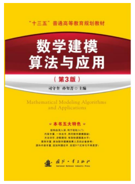
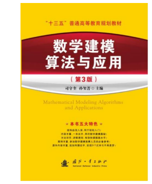


数学建模算法与应用

第4章

4.6网络最大流问题

4.7MATLAB的图论工具箱及其应用



目录 CONTENTS

01 最大流与最小费用流问题

4.6.1 最大流问题

许多系统包含了流量问题，如公路系统中有车辆流、物资调配系统中有物资流、金融系统中有现金流等。这些流问题都可归结为网络流问题，且都存在一个如何安排使流量最大的问题，即最大流问题。

1.基本概念

定义 4.15 给定一个有向图$D=(V,A)$，其中A为弧集，在V中指定了一点，称为发点或源（记为v_s），该点只有发出的弧；同时指定一个点称为收点或汇（记为v_t），该点只有进入的弧；其余的点叫中间点，对于每一条弧$(v_i,v_j)\in A$，对应有一个$c(v_i,v_j)\geq 0$（或简写为c_{ij}），称为弧的容量。通常把这样的有向图D叫作一个网络，记作$D=(V,A,C)$，其中$C=\{c_{ij}\}$。

所谓网络上的流，是指定义在弧集合A上的一个函数$f=\{f_{ij}\}=\{f(v_i,v_j)\}$，并称$f_{ij}$为弧$(v_i,v_j)$上的**流量**。

定义 4.16 满足下列条件的流 f 称为可行流。

（1）容量限制条件：对每一弧 $(v_i,v_j)\in A$，$0\le f_{ij}\le c_{ij}$

（2）平衡条件：

中间点，流出量＝流入量，即每个 $i(i\neq s,t)$ 有

$$\sum_{j:(v_i,v_j)\in A} f_{ij}-\sum_{j:(v_j,v_i)\in A} f_{ji}=0;$$

发点 v_s

$$\sum_{(v_s,v_j)\in A} f_{sj}=v;$$

收点 v_t

$$\sum_{(v_j,v_t)\in A} f_{jt}=v;$$

可行流总是存在

——零流

式中 v 称为这个可行流的流量，即发点的净输出量。

最大流问题可以写为如下的线性规划模型

$$
\max v,
$$

$$
\text{s.t.}\begin{cases}\displaystyle\sum_{(v_s,v_j)\in A} f_{sj}=v, \\ \displaystyle\sum_{j:(v_i,v_j)\in A} f_{ij}-\sum_{j:(v_j,v_k)\in A} f_{jk}=0, \quad j\neq s,t, \\ \displaystyle\sum_{(v_j,v_t)\in A} f_{jt}=v, \\ 0\le f_{ij}\le c_{ij}, \quad \forall (v_i,v_j)\in A. \end{cases} \tag{4.5}
$$

定义 4.17 设f是一个可行流，μ是从v_s到v_t的一条路，若μ满足：前向弧是非饱和弧，后向弧是非零流弧，则称μ为一条（关于可行流f）**增广路**。

若给定一个可行流$f=\{f_{ij}\}$，把网络中使$f_{ij}=c_{ij}$的弧称为**饱和弧**，使$f_{ij}<c_{ij}$的弧称为**非饱和弧**。把$f_{ij}=0$的弧称为**零流弧**，$f_{ij}>0$的弧称为**非零流弧**。

若μ是网络中联结发点v_s和收点v_t的一条路，定义路的方向是从v_s到v_t，则路上的弧被分为两类：

（1）弧的方向与路的方向一致，叫做**前向弧**。前向弧的全体记为μ^+。

（2）弧与路的方向相反，称为**后向弧**。后向弧的全体记为μ^-。

2.寻求最大流的标号法（Ford－Fulkerson）

从一个可行流出发（若网络中没有给定f，则可以设f是零流)，经过

（1）标号过程　　寻找可增广链

（2）调整过程　　增大可行流流量

（1）标号过程

每个顶点v_x的标号值有两个：

第 1 个标号值——在可能的增广路上，v_x的前驱顶点；

第 2 个标号值记为δ_x——在可能的增广路上可以调整的流量。

①初始化，给发点v_s标号为$(0,\infty)$。

②若顶点v_x已经标号，则对v_x的所有未标号的邻接顶点v_y按以下规则标号：

i）若 $(v_x,v_y)\in A$，且 $f_{xy}<c_{xy}$ 时，令 $\delta_y=\min\{c_{xy}-f_{xy},\delta_x\}$，则给顶点$v_y$标号为$(v_x,\delta_y)$，若 $f_{xy}=c_{xy}$，则不给顶点v_y标号。

