

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

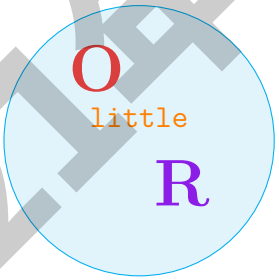
OlittleRer

运小筹月刊 2021 年 1 月 (总第 3 期)

本期作者：刘兴禄　曾文佳　王基光

本期主编：段淇耀　段宏达　何彦东　修宇璇

本期发布：曾文佳　夏旸



投稿请联系：hsinglul@163.com，欢迎广大同行的投稿！

目录

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

夏旸 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，邮箱：wangxin16@mails. tsinghua. edu. cn

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：zhangyihrmm2015@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：yanyu_chen_98@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

臧永森 清华大学，工业工程系，邮箱：zangys15@tsinghua.org.cn

左齐茹仪 清华大学，清华-伯克利深圳学院，邮箱：zqry20@mails.tsinghua.edu.cn

张婷 清华大学，清华-伯克利深圳学院，邮箱：15927584840@163.com

0.2 运小筹公众号简介

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。

欢迎读者朋友们加入我们的读者群，大家一起探讨学术问题。

Figure 0.2.1: 运小筹读者 4 群二维码

Chapter 1

2020-11-06：最速下降法：详细案例+Python 实现

刘兴禄，清华大学，深圳国际研究生院，清华伯克利深圳学院（博士在读）
CSDN 主页：https://blog.csdn.net/HsinglukLiu?spm=1011.2124.3001.5343

1.1 最速下降法 (The steepest descent method)

> 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。

在无约束非线性函数的最优化中，**最速下降法 (The steepest descent method)** 是一个著名的基础算法。本文就来用一个实例来学习最速下降法。

1.1.1 最速下降法的原理

假设有一个多元非线性函数

$$f(x_1, x_2, \cdots, x_n)$$

，其定义域为 $x \in \mathbb{R}^n$，那么如何求该函数的最大值或者最小值呢？

当然，我们可以用求导的方法。但是有些时候直接求导并不方便，此时我们可以用最速下降法来求解。

最速下降法的基本思想就是：

从一个初始点开始，逐步沿着以当前点为基准，函数值变化最快的方向走，一直走到最优解为止。因此，在有了一个初始点以后，我们就需要决策以下两个事情：

1. 下一步要朝着什么方向走 (方向);
2. 沿着该方向走多远 (步长)。

具体如何选择方向和步长，见下面的 PPT。

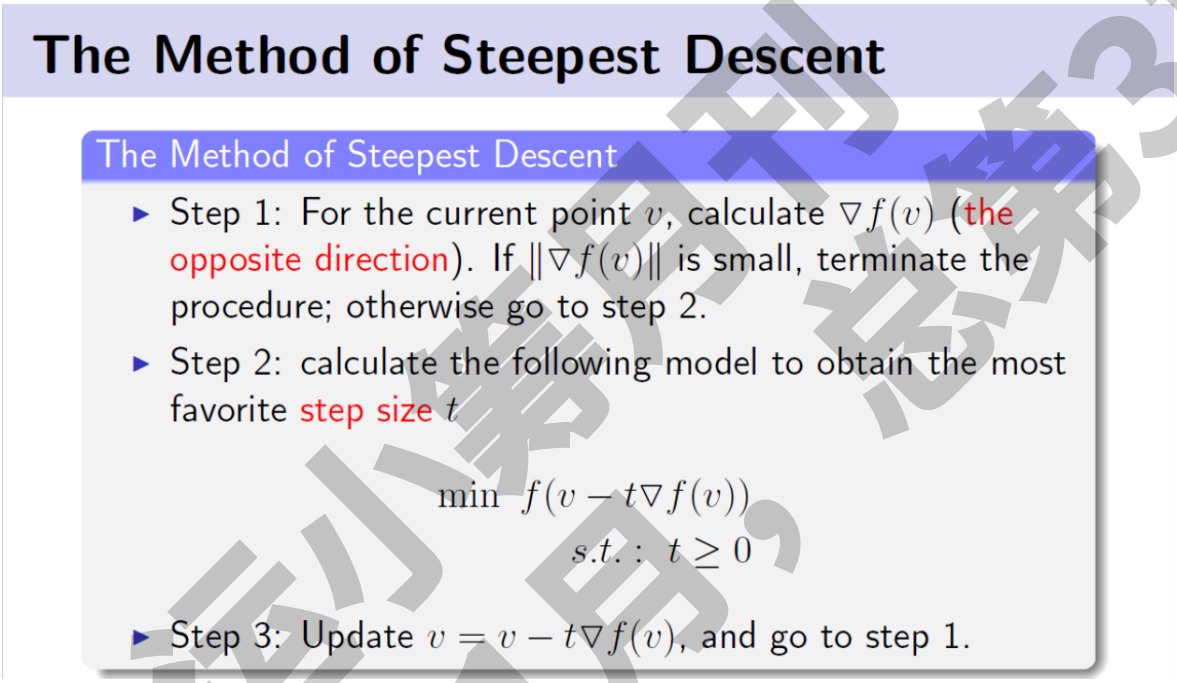


Figure 1.1.1: 最速下降法的原理 *(1)*

下图是一个很直观的例子，在当前点，沿着不同的方向走，函数值的变化速度是不同的，但是在等高线的切线的垂直方向，是变化最快的。

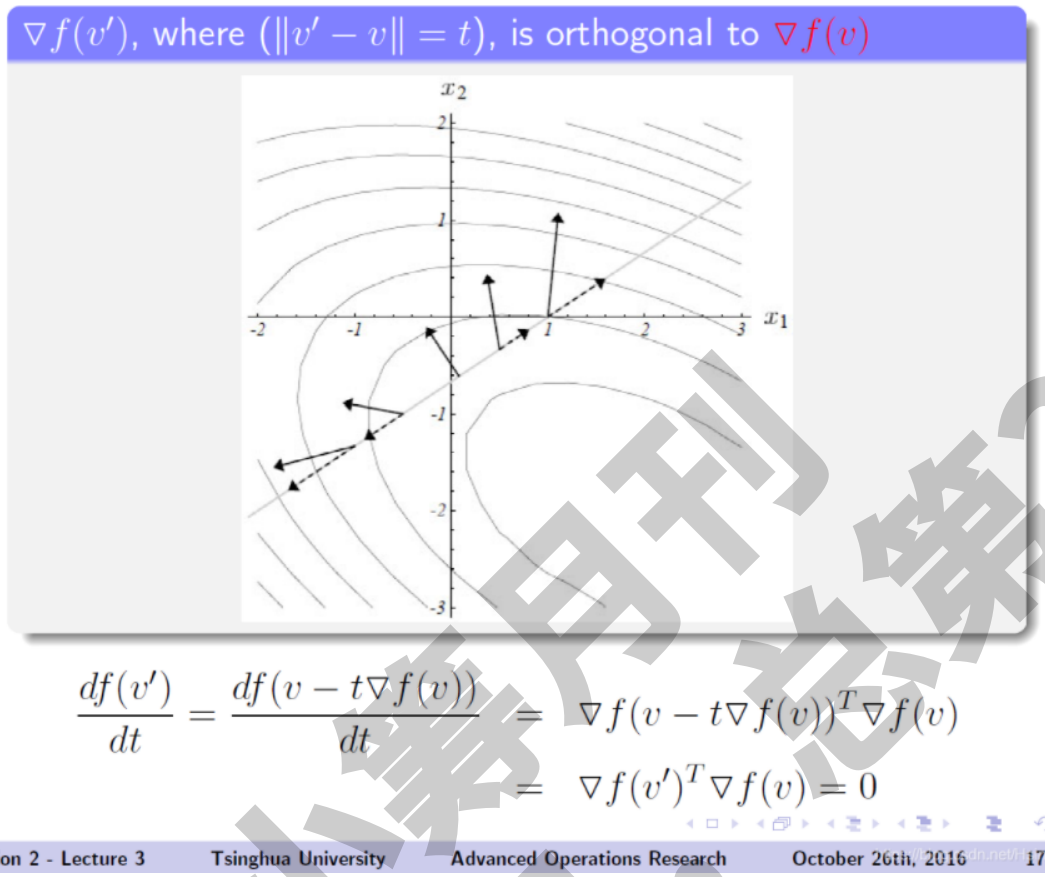


Figure 1.1.2: 最速下降法的原理 *(2)*

也就是说，我们得去找平行于该点梯度的方向，沿着这个方向 (当为 max 问题) 或者沿着这个方向的反方向 (当为 min 问题) 去更新当前位置。

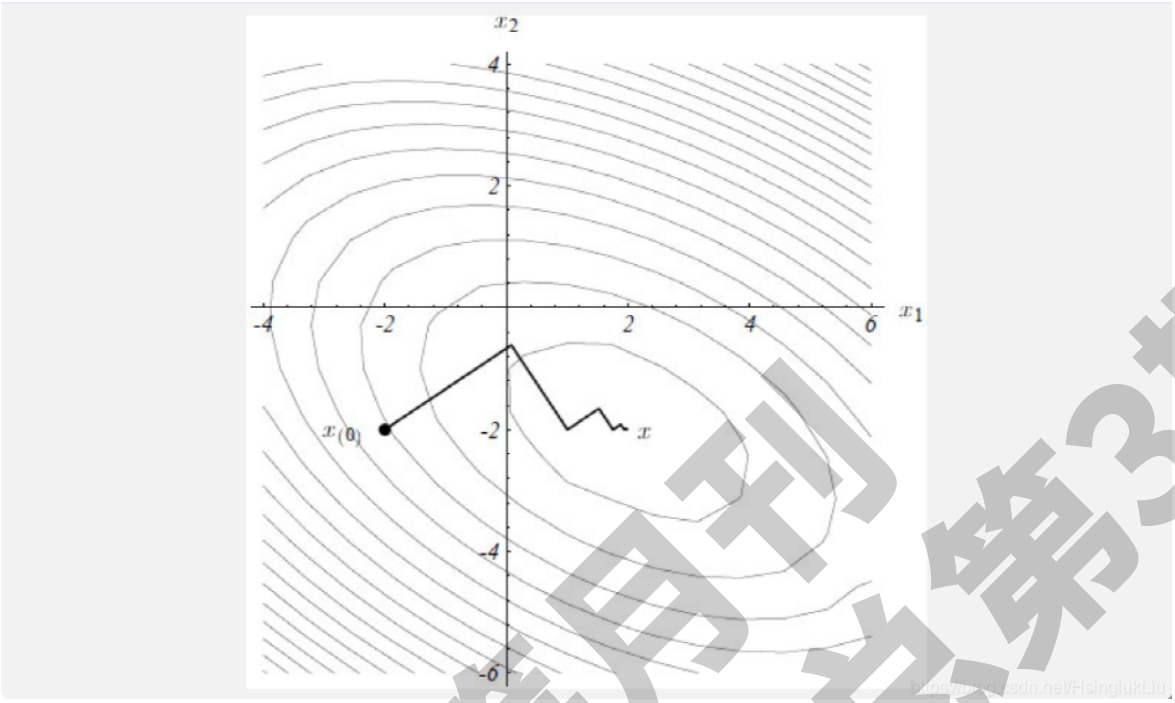


Figure 1.1.3: 最速下降法的原理 *(3)*

就像上图一样，一步一步，最终走到最优解对应的点。

1.1.2 Python 实现最速下降法实例

我们看下面的函数

$$f(x_1, x_2) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2, \tag{1.1.1}$$

$$x_1, x_2 \in \mathbb{R}^1 \tag{1.1.2}$$

给定初始点 $(0.5, 0.5)$，用最速下降法找到该函数的最大值。

我们用 python 中的sympy包来实现最速下降法求解上面的问题。

首先，我们来介绍我们将要用到的 sympy 包中的几个函数。结合 jupyter notebook 来给大家做个展示。ps. 这个包还是挺好用的，结合 jupyter notebook 之后，可视化非常不错，所见即所得。

1.1.3 sympy包中用到的函数

构建符号变量和符号函数数

Code

```
from sympy import *
x_1 = symbols('x_1')
x_2 = symbols('x_2')

fun = 2 * x_1 * x_2 + 2 * x_2 - x_1**2 - 2 * x_2**2
fun
```

这个是用来构造两个符号变量 x_1, x_2，就像代数中用字母代替变量一样。然后可以定义出我们的函数

$$f(x_1, x_2) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2, \tag{1.1.3}$$

jupyter notebook 中的显示效果是这样的

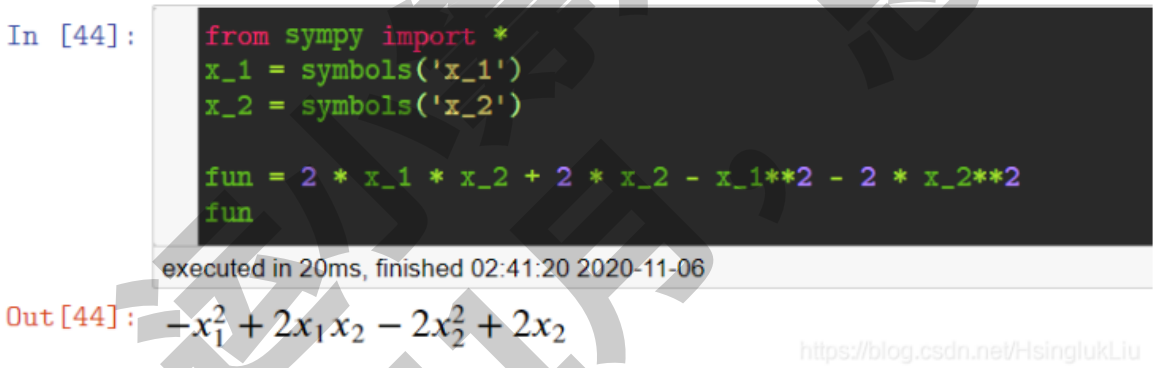


Figure 1.1.4: 函数 $f(x_1, x_2) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2$

可以看到 jupyter notebook 中直接就显示出了数学公式格式的形式，这是因为 jupyter notebook 中内嵌了 LaTeX 相关支持包的缘故。总之这样可视化就非常不错。

对符号函数求导

下面我们来计算 $f(x_1, x_2)$ 的偏导数，$\frac{\partial f}{\partial x_1}$，代码很简单，用函数diff(函数, 变量).

