数学建模算法与应用

第4章　图与网络模型及方法

图论起源于 18 世纪。第一篇图论论文是瑞士数学家欧拉于 1736 年发表的“哥尼斯堡的七座桥”。1847 年，克希霍夫为了给出电网络方程而引进了“树”的概念 1857 年，凯莱在计数烷C_nH_{2n+2}的同分异构物时，也发现了“树”。哈密尔顿于 1859 年提出“周游世界”游戏，用图论的术语，就是如何找出一个连通图中的生成圈、近几十年来，由于计算机技术和科学的飞速发展，大大地促进了图论研究和应用，图论的理论和方法已经渗透到物理、化学、通讯科学、建筑学、运筹学、生物遗传学、心理学、经济学、社会学等学科中。

图论中所谓的“图”是指某类具体事物和这些事物之间的联系。如果我们用点表示这些具体事物，用连接两点的线段(直的或曲的)表示两个事物的特定的联系，就得到了描述这个“图”的几何形象。图论为任何一个包含了一种二元关系的离散系统提供了一个数学模型，借助于图论的概念、理论和方法，可以对该模型求解。哥尼斯堡七桥问题就是一个典型的例子。在哥尼斯堡有七座桥将普莱格尔河中的两个岛及岛与河岸联结起来，问题是要从这四块陆地中的任何一块开始通过每一座桥正好一次，再回到起点。

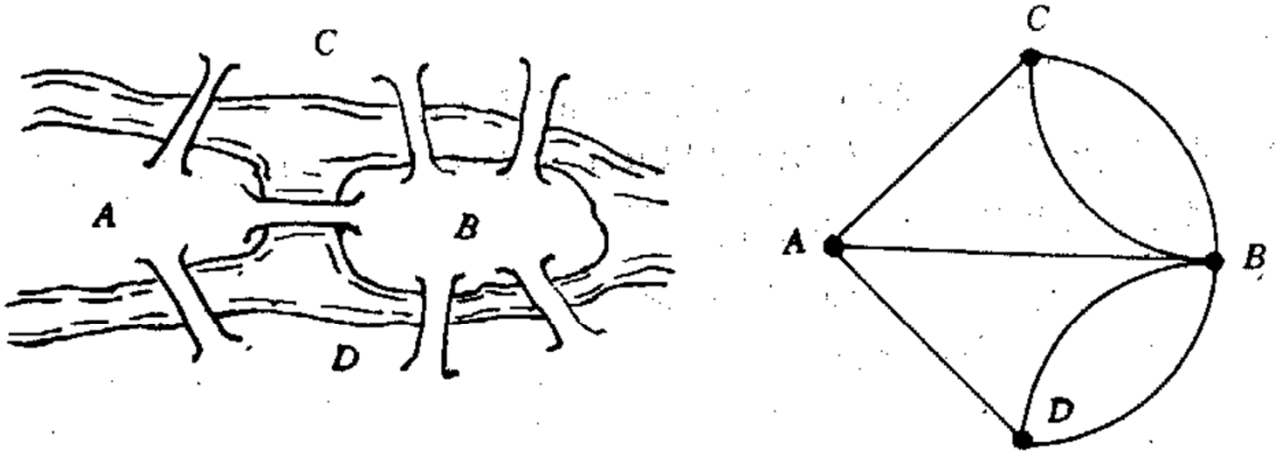

图 4.1　哥尼斯堡七桥问题

4.1　图的基本概念与数据结构

4.1.1　基本概念

所谓的图，直观地讲就是在平面上n个点，把其中的一些点对用曲线或直线连接起来，不考虑点的位置与连线曲直长短，这样形成一个关系结构就是一个图。记成$G=(V,E)$，V是以上述点为元素的顶点集，E是以上述连线为元素的边集。

如果各条边都加上方向，则称为有向图，否则称为无向图。如果有的边有方向，有的边无方向，则称为混合图。

如果任两顶点间最多有一条边，且每条边的两个端点皆不重合的图，称为简单图。

如果图的两顶点间有边相连，则称此两顶点相邻，每一对顶点都相邻的图称为完全图，否则称为非完全图，完全图记为$K_{|V|}$。

若$V = X \cup Y$，$X \cap Y = \Phi$，$|X| \cdot |Y| \neq 0$（这里$|X|$表示顶点集X中元素的个数），且X中无相邻的顶点对，Y中亦然，则称图G为二分图；特别地，若对任意$u \in X$，u与Y中每个顶点相邻，则称图G为完全二分图，记为$K_{|X|,|Y|}$。

设$v \in V$是边$e \in E$的端点，则称v与e相关联，与顶点v关联的边数称为该顶点的度，记为$d(v)$，度为奇数的顶点称为奇顶点，度为偶数的顶点称为偶顶点。可以证明$\sum_{v \in V(G)} d(v) = 2|E|$，即所有顶点的度数之和是边数的两倍，且由此可知奇顶点的总数是偶数。

设$W=v_0e_1v_1e_2\cdots e_kv_k$，其中$e_i\in E$，$1\le i\le k$，$v_j\in V$ $0\le j\le k$，e_i与v_{i-1}和v_i关联，称W是图G的一条道路，k为路长，v_0为起点，v_k为终点；各边相异的道路称为迹；各顶点相异的道路称为轨道。若W是一轨道，可记为$P(v_0,v_k)$；起点与终点重合的道路称为回路；起点与终点重合的轨道称为圈，即对轨道$P(v_0,v_k)$，当$v_0=v_k$时成为一圈；图中任两顶点之间都存在道路的图，称为连通图。图中含有所有顶点的轨道称为 Hamilton 轨，闭的 Hamilton 轨称为 Hamilton 圈；含有 Hamilton 圈的图称为 Hamilton 图。

称两顶点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；在完全二分图$K_{|X|,|Y|}$中，X中两顶点之间的距离为偶数，X中的顶点与Y中的顶点的距离为奇数。

赋权图是指每条边都有一个(或多个)实数对应的图，这个(些)实数称为这条边的权(每条边可以具有多个权)。赋权图在实际问题中非常有用。根据不同的实际情况，权数的含义可以各不相同。例如，可用权数代表两地之间的实际距离或行车时间，也可用权数代表某工序所需的加工时间等。

4.1.2 图与网络的数据结构

为了在计算机上实现网络优化的算法，首先我们必须有一种方法（即数据结构）在计算机上来描述图与网络。一般来说，算法的好坏与网络的具体表示方法，以及中间结果的操作方案是有关系的。这里我们介绍计算机上用来描述图与网络的 2 种主要表示方法：邻接矩阵表示法和稀疏矩阵表示法。在下面数据结构的讨论中，首先假设$G=(V,E)$是一个简单无向图，顶点集合$V=\{v_1,\cdots,v_n\}$，边集$E=\{e_1,\cdots,e_m\}$，记$|V|=n$，$|E|=m$。

1. 邻接矩阵表示法

邻接矩阵是表示顶点之间相邻关系的矩阵，邻接矩阵记作$W=(w_{ij})_{n\times n}$，当G为赋权图时

$$w_{ij}=\begin{cases}\text{权值，当}v_i\text{与}v_j\text{之间有边时，}\\ 0\text{或}\infty\text{，当}v_i\text{与}v_j\text{之间无边时.}\end{cases}$$

当G为非赋权图时，

$$w_{ij}=\begin{cases}1\text{，当}v_i\text{与}v_j\text{之间有边时，}\\ 0\text{，当}v_i\text{与}v_j\text{之间无边时.}\end{cases}$$

采用邻接矩阵表示图，直观方便，通过查邻接矩阵元素的值可以很容易地查找图中任两个顶点v_i和v_j之间有无边，以及边上的权值。当图的边数m远小于顶点数n时，邻接矩阵表示法会造成很大的空间浪费。

2. 稀疏矩阵表示法

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稀疏矩阵，只要存放非零元素的行标、列标、非零元素的值即可，可以按如下方式存储（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 的命令 sparse 命令，就可以把邻接矩阵转化为稀疏矩阵的表示方式。

