图论与网络模型

Graph and Network Modeling

主讲人: 薛 毅 教授

北京工业大学应用数理学院

运筹学学科部主任

xueyi@bjut.edu.cn

第六章
图论与网络模型

1 图的基本概念

1.1. 从Königsberg（哥尼斯堡）七桥问题谈起

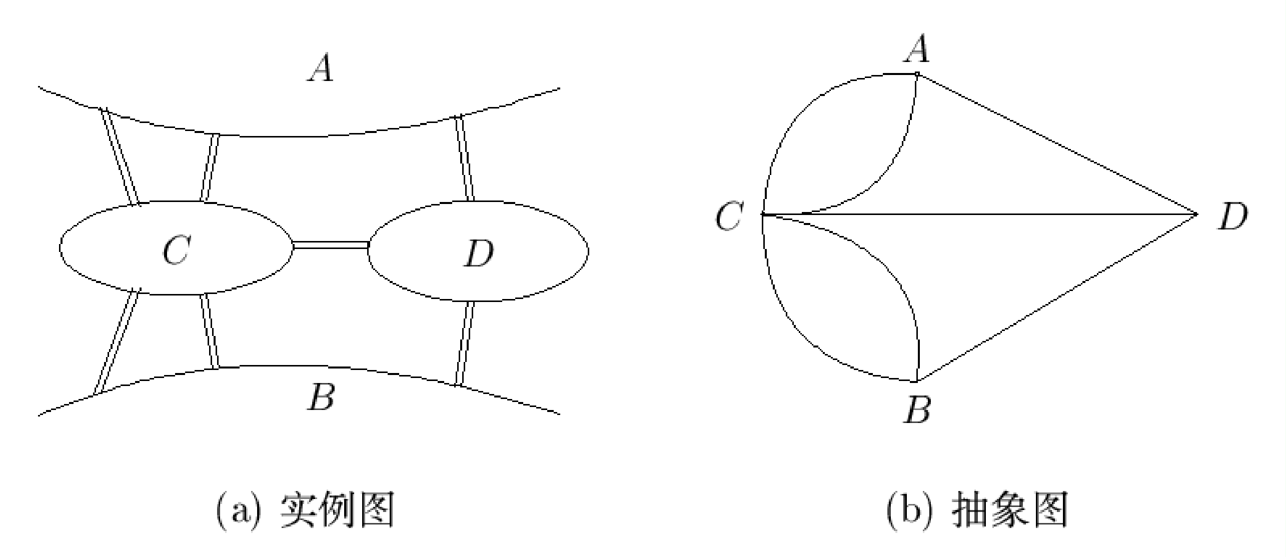


图6.1.1 Königsberg七桥问题

Euler（欧拉）在1736年发表的论文中, 解决了著名的Königsberg七桥问题, 证明这是不可能的. 人们将Euler的论文作为图论的第一篇论文, 从此逐渐形成了一个具有广泛应用的运筹学分支—图论. 图论目前已广泛地应用在物理学、化学、控制论、信息论、科学管理、电子计算机等各个领域, 特别随着计算机的发展, 图论的理论与应用得到了进一步地发展.

1.2. 图的基本概念

从直观上看, 所谓图是由点和边组成的图形, 如图6.1.2和图6.1.3所示. 下面我们给出图的定义.

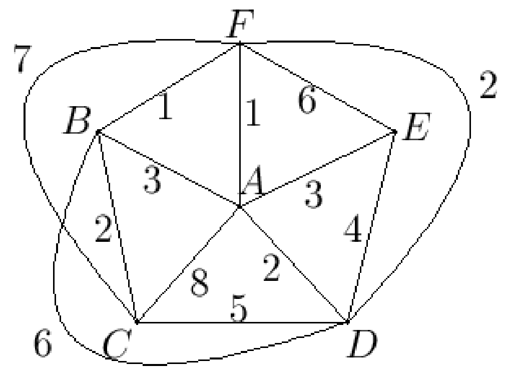


图 6.1.2

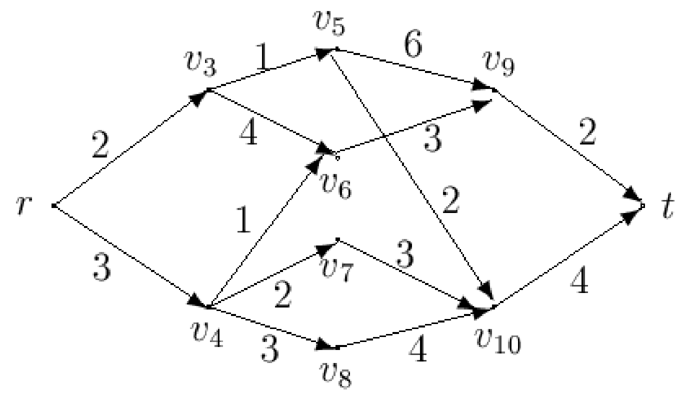


图 6.1.3

定 义6.1.1 图G是一个偶对(V, E)，其中$V = \{v_1, v_2, \cdots, v_n\}$是有限集, 其元素称为顶点, $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$, E中的元素称为边. 若图中边是有序对(v_i, v_j), 则称图为有向图, 称$e = (v_i, v_j)$是从v_i连接v_j, 称v_i为e的尾, 称v_j为e的头. 若去掉有向图的方向, 得到的图称为基础图.

定义6.1.2 如果$G'(V', E')$是一个图, 并且$V' \subset V$, $E' \subset E$, 则称G'是$G(V, E)$的子图. 对于图$G(V, E)$, 如果$(v_i, v_j) \in E$, 赋以一个实数$w(v_i, v_j)$, 则称$w(v_i, v_j)$为边(v_i, v_j)的权, G连同边上的权称为赋权图.

定义6.1.3 如果$(v_i, v_j) \in E$, 则称v_j与v_i邻接, 具有n个顶点的图的邻接矩阵是一个$n \times n$阶矩阵$A = (a_{ij})_{n\times n}$, 其分量为

$$a_{ij} = \begin{cases} 1, & \text{如果}(v_i, v_j) \in E, \\ 0, & \text{其他}. \end{cases}$$

n个顶点的赋权图的赋权矩阵是一个$n \times n$阶矩阵$W = (w_{ij})_{n\times n}$, 其分量为

$$w_{ij} = \begin{cases} w(v_i, v_j), & \text{如果}(v_i, v_j) \in E, \\ \infty, & \text{其他}. \end{cases}$$

例6.1.1 (a) 图6.1.4是一个图, 构造它的邻接矩阵A, 并计算A^2.

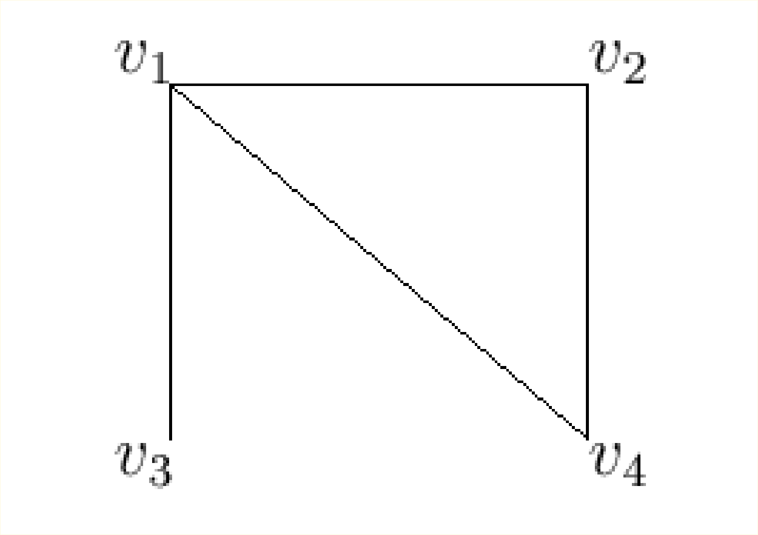


图6.1.4

$$A=\begin{bmatrix}0&1&1&1\\1&0&0&1\\1&0&0&0\\1&1&0&0\end{bmatrix},\quad A^2=\begin{bmatrix}3&1&0&1\\1&2&1&1\\0&1&1&1\\1&1&1&2\end{bmatrix}.$$

矩阵A^2有它的几何意义：A^2的元素$A^2(i,j)$表示从顶点i到顶点j长度为2的路径的数目（后面将详细给出两顶点间的路径以及其长度的严格定义）。

(b) 图6.1.2是一个图其赋权矩阵为

$$W = \begin{bmatrix} \infty & 1 & 3 & 2 & 8 & 3 \\ 1 & \infty & 6 & 2 & 7 & 1 \\ 3 & 6 & \infty & 4 & \infty & \infty \\ 2 & 2 & 4 & \infty & 5 & 6 \\ 8 & 7 & \infty & 5 & \infty & 2 \\ 3 & 1 & \infty & 6 & 2 & \infty \end{bmatrix}.$$

定义6.1.4 无向图G的一条途径是指一个有限的非空序列$W = v_0, e_1, v_1, e_2, \cdots, e_m, v_m$，它的项交替的顶点和边. 称$m$为$W$的长. 若途径的边$e_1, e_2, \cdots, e_m$互不相同, 则称$W$为迹. 若顶点$v_0, v_1, \cdots, v_m$互不相同, 则称$W$为路. 如果$v_0 = v_m$, 并且没有其他相同的顶点, 则称为圈.