Code

```
grad_1 = diff(fun, x_1)
grad_1
```

如下图

In [45]:
```
grad_1 = diff(fun, x_1)
grad_1
```
executed in 15ms, finished 02:45:24 2020-11-06

Out[45]: $-2x_1 + 2x_2$

Figure 1.1.5: 对符号函数求导

求函数值

有了符号函数，我们怎么知道自变量 x_1, x_2 取具体值得时候，符号函数的取值呢？我们用函数 函数.subs(变量 1: 变量 1 的取值, ...).evalf()，具体代码为

Code
```
fun_value = fun.subs({x_1:4, x_2: 2}).evalf()
fun_value
```

In [52]:
```
fun_value = fun.subs({x_1:4, x_2: 2}).evalf()
fun_value
```
executed in 11ms, finished 02:51:28 2020-11-06

Out[52]: -4.0

Figure 1.1.6: 求函数值

还是非常智能的。

求解方程的零点

为了寻找下降速度最快的方向，我们需要利用之前 PPT 中的方法去求解方程组。这里我们用到函数 'solve(函数，变量)'

我们举一个例子，假设有函数

$$g(x) = 4x + 5 \tag{1.1.4}$$

我们求解

$$g(x) = 4x + 5 = 0 \tag{1.1.5}$$

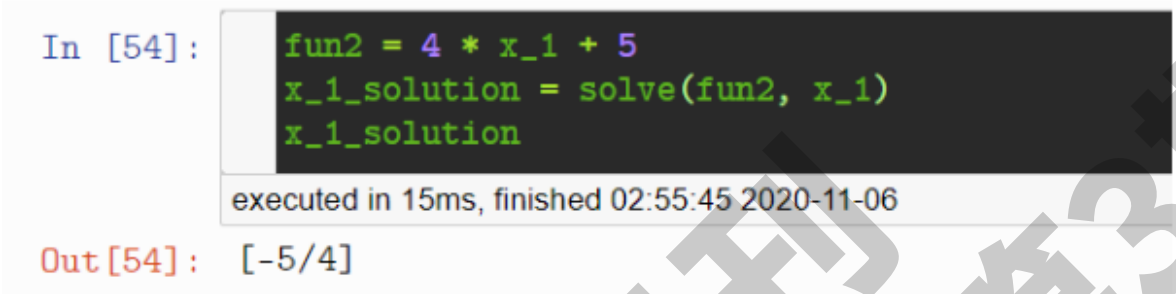


Figure 1.1.7: 求解方程的零点

1.1.4 Python 实现最速下降法求解上述算例的完整代码

Code

```
import math
from sympy import *

# define symbol variable
x_1 = symbols('x_1')
x_2 = symbols('x_2')

# define objective function
fun = 2 * x_1 * x_2 + 2 * x_2 - x_1 ** 2 - 2 * x_2 ** 2
fun

# take derivative of x_1 and x_2
grad_1 = diff(fun, x_1)
grad_2 = diff(fun, x_2)

# define parameters
MaxIter = 100
epsilon = 0.0001

# define initial point
x_1_value = 0.5
x_2_value = 0.5

iter_cnt = 0
current_step_size = 10000

grad_1_value = (float)(grad_1.subs({x_1:x_1_value, x_2: x_2_value}).evalf())
grad_2_value = (float)(grad_2.subs({x_1:x_1_value, x_2: x_2_value}).evalf())
```

```
29
30  current_obj = fun.subs({x_1: x_1_value, x_2: x_2_value}).evalf()
31
32  print('itCnt: %2d  cur_point (%3.2f, %3.2f)   cur_Obj: %5.4f     grad_1: %5.4f     grad_2 : %5.4f
     ↪   step_size : %5.4f' % (iter_cnt, x_1_value, x_2_value, current_obj, grad_1_value, grad_2_value,
     ↪   current_step_size))
33
34  # while (iter_cnt <= MaxIter and abs(grad_1_value) + abs(grad_2_value) >= epsilon):
35  while(abs(grad_1_value) + abs(grad_2_value) >= epsilon):
36      iter_cnt += 1
37      # find the step size
38      t = symbols('t')
39      x_1_updated = x_1_value + grad_1_value * t
40      x_2_updated = x_2_value + grad_2_value * t
41      Fun_updated = fun.subs({x_1: x_1_updated, x_2: x_2_updated})
42      grad_t = diff(Fun_updated, t)
43      t_value = solve(grad_t, t)[0]  # solve grad_t == 0
44
45      # update x_1_value and x_2_value
46      grad_1_value = (float)(grad_1.subs({x_1: x_1_value, x_2: x_2_value}).evalf())
47      grad_2_value = (float)(grad_2.subs({x_1: x_1_value, x_2: x_2_value}).evalf())
48
49      x_1_value = (float)(x_1_value + t_value * grad_1_value)
50      x_2_value = (float)(x_2_value + t_value * grad_2_value)
51
52      current_obj = fun.subs({x_1: x_1_value, x_2: x_2_value}).evalf()
53      current_step_size = t_value
54
55      print('itCnt: %2d  cur_point (%3.2f, %3.2f)   cur_Obj: %5.4f     grad_1: %5.4f     grad_2 : %5.4f
         ↪   step_size : %5.4f' % (iter_cnt, x_1_value, x_2_value, current_obj, grad_1_value,
         ↪   grad_2_value, current_step_size))
```

运行结果如下

Code

```
1   itCnt:  0  cur_point (0.50, 0.50)   cur_Obj: 0.7500     grad_1: 0.0000     grad_2 : 1.0000     step_size : 10000.0000
2   itCnt:  1  cur_point (0.50, 0.75)   cur_Obj: 0.8750     grad_1: 0.0000     grad_2 : 1.0000     step_size : 0.2500
3   itCnt:  2  cur_point (0.50, 0.75)   cur_Obj: 0.8750     grad_1: 0.5000     grad_2 : 0.0000     step_size : 0.0000
4   itCnt:  3  cur_point (0.75, 0.75)   cur_Obj: 0.9375     grad_1: 0.5000     grad_2 : 0.0000     step_size : 0.5000
5   itCnt:  4  cur_point (0.75, 0.75)   cur_Obj: 0.9375     grad_1: 0.0000     grad_2 : 0.5000     step_size : 0.0000
6   itCnt:  5  cur_point (0.75, 0.88)   cur_Obj: 0.9688     grad_1: 0.0000     grad_2 : 0.5000     step_size : 0.2500
7   itCnt:  6  cur_point (0.75, 0.88)   cur_Obj: 0.9688     grad_1: 0.2500     grad_2 : 0.0000     step_size : 0.0000
8   itCnt:  7  cur_point (0.88, 0.88)   cur_Obj: 0.9844     grad_1: 0.2500     grad_2 : 0.0000     step_size : 0.5000
9   itCnt:  8  cur_point (0.88, 0.88)   cur_Obj: 0.9844     grad_1: 0.0000     grad_2 : 0.2500     step_size : 0.0000
10  itCnt:  9  cur_point (0.88, 0.94)   cur_Obj: 0.9922     grad_1: 0.0000     grad_2 : 0.2500     step_size : 0.2500
11  itCnt: 10  cur_point (0.88, 0.94)   cur_Obj: 0.9922     grad_1: 0.1250     grad_2 : 0.0000     step_size : 0.0000
12  itCnt: 11  cur_point (0.94, 0.94)   cur_Obj: 0.9961     grad_1: 0.1250     grad_2 : 0.0000     step_size : 0.5000
13  itCnt: 12  cur_point (0.94, 0.94)   cur_Obj: 0.9961     grad_1: 0.0000     grad_2 : 0.1250     step_size : 0.0000
14  itCnt: 13  cur_point (0.94, 0.97)   cur_Obj: 0.9980     grad_1: 0.0000     grad_2 : 0.1250     step_size : 0.2500
15  itCnt: 14  cur_point (0.94, 0.97)   cur_Obj: 0.9980     grad_1: 0.0625     grad_2 : 0.0000     step_size : 0.0000
16  itCnt: 15  cur_point (0.97, 0.97)   cur_Obj: 0.9990     grad_1: 0.0625     grad_2 : 0.0000     step_size : 0.5000
17  itCnt: 16  cur_point (0.97, 0.97)   cur_Obj: 0.9990     grad_1: 0.0000     grad_2 : 0.0625     step_size : 0.0000
18  itCnt: 17  cur_point (0.97, 0.98)   cur_Obj: 0.9995     grad_1: 0.0000     grad_2 : 0.0625     step_size : 0.2500
19  itCnt: 18  cur_point (0.97, 0.98)   cur_Obj: 0.9995     grad_1: 0.0312     grad_2 : 0.0000     step_size : 0.0000
20  itCnt: 19  cur_point (0.98, 0.98)   cur_Obj: 0.9998     grad_1: 0.0312     grad_2 : 0.0000     step_size : 0.5000
```

```
itCnt: 20  cur_point (0.98, 0.98)   cur_Obj: 0.9998    grad_1: 0.0000    grad_2 : 0.0312    step_size : 0.0000
itCnt: 21  cur_point (0.98, 0.99)   cur_Obj: 0.9999    grad_1: 0.0000    grad_2 : 0.0312    step_size : 0.2500
itCnt: 22  cur_point (0.98, 0.99)   cur_Obj: 0.9999    grad_1: 0.0156    grad_2 : 0.0000    step_size : 0.0000
itCnt: 23  cur_point (0.99, 0.99)   cur_Obj: 0.9999    grad_1: 0.0156    grad_2 : 0.0000    step_size : 0.5000
itCnt: 24  cur_point (0.99, 0.99)   cur_Obj: 0.9999    grad_1: 0.0000    grad_2 : 0.0156    step_size : 0.0000
itCnt: 25  cur_point (0.99, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0156    step_size : 0.2500
itCnt: 26  cur_point (0.99, 1.00)   cur_Obj: 1.0000    grad_1: 0.0078    grad_2 : 0.0000    step_size : 0.0000
itCnt: 27  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0078    grad_2 : 0.0000    step_size : 0.5000
itCnt: 28  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0078    step_size : 0.0000
itCnt: 29  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0078    step_size : 0.2500
itCnt: 30  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0039    grad_2 : 0.0000    step_size : 0.0000
itCnt: 31  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0039    grad_2 : 0.0000    step_size : 0.5000
itCnt: 32  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0039    step_size : 0.0000
itCnt: 33  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0039    step_size : 0.2500
itCnt: 34  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0020    grad_2 : 0.0000    step_size : 0.0000
itCnt: 35  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0020    grad_2 : 0.0000    step_size : 0.5000
itCnt: 36  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0020    step_size : 0.0000
itCnt: 37  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0020    step_size : 0.2500
itCnt: 38  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0010    grad_2 : 0.0000    step_size : 0.0000
itCnt: 39  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0010    grad_2 : 0.0000    step_size : 0.5000
itCnt: 40  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0010    step_size : 0.0000
itCnt: 41  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0010    step_size : 0.2500
itCnt: 42  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0005    grad_2 : 0.0000    step_size : 0.0000
itCnt: 43  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0005    grad_2 : 0.0000    step_size : 0.5000
itCnt: 44  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0005    step_size : 0.0000
itCnt: 45  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0005    step_size : 0.2500
itCnt: 46  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0002    grad_2 : 0.0000    step_size : 0.0000
itCnt: 47  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0002    grad_2 : 0.0000    step_size : 0.5000
itCnt: 48  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0002    step_size : 0.0000
itCnt: 49  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0002    step_size : 0.2500
itCnt: 50  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0001    grad_2 : 0.0000    step_size : 0.0000
itCnt: 51  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0001    grad_2 : 0.0000    step_size : 0.5000
itCnt: 52  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0001    step_size : 0.0000
itCnt: 53  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0000    grad_2 : 0.0001    step_size : 0.2500
itCnt: 54  cur_point (1.00, 1.00)   cur_Obj: 1.0000    grad_1: 0.0001    grad_2 : 0.0000    step_size : 0.0000
```

这样就完成了。

当然了，在这个例子中，从第 3 步左右的迭代开始，后续的点就非常近了，因此，步长就需要动态的调整。具体文献之后补充。

Chapter 2

2020-11-09: Python 调用 Gurobi: Shortest Path Problem(最短路问题)及其对偶问题的一些探讨

刘兴禄，清华大学，深圳国际研究生院，清华伯克利深圳学院（博士在读）
CSDN 主页：https://blog.csdn.net/HsinglukLiu?spm=1011.2124.3001.5343

2.1 Python 调用 Gurobi：Shortest Path Problem 及其对偶问题的一些探讨

最短路问题 (Shortest Path Problem，SPP) 是一类非常经典的问题。基本的 SPP 不是 NP-hard, 可以用 Dijkstra 等算法在多项式时间内求解到最优解。今天我们不探讨这些求解算法，仅探讨用求解器 Gurobi 和 Python 来求解这个问题。

我们首先来看一个例子网络：

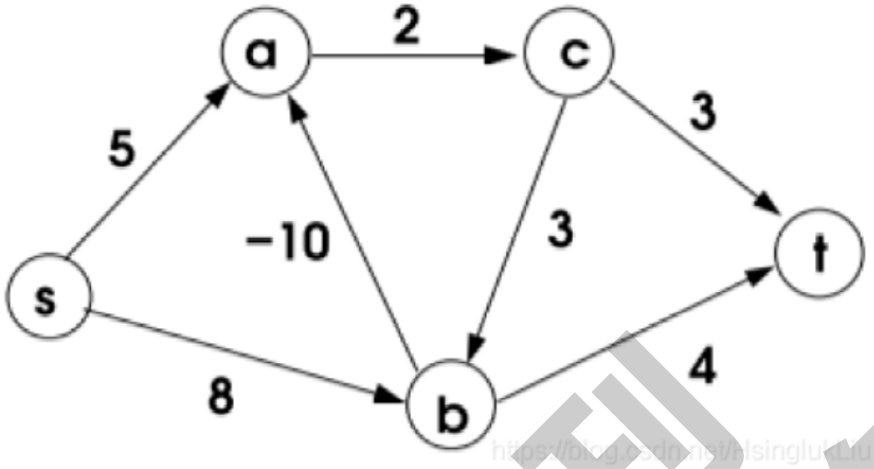


Figure 2.1.1: *SPP* 例子网络

- s 为起点，t 为终点。
- 注意到边 $b \to a$ 的权重是 -10，那么在这个问题中就存在负环，例如 $b \to a \to c \to b$，这个环的总权重为 -5。这种情况下，对偶问题是无解的。用这个例子目的就是稍微拓展一下这一点。

