凸优化大作业

两种目标函数下多商品流问题重述及样例求解

摘要

本文针对作业要求，对于流量工程、网络流问题、多商品流问题，给出了问题重述与形式化描述，介绍了 MAU 与 M/M/1 两种目标函数的作用目标及将两种非光滑非线性目标函数线性化的方式，最终给出一个多商品流问题的具体示例，使用 C++调用 Cplex 求解器对问题进行求解，两个目标函数的最优值分别为 0.8 与 92.6667，并使用 Python 调用 networkx 库绘制了流量图。

。

关键词 流量工程，多商品流问题，凸优化，非光滑优化，线性规划，Cplex.

1 问题重述

1.1 流量工程

流量工程，指的是针对网络中数据流的行为进行分析预测与管理，以实现针对网络性能的优化的一系列方法，而由于网络种类的复杂性，网络本身的复杂性，使得不同种的网络存在不同的优化目标与建模方法。[1]由题目界定，本文仅讨论自治域内的流量工程，假定进入和外出结点已知，则流量工程仅依赖域目标函数与路由机制，在确定以上信息后，为了实现流量工程的优化需要求解一个最优化问题，此问题可以表述为一个多商品流问题。[2]

1.2 网络流问题

多商品流问题是一个网络流问题。网络流问题是一个在有向图中分配流量，使得每条边的流量不超过容量约束，在实现模型所规定的流量移动目标的同时，目标函数达到最优的问题。[2]

根据题目[2]的给定符号标记，将网络记作$G = (N, E)$,其中N表示图的节点集，E表示图的弧集，对于每条弧$l \in E$,其容量记为c_l。

1.3 多商品流问题

1.3.1 多商品流问题的概念与约束

多商品流问题则是网络流问题的基础上，加入了具有不同源和宿的商品流，要求在满足商品流给定流量的条件下，把每个商品流看作一个商品，获得一个流量分配方案使得整个网络的拥塞或者负载最小化。[2]

根据题目[2]的给定符号标记，把一个源-目的对记作(s_m, t_m, d_m)表示该商品流流量在节点s_m流入网络，在节点t_m流出网络，流量需求为d_m，再令在弧l上为商品流m存在的流量为f_{ml}，结合网络流问题的符号规定与规则限制，可以给出多商品流问题的约束条件：

$$f_l = \sum_{\mathrm{m=1}}^{\mathrm{M}} \mathrm{f_{ml}} \le c_l, \forall l \in E \qquad (1a)$$

$$\sum_{l:l=(i,j)\in E} f_{ml} - \sum_{l:l=(j,i)\in E} f_{ml} = b_{mi}, m = 1, \dots, M, \forall i \in N \qquad (1b)$$

$$f_{ml} \geq 0, m = 1, \dots, M, \forall l \in E \qquad (1c)$$

其中，(1a)、(1c)两个约束来自网络流问题，要求对于每条边上流量不能超过网络边的容纳极限以及非负，(1b)的约束来自多商品流问题，要求每个商品流中的源-目的对的总流量符合各个商品流的流量需求。

1.3.2 多商品流问题的目标函数

1.3.2.1 抽象的目标函数

在给定商品流及其流量时，多商品流问题的变量仅存f_{ml}，因此记目标函数为$\phi(f)$。当具体到流量工程问题时，结合极小化网络拥塞的目标，常考虑$\phi(f)$为f的非减函数，并求其最小值来获得各边上的最优流量分布，结合以上的符号规定，目标函数可以写作如下形式：

$$minimize \quad \phi(f)$$

而在具体的优化计算中，最大弧利用率(Maximum Arc Utilization, MAU) 和与M/M/1延迟公式是使用较多的两个目标函数，下面对两个函数进行简要介绍。

1.3.2.1 最大弧利用率目标函数

MAU 表示的是在网络中所有边上使用流量占边的容量的比例最大的比例值，基于以上的符号规定，可以表示为如下形式：

$$\phi(f) = max_{l\in E} \quad f_l/c_l$$

通过极小化 MAU 能够使得流量从通道占用率最高的链路转移向占用率相对较低的链路，并最终获得各链路通道资源使用率相对一致的结果。但是在实际的求解中，由于该函数是分段光滑函数，在可数点上不可导，不能直接使用线性规划对其求解，由其目标函数为最小化一个最大化的分段线性函数，考虑使用课程中学习的表述将其线性化，具体来说引入变量$z = max_{l\in E} \quad f_l/c_l$，将等式约束分为两个不等式约束，即：

$$z \leq max_{l\in E} \quad f_l/c_l$$
$$z \geq max_{l\in E} \quad f_l/c_l$$

其中，$z \geq max_{l\in E} \quad f_l/c_l$的约束与$\frac{f_l}{c_l} - z \leq 0 \; \forall l \in E$等价，则等式约束可以被写作：

$$z \leq max_{l\in E} \quad f_l/c_l$$
$$\frac{f_l}{c_l} - z \leq 0 \; \forall l \in E$$

此时，若我们规定目标函数为$minimize \quad z$，并去掉$z \leq max_{l\in E} \quad f_l/c_l$的约束，则最终求得的$z$是在满足$z \geq max_{l\in E} \quad f_l/c_l$的约束条件下，能够取到的最小的$z$，则$z = max_{l\in E} \quad f_l/c_l$，满足引入变量时的规定。