对于无向图，由于邻接矩阵是对称阵，Matlab 中只需使用邻接矩阵的下三角元素，即 Matlab 只存储邻接矩阵下三角元素中的非零元素。

稀疏矩阵只是一种存储格式。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。

4.2 最短路问题

4.2.1 两个指定顶点之间的最短路径

问题如下，给出了一个连接若干个城镇的铁路网络在这个网络的两个指定城镇间，找一条最短铁路线。

构造赋权图 $G=(V,E,W)$，其中顶点集 $V=\{v_1,\cdots,v_n\}$，这里 $v_1,\cdots,v_n$ 表示各个小城镇，E 为边的集合，邻接矩阵 $W=(w_{ij})_{n\times n}$，这里 w_{ij} 表示顶点 v_i 和 v_j 之间直通铁路的距离，若顶点 v_i 和 v_j 之间无铁路，则 $w_{ij}=\infty$。问题就是求赋权图 G 中指定的两个顶点 u_0,v_0 间的具有最小权的路。这条路叫做 u_0,v_0 间的最短路，它的权叫做 u_0,v_0 间的距离，亦记作 $d(u_0,v_0)$。

求最短路已有成熟的算法，如迪克斯特拉（Dijkstra）算法，其基本思想是按距u_0从近到远为顺序，依次求得u_0到G的各顶点的最短路和距离，直至v_0（或直至G的所有顶点），算法结束。为避免重复并保留每一步的计算信息，采用了标号算法。下面是该算法。

(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$\min_{v\in\overline{S}_i}\{l(v)\}$，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$i=|V|-1$，停止；若$i<|V|-1$，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。

例 4.1 某公司在六个城市$c_1, c_2, \cdots, c_6$中有分公司，从c_i到c_j的直接航程票价记在下述矩阵的(i, j)位置上。（∞表示无直接航路），请帮助该公司设计一张城市c_1到其他城市间的票价最便宜的路线图。

$$\begin{bmatrix} 0 & 50 & \infty & 40 & 25 & 10 \\ 50 & 0 & 15 & 20 & \infty & 25 \\ \infty & 15 & 0 & 10 & 20 & \infty \\ 40 & 20 & 10 & 0 & 10 & 25 \\ 25 & \infty & 20 & 10 & 0 & 55 \\ 10 & 25 & \infty & 25 & 55 & 0 \end{bmatrix}$$

用矩阵$a_{n\times n}$（n为顶点个数）存放各边权的邻接矩阵，行向量pb、$index_1$、$index_2$、d分别用来存放P标号信息、标号顶点顺序、标号顶点索引、最短通路的值。其中分量

$$pb(i)=\begin{cases}1 & \text{当第}i\text{顶点的标号已成为}P\text{标号;}\\ 0 & \text{当第}i\text{顶点的标号未成为}P\text{标号;}\end{cases}$$

$index_2(i)$ 存放始点到第i顶点最短通路中第i顶点前一顶点的序号；

$d(i)$ 存放由始点到第i顶点最短通路的值。

求得c_1到$c_2,\cdots,c_6$的最便宜票价分别为35，45，35，25，10。

4.2.2 两个指定顶点之间最短路问题的数学规划模型

假设有向图有n个顶点，现需要求从顶点v_1到顶点v_n的最短路。设$W=(w_{ij})_{n\times n}$为邻接矩阵，其分量为

$$w_{ij}=\begin{cases}\text{边}v_iv_j\text{的权值}, & v_iv_j\in E,\\ \infty, & \text{其它},\end{cases}$$

决策变量为x_{ij}，当$x_{ij}=1$，说明弧v_iv_j位于顶点v_1至顶点v_n的最短路上；否则$x_{ij}=0$。其数学规划表达式为

$$\min \sum_{v_iv_j\in E} w_{ij}x_{ij},$$

$$\text{s.t.} \sum_{\substack{j=1\\ v_iv_j\in E}}^{n} x_{ij}-\sum_{\substack{j=1\\ v_jv_i\in E}}^{n} x_{ji}=\begin{cases}1, & i=1,\\ -1, & i=n,\\ 0, & i\neq 1,n,\end{cases}$$

$$x_{ij}=0\text{或}1.$$

例 4.2 在图 4.2 中，用点表示城市，现有 $A, B_1, B_2, C_1, C_2, C_3, D$ 共 7 个城市。点与点之间的连线表示城市间有道路相连。连线旁的数字表示道路的长度。现计划从城市 A 到城市 D 铺设一条天然气管道，请设计出最小长度管道铺设方案。

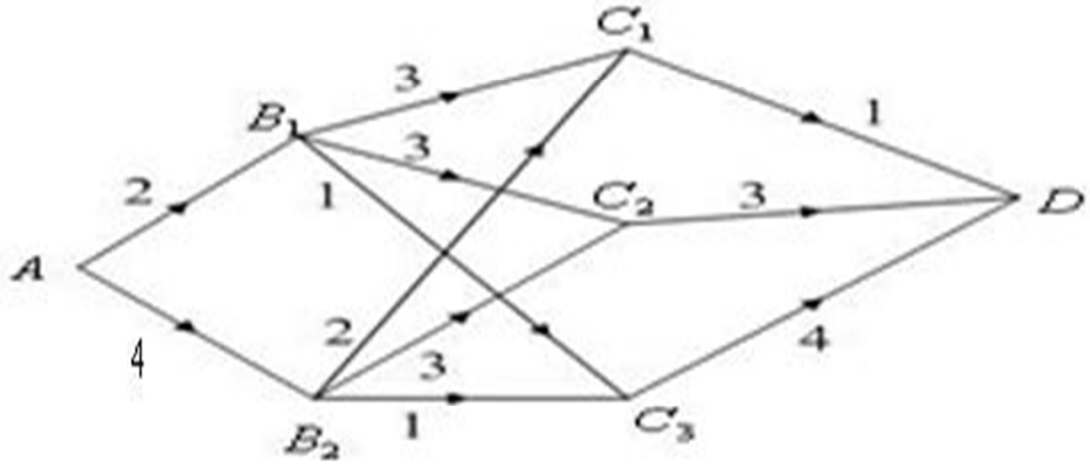

图 4.2 7 个城市间的连线图

求得最短铺设方案是铺设 AB_1, B_1C_1, C_1D 段，最短铺设长度为 6。

例 4.3（无向图的最短路问题）求图 4.3 中v_1到v_{11}的最短路。

分析　例 4.2 处理的问题属于有向图的最短路问题，本例是处理无向图的最短路问题，在处理方式上与有向图的最短路问题有一些差别，这里选择赋权邻接矩阵的方法编写 LINGO 程序。

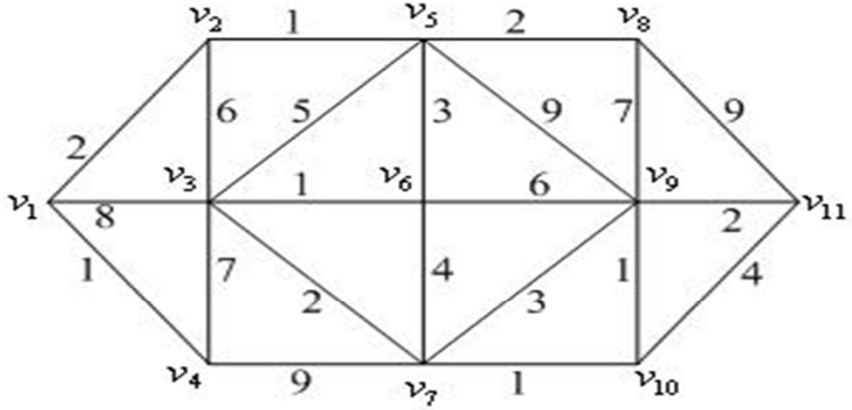

图 4.3　赋权无向图

与有向图相比较，在程序中只增加了一个语句@sum(cities(j):x(j,1))=0，即从顶点 1 离开后，再不能回到该顶点。

求得的最短路径为 1→2→5→6→3→7→10→9→11，最短路径长度为 13。

4.2.3 每对顶点之间的最短路径

计算赋权图中各对顶点之间最短路径，显然可以调用Dijkstra 算法。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$n-1$次这样的操作，就可得到从每一个顶点到其它顶点的最短路径。这种算法的时间复杂度为$O(n^3)$。第二种解决这一问题的方法是由 Floyd, R. W.提出的算法，称之为 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

