如何优雅地写出大规模线性规划的对偶

原创 刘兴禄 数据魔术师 今天

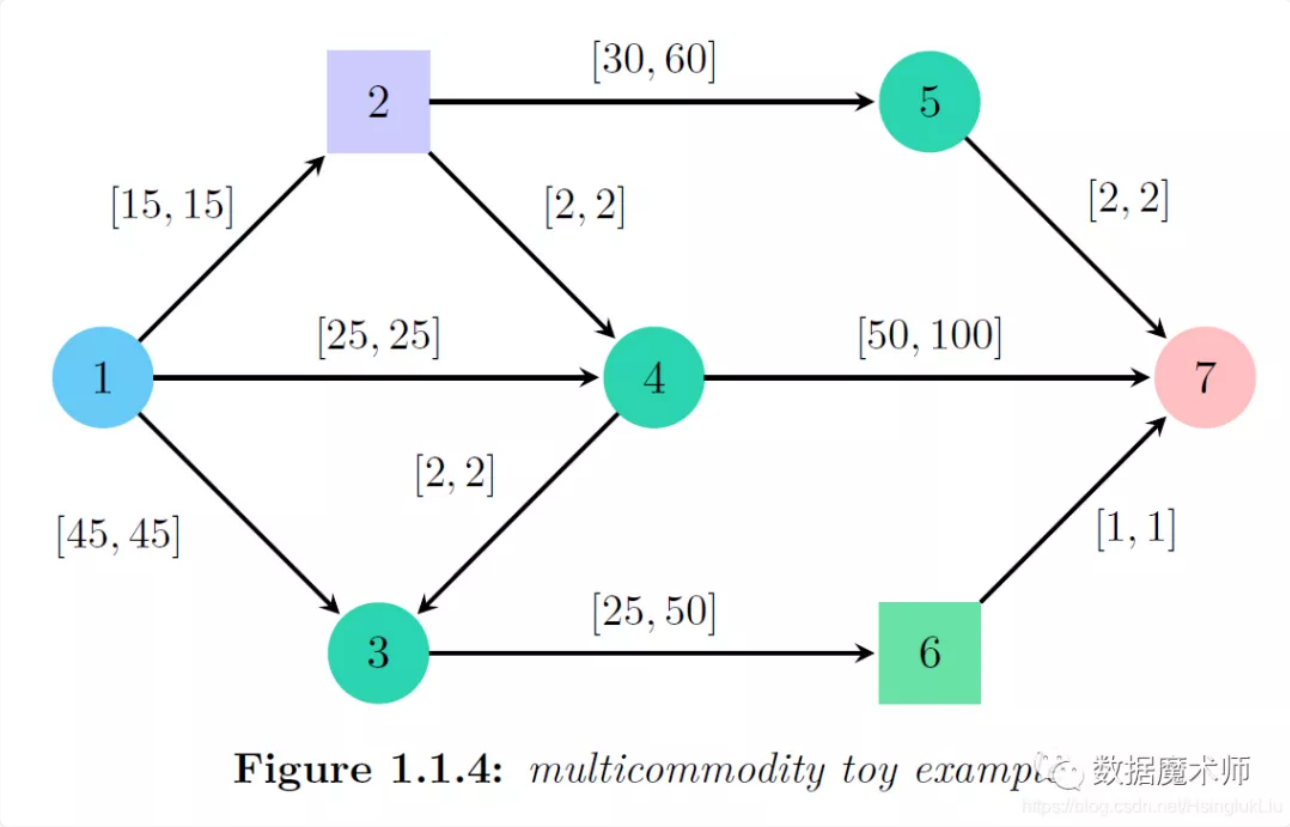


Figure 1.1.4: *multicommodity toy example*

对偶理论 Duality Theory 在运筹学数学规划部分占据着举足轻重的地位，也属于比较高阶的理论。 Duality Theory 在 精确算法设计 中也经常用到，在 Robust Optimization 等涉及到多层规划 Multi level 的问题中，也有非常广泛的应用，很多时候可以化腐朽为神奇。尤其在 Robust Optimization 中，有些问题可以巧妙的将内层 inner level 的模型转化成 LP ，从而可以通过对偶，将双层 bi-level 的模型，转化成单阶段 single level 的模型，从而用单层的相关算法来求解 RObust Optimization 问题。

今天我们就来看看，在实际的科研当中，遇到的一些稍微复杂一点的 LP ，我们如何写出其对偶问题。

> 实际上在一些顶刊中，例如transportation Science等，比较近期的文章，也时不时会看到这样的操作。这个操作其实并不是抬手就能搞定的，很多时候需要反复修改，才能将对偶问题正确的写出来。 据我所知，我似乎是第一个写这样博文的博主。(如果有比我更早的，请告知我掐了这段)

先来看一个比较容易的线性规划问题：

$$\begin{aligned} \max \quad Z = & 2x_1 + 3x_2 \\ & 5x_1 + 4x_2 \leqslant 170 \quad \rightarrow \quad y_1 \\ & 2x_1 + 3x_2 \leqslant 100 \quad \rightarrow \quad y_2 \\ & x_1, x_1 \geqslant 0 \end{aligned}$$