综上，通过表述上的修饰可以将 MAU 公式表示为多个线性函数的逐点极大值以应用线性规划对其求解，具体形式如下：

$$minimize \quad z$$

$$\sum_{m=1}^{M} \frac{1}{c_l} f_{ml} - z \leq 0, \forall l \in E$$

1.3.2.2 M/M/1 延迟公式

M/M/1 延迟公式提出的思想是最小化边运输能力的占用率，假设当某条边的

占用率接近 100%，那么就需要控制此时在此边上增加流量带来的损失变得很大，[3]由这样的思想建立了以下的目标函数：

$$\phi(f) = \sum_{l \in E} \phi(f_l)$$

其中

$$\phi(f_l) = \begin{cases} f_l, & \frac{f_l}{c_l} \le \frac{1}{3} \\ 3f_l - \frac{2}{3}c_l & \frac{1}{3} < \frac{f_l}{c_l} \le \frac{2}{3} \\ 10f_l - \frac{16}{3}c_l & \frac{2}{3} < \frac{f_l}{c_l} \le \frac{9}{10} \\ 70f_l - \frac{178}{3}c_l & \frac{9}{10} < \frac{f_l}{c_l} \le 1 \\ 500f_l - \frac{1468}{3}c_l & 1 < \frac{f_l}{c_l} \le \frac{11}{10} \\ 5000f_l - \frac{16318}{3}c_l & \frac{11}{10} < \frac{f_l}{c_l} < \infty \end{cases}$$

与 MAU 目标函数相类似，该目标函数同样是分段光滑的，存在可数个不可导的点，可以通过类似 MAU 函数的变化形式将其表述作线性规划，其中对于每个$\phi(f_l)\ l \in E$，设存在变量$z_l = \phi(f_l)$,将其写作不等式约束如下：

$$z_l \le \phi(f_l)$$
$$z_l \ge \phi(f_l)$$

由$\phi(f_l)$函数一阶导数单调递增与函数连续的特点，能够将$z_l \ge \phi(f_l)$写作如下形式：

$$\begin{cases} \sum_m f_{ml} - z_l \le 0 \\ 3\sum_m f_{ml} - z_l \le \frac{2}{3}c_l \\ 10\sum_m f_{ml} - z_l \le \frac{16}{3}c_l \\ 70\sum_m f_{ml} - z_l \le \frac{178}{3}c_l \\ 500\sum_m f_{ml} - z_l \le \frac{1468}{3}c_l \\ 5000\sum_m f_{ml} - z_l \le \frac{16318}{3}c_l \end{cases}$$

类似 MAU 函数的处理规定目标函数为$minimize\ z_l$以代替约束$z_l \le \phi(f_l)$，问题目标函数被表述为线性函数，可以使用线性规划对其求解，其具体形式如下：

$$\text{minimize} \sum_{l \in E} z_l$$

$$\begin{cases} \sum_m f_{ml} - \mathrm{z_l} \le 0 & \forall\, l \in E \\ 3\sum_m f_{ml} - \mathrm{z_l} \le \frac{2}{3} c_l & \forall\, l \in E \\ 10\sum_m f_{ml} - \mathrm{z_l} \le \frac{16}{3} c_l & \forall\, l \in E \\ 70\sum_m f_{ml} - \mathrm{z_l} \le \frac{178}{3} c_l & \forall\, l \in E \\ 500\sum_m f_{ml} - \mathrm{z_l} \le \frac{1468}{3} c_l & \forall\, l \in E \\ 5000\sum_m f_{ml} - \mathrm{z_l} \le \frac{16318}{3} c_l & \forall\, l \in E \end{cases}$$

2. 问题样例介绍及求解

结合以上的模型建构与题目给出的样例，能够形式化的写出一个多商品流问题的目标函数与约束条件，进一步通过 C++建模，利用 Cplex 求解器进行问题的求解。

2.1 问题样例介绍

题目所给样例共有 7 个节点，13 条可行弧，4 个需求量为 4 的商品流，其中每条弧容量为 5 个单位。为了之后叙述方便，我们把题目中网络 G 用点弧关联矩阵 A 表示，其中 A[i][j]如果为 1 表示第 i 个结点在第 j 条弧上是出发点，A[i][j]如果为-1 表示第 i 个结点在第 j 条弧上是到达点，A[i][j]如果为 0 表示第 i 个结点不在第 j 条弧上。

$$\mathrm{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 1 & 0 & 0 & -1 & \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 \end{bmatrix}$$

问题中 4 个需求量为 4 的商品流用 B 矩阵表示，其中 B[i][j]表示第 i 个结点在第 j 个需求中是否被需求流出或流入，如果为正则流出相应的流量，如果为负则流入相应的流量，如果为零则不需要流入或流出。

$$\mathrm{B} = \begin{bmatrix} 4 & 4 & 0 & 4 \\ -4 & 0 & -4 & 0 \\ 0 & -4 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 \end{bmatrix}$$

