: $y_{01} + y_{05} = 1$

节点 3 入度: $y_{23} + y_{43} = 1$

节点 1 入度和出度: $x_{01} = x_{12} + x_{14} \leq 1$

节点 2 入度和出度: $y_{12} + y_{52} = y_{23} = 1$

节点 4 入度和出度: $y_{14} = y_{43} \leq 1$

节点 5 入度和出度: $y_{05} = y_{52} \leq 1$

注 1: 在 Gurobi7.0 中虽然可以设置多目标, 但是这些目标函数必须都是线性的。而本模型中 obj1 是非线性的, 所以不能使用 Gurobi7.0 多目标函数的特性。如果只是单目标就可以是多种形式的: 线性的、二次、分段等。所以只能先设置目标函数为 obj1, 将得到的最优 obj1 作为约束, 再设置目标函数为 obj2。

注 2: obj1 中的乘法运算其实是为了模拟逻辑&运算, Gurobi7.0 新增了 And 和 Or 约束函数, 比乘法要效率高一些。另外在某些模型中, 也可以用线性不等式模拟&运算, 比如 $c=a\&b$ 可以改写成 $a+b \leq 1+c$ (c 不能有其它约束, c 需要在目标函数中线性出现, 目标函数是 minimize 的)。

4 实验结果&结论

一共测试了 43 个 case，和基于 LKH 算法的结果进行对比。因为 Gurobi 很需要内存和计算，由于是笔记本跑的程序，内存和计算能力很差，如果在大内存高计算能力的机器上跑，结果可能差别很大。Gurobi 模型代码和实验结果下载：https://github.com/Victoriayhk/future_net/tree/master/Linear-Programming。其中 [source-pre/](#) 为单条路代码，[source-semi/](#) 为双路代码，[test_result.xlsx](#) 为测试结果数据。

可行解个数	Gurobi	基于 LKH 算法
重复边个数 obj1	23(全部最优)	40(22 个最优)
双路权值和 obj2	33(全部最优)	

注 1: Gurobi 是先后求解 obj1 和 obj2 的，而 LKH 是一起求。

注 2: 模型目标函数是 maximize 或 minimize，所以 Gurobi 的解都是最优解。

注 3: Gurobi 未求出解的 case 都是规模很大的 case，此时模型中的变量和约束很多，导致模型在可行时间内跑不出解。

注 4: LKH 未求出解的 3 个 case 都是稀疏图，而且有几个 case 即使求出解，但 obj1 非常差。而 Gurobi 却可以轻易求解这些 case。

注 5: 对于大规模模型输入，有可能会产生结果错误，因为 Gurobi 是使用确定位数表示所以变量（整数、连续型），所以有可能会出可接受的错误。比如说，一个 binary 的变量最后的取值可能是 0.0000011。

结论 1: LKH 适合求解稠密图，对于求解大规模 case 有很大优势，结果是近似解（但 LKH 很强大，很高概率得到最优解）。求解器适合求解中小规模 case，在稀疏图上有很大优势，使用的是精确算法，结果是最优解。

结论 2: 如果不考虑时间因素或一定要求最优解的时候，优先推荐求解器，因为求解器得到的解是最优解。