其对偶问题比较容易写出：

$$\begin{aligned} \min \quad W = & 170y_1 + 100y_2 \\ & 5y_1 + 2y_2 \geqslant 2 \\ & 4y_1 + 3y_2 \geqslant 3 \\ & y_1, y_1 \geqslant 0 \end{aligned}$$

基本原则如图：

原问题(对偶问题)	对偶问题(原问题)
目标函数max	目标函数min
变量{ n个 / ≥0 / ≤0 / 无约束	n个 / ≥ / ≤ / = }约束条件
目标函数中变量的系数	约束条件右端项
约束条件{ m个 / ≤ / ≥ / =	m个 / ≥0 / ≤0 / 无约束 }变量
约束条件右端项	目标函数中变量的系数

但是假如是最短路问题：

最短路问题

$$
\begin{aligned}
\max \quad & \sum_{e \in A} d_e x_e \\
& \sum_{e \in \text{out}(i)} x_e - \sum_{e \in \text{in}(i)} x_e = \begin{cases} 1, & \text{if } i = s \\ -1, & \text{if } i = t \\ 0, & \text{else} \end{cases} \\
& 0 \leqslant x_e \leqslant 1, \qquad \forall e \in A
\end{aligned}
$$

这里大括号里有几个条件判断，就不是那么容易了。也许这个还比较容易，那再看看这个。多商品流问题 Multicommodity Network Flow Problem

多商品流问题Multicommodity Network Flow Problem

- K origin-destination pairs of nodes, $(s_1, t_1, d_1), (s_2, t_2, d_2), \cdots, (s_k, t_k, d_k)$.
- d_k : demand, amount of flow that must be sent from s_k to t_k.
- u_{ij} : capacity on (i, j) shared by all commodities
- c_{ij}^k : cost of sending 1 unit of commodity k in (i, j)
- x_{ij}^k : flow of commodity k in (i, j)

模型如下

$$
\begin{aligned}
\min \quad & \sum_{(i,j) \in A} \sum_{k} c_{ij}^k x_{ij}^k \\
& \sum_{j} x_{ij}^k - \sum_{j} x_{ji}^k = \begin{cases} d_k, & \text{if } \ i = s_k \\ -d_k, & \text{if } \ i = t_k \\ 0, & \text{otherwise} \end{cases} \\
& \sum_{k} x_{ij}^k \leqslant u_{ij}, \qquad \forall (i, j) \in A \\
& x_{ij}^k \geqslant 0, \qquad \forall (i, j) \in A, k \in K
\end{aligned}
$$

现在呢？还能很容易的写出来吗？若果还能，大神请受小弟一拜！哈哈哈

能轻松写出来的非人类大神可以前走左拐去刷剧了，咱这些普通人就接着往下看把。

> 注意，上面的Multicommodity Flow Problem和Shortest Path Problem都是Linear Programming,可以对偶的。但是对于Integer Programming和Mixed Integer Programming来讲，是不能对偶的。这一点一定要搞清楚。

对于这种稍微复杂一些的 LP ，我们怎么能写出对偶还保证正确，可debug找错的呢？我的方法就是借助 Excel + 具体小算例 。

借助Excel和具体小算例写出大规模LP的对偶

为了大家理解方便，我们不要直接去硬钢 Multicommodity Network Flow (理论上搞定了 Multicommodity Network Flow ，其实就具备搞定大多数可以对偶的LP的潜力了).我们先以 SPP 来开个胃。

Dual Problem :Shortest Path Problem(最短路问题)

小算例

我们先来引入一个小算例。该 算例 来自参考文献[^2],我做了点修改。为了显示我比较认真，我还专门无聊用 LaTeX + Tikz 重新画了一个小花图： (咋样，看着还舒服吧)