2.2 问题建模与求解

依照以上的符号规定，结合网络流问题与多商品流问题的讨论，分别针对MAU 目标函数与 M/M/1 目标函数就能够建立起如下的问题模型，并针对其使用程序求解。

2.2.1 MAU 目标函数的建模与求解

参照以上符号规定，目标函数为 MAU 的模型如下：

$$minimize \quad z$$

$$\sum_{m=1}^{M} \frac{1}{c_l} f_{ml} - z \leq 0, \forall l \in E$$

$$f_l = \sum_{m=1}^{M} \mathrm{f_{ml}} \leq c_l, \forall l \in E$$

$$\sum_{l:l=(i,j)\in E} f_{ml} - \sum_{l:l=(j,i)\in E} f_{ml} = b_{mi}, m = 1, \dots, M, \forall i \in N$$

$$f_{ml} \geq 0, m = 1, \dots, M, \forall l \in E$$

调用 Cplex 完成求解后输出如图 1，其中 z 的最优值为 0.8，各$\mathrm{f_{ml}}$的值通过以下的 F 矩阵表示，其中F[i][j]表示在第 i 条商品流中，第 j 条弧经过了多少流量，最终的 F 矩阵如下：

$$\mathrm{F} = \begin{bmatrix} 4&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&4&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&4&4&0&4&0&4&0&0 \\ 0&0&4&0&0&0&0&4&0&0&0&4&0 \end{bmatrix}$$

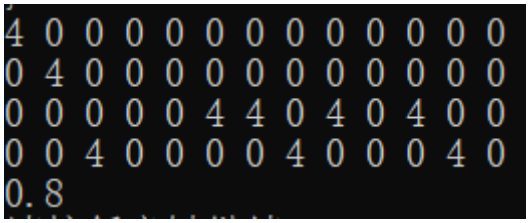

图 1　MAU 目标函数

求解结果表示为完成第 1 个商品流需求，从第 1 个点向第 2 个点运输了 4 个单位流量；为完成第 2 个商品流需求，从第 1 个点向第 3 个点运输了 4 个单位流量；为完成第 3 个商品流需求，从第 3 个点向第 6 个点运输了 4 个单位流量，从第 6 个点向第 4 个点运输了 4 个单位流量，从第 4 个点向第 5 个点运输了 4 个单位流量，从第 5 个点向第 2 个点运输了 4 个单位流量；为完成第 4 个商品流需求，从第 1 个点向第 4 个点运输了 4 个单位流量，从第 4 个点向第 6 个点运输了 4 个单位流量，从第 6 个点向第 7 个点运输了 4 个单位流量，其图形化表达如下图 2：

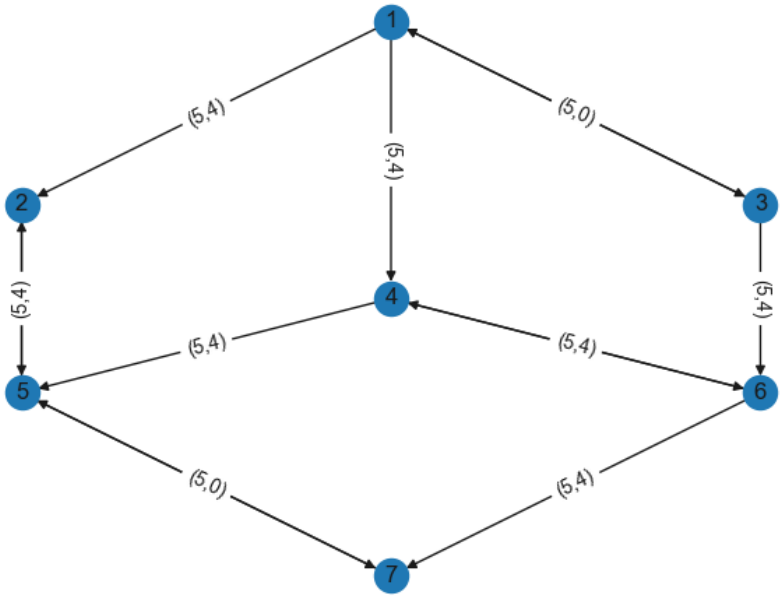

图 2 MAU 函数下最优路由

2.2.1 M/M/1 目标函数的建模与求解

参照以上符号规定，目标函数为 M/M/1 线性近似的模型如下：

$$\text{minimize} \quad \sum_{l \in E} z_l$$

$$\sum_m f_{ml} - z_l \le 0 \qquad \forall l \in E$$

$$3\sum_m f_{ml} - z_l \le \frac{2}{3} c_l \qquad \forall l \in E$$

$$10\sum_m f_{ml} - z_l \le \frac{16}{3} c_l \qquad \forall l \in E$$

$$70\sum_m f_{ml} - z_l \le \frac{178}{3} c_l \qquad \forall l \in E$$

$$500\sum_m f_{ml} - z_l \le \frac{1468}{3} c_l \qquad \forall l \in E$$

$$5000\sum_m f_{ml} - z_l \le \frac{16318}{3} c_l \qquad \forall l \in E$$

$$f_l = \sum_{m=1}^{M} f_{ml} \le c_l, \forall l \in E$$

$$\sum_{l:l=(i,j)\in E} f_{ml} - \sum_{l:l=(j,i)\in E} f_{ml} = b_{mi}, m = 1, \dots, M, \forall i \in N$$

$$f_{ml} \ge 0, m = 1, \dots, M, \forall l \in E$$

调用 Cplex 完成求解后输出如图 3，其$\phi(f) = \sum_{l\in E} z_l$的最优值为 92.6667，各$f_{ml}$的值通过以下的 F 矩阵表示，其中F[i][j]表示在第 i 条商品流中，第 j 条弧经