ii）$(v_y,v_x)\in A$，且$f_{yx}>0$，令$\delta_y=\min\{f_{yx},\delta_x\}$，则给$v_y$标号为$(-v_x,\delta_y)$，这里第一个标号值$-v_x$，表示在可能的增广路上，$(v_y,v_x)$为反向弧；若$f_{yx}=0$，则不给$v_y$标号。

③不断地重复步骤②直到收点v_t被标号，或不再有顶点可以标号为止。当v_t被标号时，表明存在一条从v_s到v_t的增广路，则转向增流过程(2)。如若v_t点不能被标号，且不存在其它可以标号的顶点时，表明不存在从v_s到v_t的增广路，算法结束，此时所获得的流就是最大流。

（2）增流过程

①令$v_y=v_t$。

②若v_y的标号为(v_x,δ_t)，则$f_{xy}=f_{xy}+\delta_t$；若v_y的标号为$(-v_x,\delta_t)$，则$f_{yx}=f_{yx}-\delta_t$。

③若$v_y=v_s$，把全部标号去掉，并回到标号过程（A）。否则，令$v_y=v_x$，并回到增流过程②。

3.用MATLAB求网络最大流

例 4.17 求图 4.15 中从①到⑧的最大流。

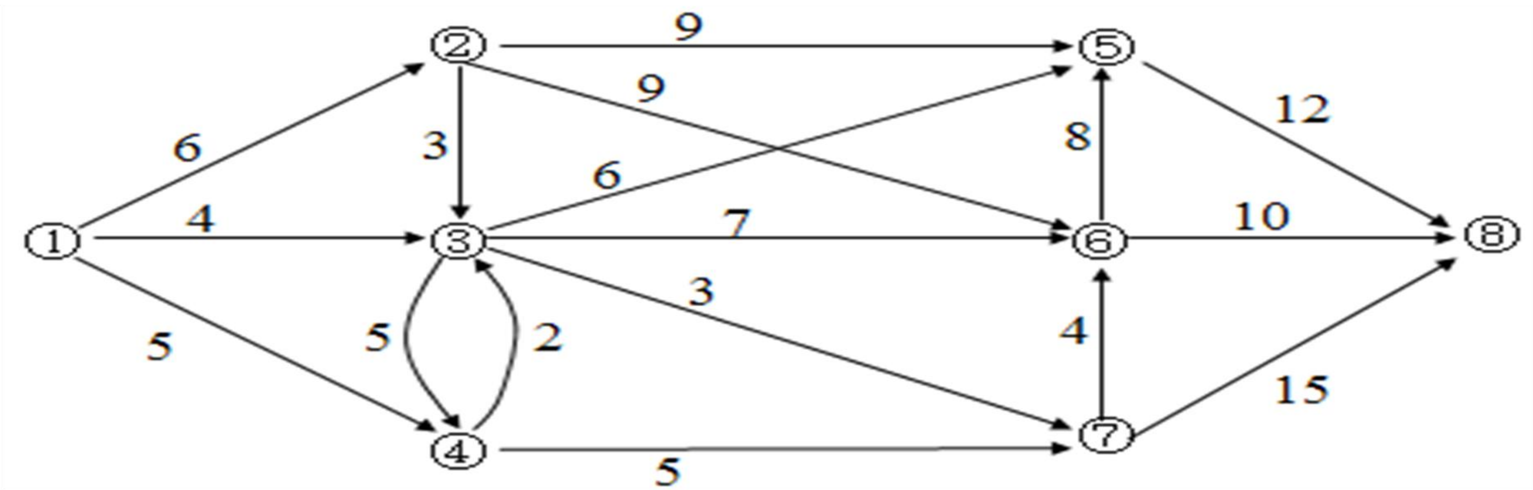


图 4.15 赋权有向图

解 利用 MATLAB 软件求得的最大流量是 15。

```
clc, clear
a = zeros(8); a(1,[2:4])=[6,4,5];
a(2,[3,5,6])=[3,9,9]; a(3,[4:7])=[5,6,7,3];
a(4,[3,7])=[2,5]; a(5,8)=12;
a(6,[5,8])=[8,10]; a(7,[6,8])=[4,15];
G = digraph(a);
H=plot(G,'EdgeLabel',G.Edges.Weight);
[M,F]=maxflow(G,1,8)  %求有向图的最大流
F.Edges, highlight(H,F) %显示最大流并画出最大流
```

利用数学规划求解最大流的 MATLAB 程序如下

```
clc, clear, close all
a = zeros(8); a(1,[2:4])=[6,4,5];
a(2,[3,5,6])=[3,9,9]; a(3,[4:7])=[5,6,7,3];
a(4,[3,7])=[2,5]; a(5,8)=12;
a(6,[5,8])=[8,10]; a(7,[6,8])=[4,15];
prob = optimproblem('ObjectiveSense','max');
f = optimvar('f',8,8,'LowerBound',0);