2.1.1 Shortest Path Problem : The Model

首先，我们将 Shortest Path Problem 建模成一个 Integer Programming 模型，如下：

$$\min \sum_{(i,j)\in A} x_{ij} d_{ij} \tag{2.1.1}$$

$$\sum_{(s,j)\in A} x_{sj} = 1 \tag{2.1.2}$$

$$\sum_{(j,t)\in A} x_{jt} = 1 \tag{2.1.3}$$

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = 0, \quad \forall j \in V \backslash \{s,t\} \tag{2.1.4}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{2.1.5}$$

2.1.2 Python 调用 Gurobi 求解 SPP

下面我们用 Python 调用 Gurobi，并用上图中的算例进行建模求解。
首先，导入模块，并将上述网络转换成字典格式数据

Code

```python
from gurobipy import *
import pandas as pd
import numpy as np

Nodes = ['s', 'a', 'b', 'c', 't']

Arcs = {('s','a'): 5
        ,('s','b'): 8
        ,('a','c'): 2
        ,('b','a'): -10
        ,('c','b'): 3
        ,('b','t'): 4
        ,('c','t'): 3
       }
Arcs
```

然后建模

Code

```python
model = Model('dual problem')

# add decision variables
X = {}
for key in Arcs.keys():
    index = 'x_' + key[0] + ',' + key[1]
    X[key] = model.addVar(vtype=GRB.BINARY
                          , name= index
                         )

# add objective function
obj = LinExpr(0)
for key in Arcs.keys():
    obj.addTerms(Arcs[key], X[key])

model.setObjective(obj, GRB.MINIMIZE)

# constraint1 1 and constraint 2
lhs_1 = LinExpr(0)
lhs_2 = LinExpr(0)
for key in Arcs.keys():
    if(key[0] == 's'):
        lhs_1.addTerms(1, X[key])
    elif(key[1] == 't'):
        lhs_2.addTerms(1, X[key])
model.addConstr(lhs_1 == 1, name = 'start flow')
model.addConstr(lhs_2 == 1, name = 'end flow')

# constraints 3
```

```python
for node in Nodes:
    lhs = LinExpr(0)
    if(node != 's' and node != 't'):
        for key in Arcs.keys():
            if(key[1] == node):
                lhs.addTerms(1, X[key])
            elif(key[0] == node):
                lhs.addTerms(-1, X[key])
    model.addConstr(lhs == 0, name = 'flow conservation')

model.write('model_spp.lp')
model.optimize()

print(model.ObjVal)
for var in model.getVars():
    if(var.x > 0):
        print(var.varName, '\t', var.x)
```

求解结果如下

Code

```
Solution count 1: 3

Optimal solution found (tolerance 1.00e-04)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap 0.0000%
3.0
x_s,b       1.0
x_a,c       1.0
x_b,a       1.0
x_c,t       1.0
```

可以看到，最短路为 $s \to b \to a \to c \to t$，路径长度为 3.

2.1.3 SPP 的对偶问题

下面我们写出这个问题的对偶问题，我们对每个约束引入对偶变量 π_i，如下

$$\min \sum_{(i,j)\in A} x_{ij} d_{ij} \tag{2.1.6}$$

$$\sum_{(s,j)\in A} x_{sj} = 1 \quad \to \quad \pi_s \tag{2.1.7}$$

$$\sum_{(j,t)\in A} x_{jt} = 1 \quad \to \quad \pi_t \tag{2.1.8}$$

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,k)\in A} x_{jk} = 0, \quad \forall j \in V \backslash \{s,t\} \quad \to \quad \pi_j \tag{2.1.9}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A \tag{2.1.10}$$

然后 SPP 的对偶问题变成

$$\begin{aligned}
\max \quad & \pi_s + \pi_t && && (2.1.11)\\
& \pi_s - \pi_j \leqslant d_{sj}, && \forall j \in \{a,b\} && (2.1.12)\\
& \pi_j + \pi_t \leqslant d_{jt}, && \forall j \in \{b,c\} && (2.1.13)\\
& \pi_i - \pi_j \leqslant d_{ij}, && \forall i,j \in A, i,j \in V \setminus \{s,t\} && (2.1.14)\\
& \pi \text{ free} && && (2.1.15)
\end{aligned}$$

2.1.4 Python 调用 Gurobi 求解 SPP 的对偶问题

下面调用 gurobi 进行求解

Code

```
model = Model('dual problem')

pi_a = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_a")
pi_b = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_b")
pi_c = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_c")
pi_s = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_s")
pi_t = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_t")

obj = LinExpr(0)
obj.addTerms(1 , pi_s)
obj.addTerms(1 , pi_t)
model.setObjective(obj, GRB.MAXIMIZE)

# lhs relation , rhs
model.addConstr(pi_s - pi_a <= 5)
model.addConstr(pi_s - pi_b <= 8)
model.addConstr(pi_t + pi_c <= 3)
model.addConstr(pi_t + pi_b <= 4)
model.addConstr(pi_b - pi_a <= -10)
model.addConstr(pi_a - pi_c <= 2)
model.addConstr(pi_c - pi_b <= 3)

model.write('model2.lp')
model.optimize()

print(model.ObjVal)
for var in model.getVars():
    print(var.varName, '\t', var.x)
```

求解之后，结果为无解

Code

```
Gurobi Optimizer version 9.0.1 build v9.0.1rc0 (win64)
Optimize a model with 7 rows, 5 columns and 14 nonzeros
Model fingerprint: 0xd51f0a26
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+03, 1e+03]
  RHS range        [2e+00, 1e+01]
Presolve time: 0.00s
Presolved: 7 rows, 5 columns, 14 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.0100000e+03   5.026250e+02   0.000000e+00      0s

Solved in 2 iterations and 0.01 seconds
Infeasible model
```

让我们修改边 $b \to a$ 的长度为 10，也就是把下面的约束改成

Code

```
# model.addConstr(pi_b - pi_a <= -10)
model.addConstr(pi_b - pi_a <= 10)
```

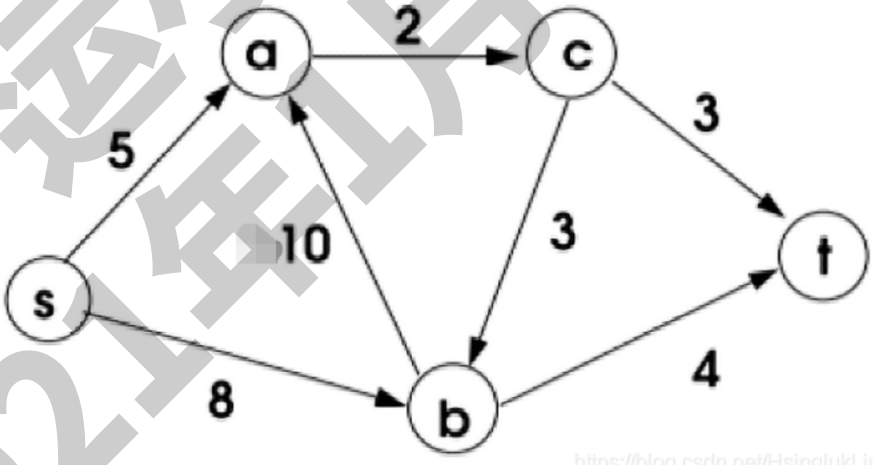


Figure 2.1.2: 修改后的 *SPP* 例子网络

这样就有解了

Code

```
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.0100000e+03   5.020000e+02   0.000000e+00      0s
```

```
       5    1.0000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 5 iterations and 0.01 seconds
Optimal objective  1.000000000e+01
10.0
pi_a      995.0
pi_b      992.0
pi_c      993.0
pi_s      1000.0
pi_t      -990.0
```

可以看到，最短路为 $s \to a \to c \to t$，路径长度为 10.

2.2 SPP 模型学术界的标准写法及其求解

2.2.1 SPP 模型学术界的标准写法

上面的写法是比较适合初学者的，学者们一般不这么写，下面我们来整理一下比较专业的写法。主要参考文献 (Garg and Könemann 2007) 和 (Cappanera and Scaparra 2011)。我们将 SPP 的模型可以写为：

$$\max \sum_{e \in A} d_e x_e \tag{2.2.1}$$

$$\sum_{e \in \text{out}(i)} x_e - \sum_{e \in \text{in}(i)} x_e = b_i, \qquad \forall i \in V \tag{2.2.2}$$

$$x_e \in \{0,1\}, \qquad \forall e \in A \tag{2.2.3}$$

其中当 $i = s$ 时，$b_i = 1$, 当 $i = t$ 时，$b_i = -1$, 否则 $b_i = 0$。

当然，我这里写 out, in 什么的纯粹为了看着直观，其实也不是很直观，论文里有的是 FS，BS。

也有写成下面形式的

$$\max \quad \sum_{e \in A} d_e x_e \tag{2.2.4}$$

$$\sum_{e \in \text{out}(i)} x_e - \sum_{e \in \text{in}(i)} x_e = \begin{cases} 1, & \text{if } i = s \\ -1, & \text{if } i = t \\ 0, & \text{else} \end{cases} \tag{2.2.5}$$

$$x_e \in \{0,1\}, \qquad \forall e \in A \tag{2.2.6}$$

看个人喜好了。

由于 SPP 具有整数最优解特性，也就是将决策变量 x_e 松弛成 $0 \leqslant x_e \leqslant 1$, 依然总是存在整数最优解。因此，SPP 等价于下面的 LP。

$$\min \sum_{e \in A} d_e x_e \tag{2.2.7}$$

$$\sum_{e \in \text{out}(i)} x_e - \sum_{e \in \text{in}(i)} x_e = b_i, \qquad \forall i \in V \tag{2.2.8}$$

$$0 \leqslant x_e \leqslant 1, \qquad \forall e \in A \tag{2.2.9}$$

相应的，对偶问题则变成

$$\max \qquad \pi_s - \pi_t \tag{2.2.10}$$

$$\pi_i - \pi_j \leqslant d_{ij}, \qquad \forall (i,j) \in A \tag{2.2.11}$$

$$\pi_i \ \text{ free} \tag{2.2.12}$$

到这一步其实已经结束了。我们可以做一个比较直观的理解 (这个理解理论上有缺陷):

> π_i 其实就是从 s 点出发，到达点 j 的收益的一个衡量。当然，数值上并不是这样的，之后我们做一步操作，就可以让数值上也相等了。

首先把所有对偶变量取个相反数，等价转化为

$$\max \qquad \pi_t - \pi_s \tag{2.2.13}$$

$$\pi_j - \pi_i \leqslant d_{ij}, \qquad \forall (i,j) \in A \tag{2.2.14}$$

$$\pi_i \ \text{ free} \tag{2.2.15}$$

因为 π_i 是 free 的，因此是等价的。

由于 π_i 是无约束的，而我们只是最大化 $\pi_s - \pi_t$，因此我们可以直接把 π_s 设置成 0，也就是约束 $\pi_s = 0$，这样并不会改变最优解。OK，我们仍然保持目标函数为 max，那我们就可以把模型等价转化为

$$\max \quad \pi_t \tag{2.2.16}$$

$$\pi_j - \pi_i \leqslant d_{ij}, \qquad \forall (i,j) \in A \tag{2.2.17}$$

$$\pi_i \ \text{ free}, i \neq s, \pi_s = 0 \tag{2.2.18}$$

这样我们就能保证所有的对偶变量 π_i 都是非负的，而且这个模型与原模型完全等价，而且问题复杂度相同。

2.2.2 SPP 对偶问题的直观理解

依据这个模型，我们就可以对对问题进行一个比较深入的直观理解了。

> 就像生产计划问题的对偶问题，可以按照出租的逻辑来理解一样，SPP 问题的原问题是一个客户要在网络上从起点到终点走一条最短路，花销最少。那 SPP 的对偶问题就可以理解为，我是网络的运营者，我需要在每个结点设置收费金额（或者惩罚金额），我设置的惩罚金额必须要使得我的收益在对方走最短路的情况下最大化。当然，对方不走最短路，我的收益会更大，这个惩罚，在一条边连接的两个节点上，差值不能超过这条边的权重。
>
> 这里，我们戏称运营者为地主。地主为了让平民走得最短，就得拼命的加成法，狠狠收费。怎么衡量地主收费收的是不是狠呢？下面给出一些直观理解的分享 (** 并不完全严谨，仅供参考，帮助理解 **)
>
> 一条边 (i,j) 的收费上限是 d_{ij}。地主收费不狠，那就是 $\pi_j - \pi_i << d_{ij}$，说明他还有点良心。余量越多，他的收益就越小，对应的，原问题相应的解就越远离最短路。收的越狠，接近 $\pi_j - \pi_i \approx d_{ij}$ 了，原问题（平民）就不敢瞎浪了，乖乖走到最短路，毕竟财力限制了他的想象力。当 $\pi_j - \pi_i = d_{ij}$，这就使得原问题走到最短路。

改进的对偶问题的解，也就是原问题的对偶变量，正好也符合上面的理解。

2.3 SPP 模型学术界的标准形式的求解

按照修改的模型，我们再将今天的例子求解一下，约束按照修改的模型进行构建

Code

```
from gurobipy import *
import pandas as pd
import numpy as np

```

```
5
6  model = Model('dual problem')
7
8  pi_a = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_a")
9  pi_b = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_b")
10 pi_c = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_c")
11 pi_s = model.addVar(lb=0, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_s")
12 pi_t = model.addVar(lb=-1000, ub=1000, vtype=GRB.CONTINUOUS, name= "pi_t")
13
14 obj = LinExpr(0)
15 # obj.addTerms(1 , pi_s)
16 obj.addTerms(1 , pi_t)
17 model.setObjective(obj, GRB.MAXIMIZE)
18
19 '''
20 Arcs =
21 {('s', 'a'): 5,
22  ('s', 'b'): 8,
23  ('a', 'c'): 2,
24  ('b', 'a'): -10,
25  ('c', 'b'): 3,
26  ('b', 't'): 4,
27  ('c', 't'): 3}
28 '''
29 # lhs relation , rhs
30 # 注意这里约束都进行了调整
31 model.addConstr(pi_a - pi_s <= 5)
32 model.addConstr(pi_b - pi_s <= 8)
33 model.addConstr(pi_t - pi_c <= 3)
34 model.addConstr(pi_t - pi_b <= 4)
35 model.addConstr(pi_a - pi_b <= 10)
36 model.addConstr(pi_c - pi_a <= 2)
37 model.addConstr(pi_b - pi_c <= 3)
38 model.addConstr(pi_s == 0)
39
40 model.write('model2.lp')
41 model.optimize()
42
43 print(model.ObjVal)
44 for var in model.getVars():
45     print(var.varName, '\t', var.x)
```

