图论算法及其 MATLAB 程序代码

求赋权图 $G=(V, E, F)$**中任意两点间的最短路的** Warshall-Floyd **算法：**

设 $\boldsymbol{A}=(a_{ij})_{n\times n}$ **为赋权图** $G=(V, E, F)$**的矩阵，当** $v_iv_j\in E$ **时** $a_{ij}=F(v_iv_j)$**，否则取** $a_{ii}=0$, $a_{ij}=+\infty(i\neq j)$, d_{ij} **表示从** v_i **到** v_j **点的距离，** r_{ij} **表示从** v_i **到** v_j **点的最短路中一个点的编号.**

① **赋初值. 对所有** i, j, $d_{ij}=a_{ij}$, $r_{ij}=j$. $k=1$. **转向②**

② **更新** d_{ij}, r_{ij}. **对所有** i, j, **若** $d_{ik}+d_{kj}<d_{ij}$, **则令** $d_{ij}=d_{ik}+d_{kj}$, $r_{ij}=k$, **转向③.**

③ **终止判断. 若** $d_{ii}<0$, **则存在一条含有顶点** v_i **的负回路，终止；或者** $k=n$ **终止；否则令** $k=k+1$, **转向②.**

最短路线可由 r_{ij} **得到.**

例 1　求图 6-4 **中任意两点间的最短路.**

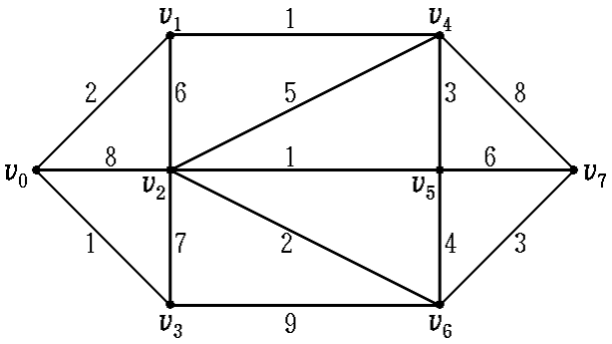

图 6-4

解：用 Warshall-Floyd **算法**, MATLAB **程序代码如下：**

```
n=8;A=[0  2   8   1   Inf  Inf  Inf  Inf
       2  0   6   Inf  1    Inf  Inf  Inf
       8  6   0   7    5    1    2    Inf
       1  Inf 7   0    Inf  Inf  9    Inf
       Inf 1  5   Inf  0    3    Inf  8
       Inf Inf 1  Inf  3    0    4    6
       Inf Inf 2  9    Inf  4    0    3
       Inf Inf Inf Inf  8    6    3    0];   % MATLAB 中, Inf 表示∞
D=A;        %赋初值
for(i=1:n)for(j=1:n)R(i,j)=j;end;end     %赋路径初值
for(k=1:n)for(i=1:n)for(j=1:n)if(D(i,k)+D(k,j)<D(i,j))D(i,j)=D(i,k)+D(k,j);         %更新 dij
       R(i,j)=k;end;end;end         %更新 rij
       k      %显示迭代步数
       D      %显示每步迭代后的路长
       R      %显示每步迭代后的路径
       pd=0;for i=1:n   %含有负权时
            if(D(i,i)<0)pd=1;break;end;end  %存在一条含有顶点 vi 的负回路
       if(pd)break;end        %存在一条负回路，终止程序
end  %程序结束
```

Kruskal 避圈法：将图 G 中的边按权数从小到大逐条考察，按不构成圈的原则加入到 T 中(若有选择时，不同的选择可能会导致最后生成树的权数不同)，直到 $q(T)=p(G)-1$ 为止，即 T 的边数$=G$ 的顶点数-1 为止.