v = optimvar('v'); prob.Objective = v;
con1 = [sum(f(1,:))==v
      sum(f(:,[2:end-1]))'==sum(f([2:end-1],:),2)
      sum(f(:,8))==v];
prob.Constraints.con1 = con1;
prob.Constraints.con2 = f<=a;
[sol, fval, flag, out] = solve(prob)
ff = sol.f  %显示最大流对应的矩阵
```

4.6.2 最小费用流问题

在许多实际问题中，往往还要考虑网络上流的费用问题。例如，在运输问题中，人们总是希望在完成运输任务的同时，寻求一个使总的运输费用最小的运输方案。

1.最小费用流的线性规划模型

设 f_{ij} 为弧 (v_i,v_j) 上的流量，b_{ij} 为弧 (v_i,v_j) 上的单位费用，c_{ij} 为弧 (v_i,v_j) 上的容量，则最小费用流问题可以用如下的线性规划问题描述：

$$
\min \sum_{(v_i,v_j)\in A} b_{ij} f_{ij},
$$

$$
\text{s.t.}\begin{cases}
\displaystyle\sum_{(v_s,v_j)\in A} f_{sj} = v, \\
\displaystyle\sum_{j:(v_i,v_j)\in A} f_{ij} - \sum_{j:(v_j,v_k)\in A} f_{jk} = 0, \quad j \neq s,t, \\
\displaystyle\sum_{(v_j,v_t)\in A} f_{jt} = v, \\
0 \le f_{ij} \le c_{ij}, \quad \forall (v_i,v_j) \in A.
\end{cases} \tag{4.6}
$$

当$v=$最大流$v_{\max}$时，本问题就是最小费用最大流问题；如果$v>v_{\max}$，本问题无解。

2.最小费用流的对偶算法

1961 年，Busacker 和 Gowan 提出了一种求最小费用流的迭代法。其步骤如下：

(1)求出从发点到收点的最小费用通路$\mu(v_s,v_t)$。

(2)对该通路$\mu(v_s,v_t)$分配最大可能的流量：

$$\bar{f}=\min_{(v_i,v_j)\in\mu(v_s,v_t)}\{c_{ij}\}$$

并让通路上的所有边的容量相应减少$\bar{f}$。这时，对于通路上的饱和边，其单位流费用相应改为∞。

(3)作该通路$\mu(v_s,v_t)$上所有边(v_i,v_j)的反向边(v_j,v_i)。令 $c_{ji}=\bar{f}$，$b_{ji}=-b_{ij}$.

