第 11 章　图论模型

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

图论是近三十年来发展非常活跃的一个数学分支。大量的最优化问题都可以抽象成网络模型结构来加以解释、描述和求解。它在建模时，具有直观、易理解、适应性强等特点，已广泛应用于管理科学、物理学、化学、计算机科学、信息论、控制论、社会科学（心理学、教育学等）以及军事科学等领域。一些实际网络，如运输网、电话网、电力网、计算机局域网等，都可以用图的理论加以描述和分析，并借助于计算机算法直接求解。这一理论与线性规划、整数规划等优化理论和方法相互渗透，促进了图论方法在实际问题建模中的应用。

本章主要介绍部分图论的基本概念，以及利用图论思想构建一些常用的模型和模型求解的方法。

11.1　图与网络的基础理论

11.1.1　图与网络的基本概念

所谓图，概况地讲就是由一些点和这些点之间的连线组成的。

1. 无向图和有向图

定义 11.1　一个无向图 G 是由非空顶点集 V 和边集 E 按一定的对应关系构成的连接结构，记为 $G=(V,E)$。其中非空集合 $V=\{v_1,v_2,\cdots,v_n\}$ 为 G 的顶点集，V 中的元素称为 G 的顶点，其元素的个数为顶点数；集合 $E=\{e_1,e_2,\cdots,e_m\}$ 为 G 的边集，E 中的元素称为 G 的边，其元素的个数为图 G 的边数。

以下用 $|V|$ 表示图 $G=(V,E)$ 中顶点的个数，$|E|$ 表示边的条数。

图 G 的每一条边是由连接 G 中两个顶点而得的一条线（可以是直线或曲线），因此与 G 的顶点对相对应，通常记作 $e_k=(v_i,v_j)$，其中，顶点 v_i,v_j 称为边 e_k 的两个端点，有时也说边 e_k 与顶点 v_i,v_j 关联。

对无向图来说，对应一条边的顶点对表示是无序的，即 (v_i,v_j) 和 (v_j,v_i) 表示同一条边 e_k。

有公共端点的两条边，或称邻边。同样，同一条边 e_k 的两个端点（v_i 和 v_j）称为是相邻的顶点。

带有方向的边称为有向边，又称为弧。如果给无向图的每条边规定一个方向，我们就得到有向图。

定义 11.2　有向图通常记为 $D=(V,A)$，其中非空集合 $V=\{v_1,v_2,\cdots,v_n\}$ 为 D 的顶点集，$A=\{a_1,a_2,\cdots,a_m\}$ 为 D 的弧集合，每一条弧与一个有序的顶点对相对应，弧 $a_k=(v_i,v_j)$ 表示弧的方向自顶点 v_i 指向 v_j，v_i 称为弧 a_k 的始端，v_j 称为弧 a_k 的末端或终端，其中 a_k 称为 v_i 的出弧，称为 v_j 的入弧。

与无向图不同，在有向图情形下，(v_i,v_j) 与 (v_j,v_i) 表示不同的弧。

把有向图 $D=(V,A)$ 中所有弧的方向都去掉，得到的边集用 E 表示，就得到与有向图 D 对应的无向图 $G=(V,E)$，称 G 为有向图 D 的基本图，称 D 为 G 的定向图。

例 11.1　设 $V=\{v_1,v_2,v_3,v_4,v_5\}$，$E=\{e_1,e_2,e_3,e_4,e_5\}$，其中

$$e_1=(v_1,v_2)，\ e_2=(v_2,v_3)，\ e_3=(v_2,v_3)，\ e_4=(v_3,v_4)，\ e_5=(v_4,v_4).$$

则 $G=(V,E)$ 是一个图，其图形如图 11.1 所示。

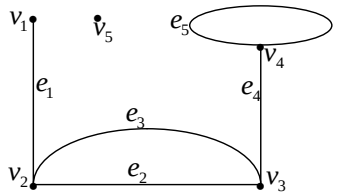


图 11.1　非简单图示例

2. 简单图、完全图、赋权图

定义 11.3　如果一条边的两个端点是同一个顶点，则称这条边为环。如果有两条边或多条边的端点是同一对顶点，则称这些边为重边或平行边。称不与任何边相关联的顶点为孤立点。

图 11.1 中，边 e_2 和 e_3 为重边，e_5 为环，顶点 v_5 为孤立点。

定义 11.4　无环且无重边的图称为简单图。

如果不特别申明，一般的图均指简单图。

图 11.1 不是简单图，因为图中既含有重边（e_2 和 e_3）又含环（e_5）。

定义 11.5　任意两顶点均相邻的简单图称为完全图。含 n 个顶点的完全图记为 K_n。

定义 11.6　如果图 G 的每条边 e 都附有一个实数 $w(e)$，则称图 G 为赋权图，实数 $w(e)$ 称为边 e 的权。

赋权图也称为网络。赋权图中的权可以是距离、费用、时间、效益、成本等。赋权图 G 一般记作 $G=(V,E,\boldsymbol{W})$，其中 $\boldsymbol{W}$ 为权重的邻接矩阵。赋权图也可以记作 $N=(V,E,\boldsymbol{W})$。

如果有向图 D 的每条弧都被赋予了权，则称 D 为有向赋权图。以后对于无向图、有向图或网络都可以用 G 表示，从下文中就能够区分出无向的还是有向的，赋权的还是非赋权的。

3. 顶点的度

定义 11.7　（1）在无向图中，与顶点 v 关联的边的数目（环算两次）称为 v 的度，记为 $d(v)$。

（2）在有向图中，从顶点 v 引出的弧的数目称为 v 的出度，记为 $d^+(v)$，从顶点 v 引入的弧的数目称为 v 的入度，记为 $d^-(v)$，$d(v)=d^+(v)+d^-(v)$ 称为 v 的度。

度为奇数的顶点称为奇顶点，度为偶数的顶点称为偶顶点。

定理 11.1　给定图 $G=(V,E)$，所有顶点的度数之和是边数的 2 倍，即

$$\sum_{v\in V} d(v)=2|E|.$$

推论 11.1　任何图中奇顶点的总数必为偶数。

4. 子图与图的连通性

定义 11.8　设 $G_1=(V_1,E_1)$ 与 $G_2=(V_2,E_2)$ 是两个图，并且满足 $V_1\subset V_2$，$E_1\subset E_2$，则称 G_1 是 G_2 的子图，G_2 称为 G_1 的母图。如 G_1 是 G_2 的子图，且 $V_1=V_2$，则称 G_1 是 G_2 的生成子图（支撑子图）。

定义 11.9　设 $W=v_0e_1v_1e_2\cdots e_kv_k$，其中 $e_i\in E(i=1,2,\cdots,k)$，$v_j\in V(j=0,1,\cdots,k)$，e_i 与 v_{i-1} 和 v_i 关联，称 W 是图 G 的一条道路（walk），简称路，k 为路长，v_0 为起点，v_k 为终点；各边相异的道路称为迹（trail）；各顶点相异的道路称为轨道（path），记为 $P(v_0,v_k)$；起点和终点重合的道路称为回路；起点和终点重合的轨道称为圈，即对轨道 $P(v_0,v_k)$，当 $v_0=v_k$ 时成为一个圈。