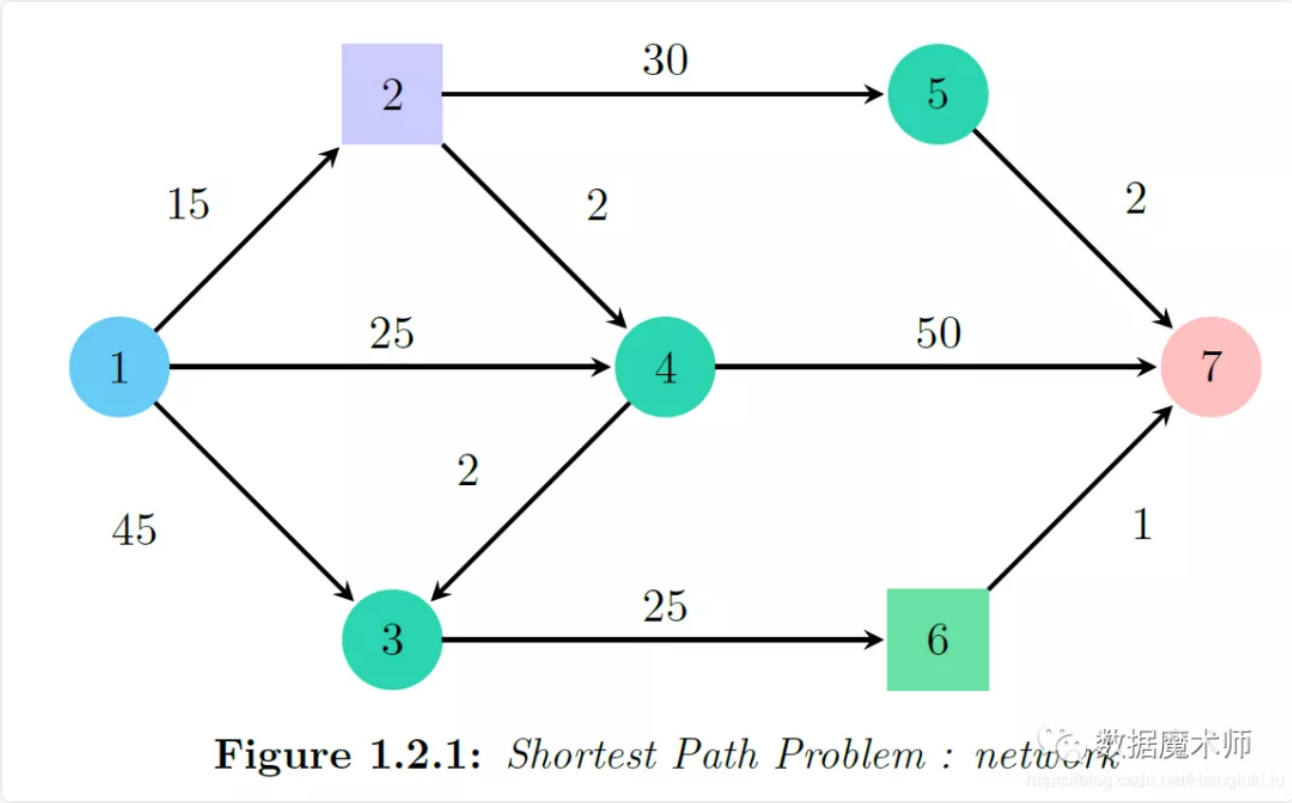


Figure 1.2.1: *Shortest Path Problem : network*

我们再来看一下 SPP 的模型：

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

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

$$0 \leqslant x_e \leqslant 1, \qquad \forall e \in A$$

按照这个模型，我们手动把这个模型具体的写出来。为了之后的操作，我们直接写到 Excel 里。

Excel+小算例写出SPP的对偶问题

SPP 模型如下：

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	distance	15	25	45	30	2	2	50	2	25	1			
2	min	x_1_2	x_1_4	x_1_3	x_2_5	x_2_4	x_5_7	x_4_7	x_4_3	x_3_6	x_6_7		RHS	Dual Var
3	org, 1	1	1	1								=	1	pi_1
4	inter, 2	-1			1	1						=	0	pi_2
5	inter, 3			-1					-1	1		=	0	pi_3
6	inter, 4		-1			-1		1	1			=	0	pi_4
7	inter, 5				-1		1					=	0	pi_5
8	inter, 6									-1	1	=	0	pi_6
9	des, 7						-1	-1			-1	=	-1	pi_7

我们把这个表叫 Primal tabular 其中，每一列代表一个变量$x_{ij}, \forall (i,j) \in A$

1. 第一行代表该条(i,j)的距离
2. 第二行代表变量$x_{ij}, \forall (i,j) \in A$
3. 第一行和第二行就组成了目标函数$\sum_{e \in A} d_e x_e$
4. 第3-9行代表每个结点$\forall i \in V$的约束
5. 最后一列代表每个约束的 Dual variable

OK，我们按照对偶的方法，将 Primal tabular 的 RHS 和 Dual variabe 拷贝，转置成2行，放在一个新表格(我们叫做 Dual tabular)的头两行，然后将 Primal tabular 的整个约束系数矩阵拷贝，转置到 Dual tabular 头两行下面。再把原问题 Primal tabular 的 distance 行和 min 行拷贝，转置，放在 Dual tabular 的右面。 再把 Dual tabular 中改成 max 。为了明确 Dual Problem 中 各个变量的符号（正负性） 以及 每个约束的符号(relation) ，我们在 Dual tabular 中加入一行（就是第三行）,表示变量的符号。同时在 Dual tabular 约束矩阵后加入一列，表示约束的符号。

操作完就是这样的

	A	B	C	D	E	F	G	H	I	J
1	RHS	1	0	0	0	0	0	-1		
2	Dual Var	pi_1	pi_2	pi_3	pi_4	pi_5	pi_6	pi_7		
3	max	=	=	=	=	=	=	=		
4	x_1_2	1	-1						<=	15
5	x_1_4	1			-1				<=	25
6	x_1_3	1		-1					<=	45
7	x_2_5		1			-1			<=	30
8	x_2_4		1		-1				<=	2
9	x_5_7					1		-1	<=	2
10	x_4_7				1			-1	<=	50
11	x_4_3			-1	1				<=	2
12	x_3_6			1			-1		<=	2
13	x_6_7						1	-1	<=	1

