第 5 讲　图论模型

司守奎
烟台市，海军航空大学
Email：sishoukui@163.com

5.1　图与网络的数据结构

5.1.1 图的概念

定义 5.1　非赋权图

一个非赋权图是由点集 $V=\{v_1,v_2,\cdots,v_n\}$ 和 V 中元素的无序对的一个集合 $E=\{e_1,e_2,\cdots,e_m\}$ 所构成的二元组，记为 $G=(V,E)$，V 中的元素 v_i 叫做顶点，E 中的元素 e_k 叫做边。

定义 5.2　赋权图

赋权图（网络）G 是一个三元组，记为 $G=(V,E,W)$，其中 $V=\{v_1,\cdots,v_n\}$ 为顶点集合，E 为边的集合，$W=(w_{ij})_{n\times n}$ 为邻接矩阵（或权重矩阵），其中

$$w_{ij}=\begin{cases}\text{顶点}v_i\text{与}v_j\text{之间边的权重}, & (v_i,v_j)\in E\\ 0(\text{或}\infty), & v_i\text{与}v_j\text{之间无边时.}\end{cases} \tag{1}$$

注 5.1　当两个顶点之间不存在边或弧时，根据实际问题的含义或算法需要，对应的权重可以取为 0 或 ∞。

当顶点与顶点之间的关系是对称关系时，对应的图是无向图，除非特殊说明，我们所说的图都是指无向图。当顶点与顶点之间的关系是非对称关系时，对应的图为有向图。

定义 5.3　有向图

有向图 D 是一个二元组，记为 $D=(V,A)$，其中 V 为顶点集合，A 为弧（带箭头的边）的集合。

定义 5.4　赋权有向图

赋权有向图 D 是一个三元组，记为 $D=(V,A,W)$，其中 $V=\{v_1,\cdots,v_n\}$ 为顶点集合，A 为弧（带箭头的边）的集合，$W=(w_{ij})_{n\times n}$ 为邻接矩阵，其中

$$w_{ij}=\begin{cases}\text{顶点}v_i\to v_j\text{的弧的权重, 当}v_i\to v_j\text{有弧时}\\ 0(\text{或}\infty), & \text{当}v_i\to v_j\text{无弧时.}\end{cases} \tag{2}$$

当 G（或 D）为非赋权图时，也可以看成赋权图，邻接矩阵

$$w_{ij}=\begin{cases}1, & \text{当}v_i\text{与}v_j\text{间有边(弧)时,}\\ 0, & \text{当}v_i\text{与}v_j\text{间无边(弧)时.}\end{cases} \tag{3}$$

5.1.2　MATLAB 中邻接矩阵的表示

在 MATLAB 中邻接矩阵以稀疏矩阵的格式存储。

在数学上，稀疏矩阵是指矩阵中零元素很多，非零元素很少的矩阵。对于计算机的数据结构，稀疏矩阵只是一种存储格式，只存放非零元素的行地址、列地址和非零元素本身的值，即按如下方式存储

（非零元素的行地址，非零元素的列地址），非零元素的值。

在 MATLAB 中无向图和有向图邻接矩阵的使用上有很大差异。

对于有向图，只要写出邻接矩阵，直接使用 MATLAB 的 sparse 命令，把邻接矩阵转化为稀疏矩阵的表示方式，供 MATLAB 工具箱使用。

对于无向图，由于邻接矩阵是对称阵，MATLAB 中只使用邻接矩阵的下三角元素，即需要 MATLAB 先截取邻接矩阵的下三角部分，再用 sparse 命令转化为稀疏矩阵。

在 MATLAB 中，普通矩阵使用 sparse 命令变成稀疏矩阵，稀疏矩阵使用 full 命令变成普通矩阵。

另外要注意，在数学上按照邻接矩阵的定义式(1)，如果两个顶点间无边连接，对应的元素为 0 或 ∞。在 MATLAB 工具箱中，由于使用稀疏矩阵，隐含着两个顶点间无边连接时，

邻接矩阵的对应元素为 0。

5.1.3　例题

例 5.1　**图 5.1** 所示的图，其邻接矩阵为

$$A=\begin{bmatrix}0&9&2&4&7\\9&0&3&4&0\\2&3&0&8&4\\4&4&8&0&6\\7&0&5&6&0\end{bmatrix}$$

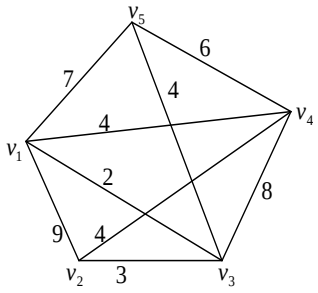

图 5.1　赋权无向图

用 MATLAB 重新画出图 5.1。

```
clc, clear
a=zeros(5); %邻接矩阵初始化
a(1,[2:5])=[9 2 4 7]; a(2,[3 4])=[3 4]; %输入邻接矩阵的上三角元素
a(3,[4 5])=[8 4]; a(4,5)=6;
a=a'; b=sparse(a)　%变成下三角矩阵，并转化为稀疏矩阵
h=biograph(b,[],'ShowWeights','on','ShowArrows','off')　%生成图形对象
set(h,'LayoutType','equilibrium'); %设置属性:图形的布局是平衡的
view(h) %显示图形
```

例 5.2　图 5.2 所表示的有向图的邻接矩阵

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

用 MATLAB 重新画出图 5.2。

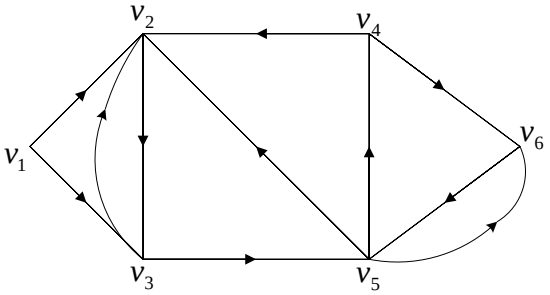

图 5.2　非赋权有向图

```
clc, clear
a=zeros(6);
a(1,[2 3])=1; a(2,3)=1; a(3,[2 5])=1;
a(4,[2 6])=1; a(5,[2 4 6])=1; a(6,5)=1;
b=sparse(a);
h=biograph(b,[],'ShowArrows','on')　%生成图形对象
set(h,'LayoutType','equilibrium'); %设置属性:图形的布局是平衡的
view(h) %显示图形
```

5.2　MATLAB 图论工具箱的命令

MATLAB 图论工具箱的命令见表 5.1。

表 5.1　MATLAB 图论工具箱的相关命令

命令名	功能
graphallshortestpaths	求图中所有顶点对之间的最短距离
graphconncomp	找无向图的连通分支，或有向图的强（弱）连通分支
graphisdag	测试有向图是否含有圈，不含圈返回 1，否则返回 0
graphisomorphism	确定两个图是否同构，同构返回 1，否则返回 0
graphisspantree	确定一个图是否是生成树，是返回 1，否则返回 0
graphmaxflow	计算有向图的最大流
graphminspantree	求图的最小生成树
graphpred2path	把前驱顶点序列变成路径的顶点序列
graphshortestpath	求图中指定的一对顶点间的最短距离和最短路径
graphtopoorder	执行有向无圈图的拓扑排序
graphtraverse	求从一顶点出发，所能遍历图中的顶点

5.3　最小生成树

5.3.1　prim 算法构造最小生成树

构造连通赋权图 $G=(V,E,W)$ 的最小生成树，设置两个集合 P 和 Q，其中 P 用于存放 G 的最小生成树中的顶点，集合 Q 存放 G 的最小生成树中的边。令集合 P 的初值为 $P=\{v_1\}$（假设构造最小生成树时，从顶点 v_1 出发），集合 Q 的初值为 $Q=\Phi$（空集）。prim 算法的思想是，从所有 $p\in P$，$v\in V-P$ 的边中，选取具有最小权值的边 pv，将顶点 v 加入集合 P 中，将边 pv 加入集合 Q 中，如此不断重复，直到 $P=V$ 时，最小生成树构造完毕，这时集合 Q 中包含了最小生成树的所有边。