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

$$a_{ii}=0,\ i=1,2,\cdots,n.$$

对于无向图，A_0是对称矩阵，$a_{ij}=a_{ji}$。

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。

例4.4　用Floyd算法求解例4.1。

4.3 最小生成树问题

4.3.1 基本概念

连通的无圈图叫做树，记之为T；其度为 1 的顶点称为叶子顶点；显然有边的树至少有两个叶子顶点。

若图$G=(V(G),E(G))$和树$T=(V(T),E(T))$满足$V(G)=V(T)$，$E(T)\subset E(G)$，则称T是G的生成树。图G连通的充分必要条件为G有生成树，一个连通图的生成树的个数很多。

树有下面常用的五个充要条件。

定理 4.1 （1）$G=(V,E)$是树当且仅当G中任二顶点之间有且仅有一条轨道。

（2）G是树当且仅当G无圈，且$|E|=|V|-1$。

（3）G是树当且仅当G连通，且$|E|=|V|-1$。

（4）G是树当且仅当G连通，且$\forall e\in E$，$G-e$不连通。

（5）G是树当且仅当G无圈，$\forall e\notin E$，$G+e$恰有一个圈。

4.3.2 最小生成树

欲修筑连接n个城市的铁路，已知i城与j城之间的铁路造价为c_{ij}，设计一个线路图，使总造价最低。

上述问题的数学模型是在连通赋权图上求权最小的生成树。赋权图的具最小权的生成树叫做最小生成树。

4.3.2.1 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

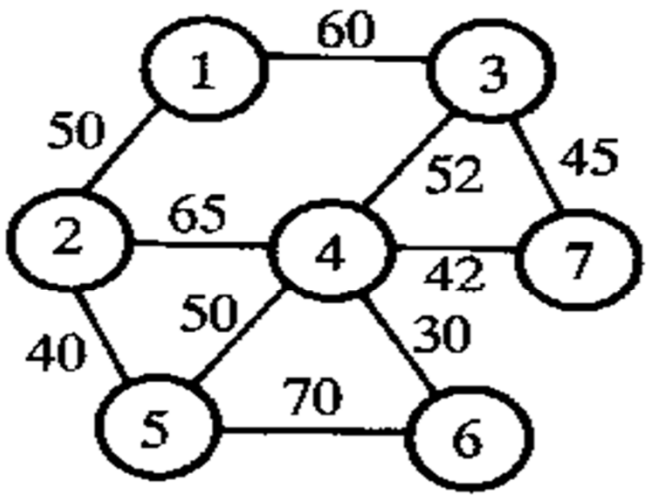

图 4.4　最小生成树问题

例 4.5　用 prim 算法求图 4.4 的最小生成树。

求得最小生成树的边集为$\{v_1v_2, v_2v_5, v_5v_4, v_4v_6, v_4v_7, v_7v_3\}$。

4.3.2.2 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$e_1 \in E(G)$，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$E(G)-\{e_1, e_2, \cdots, e_i\}$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$E(G)-\{e_1, e_2, \cdots, e_i\}$中权值最小的边。

（3）直到选得$e_{|V|-1}$为止。

例 4.6　用 Kruskal 算法构造例 4.5 的最小生成树。

用$index_{2\times n}$存放各边端点的信息，当选中某一边之后，就将此边对应的顶点序号中较大序号u改为此边的另一序号v，同时把后面边中所有序号为u的改为v。此方法的几何意义是将序号u的这个顶点收缩到v顶点，u顶点不复存在。后面继续寻查时，发现某边的两个顶点序号相同时，认为已被收缩掉，失去了被选取的资格。

求解结果和例 4.5 相同。

4.4 网络最大流问题

4.4.1 基本概念与基本定理

1. 网络与流

定义 4.1 给一个有向图$D=(V,A)$，其中A为弧集，在V中指定了一点，称为发点（记为v_s），和另一点，称为收点（记为v_t），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。通常我们就把这样的有向图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。

2. 可行流与最大流

定义 4.2 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

对于中间点，流出量＝流入量，即对于每个 $i(i \ne s,t)$ 有

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

对于发点 v_s，记

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

对于收点 v_t，

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

式中 $v(f)$ 称为这个可行流的流量，即发点的净输出量。

可行流总是存在的，例如零流。

最大流问题可以写为如下的线性规划模型

$$\max v(f)$$

s.t.

$$\sum_{j:(v_i,v_j)\in A} f_{ij} - \sum_{j:(v_j,v_i)\in A} f_{ji} = \begin{cases} v(f), & i = s, \\ -v(f), & i = t, \\ 0, & i \neq s,t, \end{cases} \tag{4.1}$$

$$0 \leq f_{ij} \leq c_{ij}, \quad \forall (v_i,v_j) \in A.$$

3．增广路

若给一个可行流 $f=\{f_{ij}\}$，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，我们定义路的方向是从 v_s 到 v_t，则路上的弧被分为两类：一类是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^+。另一类弧与路的方向相反，称为后向弧。后向弧的全体记为 μ^-。

定义 4.3 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（关于可行流 f ）一条增广路。

4.4.2　寻求最大流的标号法（Ford－Fulkerson）

从v_s到v_t的一个可行流出发（若网络中没有给定f，则可以设f是零流），经过标号过程与调整过程，即可求得从v_s到v_t的最大流。这两个过程的步骤分述如下。

（A）标号过程

在下面的算法中，每个顶点v_x的标号值有两个，v_x的第一个标号值表示在可能的增广路上，v_x的前驱顶点；v_x的第二个标号值记为δ_x，表示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（B）。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。

（B）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）。

例4.7　现需要将城市s的石油通过管道运送到城市t，中间有4个中转站v_1,v_2,v_3和v_4，城市与中转站的连接以及管道的容量如图4.5所示，求从城市s到城市t的最大流。

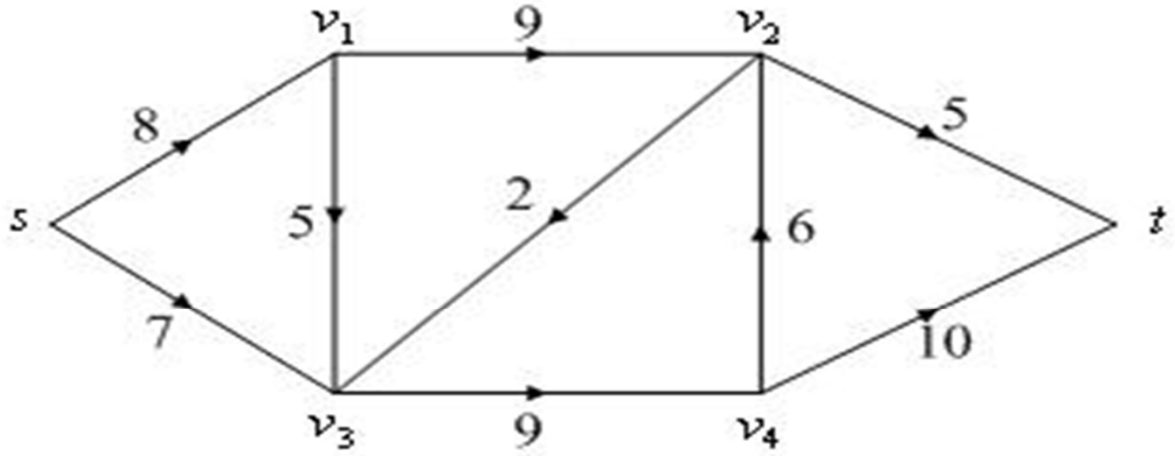

图4.5　网络图

4.5 最小费用最大流问题

4.5.1 最小费用最大流

给定网络$D=(V,A,C)$，每一弧$(v_i,v_j)\in A$上，除了已给容量c_{ij}外，还给了一个单位流量的费用$b(v_i,v_j)\geq 0$（简记为b_{ij}）。所谓最小费用最大流问题就是要求一个从发点v_s到收点v_t的最大流，使流的总输送费用$\sum_{(v_i,v_j)\in A} b_{ij}f_{ij}$取最小值。

最小费用最大流问题可以归结为两个线性规划问题，首先用（4.1）的线性规划模型求出最大流量$v(f_{\max})$，然后用如下的线性规划模型求出最大流对应的最小费用。