求解结果如下

Code

```
1 Solved in 2 iterations and 0.01 seconds
2 Optimal objective  1.000000000e+01
3 10.0
4 pi_a      5.0
5 pi_b      6.0
```

```
pi_c      7.0
pi_s      0.0
pi_t      10.0
```

看到了把，跟前面的理解是相同的。$\pi_i, \forall i \in V$ 可以看做是在点 i 处地主能够收到的 '过路费 '的最大值。在终点能够收到的过路费的最大值 π_t 就是最终的目标函数。而过路费必须满足决策者提出的 $\pi_j - \pi_i \leqslant d_{ij}$ 的反垄断要求。

然后平民就乖乖走出了最短路。这问题就这么愉快的解决了。

不知道小伙伴们有没有一个疑问，上面我写这个 SPP 的对偶的时候，比较容易的就写出来了，也就是像 SPP 这样的问题，写对偶还是比较简单的，但是对于一些比较复杂的问题，约束里面包含了挺多复杂的东西，怎么写出对偶呢？之后的博文会陆续推出如何写出大规模 LP 的对偶问题。

先去搞研究了，今天就写到这里啦，下期再见。

Chapter 3

2021-01-03：机器学习运用到 VRP 的若干小知识

曾文佳，清华大学，工业工程系/深圳国际研究生院（硕士在读）
csdn 主页：https://blog.csdn.net/weixin_46991173

车辆路径规划问题（Vehicle Routing Problem，VRP）是运筹学领域十分经典的组合优化问题。近几十年来已经有众多学者通过启发式以及一些精确算法对该问题进行了全面且深入的研究。

3.1 相关文献

通过机器学习解决旅行商问题（TSP）和车辆路径规划问题（VRP）一般有两种方式：

3.1.1 端到端方法（Pointer Network +Attention Model）

- Vinyals 等人（2015）首先引入了 Pointer Network，以 Seq2Seq 模型为灵感，以解决 TSP。他们使用注意力模型以监督方式学习不同节点的顺序。
- Bello 等人（2016 年）开发了一种 RL 算法来训练 Pointer Network。他们的框架从问题实例中学习最佳策略，不需要监督解决方案。
- Nazari 等人（2018）用新设计改进了 Pointer Network，使输入序列的模型不变性得以扩展，并对其进行扩展以解决 VRP。
- Kool 等人（2019）提出了一个基于注意力层的模型，以及一种 RL 算法来训练具有简单但有效基线的模型。

3.1.2 强化学习方法（Policy Gradient、Actor Critic）

除了上述端到端的方法，强化学习也在近年被运用到 VRP 问题中。这种方法一般基于元启发式算法的基础之上，通过 RL 可以准确有效的进行算子的选择。

- Chen & Tian（2019）提出了针对 VRP 的 NeuRewriter Model。他们定义了重写规则集，并训练了两个策略网络（区域选择策略和规则选择策略）以获得下一个状态。给定一个初始解决方案，他们的目标是找到以最小的成本实现解决方案的一系列步骤。
- Hao Lu 等人提出了一个“Learn to Improve”(L2I) 方法，更加高效，并且与 OR 方法进行了比较更优。其核心思想就是在元启发式迭代搜索的过程中，加入了 RL 来帮助更有效的选择算子。

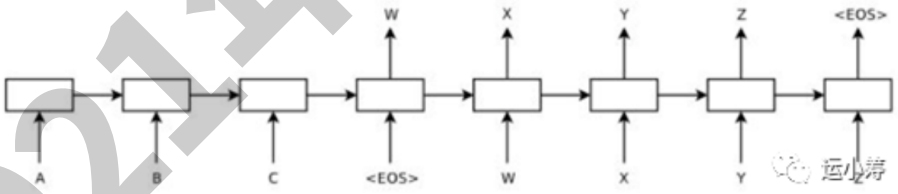
3.2 背景小知识

那么有哪些机器学习中的小知识被广泛的运用到 VRP 问题的解决中来呢，本文将简单的介绍 3 个最基础的背景小知识：注意力机制、指针网络和策略网络。

3.2.1 注意力机制（Attention Mechanism）

首先，我们要理解 VRP 的输入和输出，其实就是输入一些离散的点，然后把它们连接为一条条路，这样就完成了车辆路径的规划。所以我们可以将其作为一个Seq2Seq的过程，客户点的序列进行处理后作为输入，而生成的最终路径就是输出。Seq2Seq是通过 encoder 和 decoder 的过程进行翻译，这一过程在机器翻译的场景中十分常见，这里就产生了一个问题。

在机器翻译中一直存在对齐的问题。也就是说源语言的某个单词应该和目标语言的哪个单词对应的问题，比如说“Who are you”对应“你是谁”，由于英文和中文常用的语序不同，如果我们简单地按照顺序将词汇进行翻译，显然会得到一个错误的翻译结果“谁是你”。下图展示了 Seq2Seq 的过程（图片来源于网络）。



那么如何让词之间实现对齐，让模型准确的知道“你”是和“you”对应的呢？这时候，我们就得用到注意力机制。‘注意力机制（Attention Mechanism）‘会给输入序列的每一个元素分配一个权重，这样在预测“你”这个字的时候输入序列中的“you”这个词的权重最大，从而实现了 软对齐。

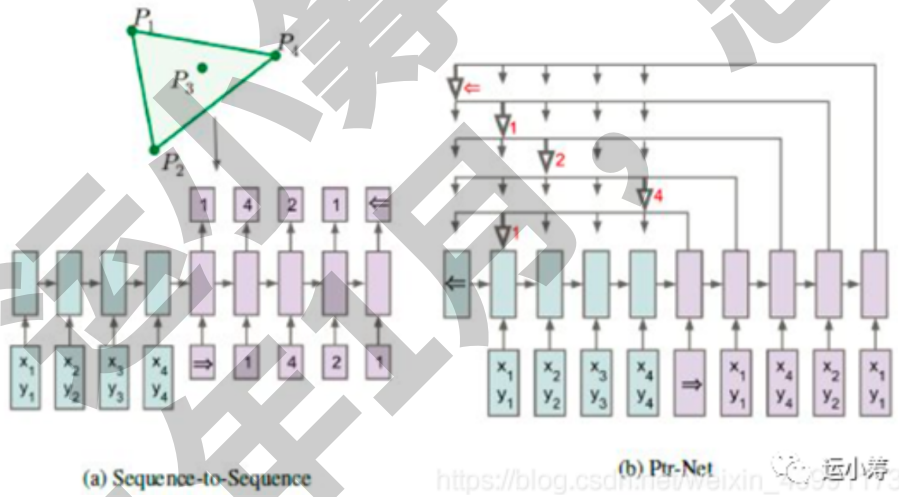
传统的注意力公式如下。第一条公式中 e_j 和 d_i 分别是 encoder 和 decoder 的隐状态，其余是可学习的参数；得到的 u_j^i 通过 softmax 操作得到分配给输入序列的权重 a_j^i。之后再根据该权重求加权和，得到 d_i' 后加到 decoder 的隐状态上，即可进行解码和预测。

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, ..., n) a_j^i = \text{softmax}(u_j^i) \quad j \in (1, ..., n) d_i' = \sum_{j=1}^{n} a_j^i e_j$$

3.2.2 指针网络（Pointer Network）

解决了软对齐的问题，接下来还要处理另一个问题：传统注意力机制的输出是输出词汇表中的词汇，接着上面的例子来说，输出是英文词汇对应的中文词汇表“你”、“是”、“谁”。但是在 VRP 中，‘我们希望输出的路径序列中的组成元素恰恰就是输入中的客户点‘。

对于这种输出往往是输入的子集的问题。考虑能不能找到一种指针，每个指针对应输入序列的一个元素，我们可以直接通过指针来讲输入序列进行变换和输出，而不是输出词汇表。这就是 ‘指针网络（Pointer Networks）‘。下图展示了传统 Seq2Seq 和指针网络输入输出的区别（图片来源于网络）。



下面是指针网络的公式，相比传统注意力机制更加简单，公式一与传统注意力机制相同，公式二直接将 softmax 得到的权重值作为了输出，充当指针。

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, ..., n) p\ C_i | C_1, ..., C_{i-1}, \mathcal{P}) = \text{softmax}(u^i)$$

3.2.3 策略网络（Policy Network）

接下来要将的策略网络就是强化学习比较核心的内容了。其实对于 VRP，也有很多学者用 DQN 等方法进行解决，大家可以更多的查阅文献进行了解。此处，我仅着重介绍策略网络。

Figure 3.2.1: 策略网络

策略网络其实十分简单粗暴，仅仅是循环这 3 个步骤：

1. 作出行为
2. 得到结果
3. 根据结果来增加/减少下次该行为被选择的概率。

举个小例子：

小老鼠在一个房间中爬行，它可以进行上下左右移动的行为。如果他遇到猫或者捕鼠夹，他会受伤或死亡，这时环境返回的 reward 将会很小甚至是负的。但如果他遇到奶酪，就会得到一个较大的正 reward。策略网络将根据这些 reward 来增大或减少行为发生的概率，从而使小老鼠学会正确的找奶酪路径。(图片来源于网络）

Figure 3.2.2: 小例子

下图给出了经典的蒙特卡洛梯度策略强化算法 Monte-Carlo Policy Gradient(图片来源于网络)，其中：

- π：自变量为 的策略函数，s 状态下 a 动作发生的概率
- v_t：奖励函数，根据动作好坏决定改进方向取对数：以得到更好的收敛性
- ∇：梯度，即参数改进的方向
- α：学习率

function REINFORCE
Initialise θ arbitrarily
for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
for $t = 1$ to $T - 1$ **do**
$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
end for
end for
return θ
end function

Figure 3.2.3: *Monte-Carlo Policy Gradient*

Chapter 4

2021-01-06：初识随机规划：一个小小例子

刘兴禄，清华大学，深圳国际研究生院，清华伯克利深圳学院（博士在读）
CSDN 主页：https://blog.csdn.net/HsinglukLiu?spm=1011.2124.3001.5343

4.1 初识随机规划：一个小小例子

本文中的确定性问题的例子来源于《运筹学》第四版，运筹学编写组编著。

‘随机规划‘和‘鲁棒优化‘都是运筹优化中比较高阶的内容。二者都是考虑不确定情形下的数学规划，但是两者又有不同。‘随机规划‘旨在优化不确定情形下的目标函数的‘期望‘等，比如‘总收益的期望值‘等。但是‘鲁棒优化‘致力于使得最坏的情况最好。所以相比而言，基本的‘鲁棒优化‘获得的结果会比较保守。

本文主要介绍‘随机规划‘。‘随机规划‘是一种非常常用的用来处理不确定参数下的优化方法。本节不涉及非常深入的部分，仅以一个非常小的例子，来跟大家介绍什么是‘随机规划‘，以及‘随机规划‘能干什么。

4.1.1 生产计划的例子

一个工厂生产 2 种产品 A 和 B，A 产品需要 1 个单位人工工时和 4 个单位的设备 1 的工时，B 产品需要 2 个单位的人工工时和 4 个单位的设备 2 的工时。且该厂一共可提供人工工时、设备 1 工时和设备 2 的工时分别为 8,16 和 12。且生产单位产品 A、B 的净利润分别为 2 和 3。假设该工厂生产的产品是供不应求的，问，该工厂应该如何生产 (可以是连续的量)，使得净利润最大化。

根据上述描述，我们引入下面的决策变量：x_1, x_2 分别代表生产产品 A B 的量，则该问题的数学模型为

$$\max \quad 2x_1 + 3x_2 \tag{4.1.1}$$

$$s.t. \quad x_1 + 2x_2 \leqslant 8 \tag{4.1.2}$$

$$4x_1 \leqslant 16 \tag{4.1.3}$$

$$4x_2 \leqslant 12 \tag{4.1.4}$$

$$x_1, x_2 \geqslant 0 \tag{4.1.5}$$

我们用 Pytho 调用 ‘Gurobi‘对其进行求解，具体代码如下：

Code

```
from gurobipy import *
model = Model('production plan')
x = {}
for i in range(1,3):
    x[i] = model.addVar(lb = 0, ub = GRB.INFINITY, vtype = GRB.CONTINUOUS, name = 'x_' + str(i))

model.setObjective(2 * x[1] + 3 * x[2], GRB.MAXIMIZE)
model.addConstr(x[1] + 2 * x[2] <= 8, name = 'cons_1')
model.addConstr(4 * x[1] <= 16, name = 'cons_2')
model.addConstr(4 * x[2] <= 12, name = 'cons_2')

# solve
model.optimize()

# print the results
print('Obj = ', model.ObjVal)
for key in x.keys():
    print(x[key].varName, ' = ', x[key].x)
```

求解结果如下

Code

```
Solved in 1 iterations and 0.02 seconds
Optimal objective  1.400000000e+01
Obj: 14.0
x_1  =  4.0
x_2  =  2.0
```

即生产 4 单位 A 产品和 2 单位 B 产品，可以获得 14 个单位的利润。

这是一个非常简单的例子，下面我们基于这个例子，来引入什么是随机规划 (‘Stochastic Programming‘)。

4.1.2 参数的不确定性

我们将上述模型抽象称为一个参数形式的模型，其中 c,a,b 均为参数。

$$\max\ c_1x_1 + c_2x_2 \tag{4.1.6}$$
$$s.t.\quad a_{11}x_1 + a_{12}x_2 \leqslant b_1 \tag{4.1.7}$$
$$a_{21}x_1 \qquad\quad \leqslant b_2 \tag{4.1.8}$$
$$a_{32}x_2 \leqslant b_3 \tag{4.1.9}$$
$$x_1, x_2 \geqslant 0 \tag{4.1.10}$$

上述模型中，c_i 即为产品 i 的净利润，a_{ij} 即为资源消耗参数，b_i 即为可用资源的数量。在确定性的问题中，我们假设所有参数，即 c, a, b 都是可以提前给定的。比如商品的售价、可用的资源以及生产的时间等。

但是在实际的场景中，这些参数也许是不确定的，比如生产所需要的工时，可能会由于机器、人工的误差，导致一些波动 (即 a 出现波动)。一些产品的价格，也许会随着季节，供需关系的变动而变动 (即 c 出现波动)。生产资料也有可能出现同样的不确定性 (即 b 出现波动)。