prim 算法如下：

（1）$P=\{v_1\}$，$Q=\Phi$；

（2）while $P\sim=V$

找最小边 pv，其中 $p\in P, v\in V-P$；

$P=P+\{v\}$；

$Q=Q+\{pv\}$；

end

5.3.2　Kruskal 算法构造最小生成树

科茹斯克尔（Kruskal）算法是一个好算法。Kruskal 算法如下：

（1）选 $e_1\in E$，使得 e_1 是权值最小的边。

（2）若 $e_1,e_2,\cdots,e_i$ 已选好，则从 $E-\{e_1,e_2,\cdots,e_i\}$ 中选取 e_{i+1}，使得

i）$\{e_1,e_2,\cdots,e_i,e_{i+1}\}$ 中无圈，且

ii）e_{i+1} 是 $E-\{e_1,e_2,\cdots,e_i\}$ 中权值最小的边。

（3）直到选得 $e_{|V|-1}$ 为止。

5.3.2 最小生成树举例

例 5.3　一个乡有 9 个自然村，其间道路及各道路长度如图 5.3 所示，各边上的数字表示距离，问架设通讯线时，如何拉线才能使用线最短。

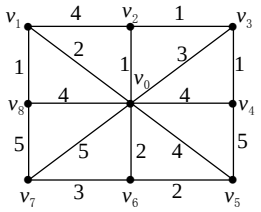

图 5.3　连通图及对应的最小生成树

解　这就是一个最小生成树问题，用 Kruskal 算法求解。先将边按大小顺序由小至大排列：

$(v_0,v_2)=1$，$(v_2,v_3)=1$，$(v_3,v_4)=1$，$(v_1,v_8)=1$，$(v_0,v_1)=2$，

$(v_0,v_6)=2$，$(v_5,v_6)=2$，$(v_0,v_3)=3$，$(v_6,v_7)=3$，$(v_0,v_4)=4$，

$(v_0,v_5)=4$，$(v_0,v_8)=4$，$(v_1,v_2)=4$，$(v_0,v_7)=5$，$(v_7,v_8)=5$，

$(v_4,v_5)=5$

然后按照边的排列顺序，取定

$e_1=(v_0,v_2)$，$e_2=(v_2,v_3)$，$e_3=(v_3,v_4)$，$e_4=(v_1,v_8)$，

$e_5=(v_0,v_1)$，$e_6=(v_0,v_6)$，$e_7=(v_5,v_6)$，

由于下一个未选边中的最小权边 (v_0,v_3) 与已选边 e_1,e_2 构成圈，所以排除。选 $e_8=(v_6,v_7)$。得到图 5.4，就是图 G 的一颗最小生成树，它的权是 13。

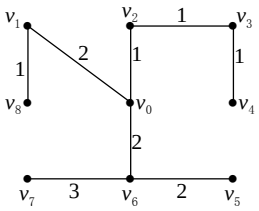

图 5.4　生成的最小生成树

求最小生成树的 Kruskal 算法的 MATLAB 程序如下（用 MATLAB 计算时，顶点 $v_0,v_1,\cdots,v_8$ 分别编号为 $1,2,\cdots,9$）：

```
clc, clear
a=zeros(9);
a(1,[2:9])=[2 1 3 4 4 2 5 4];
a(2,[3 9])=[4 1]; a(3,4)=1; a(4,5)=1;
a(5,6)=5; a(6,7)=2; a(7,8)=3; a(8,9)=5;
a=a';   %转成 MATLAB 需要的下三角元素
a=sparse(a);   %转换为稀疏矩阵
b=graphminspantree(a,'Method','Kruskal') %注意要写 Kruskal 算法，否则使用 Prim 算法
L=sum(sum(b))   %求最小生成树的权重
view(biograph(b,[],'ShowArrows','off','ShowWeights','on'))   %画最小生成树,
```

5.3.3　最小生成树的数学规划模型

根据最小生成树问题的实际意义和实现方法，也可以用数学规划模型来描述，同时能够方便地应用 LINGO 软件来求解这类问题。

顶点 v_1 表示树根，总共有 n 个顶点。顶点 v_i 到顶点 v_j 边的权重用 w_{ij} 表示，当两个顶点之间没有边时，对应的权重用 M（充分大的实数）表示，这里 $w_{ii}=M,i=1,2,\cdots,n$。

引入 0–1 变量

$$x_{ij}=\begin{cases}1, & \text{当从}v_i\text{到}v_j\text{的边在树中},\\ 0, & \text{当从}v_i\text{到}v_j\text{的边不在树中}.\end{cases}$$

目标函数是使得 $z=\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_{ij}$ 最小化。

约束条件分成如下 4 类：

（1）根 v_1 至少有一条边连接到其他的顶点，

$$\sum_{j=1}^{n}x_{1j}\geq 1.$$

（2）除根外，每个顶点只能有一条边进入，

$$\sum_{i=1}^{n}x_{ij}=1,\quad j=2,\cdots,n.$$

以上两约束条件是必要的，但不是充分的，需要增加一组变量 $u_j(j=1,2,\cdots,n)$，再附加约束条件[3]：

（3）限制 u_j 的取值范围为：

$$u_1=0,\quad 1\leq u_i\leq n-1,\quad i=2,3,\cdots,n.$$

（4）各条边不构成子圈，

$$u_j\geq u_k+x_{kj}-(n-2)(1-x_{kj})+(n-3)x_{jk},\quad k=1,\cdots,n,\ j=2,\cdots,n.$$

综上所述，最小生成树问题的 0－1 整数规划模型如下：

$$\min\quad z=\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_{ij}\ , \tag{4}$$

$$\text{s.t.}\begin{cases}\sum_{j=1}^{n}x_{1j}\geq 1,\\ \sum_{i=1}^{n}x_{ij}=1,\quad j=2,\cdots,n,\\ u_1=0,\quad 1\leq u_i\leq n-1,\quad i=2,3,\cdots,n,\\ u_j\geq u_k+x_{kj}-(n-2)(1-x_{kj})+(n-3)x_{jk},\quad k=1,\cdots,n,\ j=2,\cdots,n,\\ x_{ij}=0\text{或}1,\quad i,j=1,2,\cdots,n.\end{cases} \tag{5}$$

例 5.4（续例 5.3）　利用数学规划模型(4)，(5)和 LINGO 软件求解**例 5.3** 的问题。

求最小生成树的 LINGO 程序如下（用 LINGO 计算时，顶点 $v_0,v_1,\cdots,v_8$ 分别编号为 $1,2,\cdots,9$）：

```
model:
sets:
vertex/1..9/:u;
edge(vertex,vertex):w,x;
endsets
data:
w=10000;  !初始化，每个元素取充分大的正数;
enddata
calc:
w(1,2)=2; w(1,3)=1; w(1,4)=3; w(1,5)=4; w(1,6)=4; w(1,7)=2;
w(1,8)=5; w(1,9)=4; w(2,3)=4; w(2,9)=1; w(3,4)=1; w(4,5)=1;
w(5,6)=5; w(6,7)=2; w(7,8)=3; w(8,9)=5;
n=@size(vertex);
@for(vertex(j)|j#lt#n:@for(vertex(i)|i#gt#j:w(i,j)=w(j,i)));
```

```
endcalc
min=@sum(edge(i,j):w(i,j)*x(i,j));
@sum(vertex(j):x(1,j))>=1;
@for(vertex(j)|j#ge#2:@sum(vertex(i):x(i,j))=1);
@for(edge(i,j):@bin(x(i,j)));
u(1)=0;
@for(vertex(i)|i#ge#2:u(i)>=1; u(i)<=n-1);
@for(vertex(k):@for(vertex(j)|j#ge#2 :
u(j)>=u(k)+x(k,j)-(n-2)*(1-x(k,j))+(n-3)*x(j,k)));
end
```

5.4 最短路算法

求最短路的算法有 Dijkstra 标号算法和 Floyd 算法等方法。Dijkstra 标号算法只适用于权重是非负的情形。

5.4.1 Dijkstra 算法

给定赋权图 $G=(V,E,W)$，其中 $V=\{v_1,\cdots,v_n\}$ 为顶点集合，E 为边的集合，邻接矩阵 $W=(w_{ij})_{n\times n}$，求顶点 u_0 到 v_0 的最短距离 $d(u_0,v_0)$。记 $l(v_i)$ 表示顶点 v_i 的标号值。

Dijkstra 标号算法的计算步骤如下：

（1）令 $l(u_0)=0$，对 $v\neq u_0$，令 $l(v)=\infty$，$S_0=\{u_0\}$，$i=0$。

（2）对每个 $v\in\overline{S}_i$（$\overline{S}_i=V\setminus S_i$），用

$$\min_{u\in S_i}\{l(v),l(u)+w(uv)\}$$

代替 $l(v)$，这里 $w(uv)$ 表示顶点 u 和 v 之间边的权值。计算 $\min\limits_{v\in\overline{S}_i}\{l(v)\}$，把达到这个最小值的一个顶点记为 u_{i+1}，令 $S_{i+1}=S_i\cup\{u_{i+1}\}$。

（3）若 $i=n-1$ 或 v_0 进入 S_i，算法终止；否则，用 $i+1$ 代替 i，转（2）。

5.4.2 Floyd 算法

对于赋权图 $G=(V,E,A_0)$，其中顶点集 $V=\{v_1,\cdots,v_n\}$，邻接矩阵

$$A_0=\begin{bmatrix}a_{11}&a_{12}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2n}\\\vdots&\vdots&\cdots&\vdots\\a_{n1}&a_{n2}&\cdots&a_{nn}\end{bmatrix},$$

这里

$$a_{ij}=\begin{cases}\text{权值，}\ \text{当}v_i\text{与}v_j\text{之间有边时，}\\ \infty，\ \text{当}v_i\text{与}v_j\text{之间无边时，}\end{cases}\quad(i\neq j)$$

$a_{ii}=0$，$i=1,2,\cdots,n$.