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

$$\text{s.t.} \quad 0 \le f_{ij} \le c_{ij}, \quad \forall (v_i,v_j) \in A, \tag{4.2}$$

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

其中

$$d_i = \begin{cases} v(f_{\max}), & i = s, \\ -v(f_{\max}), & i = t, \\ 0, & i \neq s,t. \end{cases}$$

这里$v(f_{\max})$表示(4.1)线性规划模型求得的最大流的流量。

例 4.8（最小费用最大流问题）（续例 4.7）　由于输油管道的长短不一或地质等原因，使每条管道上运输费用也不相同，因此，除考虑输油管道的最大流外，还需要考虑输油管道输送最大流的最小费用。图 4.6 所示是带有运费的网络，其中第 1 个数字是网络的容量，第 2 个数字是网络的单位运费。

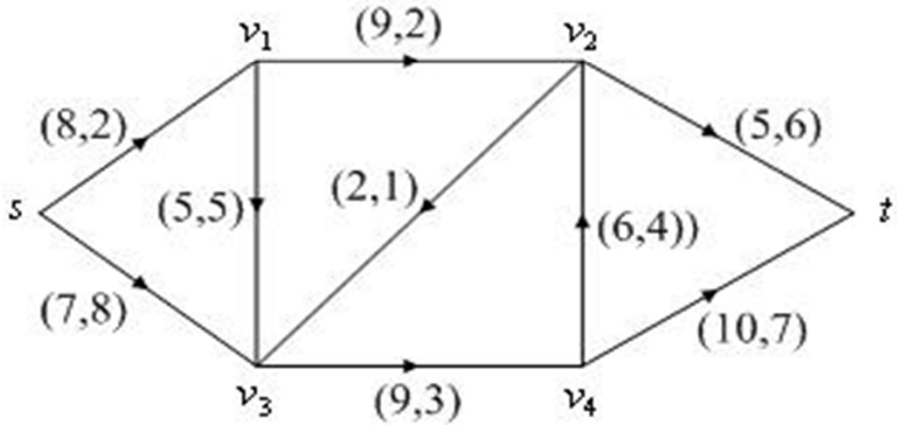

图 4.6　最小费用最大流的网络图

求得最大流的最小费用是 205。

4.5.2 求最小费用流的一种迭代方法

这里所介绍的求最小费用流的迭代方法，是由Busacker 和 Gowan 在 1961 年提出的。其主要步骤如下

（1）求出从发点到收点的最小费用通路$\mu(s,t)$。

（2）对该通路$\mu(s,t)$分配最大可能的流量

$$\overline{f} = \min_{(v_i,v_j)\in\mu(s,t)}\{c_{ij}\}$$

并让通路上的所有边的容量相应减少 $\overline{f}$。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$\mu(s,t)$上所有边(v_i,v_j)的反向边(v_j,v_i)。令

$$c_{ji}=\overline{f}\ ,\ b_{ji}=-b_{ij}$$

（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指定的$v(f)$为止（或者再也找不到从v_s到v_t的最小费用通路）。

4.6 Matlab 的图论工具箱

4.6.1 Matlab 图论工具箱的命令

Matlab 图论工具箱的命令见表 4.1。

表 4.1 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
graphisomorphism	确定两个图是否同构，同构返回 1，否则返回 0

命令名	功能
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	在图中找最小生成树
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

4.6.2　应用举例

例 4.9　用 Matlab 工具箱求图 4.7 中从v_1到v_{11}的最短路和最短路径。

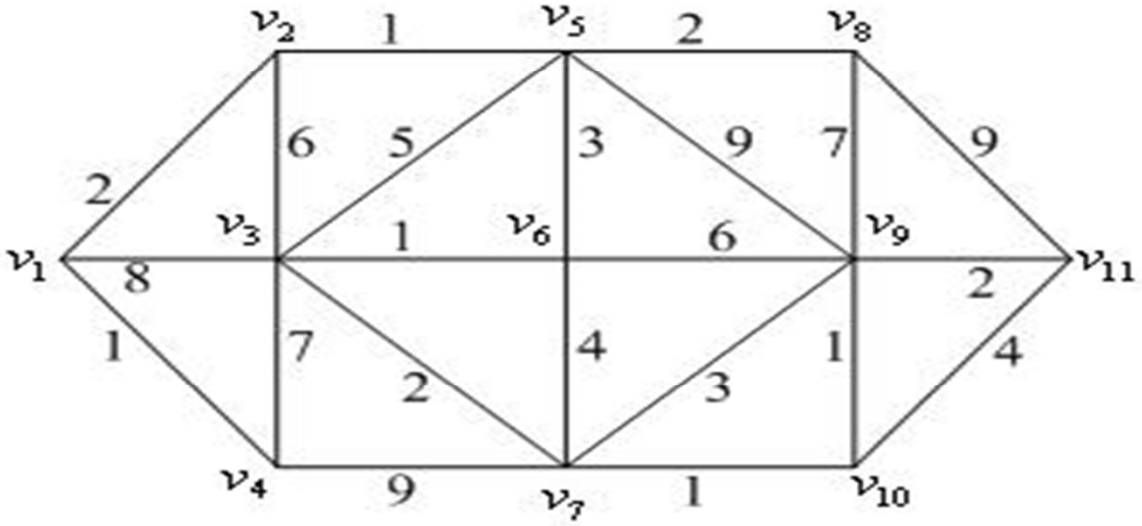

图 4.7　无向图的最短路

求得最短路径为

$$v_1 \to v_2 \to v_5 \to v_6 \to v_3 \to v_7 \to v_{10} \to v_9 \to v_{11}$$，

最短路径的长度为 13。

例 4.10（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?

解　该问题可以使用图论中的最短路算法进行求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，可以把相应的权重取为∞。