Kruskal 避圈法的 MATLAB 程序代码如下：

```
n=8;A=[0 2 8 1 0 0 0 0
    2 0 6 0 1 0 0 0
    8 6 0 7 5 1 2 0
    1 0 7 0 0 0 9 0
    0 1 5 0 0 3 0 8
    0 0 1 0 3 0 4 6
    0 0 2 9 0 4 0 3
    0 0 0 0 8 6 3 0];
k=1;        %记录 A 中不同正数的个数
for(i=1:n-1)for(j=i+1:n)      %此循环是查找 A 中所有不同的正数
    if(A(i,j)>0)x(k)=A(i,j);  %数组 x 记录 A 中不同的正数
        kk=1;       %临时变量
        for(s=1:k-1)if(x(k)==x(s))kk=0;break;end;end     %排除相同的正数
        k=k+kk;end;end;end
k=k-1       %显示 A 中所有不同正数的个数
for(i=1:k-1)for(j=i+1:k)      %将 x 中不同的正数从小到大排序
    if(x(j)<x(i))xx=x(j);x(j)=x(i);x(i)=xx;end;end;end
T(n,n)=0;  %将矩阵 T 中所有的元素赋值为 0
q=0; %记录加入到树 T 中的边数
for(s=1:k)if(q==n)break;end      %获得最小生成树 T，算法终止
    for(i=1:n-1)for(j=i+1:n)if (A(i,j)==x(s))T(i,j)=x(s);T(j,i)=x(s);  %加入边到树 T 中
        TT=T;       %临时记录 T
        while(1)pd=1;   %砍掉 TT 中所有的树枝
            for(y=1:n)kk=0;
                for(z=1:n)if(TT(y,z)>0)kk=kk+1;zz=z;end;end     %寻找 TT 中的树枝
                if(kk==1)TT(y,zz)=0;TT(zz,y)=0;pd=0;end;end   %砍掉 TT 中的树枝
            if(pd)break;end;end  %已砍掉了 TT 中所有的树枝
        pd=0;       %判断 TT 中是否有圈
        for(y=1:n-1)for(z=y+1:n)if(TT(y,z)>0)pd=1;break;end;end;end
        if(pd)T(i,j)=0;T(j,i)=0;        %假如 TT 中有圈
        else q=q+1;end;end;end;end;end
T     %显示近似最小生成树 T，程序结束
```

求二部图 G 的最大匹配的算法(匈牙利算法)，其基本思想是：从 G 的任意匹配 M 开始，对 X 中所有 M 的非饱和点，寻找 M –增广路. 若不存在 M –增广路，则 M 为最大匹配；若存在 M –增广路 P，则将 P 中 M 与非 M 的边互换得到比 M 多一边的匹配 M_1，再对 M_1 重复上述过程.

设 $G=(X, Y, E)$ 为二部图，其中 $X=\{x_1, x_2, \cdots, x_n\}$, $Y=\{y_1, y_2, \cdots, y_n\}$. 任取 G 的一初始匹配 M (如任取 $e\in E$，则 $M=\{e\}$ 是一个匹配).

① 令 $S=f$, $T=f$，转向②.

② 若 M 饱和 $X\setminus S$ 的所有点，则 M 是二部图 G 的最大匹配. 否则，任取 M 的非饱和点 $u\in X\setminus S$，令 $S=S\cup\{u\}$，转向③.

③ 记 $N(S)=\{v \mid u\in S, uv\in E\}$. 若 $N(S)=T$，转向②. 否则取 $y\in N(S)\setminus T$. 若 y 是 M 的饱和点，转向④，否则转向⑤.

④ 设 $xy\in M$，则令 $S=S\cup\{x\}$, $T=T\cup\{y\}$，转向③.

⑤ $u-y$ 路是 M–增广路，设为 P，并令 $M=M\oplus P$，转向①. 这里 $M\oplus P=M\cup P\setminus M\cap P$，是对称差.

由于计算 M–增广路 P 比较麻烦，因此将迭代步骤改为：

① 将 X 中 M 的所有非饱和点(不是 M 中某条边的端点)都给以标号 0 和标记*，转向②.

② 若 X 中所有有标号的点都已去掉了标记*，则 M 是 G 的最大匹配. 否则任取 X 中一个既有标号又有标记*的点 x_i，去掉 x_i 的标记*，转向③.

③ 找出在 G 中所有与 x_i 邻接的点 y_j (即 $x_iy_j\in E$)，若所有这样的 y_j 都已有标号，则转向②，否则转向④.

④ 对与 x_i 邻接且尚未给标号的 y_j 都给定标号 i. 若所有的 y_j 都是 M 的饱和点，则转向⑤，否则逆向返回. 即由其中 M 的任一个非饱和点 y_j 的标号 i 找到 x_i，再由 x_i 的标号 k 找到 y_k，$\cdots$，最后由 y_t 的标号 s 找到标号为 0 的 x_s 时结束，获得 M –增广路 $x_sy_t\ \cdots\ x_iy_j$，记 $P=\{x_sy_t, \cdots, x_iy_j\}$，重新记 M 为 $M\oplus P$，转向①.