称以两顶点 u,v 分别为起点和终点的最短轨道之长为顶点 u,v 的距离。

定义 11.10　在无向图 G 中，如果从顶点 u 到顶点 v 存在道路，则称顶点 u 和 v 是连通的。

如果图G中的任意两个顶点u和v都是连通的，则称图G是连通图，否则称为非连通图。非连通图中的连通子图，称为连通分支。

在有向图D中，如果对于任意两个顶点u和v，从u到v和从v到u都存在道路，则称图D是强连通图。

11.1.2　图的矩阵表示

设图的顶点个数为n，边（或弧）的条数为m。

对于无向图$G=(V,E)$，其中$V=\{v_1,v_2,\cdots,v_n\}$，$E=\{e_1,e_2,\cdots,e_m\}$。

对于有向图$D=(V,A)$，其中$V=\{v_1,v_2,\cdots,v_n\}$，$A=\{a_1,a_2,\cdots,a_m\}$。

1. 关联矩阵

对于无向图G，其关联矩阵$\boldsymbol{M}=(m_{ij})_{n\times m}$，其中

$$m_{ij}=\begin{cases}1, & \text{顶点}v_i\text{与边}e_j\text{关联},\\ 0, & \text{顶点}v_i\text{与边}e_j\text{不关联},\end{cases}\quad i=1,2,\cdots,n,\quad j=1,2,\cdots,m.$$

对有向图G，其关联矩阵$\boldsymbol{M}=(m_{ij})_{n\times m}$，其中

$$m_{ij}=\begin{cases}1, & \text{顶点}v_i\text{是弧}a_j\text{的始端},\\ -1, & \text{顶点}v_i\text{是弧}a_j\text{的末端},\\ 0, & \text{顶点}v_i\text{与弧}a_j\text{不关联},\end{cases}\quad i=1,2,\cdots,n,\quad j=1,2,\cdots,m.$$

2. 邻接矩阵

对无向非赋权图G，其邻接矩阵$\boldsymbol{W}=(w_{ij})_{n\times n}$，其中

$$w_{ij}=\begin{cases}1, & \text{顶点}v_i\text{与}v_j\text{相邻},\\ 0, & i=j\text{或顶点}v_i\text{与}v_j\text{不相邻},\end{cases}\quad i,j=1,2,\cdots,n.$$

对有向非赋权图D，其邻接矩阵$\boldsymbol{W}=(w_{ij})_{n\times n}$，其中

$$w_{ij}=\begin{cases}1, & \text{弧}(v_i,v_j)\in A,\\ 0, & i=j\text{或顶点}v_i\text{到}v_j\text{无弧},\end{cases}\quad i,j=1,2,\cdots,n.$$

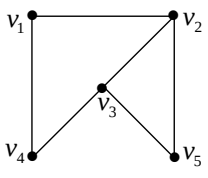
对无向赋权图G，其邻接矩阵$\boldsymbol{W}=(w_{ij})_{n\times n}$，其中

$$w_{ij}=\begin{cases}\text{顶点}v_i\text{与}v_j\text{之间边的权}, & (v_i,v_j)\in E,\\ 0(\text{或}\infty), & v_i\text{与}v_j\text{之间无边},\end{cases}\quad i,j=1,2,\cdots,n.$$

注 11.1　当两个顶点之间不存在边时，根据实际问题的含义或算法需要，对应的权可以取为 0 或∞，这里的邻接矩阵是数学上的邻接矩阵。

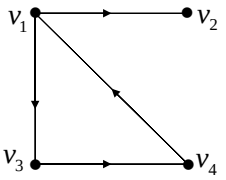
有向赋权图的邻接矩阵可类似定义。

例 11.2　非赋权无向图及邻接矩阵见图 11.2，非赋权有向图及邻接矩阵见**图** 11.3，赋权无向图及邻接矩阵见**图** 11.4。



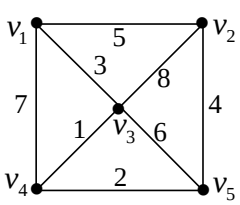
$$\begin{bmatrix}0&1&0&1&0\\1&0&1&0&1\\0&1&0&1&1\\1&0&1&0&0\\0&1&1&0&0\end{bmatrix}$$

图 11.2　**非赋权无向图及邻接矩阵**



$$\begin{bmatrix}0&1&1&0\\0&0&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

图 11.3　非赋权有向图及邻接矩阵



$$\begin{bmatrix} 0 & 5 & 3 & 7 & 0 \\ 5 & 0 & 8 & 0 & 4 \\ 3 & 8 & 0 & 1 & 6 \\ 7 & 0 & 1 & 0 & 2 \\ 0 & 4 & 6 & 2 & 0 \end{bmatrix}$$

图 11.4　赋权无向图及邻接矩阵

11.2　Matlab 工具箱简介

1.图的生成

graph：无向图（undirected Graph）；

digraph：有向图（directed Graph）；

```
G = graph                %创建空的无向图对象。
G = graph(A)             %使用邻接矩阵 A 创建赋权无向图。
G = graph(A,nodes)  %使用邻接矩阵 A 和节点名称 nodes 创建赋权无向图。
G = graph(s,t)           %使用节点对组 s,t 创建无向图。
G = graph(s,t,weights)   %使用节点对组 s,t 和权重向量 weights 创建赋权无向图。
G = graph(s,t,weights,nodes)  %使用字符向量元胞数组或字符串数组 nodes 指定节点名称。
G = graph(s,t,weights,num)    %使用数值标量 num 指定图中的节点数。
G = graph(A[,nodes],type)     %仅使用 A 的上或下三角形阵构造赋权图，type 可以是'upper' 或 'lower'。
```

例 11.3　画出一个 60 个顶点的 3 正则图（每个顶点的度都为 3）。

```
%程序文件 ex11_3.m
G = graph(bucky);
p = plot(G)
```