如果我们按照确定性情形下的参数去做计划 (即 $c_1 = 2, c_2 = 3, a_{11} = 1, a_{12} = 2, \cdots$)，在实际生产活动中，由于随机的因素，可能导致真正的参数跟确定情形下考虑的参数不一致。比如员工们的状态不好，导致人工生产时间变长了一些 (比如 a_{11} 变成了 1.05 等)，或者设备启动等除了点小意外，使得需要的机器工时多了一些等。在这种情况下，我们原来考虑确定性参数的生产计划，在随机的情形下也许就会不可行，或者说受到较大的影响。

一种可行的解决方案就是：随机规划。

体现在上面的例子中，就是我们考虑模型的参数是不确定的。我们用下面的数学语言来描述

$$\max\ \tilde{c}_1x_1 + \tilde{c}_2x_2 \tag{4.1.11}$$
$$s.t.\quad \tilde{a}_{11}x_1 + \tilde{a}_{12}x_2 \leqslant \tilde{b}_1 \tag{4.1.12}$$
$$\tilde{a}_{21}x_1 \qquad\quad \leqslant \tilde{b}_2 \tag{4.1.13}$$
$$\tilde{a}_{32}x_2 \leqslant \tilde{b}_3 \tag{4.1.14}$$
$$x_1, x_2 \geqslant 0 \tag{4.1.15}$$

其中 $\tilde{c}, \tilde{a}, \tilde{b}$ 表示这些参数是不确定的。但是这些参数又不是完全没有已知信息。我们假

设他们的一些概率分布信息是已知的。比如均值、方差等。

例如，我们假设已知

$$\mathbb{E}(\tilde{a}_{ij}) = \mu_{ij}, \quad \sigma_{ij} = 1 \tag{4.1.16}$$

$$\mathbb{E}(\tilde{b}_i) = \mu_i, \quad \sigma_i = 1 \tag{4.1.17}$$

但是有了这些分布信息，我们如何去处理呢？一个可选的方法就是，根据这些分布信息，生成若干个具有代表性的场景 (Scenarios)，每一个场景就对应一组参数，一个模型，并且这个场景有其对应的概率。我们可以用这些场景，去代表现实情况的所有可能。基于这些场景，我们可以给出一个综合考虑了随机因素的生产计划。这种方法也叫基于场景的建模方法 (Scenario-based)。至于如何使得生成的场景比较具有代表性，那就又是一大部分内容了，这里我们不做具体展开。(不过可以先提一下，一般比较常用的就是 'Sample Average Approximation, SAA')

为了方便，我们只考虑 a 不确定，c 和 b 的不确定在这里先不做考虑。首先，我们假设确定性模型中的参数 a 的取值就等于每个参数的均值，且所有参数的方差均为 0.2，且服从正态分布，即

$$\mathbb{E}(\tilde{a}_{11}) = \mu_{11} = 1, \quad \sigma_{11} = 0.2 \tag{4.1.18}$$

$$\mathbb{E}(\tilde{a}_{12}) = \mu_{12} = 2, \quad \sigma_{12} = 0.2 \tag{4.1.19}$$

$$\mathbb{E}(\tilde{a}_{21}) = \mu_{21} = 4, \quad \sigma_{21} = 0.2 \tag{4.1.20}$$

$$\mathbb{E}(\tilde{a}_{32}) = \mu_{32} = 4, \quad \sigma_{32} = 0.2 \tag{4.1.21}$$

比如，根据这个信息，在实际的情况中，参数可能是这样的

$$\max \ 2x_1 + 3x_2 \tag{4.1.22}$$

$$s.t. \quad 1.10x_1 + 2.13x_2 \leqslant 8 \tag{4.1.23}$$

$$3.73x_1 \qquad \leqslant 16 \tag{4.1.24}$$

$$3.90x_2 \leqslant 12 \tag{4.1.25}$$

$$x_1, x_2 \geqslant 0 \tag{4.1.26}$$

但是也有可能是这样的

$$\max\ 2x_1 + 3x_2 \tag{4.1.27}$$

$$s.t.\quad 0.95x_1 + 2.11x_2 \leqslant 8 \tag{4.1.28}$$

$$3.78x_1 \qquad \leqslant 16 \tag{4.1.29}$$

$$4.22x_2 \leqslant 12 \tag{4.1.30}$$

$$x_1, x_2 \geqslant 0 \tag{4.1.31}$$

总之，就是有无数种可能。我们无法穷举。

4.1.3 随机规划模型（Stochastic Programming）

为了能够处理这种情况，我们生成 K 个非常非常具有代表性的场景 (Scenarios), 假设每种场景出现的可能性为 $p_k, k \in K$，且 $\sum_{k\in K} q_k = 1$。则生产计划的随机规划模型就可以写成

$$\max \sum_{k\in K} p_k(c_1x_1^k + c_2x_2^k) \tag{4.1.32}$$

$$s.t.\quad a_{11}^k x_1^k + a_{12}^k x_2^k \leqslant 8, \quad \forall k \in K \tag{4.1.33}$$

$$a_{21}^k x_1^k \qquad\qquad \leqslant 16, \quad \forall k \in K \tag{4.1.34}$$

$$a_{22}^k x_2 \leqslant 12, \quad \forall k \in K \tag{4.1.35}$$

$$x_1^k, x_2^k \geqslant 0, \quad \forall k \in K \tag{4.1.36}$$

即，在每一种场景 k 下，我们都需要决策出那种场景 k 下的生产计划 x_1^k, x_2^k，我们最终的计划，要使得我们考虑的 K 个场景下的总期望收益最大化，即 $\max \sum_{k\in K} p_k(c_1x_1^k + c_2x_2^k)$。

我们将上述模型再写地紧凑一些，即为

$$\max \sum_{k\in K} p_k \sum_{i\in I} c_i x_i^k \tag{4.1.37}$$

$$s.t.\quad \sum_{i\in I} a_{ji}^k x_i^k \leqslant b_j, \qquad \forall j \in J, \forall k \in K \tag{4.1.38}$$

$$x_i^k \geqslant 0, \qquad \forall i \in I, \forall k \in K \tag{4.1.39}$$

其中 $I = \{1, 2\}$ 表示产品的集合，$J = \{1, 2, 3\}$ 表示资源的集合。

这就是非常常见的随机规划模型。一些相关论文的模型本质上跟上面的形式是类似的。

4.1.4 Python 调用 Gurobi 求解随机规划模型

接下来我们用 Python 调用 Gurobi 对上述随机规划进行求解。代码中的 Scenarios‘不是用 SAA‘做的，而是我根据分布信息随机生成的，只是为了展示一下这个过程。

我们考虑 5 种场景，即 $K = 5$，且 $p_i = 0.2, i = 1, 2, \cdots, 5$。结果如下

Code

```
scenario_num = 5
sto_model = Model('Stochastic Programming')
prob = [0.2, 0.2, 0.2, 0.2, 0.2]
# prob = [0.1, 0.2, 0.3, 0.15, 0.25]
profit = [2, 3]
b = [8, 16, 12]

# generate parameters under scenarios
a_11 = np.random.normal(1, 0.2, scenario_num)  # mu, sigma, sample_number
a_12 = np.random.normal(2, 0.2, scenario_num)
a_21 = np.random.normal(4, 0.2, scenario_num)
a_32 = np.random.normal(4, 0.2, scenario_num)

x_sto = {}
for i in range(2):
    # creat variables
    for s in range(scenario_num):
        x_sto[i, s] = sto_model.addVar(lb = 0, ub = GRB.INFINITY, vtype = GRB.CONTINUOUS, name =
        ↪  'x_' + str(i) + '_' + str(s))

# set objective
obj = LinExpr(0)
for key in x_sto.keys():
    product_ID = key[0]
    scenario_ID = key[1]
    obj.addTerms(prob[scenario_ID] * profit[product_ID], x_sto[key])
sto_model.setObjective(obj, GRB.MAXIMIZE)

# add constraints:
# constraints 1
for s in range(scenario_num):
    lhs = LinExpr(0)
    lhs.addTerms(a_11[s], x_sto[0, s])
    lhs.addTerms(a_12[s], x_sto[1, s])
    sto_model.addConstr(lhs <= b[0], name = 'cons_' + str(0))

# constraints 2
for s in range(scenario_num):
    lhs = LinExpr(0)
    lhs.addTerms(a_21[s], x_sto[0, s])
    sto_model.addConstr(lhs <= b[1], name = 'cons_' + str(1))

# constraints 3
```

```
for s in range(scenario_num):
    lhs = LinExpr(0)
    lhs.addTerms(a_32[s], x_sto[1, s])
    sto_model.addConstr(lhs <= b[2], name = 'cons_' + str(2))

# solve
sto_model.optimize()

print('Obj = ', sto_model.ObjVal)
# for key in x_sto.keys():
#     print(x_sto[key].varName, ' = ', x_sto[key].x)
for s in range(scenario_num):
    print(' ------  scenario   ', s, '  ------- ')
    for i in range(2):
        print(x_sto[i,s].varName, ' = ', x_sto[i,s].x)
    print('\n\n')
```

求解结果如下

Code

```
Solved in 4 iterations and 0.01 seconds
Optimal objective  1.488085782e+01
Obj =  14.880857821250329
 ------  scenario    0   -------
x_0_0  =  3.8650223745517267
x_1_0  =  2.4927517036690205



 ------  scenario    1   -------
x_0_1  =  3.999245964351986
x_1_1  =  3.0288732634647424



 ------  scenario    2   -------
x_0_2  =  1.706111126833073
x_1_2  =  3.0604202728741234



 ------  scenario    3   -------
x_0_3  =  3.9331434475778084
x_1_3  =  2.4207092574101026



 ------  scenario    4   -------
x_0_4  =  4.003138883536607
```

```
x_1_4  =  2.127567340098422
```

改变场景的出现概率，比如考虑

Code

```
prob = [0.1, 0.2, 0.3, 0.15, 0.25]
```

则求解结果如下：

Code

```
Solved in 5 iterations and 0.01 seconds
Optimal objective  1.406161189e+01
Obj =  14.061611891889607
 ------  scenario    0   -------
x_0_0  =  4.371132256077677
x_1_0  =  1.6122311882908937



 ------  scenario    1   -------
x_0_1  =  3.9240152749431267
x_1_1  =  2.280958241360689



 ------  scenario    2   -------
x_0_2  =  2.1745705126924615
x_1_2  =  3.2383131316117613



 ------  scenario    3   -------
x_0_3  =  4.215519897219448
x_1_3  =  2.0395998835922193



 ------  scenario    4   -------
x_0_4  =  3.992288784058322
x_1_4  =  1.82358745935411
```

可以看到，随机规划实际上是给出了一系列的解决方案。在每一种场景下，都会有一组解。

- 可以这么去理解，一个 Scenario 就是一种实际情况的可能。比如说，今天做好了生产计划，可是到了明天，‘设备 1‘真的出问题了，其加工时长变长了，那就对应一个场景。如果加工时长没变，但是有 ‘员工 1‘状态非常好，速度比平均状态快了，那就对应另外一种场景……所有的这些可能出现的情况，都可以用我们生成的这 ‘5 种 ‘Scenario 去代表，也就是说，到了第二天，我们发现真实情况是 ‘情况 1‘，我们拿 ‘情况 1‘与我们考虑的 ‘5 种 ‘Scenario 对比，发现 ‘情况 1‘和 ‘Scenario 2‘非常相似，那么此时，‘Scenario 2‘就可以代表 ‘情况 1‘。同样的，其他情况也可以通过这种方式去对应。也就是说，‘5 种 ‘Scenario 就近似的代表了所有可能发生的情况，并且我们的 ‘随机规划模型 ‘提供了每种 ‘Scenario‘下的相应的解，只需要对应去执行就可以了。
- 而我们的目标函数，是这些所有情况下的总收益的期望值最大。即，不管明天发生什么样的随机情况，我们期望能获得的收益就是那么多。至于具体获得多少，那就看具体实际情况是哪种 ‘Scenario‘了。

总结一下，随机规划实际上是给出了你考虑所有情形下的解。这些解的平均目标函数是达到了最大。在实际中，发生了哪一种对应的情况，我们就去执行那种场景下对应的解即可。

上面的例子给大家直观的展示了什么是随机规划。想要继续深入的读者，可以自行阅读相关书籍 (参考文献中给出了几个参考书籍)。

参考书籍：Shapiro 2003、Birge and Louveaux 2011。

Chapter 5

2021-01-17: Python 调用 Gurobi: Assignment Problem 简单案例

刘兴禄，清华大学，深圳国际研究生院，清华伯克利深圳学院（博士在读）
CSDN 主页：https://blog.csdn.net/HsinglukLiu?spm=1011.2124.3001.5343

5.1 Python 调用 Gurobi: Assignment Problem 简单案例

5.1.1 Assignment Problem Model

我们考虑一个打车订单分配问题，假设有一个乘客 'rider'集合 R, 一个司机 'driver'的集合 D, 我们考虑顾客数量和司机数量相等的情况，来将顾客和司机进行一对一的匹配，这个问题可以建模为一个整数规划 'Integer Programming'，但是这个问题非常特殊，将 'binary constraints'松弛掉，依然存在整数最优解。模型如下：

$$\max \sum_{(i,j)\in A} x_{ij} p_{ij} \tag{5.1.1}$$

$$\sum_{i} x_{ij} = 1, \qquad \forall j \in D \tag{5.1.2}$$

$$\sum_{j} x_{ij} = 1, \qquad \forall i \in R \tag{5.1.3}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in R, j \in D \tag{5.1.4}$$

首先我们生成一个 profit 矩阵，也就是两两匹配的收益矩阵

Code

```
from gurobipy import *
```

```
import pandas as pd
import numpy as np
import random

# generate matching value matrix
order_num = 20
profit_matrix = np.zeros((order_num, order_num))
for i in range(order_num):
    for j in range(order_num):
        if(i == j):
            profit_matrix[i][j] = 0
        else:
            random.seed(i * order_num + j)
            profit_matrix[i][j] = round(10 * random.random(), 1)
profit_matrix

P = pd.DataFrame(profit_matrix)
P
```