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）。

现在规定状态向量与转移向量之间的运算为

$$0+0=0，1+0=1，0+1=1，1+1=0$$

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。

赋权图G之间的状态转移关系见图 4.8，最终求得的状态转移顺序为

1　　6　　3　　7　　2　　8　　5　　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。

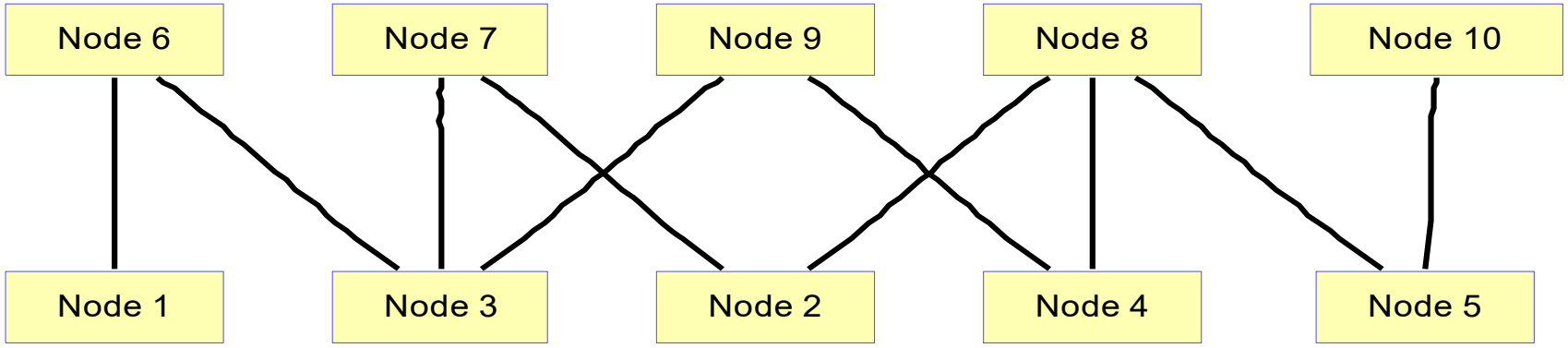

图 4.8　可行状态之间的转移

例 4.11　求图 4.9 所示有向图中v_s到v_t的最短路径及长度

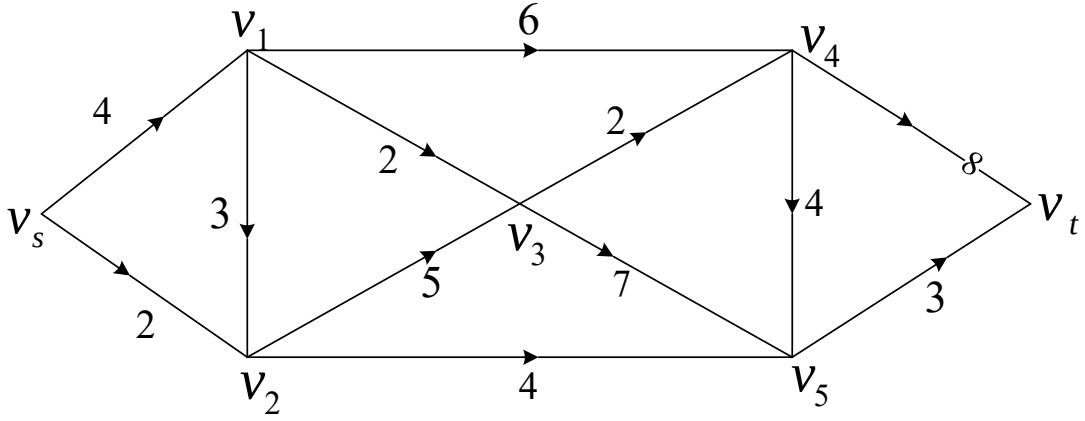

图 4.9　有向图的最短路

解　该赋权有向图中顶点集$V=\{v_s,v_1,\cdots,v_5,v_t\}$中总共有 7 个顶点，邻接矩阵

$$W=\begin{bmatrix}
0 & 4 & 2 & \infty & \infty & \infty & \infty \\
\infty & 0 & 3 & 2 & 6 & \infty & \infty \\
\infty & \infty & 0 & 5 & \infty & 4 & \infty \\
\infty & \infty & \infty & 0 & 2 & 7 & \infty \\
\infty & \infty & \infty & \infty & 0 & 4 & 8 \\
\infty & \infty & \infty & \infty & \infty & 0 & 3 \\
\infty & \infty & \infty & \infty & \infty & \infty & 0
\end{bmatrix}.$$

例 4.12　设有 9 个节点$v_i(i=1,\cdots,9)$，他们的坐标分别为(x_i,y_i)，具体数据见表 4.2。任意两个节点之间的距离为

$$d_{ij}=|x_i-x_j|+|y_i-y_j|,$$

问怎样连接电缆，使每个节点都连通，且所用的总电缆长度为最短?

表 4.2　点的坐标数据表

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$W=(w_{ij})_{9\times9}$为邻接矩阵，其中$w_{ij}=d_{ij}$，$i,j=1,2,\cdots,9$。求总电缆长度最短的问题实际上就是求图G的最小生成树。

例 4.13　求图 4.10 中从①到⑧的最大流。

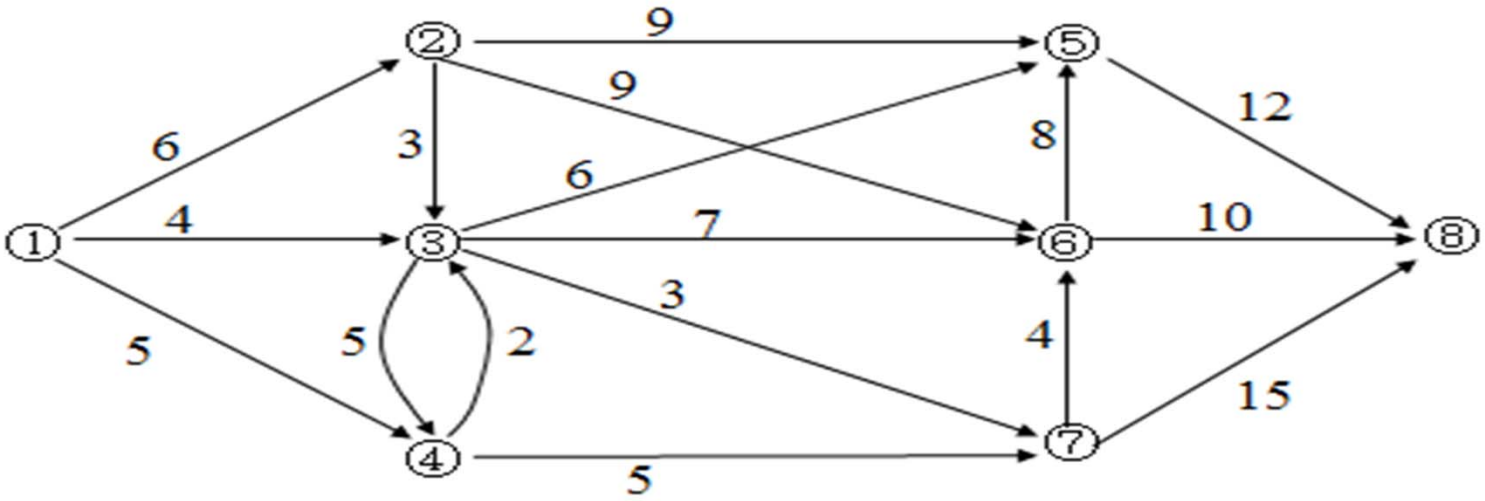

图 4.10　最大流问题的网络图

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 4.10 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，并添加两条弧，删除顶点 4 到顶点 3 的权重为 2 的弧，加入的两条弧的容量都是 2。

4.7　旅行商（TSP）问题

4.7.1　修改圈近似算法

一名推销员准备前往若干城市推销产品，然后回到他的出发地。如何为他设计一条最短的旅行路线（从驻地出发，经过每个城市恰好一次，最后返回驻地）？这个问题称为旅行商问题。用图论的术语说，就是在一个赋权完全图中，找出一个有最小权的 Hamilton 圈。称这种圈为最优圈。目前还没有求解旅行商问题的有效算法。所以希望有一个方法以获得相当好（但不一定最优）的解。

一个可行的办法是首先求一个 Hamilton 圈C，然后适当修改C以得到具有较小权的另一个 Hamilton 圈。修改的方法叫做改良圈算法。设初始圈$C = v_1v_2\cdots v_nv_1$。

（1）对于$1 \le i < i+1 < j \le n$，构造新的 Hamilton 圈

$$C_{ij} = v_1v_2\cdots v_iv_jv_{j-1}v_{j-2}\cdots v_{i+1}v_{j+1}v_{j+2}\cdots v_nv_1,$$

它是由C中删去边v_iv_{i+1}和v_jv_{j+1}，添加边v_iv_j和$v_{i+1}v_{j+1}$而得到的。若$w(v_iv_j)+w(v_{i+1}v_{j+1}) < w(v_iv_{i+1})+w(v_jv_{j+1})$，则以$C_{ij}$代替$C$，$C_{ij}$叫做$C$的改良圈。

（2）转（1），直至无法改进，停止。

例 4.14　从北京(Pe)乘飞机到东京(T)、纽约(N)、墨西哥城(M)、伦敦(L)、巴黎(Pa)五城市做旅游，每城市恰去一次再回北京，应如何安排旅游线，使旅程最短？用修改圈算法，求一个近似解。各城市之间的航线距离如表 4.3。

表 4.3　六城市间的距离

	L	M	N	Pa	Pe	T
L		56	35	21	51	60
M	56		21	57	78	70
N	35	21		36	68	68
Pa	21	57	36		51	61
Pe	51	78	68	51		13
T	60	70	68	61	13	

求得近似圈为5→4→1→3→2→6→5；近似圈的长度为211。

实际上我们我们可以用下节的数学规划模型求得精确的最短圈长度为 211，这里的近似算法凑巧求出了准确解。

4.7.2 旅行商问题的数学规划模型

设城市的个数为n，d_{ij}是两个城市i与j之间的距离，$x_{ij}=0$或 1（1 表示走过城市i到城市j的路，0 表示没有选择走这条路）。则有

$$\min \sum_{i\neq j} d_{ij}x_{ij},$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij}=1,\ i=1,2,\cdots,n,\ \text{(每个点只有一条边出去)},$$

$$\sum_{i=1}^{n} x_{ij}=1,\ j=1,2,\cdots,n,\ \text{(每个点只有一条边进去)},$$

$$\sum_{i,j\in s} x_{ij}\leq |s|-1,\ 2\leq |s|\leq n-1,\ s\subset\{1,2,\cdots,n\},\ \text{即}s\text{为}\{1,2,\cdots,n\}\text{的真子集},$$