过了多少流量，最终的 F 矩阵如下：

$$F=\begin{bmatrix} 3.83 & 0 & 0.167 & 0 & 0 & 0 & 0.167 & 0 & 0.167 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0.67 & 0 & 0 & 0 & 0.67 & 3.33 & 1.67 & 0 & 3.33 & 0 & 1.67 & 1.67 & 1.67 \\ 0 & 0 & 4 & 0 & 0 & 0 & 2.33 & 1.67 & 0 & 2.33 & 0 & 1.67 & 0 \end{bmatrix}$$

各 z_l的值通过 Z 矩阵表示，其中 Z[i]表示 z_i的取值，最终的 Z 矩阵如下：

$$Z=[18.33 \quad 13.33 \quad 15 \quad 0 \quad 0.67 \quad 6.67 \quad 15 \quad 1.67 \quad 8.33 \quad 3.67 \quad 1.67 \quad 6.67 \quad 1.67]$$

```
3.83333 0 0.166667 0 0 0 0.166667 0 0.166667 0 0 0 0
0 4 0 0 0 0 0 0 0 0 0 0 0
0.666667 0 0 0 0.666667 3.33333 1.66667 0 3.33333 0 1.66667 1.66667 1.66667
0 0 4 0 0 0 2.33333 1.66667 0 2.33333 0 1.66667 0
18.3333 13.3333 15 0 0.666667 6.66667 15 1.66667 8.33333 3.66667 1.66667 6.66667 1.66667
92.6667请按任意键继续. . .
```

图 3 M/M/1 目标函数

求解结果表示为完成第 1 个商品流需求，从第 1 个点向第 2 个点运输了 3.83333 个单位流量，从第 1 个点向第 4 个点运输了 0.16667 个单位流量，从第 4 个点向第 5 个点运输了 0.16667 个单位流量，从第 5 个点向第 2 个点运输了 0.16667 个单位流量；为完成第 2 个商品流需求，从第 1 个点向第 3 个点运输了 4 个单位流量；为完成第 3 个商品流需求，从第 3 个点向第 1 个点运输了 0.66667 个单位流量，从第 1 个点向第 2 个点运输了 0.66667 个单位流量，从第 3 个点向第 6 个点运输了 3.33333 个单位流量，从第 6 个点向第 4 个点运输了 1.66667 个单位流量，从第 6 个点向第 7 个点运输了 1.66667 个单位流量，从第 4 个点向第 5 个点运输了 1.66667 个单位流量，从第 7 个点向第 5 个点运输了 1.66667 个单位流量，从第 5 个点向第 2 个点运输了 1.66667 个单位流量；为完成第 4 个商品流需求，从第 1 个点向第 4 个点运输了 4 个单位流量，从第 4 个点向第 5 个点运输了 2.33333 个单位流量，从第 4 个点向第 6 个点运输了 1.66667 个单位流量，从第 5 个点向第 7 个点运输了 1.66667 个单位流量，从第 6 个点向第 7 个点运输了 1.66667 个单位流量，其图形化表达如下图 4：

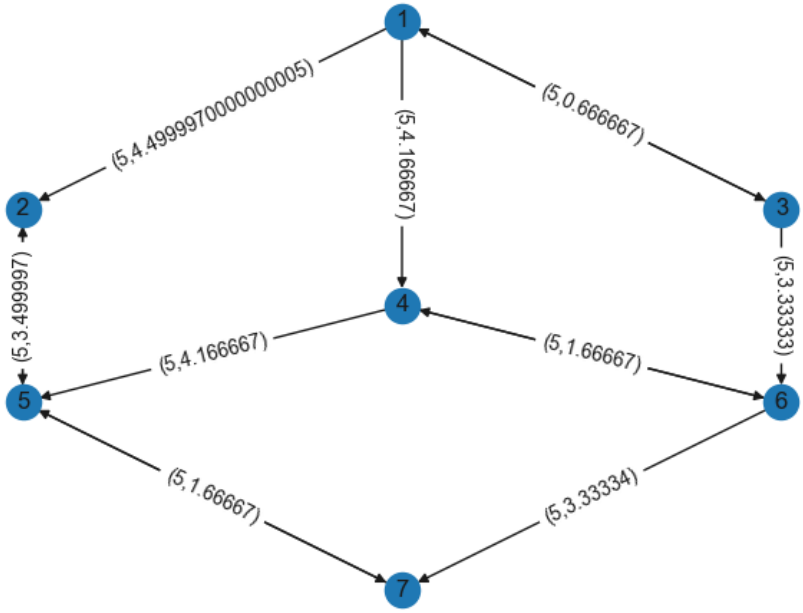

图 4 M/M/1 函数最优路由

2.3 结果分析

通过以上的两个目标函数下的模型建立与求解，能够发现在 MAU 函数下，边上的流量分布往往只为 0 或 4，对于流量资源的利用并不充分，这可能是仅关注最大的流量占用率边的优化所导致的结果；而在 M/M/1 将其他边的流量占用率纳入考虑之后各需求的实现发生了较大的变化，其中(1,2)边、(1,4)边、(4,5)边的占用率超过了 MAU 下的 0.8，同时所有的边都存在流量通过，对于流量资源的利用更有效的同时，部分边的流量压力也变得更大，通过这点说明弧与弧之间在完成给定需求的地位并不平等，部分弧显得更为“有用”，因而可以考虑将其余边的容量向(1,2)边、(1,4)边、(4,5)边转移。