所画的图形如图 11.5 所示。

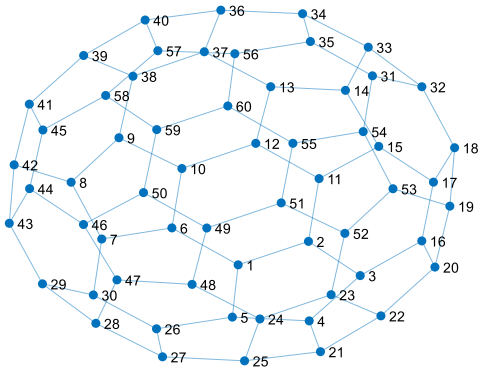


图 11.5　3 正则图

2. 数据存储结构

Matlab 存储网络的相关数据时，使用了稀疏矩阵，这有利于在存储大规模稀疏网络时节省存储空间。

例 11.4（续**例** 11.2）　画出**图** 11.3 的非赋权有向图并再次导出邻接矩阵和关联矩阵。

```
%程序文件 ex11_4.m
clc, clear, close all
a=zeros(4);              %邻接矩阵初始化
a([4,5,9,15])=1;   %一维索引赋值
nodes =cellstr(strcat('v',int2str([1:4]')))
G = digraph(a, nodes);
```

```
plot(G,'LineWidth',1.5,'Layout','circle','NodeFontSize',15)
W1 = adjacency(G)   %导出邻接矩阵的稀疏矩阵
W2 = incidence(G)   %导出关联矩阵的稀疏矩阵
```

所画的图如图 11.6 所示。

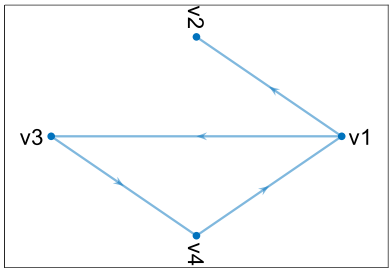


图 11.6　Matlab 画的有向图

例 11.5（**续例** 11.2）导出**图** 11.4 所示赋权图的邻接矩阵和关联矩阵，并重新画图。

```
%程序文件 ex11_5.m
clc, clear, close all
a=zeros(5); a(1,[2:4])=[5,3,7]; a(2,[3,5])=[8,4];
a(3,[4,5])=[1,6]; a(4,5)=2;   %输入邻接矩阵上三角元素
G=graph(a,'Upper')
W1 = adjacency(G,'weighted'), W2 = incidence(G)
plot(G,'Layout','force','EdgeLabel',G.Edges.Weight)
```

11.3　最小生成树

树（tree）是图论中非常重要的一类图，它非常类似于自然界中的树，结构简单、应用广泛，最小生成树问题则是其中的经典问题之一。在实际应用中，许多问题的图论模型都是最小生成树，如通信网络建设、有线电缆铺设、加工设备分组等。

11.3.1　基本概念和算法

1. 基本概念

定义 11.11　连通的无圈图称为树。

例如，图 11.7 给出的 G_1 是树，但 G_2 和 G_3 则不是树。

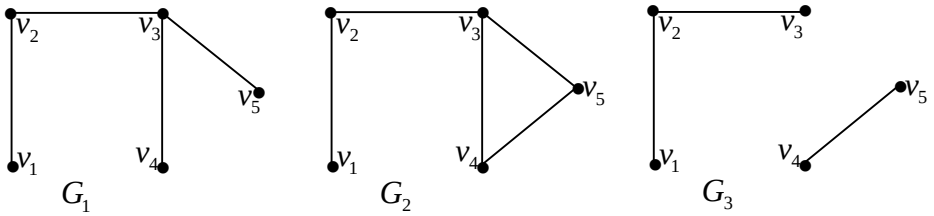


图 11.7　树与非树

定理 11.2　设 G 是具有 n 个顶点 m 条边的图，则以下命题等价。

（1）图 G 是树；

（2）图 G 中任意两个不同顶点之间存在唯一的路。

（3）图 G 连通，删除任一条边均不连通；

（4）图 G 连通，且 $n=m+1$；

（5）图 G 无圈，添加任一条边可得唯一的圈；

（6）图 G 无圈，且 $n=m+1$。

定义 11.12　若图 G 的生成子图 H 是树，则称 H 为 G 的生成树或支撑树。

一个图的生成树通常不唯一。

定理 11.3　连通图的生成树一定存在。

证明　给定连通图 G，若 G 无圈，则 G 本身就是自己的生成树。若 G 有圈，则任取 G 中一个圈 C，记删除 C 中一条边后所得之图为 G'。显然 G' 中圈 C 已经不存在，但 G' 仍然连

通。若G'中还有圈，再重复以上过程，直至得到一个无圈的连通图H。易知H是G的生成树。

定理 11.3 的证明方法也是求生成树的一种方法，称为“破圈法”。

定义 11.13　在赋权图G中，边权之和最小的生成树称为G的最小生成树。

一个简单连通图只要不是树，其生成树一般不唯一，而且非常多。一般地，n个顶点的完全图，其不同生成树的个数为n^{n-2}。因而，寻求一个给定赋权图的最小生成树，一般是不能用枚举法的。例如，20 个顶点的完全图有20^{18}个生成树，20^{18}有 24 位。所以，通过枚举求最小生成树是无效的算法，必须寻求有效的算法。

构造连通图最小生成树的算法有 Kruskal 算法和 Prim 算法。

对于赋权连通图$G=(V,E,\boldsymbol{W})$，其中V为顶点集合，E为边的集合，$\boldsymbol{W}$为邻接矩阵，这里顶点集合V中有n个顶点，下面构造它的最小生成树。

2.Kruskal 算法

Kruskal 算法思想：每次将一条权最小的边加入子图T中，并保证不形成圈。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

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

②e_{i+1}是$E-\{e_1,e_2,\cdots,e_i\}$中权值最小的边。

（3）直到选得e_{n-1}为止。

3.Prim 算法

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

4.最小生成树举例

例 11.6　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** 11.1。任意两个节点之间的距离为

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

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 11.1　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$W=(d_{ij})_{9\times9}$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$\{v_1v_2,v_2v_3,v_3v_4,v_4v_5,v_4v_6,v_6v_8,v_6v_9,v_7v_8\}$，电缆总长度的最小值为

86.4991。

计算的Matlab程序如下：

```
clc, clear, xy=load('data11_6.txt');
d=dist(xy);                                  %求 xy 的两两列向量间的欧氏距离
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,'XData',xy(1,:),'YData',xy(2,:));   %画无向图的度量图
highlight(h,T,'EdgeColor','r','LineWidth',2) %加粗最小生成树的边
```

11.3.2 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 Matlab 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 （充分大的正实数）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）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，

$$u_i-u_j+nx_{ij}\leq n-1,\quad i=1,\cdots,n,\ j=2,\cdots,n\ .$$

综上所述，最小生成树问题的 $0-1$ 整数规划模型如下：

$$\min\quad z=\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_{ij}\ , \tag{11.1}$$

$$\text{s.t.}\begin{cases}\sum_{j=1}^{n} x_{1j} \geq 1,\\ \sum_{i=1}^{n} x_{ij} = 1, \quad j = 2,\cdots,n,\\ u_1 = 0, \quad 1 \leq u_i \leq n-1, \quad i = 2,3,\cdots,n,\\ u_i - u_j + nx_{ij} \leq n-1, \quad i = 1,\cdots,n,\ j = 2,\cdots,n,\\ x_{ij} = 0\text{或}1, \quad i,\ j = 1,2,\cdots,n.\end{cases} \tag{11.2}$$