⑤ 将 y_j 在 M 中与之邻接的点 x_k (即 $x_ky_j\in M$)，给以标号 j 和标记*，转向②.

例 1　求图 6-9 中所示的二部图 G 的最大匹配.

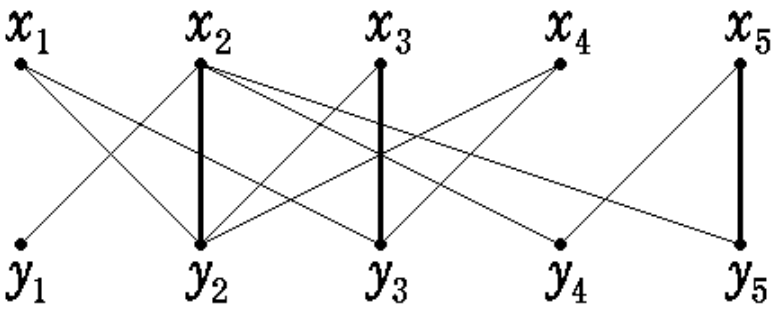

图 6-9

匈牙利算法的 MATLAB **程序代码如下：**

```
m=5;n=5;A=[0  1   1   0   0
           1   1   0   1   1
           0   1   1   0   0
           0   1   1   0   0
           0   0   0   1   1];
M(m,n)=0;
for(i=1:m)for(j=1:n)if(A(i,j))M(i,j)=1;break;end;end      %求初始匹配 M
    if(M(i,j))break;end;end      %获得仅含一条边的初始匹配 M
while(1)
    for(i=1:m)x(i)=0;end  %将记录 X 中点的标号和标记*
    for(i=1:n)y(i)=0;end  %将记录 Y 中点的标号和标记*
    for(i=1:m)pd=1;       %寻找 X 中 M 的所有非饱和点
        for(j=1:n)if(M(i,j))pd=0;end;end
        if(pd)x(i)=-n-1;end;end     %将 X 中 M 的所有非饱和点都给以标号 0 和标记*，程序中用 n+1 表示 0 标号，标号为负数时表示标记*
    pd=0;
    while(1)xi=0;
        for(i=1:m)if(x(i)<0)xi=i;break;end;end       %假如 X 中存在一个既有标号又有标记*的点，则任取 X 中一个既有标号又有标记*的点 xi
        if(xi==0)pd=1;break;end   %假如 X 中所有有标号的点都已去掉了标记*，算法终止
        x(xi)=x(xi)*(-1); %去掉 xi 的标记*
        k=1;
        for(j=1:n)if(A(xi,j)&y(j)==0)y(j)=xi;yy(k)=j;k=k+1;end;end    %对与 xi 邻接且尚未给标号的 yj 都给以标号 i
        if(k>1)k=k-1;
            for(j=1:k)pdd=1;
                for(i=1:m)if(M(i,yy(j)))x(i)=-yy(j);pdd=0;break;end;end       %将 yj 在 M 中与之邻接的点 xk (即 xkyj∈M)，给以标号 j 和标记*
                if(pdd)break;end;end
            if(pdd)k=1;j=yy(j);     %yj 不是 M 的饱和点
                while(1)P(k,2)=j;P(k,1)=y(j);j=abs(x(y(j)));  %任取 M 的一个非饱和点 yj，逆向返回
                    if(j==n+1)break;end  %找到 X 中标号为 0 的点时结束，获得 M-增广路 P
                    k=k+1;end
                for(i=1:k)if(M(P(i,1),P(i,2)))M(P(i,1),P(i,2))=0;    %将匹配 M 在增广路 P 中出现的边去掉
                    else M(P(i,1),P(i,2))=1;end;end %将增广路 P 中没有在匹配 M 中出现的边加入到匹配 M 中
                break;end;end;end
    if(pd)break;end;end  %假如 X 中所有有标号的点都已去掉了标记*，算法终止
M    %显示最大匹配 M，程序结束
```

利用可行点标记求最佳匹配的算法步骤如下：

设 $G=(X,Y,E,F)$**为完备的二部赋权图，**L **是其一个初始可行点标记，通常取**

$$\begin{cases} L(x)=\max\{F(xy)\mid y\in Y\}, & x\in X, \\ L(y)=0, & y\in Y. \end{cases}$$