具体如下：

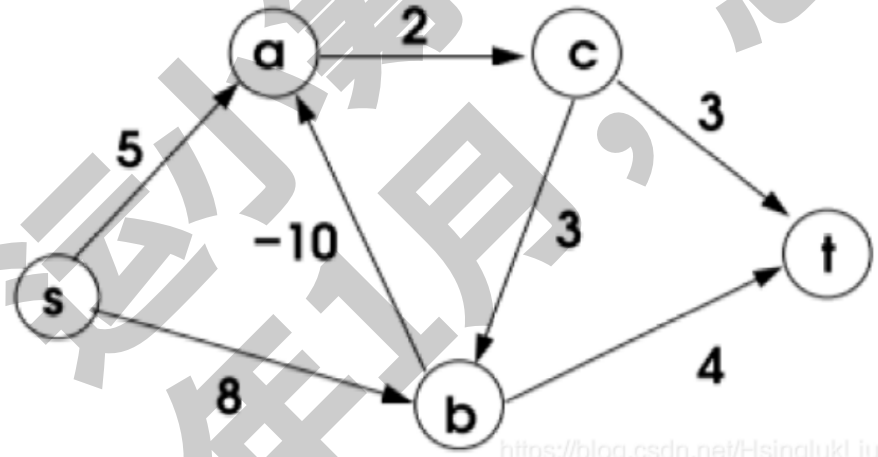


Figure 5.1.1: *profit* 矩阵

5.1.2 Python 调用 Gurobi 建模求解 Assignment Problem

我们用 ‘python‘调用 ‘gurobi‘建立上述模型，并直接调用 ‘gurobi‘的算法进行求解，结果如下：注：这里主要用到的函数有：

- `model.addVar(lb, ub, vtype, name)`
- `expr.addTerms(coef, var)`
- `model.addConstr()`

- model.setObjective(expr, GRB.MAXIMIZE)
- model.write('model.lp')
- model.optimize()
- model.getVars()
- var.varName
- var.x

等常用函数，其中model 是gurobipy.Model类型的对象，var是gurobipy.Var类的对象，gurobipy.Var的 x属性表示变量 var在当前求得的结果中的具体值。

当然还涉及到下面几个类：

- LinExpr类：用来构建线性表达式
- gurobipy.Model类：用来构建 'model'对象

Code

```
# construct model object
model = Model('Assignment_Problem')

# introduce decision variable by cycling
x = [[[] for i in range(order_num)] for j in range(order_num)]
for i in range(order_num):
    for j in range(order_num):
        x[i][j] = model.addVar(lb = 0
                               ,ub = 1
                               ,vtype = GRB.CONTINUOUS   # decision variable type
                               ,name = "x_" + str(i) + "_" + str(j)
                               )
#         x[i][j] = model.addVar(vtype = GRB.BINARY   # decision variable type
#                                ,name = "x_" + str(i) + "_" + str(j)
#                                )

# objective function
obj = LinExpr(0)

for i in range(order_num):
    for j in range(order_num):
        obj.addTerms(profit_matrix[i][j], x[i][j])

model.setObjective(obj, GRB.MAXIMIZE)

# Constraint 1
for j in range(order_num):
    expr = LinExpr(0)
    for i in range(order_num):
        expr.addTerms(1, x[i][j])
    model.addConstr(expr == 1, name="D_" + str(i))

```

```python
# Constraint 2
for i in range(order_num):
    expr = LinExpr(0)
    for j in range(order_num):
        expr.addTerms(1, x[i][j])
    model.addConstr(expr == 1, name="R_" + str(i))

# solve the constructed model
model.write('model.lp')
model.optimize()

# print optimal solution
for var in model.getVars():
    if(var.x > 0):
        print(var.varName, '\t', var.x)
```

输出结果如下：

可以看到，即使把变量 x_{ij}‘GRB.BIINARY‘改变成 ‘GRB.CONTINOUS‘，最优解仍然是整数解

Code

```
Warning: linear constraint 0 and linear constraint 1 have the same name "D_19"
Gurobi Optimizer version 9.0.1 build v9.0.1rc0 (win64)
Optimize a model with 40 rows, 400 columns and 800 nonzeros
Model fingerprint: 0x4f6b3c13
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e-01, 1e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Presolve time: 0.01s
Presolved: 40 rows, 400 columns, 800 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.8840000e+02   1.500000e+01   0.000000e+00      0s
       9    1.8650000e+02   0.000000e+00   0.000000e+00      0s

Solved in 9 iterations and 0.01 seconds
Optimal objective  1.865000000e+02
x_0_15      1.0
x_1_0       1.0
x_2_16      1.0
x_3_10      1.0
x_4_13      1.0
x_5_19      1.0
x_6_4       1.0
```

```
x_7_12      1.0
x_8_18      1.0
x_9_5      1.0
x_10_17      1.0
x_11_8      1.0
x_12_3      1.0
x_13_6      1.0
x_14_9      1.0
x_15_2      1.0
x_16_7      1.0
x_17_11      1.0
x_18_1      1.0
x_19_14      1.0
```

Chapter 6

2021-01-20：两阶段随机规划 (Two-stage Stochastic Programming)：一个详细的例子

刘兴禄，清华大学，深圳国际研究生院，清华伯克利深圳学院（博士在读）
CSDN 主页：https://blog.csdn.net/HsinglukLiu?spm=1011.2124.3001.5343

6.1 初识随机规划 (2)：两阶段随机规划 (一个详细的例子)

本文中的确定性问题的例子来源于《运筹学》第四版，运筹学编写组编著。

6.1.1 两阶段随机规划模型

上篇文章介绍了随机规划的一个小例子。
为了方便阅读，我们把那个例子拿过来，再复习一遍。

一个工厂生产 2 种产品 A 和 B，A 产品需要 1 个单位人工工时和 4 个单位的设备 1 的工时，B 产品需要 2 个单位的人工工时和 4 个单位的设备 2 的工时。且该厂一共可提供人工工时、设备 1 工时和设备 2 的工时分别为 8,16 和 12。且生产单位产品 A、B 的净利润分别为 2 和 3。假设该工厂生产的产品是供不应求的，问，该工厂应该如何生产 (可以是连续的量)，使得净利润最大化。根据上述描述，我们引入下面的决策变量：

x_1, x_2 分别代表生产产品 A B 的量，则该问题的数学模型为

$$\max\ 2x_1 + 3x_2 \tag{6.1.1}$$
$$s.t.\quad x_1 + 2x_2 \leqslant 8 \tag{6.1.2}$$
$$4x_1 \leqslant 16 \tag{6.1.3}$$
$$4x_2 \leqslant 12 \tag{6.1.4}$$
$$x_1, x_2 \geqslant 0 \tag{6.1.5}$$

但是上面的文章只是让读者直观的感受一下随机规划究竟是什么，文中的随机规划模型如下：

$$\max \sum_{k\in K} p_k(c_1x_1^k + c_2x_2^k) \tag{6.1.6}$$
$$s.t.\quad a_{11}^k x_1^k + a_{12}^k x_2^k \leqslant 8,\quad \forall k \in K \tag{6.1.7}$$
$$a_{21}^k x_1^k \leqslant 16,\quad \forall k \in K \tag{6.1.8}$$
$$a_{32}^k x_2 \leqslant 12,\quad \forall k \in K \tag{6.1.9}$$
$$x_1^k, x_2^k \geqslant 0,\quad \forall k \in K \tag{6.1.10}$$

其紧凑形式为

$$\max \sum_{k\in K} p_k \sum_{i\in I} c_i x_i^k \tag{6.1.11}$$
$$s.t.\quad \sum_{i\in I} a_{ji}^k x_i^k \leqslant b_j,\qquad \forall j \in J, \forall k \in K \tag{6.1.12}$$
$$x_i^k \geqslant 0,\qquad \forall i \in I, \forall k \in K \tag{6.1.13}$$

其中 $I = \{1, 2\}$ 表示产品的集合，$J = \{1, 2, 3\}$ 表示资源的集合。a_{ji}^k 表示第 $j \in J$ 种资源被第 $i \in I$ 种商品在第 $k \in K$ 个场景下需要的量。

细心的读者会发现，上面的模型，其实可以等价为：分别求解 K 个线性规划，然后将 K 个单独的线性规划的目标值加权求和一下。原因是这 K 个场景下的模型没有任何联系，是相互独立的。

难道说随机规划就是很多个确定性线性规划简单的分别求解一下，然后把目标函数加和

一下这么简单吗？当然不是。

6.1.2 3 层决策：战略层（strategic），战术层（tactical）和作业层（operational）

一般情况下，随机规划都是有多个阶段的决策的。一般来讲，决策有三个层次：战略层（strategic），战术层（tactical）和作业层（operational）。从 strategic 到 tactical 再到 operational，决策的周期越来越短，决策的事情越来越具体化。随机规划中，比较常见的是两个阶段的决策 two-stage decision，因此，读论文的时候，经常看到关于随机规划的文章是把问题建模为一个 two-stage mixed integer programming 的模型求解的。

所谓的 two-stage，一般可以认为包含两种决策，一种是比较长期的，上层的决策，比如服务网络怎么搭建，物流网络怎么搭建，基础设施怎么搭建等，这种决策会影响企业较长时间的发展，我们称其为战略层决策 (strategic level)，有的决策也许没有到战略层的级别，可以称为战术层决策 (tactical level)。第二种决策就是非常具体化的，比如生产计划，采购计划，VRP 的路径规划等，都是非常具体、细节的决策，这一类决策对企业的发展影响周期较短，可以称为作业层决策 (operational level)。为了方便统一，本文中我们统一使用 tactical level 和 operational level 来表示上层决策和下层决策。

6.1.3 Two-stage Stochastic Programming

6.1.4 Two-stage Stochastic Programming 的直观写法

还是回到随机规划上来。我们回去看之前的例子，例子中说，我们有生产产品 A 和产品 B 的设备 1 和设备 2，有了设备，我们可以决定生产 A 和 B。但是在实际中，也许我们可以再往前座一层决策，那就是，我们决策一下，要不要接产品 A 和产品 B 的订单，也就是要不要生产产品 A 和产品 B。例如，如果要生产 A，我们就需要准备生产 A 所需的设备等，这些活动是需要花费一些成本的，我们可以认为这些成本是固定的，即固定成本或者启动成本，固定成本可以认为就是需要去购置机器，雇佣人员等。这个决策，可以看做是上层决策 (tactical level)，很明显，这个决策并没有涉及具体生产多少产品的决策。

在决定了要生产 A 以后，也准备好了所有的所需材料和人工，我们基于这些材料，再去制定具体要生产多少商品的生产计划。这个生产计划，就是非常具体的决策，可以看做是作业层决策 (operational level), 即下层决策。

假设决定生产产品 A 和产品 B 的固定成本分别为 f_A 和 f_B。我们引入上层决策 (tactical level) 的决策变量：

$$z_i \in \{0,1\}, i = A \text{ or } B \tag{6.1.14}$$

表示我们决定是否生产产品 A 或者 B。

之后，具体操作层的决策，就是每种产品生产多少，即为 x_i^k，在第 k 个场景下，生产产品 i 的数量。

因此，Two-stage Stochastic Programming 的模型可以写成

$$\max \sum_{k\in K} p_k \left(\sum_{i\in I} c_i x_i^k \right) - \sum_{i\in I} f_i z_i \tag{6.1.15}$$

$$s.t. \quad \sum_{i\in I} a_{ji}^k x_i^k \leqslant b_j, \quad \forall j \in J, \forall k \in K \tag{6.1.16}$$

$$\sum_{k\in K} x_i^k - M z_i \leqslant 0, \quad \forall i \in I \tag{6.1.17}$$

$$z_i \in \{0,1\}, x_i^k \geqslant 0, \quad \forall i \in I, \forall k \in K \tag{6.1.18}$$

第 2 条约束表示，只要有一个场景下，$x_i^k > 0$，则 $z_i = 1$。也就是，只要有一种情况下，生产了产品 $i \in \{A, B\}$，则我们就需要在一开始就要决定生产产品 $i \in \{A, B\}$，且准备好生产 $i \in \{A, B\}$ 的所有预备材料。

上面的模型虽然包含了两阶段决策：上层决策 (tactical level) 和作业层决策 (operational level)。但是模型看上去看是单阶段的模型。不过上面的写法是没问题的，只是没有体现出两阶段的思想。

Two-stage Stochastic Programming 的更学术的写法

为了体现出两阶段的思想，我们换一种更专业，更学术的写法，学术论文里大家一般也都是按照下面的方法写的。

我们按照事情发生的先后顺序的逻辑来写。

首先，我们需要做一个上层决策 (tactical level)，这个决策即为：要不要生产产品 A 和产品 B。做出的决策必须要使得总的期望净利润最大化。即

$$\max \quad \mathbb{E}_p \left[h\left(\mathbf{z}, K\right) \right] - \sum_{i\in I} f_i z_i \tag{6.1.19}$$

$$z_i \in \{0,1\}, \qquad \forall i \in I \tag{6.1.20}$$

这里，符号上可能有些难理解，我在这里详细的解释一下。$\mathbb{E}_p \left[h\left(\mathbf{z}, K\right) \right]$，就是在各个场景出现的概率为 p 时 (例如 $p = [0.2, 0.2, 0.2, 0.2, 0.2]$)，决策变量 $\mathbf{z} = [z_A, z_B]^T$ 的取值确定以后，在所有场景 K 下的收入的期望值。实际上，就是第二阶段决策产生的收入的期望值，但

是现在我们还没有决策第二阶段的生产计划，因此我们先用一个符号去表示第二阶段的期望收入值。

当我们确定了是否要生产产品 A、B 的决策 $\mathbf{z}=[z_A,z_B]^T$ 以后，第二阶段，也就是制定生产计划的阶段，我们会面临不确定的情况，也就是面临多种场景 $\forall k\in K$。对于每一种场景 $k\in K$，当这种场景 k 真的发生了的时候，我们需要做出在这种场景下的最优的生产计划。最优的生产计划可以用下面的模型规划出来

> 注意，之前是 $h(\mathbf{z},K)$, 是指在所有的场景 K 下的所有决策，这里我们固定了某一种场景 k，因此变成了 $h(\mathbf{z},k)$。此时 $\mathbf{z}$ 已经可以看做是一个给定的值，因为在做第二阶段生产计划的时候，决定生产还是不生产 A 和 B 是已经知道的事实

$$h(\mathbf{z},k)=\max_{\mathbf{x}(k)}\sum_{i\in I}c_ix_i^k \tag{6.1.21}$$

$$s.t.\quad \sum_{i\in I}a_{ji}^kx_i^k\leqslant b_j,\qquad \forall j\in J,\forall k\in K \tag{6.1.22}$$

$$\sum_{k\in K}\sum_{i\in I}x_i^k-Mz_i\leqslant 0,\quad \forall i\in I \tag{6.1.23}$$

$$x_i^k\geqslant 0,\qquad \forall i\in I,\forall k\in K \tag{6.1.24}$$

怎么样，这样写，是不是就比较好理解两个阶段的决策上层决策 (tactical level) 和作业层决策 (operational level) 之间是什么关系了。