例 11.7（续**例** 11.6） 利用数学规划模型(11.1)-(11.2)求解**例** 11.6。

```
%程序文件 ex11_7.m
clc, clear, xy=load('data11_6.txt');
w=dist(xy);                                  %求 xy 的两两列向量间的欧氏距离
w(w==0) = 1000000;                           %这里 1000000 表示充分大的正实数
n=size(w,1); prob = optimproblem;
x = optimvar('x',n,n,'Type','integer','LowerBound',0,'UpperBound',1);
u = optimvar('u',n,'LowerBound',0);
prob.Objective = sum(sum(w.*x));
prob.Constraints.con1 = [sum(x(:,[2:n]))'==1; u(1)==0];
con2 = [1<=sum(x(1,:)); 1<=u(2:n); u(2:n)<=n-1];
for i = 1:n
    for j = 2:n
        con2=[con2; u(i)-u(j)+n*x(i,j)<=n-1];
    end
end
prob.Constraints.con2 = con2;
[sol,fval,flag,out] = solve(prob)
[i,j]=find(sol.x);
ind = [i'; j']   %输出树的顶点编号
```

11.4 最短路算法

最短路径问题是图论中非常经典的问题之一，旨在寻找图中两顶点之间的最短路径。作为一个基本工具，实际应用中的许多优化问题，如管道铺设、线路安排、厂区布局、设备更新等，都可被归结为最短路径问题来解决。

定义 11.14 设图G是赋权图，Γ为G中的一条路。则称Γ的各边权之和为路Γ的长度。

对于连通的赋权图G的两个顶点u_0和v_0，从u_0到v_0的路一般不止一条，其中最短的（长度最小的）一条称为从u_0到v_0的最短路；最短路的长称为从u_0到v_0的距离，记为$d(u_0,v_0)$。

求最短路的算法有 Dijkstra（迪克斯特拉）标号算法和 Floyd（弗洛伊德）算法等方法，但 Dijkstra 标号算法只适用于边权是非负的情形。最短路问题也可以归结为一个$0-1$整数规划模型。

11.4.1 固定起点的最短路

寻求从一固定起点u_0到其余各点的最短路，最有效的算法之一是 E. W. Dijkstra 于 1959 年提出的 Dijkstra 算法。这个算法是一种迭代算法，它的依据是一个重要而明显的性质：最短路是一条路，最短路上的任一子段也是最短路。

对于给定的赋权无向图或有向图$G=(V,E,\boldsymbol{W})$，其中$V=\{v_1,\cdots,v_n\}$为顶点集合，E为边（或弧）的集合，邻接矩阵$\boldsymbol{W}=(w_{ij})_{n\times n}$，这里

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

$w_{ii}=0$， $i=1,2,\cdots,n$.

u_0 为 V 中的某个固定起点，求顶点 u_0 到 V 中另一顶点 v_0 的最短距离 $d(u_0,v_0)$，即为求 u_0 到 v_0 的最短路。

Dijkstra 算法的基本思想是：按距固定起点 u_0 从近到远为顺序，依次求得 u_0 到图 G 各顶点的最短路和距离，直至某个顶点 v_0（或直至图 G 的所有顶点）。

为避免重复并保留每一步的计算信息，对于任意顶点 $v\in V$，定义两个标号：

$l(v)$：顶点 v 的标号，表示从起点 u_0 到 v 的当前路的长度；

$z(v)$：顶点 v 的父顶点标号，用以确定最短路的路线。

另外用 S_i 表示具有永久标号的顶点集。Dijkstra 标号算法的计算步骤如下：

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

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

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

这里 $w(uv)$ 表示顶点 u 和 v 之间边的权值，如果此次迭代利用顶点 $\tilde{u}$ 修改了顶点 v 的标号值 $l(v)$，则 $z(v)=\tilde{u}$，否则 $z(v)$ 不变。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 $S_{i+1}=S_i\bigcup\{u_{i+1}\}$。

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

算法结束时，从 u_0 到各顶点 v 的距离由 v 的最后一次标号 $l(v)$ 给出。在 v 进入 S_i 之前的标号 $l(v)$ 叫 T 标号，v 进入 S_i 时的标号 $l(v)$ 叫 P 标号。算法就是不断修改各顶点的 T 标号，直至获得 P 标号。若在算法运行过程中，将每一顶点获得 P 标号所由来的边在图上标明，则算法结束时，u_0 至各顶点的最短路也在图上标示出来了。

例 11.8　现有一只装满 12 斤酒的瓶子和三只分别装 10 斤、6 斤和 3 斤酒的空瓶，如何才能将这 12 斤酒分成三等份。此问题即为泊松分酒问题。

解　在分酒过程中，四个瓶中酒量分别用 x_1,x_2,x_3,x_4 表示（依照瓶子容量从大到小排列），并称向量 $[x_1,x_2,x_3,x_4]$ 为一个状态。每经过一步倒酒操作（装满一个瓶子或倒空一个瓶子），状态就会发生变化。若 $[x_1,x_2,x_3,x_4]$ 为一个状态，则它满足下列条件：

（1）$0\le x_1\le 12$，$0\le x_2\le 10$，$0\le x_3\le 6$，$0\le x_4\le 3$，且 x_i 为整数；

（2）$x_1+x_2+x_3+x_4=12$；

（3）x_1,x_2,x_3,x_4 中至少有一个变量取得最大值（装满）或最小值（倒空）。

将第 i 个状态看作一个顶点 v_i，构造赋权有向图 $D=(V,A,\boldsymbol{W})$，其中顶点集 $V=\{v_1,v_2,\cdots,v_n\}$ 为所有的状态，通过枚举法得到状态总数 $n=169$；A 为弧的集合，当两个状态 v_i,v_j 有两个分量不同时，且 v_j 的改变分量中至少有一个达到最大值或最小值，则 v_i 到 v_j 存在弧；邻接矩阵 $\boldsymbol{W}=(w_{ij})_{n\times n}$，其中

$$w_{ij}=\begin{cases}1,\ & v_i到v_j间存在弧，\\ 0,\ & 否则.\end{cases}$$

将 12 斤酒三等分等价于求由初始状态 $[12,0,0,0]$ 到终止状态 $[4,4,4,0]$ 的一条最短路径。

利用 Matlab 求得的状态遍历顺序如下：

12	0	0	0
2	10	0	0
2	4	6	0
8	4	0	0
8	1	0	3
11	1	0	0

```
    11     0     0     1
     1    10     0     1
     1     4     6     1
     1     4     4     3
     4     4     4     0
```