按照上面那个关系图中的信息，我们可以确定，对偶变量π_i都是无约束的，我们用 = 表示， Dual Problem 中的约束都是$\leqslant$的。这样，对偶就完成了。

但是，这还是一个具体的算例的 Dual ，我们需要将这个具体的算例，通过提取信息整理，化成一个general的公式形式。

将Excel中的Dual tabular转化成公式形式

我们观察上图，每一行都对应一条弧$(i,j) \in A$,例如第一行是$(1,2)$，第二行是$(1,4)$等。可以看到，对应出发点的变量系数全是1，对应终点的系数全是-1，无一例外，因此，我们可以断定，这个约束可以这么写：

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

结合目标函数，以及变量的符号，我们可以写出 SPP 的对偶问题：

$$\begin{aligned} \max \quad & \pi_s - \pi_t \\ & \pi_i - \pi_j \leqslant d_{ij}, \quad \forall (i,j) \in A \\ & \pi_i \quad \text{free} \end{aligned}$$

大功告成，怎么样， 有没有点内味了 ？

接下来，我们啃一个稍微难啃一些的骨头 Multicommodity Network Flow Problem .

Dual Problem :Multicommodity Network Flow Problem(最短路问题)

这个问题相比 SPP 难度还是大挺多的。我们首先上数学模型。

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} \sum_{k} c_{ij}^k x_{ij}^k \\ & \sum_j x_{ij}^k - \sum_j x_{ji}^k = \begin{cases} d_k, & \text{if } i = s_k \\ -d_k, & \text{if } i = t_k \\ 0, & \text{otherwise} \end{cases} \\ & \sum_k x_{ij}^k \leqslant u_{ij}, \qquad \forall (i,j) \in A \\ & x_{ij}^k \geqslant 0, \qquad \forall (i,j) \in A, k \in K \end{aligned}$$