（除起点和终点外，各边不构成圈）

$$x_{ij}\in\{0,1\},\ i,j=1,2,\cdots,n,\ i\neq j.$$

例 4.15　已知 SV 地区各城镇之间距离见表 4.4，某公司计划在 SV 地区做广告宣传，推销员从城市 1 出发，经过各个城镇，再回到城市 1。为节约开支，公司希望推销员走过这 10 个城镇的总距离最少。

表 4.4　城镇之间的距离

	2	3	4	5	6	7	8	9	10
1	8	5	9	12	14	12	16	17	22
2		9	15	17	8	11	18	14	22
3			7	9	11	7	12	12	17
4				3	17	10	7	15	18
5					8	10	6	15	15
6						9	14	8	16
7							8	6	11
8								11	11
9									10

求得的最短路径为

1→2→6→9→7→10→8→5→4→3→1，

最短路径长度为 73。

4.8 计划评审方法和关键路线法

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）。

4.8.1 计划网络图

例 4.16 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 4.5 所示，求完成该项目的最短时间。

表 4.5 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			

4.8.1.1　计划网络图的概念

定义 4.4　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 4.11 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 4.11　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 4.5　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。

4.8.1.2　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。

4.8.2　时间参数

1　事件时间参数

1）事件的最早时间

事件 j 的最早时间用 $t_E(j)$ 表示，它表明以事件 j 为始点的各工作最早可能开始的时间，也表示以事件 j 为终点的各工作的最早可能完成时间，它等于从始点事件到该事件的最长路线上所有工作的工时总和。事件最早时间可用下列递推公式，按照事件编号从小到大的顺序逐个计算。

设事件编号为 $1,2,\cdots,n$，则

$$\begin{cases} t_E(1)=0, \\ t_E(j)=\max\limits_i\{t_E(i)+t(i,j)\}, \end{cases} \tag{4.3}$$

其中$t_E(i)$是与事件j相邻的各紧前事件的最早时间，$t(i,j)$是作业(i,j)所需的工时。

终点事件的最早时间显然就是整个工程的总最早完工期，即

$t_E(n)=$总最早完工期.

2）事件的最迟时间

事件i的最迟时间用$t_L(i)$表示，它表明在不影响任务总工期条件下，以事件i为始点的工作的最迟必须开始时间，或以事件i为终点的各工作的最迟必须完成时间。由于一般情况下，都把任务的最早完工时间作为任务的总工期，所以事件最迟时间的计算公式为

$$\begin{cases} t_L(n) = \text{总工期（或}t_E(n)\text{）}, \\ t_L(i) = \min\limits_j \{t_L(j) - t(i,j)\}, \end{cases} \tag{4.4}$$

其中$t_L(j)$是与事件i相邻的各紧后事件的最迟时间。

公式（4.4）也是递推公式，但与（4.3）相反，是从终点事件开始，按编号由大至小的顺序逐个由后向前计算。

2　工作的时间参数

1）工作的最早可能开工时间与工作的最早可能完工时间

一个工作(i,j)的最早可能开工时间用$t_{ES}(i,j)$表示。任何一件工作都必须在其所有紧前工作全部完工后才能开始。工作(i,j)的最早可能完工时间用$t_{EF}(i,j)$表示，它表示工作按最早开工时间开始所能达到的完工时间。它们的计算公式为

$$\begin{cases} t_{ES}(1,j)=0, \\ t_{ES}(i,j)=\max\limits_{k}\{t_{ES}(k,i)+t(k,i)\}, \\ t_{EF}(i,j)=t_{ES}(i,j)+t(i,j). \end{cases} \tag{4.5}$$

这组公式也是递推公式。即所有从总开工事件出发的工作$(1, j)$，其最早可能开工时间为零；任一工作(i, j)的最早开工时间要由它的所有紧前工作(k, i)的最早开工时间决定；工作(i, j)的最早完工时间显然等于其最早开工时间与工时之和。

2）工作的最迟必须开工时间与工作的最迟必须完工时间

一个工作(i,j)的最迟开工时间用$t_{LS}(i,j)$表示。它表示工作(i,j)在不影响整个任务如期完成的前提下，必须开始的最晚时间。

工作(i,j)的最迟必须完工时间用$t_{LF}(i,j)$表示。它表示工作(i,j)按最迟时间开工，所能达到的完工时间。它们的计算公式为

$$\begin{cases} t_{LF}(i,n) = \text{总完工期（或}t_{EF}(i,n)\text{）}, \\ t_{LS}(i,j) = \min\limits_{k}\{t_{LS}(j,k) - t(i,j)\}, \\ t_{LF}(i,j) = t_{LS}(i,j) + t(i,j). \end{cases} \tag{4.6}$$

这组公式是按工作的最迟必须开工时间由终点向始点逐个递推的公式。凡是进入总完工事件n的工作(i,n)，其最迟完工时间必须等于预定总工期或等于这个工作的最早可能完工时间。任一工作(i,j)的最迟必须开工时间由它的所有紧后工作(j,k)的最迟开工时间确定。而工作(i,j)的最迟完工时间显然等于本工作的最迟开工时间与工时的和。

由于任一个事件i（除去始点事件和终点事件），既表示某些工作的开始又表示某些工作的结束。所以从事件与工作的关系考虑，用公式（4.5），公式（4.6）求得的有关工作的时间参数也可以通过事件的时间参数公式（4.3），公式（4.4）来计算。如工作(i,j)的最早可能开工时间$t_{ES}(i,j)$就等于事件i的最早时间$t_E(i)$。工作(i,j)的最迟必须完工时间等于事件j的最迟时间