在以上的分析下，尝试在使用 M/M/1 函数作为目标函数的同时，将各边的容量c_l作为变量加入模型中，并添加所有边的总容量为定值的约束如下：

$$\sum_{l\in E} c_l = 65$$

调用 Cplex 完成求解后输出如图 3，其$\phi(f) = \sum_{l\in E} z_l$的最优值为 40.6667，各$f_{ml}$的值通过以下的 F 矩阵表示，其中F[i][j]表示在第 i 条商品流中，第 j 条弧经过了多少流量，最终的 F 矩阵如下：

$$F = \begin{bmatrix} 4&0&0&0&0&0&0&0&0&0&0&0 \\ 0&4&0&0&0&0&0&0&0&0&0&0 \\ 4&0&0&0&4&0&0&0&0&0&0&0 \\ 4&0&0&4&0&0&0&0&4&0&0&0 \end{bmatrix}$$

各 z_l的值通过 Z 矩阵表示，其中 Z[i]表示 z_i的取值，最终的 Z 矩阵如下：

$$Z = [20.6667\ 4\ 0\ 4\ 4\ 0\ 0\ 0\ 8\ 0\ 0\ 0]$$

各 c_l的值通过 C 矩阵表示，其中 C[i]表示 z_i的取值，最终的 C 矩阵如下

$$C = [23\ 12\ 0\ 12\ 12\ 0\ 0\ 0\ 0\ 6\ 0\ 0\ 0]$$

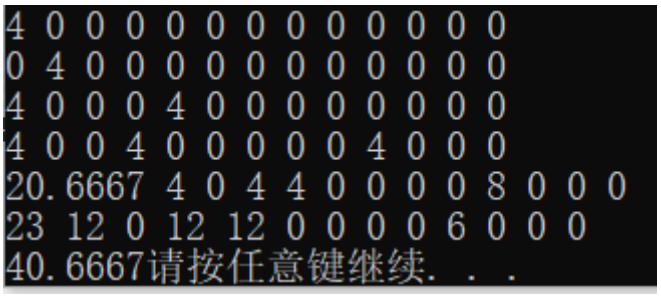

图 5 M/M/1 函数可变弧容量求解

由此可以看出，在以上提出的问题中，各弧上的容量还存在有一定的优化空间可以继续挖掘，但是同时，此问题最优解中部分弧上的容量为零显然也是不合理的，需要更多的历史数据挖掘需求规律才能进一步针对各弧的容量分配进行优化。

参考文献

[1]周桐庆,蔡志平,夏竟,徐明.基于软件定义网络的流量工程[J].软件学报,2016,27(02):394-417.
[2]刘红英.多商品流问题在流量工程上的应用.北京
[3] B. Fortz and M. Thorup. “Internet traffic engineering by optimizing OSPF weights”, Proc. IEEE INFOCOM, 2000, Tel Aviv, Israel, pp. 519 - 528.