再次考察图6.1.4, v_1, v_2, v_4是一条路, 而v_2, v_4, v_1, v_2是一个圈, 其圈的长度为3.

对于有向图G, 若$W = v_0, e_1, v_1, e_2, \cdots, e_m, v_m$, 且$e_i$有头$v_i$和尾$v_{i-1}$, 则称$W$ 为有向途径, 类似地, 可以定义有向迹, 有向路和有向圈.

完全图的定向图称为竞赛图.

定理6.1.1 如果存在u到v的途径，则一定存在u到v的路. 如果图G的顶点个数为n, 则这个路的长度小于等于$n-1$.

定 义6.1.5 设$G(V,E)$是无向图, 若$(u,v)\in E$, 则称(u,v)与u关联. 设$v\in V$, 顶点v的度定义为与v关联的边数, 记作$d(v)$. 称度为奇数的顶点为奇点, 称度为偶数的顶点为偶点.

在图6.1.4中, $d(v_1)=3$, $d(v_2)=2$.

在图$G(V,E)$中, 边(u,v)与u关联, 也与v关联, 因此, 边(u,v)对$d(u)$贡献一次, 对$d(v)$贡献一次, 故得到如下定理.

定理6.1.2

$$\sum_{v\in V} d(v) = 2\,\varepsilon,$$

其中ε表示图G的边数.

推论： 任何图中, 奇点的个数为偶数.

1.3. 连通性

定义6.1.6 如果图G中任意两点u, v, 均存在一条连接u, v的路, 则称图G是连通的, 特别, 对于有向图, 称为强连通的, 如果去掉有向图的方向, 其图是连通的, 则称该有向图是弱连通的.

例6.1.2

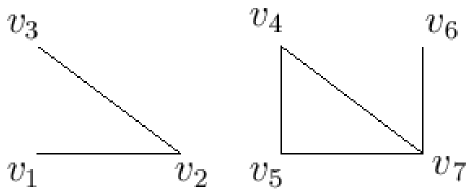


图 6.1.5 (a) 无向图

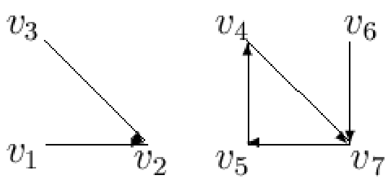


图 6.1.5 (b) 有向图

定理6.1.2 设A是n个顶点图G的邻接矩阵，则$A^m(i,j)$是从顶点i到顶点j长度为m的途径的数目. 因此，图G是连通的充分必要条件是: 对任意的(i,j), 存在$1 \le k \le n$, 使得$A^k(i,j) \ge 1$.

定义6.1.7 对于图G，如果存在$m > 0$，使得$A^m(i,j) > 0, \forall i,j \in V$, 则称图$G$是正则的.

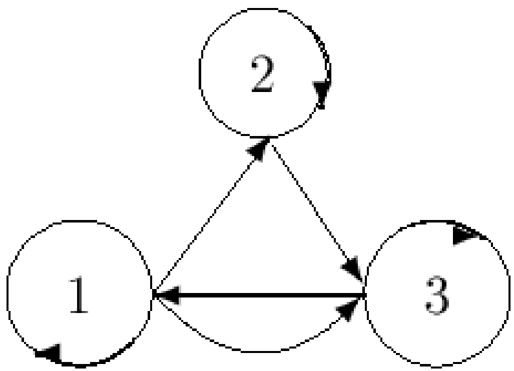


图 6.1.6 正则图

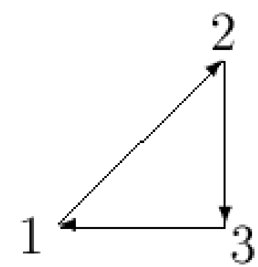


图 6.1.7 非正则图

定义6.1.8 称图G的最大连通子图G'为图G的连通分支.

1.4. 最短路问题

若H是赋权图的一个子图，则H的权$w(H)$是指它的各边的权和 $\sum\limits_{e\in E(H)} w(e)$. 所谓最短路问题就是找出赋权图中指定两点$u_0, v_0$之间的最小权路. 这里我们假定$w(e)\geq 0, \forall e\in E$.

为了叙述清楚起见，把赋权图中一条路的权称为它的长，把(u,v)路的最小权称为u和v之间的距离，并记作$d(u,v)$. 在下面的算法中，我们假定所有的权均为正，并且若$(u,v)\notin E$，则规定$w(u,v)=+\infty$.

Dijkstra(1959)以及Whiting和Hillier(1960)各自独立发现了最短路算法。目前使用的最短路算法是由Dijkstra改进得到的，因此称为Dijkstra算法，这个算法不仅找到了最短的(u_0, v_0)路, 而且给出了从u_0到G的所有其他顶点的最短路.

算法6.1.2 (Dijkstra算法)

(1) 置$l(u_0)=0$对$v\neq u_0$, $l(v)=\infty$, $S_0=\{u_0\}$且$i=0$.

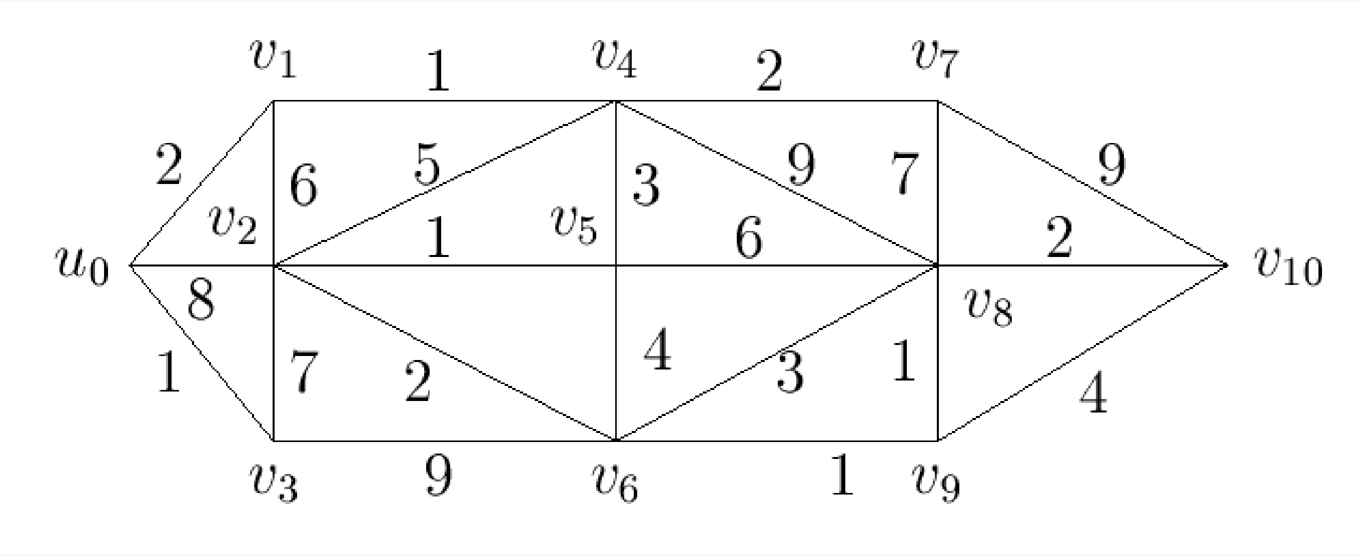
(2) 对每个$v\in\bar{S}_i$, 置

$$l(v)=\min\left\{l(v),\min_{u\in S_i}\{d(u_0,u)+w(u,v)\}\right\}.$$

计算$\min\limits_{v\in\bar{S}}\{l(v)\}$, 并把达到这个最小值的一个顶点记为$u_{i+1}$, 置$S_{i+1}=S_i\bigcup\{u_{i+1}\}$.

(3) 若$i=n-1$, 则停止; 否则置$i=i+1$, 转(2).

例6.1.3 用Dijkstra算法求下图中u_0到v_i的各点的最短路。



运用Dijkstra算法标出点u_0到v_i各点的最短路，并将相应的最短路标在方框内，用粗线标明u_0到v_i的最短路，具体结果如图6.1.8所示.