看看吧，又有什么if $i = s_k$之类的，变量还是x_{ij}^k,你尝试自己先写一下，是不是觉得 脑瓜子嗡嗡的 ，哈哈哈。

具体算例

都不是事儿，咱一起来刚一下。同样的把文献[^2]中的算例原模原样搬过来看看（当然图还是我自己画的）

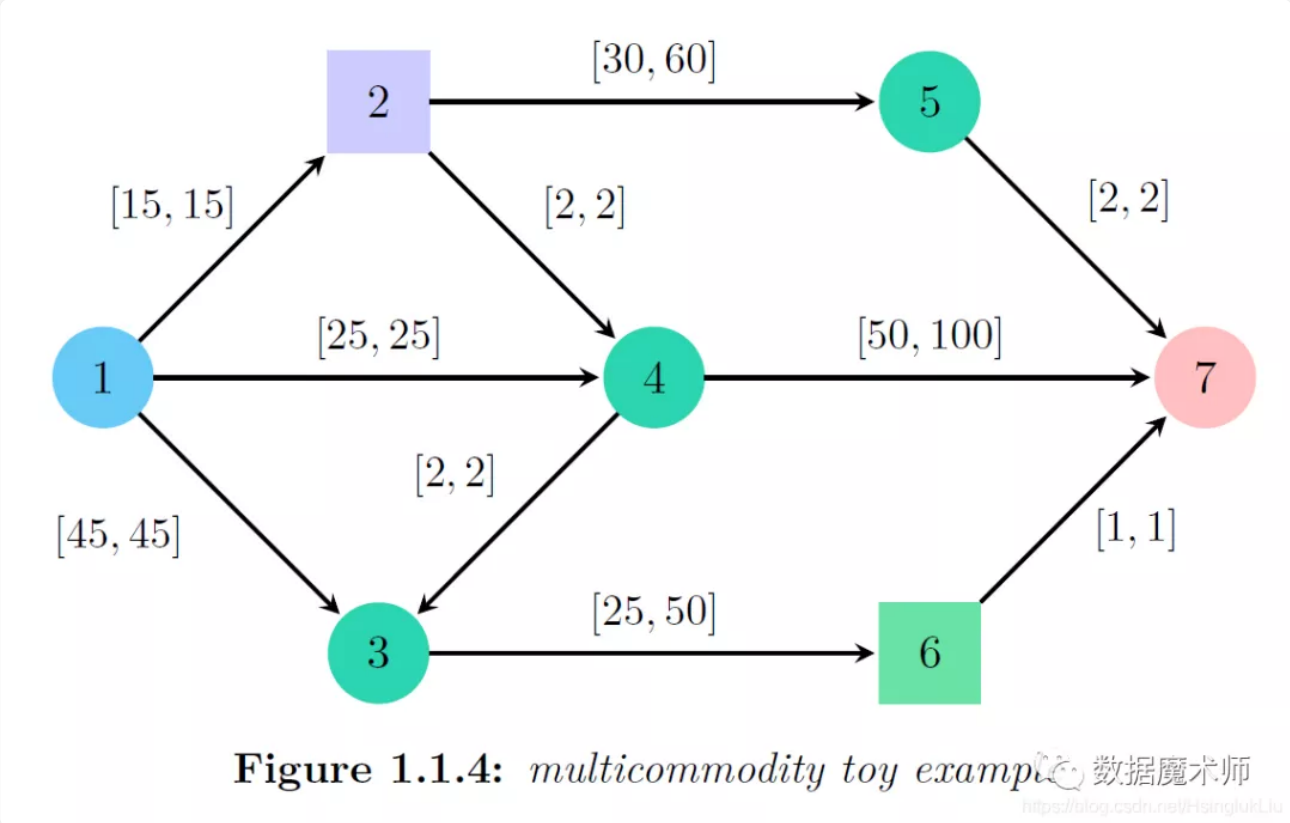


Figure 1.1.4: *multicommodity toy example*

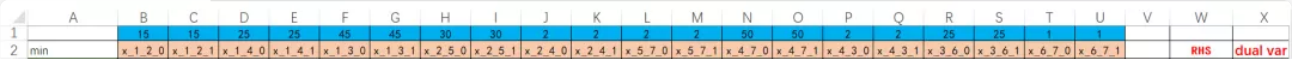
Excel+小算例写出MNF的原模型

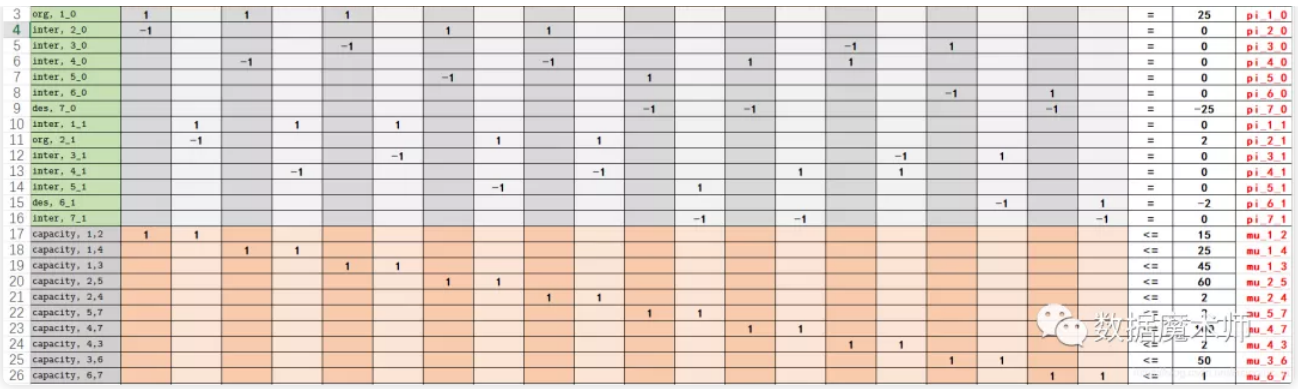
我们考虑有两个commodity:

```
commodity = [[1, 7, 25],  # s_i, d_i, demand
   [2, 6, 2]
   ]
```