即经过 10 步操作可完成酒的等分。当然，操作过程不是唯一的。

计算的 Matlab 程序如下：

```
clc, clear
sm=[12,10,6,3]; %四个瓶子容量
s=[];           %状态集合初始化
for x4=0:3
    for x3=0:6
        for x2=0:10
            t=[12-x4-x3-x2,x2,x3,x4];
            if t(1)>=0 & (~all(t) | ~all(sm-t))
                s=[s; t];
            end
        end
    end
end
s, n=size(s,1)   %显示状态集合和状态个数
w=zeros(n);    %邻接矩阵初始化
for i=1:n
    for j=1:n
        vi=s(i,:); vj=s(j,:);
        ind=find(vi-vj);   %查找改变分量的位置
        if length(ind)==2 & (~all(vj(ind)) | ~all(sm(ind)-vj(ind)))
            w(i,j)=1;
        end
    end
end
G=digraph(w);              %构造赋权有向图
m=find(ismember(s,[4,4,4,0],'rows'))
[path,d]=shortestpath(G,1,m) %求有向图的最短路径和距离
ps=s(path,:)                 %显示最短路径遍历的状态
```

注 11.2　在利用 Matlab 工具箱计算时，如果两个顶点之间没有边，对应的邻接矩阵元素为 0，而不是像数学理论上对应的邻接矩阵元素为 $+\infty$。下面同样约定算法上的数学邻接矩阵和 Matlab 工具箱调用时的邻接矩阵是不同的。

11.4.2　所有顶点对之间最短路的 Floyd 算法

利用 Dijkstra 算法，当然还可以寻求赋权图中所有顶点对之间最短路。具体方法是：每次以不同的顶点作为起点，用 Dijkstra 算法求出从该起点到其余顶点的最短路径，反复执行 $n-1$（n 为顶点个数）次这样的操作，就可得到每对顶点之间的最短路。但这样做需要大量的重复计算，效率不高。为此，R. W. Floyd 另辟蹊径，于 1962 年提出了一个直接寻求任意两顶点之间最短路的算法。

Floyd 算法允许赋权图中包含负权的边或弧，但是，对于赋权图中的每个圈 C，要求圈 C 上所有弧的权总和为非负。而 Dijkstra 算法要求所有边或弧的权都是非负的。Floyd 算法包含三个关键算法：求距离矩阵、求路径矩阵、最短路查找算法。

设所考虑的赋权图 $G=(V,E,\boldsymbol{A}_0)$，其中顶点集 $V=\{v_1,\cdots,v_n\}$，邻接矩阵

$$\boldsymbol{A}_0=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},$$

这里

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

$a_{ii}=0$， $i=1,2,\cdots,n$.

对于无向图， $\boldsymbol{A}_0$ 是对称矩阵， $a_{ij}=a_{ji}$， $i,j=1,2,\cdots,n$。

1.求距离矩阵的算法

通常所说的 Floyd 算法，一般是指求距离矩阵的算法，实际是一个经典的动态规划算法，其基本思想是递推产生一个矩阵序列 $\boldsymbol{A}_1,\boldsymbol{A}_2,\cdots,\boldsymbol{A}_k,\cdots,\boldsymbol{A}_n$， 其中矩阵 $\boldsymbol{A}_k=(a_k(i,j))_{n\times n}$， 其第 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距离值。

2.建立路径矩阵的算法

如果在求得两点间的最短距离时，还需要求得两点间的最短路径，需要在上面距离矩阵 $\boldsymbol{A}_k$ 的迭代过程中，引入一个路由矩阵 $\boldsymbol{R}_k=(r_k(i,j))_{n\times n}$ 来记录两点间路径的前驱后继关系，其中 $r_k(i,j)$ 表示从顶点 v_i 到顶点 v_j 的路径经过编号为 $r_k(i,j)$ 的顶点。

路径矩阵的迭代过程如下：

（1）初始时

$$\boldsymbol{R}_0=\boldsymbol{0}_{n\times n}.$$

（2）迭代公式为

$$\boldsymbol{R}_k=(r_k(i,j))_{n\times n},$$

其中

$$r_k(i,j)=\begin{cases} k, & \text{若}a_{k-1}(i,j)>a_{k-1}(i,k)+a_{k-1}(k,j), \\ r_{k-1}(i,j), & \text{否则}. \end{cases}$$

直到迭代到 $k=n$， 算法终止。

3.最短路的路径查找算法。

查找 v_i 到 v_j 最短路径的方法如下：

若 $r_n(i,j)=p_1$，则点 v_{p_1} 是顶点 v_i 到顶点 v_j 的最短路的中间点，然后用同样的方法再分头查找。若

（1）向顶点 v_i 反向追踪得： $r_n(i,p_1)=p_2$， $r_n(i,p_2)=p_3$， $\cdots$， $r_n(i,p_s)=0$；

（2）向顶点 v_j 正向追踪得： $r_n(p_1,j)=q_1$， $r_n(q_1,j)=q_2$， $\cdots$， $r_n(q_t,j)=0$；

则由点 v_i 到 v_j 的最短路径为： $v_i,v_{p_s},\cdots,v_{p_2},v_{p_1},v_{q_1},v_{q_2},\cdots,v_{q_t},v_j$。

综上所述，求距离矩阵 $\boldsymbol{D}=(d_{ij})_{n\times n}$ 和路径矩阵 $\boldsymbol{R}=(r_{ij})_{n\times n}$ 的 Floyd 算法如下：

第一步：初始化， $k=0$，对 $i,j=1,2,\cdots,n$， 令 $d_{ij}=a_{ij}$， $r_{ij}=0$。

第二步：迭代，$k=k+1$，对 $i,j=1,2,\cdots,n$，若 $d_{ij}>d_{ik}+d_{kj}$，则令 $d_{ij}=d_{ik}+d_{kj}$，$r_{ij}=k$；否则 d_{ij} 和 r_{ij} 不更新。

第三步：算法终止条件，如果 $k=n$， 算法终止；否则，转第二步。

Floyd 算法的时间复杂度为 $O(n^3)$， 空间复杂度为 $O(n^2)$。

例 11.9（续例 11.2） 求图 11.4 所示赋权无向图中所有顶点对之间的最短距离。

解 使用 Floyd 算法，求得所有顶点对之间的最短距离矩阵为

$$\begin{bmatrix} 0 & 5 & 3 & 4 & 6 \\ 5 & 0 & 7 & 6 & 4 \\ 3 & 7 & 0 & 1 & 3 \\ 4 & 6 & 1 & 0 & 2 \\ 6 & 4 & 3 & 2 & 0 \end{bmatrix}.$$

```
%程序文件 ex11_9_1.m
clc, clear, a = zeros(5);
a(1,[2:4])=[5,3,7];  %输入邻接矩阵的上三角元素
a(2,[3,5])=[8,4]; a(3,[4,5])=[1,6]; a(4,5)=2;
n=length(a); b=a+a';  %构造完整的邻接矩阵
b(b==0)=inf; %把零元素换成 inf
b(1:n+1:end)=0; %把对角线元素换成 0
for k=1:n
    for i=1:n
        for j=1:n
            if b(i,k)+b(k,j)<b(i,j)
                b(i,j)=b(i,k)+b(k,j);
            end
        end
    end
end
b  %显示两两顶点之间的最短距离
```