M **是** G_L **的一个匹配.**

① **若** X **的每个点都是** M **的饱和点，则** M **是最佳匹配. 否则取** M **的非饱和点** $u\in X$**，令** $S=\{u\}, T=f$**，转向②.**

② **记** $N_L(S)=\{v\mid u\in S, uv\in E_L\}$**. 若** $N_L(S)=T$**，则** G_L **没有完美匹配，转向③. 否则转向④.**

③ **调整可行点标记，计算**

$$a_L=\min\{L(x)+L(y)-F(xy)\mid x\in S, y\in Y\setminus T\}.$$

由此得新的可行顶点标记

$$H(v)=\begin{cases} L(v)-a_L, & v\in S, \\ L(v)+a_L, & v\in T, \\ L(v), & \text{否则}. \end{cases}$$

令 $L=H, G_L=G_H$**，重新给出** G_L **的一个匹配** M**，转向①.**

④ **取** $y\in N_L(S)\setminus T$**，若** y **是** M **的饱和点，转向⑤. 否则，转向⑥.**

⑤ **设** $xy\in M$**，则令** $S=S\cup\{x\}, T=T\cup\{y\}$**，转向②.**

⑥ **在** G_L **中的** $u-y$ **路是** M**–增广路，记为** P**，并令** $M=M\oplus P$**，转向①.**

利用可行点标记求最佳匹配算法的 MATLAB **程序代码如下：**

```
n=4;A=[4  5    5    1
       2  2    4    6
       4  2    3    3
       5  0    2    1];
for(i=1:n)L(i,1)=0;L(i,2)=0;end
for(i=1:n)for(j=1:n)if(L(i,1)<A(i,j))L(i,1)=A(i,j);end;      %初始可行点标记 L
            M(i,j)=0;end;end
for(i=1:n)for(j=1:n)   %生成子图 Gl
            if(L(i,1)+L(j,2)==A(i,j))Gl(i,j)=1;
            else Gl(i,j)=0;end;end;end
ii=0;jj=0;
for(i=1:n)for(j=1:n)if(Gl(i,j))ii=i;jj=j;break;end;end
      if(ii)break;end;end    %获得仅含 Gl 的一条边的初始匹配 M
M(ii,jj)=1;
for(i=1:n)S(i)=0;T(i)=0;NlS(i)=0;end
while(1)
      for(i=1:n)k=1;
```

```
        for(j=1:n)if(M(i,j))k=0;break;end;end
        if(k)break;end;end
    if(k==0)break;end     %获得最佳匹配 M, 算法终止
    S(1)=i;jss=1;jst=0;    %S={xi}, T=ϕ
    while(1)
        jsn=0;
        for(i=1:jss)for(j=1:n)if(Gl(S(i),j))jsn=jsn+1;NlS(jsn)=j; %NL(S)={v|u∈S,uv∈EL}
                for(k=1:jsn-1)if(NlS(k)==j)jsn=jsn-1;end;end;end;end;end
        if(jsn==jst)pd=1;       %判断 NL(S)=T?
            for(j=1:jsn)if(NlS(j)~=T(j))pd=0;break;end;end;end
        if(jsn==jst&pd)al=Inf;%如果 NL(S)=T, 计算 al, Inf 为∞
            for(i=1:jss)for(j=1:n)pd=1;
                for(k=1:jst)if(T(k)==j)pd=0;break;end;end
                if(pd&al>L(S(i),1)+L(j,2)-A(S(i),j))al=L(S(i),1)+L(j,2)-A(S(i),j);end;end;end
            for(i=1:jss)L(S(i),1)=L(S(i),1)-al;end  %调整可行点标记
            for(j=1:jst)L(T(j),2)=L(T(j),2)+al;end  %调整可行点标记
            for(i=1:n)for(j=1:n)    %生成子图 GL
                    if(L(i,1)+L(j,2)==A(i,j))Gl(i,j)=1;
                    else Gl(i,j)=0;end
                    M(i,j)=0;k=0;end;end
            ii=0;jj=0;
            for(i=1:n)for(j=1:n)if(Gl(i,j))ii=i;jj=j;break;end;end
                if(ii)break;end;end    %获得仅含 Gl 的一条边的初始匹配 M
            M(ii,jj)=1;break
        else %NL(S)≠T
            for(j=1:jsn)pd=1;       %取 y∈NL(S)\T
                for(k=1:jst)if(T(k)==NlS(j))pd=0;break;end;end
                if(pd)jj=j;break;end;end
            pd=0;       %判断 y 是否为 M 的饱和点
            for(i=1:n)if(M(i,NlS(jj)))pd=1;ii=i;break;end;end
            if(pd)jss=jss+1;S(jss)=ii;jst=jst+1;T(jst)=NlS(jj);  %S=S∪{x}, T=T∪{y}
            else %获得 Gl 的一条 M-增广路, 调整匹配 M
                for(k=1:jst)M(S(k),T(k))=1;M(S(k+1),T(k))=0;end
                if(jst==0)k=0;end
                M(S(k+1),NlS(jj))=1;break;end;end;end;end
MaxZjpp=0;
for(i=1:n)for(j=1:n)if(M(i,j))MaxZjpp=MaxZjpp+A(i,j);end;end;end
M     %显示最佳匹配 M
MaxZjpp   %显示最佳匹配 M 的权, 程序结束
```