对于无向图，A_0 是对称矩阵，$a_{ij}=a_{ji}$，$i,j=1,2,\cdots,n$。

Floyd 算法的基本思想是递推产生一个矩阵序列 $A_1,\cdots,A_k,\cdots,A_n$，其中矩阵 A_k 的第 i 行第 j 列元素 $A_k(i,j)$ 表示从顶点 v_i 到顶点 v_j 的路径上所经过的顶点序号不大于 k 的最短路径长度。

计算时用迭代公式

$$A_k(i,j)=\min(A_{k-1}(i,j),A_{k-1}(i,k)+A_{k-1}(k,j))，$$

k 是迭代次数，$i,j,k=1,2,\cdots,n$。

最后，当 $k=n$ 时，A_n 即是各顶点之间的最短通路值。

5.4.3 最短路问题举例

1.无向图的最短路

例 5.5（续例 5.1） 对于例 5.1 所示的无向图。（1）求顶点 v_1 到顶点 v_5 的最短距离及最

短路径；（2）求顶点 v_2 到所有顶点的最短距离。

```
clc, clear
a=zeros(5); %邻接矩阵初始化
a(1,[2:5])=[9 2 4 7]; a(2,[3 4])=[3 4]; %输入邻接矩阵的上三角元素
a(3,[4 5])=[8 4]; a(4,5)=6;
a=a'; b=sparse(a)  %变成下三角矩阵，并转化为稀疏矩阵
[d1,path1]=graphshortestpath(b,1,5,'Directed',0) %注意要设置 Directed 属性值为 0 或 false
%下面标识出顶点 1 到 5 的最短路径
h=biograph(b,[],'ShowWeights','on','ShowArrows','off')  %生成图形对象
set(h,'LayoutType','equilibrium'); %设置属性:图形的布局是平衡的
set(h.Nodes(path1),'Color',[1 0.4 0.4])
fowEdges = getedgesbynodeid(h,get(h.Nodes(path1),'ID'));
revEdges = getedgesbynodeid(h,get(h.Nodes(fliplr(path1)),'ID'));
edges = [fowEdges;revEdges];
set(edges,'LineColor',[1 0 0])
set(edges,'LineWidth',1.5)
view(h) %画出最短路径的图形
[d2,path2]=graphshortestpath(b,2,[1:5],'Directed',0) %求顶点 2 都所有顶点的最短距离
```

求得的从 v_1 到 v_5 的最短距离为 6，最短路径是 $v_1 \to v_3 \to v_5$。

2.有向图的最短路径

例 5.6（续例 5.2）　在例 5.2 所示的有向图中，求 v_2 到 v_4 的最短距离。

```
clc, clear
a=zeros(6);
a(1,[2 3])=1; a(2,3)=1; a(3,[2 5])=1;
a(4,[2 6])=1; a(5,[2 4 6])=1; a(6,5)=1;
b=sparse(a);
[d,path]=graphshortestpath(b,2,4)  %默认为有向图，不需设置 Directed 属性
h=biograph(b,[],'ShowArrows','on')  %生成图形对象
set(h,'LayoutType','equilibrium'); %设置属性:图形的布局是平衡的
set(h.Nodes(path),'Color',[1 0.4 0.4])
edges = getedgesbynodeid(h,get(h.Nodes(path),'ID'));
set(edges,'LineColor',[1 0 0])
set(edges,'LineWidth',1.5)
view(h) %显示图形
```

求得的 v_2 到 v_4 的最短距离为 3，最短路径如图 5.5 所示。

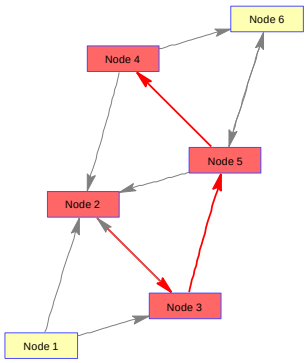

图 5.5　有向图的最短路径

例 5.7　设备更新问题。某企业使用一台设备，在每年年初，企业领导部门就要决定是购置新的，还是继续使用旧的。若购置新设备，就要支付一定的购置费用；若继续使用旧设备，则需支付更多的维修费用。现在的问题是如何制定一个几年之内的设备更新计划，使得

总的支付费用最少。我们用一个五年之内要更新某种设备的计划为例，若已知该种设备在各年年初的价格如**表** 5.2 所示，还已知使用不同时间（年）的设备所需要的维修费用如**表** 5.3 所示。如何制定总的支付费用最少的设备更新计划？

表 5.2　设备价格表

第 1 年	第 2 年	第 3 年	第 4 年	第 5 年
11	11	12	12	13

表 5.3　维修费用表

使用年限	0－1	1－2	2－3	3－4	4－5
维修费用	4	5	7	10	17

解　可以把这个问题化为图论中的最短路问题。

构造赋权有向图 $D=(V,A,W)$，其中顶点集 $V=\{v_1,v_2,\cdots,v_6\}$，这里 v_i（$i=1,\cdots,5$）表示第 i 年初的时刻，v_6 表示第 5 年末的时刻，A 为弧的集合，邻接矩阵 $W=(w_{ij})_{6\times6}$，这里 w_{ij} 表示时刻 v_i 购置新设备使用到时刻 v_j，购置新设备的费用和维修费用之和。则邻接矩阵

$$W=\begin{bmatrix} 0 & 15 & 20 & 27 & 37 & 54 \\ \infty & 0 & 15 & 20 & 27 & 37 \\ \infty & \infty & 0 & 16 & 21 & 28 \\ \infty & \infty & \infty & 0 & 16 & 21 \\ \infty & \infty & \infty & \infty & 0 & 17 \\ \infty & \infty & \infty & \infty & \infty & 0 \end{bmatrix}.$$

则制定总的支付费用最小的设备更新计划，就是在有向图 D 中求从 v_1 到 v_6 的费用最短路。

利用Dijkstra算法，使用MATLAB软件，求得 v_1 到 v_6 的最短路径为 $v_1\to v_3\to v_6$，最短路径的长度为48。设备更新最小费用路径见图5.6中的粗线所示，即设备更新计划为第1年初买进新设备，使用到第2年底，第3年初再购进新设备，使用到第5年底。

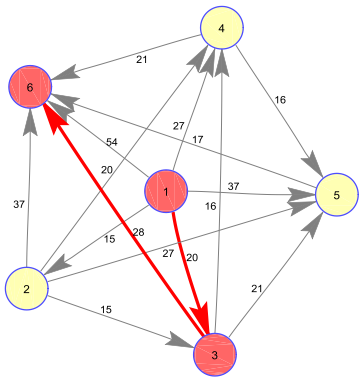

图 5.6　设备更新最小费用示意图

计算的MATLAB程序如下

```
clc, clear, close all
a=zeros(6); %邻接矩阵初始化
a(1,[2:6])=[15 20 27 37 54];
a(2,[3:6])=[15 20 27 37];
a(3,[4:6])=[16 21 28];
a(4,[5,6])=[16 21]; a(5,6)=17;   %输入有向图的邻接矩阵
b=sparse(a); %转化成稀疏矩阵
[d,path]=graphshortestpath(b,1,6) %默认为有向图，这里不需设置 Directed 属性
vname=cellstr(int2str([1:6]')); %构造顶点的字符，必须为细胞数组
```

```
h=biograph(b,vname,'ShowArrows','on','ShowWeights','on');  %生成图形对象
set(h.Nodes,'shape','circle'); %顶点画成圆形
set(h,'LayoutType','equilibrium'); %图形的布局是平衡的
set(h.Nodes(path),'Color',[1 0.4 0.4])
edges=getedgesbynodeid(h,get(h.Nodes(path),'ID'));
set(edges,'LineColor',[1 0 0])
set(edges,'LineWidth',2.5) %用粗线表示最短路径
view(h) %显示图形
```

3.所有顶点对之间的最短距离

下面举一个 Floyd 算法的例子，求所有顶点对之间的最短距离。

例 5.8　某连锁企业在某地区有 6 个销售点，已知该地区的交通网络如**图 5.7** 所示，其中点代表销售点，边表示公路，边上的权重为销售点间公路距离，问仓库应建在哪个小区，可使离仓库最远的销售点到仓库的路程最近？

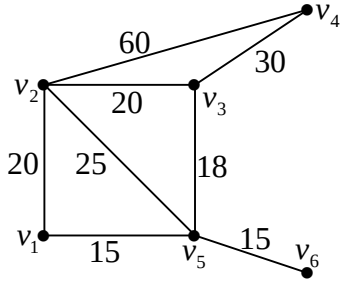

图 5.7　销售点之间距离

解　这是个选址问题，可以化为一系列求最短路问题。先求出 v_1 到所有各点的最短路长 d_{1j}，令 $D(v_1)=\max(d_{12},\cdots,d_{16})$，表示若仓库建在 v_1，则离仓库最远的销售点距离为 $D(v_1)$。再依次计算 $v_2,v_3,\cdots,v_6$ 到所有各点的最短距离，类似求出 $D(v_2),\cdots,D(v_6)$。$D(v_i)$（$i=1,\cdots,6$）中最小者即为所求，由上面的分析知，我们需要求所有的顶点对之间的最短距离，可以使用 Floyd 算法，用 MATLAB 软件的计算结果见表 5.4。

表 5.4　所有顶点对之间的最短距离

销售点	v_1	v_2	v_3	v_4	v_5	v_6	$D(v_i)$
v_1	0	20	33	63	15	30	63
v_2	20	0	20	50	25	40	50
v_3	33	20	0	30	18	33	33
v_4	63	50	30	0	48	63	63
v_5	15	25	18	48	0	15	48
v_6	30	40	33	63	15	0	63