直积利用 Matlab 工具箱函数 distances，使用 Dijkstra 算法求解的 Matlab 程序如下：

```
%程序文件 ex11_9_2.m
clc, clear
E=[1,2,5;1,3,3;1,4,7;2,3,8;2,5,4
    3,4,1;3,5,6;4,5,2];          %逐行输入每条边的数据
G=graph(E(:,1),E(:,2),E(:,3))  %构造图的另一种方式
d=distances(G)
```

11.4.3 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 $G=(V,E,\boldsymbol{W})$，其中 $V=\{v_1,\cdots,v_n\}$ 为顶点集合，E 为边的集合，邻接矩阵 $\boldsymbol{W}=(w_{ij})_{n\times n}$，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

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

于是最短路问题的数学模型为

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}x_{ij}\ ,\tag{11.3}$$

$$\text{s.t.}\begin{cases}\sum\limits_{j=1}^{n}x_{ij}=\sum\limits_{j=1}^{n}x_{ji}, \quad i=2,3,\cdots,n且i\neq m,\\ \sum\limits_{j=1}^{n}x_{1j}=1,\\ \sum\limits_{j=1}^{n}x_{j1}=0,\\ \sum\limits_{j=1}^{n}x_{jm}=1,\\ x_{ij}=0或1, \quad i,j=1,2,\cdots,n.\end{cases} \tag{11.4}$$

这是一个$0-1$整数规划模型。

例 11.10　在**图** 11.8 中，求从v_2到v_4的最短路径和最短距离。

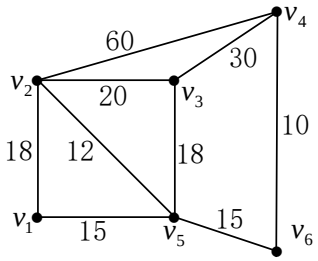


图 11.8　**赋权无向图**

解　用$G=(V,E,W)$表示图 11.8 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$W=(w_{ij})_{6\times6}$，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

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

$$\text{s.t.}\begin{cases}\sum\limits_{j=1}^{6}x_{ij}=\sum\limits_{j=1}^{6}x_{ji}, \quad i=1,3,5,6,\\ \sum\limits_{j=1}^{n}x_{2j}=1,\\ \sum\limits_{j=1}^{n}x_{j2}=0,\\ \sum\limits_{j=1}^{n}x_{j4}=1,\\ x_{ij}=0或1, \quad i,j=1,2,\cdots,n.\end{cases}$$

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 Matlab 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。

```
%程序文件 ex6_10.m
clc, clear, a = zeros(6);
```

```
a(1,[2,5])=[18,15]; a(2,[3:5])=[20,60,12];
a(3,[4,5])=[30,18]; a(4,6)=10; a(5,6)=15;
b = a+a';                          %输入完整的邻接矩阵
b(b==0)=1000000;                   %这里 1000000 表示充分大的正实数
prob = optimproblem;
x = optimvar('x',6,6,'Type','integer','LowerBound',0,'UpperBound',1);
prob.Objective = sum(sum(b.*x));
s=setdiff([1:6],[2,4])             %非起点和终点的集合
prob.Constraints=[sum(x(s,:),2)==sum(x(:,s))'; sum(x(2,:))==1
    sum(x(:,2))==0; sum(x(:,4))==1];
[sol,fval,flag,out]= solve(prob), xx=sol.x
[i,j] = find(xx);                  %找非零元素的行标和列标
ij = [i'; j']
```

11.5 钢管订购和运输

11.5.1 问题描述

要铺设一条 $A_1 \to A_2 \to \cdots \to A_{15}$ 的输送天然气的主管道，如图 6.1 所示。经筛选后可以生产这种主管道钢管的钢厂有 $S_1, S_2, \cdots S_7$。图中粗线表示铁路，单细线表示公路，双细线表示要铺设的管道(假设沿管道或者原来有公路，或者建有施工公路)，圆圈表示火车站，每段铁路、公路和管道旁的阿拉伯数字表示里程(单位 km)。

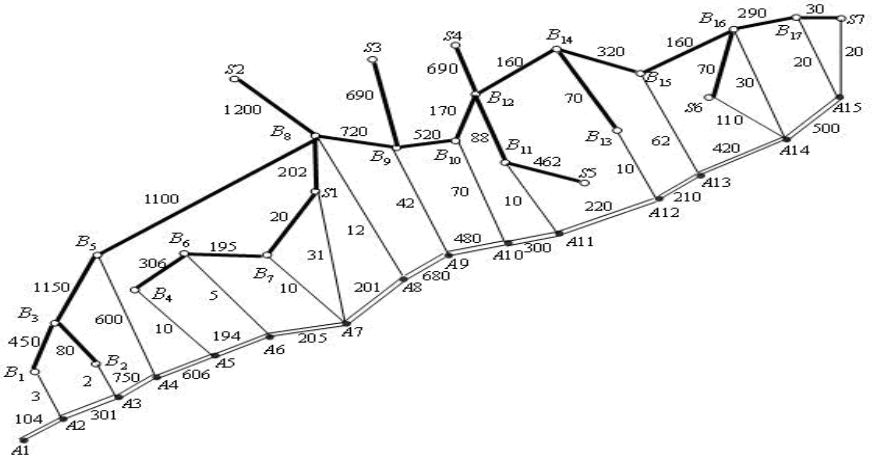


图 6.1 交通网络及管道图

为方便计，1km 主管道钢管称为 1 单位钢管。

一个钢厂如果承担制造这种钢管，至少需要生产 500 个单位。钢厂 S_i 在指定期限内能生产该钢管的最大数量为 s_i 个单位，钢管出厂销价 1 单位钢管为 p_i 万元，见表 6.1；1 单位钢管的铁路运价见表 6.2。

表 6.1 各钢管厂的供货上限及销价

i	1	2	3	4	5	6	7
s_i	800	800	1000	2000	2000	2000	3000
p_i	160	155	155	160	155	150	160

表 6.2 单位钢管的铁路运价

里程(km)	≤300	301～350	351～400	401～450	451～500
运价(万元)	20	23	26	29	32
里程(km)	501～600	601～700	701～800	801～900	901～1000

运价(万元)	37	44	50	55	60

1000km 以上每增加 1 至 100km 运价增加 5 万元。公路运输费用为 1 单位钢管每公里 0.1 万元（不足整公里部分按整公里计算）。钢管可由铁路、公路运往铺设地点（不只是运到点 $A_1,A_2,\cdots,A_{15}$，而是管道全线）。