从一个可行流 f 开始，求最大流的 Ford--Fulkerson 标号算法的基本步骤：

⑴ 标号过程

① 给发点 v_s 以标号$(+, +\infty)$，$d_s = +\infty$.

② 选择一个已标号的点 x，对于 x 的所有未给标号的邻接点 y，按下列规则处理：

当 $yx \in E$，且 $f_{yx} > 0$ 时，令$d_y = \min\{f_{yx}, d_x\}$，并给 y 以标号 $(x-, d_y)$.

当 $xy \in E$，且 $f_{xy} < C_{xy}$ 时，令$d_y = \min\{C_{xy} - f_{xy}, d_x\}$，并给 y 以标号 $(x+, d_y)$.

③ 重复②直到收点 v_t 被标号或不再有点可标号时为止. 若 v_t 得到标号，说明存在一条可增广链，转⑵调整过程；若 v_t 未得到标号，标号过程已无法进行时，说明 f 已经是最大流.

⑵ 调整过程

④ 决定调整量$d = d_{vt}$，令 $u = v_t$.

⑤ 若 u 点标号为$(v+, d_u)$，则以 $f_{vu} + d$ 代替 f_{vu}；若 u 点标号为$(v-, d_u)$，则以 $f_{vu} - d$ 代替 f_{vu}.

⑥ 若 $v = v_s$，则去掉所有标号转⑴重新标号；否则令 $u = v$，转⑤.

算法终止后，令已有标号的点集为 S，则割集(S, S^c)为最小割，从而 $W_f = C(S, S^c)$.

例 1　求图 6-19 所示网络的最大流.