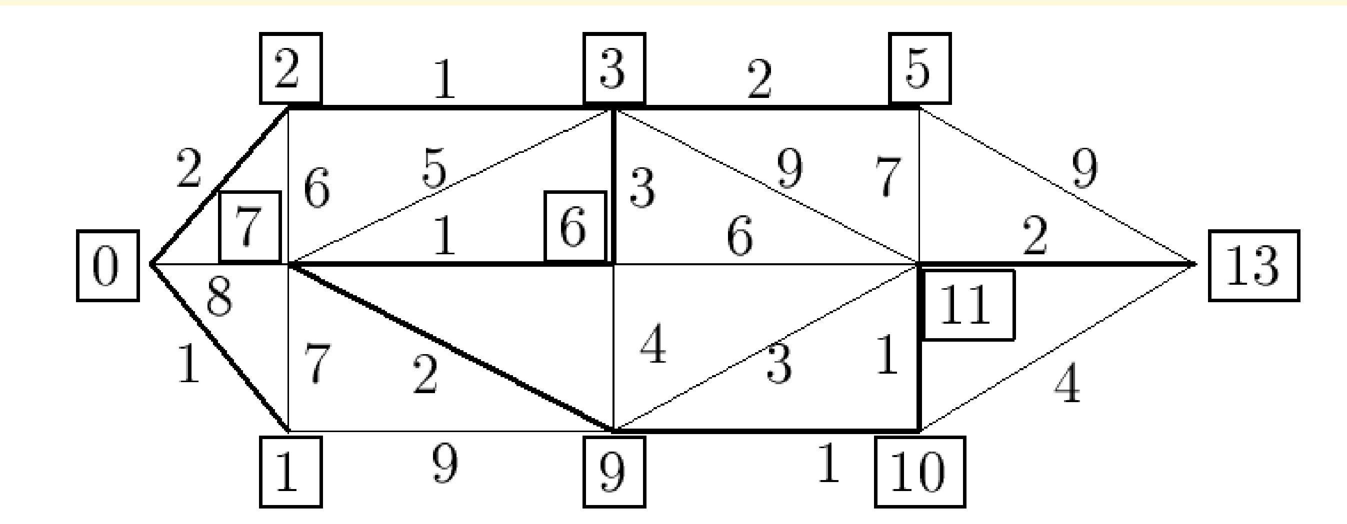


图6.1.8 最短路算法

访问主页
标题页
◀◀ ▶▶
◀ ▶
第 24 页 共 134 页
返回
全屏显示
关闭
退出

2 Euler环游和Hamilton圈

定义6.2.1 图G的环游是指经过G的每条边至少一次的闭途径. Euler环游是指经过每条边恰好一次的环游. 一个图若包含Euler环游, 则称该图为Euler图.

定理6.2.1 一个非空连通图是Euler图的必要充分条件是: 没有奇点.

从直观上看, 由于每条边仅经过一次回到原点, 因此, 对于某个顶点, 有一个进来的边就有一个出去的边; 同样, 有一条出去的边就会有一条进来的边.

由定理6.2.1可知, Königsberg 七桥问题无解, 因为该图的4个顶点度全是奇数. 由定理6.2.1, 也可以得到图一笔画的必要充分条件是: 它至多有两个奇点. (不必回到出发点).

2.1. Hamilton (哈密顿)圈

定 义6.2.2 包 含 图G的 每 个 顶 点 的 路 称 为Hamilton路, 包 含 图G的 每 个 顶 点 的 圈 称 为Hamilton圈. 一个图若包含Hamilton圈, 则称这个图为Hamilton图.

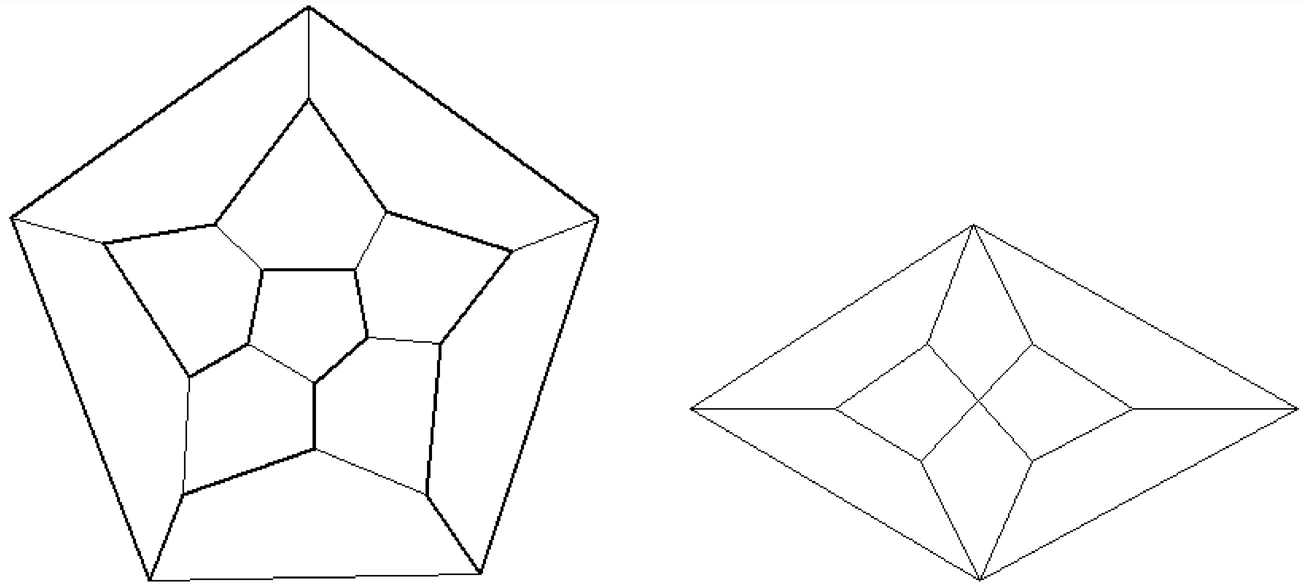


图 6.2.1(a) Hamilton 图, 图 6.2.1 (b) 非 Hamilton 图

2.2. 中国邮递员问题

邮递员的工作是: 在邮局里选出邮件, 递送邮件, 然后再返回邮局. 自然, 他必须走过他投递范围内的每一条街道至少一次. 在这个前提下, 希望选择一条尽可能短的路线. 该问题是由中国数学家管梅谷首次研究的, 因此称为中国邮递员问题.

中国邮递员问题本质上是在非负赋权连通图中找出一个最小权的环游, 称这种环游为最优环游.

如果非负赋权图G是Euler图，我们只要找出它的Euler环游，就得到最优环游，因为Euler环游每条边仅经过一次. Fleury算法提供了一个求Euler环游的好算法.

如果非负赋权图G不是Euler图，我们用添加重复边的方法，使图G扩充为一个Euler图G^*，并要求

$$\sum_{e \in E(G^*) \setminus E(G)} w(e) \quad \text{尽可能小},$$

再求G^*的Euler环游，这就得到中国邮递员问题的解.

为了理解上述方法的思想, 我们举一个简单的例子, 在例中, 图G仅有两个奇点. (见图6.2.2, 其中图6.2.2 (b)中的曲线是增加的边).

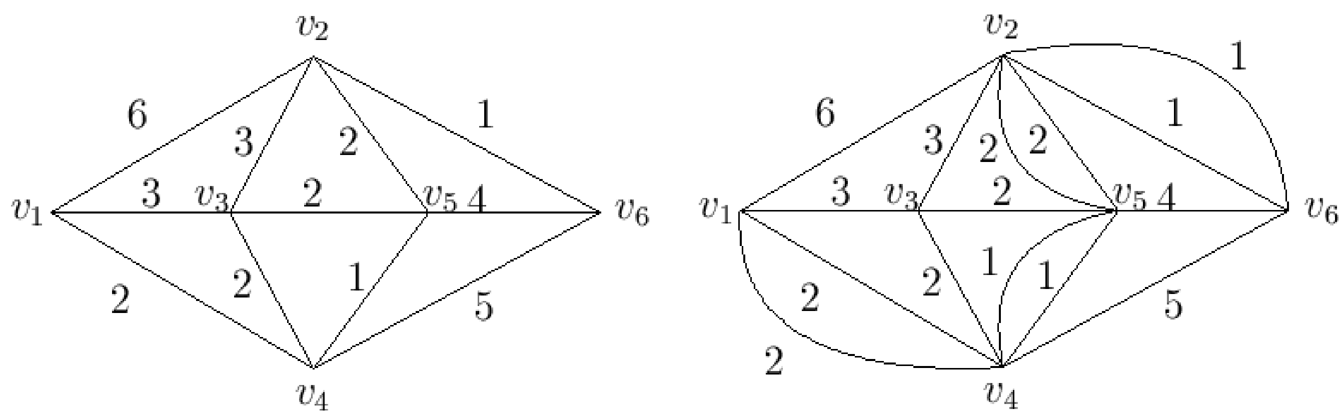


图 6.2.2 (a) 图 G　　图 6.2.2 (b) 图 G^*

图的基本概念
Euler环游和Hamilton圈
树和生成树
最大匹配
最大流问题
用LINGO软件包求解组合...
实例分析— 通讯网络的最...
访问主页
标题页
第 30 页 共 134 页
返回
全屏显示
关闭
退出

2.3. 旅行商问题

一个旅行商想去走访若干城市, 然后回到他的出发地. 给定各城市之间所需的旅行时间后, 怎样计划他的路线, 使他能对每个城市恰好进行一次访问, 而总时间最短, 这个问题称为旅行商问题(或货郎担问题).