由于 $D(v_3)=33$ 最小，所以仓库应建在 v_3，此时离仓库最远的销售点（v_1 和 v_6）距离为 33。

计算的 MATLAB 程序如下

```
clc, clear
a=zeros(6);
a(1,[2 5])=[20 15];
a(2,[3:5])=[20 60 25];
a(3,[4 5])=[30 18]; a(5,6)=15;
b=a'; b=sparse(b);
d=graphallshortestpaths(b,'Directed',0) %要设置 Directed 的属性值为 0 或 false
d1=max(d,[],2)   %逐行求最大值
[d2,ind]=min(d1) %求向量的最小值,及最小值的地址
v=find(d(ind,:)==d2)   %求向量中取值为 d2 的地址
```

5.4.4 最短路问题的 0-1 整数规划模型

下面我们以无向图为例来说明最短路的$0-1$整数规划模型，对有向图来说也是一样的。

对于给定的赋权图$G=(V,E,W)$，其中$V=\{v_1,\cdots,v_n\}$为顶点集合，E为边的集合，邻接矩阵$W=(w_{ij})_{n\times n}$，这里

$$w_{ij}=\begin{cases}v_i与v_j之间边的权值，\ 当v_i与v_j之间有边时，\\ \infty，\ 当v_i与v_j之间无边时,\end{cases}\quad（i,j=1,2,\cdots,n）.$$

现不妨求从v_1到$v_m(m\le n)$的最短路径。引进$0-1$变量

$$x_{ij}=\begin{cases}1,\quad 边(v_i,v_j)位于从v_1到v_m的最短路径上,\\ 0,\quad 否则,\end{cases}\quad（i,j=1,2,\cdots,n）.$$

于是最短路问题的数学模型为

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_{ij}\ ,\tag{6}$$

$$\text{s.t.}\begin{cases}\sum\limits_{j=1}^{n}x_{ij}=\sum\limits_{j=1}^{n}x_{ji},\quad i=2,3,\cdots,n且i\ne m,\\ \sum\limits_{j=1}^{n}x_{1j}=1,\\ \sum\limits_{j=1}^{n}x_{j1}=0,\\ \sum\limits_{j=1}^{n}x_{jm}=1,\\ x_{ij}=0或1,\quad i,j=1,2,\cdots,n.\end{cases}\tag{7}$$

这是一个$0-1$整数规划模型，可以直接用 LINGO 软件求解。

例 5.9　在图 5.8 中，求从v_2到v_4的最短路径和最短距离。

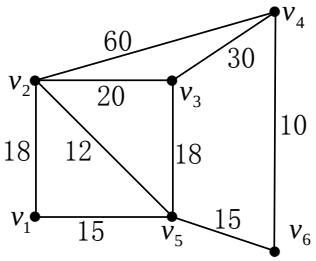

图 5.8　赋权无向图

解　用$G=(V,E,W)$表示图 5.8 所示的赋权无向图，其中$V=\{v_1,\cdots,v_6\}$为顶点集合，E为边的集合，邻接矩阵$W=(w_{ij})_{6\times 6}$，这里

$$w_{ij}=\begin{cases}v_i与v_j之间边的权值，\ 当v_i与v_j之间有边时，\\ \infty，\ 当v_i与v_j之间无边时,\end{cases}\quad（i,j=1,2,\cdots,6）.$$

引进$0-1$变量

$$x_{ij}=\begin{cases}1,\quad 边(v_i,v_j)位于从v_2到v_4的最短路径上,\\ 0,\quad 否则,\end{cases}\quad（i,j=1,2,\cdots,6）.$$

于是最短路问题的数学模型为

$$\min\sum_{i=1}^{6}\sum_{j=1}^{6}w_{ij}x_{ij}\ ,$$

$$\text{s.t.}\begin{cases}\sum_{j=1}^{6} x_{ij}=\sum_{j=1}^{6} x_{ji}, & i=1,2,3,\cdots,n\text{且}i\neq 2,4,\\ \sum_{j=1}^{n} x_{2j}=1,\\ \sum_{j=1}^{n} x_{j2}=0,\\ \sum_{j=1}^{n} x_{j4}=1,\\ x_{ij}=0\text{或}1, & i,j=1,2,\cdots,n.\end{cases}$$

其中的第 1 个约束条件表示对于非起点和终点的其他顶点，进入的边数等于出来的边数，第 2 个约束条件表示起点只能发出一条边，第 3 个约束条件表示起点不能进入边，第 4 个约束条件表示终点只能进入 1 条边。

利用 LINGO 软件求得的最短路径为 $v_2 \to v_5 \to v_6 \to v_4$，对应的最短距离为 37。

计算的 LINGO 程序如下：

```
model:
sets:
num/1..6/;
road(num,num):w,x;
endsets
data:
w=100000;  !邻接矩阵初始化;
enddata
calc: !以下先输入邻接矩阵的上三角元素;
w(1,2)=18; w(1,5)=15; w(2,3)=20; w(2,4)=60; w(2,5)=12;
w(3,4)=30; w(3,5)=18; w(4,6)=10; w(5,6)=15;
@for(num(j)|j#lt#@size(num):@for(num(i)|i#gt#j:w(i,j)=w(j,i))); !输入下三角元素;
endcalc
min=@sum(road(i,j):w(i,j)*x(i,j));
@for(num(i)|i#ne#2 #and# i#ne#4:@sum(num(j):x(i,j))=@sum(num(j):x(j,i)));
@sum(num(j):x(2,j))=1;   @sum(num(j):x(j,2))=0;
@sum(num(j):x(j,4))=1;
@for(road(i,j):@bin(x(i,j)));
end
```

5.5 最大流

5.5.1 有向图的最大流

定义 5.5 设有向连通图 $D=(V,A)$，G 的每条弧 (v_i,v_j) 上有非负数 c_{ij} 称为边的容量，仅有一个入次为 0 的点 v_s 称为发点（源），一个出次为 0 的点 v_t 称为收点（汇），其余点为中间点，这样的网络 D 称为容量网络，常记做 $D=(V,E,C)$。

对任一G中的弧(v_i,v_j)有流量f_{ij}，称集合$f=\{f_{ij}\}$为网络G上的一个流。称满足下列条件的流f为可行流：

（1）容量限制条件：对G中每条弧(v_i,v_j)，有$0\le f_{ij}\le c_{ij}$。

（2）平衡条件：对中间点v_i，有$\sum_j f_{ij}=\sum_k f_{ki}$，即物资的输入量与输出量相等。

对收、发点v_t,v_s，有$\sum_i f_{si}=\sum_j f_{jt}=v$，$v$为网络流的总流量。

可行流总是存在的，例如$f=\{0\}$就是一个流量为 0 的可行流。所谓最大流问题就是在容量网络中，寻找流量最大的可行流。

一个流$f=\{f_{ij}\}$，当$f_{ij}=c_{ij}$，则称流f对边(v_i,v_j)是饱和的，否则称f对(v_i,v_j)不饱和。

定义 5.6　容量网络D，若μ为网络中从v_s到v_t的一条链，给μ定向为从v_s到v_t，μ上的边凡与μ同向称为前向边，凡与μ反向称为后向边，其集合分别用μ^+和μ^-表示，f是一个可行流，如果满足

$$\begin{cases}0\le f_{ij}<c_{ij},(v_i,v_j)\in\mu^+,\\ c_{ij}\ge f_{ij}>0,(v_i,v_j)\in\mu^-.\end{cases}$$

则称μ为从v_s到v_t的（关于f的）可增广链。

下面给出求有向图最大流的标号算法。

设已有一个可行流f，算法可分为两步：第 1 步是标号过程，通过标号来寻找可增广链；第 2 步是调整过程，沿可增广链调整f以增加流量。

1.标号过程

（1）给发点以标号$(\Delta,+\infty)$。

（2）选择一个已标号的顶点v_i，对于v_i的所有未给标号的邻接点v_j按下列规则处理：

（a）若边$(v_j,v_i)\in E$，且$f_{ji}>0$，则令$\delta_j=\min(f_{ji},\delta_i)$，并给$v_j$以标号$(-v_i,\delta_j)$。

（b）若边$(v_i,v_j)\in E$，且$f_{ij}<c_{ij}$时，令$\delta_j=\min(c_{ij}-f_{ij},\delta_i)$，并给$v_j$以标号$(v_i,\delta_j)$。

（3）重复（2）直到收点v_t被标号或不再有顶点可标号时为止。

若v_t得到标号，说明存在一条可增广链，转（第 2 步）调整过程。若v_t未获得标号，标号过程已无法进行时，说明f已是最大流。

2.调整过程

（1）令$f_{ij}'=\begin{cases}f_{ij}+\delta_t,\text{若}(v_i,v_j)\text{是可增广链上的前向边}\\ f_{ij}-\delta_t,\text{若}(v_i,v_j)\text{是可增广链上的后向边}\\ f_{ij},\text{若}(v_i,v_j)\text{不在可增广链上}\end{cases}$