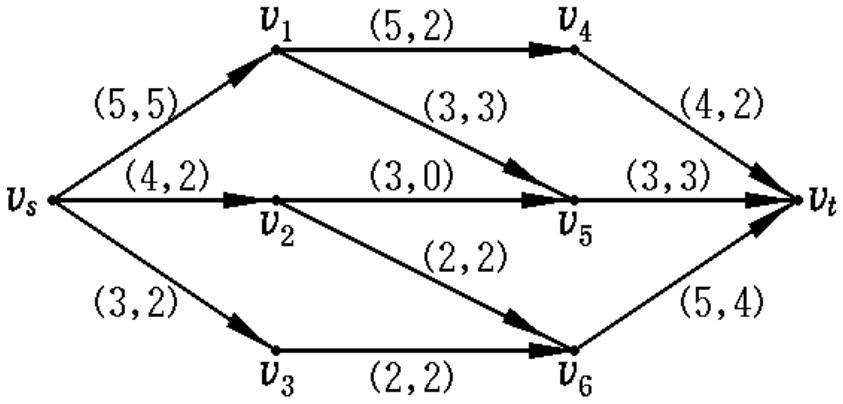

图 6-19

利用 Ford--Fulkerson 标号法求最大流算法的 MATLAB 程序代码如下：

```
n=8;C=[0 5   4   3   0   0   0   0
       0   0   0   0   5   3   0   0
       0   0   0   0   0   3   2   0
       0   0   0   0   0   0   2   0
       0   0   0   0   0   0   0   4
       0   0   0   0   0   0   0   3
       0   0   0   0   0   0   0   5
       0   0   0   0   0   0   0   0];   %弧容量
for(i=1:n)for(j=1:n)f(i,j)=0;end;end    %取初始可行流 f 为零流
for(i=1:n)No(i)=0;d(i)=0;end      %No,d 记录标号
```

```
while(1)
    No(1)=n+1;d(1)=Inf;  %给发点 vs 标号
    while(1)pd=1;   %标号过程
        for(i=1:n)if(No(i))      %选择一个已标号的点 vi
            for(j=1:n)if(No(j)==0&f(i,j)<C(i,j))      %对于未给标号的点 vj，当 vivj 为非饱和弧时
                No(j)=i;d(j)=C(i,j)-f(i,j);pd=0;
                if(d(j)>d(i))d(j)=d(i);end
            elseif(No(j)==0&f(j,i)>0)    %对于未给标号的点 vj，当 vjvi 为非零流弧时
                No(j)=-i;d(j)=f(j,i);pd=0;
                if(d(j)>d(i))d(j)=d(i);end;end;end;end;end
        if(No(n)|pd)break;end;end %若收点 vt 得到标号或者无法标号，终止标号过程
    if(pd)break;end %vt 未得到标号，f 已是最大流，算法终止
    dvt=d(n);t=n;    %进入调整过程，dvt 表示调整量
    while(1)
        if(No(t)>0)f(No(t),t)=f(No(t),t)+dvt;     %前向弧调整
        elseif(No(t)<0)f(No(t),t)=f(No(t),t)-dvt;end  %后向弧调整
        if(No(t)==1)for(i=1:n)No(i)=0;d(i)=0;end;break;end   %当 t 的标号为 vs 时，终止调整过程
        t=No(t);end;end;      %继续调整前一段弧上的流 f
wf=0;for(j=1:n)wf=wf+f(1,j);end %计算最大流量
f      %显示最大流
wf     %显示最大流量
No     %显示标号，由此可得最小割，程序结束
```

设网络 $G=(V,E,C)$，取初始可行流 f 为零流，求解最小费用流问题的迭代步骤：

① 构造有向赋权图 $G_f=(V,E_f,F)$，对于任意的 $v_iv_j\in E$，E_f，F 的定义如下：

当 $f_{ij}=0$ 时，$v_iv_j\in E_f$，$F(v_iv_j)=b_{ij}$；

当 $f_{ij}=C_{ij}$ 时，$v_jv_i\in E_f$，$F(v_jv_i)=-b_{ij}$；

当 $0<f_{ij}<C_{ij}$ 时，$v_iv_j\in E_f$，$F(v_iv_j)=b_{ij}$，$v_jv_i\in E_f$，$F(v_jv_i)=-b_{ij}$.

转向②.

② 求出有向赋权图 $G_f=(V,E_f,F)$ 中发点 v_s 到收点 v_t 的最短路 m，若最短路 m 存在转向③；否则 f 是所求的最小费用最大流，停止.

③ 增流. 同求最大流的方法一样，重述如下：

令 $d_{ij}=\begin{cases}C_{ij}-f_{ij}, & v_iv_j\in m^+,\\ f_{ij}, & v_iv_j\in m^-.\end{cases}$ $d=\min\{d_{ij}\mid v_iv_j\in m\}$，重新定义流 $f=\{f_{ij}\}$ 为

$$f_{ij}=\begin{cases}f_{ij}+d, & v_iv_j\in m^+,\\ f_{ij}-d, & v_iv_j\in m^-,\\ f_{ij} & \text{其它}.\end{cases}$$

如果 W_f 大于或等于预定的流量值，则适当减少 d 值，使 W_f 等于预定的流量值，那么 f 是所求的最小费用流，停止；否则转向①.

求解含有负权的有向赋权图 $G=(V,E,F)$ 中某一点到其它各点最短路的 Ford 算法.

当 $v_iv_j\in E$ 时记 $w_{ij}=F(v_iv_j)$，否则取 $w_{ii}=0$，$w_{ij}=+\infty(i\neq j)$. v_1 到 v_i 的最短路长记为 $p(i)$，v_1 到 v_i 的最短路中 v_i 的前一个点记为 $q(i)$. Ford 算法的迭代步骤：

① 赋初值 $p(1)=0$，$p(i)=+\infty$，$q(i)=i$，$i=2,3,\cdots,n$.

② 更新 $p(i)$，$q(i)$. 对于 $i=2,3,\cdots,n$ 和 $j=1,2,\cdots,n$，如果 $p(i)<p(j)+w_{ji}$，则令

$$p(i)=p(j),\ q(i)=j.$$

③ 终止判断：若所有的 $p(i)$ 都无变化，停止；否则转向②.