3　时差

工作的时差又叫工作的机动时间或富裕时间，常用的时差有两种。

1）工作的总时差

在不影响任务总工期的条件下，某工作(i,j)可以延迟其开工时间的最大幅度，叫做该工作的总时差，用$R(i,j)$表示。其计算公式为

$$R(i,j)=t_{LF}(i,j)-t_{EF}(i,j), \qquad (4.7)$$

即工作(i,j)的总时差等于它的最迟完工时间与最早完工时间的差。显然$R(i,j)$也等于该工作的最迟开工时间与最早开工时间之差。

2）工作的单时差

工作的单时差是指在不影响紧后工作的最早开工时间条件下，此工作可以延迟其开工时间的最大幅度，用$r(i,j)$表示。其计算公式为

$$r(i,j)=t_{ES}(j,k)-t_{EF}(i,j), \tag{4.8}$$

即单时差等于其紧后工作的最早开工时间与本工作的最早完工时间之差。

4.8.3　计划网络图的计算

以例 4.16 的求解过程为例介绍计划网络图的计算方法

1　建立计划网络图

首先建立计划网络图。按照上述规则，建立例 4.16 的计划网络图，如图 4.12 所示。

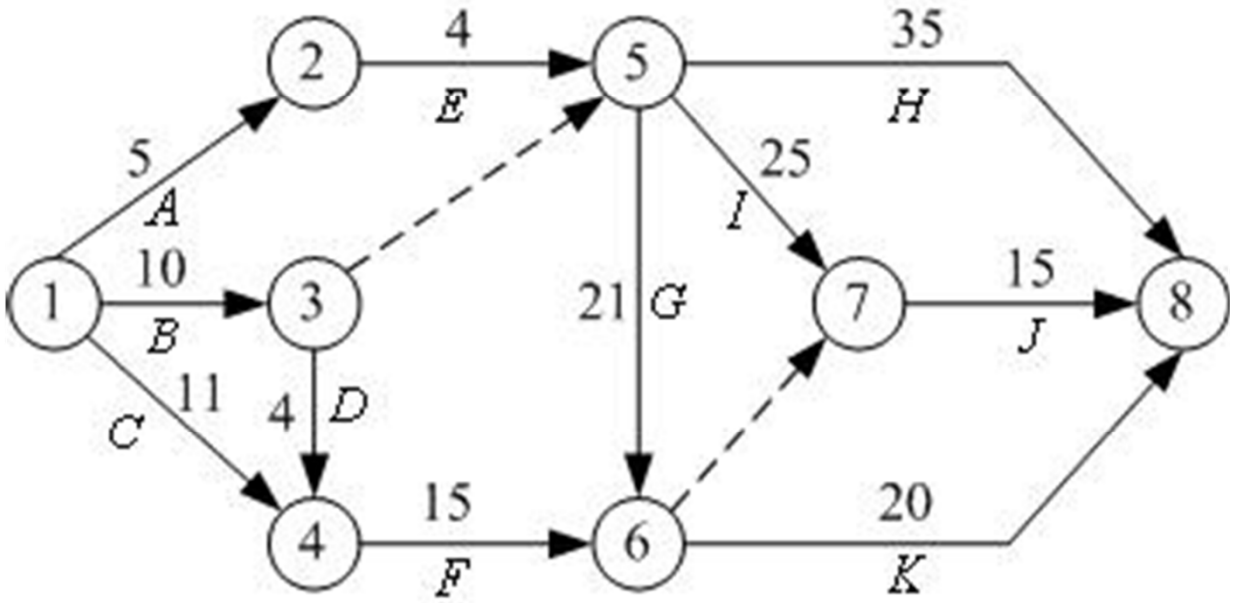

图 4.12　例 4.16 的计划网络图

2　写出相应的规划问题

设x_i是事件i的开始时间，1为最初事件，n为最终事件。希望总的工期最短，即极小化x_n，为了求所有事件的最早开工时间，把目标函数取为$\min \sum_{i \in V} x_i$。设t_{ij}是作业(i, j)的计划时间，因此，对于事件i与事件j有不等式

$$x_j \geq x_i + t_{ij},$$

由此得到相应的数学规划问题

$$\min \sum_{i \in V} x_i,$$

$$\text{s.t.} \quad x_j \geq x_i + t_{ij},\ (i, j) \in A,\ i, j \in V, \tag{4.9}$$

$$x_i \geq 0,\ i \in V,$$

其中V是所有的事件集合，A是所有作业的集合。

3　问题求解

计算结果给出了各个项目的开工时间，如$x_1=0$，则作业A,B,C的开工时间均是第 0 天；$x_2=5$，作业E的开工时间是第 5 天；$x_3=10$，则作业D的开工时间是第 10 天等等。每个作业只要按规定的时间开工，整个项目的最短工期为 51 天。

例 4.17（续例 4.16）求例 4.16 中每个作业的最早开工时间、最迟开工时间和作业的关键路径。

解　分别用x_i, z_i表示第i（$i=1,\cdots,8$）个事件的最早时间和最迟时间，t_{ij}表示作业(i,j)的计划时间，$es_{ij}, ls_{ij}, ef_{ij}, lf_{ij}$分别表示作业$(i,j)$的最早开工时间，最迟开工时间，最早完工时间，最晚完工时间。对应作业的最早开工时间与最迟开工时间相同，就得到项目的关键路径。

首先使用数学规划模型（4.9），求事件的最早开工时间x_i（$i=1,\cdots,8$）。然后用下面的递推公式求其它指标。

$$z_n = x_n, (\text{这里} n=8)$$

$$z_i = \min_j \{z_j - t_{ij}\}, \quad i = n-1,\cdots,1, (i,j) \in A \tag{4.10}$$

$$es_{ij} = x_i, (i,j) \in A \tag{4.11}$$

$$lf_{ij} = z_j, (i,j) \in A \tag{4.12}$$

$$ls_{ij} = lf_{ij} - t_{ij}, (i,j) \in A \tag{4.13}$$

$$ef_{ij} = x_i + t_{ij}, (i,j) \in A \tag{4.14}$$

使用公式（4.11）和（4.13）可以得到所有作业的最早开工时间和最迟开工时间，如表 4.6 所示，方括号中第 1 个数字是最早开工时间，第 2 个数字是最迟开工时间。

表 4.6　作业数据

作业(i,j)	开工时间（天）	计划完成时间（天）	作业(i,j)	开工时间（天）	计划完成时间（天）
$A(1,2)$	[0,1]	5	$G(5,6)$	[10,10]	21
$B(1,3)$	[0,0]	10	$H(5,8)$	[10,16]	35
$C(1,4)$	[0,5]	11	$I(5,7)$	[10,11]	25
$D(3,4)$	[10,12]	4	$J(7,8)$	[35,36]	15
$E(2,5)$	[5,6]	4	$K(6,8)$	[31,31]	20
$F(4,6)$	[14,16]	15			

从表 4.6 可以看出，当最早开工时间与最迟开工时间相同时，对应的作业在关键路线上，因此可以画出计划网络图中的关键路线，如图 4.13 粗线所示。关键路线为 1→3→5→6→8。

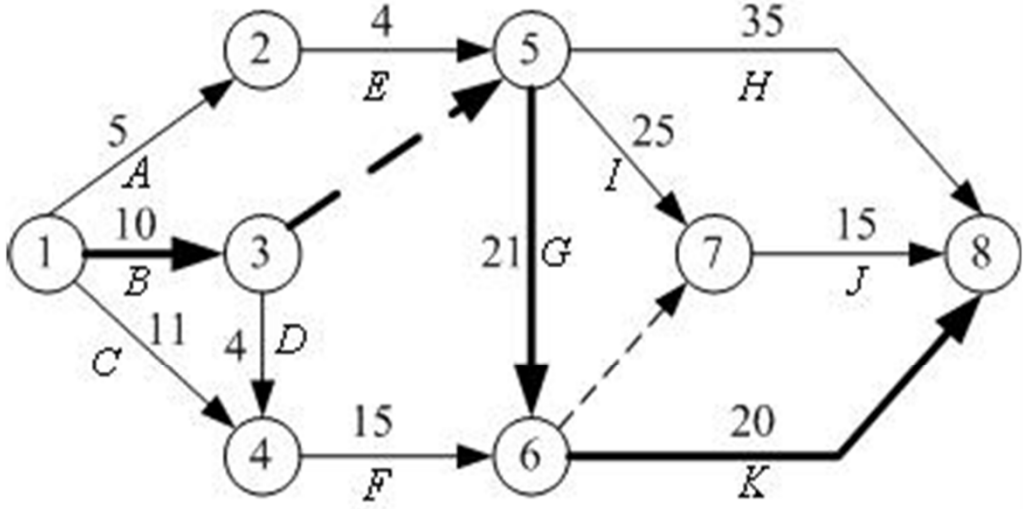

图 4.13　带有关键路线的计划网络图

4　将关键路线看成最长路

如果将关键路线看成最长路，则可以按照求最短路的方法（将求极小改为求极大）求出关键路线。

设x_{ij}为 0-1 变量，当作业(i,j)位于关键路线上取 1，否则取 0。数学规划问题写成

$$\max \sum_{(i,j)\in A} t_{ij}x_{ij},$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1, & i=1, \\ -1, & i=n, \\ 0, & i\neq 1,n, \end{cases}$$

$$x_{ij}=0 \text{ 或 } 1,\ (i,j)\in A.$$

例 4.18　用最长路的方法，求解例 4.16。

求得工期需要 51 天，关键路线为 1→3→5→6→8。

4.8.4　关键路线与计划网络的优化

例 4.19(关键路线与计划网络的优化)　假设例 4.16 中所列的工程要求在 49 天内完成。为提前完成工程，有些作业需要加快进度，缩短工期，而加快进度需要额外增加费用。表 4.7 列出例 4.16 中可缩短工期的所有作业和缩短一天工期额外增加的费用。现在的问题是，如何安排作业才能使额外增加的总费用最少。

表 4.7　工程作业数据

作业 (i,j)	计划完成时间（天）	最短完成时间（天）	缩短 1 天增加的费用（元）	作业 (i,j)	计划完成时间（天）	最短完成时间（天）	缩短 1 天增加的费用（元）
$B(1,3)$	10	8	700	$H(5,8)$	35	30	500
$C(1,4)$	11	8	400	$I(5,7)$	25	22	300
$E(2,5)$	4	3	450	$J(7,8)$	15	12	400
$G(5,6)$	21	16	600	$K(6,8)$	20	16	500