我们来汇总一下上面的两阶段模型。

首先，第一阶段，作为企业的决策者，我们需要作出是否生产产品 A 和 B 的上层决策 (tactical level decisions)，使得总的期望净利润最大化。这一阶段的决策，是和随机场景无关的，无论场景怎么变，这一阶段的决策产生的固定成本都是相同的。模型为

$$\max\quad \mathbb{E}_p\left[h(\mathbf{z},K)\right]-\sum_{i\in I}f_iz_i \tag{6.1.25}$$

$$z_i\in\{0,1\},\qquad \forall i\in I \tag{6.1.26}$$

制定好了上层决策之后，真正到了要生产的时候，我们观察到某种可能的场景 $k\in K$ 发生了，我们需要作出该场景下的最优生产决策，即：第二阶段的决策：亦即下层决策 (operational level decisions)，对于场景 $k\in K$，我们有

$$h\left(\mathbf{z},k\right)=\max_{\mathbf{x}(k)}\sum_{i\in I}c_ix_i^k \tag{6.1.27}$$

$$s.t.\quad \sum_{i\in I}a_{ji}^kx_i^k\leqslant b_j,\quad \forall j\in J,\forall k\in K \tag{6.1.28}$$

$$\sum_{k\in K}\sum_{i\in I}x_i^k-Mz_i\leqslant 0,\qquad \forall i\in I \tag{6.1.29}$$

$$x_i^k\geqslant 0,\qquad \forall i\in I,\forall k\in K \tag{6.1.30}$$

怎么样，这么写是不是高大上一些。以后小伙伴们写论文，大概都需要这么写的。这样比较专业一些。如果按照之前的那种写法，可能会被喷不专业。哈哈哈。

两阶段的写法比较容易理解两阶段决策的逻辑，但在真正求解的时候，是和前面那种写法是完全一样的。为了方便大家查看，我再抄一遍过来。

合并书写的方式

$$\max\sum_{k\in K}p_k\left(\sum_{i\in I}c_ix_i^k\right)-\sum_{i\in I}f_iz_i \tag{6.1.31}$$

$$s.t.\quad \sum_{i\in I}a_{ji}^kx_i^k\leqslant b_j,\quad \forall j\in J,\forall k\in K \tag{6.1.32}$$

$$\sum_{k\in K}x_i^k-Mz_i\leqslant 0,\quad \forall i\in I \tag{6.1.33}$$

$$z_i\in\{0,1\},x_i^k\geqslant 0,\quad \forall i\in I,\forall k\in K \tag{6.1.34}$$

我们真正进行求解的时候，还是用这种写法去求解就可以。还是那句话，求解器根本不知道你的决策变量哪个是上层决策，哪个是下层决策。当你把模型丢给求解器，人家直接就求解了，管你什么一阶段二阶段呢，哈哈哈。除非你自己根据一阶段二阶段设计了专门的算法，先固定哪个，后做哪个。比如说，你用 Benders Decomposition，把第一阶段的 z 作为复杂变量，把第二阶段的 x 作为简单变量，那就另当别论。

至于随机规划更为复杂的例子，大家可以去读我的母系，工业工程系毕业的黄一潇师兄 2017 年发在 TRB 上的论文 Time-dependent vehicle routing problem with path flexibility。(Huang et al. 2017)

下面我们继续用 Python 调用 Gurobi 来求解上述 Two-stage Stochastic Programming。

6.1.5 Python 调用 Gurobi 求解 Two-stage Stochastic Programming

我们沿用前一篇文章的所有关于均值和方差的假设，即

$$\mathbb{E}(\tilde{a}_{11}) = \mu_{11} = 1, \quad \sigma_{11} = 0.2 \tag{6.1.35}$$

$$\mathbb{E}(\tilde{a}_{12}) = \mu_{12} = 2, \quad \sigma_{12} = 0.2 \tag{6.1.36}$$

$$\mathbb{E}(\tilde{a}_{21}) = \mu_{21} = 4, \quad \sigma_{21} = 0.2 \tag{6.1.37}$$

$$\mathbb{E}(\tilde{a}_{32}) = \mu_{32} = 4, \quad \sigma_{32} = 0.2 \tag{6.1.38}$$

我们设置 $f_A = 6, f_B = 4$。我们用 Python 调用 Gurobi 对其进行求解。

为了方便与只有操作层决策的版本进行对比，我们首先将不确定参数固定住，即保证二者使用同一组约束系数。

Code

```
scenario_num = 5
# generate parameters under scenarios
a_11 = np.random.normal(1, 0.2, scenario_num)  # mu, sigma, sample_number
a_12 = np.random.normal(2, 0.2, scenario_num)
a_21 = np.random.normal(4, 0.2, scenario_num)
a_32 = np.random.normal(4, 0.2, scenario_num)
print('a_11:', a_11)
print('a_12:', a_12)
print('a_21:', a_21)
print('a_32:', a_32)
```

产生的随机输入参数为

Code

```
a_11:  [0.90039351 1.38590641 1.18988416 1.01751025 0.7549129 ]
a_12:  [2.1688726  1.79995693 1.69104578 2.23760596 2.06338852]
a_21:  [4.18417176 4.06374553 4.17136612 3.86979488 3.79315143]
a_32:  [4.1363189  3.83931807 3.86209004 3.9088935  4.00349583]
```

Two-stage Stochastic Programming

Code

```
from gurobipy import *


sto_model = Model('Stochastic Programming')
prob = [0.2, 0.2, 0.2, 0.2, 0.2]
# prob = [0.1, 0.2, 0.3, 0.15, 0.25]
profit = [2, 3]
fix_cost = [6, 4]
b = [8, 16, 12]
big_M = 1000

```

```
12
13  x_sto = {}
14  z = {}
15  for i in range(2):
16      # creat variables
17      z[i] = sto_model.addVar(lb = 0, ub = 1, vtype = GRB.BINARY, name = 'z_' + str(i) + '_' + str(i))
18      for s in range(scenario_num):
19          x_sto[i, s] = sto_model.addVar(lb = 0, ub = GRB.INFINITY, vtype = GRB.CONTINUOUS, name =
            ↪  'x_' + str(i) + '_' + str(s))
20
21  # set objective
22  obj = LinExpr(0)
23  for key in x_sto.keys():
24      product_ID = key[0]
25      scenario_ID = key[1]
26      obj.addTerms(prob[scenario_ID] * profit[product_ID], x_sto[key])
27  for key in z.keys():
28      obj.addTerms(-fix_cost[key], z[key])
29
30  sto_model.setObjective(obj, GRB.MAXIMIZE)
31
32  # add constraints:
33  # constraints 1-1
34  for s in range(scenario_num):
35      lhs = LinExpr(0)
36      lhs.addTerms(a_11[s], x_sto[0, s])
37      lhs.addTerms(a_12[s], x_sto[1, s])
38      sto_model.addConstr(lhs <= b[0], name = 'cons_' + str(0))
39
40  # constraints 1-2
41  for s in range(scenario_num):
42      lhs = LinExpr(0)
43      lhs.addTerms(a_21[s], x_sto[0, s])
44      sto_model.addConstr(lhs <= b[1], name = 'cons_' + str(1))
45
46  # constraints 1-3
47  for s in range(scenario_num):
48      lhs = LinExpr(0)
49      lhs.addTerms(a_32[s], x_sto[1, s])
50      sto_model.addConstr(lhs <= b[2], name = 'cons_' + str(2))
51
52  # constraints 2
53  for i in range(2):
54      lhs = LinExpr(0)
55      for s in range(scenario_num):
56          lhs.addTerms(1, x_sto[i, s])
57      sto_model.addConstr(lhs <= big_M * z[i])
58
59  # solve
60  sto_model.optimize()
```

```

print('Obj = ', sto_model.ObjVal)
print('\n\n  ------  tactical level decision  ------- ')
for key in z.keys():
    print(z[key].varName, ' = ', z[key].x)
print('\n\n ------  operational level decision  ------- ')
for s in range(scenario_num):
    print(' ------  scenario   ', s, '  ------- ')
    for i in range(2):
        print(x_sto[i,s].varName, ' = ', x_sto[i,s].x)
    print('\n\n')
```

求解结果为

Code

```
 -- Obj =  5.120668471999894


  ------  tactical level decision  -------
z_0_0  =  0.0
z_1_1  =  1.0


 ------  operational level decision  -------
 ------  scenario    0   -------
x_0_0  =  0.0
x_1_0  =  2.901130275374204


 ------  scenario    1   -------
x_0_1  =  0.0
x_1_1  =  3.1255550569323436


 ------  scenario    2   -------
x_0_2  =  0.0
x_1_2  =  3.107125898642543


 ------  scenario    3   -------
x_0_3  =  0.0
x_1_3  =  3.069922473497831


 ------  scenario    4   -------
```

```
x_0_4  =  0.0
x_1_4  =  2.997380415552898
```

可以看到，我们最终是决定，不生产产品 A，只生产产品 B，因为产品 A 的固定成本高 ($f_A = 6$)，但是利润比较低 (为 2)。而产品 B 的固定成本低 ($f_A = 4$)，而且利润高（为 3）。

接下来，我们就对比一下之前那种只有操作层的版本。

Single stage version

Code

```python
scenario_num = 5
sto_model = Model('Stochastic Programming')
prob = [0.2, 0.2, 0.2, 0.2, 0.2]
# prob = [0.1, 0.2, 0.3, 0.15, 0.25]
profit = [2, 3]
b = [8, 16, 12]

# generate parameters under scenarios
# a_11 = np.random.normal(1, 0.2, scenario_num)  # mu, sigma, sample_number
# a_12 = np.random.normal(2, 0.2, scenario_num)
# a_21 = np.random.normal(4, 0.2, scenario_num)
# a_32 = np.random.normal(4, 0.2, scenario_num)

x_sto = {}
for i in range(2):
    # creat variables
    for s in range(scenario_num):
        x_sto[i, s] = sto_model.addVar(lb = 0, ub = GRB.INFINITY, vtype = GRB.CONTINUOUS, name =
        ↪  'x_' + str(i) + '_' + str(s))

# set objective
obj = LinExpr(0)
for key in x_sto.keys():
    product_ID = key[0]
    scenario_ID = key[1]
    obj.addTerms(prob[scenario_ID] * profit[product_ID], x_sto[key])
sto_model.setObjective(obj, GRB.MAXIMIZE)

# add constraints:
# constraints 1
for s in range(scenario_num):
    lhs = LinExpr(0)
    lhs.addTerms(a_11[s], x_sto[0, s])
    lhs.addTerms(a_12[s], x_sto[1, s])
    sto_model.addConstr(lhs <= b[0], name = 'cons_' + str(0))

# constraints 2
for s in range(scenario_num):
```

```
    lhs = LinExpr(0)
    lhs.addTerms(a_21[s], x_sto[0, s])
    sto_model.addConstr(lhs <= b[1], name = 'cons_' + str(1))

# constraints 3
for s in range(scenario_num):
    lhs = LinExpr(0)
    lhs.addTerms(a_32[s], x_sto[1, s])
    sto_model.addConstr(lhs <= b[2], name = 'cons_' + str(2))

# solve
sto_model.optimize()

print('Obj = ', sto_model.ObjVal)
# for key in x_sto.keys():
#     print(x_sto[key].varName, ' = ', x_sto[key].x)
for s in range(scenario_num):
    print(' ------  scenario   ', s, '  ------- ')
    for i in range(2):
        print(x_sto[i,s].varName, ' = ', x_sto[i,s].x)
    print('\n\n')
```

求解结果如下

Code

```
 --- Obj =  13.896545328523917
 ------  scenario    0   -------
x_0_0  =  3.8239347951176703
x_1_0  =  2.101070361744041



 ------  scenario    1   -------
x_0_1  =  1.71305615957035
x_1_1  =  3.1255550569323436



 ------  scenario    2   -------
x_0_2  =  2.307542152652112
x_1_2  =  3.107125898642543



 ------  scenario    3   -------
x_0_3  =  4.134586067378273
x_1_3  =  1.6951225436724826
```

```
------ scenario    4   -------
x_0_4  =  4.218128458180001
x_1_4  =  2.333869931282845
```

是吧，这两个的结果还是完全不一样的吧。

此时，由于上层决策变量 $z_i, \forall i \in I$ 和下层决策变量 $x_i^k, \forall i \in I, k \in K$ 是有耦合关系的，是一揽子决策。场景和场景之间也是有一些关系的，比如在场景 1 下，我们生产了产品 A，即 $x_A^1 > 0$，那么 z_A 就一定为 1，因此在场景 2 下，也许就会生产 x_A^2 也许就大于 0 了。但是，也许你单独求解多个模型的时候，$x_A^2 = 0$。这种情形是有可能会出现的。在这种情况下，单独求解每个场景下的模型，然后加权求和一下，就和随机规划的模型的结果是不等价的。

所以说，随机规划肯定不是单独求解多个线性规划，然后加权求和一下。它是有自己独特的强大功能在的。

好啦，这篇文章就是完整的给大家展示了两阶段随机规划的一个小例子，从构建决策，到建模，到代码实现。相信大家读完这篇文章，应该对随机规划有了一个非常清楚的初步认识。之后如果想要继续深入，欢迎阅读下面的两本书。或者持续关注我们的公众号。

Chapter 7

2021-01-30：运筹学与管理科学 TOP 期刊揭秘—Transportation Research 系列

王基光，清华大学，深圳国际研究生院，清华伯克利深圳学院（硕士在读）
联系方式：wangjg2020@163.com

7.1 TR 系列主编汇总

TR系列主编汇总

TR系列简介

TR系列都属于爱思唯尔旗下ELSEVIER。
详细介绍
Transportation Research Part A: Policy and Practice
https://www.journals.elsevier.com/transportation-research-part-a-policy-and-practice
Transportation Research Part B: Methodological
https://www.journals.elsevier.com/transportation-research-part-b-methodological
Transportation Research Part C: Emerging Technologies
https://www.sciencedirect.com/journal/transportation-research-part-c-emerging-technologies
Transportation Research Part D: Transport and Environment
https://www.sciencedirect.com/journal/transportation-research-part-d-transport-and-environment
Transportation Research Part E: Logistics and Transportation Review
https://www.sciencedirect.com/journal/transportation-research-part-e-logistics-and-transportation-review
Transportation Research Part F: Traffic Psychology and Behaviour
https://www.sciencedirect.com/journal/transportation-research-part-f-traffic-psychology-and-behaviour

TR系列主编介绍

这里我们汇总了TR系列的各位主编和一些管理科学或交通物流领域的副主编等（其中不乏我们国内知名大学的教授，如浙江大学，北京航空航天大学等），包括他们的研究方向以及个人主页。

Figure 7.1.1: *TR 系列主编汇总 (1)*

TRA

Co-Editors-in-Chief:

J. de D. Ortúzar，PhD
智利天主教大学，圣地亚哥，智利
Pontifical Catholic University of Chile, Santiago de Chile, Chile
Juan de Dios Ortúzar维基百科
J. de D. Ortúzar（1949.1.24）,智利天主教大学（PUC）名誉教授

研究方向：
discrete choice models
valuation of externalities
design and collection of mobility and preference surveys
transportation forecasting.