附录

1 C++调用 Cplex 求解程序

```
#include<ilcplex/ilocplex.h>

ILOSTLBEGIN

void Mau(int A[][13], int B[][4]);
void MM1(int A[][13], int B[][4]);
void MM1_p(int A[][13], int B[][4]);
int main()
{

    int A[][13] = { {1,1,1,0,-1,0,0,0,0,0,0,0,0},
                    {-1,0,0,1,0,0,0,0,-1,0,0,0,0},
                    {0,-1,0,0,1,1,0,0,0,0,0,0,0},
                    {0,0,-1,0,0,0,1,1,0,0,-1,0,0},
                    {0,0,0,-1,0,0,-1,0,1,1,0,0,-1},
                    {0,0,0,0,0,-1,0,-1,0,0,1,1,0},
                    {0,0,0,0,0,0,0,0,0,-1,0,-1,1} };
    int B[][4] = {  {4,4,0,4},
                    {-4,0,-4,0},
                    {0,-4,4,0},
                    {0,0,0,0},
                    {0,0,0,0},
                    {0,0,0,0},
                    {0,0,0,-4} };
    Mau(A, B);
    MM1(A, B);
    MM1_p(A, B);
}

void Mau(int A[][13], int B[][4])
{
    IloEnv env;
    IloModel model(env);
    IloNumVarArray var(env);
    IloRangeArray con(env);

    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 13; j++)var.add(IloNumVar(env));
    }
```

```
    var.add(IloNumVar(env));
    IloObjective obj = IloMinimize(env, var[4 * 13]);
    IloExprArray temp;
    temp = IloExprArray(env, 13);
    for (int i = 0; i < 13; i++)
    {
        temp[i] = IloExpr(env);
        for (int j = 0; j < 4; j++)
        {
            temp[i] += var[13 * j + i] * 0.2;
        }
        temp[i] -= var[4 * 13];
        con.add(temp[i] <= 0);
    }
    IloExprArray temp1;
    temp1 = IloExprArray(env, 13);
    for (int i = 0; i < 13; i++)
    {
        temp1[i] = IloExpr(env);
        for (int j = 0; j < 4; j++)
        {
            temp1[i] += var[13 * j + i];
        }
        con.add(temp1[i] <= 5);
    }
    IloExprArray temp2;
    temp2 = IloExprArray(env, 4 * 7);
    for (int i = 0; i < 7; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            temp2[i * 4 + j] = IloExpr(env);
            for (int k = 0; k < 13; k++)
            {
                if (A[i][k] == 1)temp2[i * 4 + j] += var[j * 13 + k];
                else if (A[i][k] == -1)temp2[i * 4 + j] -
   = var[j * 13 + k];
            }
            con.add(temp2[i * 4 + j] == B[i][j]);
        }
    }

    model.add(obj);
    model.add(con);
```

```
84.
85.     IloCplex cplex(model);
86.     cplex.solve();
87.     env.out() << model;
88.     for (int i = 0; i < 4; i++)
89.     {
90.         for (int j = 0; j < 13; j++)
91.         {
92.             env.out() << cplex.getValue(var[i * 13 + j])<<" ";
93.         }
94.         env.out() << endl;
95.     }
96.     env.out() << cplex.getValue(var[4 * 13]) << endl;
97.     system("pause");
98. }
99.
100. void MM1(int A[][13], int B[][4])
101. {
102.     IloEnv env;
103.     IloModel model(env);
104.     IloNumVarArray var(env);
105.     IloRangeArray con(env);
106.
107.     for (int i = 0; i <= 4; i++)
108.     {
109.         for (int j = 0; j < 13; j++)var.add(IloNumVar(env));
110.     }
111.     IloExpr obj_temp(env);
112.     for (int j = 0; j < 13; j++)obj_temp += var[4 * 13 + j];
113.     IloObjective obj = IloMinimize(env, obj_temp);
114.     IloExprArray temp;
115.     temp = IloExprArray(env, 6*13);
116.     for (int i = 0; i < 13; i++)
117.     {
118.         temp[i] = IloExpr(env);
119.         for (int j = 0; j < 4; j++)
120.         {
121.             temp[i] += var[13 * j + i];
122.         }
123.         temp[i] -= var[13 * 4 + i];
124.         con.add(temp[i] <= 0);
125.     }
126.     for (int i = 0; i < 13; i++)
127.     {
```

```
128.         temp[13+i] = IloExpr(env);
129.         for (int j = 0; j < 4; j++)
130.         {
131.             temp[13+i] += 3*var[13 * j + i];
132.         }
133.         temp[13+i] -= var[13 * 4 + i];
134.         con.add(temp[13+i] <= 2.0/3.0*5);
135.     }
136.     for (int i = 0; i < 13; i++)
137.     {
138.         temp[2*13 + i] = IloExpr(env);
139.         for (int j = 0; j < 4; j++)
140.         {
141.             temp[2*13 + i] += 10 * var[13 * j + i];
142.         }
143.         temp[2*13 + i] -= var[13 * 4 + i];
144.         con.add(temp[2*13 + i] <= 16.0 / 3.0 * 5);
145.     }
146.     for (int i = 0; i < 13; i++)
147.     {
148.         temp[3 * 13 + i] = IloExpr(env);
149.         for (int j = 0; j < 4; j++)
150.         {
151.             temp[3 * 13 + i] += 70 * var[13 * j + i];
152.         }
153.         temp[3 * 13 + i] -= var[13 * 4 + i];
154.         con.add(temp[3 * 13 + i] <= 178.0 / 3.0 * 5);
155.     }
156.     for (int i = 0; i < 13; i++)
157.     {
158.         temp[4 * 13 + i] = IloExpr(env);
159.         for (int j = 0; j < 4; j++)
160.         {
161.             temp[4 * 13 + i] += 500 * var[13 * j + i];
162.         }
163.         temp[4 * 13 + i] -= var[13 * 4 + i];
164.         con.add(temp[4 * 13 + i] <= 1468.0 / 3.0 * 5);
165.     }
166.     for (int i = 0; i < 13; i++)
167.     {
168.         temp[5 * 13 + i] = IloExpr(env);
169.         for (int j = 0; j < 4; j++)
170.         {
171.             temp[5 * 13 + i] += 5000 * var[13 * j + i];
```

```
172.         }
173.         temp[5 * 13 + i] -= var[13 * 4 + i];
174.         con.add(temp[5 * 13 + i] <= 16318.0 / 3.0 * 5);
175.     }
176.     IloExprArray temp1;
177.     temp1 = IloExprArray(env, 13);
178.     for (int i = 0; i < 13; i++)
179.     {
180.         temp1[i] = IloExpr(env);
181.         for (int j = 0; j < 4; j++)
182.         {
183.             temp1[i] += var[13 * j + i];
184.         }
185.         con.add(temp1[i] <= 5);
186.     }
187.     IloExprArray temp2;
188.     temp2 = IloExprArray(env, 4 * 7);