旅行商问题就是在一个赋权图中, 找出最小权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, 因此只能找一种求出相当好(不一定最优)的解.

一个可行的办法是先求一个Hamilton圈C, 然后适当修改C, 得到具有较小权的另一个Hamilton圈.

设$C = v_1v_2\cdots v_nv_1$，则对于所有适合$1 < i+1 < j < n$的i和j，可以得到一个新Hamilton圈。如图6.2.3所示，$C_{ij} = v_1v_2\cdots v_iv_jv_{j-1}\cdots v_{i+1}v_{j+1}v_{j+2}\cdots v_nv_1$，它是由$C$中删去边$v_iv_{i+1}$和$v_jv_{j+1}$添加$v_iv_j$和$v_{i+1}v_{j+1}$得到的.

若对于某一对i和j, 有

$$w(v_iv_j)+w(v_{i+1}v_{j+1})<w(v_iv_{i+1})+w(v_jv_{j+1}),$$

则圈C_{ij}将是圈C的一个改进. 在接连进行上述的一系列修改之后, 最后得到的一个圈不能再用此法改进了. 这个最后的圈几乎可以肯定不是最优的, 但有理由认为它是比较好的.

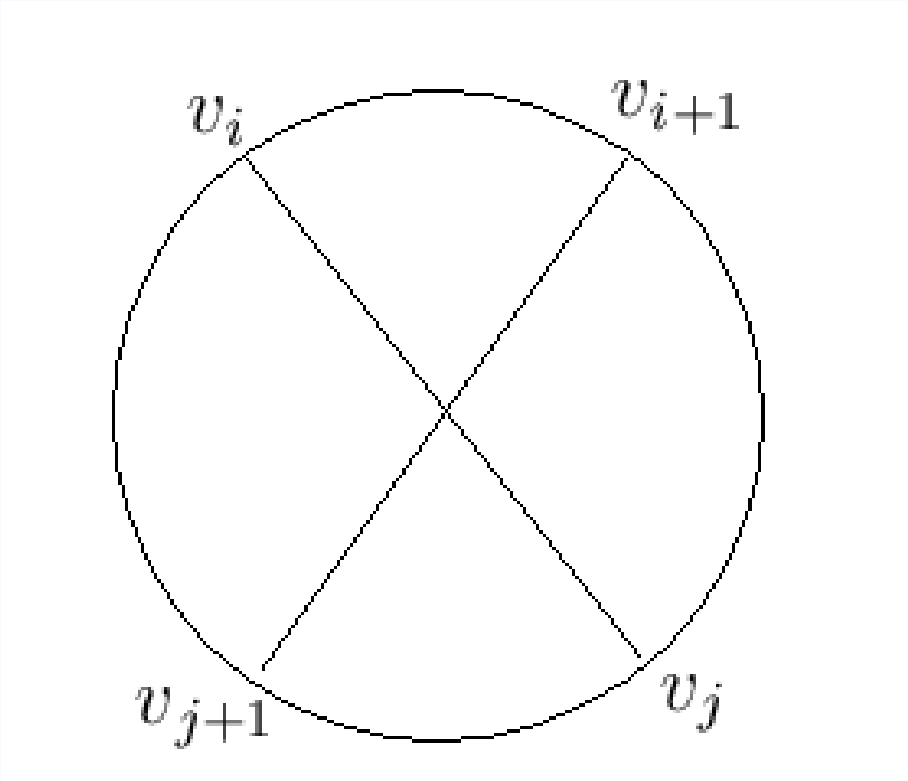


图6.2.3 改进Hamilton圈的示意图

2.4. 工件排序问题

设某台机器必须加工多种工件$J_1, J_2, \cdots, J_n$, 在一种工件加工完毕之后, 为了加工下一种工件, 机器必须调整. 如果从工件J_i到工件J_j的调整时间为t_{ij}, 求这些工件的一个排序, 使整个机器的调整时间最少.

这个问题显然与旅行商问题有关, 关于它的求解, 现在还没有已知有效的方法. 所以希望有一个方法来得到一个相当好的解(未必是最优解).

具体作法如下:

(1) 构造具有n个顶点$v_1, v_2, \cdots, v_n$的有向图G, 当$t_{ij} \leq t_{ji}$时, 有$(v_i, v_j) \in E$.

(2) 求G的有向Hamilton路$(v_{i_1}, v_{i_2}, \cdots, v_{i_n})$, 并且把工件相应的排列好.

作为一个例子，假设存在六个工件J_1, J_2, J_3, J_4, J_5和J_6, 并且调整矩阵是

$$
\begin{array}{ccccccc}
 & J_1 & J_2 & J_3 & J_4 & J_5 & J_6 \\
J_1 & 0 & 5 & 3 & 4 & 2 & 1 \\
J_2 & 1 & 0 & 1 & 2 & 3 & 2 \\
J_3 & 2 & 5 & 0 & 1 & 2 & 3 \\
J_4 & 1 & 4 & 4 & 0 & 1 & 2 \\
J_5 & 1 & 3 & 4 & 5 & 0 & 5 \\
J_6 & 4 & 4 & 2 & 3 & 1 & 0
\end{array}
$$

序列$J_1 \to J_2 \to J_3 \to J_4 \to J_5 \to J_6$的调整时间需要13个单位. 为了找出一个较好的序列, 按(1)构造有向图G，如图6.2.4所示.

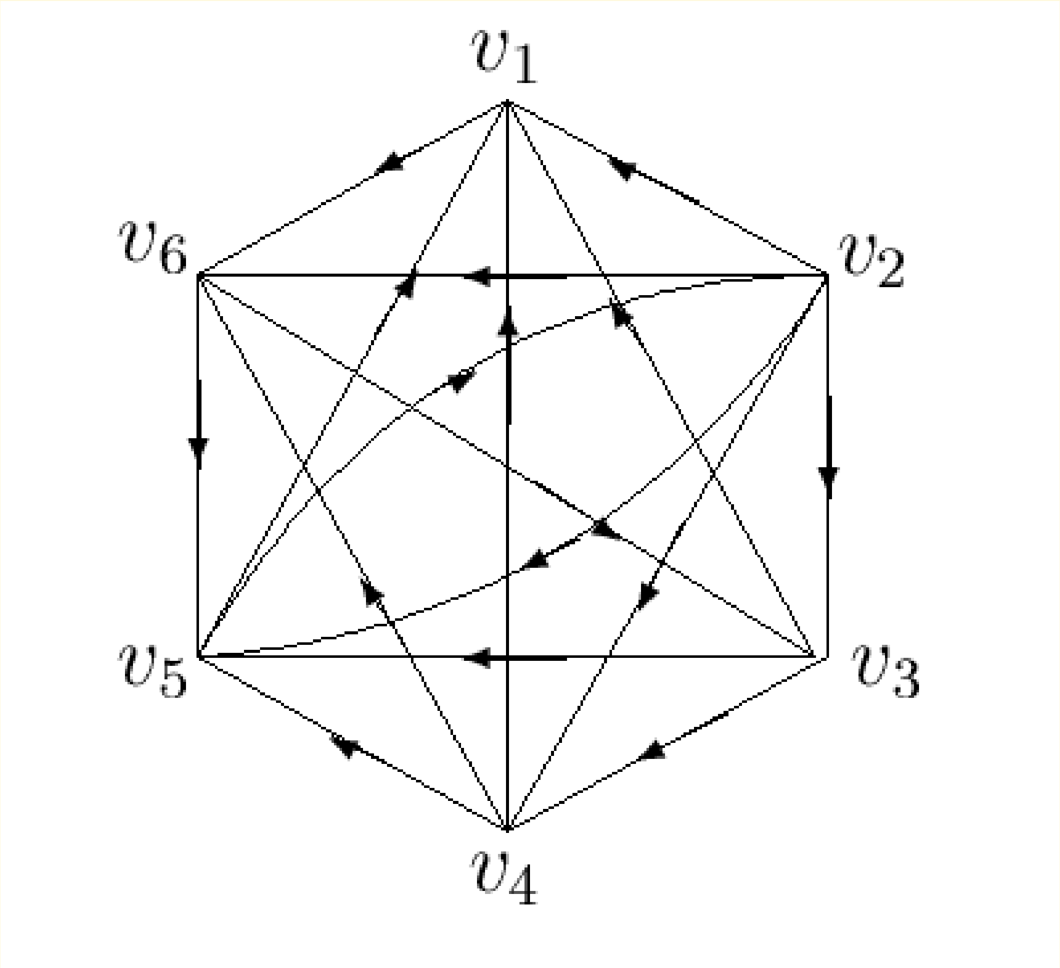


图6.2.4 按工件调整时间得到的有向图

$(v_1, v_6, v_3, v_4, v_5, v_2)$是$G$的有向Hamilton路, 因而产生序列

$$J_1 \longrightarrow J_6 \longrightarrow J_3 \longrightarrow J_4 \longrightarrow J_5 \longrightarrow J_2$$

它的调整时间只需要8个单位. 注意相反的序列

$$J_2 \longrightarrow J_5 \longrightarrow J_4 \longrightarrow J_3 \longrightarrow J_6 \longrightarrow J_1$$

是很差的, 其调整时间需要19个单位.