最近的论文：
Gutierrez, M., Hurtubia, R. and Ortúzar, J. de D. (2020) The role of habit and the built environment in the willingness to commute by bicycle. Travel Behaviour and Society 20. 62-73.
Allen, J., Muñoz, J.C. and Ortúzar, J. de D. (2019) On evasion behavior in public transport: dissatisfaction or contagion? Transportation Research Part A: Policy and Practice 130. 626-651.
Schmidt, A., Ortúzar, J. de D. and Paredes, R.D. (2019) Heterogeneity and college choice: latent class modelling for improved policy making. Journal of Choice Modelling 33

E. Cherchi, PhD
纽卡斯尔大学 土木工程与地球科学学院，英国（北京交通大学经济管理学院兼职教授）
Newcastle University School of Civil Engineering and Geosciences, Newcastle Upon Tyne, United Kingdom
Newcastle University主页
研究方向：
discrete choice analysis,
decision making process,
bounded rationality,
user benefits,
data collection (revealed and stated preference data, panel data, virtual reality experiments),
mode choice,
electric vehicles,
autonomous vehicles.
最近论文：
Bas, J., Cirillo, C. and Cherchi, E. (2020) A State Choice experiment for considering Social Conformity in the adoption of Electric Vehicles. 12th International Conference on Transport Survey Methods. Travel Survey and Big Data: how to make the best of both worlds, Portugal. (Accepted).
Yin, H. and Cherchi, E. (2020) Conducting Stated Choice Experiments within a Virtual Reality environment: an application to the choice of automated taxi 12th International Conference on Transport Survey Methods. Travel Survey and Big Data: how to make the best of both worlds, Portugal. (Accepted).
Guerrero, T.E., Guevara, C.A., Ortúzar, J. de D., Cherchi, E. (2020) Addressing Endogeneity in Strategic Mode Choice Models. 99th Seminar on Transportation Research Board. Washington DC, USA. (Accepted)

Figure 7.1.2: *TR 系列主编汇总 (2)*

Associate Editors

J.A. Carrasco
多伦多大学，加拿大
University of Toronto, Toronto, Ontario, Canada

研究方向:
Social networks and activity-travel behaviour
Information and communication technologies (ICTs) and travel behaviour
Social exclusion and accessibility issues
Activity-based models and microsimulation approaches
Integrated land use models
Use and estimation of discrete choice and statistical models applied to travel demand analysis
Relationship between urban form and travel decisions

D. Ettema
乌得勒支大学，荷兰
Utrecht University, Utrecht, Netherlands

研究方向
Improving theoretical models of accessibility, travel and equity
Implications of accessibility for (inequalities in) travel, health and well-being
The relationship between households' locational decisions, travel behavior and accessibility
The impact of new transport services (Mobility as a Service, Car and Bicycle sharing systems) on travel, accessibility and equity
How does a transition to sustainable transportation impact on transportation and land use planning, travel behaviour and equity
个人主页
https://www.uu.nl/staff/DFEttema/Profile

J. Holguin-Veras
伦斯勒理工学院，美国
Rensselaer Polytechnic Institute, Troy, New York, United States
研究方向
Decision Support Systems
Operations Research
Transportation

个人主页
https://faculty.rpi.edu/node/34743

Figure 7.1.3: *TR 系列主编汇总 (3)*

TRB

Editor-in-Chief

C.R. Bhat, PhD
德克萨斯大学奥斯汀分校，美国
The University of Texas at Austin Department of Civil Architectural and Environmental Engineering, 301 E. Dean Keeton St., Stop C1700, Austin, Texas, 78712-1085, United States

研究方向

Travel Behavior Modeling/Travel Demand Modeling
Trip-based models
Activity-based models
Micro-simulation frameworks
Travel demand management strategies
Congestion pricing
Mobile source emissions modeling
Commuter and work-related travel
Weekend travel
Integrating activity-based frameworks and dynamic traffic assignment modules
Evacuation planning using AB and DTA modules

Socio-Demographics and Land-Use Modeling
Population updating microsimulation systems
Self-selection in residential location choice
Long-term residential mobility
Explicit incorporation of built environment

Sustainable Urban Design and Physical Activity
Studying influence of urban form on bicycle use
Understanding bicycle route choice behavior
Examining physical activity participation determinants
Non-traditional work participation behavior

Activity Participation and Time-Use
Studying activity participation
Time-use and activity-travel pattern attributes

Traffic Safety
Traffic crash analysis of driver and occupant injury severity
Examination of pedestrian and bicyclist injury severity

Social Networks and ICT
Examining how social networks influence travel
Studying the influence of ICT on travel behavior

Children's Travel Behavior
Examining children's travel and activity patterns
Examining children out-of-school activity-location engagement patterns
Connected and Automated Vehicles

Other

个人主页
https://www.caee.utexas.edu/prof/bhat/home.html

Figure 7.1.4: *TR 系列主编汇总 (4)*

Distinguished Journal Editorial Board

Richard Arnott
加州大学河滨分校，美国
University of California Riverside, Riverside, United States
个人主页
https://profiles.ucr.edu/app/home/browse/group/73512

Malachy Carey
利兹大学，英国
University of Leeds, Leeds, United Kingdom

Carlos Daganzo
加州大学伯克利分校，美国
University of California Berkeley, Berkeley, United States
研究方向
econometrics, logistics, freight operations, network theory, traffic flow, and transit operations
个人主页
https://ce.berkeley.edu/people/faculty/daganzo

Terry Friesz
宾夕法尼亚州立大学，美国
Pennsylvania State University, University Park, United States
研究方向
Transportation: congestion pricing; freight transportation; DTA; network design.
Spatial economics: spatial price equilibrium; competitive facility location
Network science: network emergence; social network theory; price of anarchy
Revenue management: service pricing; revenue optimization; demand learning
个人主页
http://www.personal.psu.edu/users/t/f/tlf13/

Fred L. Mannering
南佛罗里达大学，美国
University of South Florida, Tampa, Florida, USA
研究方向
the application of statistical and econometric methods to analyze data relating to highway safety, transportation economics, travel behavior and a variety of engineering-related problems.

Figure 7.1.5: *TR 系列主编汇总 (5)*

Hai Yang
香港科技大学，香港
Hong Kong University of Science and Technology Department of Civil and Environmental Engineering, Hong Kong, Hong Kong
研究方向

- **Smart Mobility (Ride-sharing, Ride-sourcing, Taxi Industry)**
- **Traffic and Transport Dynamics**
- **Road Pricing and Tradable Travel Credit Schemes**
- **Public Transportation, Transportation Economics**
- **Transportation Network Modeling and Optimization**

http://cehyang.people.ust.hk/

TRC

Editor-in-Chief

Nikolas Geroliminis
洛桑联邦理工学院,瑞士
Ecole Polytechnique Federal de Lausanne (EPFL), Switzerland
研究方向
His research interests focus primarily on urban transportation systems, traffic flow theory and control, public transportation and on-demand transport, car sharing, Optimization, MFDs and Large Scale Networks. He is a recipient of the ERC Starting Grant METAFERW: Modeling and controlling traffic congestion and propagation in large-scale urban multimodal networks.
https://open-traffic.epfl.ch

some of Associate Editors

Yibing Wang
浙江大学，中国
Zhejiang University, Hangzhou, China
研究方向
road traffic flow modeling, surveillance, control and optimization, and in particular, freeway traffic state estimation, freeway ramp metering and variable speed limit control, urban traffic signal control, route guidance in road networks, connected and automated vehicles (CAV)
http://iits.zju.edu.cn/international/2018/0822/c21523a847062/page.htm

Soyoung Ahn
威斯康星大学麦迪逊分校，美国
University of Wisconsin Madison, Madison, Wisconsin, United States
研究方向
traffic flow theory and operations, Intelligent Transportation Systems applications, and traffic operational impacts on environment and safety
https://directory.engr.wisc.edu/cee/Faculty/Ahn_Soyoung/

Figure 7.1.6: *TR 系列主编汇总 (6)*

TRD

Co-Editors-in-Chief

R.B. Noland
新泽西州立大学,美国
Rutgers The State University of New Jersey, New Brunswick, New Jersey, United States
研究方向
Impacts of transport planning and policy on environmental outcomes
Micro-simulation of pedestrian-vehicle interactions
Non-motorized transportation
Safety analysis
https://bloustein.rutgers.edu/noland/

J.X. Cao
明尼苏达大学，美国
University of Minnesota, Minneapolis, Minnesota, United States
研究方向
Sustainable development; transportation planning; urban and regional planning
https://www.hhh.umn.edu/directory/jason-cao

Associate Editors

K. Zhang
清华大学，中国
Tsinghua University, Beijing, China
研究方向:
主要研究方向包括智能交通系统与智慧城市，统计信号处理，高精度定位，图像处理，无线光通信等。曾在日本三菱重工的智能交通中心工作近10年,从事智能交通领域的光电子应用研究。
https://www.sigs.tsinghua.edu.cn/zk/main.htm

Y. Ge
上海海事大学，中国
Shanghai Maritime University, Shanghai, China
研究方向
交通运输网络分析及应用、交通与环境以及港航运输与物流
http://ctc.shmtu.edu.cn/node/1709

W.Y. Szeto
香港大学
University of Hong Kong, Hong Kong, Hong Kong, China
研究方向
Bicycle research
Demand modeling and management
Dynamic traffic assignment
Environmentally sustainable network design
Public transport and taxi
Transport network reliability and resilience
http://www.civil.hku.hk/pp-szetowy.html

Figure 7.1.7: *TR 系列主编汇总 (7)*

TRE

Co-Editors-in-Chief

T.M. Choi
香港理工大学
The Hong Kong Polytechnic University Institute of Textiles and Clothing, 11 Yuk Choi Road, Kowloon, Hong Kong
研究方向
Fashion supply chain management: Fast fashion, contract theory, risk analysis, sustainable operations, behavioral operations.
Fashion retailing and service operations: Pricing, social influences, luxury fashion, branding.
Business analytics and information systems management.
https://www.polyu.edu.hk/itc/en/people/academic-staff/?itcsid=10

Qiang Meng
新加坡国立大学，新加坡
National University of Singapore Department of Civil and Environmental Engineering, 1 Engineering Drive 2, E1A 07-03, 117576, Singapore, Singapore
Transportation Network Modeling and Optimization
Traffic assignment models and algorithms
Transportation network design
Origin-destination matrix estimation
Congestion pricing
Work Zone Analysis
Transit network analysis
Shipping and Intermodal Freight Transportation Analysis
Liner ship fleet deployment
Liner shipping network design
Ship speed optimization
Liner route optimization
Port hinterland estimation
Intermodal freight network design
Optimal facility location
Vehicle routing problems
Quantitative Risk Assessment (QRA) of Transport Operations
QRA model building and software development for urban road tunnels
QRA model for ship traffic in the Straits of Singapore and Malacca
Formal safety assessment model for ship traffic in port waters
Probabilistic QRA Model for work zones
https://www.eng.nus.edu.sg/cee/staff/meng-qiang/

Figure 7.1.8: *TR 系列主编汇总 (8)*

some of Associate Editors

M. Hansen
加州大学伯克利分校，美国
University of California Berkeley, Berkeley, California, United States
研究方向
urban transportation planning, air transport systems modeling, air traffic flow management, aviation systems performance analysis, aviation safety, aviation environmental analysis, and air transport economics
https://ce.berkeley.edu/people/faculty/hansen

P. T-W. Lee
浙江大学，中国
Zhejiang University, Hangzhou, China

H.-J. Huang
北京航空航天大学，中国
BeiHang University School of Economics and Management, Beijing, China
http://shi.buaa.edu.cn/huanghaijun/en/index.htm

TRF

Editor

S. Charlton
汉密尔顿怀卡托大学,新西兰
University of Waikato, Hamilton, New Zealand
研究方向
applied cognitive psychology and human factors. Current projects are in the area of driver behaviour including driver attention, perception, & performance.
https://www.waikato.ac.nz/fass/about/staff/samiam

Associate Editors

P. Delhomme
法国交通发展和网络科学技术研究所，法国
French Institute of Science and Technology for Transport Development and Networks - Laboratory on Vehicle-Infrastructure-Driver Interactions, Versailles, France

J. Krems
开姆尼茨心理学研究所，德国
TU Chemnitz Institute of Psychology, Chemnitz, Germany

Figure 7.1.9: *TR 系列主编汇总 (9)*

C.G. Prato
昆士兰大学，澳大利亚
University of Queensland, Brisbane, Australia
研究方向
Travel behaviour
Traffic psychology
Road safety analysis
Traffic assignment models
Behavioural econometrics
Freight modelling
https://researchers.uq.edu.au/researcher/14111

Figure 7.1.10: *TR 系列主编汇总 (10)*

参考文献

[1] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[2] Paola Cappanera and Maria Paola Scaparra. Optimal allocation of protective resources in shortest-path networks. *Transportation Science*, 45(1):64–80, 2011.

[3] Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.

[4] Yixiao Huang, Lei Zhao, Tom Van Woensel, and Jean-Philippe Gross. Time-dependent vehicle routing problem with path flexibility. *Transportation Research Part B: Methodological*, 95:169–195, 2017.

[5] A Shapiro. Handbooks in operations research and management science. *Monte Carlo Sampling Methods*, 10, 2003.