(4)在这样构成的新网络中，重复上述步骤(1), (2) (3)，直到从发点到收点的全部流量等于v为止。

例 4.18 如图 4.16 所示带有运费的网络，求从v_s到v_t的最小费用最大流，其中弧上权重的第 1 个数字是网络的容量，第 2 个数字是网络的单位运费。

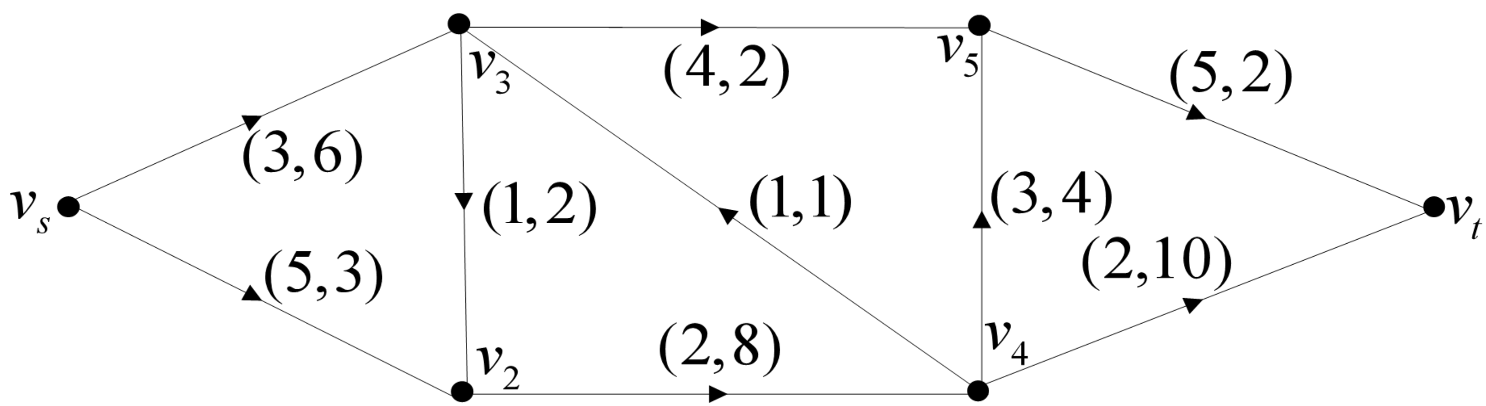
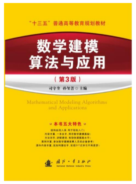


图 4.16 运费网络

解　求得的最大流为5，最小费用为63。

```
clc, clear
NN = cellstr(strcat('v',int2str([2:5]'))); %构造中间节点
NN = {'vs',NN{:},'vt'}; %添加发点和收点
L={'vs','v2',5,3; 'vs','v3',3,6; 'v2','v4',2,8; 'v3','v2',1,2
  'v3','v5',4,2; 'v4','v3',1,1; 'v4','v5',3,4; 'v4','vt',2,10
  'v5','vt',5,2};
G = digraph; G = addnode(G, NN);
G1 = addedge(G,L(:,1),L(:,2),cell2mat(L(:,3)));
[M, F] = maxflow(G1,'vs','vt') %求最大流

G2 = addedge(G, L(:,1), L(:,2), cell2mat(L(:,4)));
c = full(adjacency(G1, 'weighted')); %导出容量矩阵
b = full(adjacency(G2, 'weighted')); %导出费用矩阵
f = optimvar('f',6,6,'LowerBound',0);
prob = optimproblem; prob.Objective = sum(sum(b.*f));
con1 = [sum(f(1,:))==M
      sum(f(:,[2:end-1]))'==sum(f([2:end-1],:),2)
      sum(f(:,end))==M];
prob.Constraints.con1 = con1;
prob.Constraints.con2 = f<=c;
[sol, fval, flag, out] = solve(prob)
ff = sol.f %显示最小费用最大流对应的矩阵
```



02 MATLAB的图论工具箱及其应用

4.7.1 MATLAB图论工具箱的函数

表 4.4 MATLAB 图论工具箱的相关函数

命令名	功能
distances	求图中所有顶点对之间的最短距离
conncomp	找无向图的连通分支，或有向图的强（弱）连通分支
isdag	测试有向图是否含有圈，不含圈返回 1，否则返回 0
isomorphism	确定两个图是否同构
maxflow	计算有向图的最大流
minspantree	在图中求最小生成树
reordernodes	对图顶点重新排序
shortestpath	求图中指定的一对顶点间的最短距离和最短路径
shortestpathtree	求顶点的最短路径树
subgraph	提出子图

4.7.2 应用举例

例 4.19（渡河问题） 某人带狼、羊以及蔬菜渡河，一小船除需人划外，每次只能载一物过河。而人不在场时，狼要吃羊，羊要吃菜，问此人应如何过河?