2.5. 竞赛参加者名次的排列

在一场体育竞赛中, 若干名选手两两互相竞赛. 得出竞赛成绩后, 应该怎样排参加者的名次呢?

例如, 考察图6.2.5中的竞赛图, 它代表六个选手之间的一场竞赛结果; 我们看到选手1打败选手2, 4, 5和6, 而输给选手3, 等等.

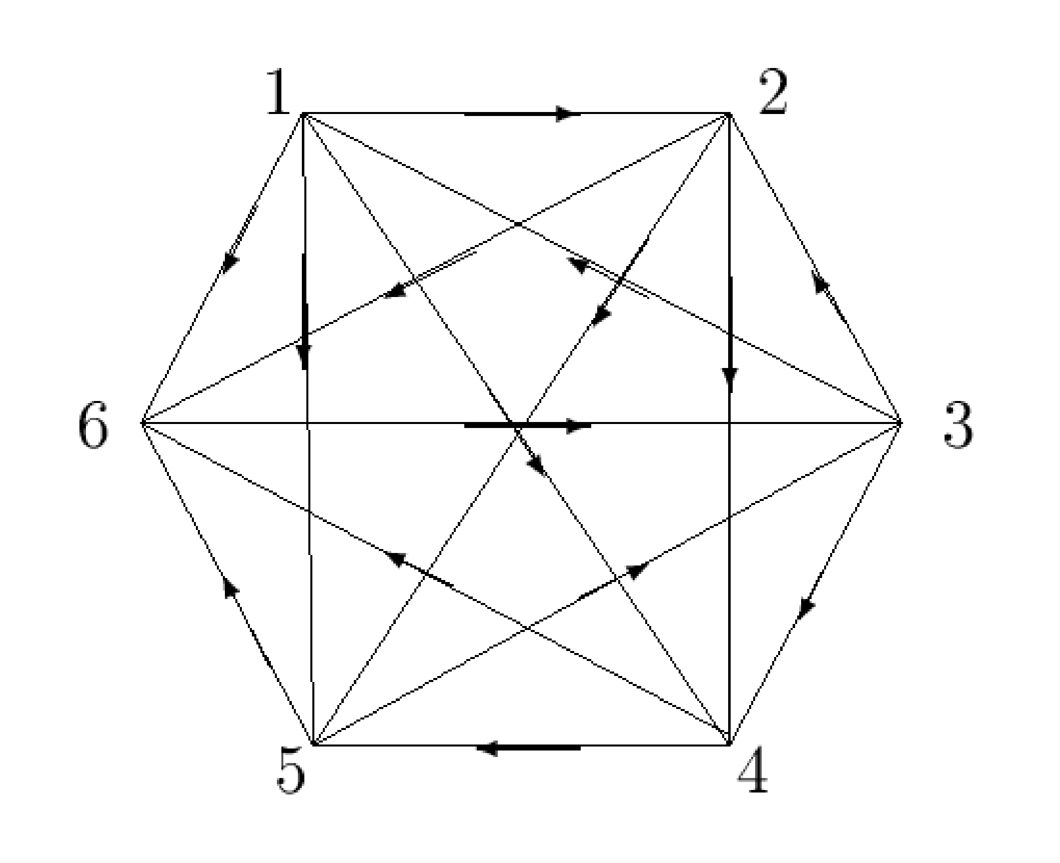


图6.2.5 竞赛图

排参加者名次的一种可能的办法是寻找这个竞赛图中的一条有向Hamilton路, 然后按照参加者对应的顶点在这条路中的位置排列名次. 例如, 有向Hamilton路

$$(3, 1, 2, 4, 5, 6)$$

表明选手3是冠军, 选手1是亚军, 等等.

可是这种排名次的方法经不起进一步的推敲, 因为一个竞赛图一般有许多条有向Hamilton路; 我们的例子中就有

$$(1, 2, 4, 5, 6, 3), \qquad (1, 4, 6, 3, 2, 5)$$

以及若干别的有向Hamilton路.

另一个办法是计算得分, 并比较它们. 在上面的例子中这样做, 得到的得分向量

$$s_1 = (4, 3, 3, 2, 2, 1).$$

这里有一个缺陷, 即这个得分向量不能区分选手2和选手3的高低, 尽管选手3打败了得分较高的选手. 于是我们导出第二级得分向量

$$s_2 = (8, 5, 9, 3, 4, 3),$$

其中每个选手的第二级得分是被他打败的选手的得分之和.

现在选手3名列第一. 继续这个程序, 得到进一步的向量,

$$s_3 = (15, 10, 16, 7, 12, 9),$$

$$s_4 = (38, 28, 32, 21, 25, 16),$$

$$s_5 = (90, 62, 87, 41, 48, 32),$$

$$s_6 = (183, 121, 193, 80, 119, 87).$$

看来选手名次的排列有点波动, 例如选手3和选手1竞争第一名的情况就是这样. 如何确定他们的名次呢? 看来需要考虑它的极限情况. 考虑图的邻接矩阵

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

构造列向量$e = (1,1,1,1,1,1)^T$.

由此得到$s_1 = Ae$, $s_2 = As_1 = A^2e$, $\cdots$, $s_i = A^ie$. 可以证明: 当竞赛图是双向连通的, 并且至少有四个顶点时, 极限存在, 即

$$\lim_{i\to\infty}\left(\frac{A}{\lambda}\right)^i e = s$$

这里λ是矩阵A的最大特征值, s是对应于λ的特征向量.

因此, 名次排列问题就转化为求邻接矩阵的最大特征值和对应的特征向量. 对于前面的问题, 其最大特征值为$\lambda = 2.232$, 相应的特征向量为

$$s = (0.238,\ 0.164,\ 0.231,\ 0.113,\ 0.150,\ 0.104).$$

即名次排列为

选手1, 选手3, 选手2, 选手5, 选手4, 选手6

3 树和生成树

3.1. 树

定义6.3.1 如果无向图是连通的, 且不包含有圈, 则称该图为树. 如果有向图中任何一个顶点都可由某一顶点v_1到达, 则称v_1为图G的根. 如果有向图G有根, 且它的基础图是树, 则称G是有向树.

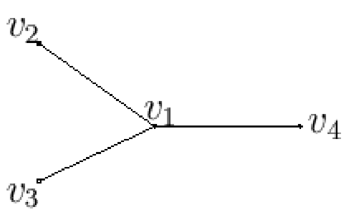


图 6.3.1 无向树

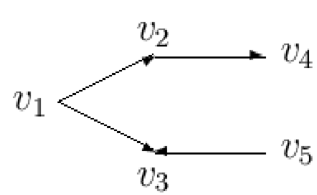


图 6.3.2 (a) 非有向树

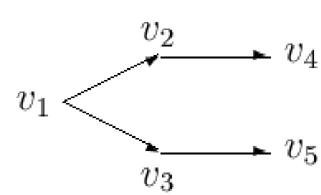


图 6.3.2 (b) 有向树

例6.3.1 图6.3.1是一棵树(无向树). 图6.3.2 (a)不是一棵有向树, 图6.3.2.(b)是一棵有向树.

定理6.3.1 设G是有限的无向图, 如果$d(v) \geq 2, \forall v \in V$, 则$G$有圈.

定理6.3.2 每棵树至少有一个顶点的度为1.

定理6.3.3 设G是连通图，且边数$<$顶点数，则图G中至少有一个顶点的度为1.

定理6.3.4 设G是具有n个顶点的无向连通图，G是树的必要充分条件是：G有$n-1$条边.