1　计划网络优化的数学表达式

设x_i是事件i的开始时间，t_{ij}是作业(i,j)的计划时间，m_{ij}是完成作业(i,j)的最短时间，y_{ij}是作业(i,j)可能减少的时间，c_{ij}是作业(i,j)缩短一天增加的费用，因此有

$$x_j - x_i \geq t_{ij} - y_{ij} \quad \text{且} \quad 0 \leq y_{ij} \leq t_{ij} - m_{ij}.$$

设d是要求完成的天数，1 为最初事件，n为最终事件，所以有$x_n - x_1 \leq d$。而问题的总目标是使额外增加的费用最小，即目标函数为$\min \sum_{(i,j)\in A} c_{ij} y_{ij}$。

由此得到相应的数学规划问题

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij},$$

$$\text{s.t.} \quad x_j - x_i + y_{ij} \geq t_{ij}, \ (i,j)\in A, \ i,j\in V,$$

$$x_n - x_1 \leq d,$$

$$0 \leq y_{ij} \leq t_{ij} - m_{ij}, \ (i,j)\in A, \ i,j\in V.$$

.2　计划网络优化的求解

作业(1,3)(B)）压缩 1 天的工期，作业(6,8)(K)压缩 1 天工期，这样可以在 49 天完工，需要多花费 1200 元。

例 4.20 （续例 4.19） 用 LINGO 软件求解例 4.19，并求出相应的关键路径、各作业的最早开工时间和最迟开工时间。

解 使用前面例子相同符号。为了得到作业的最早开工时间，在目标函数中加入$\sum_{i\in V} x_i$，建立如下的数学规划模型

$$\min \sum_{(i,j)\in A} c_{ij}y_{ij} + \sum_{i\in V} x_i ,$$

$$\text{s.t.} \quad x_j - x_i + y_{ij} \geq t_{ij}, \ (i,j)\in A, \ i,j\in V ,$$

$$x_n - x_1 \leq d ,$$

$$0 \leq y_{ij} \leq t_{ij} - m_{ij}, \ (i,j)\in A, \ i,j\in V .$$

先求出x_i, y_{ij}，其中$i \in V$，$(i,j) \in A$。

再使用迭代公式

$$z_n = x_n, (\text{这里} n = 8),$$

$$z_i = \min_j \{z_j - t_{ij} + y_{ij}\}, \quad i = n-1, \cdots, 1, (i,j) \in A,$$

$$es_{ij} = x_i, \ (i,j) \in A,$$

$$ls_{ij} = z_j - t_{ij} + y_{ij}, \ (i,j) \in A.$$

求出事件最迟时间z_i，作业最早开工时间es_{ij}，最迟开工时间ls_{ij}。

计算出所有作业的最早开工时间和最迟开工时间见表 4.8。

表 4.8　作业数据

作业(i,j)	开工时间	实际完成时间（天）	作业(i,j)	开工时间	实际完成时间（天）
$A(1,2)$	[0,0]	5	$G(5,6)$	[9,9]	21
$B(1,3)$	[0,0]	9	$H(5,8)$	[9,14]	35
$C(1,4)$	[0,4]	11	$I(5,7)$	[9,9]	25
$D(3,4)$	[9,12]	4	$J(7,8)$	[34,34]	15
$E(2,5)$	[5,5]	4	$K(6,8)$	[30,30]	19
$F(4,6)$	[13,15]	15			

当最早开工时间与最迟开工时间相同时，对应的作业就在关键路线上，图 4.14 中的粗线表示优化后的关键路线。从图 4.14 可以看到，关键路线不止一条。

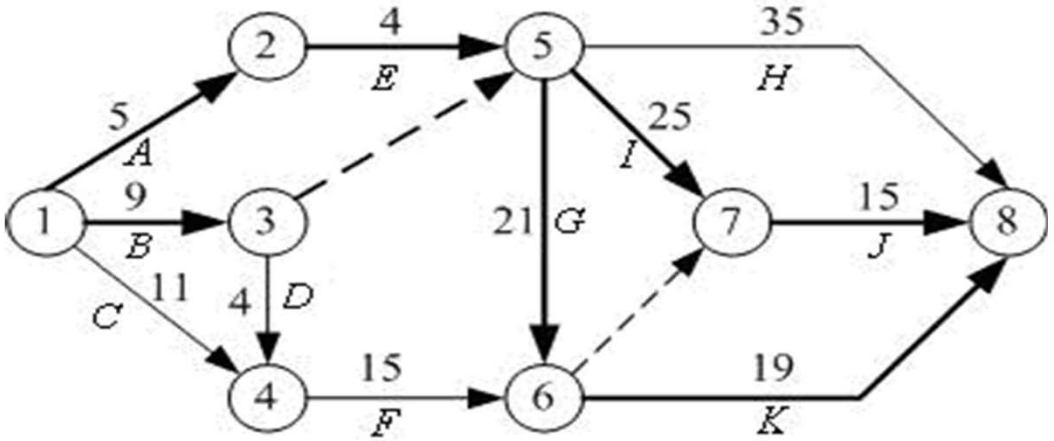

图 4.14　优化后的关键路线图

4.8.5　完成作业期望和实现事件的概率

在例 4.16 中，每项作业完成的时间均看成固定的，但在实际应用中，每一工作的完成会受到一些意外因素的干扰，一般不可能是完全确定的，往往只能凭借经验和过去完成类似工作需要的时间进行估计。通常情况下，对完成一项作业可以给出三个时间上的估计值：最乐观值的估计值（a），最悲观的估计值（b）和最可能的估计值（m）。

设t_{ij}是完成作业(i,j)的实际时间（是一随机变量），通常用下面的方法计算相应的数学期望和方差，

$$E(t_{ij})=\frac{a_{ij}+4m_{ij}+b_{ij}}{6}, \tag{4.15}$$

$$\mathrm{var}(t_{ij})=\frac{(b_{ij}-a_{ij})^2}{36}. \tag{4.16}$$

设T为实际工期，即

$$T=\sum_{(i,j)\in\text{关键路线}} t_{ij}, \tag{4.17}$$

由中心极限定理，可以假设T服从正态分布，并且期望值和方差满足

$$\bar{T}=E(T)=\sum_{(i,j)\in\text{关键路线}} E(t_{ij}), \tag{4.18}$$

$$S^2=\mathrm{var}(T)=\sum_{(i,j)\in\text{关键路线}} \mathrm{var}(t_{ij}). \tag{4.19}$$

设规定的工期为d，则在规定的工期内完成整个项目的概率为

$$P\{T \le d\} = \Phi\left(\frac{d-\bar{T}}{S}\right). \qquad (4.20)$$

@psn(x)是 LINGO 软件提供的标准正态分布函数，即

$$@\text{psn}(x) = \Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt. \qquad (4.21)$$

例 4.21　已知例 4.16 中各项作业完成的三个估计时间如表 4.9 所示。如果规定时间为 52 天，求在规定时间内完成全部作业的概率。进一步，如果完成全部作业的概率大于等于 95%，那么工期至少需要多少天?

表 4.9　作业数据

作业 (i,j)	估计时间(天)			作业 (i,j)	估计时间(天)		
	a	m	b		a	m	b
$A(1,2)$	3	5	7	$G(5,6)$	18	20	28
$B(1,3)$	8	9	16	$H(5,8)$	26	33	52
$C(1,4)$	8	11	14	$I(5,7)$	18	25	32
$D(3,4)$	2	4	6	$J(7,8)$	12	15	18
$E(2,5)$	3	4	5	$K(6,8)$	11	21	25
$F(4,6)$	8	16	18				

解　对于这个问题采用最长路的方法。

按公式（4.15）计算出各作业的期望值，再建立求关键路径的数学规划模型

$$\max \sum_{(i,j)\in A} E(t_{ij})x_{ij},$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1, & i=1, \\ -1, & i=n, \\ 0, & i\neq 1,n, \end{cases}$$

$$x_{ij}=0 \text{ 或 } 1,\ (i,j)\in A.$$

求出关键路径后，再由公式（4.16）计算出关键路线上各作业方差的估计值，最后利用分布函数@psn，即可计算出完成作业的概率与完成整个项目的时间。

计算得到关键路线的时间期望为 45.3 天，标准差为 2.14，在 52 天完成全部作业的概率为 99.9%，如果完成全部作业的概率大于等于 95％，那么工期至少需要 48.8 天。