最短路问题

解　可以用四维向量来表示状态，其中第一分量表示人，第二分量表示狼，第三分量表示羊，第四分量表示蔬菜；当人或物在此岸时相应分量取 1，在对岸时取 0。

根据题意，人不在场时，狼要吃羊，羊要吃菜，因此，人不在场时，不能将狼与羊，羊与蔬菜留在河的任一岸。例如，状态（0，1，1，0）表示人和菜在对岸，而狼和羊在此岸，这时人不在场狼要吃羊，因此，这个状态是不可行的。

通过穷举法将所有可行的状态列举出来，可行的状态有

(1，1，1，1)，(1，1，1，0)，(1，1，0，1)，

(1，0，1，1)，(1，0，1，0)，(0，1，0，1)，

(0，1，0，0)，(0，0，1，0)，(0，0，0，1)，

(0，0，0，0)

可行状态共有十种。每一次的渡河行为改变现有的状态。现构造赋权图$G=(V,E,W)$，其中顶点集合$V=\{v_1,\cdots,v_{10}\}$中的顶点（按照上面的顺序编号）分别表示上述十个可行状态，当且仅当对应的两个可行状态之间存在一个可行转移时两顶点之间才有边连接，并且对应的权重取为 1，当两个顶点之间不存在可行转移时，可以认为存在权重为∞的边。

因此问题变为在图G中寻找一条由初始状态（1，1，1，1）出发，经最小次数转移达到最终状态（0，0，0，0）的转移过程，即求从状态（1，1，1，1）到状态（0，0，0，0）的最短路径。这就将问题转化成了图论中的最短路问题。

该题的难点在于计算邻接矩阵，由于摆渡一次就改变现有的状态，为此再引入一个四维状态转移向量用它来反映摆渡情况。用 1 表示过河，0 表示未过河。例如，(1，1，0，0) 表示人带狼过河。状态转移只有四种情况，用如下的向量表示

(1，0，0，0)，(1，1，0，0)，(1，0，1，0)，(1，0，0，1)。

现在规定状态向量分量与转移向量分量之间的运算为

$$0+0=0,\ 1+0=1,\ 0+1=1,\ 1+1=0$$

根据题意，

通过上面的定义，如果某一个可行状态加上转移向量得到的新向量还属于可行状态，则这两个可行状态对应的顶点之间就存在一条边。用计算机编程时，可以利用普通向量的异或运算实现。