189.     for (int i = 0; i < 7; i++)
190.     {
191.         for (int j = 0; j < 4; j++)
192.         {
193.             temp2[i * 4 + j] = IloExpr(env);
194.             for (int k = 0; k < 13; k++)
195.             {
196.                 if (A[i][k] == 1)temp2[i * 4 + j] += var[j * 13 + k];
197.                 else if (A[i][k] == -1)temp2[i * 4 + j] -
    = var[j * 13 + k];
198.             }
199.             con.add(temp2[i * 4 + j] == B[i][j]);
200.         }
201.     }
202.
203.     model.add(obj);
204.     model.add(con);
205.
206.     IloCplex cplex(model);
207.     cplex.solve();
208.     env.out() << model;
209.     for (int i = 0; i <= 4; i++)
210.     {
211.         for (int j = 0; j < 13; j++)
212.         {
213.             env.out() << cplex.getValue(var[i * 13 + j]) << " ";
214.         }
```

```
215.         env.out() << endl;
216.     }
217.     IloNum tempx = 0;
218.     env.out() << cplex.getObjValue();
219.     for (int j = 0; j < 13; j++) tempx += cplex.getValue(var[4 * 13 + j]);

220.     //env.out()<<tempx;
221.     system("pause");
222. }
223.
224.
225. void MM1_p(int A[][13], int B[][4])
226. {
227.     IloEnv env;
228.     IloModel model(env);
229.     IloNumVarArray var(env);
230.     IloRangeArray con(env);
231.
232.     for (int i = 0; i <= 5; i++)
233.     {
234.         for (int j = 0; j < 13; j++)var.add(IloNumVar(env));
235.     }
236.     IloExpr obj_temp(env);
237.     for (int j = 0; j < 13; j++)obj_temp += var[4 * 13 + j];
238.     IloObjective obj = IloMinimize(env, obj_temp);
239.     IloExprArray temp;
240.     temp = IloExprArray(env, 6 * 13);
241.     for (int i = 0; i < 13; i++)
242.     {
243.         temp[i] = IloExpr(env);
244.         for (int j = 0; j < 4; j++)
245.         {
246.             temp[i] += var[13 * j + i];
247.         }
248.         temp[i] -= var[13 * 4 + i];
249.         con.add(temp[i] <= 0);
250.     }
251.     for (int i = 0; i < 13; i++)
252.     {
253.         temp[13 + i] = IloExpr(env);
254.         for (int j = 0; j < 4; j++)
255.         {
256.             temp[13 + i] += 3 * var[13 * j + i];
257.         }
```

```
258.          temp[13 + i] -= var[13 * 4 + i];
259.          temp[13 + i] -= var[13 * 5 + i] * 2.0 / 3.0;
260.          con.add(temp[13 + i] <=  0);
261.      }
262.      for (int i = 0; i < 13; i++)
263.      {
264.          temp[2 * 13 + i] = IloExpr(env);
265.          for (int j = 0; j < 4; j++)
266.          {
267.              temp[2 * 13 + i] += 10 * var[13 * j + i];
268.          }
269.          temp[2 * 13 + i] -= var[13 * 4 + i];
270.          temp[2 * 13 + i] -= var[13 * 5 + i]*16.0/3.0;
271.          con.add(temp[2 * 13 + i] <= 0);
272.      }
273.      for (int i = 0; i < 13; i++)
274.      {
275.          temp[3 * 13 + i] = IloExpr(env);
276.          for (int j = 0; j < 4; j++)
277.          {
278.              temp[3 * 13 + i] += 70 * var[13 * j + i];
279.          }
280.          temp[3 * 13 + i] -= var[13 * 4 + i];
281.          temp[3 * 13 + i] -= var[13 * 5 + i] * 178.0 / 3.0;
282.          con.add(temp[3 * 13 + i] <= 0);
283.      }
284.      for (int i = 0; i < 13; i++)
285.      {
286.          temp[4 * 13 + i] = IloExpr(env);
287.          for (int j = 0; j < 4; j++)
288.          {
289.              temp[4 * 13 + i] += 500 * var[13 * j + i];
290.          }
291.          temp[4 * 13 + i] -= var[13 * 4 + i];
292.          temp[4 * 13 + i] -= var[13 * 5 + i] * 1468.0 / 3.0;
293.          con.add(temp[4 * 13 + i] <= 0);
294.      }
295.      for (int i = 0; i < 13; i++)
296.      {
297.          temp[5 * 13 + i] = IloExpr(env);
298.          for (int j = 0; j < 4; j++)
299.          {
300.              temp[5 * 13 + i] += 5000 * var[13 * j + i];
301.          }
```

```
302.         temp[5 * 13 + i] -= var[13 * 4 + i];
303.         temp[5 * 13 + i] -= var[13 * 5 + i] * 16318.0 / 3.0;
304.         con.add(temp[5 * 13 + i] <= 0);
305.     }
306.     IloExprArray temp1;
307.     temp1 = IloExprArray(env, 13);
308.     for (int i = 0; i < 13; i++)
309.     {
310.         temp1[i] = IloExpr(env);
311.         for (int j = 0; j < 4; j++)
312.         {
313.             temp1[i] += var[13 * j + i];
314.         }
315.         temp1[i] -= var[13 * 5 + i];
316.         con.add(temp1[i] <= 0);
317.     }
318.     IloExprArray temp2;
319.     temp2 = IloExprArray(env, 4 * 7);
320.     for (int i = 0; i < 7; i++)
321.     {
322.         for (int j = 0; j < 4; j++)
323.         {
324.             temp2[i * 4 + j] = IloExpr(env);
325.             for (int k = 0; k < 13; k++)
326.             {
327.                 if (A[i][k] == 1)temp2[i * 4 + j] += var[j * 13 + k];
328.                 else if (A[i][k] == -1)temp2[i * 4 + j] -
    = var[j * 13 + k];
329.             }
330.             con.add(temp2[i * 4 + j] == B[i][j]);
331.         }
332.     }
333.     IloExpr temp3;
334.     temp3 = IloExpr(env);
335.     for (int i = 0; i < 13; i++)temp3 += var[13 * 5 + i];
336.     con.add(temp3 == 13*5);
337.
338.     model.add(obj);
339.     model.add(con);
340.
341.     IloCplex cplex(model);
342.     cplex.solve();
343.     env.out() << model;
344.
```

```
    for (int i = 0; i <= 5; i++)
    {
        for (int j = 0; j < 13; j++)
        {
            env.out() << cplex.getValue(var[i * 13 + j]) << " ";
        }
        env.out() << endl;
    }
    IloNum tempx = 0;
    env.out() << cplex.getObjValue();
    for (int j = 0; j < 13; j++) tempx += cplex.getValue(var[4 * 13 + j]);

    //env.out()<<tempx;
    system("pause");
}
```