（2）去掉所有标号，回到第 1 步，对可行流f'重新标号。

例 5.10　求图 5.9 中从①到⑧的最大流。

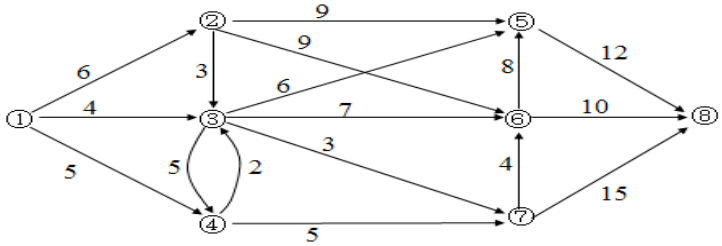

图 5.9　最大流问题的网络图

解　MATLAB 图论工具箱求解最大流的命令，只能解决权重都为正值，且两个顶点之间不能有两条弧的问题。图 5.9 中顶点 3，4 之间有两条弧，为此，在顶点 4 和顶点 3 之间加入一个虚拟的顶点 9，相当于把顶点 4 到顶点 3 的一条容量为 2 的弧，变成了顶点 4 到顶

点 9、顶点 9 到顶点 3 的容量都为 2 的两条弧。

利用 MATLAB 软件求得的最大流量是 15。求解的 MATLAB 程序如下

```
clc, clear, a=zeros(9);
a(1,2)=6; a(1,3)=4; a(1,4)=5;
a(2,3)=3; a(2,5)=9; a(2,6)=9;
a(3,4)=4; a(3,5)=6; a(3,6)=7; a(3,7)=3;
a(4,7)=5; a(4,9)=2;
a(5,8)=12; a(6,5)=8; a(6,8)=10;
a(7,6)=4; a(7,8)=15;   a(9,3)=2;
b=sparse(a);
[M,F]=graphmaxflow(b,1,8)   %求有向图的最大流
name=cellstr(int2str([1:9]'));   %构造顶点名称的细胞字符串数组
h=biograph(b,name,'ShowWeights','on') %生成图形对象
set(h,'EdgeType','segmented','LayoutType','equilibrium') %设置图形属性
view(h) %显示原有向图
h2=biograph(F,name,'ShowWeights','on') %生成图形对象
view(h2) %显示最大流对应的有向图
```

例5.9　有4 个公司来某重点高校招聘企业管理（A）、国际贸易（B）、管理信息系统（C）、工业工程（D）、市场营销（E）专业的本科毕业生。经本人报名和两轮筛选，最后可供选择的各专业毕业生人数分别为4，3，3，2，4人。若公司①想招聘A，B，C，D，E各专业毕业生各1人；公司②拟招聘4人，其中C，D专业各1人，A，B，E专业可从任两个专业中各选1人；公司③招聘4人，其中C，B，E专业各1人，再从A或D专业中选1人；公司④招聘3人，其中须有E专业1人，其余2人可从余下A，B，C，D专业中任选其中两个专业各1人。问上述4个公司是否都能招聘到各自需要的专业人才，并将此问题归结为求网络最大流问题。

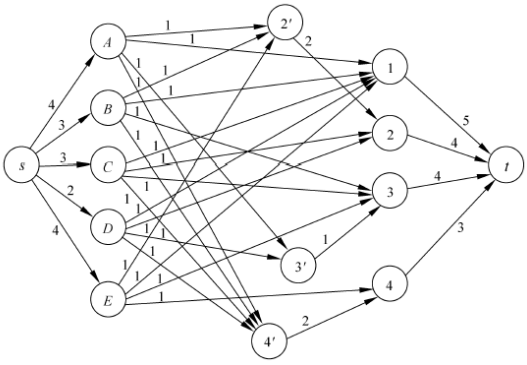

图 5.9　最大流网络

解一　前面有向图的最大流算法是针对单源和单汇的，而本题是多源多汇的，需要添加一个虚拟的源点 s，和一个虚拟的汇点 t，构造如图 5.9 所示的网络图，图中各弧旁数字为容量。把节点

$$s, A, B, C, D, E, 2', 3', 4', 1, 2, 3, 4, t$$

分别编号为1,2,…,14。然后求网络的最大流。

利用 MATLAB 求得最大流的流量为 16，即各公司都能招聘到所需人才。计算的 MATLAB 程序如下：

```
clc, clear
a=zeros(14); a(1,[2:6])=[4 3 3 2 4];
a(2,[7:10])=1; a(3,[7 9 10 12])=1;
a(4,[9:12])=1; a(5,[8:12])=1;
a(6,[7 10 12 13])=1; a(7,11)=2;
a(8,12)=1; a(9,13)=2;
a([10:13],14)=[5 4 4 3]; b=sparse(a);
```

```
[M,FlowMat]=graphmaxflow(b,1,14)
name=cellstr(int2str([1:14]'));
h=biograph(FlowMat,name)    %生成图形对象
set(h,'LayoutType','hierarchical','ShowWeights','on')    %设置图形属性
view(h) %显示图形
```

解二　我们也可以用 0-1 整数规划模型求解该问题，用 $i=1,2,3,4$ 分别表示 4 个公司，$j=1,2,\cdots,5$ 分别表示 5 个专业 A，B，C，D，E ，记第 j 个专业可供选择的毕业生人数为 a_j，引进 0-1 变量

$$x_{ij}=\begin{cases}1, & \text{第}i\text{个公司招聘第 }j\text{个专业的毕业生1名,}\\ 0, & \text{第}i\text{个公司不招聘第 }j\text{个专业的毕业生.}\end{cases}$$

建立如下的 0−1 整数规划模型

$$\max \sum_{i=1}^{4}\sum_{j=1}^{5}x_{ij}$$

$$\text{s.t.}\begin{cases}\sum\limits_{i=1}^{4}x_{ij}\le a_j,\ j=1,2,\cdots,5,\\ x_{21}+x_{22}+x_{25}\le 2,\\ x_{31}+x_{34}\le 1,\\ x_{41}+x_{42}+x_{43}+x_{44}\le 2.\\ x_{ij}=0\text{或}1,\ i=1,2,3,4,\ j=1,2,\cdots,5.\end{cases}$$

计算的 Lingo 程序如下

```
model:
sets:
com/1..4/:y; !每个公司招聘的人员总数;
stu/1..5/:a; !每个专业提供的毕业生人数;
link(com,stu):x; !0-1决策变量;
endsets
data:
a=4 3 3 2 4;
enddata
max=@sum(link:x);
@for(stu(j):@sum(com(i):x(i,j))<a(j));
x(2,1)+x(2,2)+x(2,5)<2;
x(3,1)+x(3,4)<1;
@sum(stu(j)|j#ne#5:x(4,j))<2;
@for(link:@bin(x));
@for(com(i):y(i)=@sum(stu(j):x(i,j)));
end
```

5.5.2　无向图的最大流

例 5.11　已知网络如图 5.10 所示，边上的权重表示容量，求该网络所有的顶点对之间的最大流量。

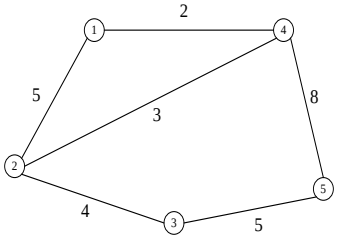

图 5.10　**网络图**

解　记图5.10所示的赋权图为$G=(V,E,C)$，其中顶点集合$V=\{1,2,3,4,5\}$，E为边的集合，邻接矩阵$C=(c_{ij})_{5\times5}$，这里c_{ij}表示顶点i和顶点j之间边的容量，当两个顶点之间无边时，相应的容量为0。设顶点i到顶点j的流为f_{ij}，流的起点为s，流的终点为t，从起点s到终点t的最大流的流量为v，建立如下最大流的线性规划模型

$$\max v$$

$$\text{s.t.}\begin{cases}\sum\limits_{j=1}^{5} f_{sj}=v,\\ \sum\limits_{i=1}^{5} f_{is}=0,\\ \sum\limits_{i=1}^{5} f_{it}=v,\\ \sum\limits_{i=1}^{5} f_{ik}=\sum\limits_{j=1}^{5} f_{kj},\ k\neq s,t,\\ 0\le f_{ij}\le c_{ij},\ i,j=1,2,\cdots,5.\end{cases}$$

其中第 2 个约束$\sum\limits_{i=1}^{5} f_{is}=0$是必须的，表示流不能再回流到起点。

求解的Lingo程序如下：

```
model:
sets:
node/1..5/;
arcs(node,node):c,f,tf;
endsets
data:
c=0; tf=0;
@text()=@table(tf); !输出到屏幕;
@text('zuida.txt')=@table(tf); !输出到纯文本文件;
enddata
calc:
c(1,2)=5; c(1,4)=2;
c (2,3)=4; c (2,4)=3;
c (3,5)=5; c (4,5)=8;
@for(arcs(i,j):c(i,j)=c(i,j)+c(j,i));
endcalc
submodel myflow:
[obj]max=v;
@for(node(i)|i #ne#s #and# i #ne#t:@sum(node(j):f(i,j))=@sum(node(k):f(k,i)));
@sum(node(j):f(s,j))=v; @sum(node(i):f(i,s))=0;
@sum(node(i):f(i,t))=v;
@for(arcs:@bnd(0,f,c));
endsubmodel
calc:
@for(node(i):@for(node(j)|j#gt#i:s=i; t=j; @solve(myflow);tf(s,t)=obj; tf(t,s)=obj;
@write(i,'　　',j,'　　',obj,@newline(1))));
@solve(myflow); !为了输出完整，这里进行了重复计算;
endcalc
end
```