定义6.3.2 若G'是包含G的全部顶点的子图，它又是树，则称G'是生成树.

3.2. 无向生成树

对于图6.1.2, 我们可以找到许多生成树. 例如

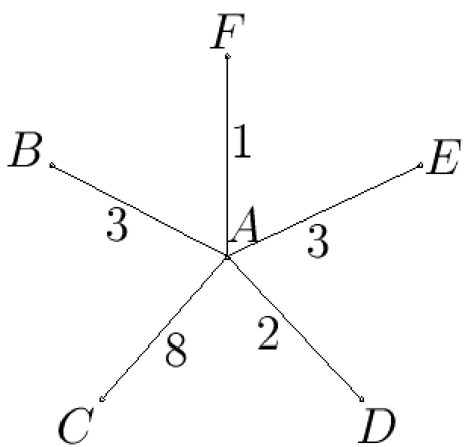


图 6.3.3 (a) 生成树

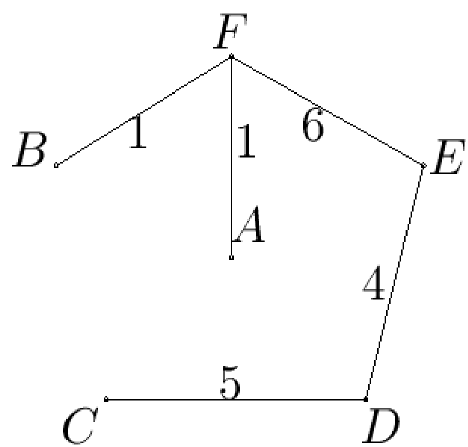


图 6.3.3 (b) 生成树

我们感兴趣的问题是: (1) 如何找生成树; (2) 如何找具最小权的生成树.

定理6.3.5 如果无向图G是有限的、连通的, 则在G中存在生成树.

定理6.3.5给出了构造生成树的一个简单算法, 这个算法就是在一个连通圈中破掉所有的圈, 剩下不含圈的连通图, 就是一棵生成树. 这个算法给它一个形象的名称叫作"破圈法".

在图中任取一条边e_1, 找一条不与e_1构成圈的边e_2, 然后再找一条不与$\{e_1, e_2\}$构造圈的边e_3, 这样继续下去, 直到这种过程不能进行, 这时所得到的图就是一棵生成树. 我们称这种方法为“避圈法”.

3.3. 连线问题

假设要建造一个连接若干城市的通讯网. 已知城市v_i与城市v_j之间通讯线路所需费用为c_{ij}, 问应在哪些城市之间架设线路, 即能使所有城市之间都能连通, 又要求架设的总费用最小?

该问题本质上是在一个赋权图中寻找最小权和的生成树, 称具有这种性质的树为最优树.

1956年Kruskal推广了生成树的"避圈法", 给出了最优树的一个算法.

Kruskal算法

(1) 选择边e_1, 使得$w(e_1)$尽可能小;

(2) 若已选定边$e_1, e_2, \cdots, e_i$, 则从$E\backslash\{e_1, e_2, \cdots, e_i\}$中选取边$e_{i+1}$ 使得

(i) $G[\{e_1, e_2, \cdots, e_{i+1}\}]$为无圈图;

(ii) $w(e_{i+1})$是满足(i)的尽可能小的权.

(3) 当(2)不能继续执行时, 停止.

利用Kruskal算法可以很容易的得到图6.1.2的最优生成树，如图6.3.4所示。

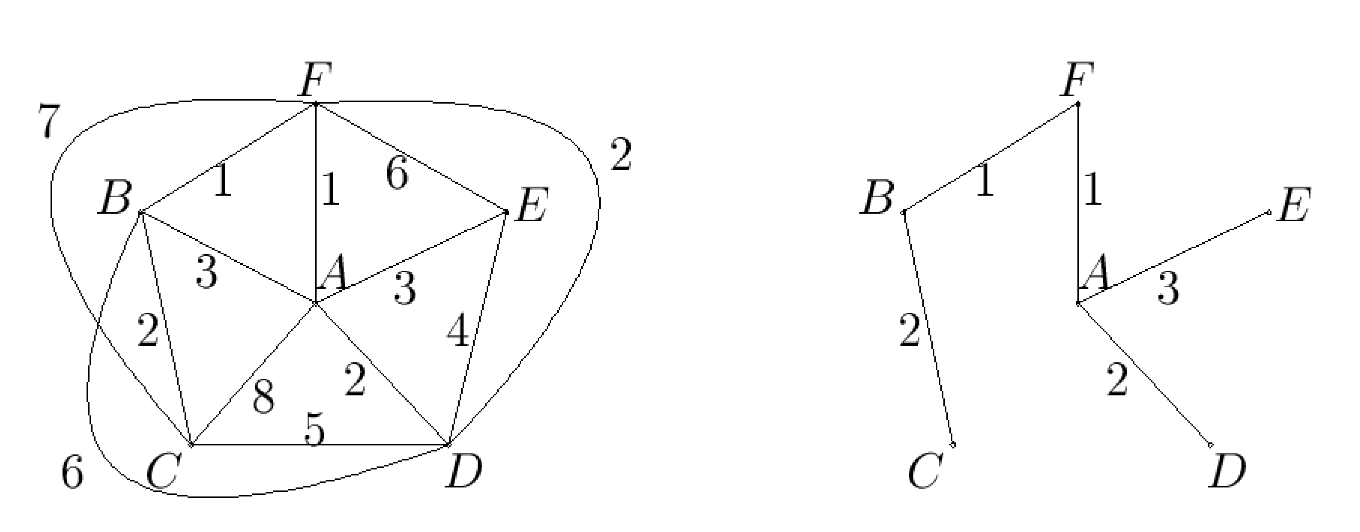


图6.3.4 最优生成树

访问主页
标题页
第 57 页 共 134 页
返回
全屏显示
关闭
退出

4 最大匹配

4.1. 定义与问题的描述

本节所要研究的问题是最优匹配, 它是最大匹配问题的一个子问题. 在本节, 仅讨论无向图的最大匹配问题.

我们有一组顶点U, 可以看作工人, 另一组顶点V, 可以看作任务, 它们有相同的大小. 连接U和V的边的权可以表示为工人u_i完成任务v_j的效益. 其问题是求这样一种工作分配, 每个工人完成一项任务, 而且效益最大. 如图6.4.1所示。

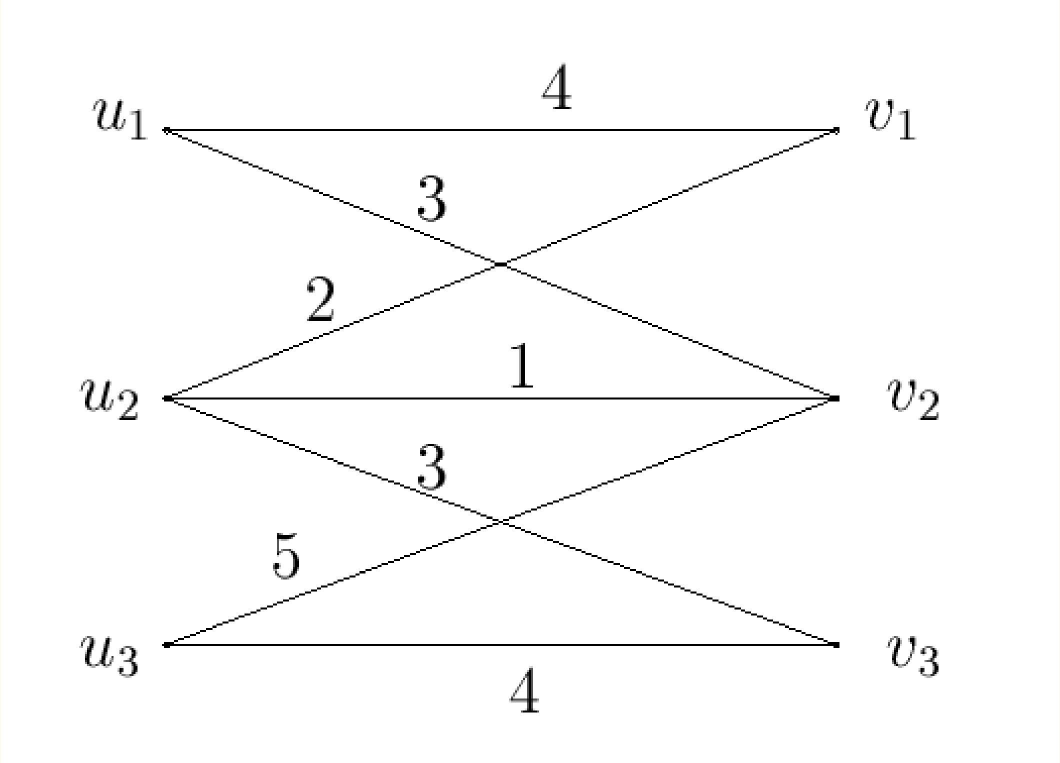


图6.4.1 二部图

定义6.4.1 如果存在V的两个子集V_1和V_2, 使得每条边有一个顶点属于V_1, 而另一个顶点属于V_2, 则称图$G(V,E)$为二部图(或偶图).