> 本来想把代码也放上的，感觉太多了，有需要的话，大家私信我，我在修改把代码放上来。

然后我们按照模型和算例网络结构，把模型具体的写出来，如下图所示

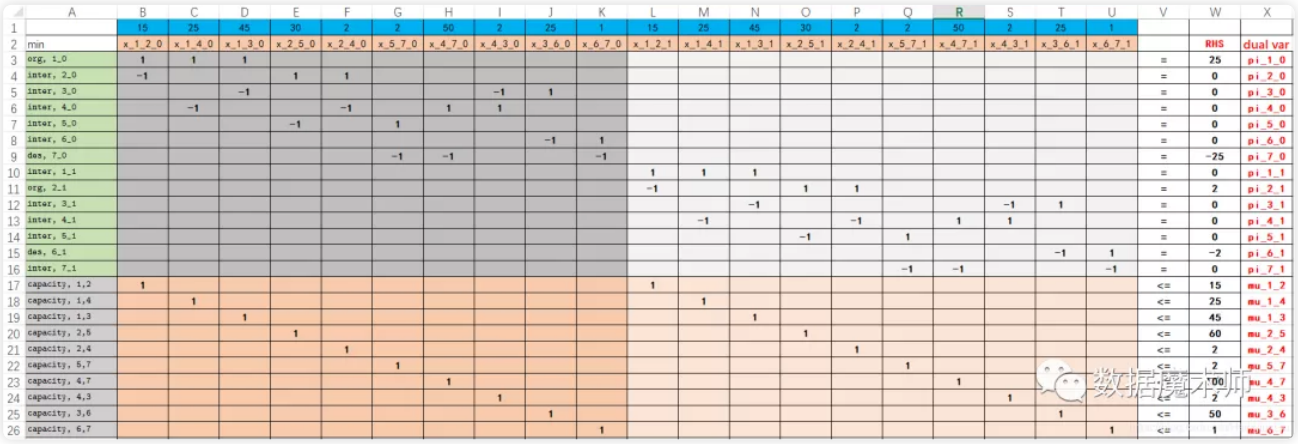




3	org, 1_0	1		1		1														=	25	pi_1_0
4	inter, 2_0	-1						1		1										=	0	pi_2_0
5	inter, 3_0					-1											-1		1	=	0	pi_3_0
6	inter, 4_0			-1						-1				1			1			=	0	pi_4_0
7	inter, 5_0							-1				1								=	0	pi_5_0
8	inter, 6_0																		-1	=	0	pi_6_0
9	des, 7_0											-1		-1					-1	=	-25	pi_7_0
10	inter, 1_1		1		1		1													=	0	pi_1_1
11	org, 2_1		-1						1		1									=	2	pi_2_1
12	inter, 3_1						-1										-1		1	=	0	pi_3_1
13	inter, 4_1				-1						-1				1		1			=	0	pi_4_1
14	inter, 5_1								-1				1							=	0	pi_5_1
15	des, 6_1																		-1	=	-2	pi_6_1
16	inter, 7_1												-1		-1					=	0	pi_7_1
17	capacity, 1,2	1	1																	<=	15	mu_1_2
18	capacity, 1,4			1	1															<=	25	mu_1_4
19	capacity, 1,3					1	1													<=	45	mu_1_3
20	capacity, 2,5							1	1											<=	60	mu_2_5
21	capacity, 2,4									1	1									<=	2	mu_2_4
22	capacity, 5,7											1	1							<=	2	mu_5_7
23	capacity, 4,7													1	1					<=	100	mu_4_7
24	capacity, 4,3															1	1			<=	2	mu_4_3
25	capacity, 3,6																	1	1	<=	50	mu_3_6
26	capacity, 6,7																		1	<=	1	mu_6_7

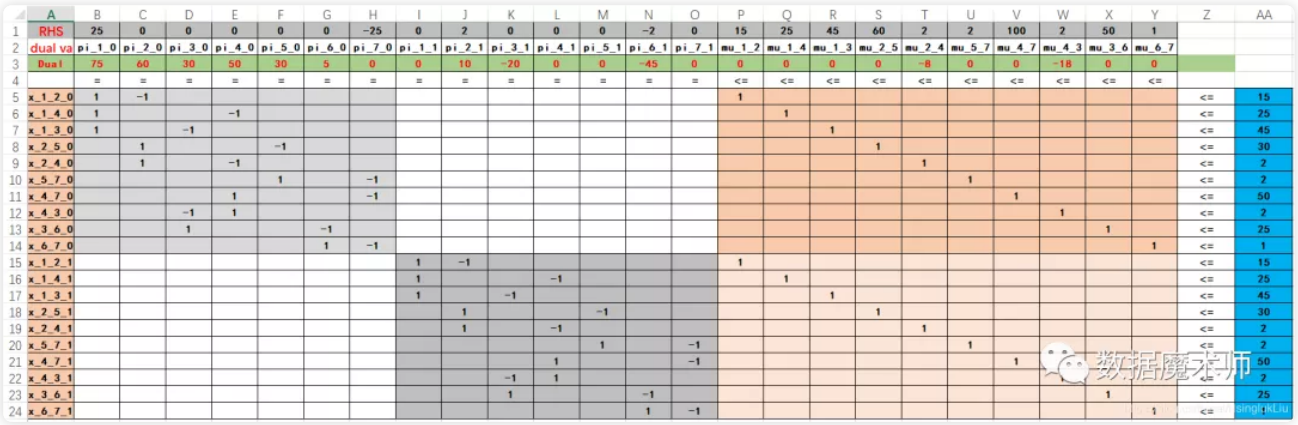
第二行的变量x_1_2_0就代表x_{ij}^k，其中0代表k。

这个看上去不太有规律，我们按照commodity k把上面的表格整一下，变成：



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1		15	25	45	30	2	2	50	2	25	1	15	25	45	30	2	2	50	2	25	1			
2	min	x_1_2_0	x_1_4_0	x_1_3_0	x_2_5_0	x_2_4_0	x_5_7_0	x_4_7_0	x_4_3_0	x_3_6_0	x_6_7_0	x_1_2_1	x_1_4_1	x_1_3_1	x_2_5_1	x_2_4_1	x_5_7_1	x_4_7_1	x_4_3_1	x_3_6_1	x_6_7_1		RHS	dual var
3	org, 1_0	1	1	1																		=	25	pi_1_0
4	inter, 2_0	-1			1	1																=	0	pi_2_0
5	inter, 3_0			-1					-1	1												=	0	pi_3_0
6	inter, 4_0		-1			-1		1	1													=	0	pi_4_0
7	inter, 5_0				-1		1															=	0	pi_5_0
8	inter, 6_0									-1	1											=	0	pi_6_0
9	des, 7_0						-1	-1			-1											=	-25	pi_7_0
10	inter, 1_1											1	1	1								=	0	pi_1_1
11	org, 2_1											-1			1	1						=	2	pi_2_1
12	inter, 3_1													-1					-1	1		=	0	pi_3_1
13	inter, 4_1												-1			-1		1	1			=	0	pi_4_1
14	inter, 5_1														-1		1					=	0	pi_5_1
15	des, 6_1																			-1	1	=	-2	pi_6_1
16	inter, 7_1																-1	-1			-1	=	0	pi_7_1
17	capacity, 1,2	1										1										<=	15	mu_1_2
18	capacity, 1,4		1										1									<=	25	mu_1_4
19	capacity, 1,3			1										1								<=	45	mu_1_3
20	capacity, 2,5				1										1							<=	60	mu_2_5
21	capacity, 2,4					1										1						<=	2	mu_2_4
22	capacity, 5,7						1										1					<=	2	mu_5_7
23	capacity, 4,7							1										1				<=	100	mu_4_7
24	capacity, 4,3								1										1			<=	2	mu_4_3
25	capacity, 3,6									1										1		<=	50	mu_3_6
26	capacity, 6,7										1										1	<=	1	mu_6_7