利用 MATLAB 软件求得赋权图G的状态转移关系见图 4.8，状态转移顺序为

$$1 \to 6 \to 3 \to 7 \to 2 \to 8 \to 5 \to 10。$$

经过7次渡河就可以把狼，羊，蔬菜运过河，第一次运羊过河，空船返回；第二次运菜过河，带羊返回；第三次运狼过河，空船返回；第四次运羊过河。

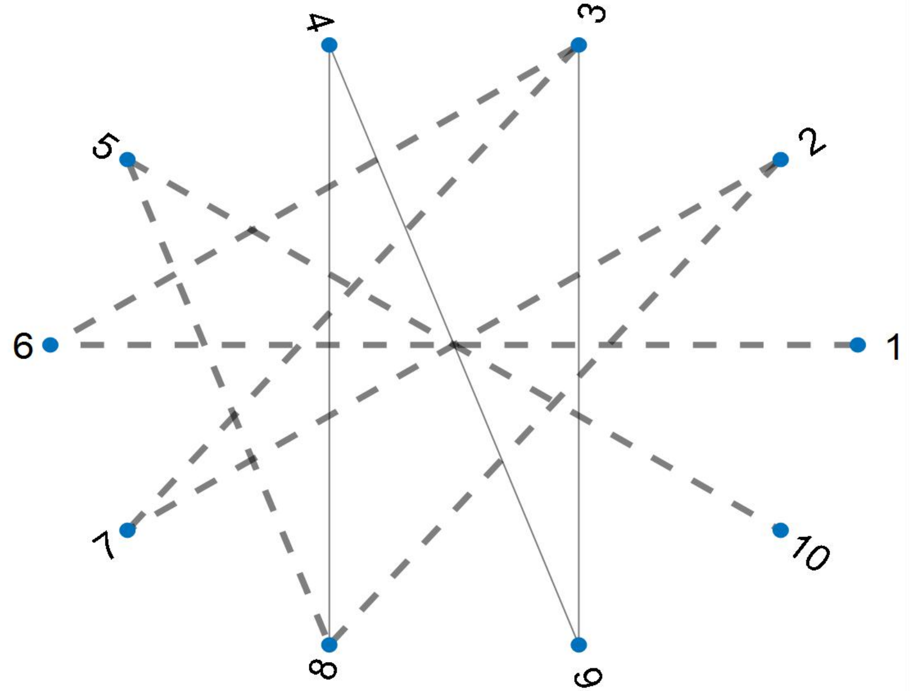


图 4.8 可行状态之间的转移

```
clc, clear, close all
a=[1 1 1 1;1 1 1 0;1 1 0 1;1 0 1 1;1 0 1 0
  0 1 0 1;0 1 0 0;0 0 1 0;0 0 0 1; 0 0 0 0]; %每一行是一个可行状态
b=[1 0 0 0;1 1 0 0;1 0 1 0;1 0 0 1]; %每一行是一个转移状态
w=zeros(10); %邻接矩阵初始化
for i=1:9
    for j=i+1:10
        for k=1:4
            if strfind(xor(a(i,:),b(k,:)),a(j,:))
                w(i,j)=1;
            end
        end
    end
end
[i,j,v]=find(w);  %找非零元素
s=cellstr(int2str([1:10]'));  %构造顶点字符串
G=graph(i,j,v,s)  %构造无向图
[p,d]=shortestpath(G,1,10)  %求最短路径和最短距离
h=plot(G,'Layout','circle','NodeFontSize',12,'EdgeColor','k')
highlight(h,p,'LineWidth',2,'LineStyle','--') %最短路径虚线加粗
```

例 4.20 设有 9 个节点$v_i(i=1,\cdots,9)$，坐标分别为(x_i,y_i)，具体数据见表 4.5。任意两个节点之间的距离为

$$d_{ij}=|x_i-x_j|+|y_i-y_j|,\quad i,j=1,2,\cdots,9,$$

问怎样连接电缆，使每个节点都连通，且所用的电缆总长度最短？

表 4.1 点的坐标数据

i	1	2	3	4	5	6	7	8	9
x_i	0	5	16	20	33	23	35	25	10
y_i	15	20	24	20	25	11	7	0	3

解 以$V=\{v_1,v_2,\cdots,v_9\}$作为顶点集，构造赋权图$G=(V,E,W)$，这里$W=(d_{ij})_{9\times 9}$为邻接矩阵。求电缆总长度最短的问题实际上就是求图G的最小生成树。

求得最小生成树的边集为

$$\{v_1v_2,v_2v_3,v_3v_4,v_3v_5,v_4v_6,v_6v_7,v_6v_8,v_8v_9\},$$

电缆总长度的最小值为110。

```
clc, clear, xy=load('data4_20.txt');
d=mandist(xy); %求xy的两两列向量间的绝对值距离
s=cellstr(strcat('v',int2str([1:9]'))); %构造顶点字符串
G=graph(d,s); %构造无向图
T=minspantree(G); %用默认的Prim算法求最小生成树
L=sum(T.Edges.Weight) %计算最小生成树的权重
T.Edges %显示最小生成树的边集
h=plot(G,'Layout','circle','NodeFontSize',12); %画无向图
highlight(h,T,'EdgeColor','r','LineWidth',2) %加粗最小生成树的边
```