在算法的每一步中，$p(i)$ 都是从 v_1 到 v_i 的最短路长度的上界. 若不存在负长回路，则从 v_1 到 v_i 的最短路长度是 $p(i)$ 的下界，经过 $n-1$ 次迭代后 $p(i)$ 将保持不变. 若在第 n 次迭代后 $p(i)$ 仍在变化时，说明存在负长回路.

例 2　在图 6-22 所示运输网络上，求 s 到 t 的最小费用最大流，括号内为(C_{ij}, b_{ij}).

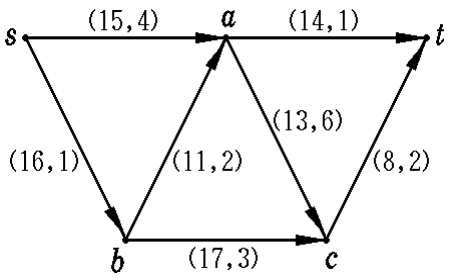

图 6-22

求最小费用最大流算法的 MATLAB 程序代码如下：

```
n=5;C=[0     15   16   0    0
       0     0    0    13   14
       0     11   0    17   0
       0     0    0    0    8
       0     0    0    0    0];   %弧容量
b=[0     4    1    0    0
   0     0    0    6    1
   0     2    0    3    0
   0     0    0    0    2
   0     0    0    0    0];   %弧上单位流量的费用
wf=0;wf0=Inf;   %wf 表示最大流量, wf0 表示预定的流量值
for(i=1:n)for(j=1:n)f(i,j)=0;end;end     %取初始可行流 f 为零流
while(1)
    for(i=1:n)for(j=1:n)if(j~=i)a(i,j)=Inf;end;end;end%构造有向赋权图
    for(i=1:n)for(j=1:n)if(C(i,j)>0&f(i,j)==0)a(i,j)=b(i,j);
        elseif(C(i,j)>0&f(i,j)==C(i,j))a(j,i)=-b(i,j);
        elseif(C(i,j)>0)a(i,j)=b(i,j);a(j,i)=-b(i,j);end;end;end
    for(i=2:n)p(i)=Inf;s(i)=i;end       %用 Ford 算法求最短路，赋初值
    for(k=1:n)pd=1;       %求有向赋权图中 vs 到 vt 的最短路
        for(i=2:n)for(j=1:n)if(p(i)>p(j)+a(j,i))p(i)=p(j)+a(j,i);s(i)=j;pd=0;end;end;end
        if(pd)break;end;end  %求最短路的 Ford 算法结束
    if(p(n)==Inf)break;end      %不存在 vs 到 vt 的最短路，算法终止. 注意在求最小费用最大流时构造有
向赋权图中不会含负权回路，所以不会出现 k=n
    dvt=Inf;t=n;      %进入调整过程, dvt 表示调整量
    while(1)   %计算调整量
        if(a(s(t),t)>0)dvtt=C(s(t),t)-f(s(t),t);      %前向弧调整量
        elseif(a(s(t),t)<0)dvtt=f(t,s(t));end      %后向弧调整量
        if(dvt>dvtt)dvt=dvtt;end
        if(s(t)==1)break;end  %当 t 的标号为 vs 时，终止计算调整量
        t=s(t);end %继续调整前一段弧上的流 f
    pd=0;if(wf+dvt>=wf0)dvt=wf0-wf;pd=1;end%如果最大流量大于或等于预定的流量值
    t=n;while(1)      %调整过程
        if(a(s(t),t)>0)f(s(t),t)=f(s(t),t)+dvt;              %前向弧调整
        elseif(a(s(t),t)<0)f(t,s(t))=f(t,s(t))-dvt;end     %后向弧调整
        if(s(t)==1)break;end  %当 t 的标号为 vs 时，终止调整过程
        t=s(t);end
    if(pd)break;end %如果最大流量达到预定的流量值
    wf=0; for(j=1:n)wf=wf+f(1,j);end;end  %计算最大流量
zwf=0;for(i=1:n)for(j=1:n)zwf=zwf+b(i,j)*f(i,j);end;end %计算最小费用
f       %显示最小费用最大流
```

```
wf    %显示最小费用最大流量
zwf   %显示最小费用，程序结束
```