现在看上去就比较清楚了。我们仍然把这个表格叫做 `Primal Tabular` .接下来我们按照同样的方法，根据 `Primal Tabular` 生成 `Dual Tabular` ,如下图



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	RHS	25	0	0	0	0	0	-25	0	2	0	0	0	-2	0	15	25	45	60	2	2	100	2	50	1		
2	dual va	pi_1_0	pi_2_0	pi_3_0	pi_4_0	pi_5_0	pi_6_0	pi_7_0	pi_1_1	pi_2_1	pi_3_1	pi_4_1	pi_5_1	pi_6_1	pi_7_1	mu_1_2	mu_1_4	mu_1_3	mu_2_5	mu_2_4	mu_5_7	mu_4_7	mu_4_3	mu_3_6	mu_6_7			
3	Dual	75	60	30	50	30	5	0	0	10	-20	0	0	-45	0	0	0	0	0	-8	0	0	-18	0	0		
4		=	=	=	=	=	=	=	=	=	=	=	=	=	=	<=	<=	<=	<=	<=	<=	<=	<=	<=	<=		
5	x_1_2_0	1	-1													1										<=	15
6	x_1_4_0	1			-1												1									<=	25
7	x_1_3_0	1		-1														1								<=	45
8	x_2_5_0		1			-1													1							<=	30
9	x_2_4_0		1		-1															1						<=	2
10	x_5_7_0					1		-1													1					<=	2
11	x_4_7_0				1			-1														1				<=	50
12	x_4_3_0			-1	1																		1			<=	2
13	x_3_6_0			1			-1																	1		<=	25
14	x_6_7_0						1	-1																	1	<=	1
15	x_1_2_1								1	-1						1										<=	15
16	x_1_4_1								1			-1					1									<=	25
17	x_1_3_1								1		-1							1								<=	45
18	x_2_5_1									1			-1						1							<=	30
19	x_2_4_1									1		-1								1						<=	2
20	x_5_7_1												1		-1						1					<=	2
21	x_4_7_1											1			-1							1				<=	50
22	x_4_3_1										-1	1														<=	2
23	x_3_6_1										1			-1										1		<=	25
24	x_6_7_1													1	-1										1	<=	1

将Excel中的Dual tabular转化成公式形式

> 为了区分u和μ，表格中的mu我就用λ代替了，因为表格中写mu省地儿。 所以大家注意λ_{ij}就是上面表格中的mu

$$
\begin{aligned}
\max & \sum_{k \in K} d_k\left(\pi_{i=s_k}^k-\pi_{i=t_k}^k\right)+\sum_{(i,j) \in A} u_{ij} \lambda_{ij} \\
& \pi_i^k-\pi_j^k+\lambda_{ij} \leqslant c_{ij}^k, \quad \forall k \in K, \forall(i,j) \in A \\
& \pi_i^k \quad \text{free}, \quad \forall k \in K, \forall i \in V \\
& \lambda_{ij} \leqslant 0, \quad \forall(i,j) \in A
\end{aligned}
$$

当然了，按照国际惯例（搞OR大佬的惯例），我们还是跟之前我写的讲 SPP 对偶的博文 https://blog.csdn.net/HsinglukLiu/article/details/107834197中的操作一样：

1. 将所有对偶变量π_i^k取相反数
2. 把原约束中$\pi_i^k-\pi_j^k$改成$\pi_j^k-\pi_i^k$
3. 将$\pi_{i=s_k}^k$设置成0，也就是$\pi_{i=s_k}^k=0$

这三个隐含小动作，大佬是不会在论文里面写的，要是没仔细钻研，你一般会一头雾水。

OK，按照国际惯例操作完后，最终 Multicommodity Network Flow Problem 模型的 Dual Problem 就变成了下面的样子

$$
\begin{aligned}
\max & \sum_{k \in K} d_k \pi_{i=t_k}^k+\sum_{(i,j) \in A} u_{ij} \lambda_{ij} \\
& \pi_j^k-\pi_i^k+\lambda_{ij} \leqslant c_{ij}^k, \quad \forall k \in K, \forall(i,j) \in A \\
& \pi_i^k \quad \text{free}, \pi_{s_k}^k=0, \quad \forall k \in K, \forall i \in V \\
& \lambda_{ij} \leqslant 0, \quad \forall(i,j) \in A
\end{aligned}
$$