图6.4.1就是一个二部图。

设M是E的一个子图, 如果M中的所有边没有公共的顶点, 则称M是二部图$G(V,E)$的一个匹配.

如果它关联的边是匹配M的一条边, 则称顶点是饱和的.

如果V_1中所有顶点是饱和, 称M是G的完美匹配.

在图6.4.2(a)中, $M = \{(1,7),(2,6),(3,5)\}$是一个匹配, 顶点1,2,3是$V = \{1,2,3,4\}$的饱和点.

在图6.4.2(b)中, 边$\{(1,5),(2,5)\}$不是一个匹配, 因为有公共的顶点5.

在图6.4.2(c)中,$M^* = \{(1,7),(2,6),(3,5),(4,8)\}$是一个完美匹配.

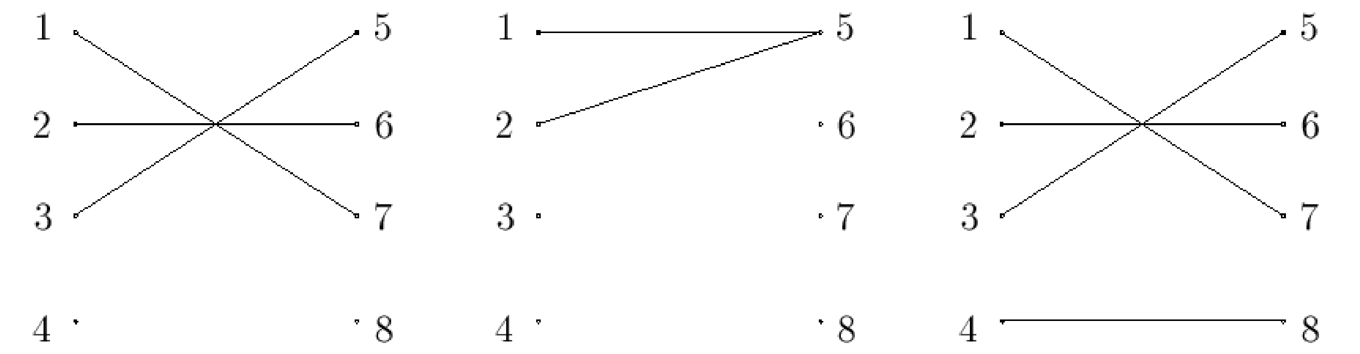


图 6.4.2 (a) 匹配图　图 6.4.2 (b) 非匹配图　图 6.4.2 (c) 完美匹配图

下面所讨论的问题是如何求一个具有最大权的完美匹配. 假定

(1) 图G是加权二部图, 且两个顶点集合为V_1和V_2.

(2) V_1和V_2均有n个顶点.

(3) 对于每个顶点$u \in V_1$和$v \in V_2$, 则边$e = (u, v)$存在.

在上述假定下, 将每个顶点组合, 可以得到$n!$个完美匹配. 我们的目标是找到一个具最大权的完美匹配.

从应用角度来看, 假定(3)是合理的, 因为如果某个工人不能完成某项任务, 可将它们之间的效益权重定义为0.

记$u_1, u_2, \cdots, u_n$是V_1的顶点, $v_1, v_2, \cdots, v_n$是V_2的顶点, 定义w_{ij}是由u_i到v_j的权重. 因此权矩阵为$W = (w_{ij})$.

定义6.4.2 称完美匹配M^*是最优的, 如果对所有的完美匹配M有

$$\sum_{(u_i,v_j)\in M^*} w_{ij} \geq \sum_{(u_i,v_j)\in M} w_{ij}.$$

在图6.4.3中, 我们可以用枚举法, 得到最大完美匹配, $M^* = \{(u_1,v_1),(u_2,v_1),(u_2,v_3)\}$.

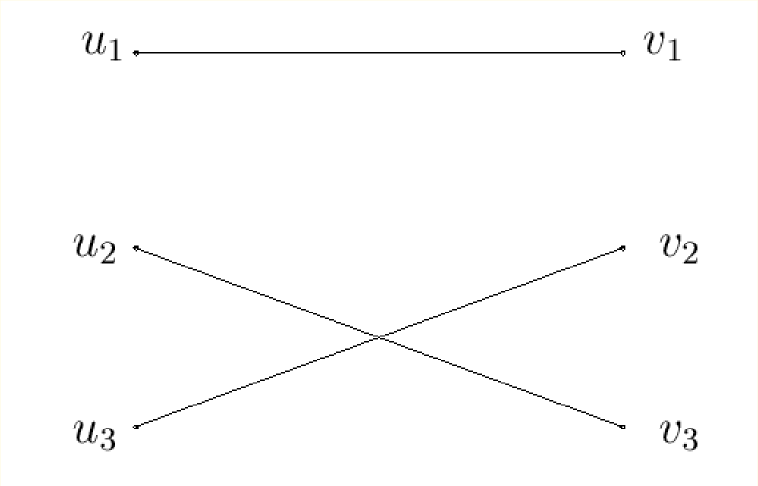


图6.4.3 完美匹配

4.2. 最优匹配算法

求最优匹配算法是由Kuhn (1955) 和Munkres (1957) 分别提出来的，因此称为Kuhn-Munkres算法。由于我们将在后面介绍如何用Lingo软件求解最优匹配的方法，这里就不介绍了。

5 最大流问题

5.1. 定义与问题的描述

定义6.5.1 设$G(V,E)$为有向图, 如果在V中有两个不同的顶点子集X和Y, 而在边集E上定义一个非负权值c, 则称G为一个网络.

称X中的顶点为源，Y中的顶点为汇, 即非源又非汇的顶点称为中间顶点, 称c为G的容量函数, 容量函数在边e上的值称为容量. 边$e=(u,v)$的容量记为$c(e)$或$c(u,v)$.

在这里我们仅讨论仅有一个源和一个汇的网络.

图6.5.1表示具有一个源x, 一个汇y, 和六个中间顶点$v_1, v_2, v_3, v_4, v_5, v_6$的网络.

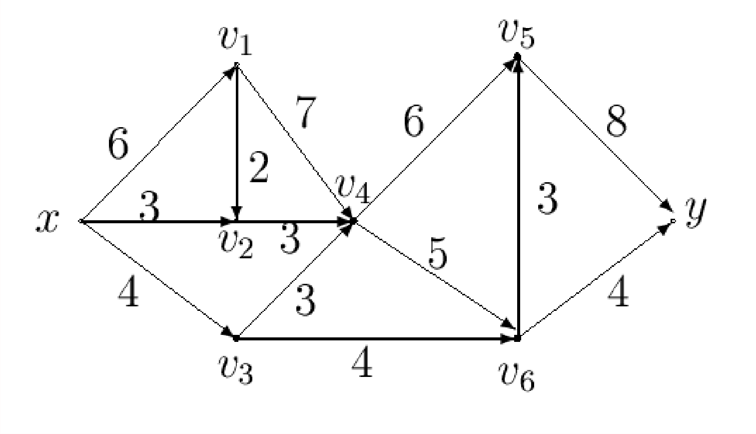


图6.5.1 一个源一个汇的网络

网络G中每一条边(u,v)有一个容量$c(u,v)$，除此之外，对边(u,v)还有一个通过边的流，记为$f(u,v)$.

显然，边(u,v)上的流量$f(u,v)$不会超过该边上的容量$c(u,v)$，即

$$0 \le f(u,v) \le c(u,v), \qquad (5.1)$$

称满足不等式(5.1)的网络G是相容的.

对于所有中间顶点u, 流入的总量应等于流出的总量, 即

$$\sum_{v \in V} f(u,v) = \sum_{v \in V} f(v,u). \tag{5.2}$$

一个网络G的流量值f定义为从源x流出的总流量, 即

$$V(f) = \sum_{v \in V} f(x,v). \tag{5.3}$$

由式(5.2)和式(5.3)可以看出, f的流量值也为流入汇y的总流量.

设V_1和V_2是顶点集V的子集, 用(V_1, V_2)表示起点在V_1中, 终点在V_2中的边的集合. 用$f(V_1, V_2)$表示(V_1, V_2)中边的流的总和, 即

$$f(V_1, V_2) = \sum_{u \in V_1, v \in V_2} f(u, v).$$

特别取$V_1 = v$, $V_2 = V$, 结合式(5.2)和式(5.3), 得到

$$f(v, V) - f(V, v) = \begin{cases} V(f), & \text{当} v = x, \\ 0, & \text{当} v \in V, v \neq x, v \neq y, \\ -V(f), & \text{当} v = y, \end{cases} \tag{5.4}$$

称满足式(5.4)的网络G为守恒的.