请制定一个主管道钢管的订购和运输计划，使总费用最小（给出总费用）。

11.5.2 问题分析

问题的建模目的是求一个主管道钢管的订购和运输策略，使总费用最小。首先，问题给出了 7 个供选择的钢厂，选择哪些？订购多少？是一个要解决的问题。其次，每一个钢厂到铺设地点大多都有多条可供选择的运输路线，应选择哪一条路线运输，取决了建模的目标。而目标总费用包含两个组成部分：订购费用和运输费用。订购费用取决于单价和订购数量，运输费用取决于往哪里运和运输路线。结合总目标的要求，可以很容易地想到应选择运费最小的路线。

从同一个钢管厂订购钢管运往同一个目的地，一旦最小运输费用路线确定，则单位钢管的运费就确定了，单位钢管的总订购及运输费用=钢管单价+运输费用。因此，同一个钢管厂订购钢管运往同一个目的地的总费用等于订购数量乘以单位钢管的购运费用（单价+单位钢管运费）。因而，在制定订购与运输计划时，要分成两个子问题考虑：

（1）运输路线及运输费用的确定：钢管可以通过铁路或公路运输。公路运费是运输里程的线性函数，但是铁路运价却是一种分段的阶跃的常数函数。因此在计算时，不管对运输里程还是费用而言，都不具有可加性，只能将铁路运价（即由运输总里程找出对应费率）和公路运价分别计算后再叠加。

（2）铺设方案的设定：从钢管厂订购若干个单位的钢管运送至枢纽点 $A_1,A_2,\cdots,A_{15}$，再由枢纽点按一个单位计分别往枢纽点两侧运送至最终铺设地点，计算从枢纽点开始的铺设费用。

虽然准备把问题分解为两个子问题进行处理，但最终优化时，必须作为一个综合的优化问题进行处理，否则无法得到全局最优解。

11.5.3 模型的建立与求解

记第 $i(i=1,2,\cdots,7)$ 个钢厂的最大供应量为 s_i，从第 i 个钢厂到铺设节点 $j(j=1,2,\cdots,15)$ 的订购和运输费用为 c_{ij}；用 $l_k=\left|A_kA_{k+1}\right|(k=1,2,\cdots,14)$ 表示管道第 k 段需要铺设的钢管量。x_{ij} 是从钢厂 S_i 运到节点 j 的钢管量，y_j 是从节点 j 向左铺设的钢管量，z_j 是从节点 j 向右铺设的钢管量。

根据题中所给数据，可以先计算出从供应点 S_i 到需求点 A_j 的最小购运费 c_{ij}（即出厂售价与运输费用之和），再根据 c_{ij} 求解总费用，总费用应包括：订购费用（已包含在 c_{ij} 中），运输费用（由各厂 S_i 经铁路、公路至各点 A_j，$i=1,2,\cdots,7$，$j=1,2,\cdots,15$），铺设管道 $A_jA_{j+1}(j=1,2,\cdots,14)$ 的运费。

1. 运费矩阵的计算模型

购买单位钢管及从 S_i（$i=1,2,\cdots,7$）运送到 A_j（$j=1,2,\cdots,15$）的最小购运费用 c_{ij} 的计算如下。

1）计算铁路任意两点间的最小运输费用

由于铁路运费不是连续的，故不能直接构造铁路费用赋权图，用 Floyd 算法来计算任意两点间的最小运输费用。但可以首先构造铁路距离赋权图，用 Floyd 算法来计算任意两点间的最短铁路距离值，再依据题中的铁路运价表，求出任意两点间的最小铁路运输费用。这就巧妙地避开铁路运费不是连续的问题。

首先构造铁路距离赋权图 $G_1=(V,E_1,\boldsymbol{W}_1)$，其中

$V=\{S_1,\cdots,S_7,A_1,\cdots,A_{15},B_1,\cdots,B_{17}\}=\{v_1,v_2,\cdots,v_{39}\}$，总共 39 个顶点的编号见图 6.1；

$\boldsymbol{W}_1=(w_{ij}^{(1)})_{39\times39}$，

$$w_{ij}^{(1)}=\begin{cases}d_{ij}^{(1)}, & i,j\text{之间有铁路直接相连}\\ +\infty, & i,j\text{之间没有铁路直接相连}\end{cases}$$，（$d_{ij}^{(1)}$ 表示 i， j 两点之间的铁路路程。）

然后应用 Floyd 算法求得任意两点间的最短铁路距离。

根据铁路运价表，可以得到铁路费用赋权完全图 $\tilde{G}_1=(V,E_1,\tilde{\boldsymbol{W}}_1)$，其中 $\tilde{\boldsymbol{W}}_1=(c_{ij}^{(1)})_{39\times39}$，这里 $c_{ij}^{(1)}$ 为第 i,j 顶点间的最小铁路运输费用，若两点间的铁路距离值为无穷大，则对应的铁路运输费用也为无穷大。

2）构造公路费用的赋权图

构造公路费用赋权图 $G_2=(V,E_2,\boldsymbol{W}_2)$，其中 V 同上， $\boldsymbol{W}_2=(c_{ij}^{(2)})_{39\times39}$，

$$c_{ij}^{(2)}=\begin{cases}0.1d_{ij}^{(2)}, & i,j\text{之间有公路直接相连}\\ +\infty, & i,j\text{之间没有公路直接相连}\end{cases}$$，（$d_{ij}^{(2)}$ 表示 i， j 两点之间的公路路程。）

3）计算任意两点间的最小运输费用

由于可以用铁路、公路交叉运送，所以任意相邻两点间的最小运输费用为铁路、公路两者最小运输费用的最小值。

构造铁路公路的混合赋权图 $G=(V,E,\boldsymbol{W})$， $\boldsymbol{W}=(c_{ij}^{(3)})_{39\times39}$，其中 $c_{ij}^{(3)}=\min\ (c_{ij}^{(1)},c_{ij}^{(2)})$。

对图 G 应用 Floyd 算法，就可以计算出所有顶点对之间的最小运输费用，最后提取需要的 S_i（$i=1,2,\cdots,7$）到 A_j （$j=1,2,\cdots,15$）的最小运输费用 $\tilde{c}_{ij}$ (单位：万元) 见表 6.3。

表 6.3　最小运费计算结果

170.7	160.3	140.2	98.6	38	20.5	3.1	21.2	64.2	92	96	106	121.2	128	142
215.7	205.3	190.2	171.6	111	95.5	86	71.2	114.2	142	146	156	171.2	178	192
230.7	220.3	200.2	181.6	121	105.5	96	86.2	48.2	82	86	96	111.2	118	132
260.7	250.3	235.2	216.6	156	140.5	131	116.2	84.2	62	51	61	76.2	83	97
255.7	245.3	225.2	206.6	146	130.5	121	111.2	79.2	57	33	51	71.2	73	87
265.7	255.3	235.2	216.6	156	140.5	131	121.2	84.2	62	51	45	26.2	11	28
275.7	265.3	245.2	226.6	166	150.5	141	131.2	99.2	76	66	56	38.2	26	2