5.6　旅行商问题

旅行商问题（Travel Salesman Problem，简记为 TSP）是指有一个旅行推销员想去若干城镇去推销商品，而每个城镇仅能经过一次，然后回到他的出发地。给定各城镇之间所需要的行走时间（或距离）后，那么该推销员应怎样安排他的行走路线，使他对每个城市恰好经

过一次的总时间（或距离）最短？

定义 5.7　包含 G 的每个顶点的轨叫做 Hamilton(哈密顿)轨；闭的 Hamilton 轨叫做 Hamilton 圈或 H 圈；含 Hamilton 圈的图叫做 Hamilton 图。

直观地讲，Hamilton 图就是从一顶点出发每顶点恰通过一次能回到出发点的那种图，即不重复地行遍所有的顶点再回到出发点。

TSP 模型是图论中的一个经典问题。用图论的语言描述就是，在赋权图中，寻找一条经过所有节点，并回到出发点的最短路，即可转化为寻找最优 Hamilton 圈问题。

TSP 模型是一个重要的组合优化问题，是 NP-难问题，至今还没有找到求解此问题的多项式时间算法。TSP 的近似算法有构造型算法和改进型算法，构造型算法按一定规则一次性地构造出一个解，而改进型算法则是以某一个解作为初始解，逐步迭代，使解得到改进。一般是先用构造型算法得到一个初始解，然后再用改进型算法逐步迭代。

近几十年来，TSP 模型有了基于智能算法的许多近似算法，如遗传算法、模拟退火算法、粒子群算法和神经元网络等算法。这些算法都有一定难度。下面我们把 TSP 模型转化为整数规划，然后用 LINGO 软件求解，该方法的优点是程序简洁、计算速度快、适用范围广。

5.6.1　TSP 模型的数学描述

对于给定的赋权图 $G=(V,E,\boldsymbol{W})$，其中 $V=\{v_1,v_2,\cdots,v_n\}$ 为顶点集，E 为边集，$W=(w_{ij})_{n\times n}$ 为邻接矩阵，这里

$$w_{ij}=\begin{cases} v_i\text{与}v_j\text{间的距离}, & \text{当}v_i\text{与}v_j\text{间存在边}, \\ \infty, & \text{当}v_i\text{与}v_j\text{间不存在边}, \end{cases} \quad (i\neq j),$$

$$w_{ii}=\infty,\quad i=1,2,\cdots,n.$$

引进 $0-1$ 变量

$$x_{ij}=\begin{cases} 1, & \text{当最短路径经过}v_i\text{到}v_j\text{的边时}, \\ 0, & \text{当最短路径不经过}v_i\text{到}v_j\text{的边时}, \end{cases} \quad i,j=1,2,\cdots,n.$$

则 TSP 模型可表示为：

$$\min \quad z=\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_{ij}\ , \tag{8}$$

$$\text{s.t.}\begin{cases} \sum\limits_{j=1}^{n}x_{ij}=1, \quad i=1,2,\cdots,n, \\ \sum\limits_{i=1}^{n}x_{ij}=1, \quad j=1,2,\cdots,n, \\ u_i-u_j+nx_{ij}\le n-1, u_i,u_j\ge 0, \quad i=1,\cdots,n,\ j=2,\cdots,n, \\ x_{ij}=0\text{或}1, \quad i,j=1,2,\cdots,n. \end{cases} \tag{9}$$

若仅考虑前两个约束条件，则是类似于指派问题的模型，对于 TSP 模型只是必要条件，并不充分。例如图 5.11 的情形，6 个城市的旅行路线若为 $v_1\to v_2\to v_3\to v_1$ 和 $v_4\to v_5\to v_6\to v_4$，则该路线虽然满足(9)式的前两个约束，但不构成整体巡回路线，它含有两个子回路，为此需要增加“不含子回路”的约束条件，这就要求增加变量 $u_i(i=1,2,\cdots,n)$，及(9)式中的第 3 个约束条件：

$$u_i-u_j+nx_{ij}\le n-1, u_i,u_j\ge 0, \quad i=1,\cdots,n,\ j=2,\cdots,n\ . \tag{10}$$

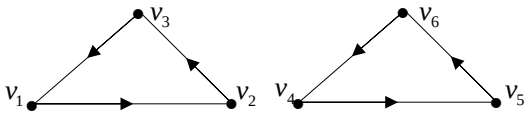

图 5.11　子回路情形

下面证明：

(1)任何含子回路的路线都必然不满足该约束条件（不管 u_i 如何取值）；

(2)全部不含子回路的整体巡回路线都可以满足该约束条件（只要 u_i 取适当值）。

用反证法证明 i)，假设存在子回路，则至少有两个子回路。那么至少有一个子回路中不含起点 v_1，例如子回路 $v_4 \to v_5 \to v_6 \to v_4$，式(10)用于该子回路，必有

$$u_4 - u_5 + n \le n-1,\quad u_5 - u_6 + n \le n-1,\quad u_6 - u_4 + n \le n-1,$$

把这三个不等式加起来得到 $0 \le -3$，这不可能，故假设不能成立。而对整体巡回，因为约束(10)式中 $j \ge 2$，不包含起点 v_1，故不会发生矛盾。

(3)对于整体巡回路线，只要 u_i 取适当值，都可以满足该约束条件：①对于总巡回上的边，$x_{ij}=1$，u_i 取整数：起点编号 $u_1=0$，第 1 个到达顶点的编号 $u_2=1$，每到达一个顶点，编号加 1，则必有 $u_i - u_j = -1$，约束条件(10)变成 $-1+n \le n-1$，必然成立。②对于非总巡回上的边，因为 $x_{ij}=0$，约束(10)变成：$u_i - u_j \le n-1$，肯定成立。

综上所述，约束条件(10)只限制子回路，不影响其他约束条件，于是 TSP 模型转化为一个整数线性规划模型，可以用 LINGO 软件求解。

5.6.2　TSP 模型的应用实例

例 5.12　已知 19 个城市之间距离示意图见图 5.12，求从 v_1 出发回到 v_1 的 TSP 路线。

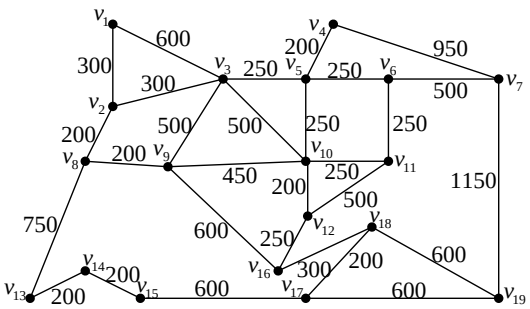

图 5.12　19 个城市间距离示意图

解　构造图 5.12 对应的赋权图 $G=(V,E,\boldsymbol{W})$，其中 $V=\{v_1,v_2,\cdots,v_{19}\}$ 为顶点集，E 为边集，$\boldsymbol{W}=(w_{ij})_{19\times 19}$ 为邻接矩阵，这里

$$w_{ij}=\begin{cases} v_i与v_j间的距离，& 当v_i与v_j间存在边时，\\ \infty，& 当v_i与v_j间不存在边时. \end{cases}\quad (i\ne j),$$

$$w_{ii}=\infty,\quad i=1,2,\cdots,19.$$

引进 0−1 变量

$$x_{ij}=\begin{cases} 1, & 当最短路径经过v_i到v_j的边时,\\ 0, & 当最短路径不经过v_i到v_j的边时, \end{cases}\quad i,j=1,2,\cdots,19.$$

则 TSP 模型可表示为：

$$\min \quad z=\sum_{i=1}^{19}\sum_{j=1}^{19} w_{ij}x_{ij}\ ,$$

$$\text{s.t.}\begin{cases}\sum\limits_{j=1}^{19} x_{ij}=1, \quad i=1,2,\cdots,19,\\ \sum\limits_{i=1}^{19} x_{ij}=1, \quad j=1,2,\cdots,19,\\ u_i-u_j+19x_{ij}\le 18, u_i,u_j\ge 0, \quad i=1,\cdots,19,\ j=2,\cdots,19,\\ x_{ij}=0\text{或}1, \quad i,j=1,2,\cdots,19.\end{cases}$$

利用 LINGO 软件求得的最优值 $z=8200$，最佳巡回路径为

$$v_1\to v_3\to v_9\to v_{10}\to v_5\to v_4\to v_7\to v_6\to v_{11}\to v_{12}\to v_{16}$$
$$\to v_{18}\to v_{19}\to v_{17}\to v_{15}\to v_{14}\to v_{13}\to v_8\to v_2\to v_1\ .$$