2 Python 绘图程序

```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

edge_labels = {}
A = [[1, 1, 1, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0],
     [-1, 0, 0, 1, 0, 0, 0, 0, -1, 0, 0, 0, 0],
     [0, -1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
     [0, 0, -1, 0, 0, 0, 1, 1, 0, 0, -1, 0, 0],
     [0, 0, 0, -1, 0, 0, -1, 0, 1, 1, 0, 0, -1],
     [0, 0, 0, 0, 0, -1, 0, -1, 0, 0, 1, 1, 0],
     [0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, -1, 1]]
F = [int(a) for a in
     "4 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 0 4 0 4 0 0 0 0 4 0 0 0 0 4 0 0 0 4 0".split(
         ' ')]
F2 = [float(a) for a in
      "3.83333 0 0.166667 0 0 0 0.166667 0 0.166667 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0.666667 0 0 0 0.666667 3.33333 1.66667 0 3.33333 0 1.66667 1.66667 1.66667 0 0 4 0 0 0 2.33333 1.66667 0 2.33333 0 1.66667 0".split(
          ' ')]

F = [[F[i + 13 * j] for i in range(13)] for j in range(4)]
F2 = [[F2[i + 13 * j] for i in range(13)] for j in range(4)]
```

```
A = [[A[i][j] for i in range(7)] for j in range(13)]
edge = [(e.index(1) + 1, e.index(-
1) + 1, {'capacity': 5, 'weight': 0}) for e in A]
for flo in F:
    for index, x in enumerate(flo):
        if x != 0:
            edge[index][2]['weight'] += x
G = nx.DiGraph()
G.add_nodes_from([i for i in range(1, 8)])
G.add_edges_from(edge)
pos = nx.spring_layout(G)
pos[1][0] = 0.5
pos[1][1] = 1
pos[2][0] = 0
pos[2][1] = 0.67
pos[3][0] = 1
pos[3][1] = 0.67
pos[4][0] = 0.5
pos[4][1] = 0.5
pos[5][0] = 0
pos[5][1] = 0.33
pos[6][0] = 1
pos[6][1] = 0.33
pos[7][0] = 0.5
pos[7][1] = 0
edge_label1 = nx.get_edge_attributes(G, 'capacity')
edge_label2 = nx.get_edge_attributes(G, 'weight')
for i in edge_label1.keys():
    edge_labels[i] = f'({edge_label1[i]:},{edge_label2[i]:})'
sns.set(style="darkgrid")
nx.draw(G, pos=pos, with_labels=True)
nx.draw_networkx_edge_labels(G, pos=pos, edge_labels=edge_labels)
plt.show()

edge = [(e.index(1) + 1, e.index(-
1) + 1, {'capacity': 5, 'weight': 0}) for e in A]
for flo in F2:
    for index, x in enumerate(flo):
        if x != 0:
            edge[index][2]['weight'] += x
G = nx.DiGraph()
G.add_nodes_from([i for i in range(1, 8)])
G.add_edges_from(edge)
pos = nx.spring_layout(G)
```

```
pos[1][0] = 0.5
pos[1][1] = 1
pos[2][0] = 0
pos[2][1] = 0.67
pos[3][0] = 1
pos[3][1] = 0.67
pos[4][0] = 0.5
pos[4][1] = 0.5
pos[5][0] = 0
pos[5][1] = 0.33
pos[6][0] = 1
pos[6][1] = 0.33
pos[7][0] = 0.5
pos[7][1] = 0
edge_label1 = nx.get_edge_attributes(G, 'capacity')
edge_label2 = nx.get_edge_attributes(G, 'weight')
for i in edge_label1.keys():
    edge_labels[i] = f'({edge_label1[i]:},{edge_label2[i]:})'
sns.set(style="darkgrid")
nx.draw(G, pos=pos, with_labels=True)
nx.draw_networkx_edge_labels(G, pos=pos, edge_labels=edge_labels)
plt.show()
```