任意两点间的最小运输费用加上出厂售价， 得到单位钢管从任一个 S_i（$i=1,2,\cdots,7$）到 A_j （$j=1,2,\cdots,15$）的购买和运送最小费用 c_{ij}。

2. 总费用的数学规划模型

目标函数

（1）从钢管厂到各枢纽点 $A_1,A_2,\cdots,A_{15}$ 的总购运费用为 $\sum_{i=1}^{7}\sum_{j=1}^{15}c_{ij}x_{ij}$。

（2）铺设管道不仅只运输到枢纽点，而是要运送并铺设到全部管线，注意到将总量为 y_j 的钢管从枢纽点往左运到每单位铺设点， 其运费应为第一公里、第二公里、…直到第 y_j 公里的运费之和，即为

$$0.1\times(1+2+\cdots+y_j)=\frac{0.1}{2}y_j(y_j+1).$$

从枢纽点 A_j 往右也一样，对应的铺设费用为

$$0.1\times(1+2+\cdots+z_j)=\frac{0.1}{2}z_j(z_j+1).$$

总的铺设费用为

$$\frac{0.1}{2}\sum_{j=1}^{15}\Big[y_j(y_j+1)+z_j(z_j+1)\Big].$$

因而，总费用为

$$\sum_{i=1}^{7}\sum_{j=1}^{15}c_{ij}x_{ij}+\frac{0.1}{2}\sum_{j=1}^{15}\Big[y_j(y_j+1)+z_j(z_j+1)\Big].$$

约束条件：

（1）根据钢管厂生产能力约束或购买限制，有

$$\sum_{j=1}^{15} x_{ij} \in \{0\} \cup [500, s_i], \quad i = 1,2,\cdots,7 .$$

(2) 购运量应等于铺设量

$$\sum_{i=1}^{7} x_{ij} = z_j + y_j, \quad j = 1,2,\cdots,15 .$$

（3）枢纽点间距约束：从两个相邻枢纽点分别往右、往左铺设的总单位钢管数应等于其间距，即

$$z_j + y_{j+1} = \left| A_j A_{j+1} \right| = l_j, \quad j = 1,2,\cdots,14 .$$

（4）端点约束：从枢纽点 A_1 只能往右铺，不能往左铺，故

$$y_1 = 0 ,$$

从枢纽点 A_{15} 只能往左铺，不能往右铺，故

$$z_{15} = 0 .$$

（5）非负约束：

$$x_{ij} \geq 0, y_j \geq 0, z_j \geq 0, \quad i = 1,2,\cdots,7, j = 1,2,\cdots,15 .$$

综上所述，建立如下数学规划模型

$$\min \sum_{i=1}^{7}\sum_{j=1}^{15} c_{ij} x_{ij} + \frac{0.1}{2}\sum_{j=1}^{15}(z_j(z_j+1) + y_j(y_j+1)) , \tag{11.5}$$

$$\text{s.t.} \begin{cases} \sum_{j=1}^{15} x_{ij} \in \{0\} \cup [500, s_i], \quad i = 1,2,\cdots,7, \\ \sum_{i=1}^{7} x_{ij} = z_j + y_j, \quad j = 1,2,\cdots,15, \\ z_j + y_{j+1} = l_j, \quad j = 1,2,\cdots,14, \\ y_1 = 0, z_{15} = 0, \\ x_{ij} \geq 0, y_j \geq 0, z_j \geq 0, \quad i = 1,2,\cdots,7, j = 1,2,\cdots,15. \end{cases} \tag{11.6}$$

3. 模型求解

使用计算机求解上述数学规划时，需要对约束条件(11.6)中的第一个非线性约束

$$\sum_{j=1}^{15} x_{ij} \in \{0\} \cup [500, s_i], \quad i = 1,2,\cdots,7 \tag{11.7}$$

进行处理。

首先把约束条件松弛为

$$0 \leq \sum_{j=1}^{15} x_{ij} \leq s_i, \quad i = 1,2,\cdots,7 . \tag{11.8}$$

求解发现第 4 个钢管厂和第 7 个钢管厂的供货量达不到 500，不符合要求。

再利用分支定界的思想，在约束(11.8)的基础上，把原问题分解为 4 个优化问题，在原来目标函数和约束条件的基础上，分别再加如下的 4 个约束条件：

$$\text{问题1:} \begin{cases} \sum_{j=1}^{15} x_{4j} = 0, \\ \sum_{j=1}^{15} x_{7j} = 0. \end{cases} \tag{11.9}$$

$$\text{问题2:}\begin{cases}\sum_{j=1}^{15} x_{4j} \geq 500, \\ \sum_{j=1}^{15} x_{7j} \geq 500.\end{cases} \tag{11.10}$$

$$\text{问题3:}\begin{cases}\sum_{j=1}^{15} x_{4j} \geq 500, \\ \sum_{j=1}^{15} x_{7j} = 0.\end{cases} \tag{11.11}$$

$$\text{问题4:}\begin{cases}\sum_{j=1}^{15} x_{4j} = 0, \\ \sum_{j=1}^{15} x_{7j} \geq 500.\end{cases} \tag{11.12}$$

利用 Matlab 软件，分别求解上述 4 个问题。从计算结果得知，问题 1 的解是最优的。求得总费用的最小值为 127.8632 亿。具体的订购和运输方案见表 6.4 所示。

表 6.4　钢管订购和运输方案

	订购量	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}
S_1	800	0	0	0	147	188	200	265	0	0	0	0	0	0	0	0
S_2	800	0	179	113	99	109	0	0	300	0	0	0	0	0	0	0
S_3	1000	0	0	124	95	117	0	0	0	664	0	0	0	0	0	0
S_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S_5	1216	0	0	271	127	202	0	0	0	0	201	415	0	0	0	0
S_6	1355	0	0	0	0	0	0	0	0	0	150	0	86	333	621	165
S_7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

习题 11

11.1　有一只装满 8 斤酒的瓶子和两只分别装 5 斤和 3 斤酒的空瓶，如何才能将这 8 斤酒分成两等份？

11.2　求图 11.9 中从 v_1 到其他各点的最短距离。

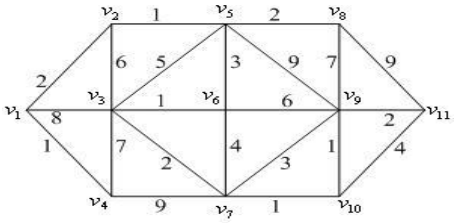


图 11.9　赋权无向图

11.3　2009 年中国研究生数学建模竞赛 D 题（110 警车配置及巡逻方案）的附件 Excel 文件“2009 年 D 题-附件.xlsx”包含“地图数据”和“道路数据”两个工作表。“地图数据”工作表给出了 307 个交叉路口的 x,y 坐标（单位：m），“道路数据”工作表给出了交叉路口之间的联系信息。

（1）绘制道路全景图；

（2）计算任意两个交叉路口之间的最短距离；

（3）求道路网络的最小生成树，并绘出该树。