计算的 LINGO 程序如下：

```
model:
sets:
city/1..19/:u;
link(city,city):w,x;
endsets
data:
w=100000;
enddata
calc: !先输入邻接矩阵的上三角元素;
w(1,2)=300; w(1,3)=600; w(2,3)=300; w(2,8)=200; w(3,5)=250;
w(3,9)=500; w(3,10)=500; w(4,5)=200; w(4,7)=950; w(5,6)=250;
w(5,10)=250; w(6,7)=500; w(6,11)=250; w(7,19)=1150; w(8,9)=200;
w(8,13)=750; w(9,10)=450; w(9,16)=600; w(10,11)=250; w(10,12)=200;
w(11,12)=500; w(12,16)=250; w(13,14)=200; w(14,15)=200; w(15,17)=600;
w(16,18)=300; w(17,18)=200; w(17,19)=600; w(18,19)=600;
n=@size(city);
@for(city(j)|j#lt#n:@for(city(i)|i#gt#j:w(i,j)=w(j,i))); !输入邻接矩阵下三角元素;
endcalc
min=@sum(link(i,j):w(i,j)*x(i,j));
@for(city(i):@sum(city(j):x(i,j))=1);
@for(city(j):@sum(city(i):x(i,j))=1);
@for(city(i):@for(city(j)|j#ge#2:u(i)-u(j)+n*x(i,j)<n-1));
@for(link(i,j):@bin(x(i,j)));
end
```

5.7 关键路径

计划评审方法（program evaluation and review technique, PERT）和关键路线法（critical path method, CPM）是网络分析的重要组成部分，它广泛地用于系统分析和项目管理。计划评审与关键路线方法是在 20 世纪 50 年代提出并发展起来的，1956 年，美国杜邦公司为了协调企业不同业务部门的系统规划，提出了关键路线法。1958 年，美国海军武装部在研制“北极星”导弹计划时，由于导弹的研制系统过于庞大、复杂，为找到一种有效的管理方法，设计了计划评审方法。由于 PERT 与 CPM 既有着相同的目标应用，又有很多相同的术语，这两种方法已合并为一种方法，在国外称为 PERT/CPM，在国内称为统筹方法（scheduling

method）。

定义 5.8 称任何消耗时间或资源的行动称为作业。称作业的开始或结束为事件，事件本身不消耗资源。

在计划网络图中通常用圆圈表示事件，用箭线表示工作，如图 5.13 所示，1，2，3 表示事件，A,B 表示作业。由这种方法画出的网络图称为计划网络图。

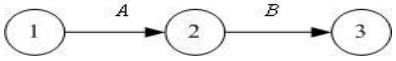

图 5.13 计划网络图的基本画法

虚作业用虚箭线“……→”表示。它表示工时为零，不消耗任何资源的虚构作业。其作用只是为了正确表示工作的前行后继关系。

定义 5.9 在计划网络图中，称从初始事件到最终事件的由各项工作连贯组成的一条路为路线。具有累计作业时间最长的路线称为关键路线。

建立计划网络图应注意的问题：

（1）任何作业在网络中用唯一的箭线表示，任何作业其终点事件的编号必须大于其起点事件。

（2）两个事件之间只能画一条箭线，表示一项作业。对于具有相同开始和结束事件的两项以上的作业，要引进虚事件和虚作业。

（3）任何计划网络图应有唯一的最初事件和唯一的最终事件。

（4）计划网络图不允许出现回路。

（5）计划网络图的画法一般是从左到右，从上到下，尽量作到清晰美观，避免箭头交叉。

例 5.13 某项目工程由 11 项作业组成（分别用代号 $A,B,\cdots,J,K$ 表示），其计划完成时间及作业间相互关系如表 5.5 所示，求作业的关键路径。

表 5.5 作业流程数据

作业	计划完成时间（天）	紧前作业	作业	计划完成时间（天）	紧前作业
A	5	—	G	21	B,E
B	10	—	H	35	B,E
C	11	—	I	25	B,E
D	4	B	J	15	F,G,I
E	4	A	K	20	F,G
F	15	C,D			

解 首先建立计划网络图，如图 5.14 所示。

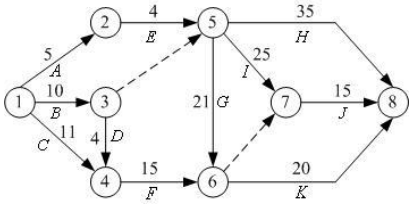

图 5.14 计划网络图

图 5.12 中的计划网络图就是一个有向图 $G=(V,\tilde{A},W)$，其中顶点集合 $V=\{1,\cdots,8\}$，$\tilde{A}$ 为弧的集合，邻接矩阵 $W=(w_{ij})_{8\times 8}$，其中

$$w_{ij}=\begin{cases} d_{ij}, & (ij)\in\tilde{A}, \\ -\infty, & (ij)\notin\tilde{A}, \end{cases} \quad i\neq j,$$

$$w_{ii}=0,\quad i=1,\cdots,8.$$

关键路径实际上是求从 1 到 8 的最长路径，网络模型中有很多求最短路径的算法，为了利用现有的最短路算法，必须把最长路径问题转化为最短路径问题，为此我们构造赋权有向图 $\bar{G}=\{V,\tilde{A},\bar{W}\}$，邻接矩阵 $\bar{W}=(\bar{w}_{ij})_{8\times 8}$，其中

$$\bar{w}_{ij}=\begin{cases}-d_{ij}, & (ij)\in\tilde{A},\\ \infty, & (ij)\notin\tilde{A},\end{cases} \quad i\neq j,$$

$$\bar{w}_{ii}=0, \quad i=1,\cdots,8.$$

这样我们只要求得赋权有向图 $\bar{G}$ 从 v_1 到 v_8 的最短路径，就等价地得到有向图 G 中的关键路径。

利用广度优先搜索算法，宽度优先搜索算法，或者 Floyd 算法，使用 MATLAB 软件可以求得关键路径为 1→3→5→6→8，关键路径的长度为 51。

计算的 MATLAB 程序如下

```
function main2
a=inf*ones(8);
a(1,2)=-5; a(1,3)=-10; a(1,4)=-11;
a(2,5)=-4; a(3,5)=0; a(3,4)=-4;
a(4,6)=-15; a(5,6)=-21; a(5,7)=-25; a(5,8)=-35;
a(6,7)=0; a(6,8)=-20; a(7,8)=-15;
[d,path]=myfloyd(a,1,8)   %调用下面的子函数
%**************************************************
%以下是我们编写的 Floyd 算法子函数，注意这里不能使用 MATLAB 工具箱
%**************************************************
function [dist,mypath]=myfloyd(a,sb,db);
% 输入：a—邻接矩阵，元素 a(i,j)是顶点 i 到 j 之间的直达距离，可以是有向的
% sb—起点的标号；db—终点的标号
% 输出：dist—最短路的距离；% mypath—最短路的路径
n=size(a,1); path=zeros(n);
for k=1:n
    for i=1:n
        for j=1:n
            if a(i,j)>a(i,k)+a(k,j)
                a(i,j)=a(i,k)+a(k,j);
                path(i,j)=k;
            end
        end
    end
end
dist=a(sb,db);
parent=path(sb,:); %从起点 sb 到终点 db 的最短路上各顶点的前驱顶点
parent(parent==0)=sb; %path 中的分量为 0，表示该顶点的前驱是起点
mypath=db; t=db;
while t~=sb
        p=parent(t); mypath=[p,mypath];
        t=p;
end
```

5.8 PageRank 算法

PageRank 算法是基于网页链接分析对关键字匹配搜索结果进行处理的。它借鉴传统引文分析思想：当网页甲有一个链接指向网页乙，就认为乙获得了甲对它贡献的分值，该值的多少取决于网页甲本身的重要程度，即网页甲的重要性越大，网页乙获得的贡献值就越高。由于网络中网页链接的相互指向，该分值的计算为一个迭代过程，最终网页根据所得分值进行检索排序。

互联网是一张有向图，每一个网页是图的一个顶点，网页间的每一个超链接是图的一个边，邻接矩阵 $B=(b_{ij})_{N\times N}$，如果从网页 i 到网页 j 有超链接，则 $b_{ij}=1$，否则为 0。

记矩阵 B 的行和为

$$r_i = \sum_{j=1}^{N} b_{ij} \ ,$$

它表示页面 i 发出的链接数目。

假如我们在上网时浏览页面并选择下一个页面的过程，与过去浏览过哪些页面无关，而仅依赖于当前所在的页面。那么这一选择过程可以认为是一个有限状态、离散时间的随机过程，其状态转移规律用 Markov 链描述。定义矩阵 $A=(a_{ij})_{N\times N}$ 如下

$$a_{ij} = \frac{1-d}{N} + d\frac{b_{ij}}{r_i}, \quad i,j=1,2,\cdots,N \ ,$$

其中 d 是模型参数，通常取 $d=0.85$，A 是 Markov 链的转移概率矩阵，a_{ij} 表示从页面 i 转移到页面 j 的概率。根据 Markov 链的基本性质，对于正则 Markov 链存在平稳分布 $x=[x_1,\cdots,x_N]^T$，满足

$$A^T x = x, \quad \sum_{i=1}^{N} x_i = 1,$$

x 表示在极限状态（转移次数趋于无限）下各网页被访问的概率分布，Google 将它定义为各网页的 PageRank 值。假设 x 已经得到，则它按分量满足方程

$$x_k = \sum_{i=1}^{N} a_{ik} x_i = (1-d) + d \sum_{i:b_{ik}=1} \frac{x_i}{r_i}.$$

网页 i 的 PageRank 值是 x_i，它链出的页面有 r_i 个，于是页面 i 将它的 PageRank 值分成 r_i 份，分别“投票”给它链出的网页。x_k 为网页 k 的 PageRank 值，即网络上所有页面“投票”给网页 k 的最终值。