OK,所有的动作都完成了。一块硬骨头啃完了。

Python调用Gurobi求解Multicommodity Network Flow Problem （仅原问题）

最后再附上求解这个问题的Python代码（对偶问题的不想写了）

```
from gurobipy import *
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
import math

Arcs = {'1,2': [15, 15]    # cost flow
        ,'1,4': [25, 25]
        ,'1,3': [45, 45]
        ,'2,5': [30, 60]
        ,'2,4': [2, 2]
```

```
        ,'5,7': [2, 2]
        ,'4,7': [50, 100]
        ,'4,3': [2, 2]
        ,'3,6': [25, 50]
        ,'6,7': [1, 1]
       }
Arcs

Nodes = [1, 2, 3, 4, 5, 6, 7]

commodity = [[1, 7, 25],  # s_i, d_i, demand
             [2, 6, 2]
]


model = Model('MultiCommodity')

# add variables
X = {}
for key in Arcs.keys():
    for k in range(len(commodity)):
        key_x = key + ',' + str(k)
        X[key_x] = model.addVar(lb=0
                                ,ub=Arcs[key][1]
                                ,vtype=GRB.CONTINUOUS
                                ,name= 'x_' + key_x
                               )
# add objective function
obj = LinExpr(0)
for key in Arcs.keys():
    for k in range(len(commodity)):
        key_x = key + ',' + str(k)
        obj.addTerms(Arcs[key][0], X[key_x])
model.setObjective(obj, GRB.MINIMIZE)

# constraints 1
for k in range(len(commodity)):
    for i in Nodes:
        lhs = LinExpr(0)
        for key_x in X.keys():
#             nodes = key_x.split(',')
            if(i == (int)(key_x.split(',')[0]) and k == (int)(key_x.split(',')[2])):
                lhs.addTerms(1, X[key_x])
```

```
            if(i == (int)(key_x.split(',')[1]) and k == (int)(key_x.split(',')[2])):
                lhs.addTerms(-1, X[key_x])
        if(i == commodity[k][0]):
            model.addConstr(lhs == commodity[k][2], name='org_, ' + str(i) + '_' + str(k))
        elif(i == commodity[k][1]):
            model.addConstr(lhs == -commodity[k][2], name='des_, ' + str(i) + '_' + str(k))
        else:
            model.addConstr(lhs == 0, name='inter_, ' + str(i) + '_' + str(k))


# constraints 2
for key in Arcs.keys():
    lhs = LinExpr(0)
    for k in range(len(commodity)):
        key_x = key + ',' + str(k)
        lhs.addTerms(1, X[key_x])
    model.addConstr(lhs <= Arcs[key][1], name = 'capacity_, ' + key)

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

for var in model.getVars():
    if(var.x > 0):
        print(var.varName, '\t', var.x)
dual = model.getAttr("Pi", model.getConstrs())
```

原问题求解结果如下：

```
Solved in 0 iterations and 0.01 seconds
Optimal objective  1.873000000e+03
x_1,2,0   2.0
x_1,4,0   22.0
x_1,3,0   1.0
x_2,5,0   2.0
x_2,4,1   2.0
x_5,7,0   2.0
x_4,7,0   22.0
x_4,3,1   2.0
x_3,6,0   1.0
x_3,6,1   2.0
x_6,7,0   1.0
```

对偶问题求解

后记

硕士的时候搞这个搞了几天，还请教了我师兄挺多。师兄的研究 Robust Service Network Design 的文章里也用到了类似这样问题的对偶，发了 Transportation Science ，我把文章也贴在这里，欢迎大家去读一读，做的非常好[^3]。可以看到，这样的技巧在科研中还是有用武之地的。

[1] :Garg, N., & Koenemann, J. (2007). Faster and simpler algorithms for multicommodity flow and other fractional packing problems. SIAM Journal on Computing, 37(2), 630-652https://doi.org/10.1137/S0097539704446232

[2]:Cappanera, P., & Scaparra, M. P. (2011). Optimal allocation of protective resources in shortest-path networks. Transportation Science, 45(1), 64-80.http://dx.doi.org/10.1287/trsc.1100.0340

[3]:Wang, Z., & Qi, M. (2020). Robust service network design under demand uncertainty. Transportation Science, 54(3), 676-689.https://doi.org/10.1287/trsc.2019.0935

- END -

文案&编辑：刘兴禄（清华大学清华伯克利深圳学院2018级博士生）

审稿人：周航 （华中科技大学管理学院本科一年级）

如对文中内容有疑问，欢迎交流。PS：部分资料来自网络。

如有需求，可以联系：

秦虎老师（华中科技大学管理学院：professor.qin@qq.com）

刘兴禄（清华大学清华伯克利深圳学院2018级博士生：hsingluk.L@gmail.com，xlliu2015@163.com）

周航 （华中科技大学管理学院本科一年级：zh20010728@126.com）