根据 Markov 链的基本性质还可以得到，平稳分布（即 PageRank 值）是转移概率矩阵 A 的转置矩阵 A^T 的最大特征值（$=1$）所对应的归一化特征向量。

例 5.14　已知一个 $N=6$ 的网络如图 5.15 所示，求它的 PageRank 取值。

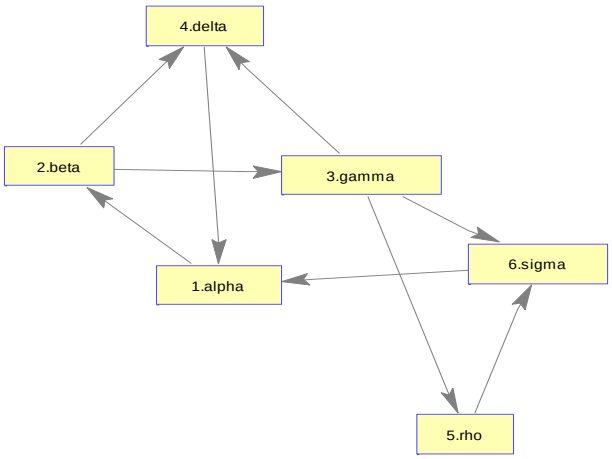

图 5.15　网络结构示意图

解　相应的邻接矩阵 B 和 Markov 链转移概率矩阵 A 分别为

$$B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$A=\begin{bmatrix} 0.025 & 0.875 & 0.025 & 0.025 & 0.025 & 0.025 \\ 0.025 & 0.025 & 0.45 & 0.45 & 0.025 & 0.025 \\ 0.025 & 0.025 & 0.025 & 0.3083 & 0.3083 & 0.3083 \\ 0.875 & 0.025 & 0.025 & 0.025 & 0.025 & 0.025 \\ 0.025 & 0.025 & 0.025 & 0.025 & 0.025 & 0.875 \\ 0.875 & 0.025 & 0.025 & 0.025 & 0.025 & 0.025 \end{bmatrix}$$

计算得到该 Markov 链的平稳分布为

$$x=\begin{bmatrix} 0.2675 & 0.2524 & 0.1323 & 0.1697 & 0.0625 & 0.1156 \end{bmatrix}^T.$$

这就是 6 个网页的 PageRank 值，其柱状图如图 5.16 所示。

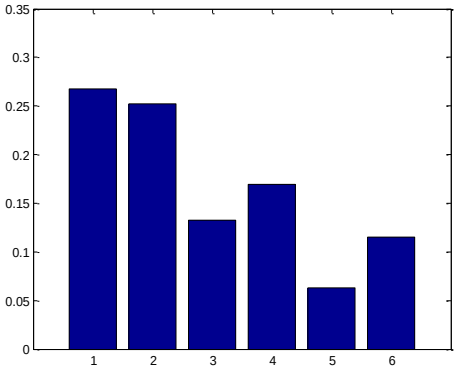

图 5.16　PageRank 值的柱状图

编号 1 的网页 alpha 的 PageRank 值最高，编号 5 的网页 rho 的 PageRank 值最低，网页的 PageRank 值从大到小的排序依次为 1,2,4,6,3,5。

计算的 MATLAB 程序如下

```
clc, clear
B=zeros(6);
B(1,2)=1; B(2,[3,4])=1;
B(3,[4:6])=1; B(4,1)=1;
B(5,6)=1; B(6,1)=1;
nodes={'1.alpha','2.beta','3.gamma','4.delta','5.rho','6.sigma'};
h=biograph(B,nodes,'ShowWeights','off','ShowArrows','on') %生成图形对象
set(h,'EdgeType','segmented','LayoutType','equilibrium'); %边的连接为线段,平衡布局
view(h) %显示图形
r=sum(B,2); n=length(B);
for i=1:n
    for j=1:n
        A(i,j)=0.15/6+0.85*B(i,j)/r(i);   %构造状态转移矩阵
    end
end
A %显示状态转移矩阵
[x,y]=eigs(A',1); %求最大特征值对应的特征向量
x=x/sum(x) %特征向量归一化
bar(x) %画 PageRank 值的柱状图
```

习题 5

5.1　用 MATLAB 分别画出下列图形：

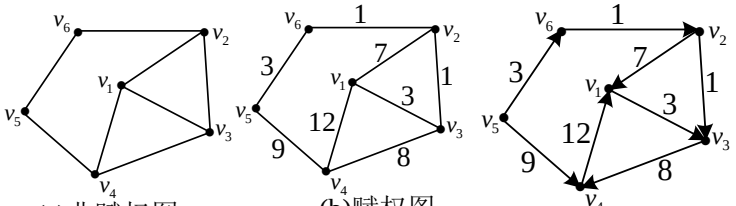

图 5.17　三种图

5.2　求图 5.18 所示赋权图的最小生成树。

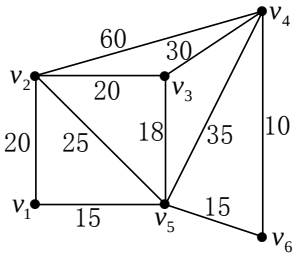

图 5.18　赋权无向图

5.3　在图 5.18 中求从 v_1 到 v_4 的最短路径和最短距离。

5.4　已知有 6 个村子，相互间道路的距离如图 5.19 所示。拟合建一所小学，已知 *A* 处有小学生 50 人，*B* 处 40 人，*C* 处 60 人，*D* 处 20 人，*E* 处 70 人，*F* 处 90 人。问小学应建在哪一个村庄，使学生上学最方便（走的总路程最短）。

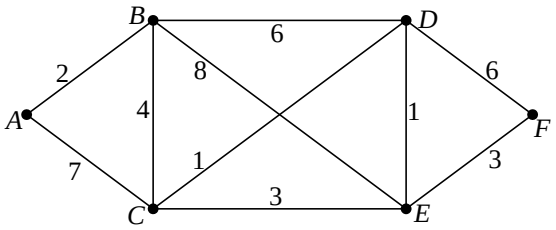

图 5.19　村庄之间道路示意图

5.5　在各种运动比赛中，为了使比赛公平、公正、合理地举行，一个基本要求是：在比赛项目排序过程中，尽可能使每个运动员不连续参加两项比赛，以便运动员恢复体力，发挥正常水平。

表 5.6 所示是某个小型运动会的比赛报名表。有 14 个比赛项目，40 名运动员参加比赛。表中第 1 行表示 14 个比赛项目，第 1 列表示 40 名运动员，表中“#”号位置表示运动员参加此项比赛。建立此问题的数学模型，并且合理安排比赛项目顺序，使连续参加两项比赛的运动员人次尽可能地少。

表 5.6　某小型运动会的比赛报名表

项目 / 运动员	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1		#	#						#				#	
2								#			#	#		
3		#		#						#				
4			#					#				#		
5											#		#	#
6					#	#								

7												#	#	
8										#				#
9		#		#						#	#			
10	#	#		#			#							
11		#		#									#	#
12								#		#				
13					#					#				#
14			#	#				#						
15			#					#				#		
16									#		#	#		
17						#								#
18							#					#		
19			#							#				
20	#		#											
21									#					#
22		#			#									
23							#					#		
24							#	#					#	#
25	#	#								#				
26					#									#
27						#					#			
28		#						#						
29	#										#	#		
30				#	#									
31						#		#				#		
32							#			#				
33				#		#								
34	#		#										#	#
35				#	#							#		
36				#			#							
37	#								#	#				
38						#		#		#				#
39					#			#	#				#	
40						#	#		#				#	

5.6　图 5.20 给出了 6 支球队的比赛结果，即 1 队战胜 2，4，5，6 队，而输给了 3 队；5 队战胜 3，6 队，而输给 1，2，4 队等等。

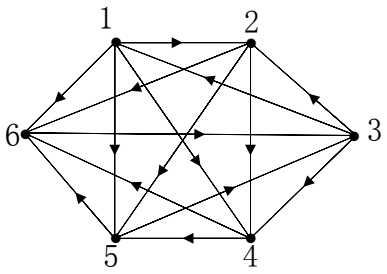

图 5.20　球队的比赛结果

（1）利用竞赛图的适当方法，给出 6 支球队的一个排名顺序；

（2）利用 PageRank 算法，再次给出 6 支球队的排名顺序。

5.7　已知 95 个目标点的数据见 Excel 文件 data1.xls，第 1 列是这 95 个点的编号，第 2,3 列是这 95 个点的 x, y 坐标，第 4 列是这些点重要性分类，标明“1”的是第一类重要目标点，标明“2”的是第二类重要目标点，未标明类别的是一般目标点，第 5，6，7 标明了这些点的连接关系。如第三行的数据

C　　-1160　　587.5　　D　　F

表示顶点 C 的坐标为 $(-1160,587.5)$，它是一般目标点，C 点和 D 点相连，C 点也和 F 点相连。

研究如下问题：

（1）画出上面的无向图，一类重要目标点用“五角星”画出，二类重要点用“*”画出，一般目标点用“.”画出。

要求必须画出无向图的度量图，顶点的位置坐标必须准确，不要画出无向图的拓扑图。

（2）当权重为距离时，求上面无向图的最小生成树，并画出最小生成树。

（3）求顶点 L 到顶点 M3 的最短距离及最短路径，并画出最短路径。