如果流f满足不等式(5.1)和式(5.4)，则称流f是可行的．如果存在可行流f^*, 使得对所有的可行流f均有

$$V(f^*) \geq V(f),$$

则称f^*为最大流.

例6.5.1 图6.5.2所示的网络G中, 每条边旁的第一个数是边的容量, 第二个数是边的流量, 例如, $c(x,v_1)=8$, $f(x,v_1)=4$, $c(v_1,v_2)=5$, $f(v_1,v_2)=1$等等.

不难验证, 网络G满足条件(5.1)和(5.4), 因此网络G的流f是可行流, 其流量值$V(f)=8$.

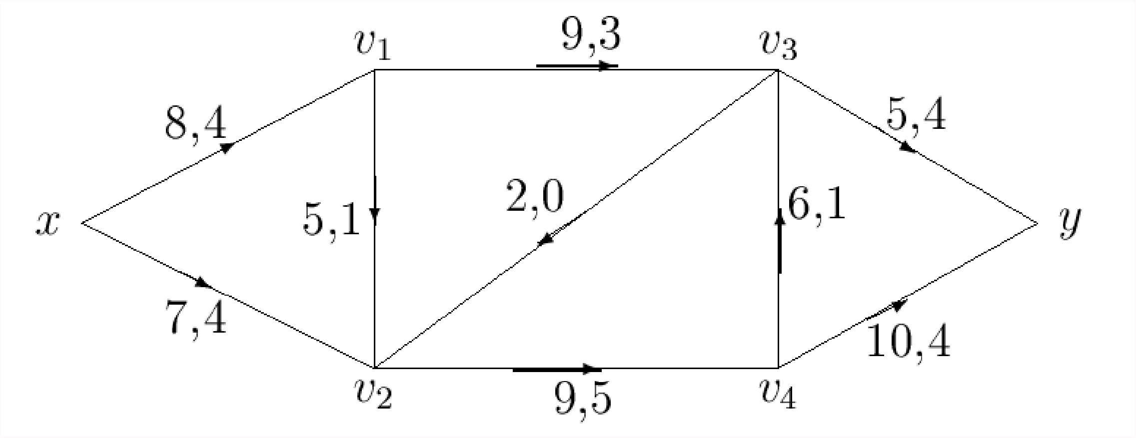


图6.5.2 具有可行流的网络

如果一条边的流量等于该边的容量, 即$f(u,v) = c(u,v)$, 称边(u,v)为饱和边; 否则称为非饱和边. 对于任意的网络G, 至少存在一个可行流, 因为对所有的边(u,v), 由于$f(u,v) = 0$满足条件(5.1)和(5.4), 此时称它为零流.

定义6.5.2 设$G(V,E)$是具有单一源x和单一汇y的网络, V_0是V的子集, V_0^c是V_0 的补集, 若$x \in V_0$, $y \in V_0^c$, 称形为(V_0, V_0^c)的边集合为网络G的割, 记为K或$K(V_0)$.

由定义可知, 网络G的一个割是分离源和汇的一个边集合.

将割K中所有边的容量之和称为割容量, 记为$c(K)$, 即

$$c(K) = \sum_{(u,v)\in(V_0,V_0^c)} c(u,v).$$

如果存在一个割K^*, 使得对于所有割K, 均有

$$c(K^*) \le c(K),$$

则称K^*为最小割.

例6.5.3 图6.5.2所示的网络G中, 取$V_0=\{x,v_1,v_2\}$, 则$V_0^c=\{v_3,v_4,y\}$, 那么割$K=\{(v_1,v_3),(v_2,v_4)\}$. 其割容量为18. 相应的流量为$f(V_0,V_0^c)=8$.

5.2. 主要结果和算法

定理6.5.1 设f是网络$G(V,E)$上的可行流, V_0是包含源x, 但不包含汇y的顶点集合, 则

$$V(f) = f(V_0, V_0^c) - f(V_0^c, V_0).$$

推论6.5.1 设f是网络$G(V,E)$上的可行流, 对于任意割$K = (V_0, V_0^c)$, 均有

$$V(f) \leq c(K).$$

定理6.5.2 设$G(V,E)$是网络, 如果存在G上的可行流f^*和割K^*, 使得

$$V(f^*) = c(K^*),$$

则f^*是最大流, K^*是最小割.

最大流问题也可以化成线性规划问题，从而可用Lingo软件求解，这里就不详细介绍最大流的求解方法。

6 用LINGO软件包求解组合优化问题

有许多组合优化问题都可以化成线性规划问题来求解, 这里我们简单介绍几种可用线性规划来求解的组合优化问题。

6.1. 运输问题

设有某种物资需要从m个产地$A_1, A_2, \cdots, A_m$运到n个销地$B_1, B_2, \cdots, B_n$, 其中每个产地的产量为$a_1, a_2, \cdots, a_m$, 每个销地的销量为$b_1, b_2, \cdots, b_n$. 设从产地A_i到销地B_j的运费单价为$c_{ij}(i = 1, 2, \cdots, m, j = 1, 2, \cdots, n)$, 问如何调运使总运费最少?

设从产地A_i到销地B_j的量为x_{ij}个单位, 则运输问题的数学表达式为:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij},$$

$$s.t. \quad \sum_{i=1}^{m} x_{ij} \geq b_j,\ j=1,2,\cdots,n \text{(运入量大于销量)},$$

$$\sum_{j=1}^{n} x_{ij} \leq a_i,\ i=1,2,\cdots,m \text{(运出量小于产量)},$$

$$x_{ij} \geq 0,\quad i=1,2,\cdots,m,\quad j=1,2,\cdots,n.$$

具体问题如下：设有3个产地($m = 3$)和4个销地($n = 4$), 其产量、销量及单位运费如下表：

表 6.6.1 3 个产地 4 个销地的运输问题

	B_1	B_2	B_3	B_4	产量
A_1	6	2	6	7	30
A_2	4	9	5	3	25
A_3	8	8	1	5	21
销量	15	17	22	12	

相应的LINGO语句为

```
model:
! A 3 Warehouse, 4 Customer
   Transportation Problem;
  sets:
    Warehouse /1..3/: a;
    Customer  /1..4/: b;
    Routes( Warehouse, Customer) : c, x;
  endsets
```

```
! The objective;
  [OBJ] min = @sum( Routes: c * x);

! The demand constraints;
  @for( Customer(j): [DEM]
    @sum( Warehouse(i): x(i,j)) >= b(j));

! The supply constraints;
  @for( Warehouse(i): [SUP]
    @sum( Customer(j): x(i,j)) <= a(i));

```

```
! Here are the parameters;
 data:
    a =   30, 25, 21 ;
    b =   15, 17, 22, 12;
    c =    6,  2,  6,  7,
           4,  9,  5,  3,
           8,  8,  1,  5;
 enddata
end
```

用LINGO软件求解，得到

$$x_{11}^*=2,\ x_{12}^*=17,\ x_{13}^*=1,$$

$$x_{21}^*=13,\ x_{24}^*=12,\ x_{33}^*=21,$$

$$f^*=161.$$

即产地A_1运往B_1, B_2, B_3的运量为$2,17,1$, 余10个单位。产地A_2运往B_1,B_4的运量为$13,12$, 余量为0。产地A_3运往B_3的运量为21, 余量为0. 总运费为161.

6.2. 最优指派（最优匹配）问题

n个人作n项工作, c_{ij}表示第i个人做第j项工作的支出, $x_{ij}=1$表示第i 个人做第j项工作, 0表示不做. 相应的线性规划问题为

$$\max \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij},$$

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

(每个人必需作一项工作)

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

(每项工作需要有一个人去作)

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

具体问题如下：设有6个人和6项工作的最优指派问题($n=6$), 其支付矩阵如下表：

表 6.6.2 6 人、6 项工作的最优指派问题

人	工作种类					
1	20	15	16	5	4	7
2	17	15	33	12	8	6
3	9	12	18	16	30	13
4	12	8	11	27	19	14
5	–	7	10	21	10	32
6	–	–	–	6	11	13

其中–表示某人无法做某项工作。

LINGO语句为

```
model:   !  Assignment model (ASSIGNMX);
  sets:
    Flight/1..6/:;
    Assign(Flight, Flight):x, c;
  endsets
  data:
    c = 20   15   16    5    4    7
        17   15   33   12    8    6
         9   12   18   16   30   13
        12    8   11   27   19   14
      -999    7   10   21   10   32
      -999 -999 -999    6   11   13;
  enddata

```


访 问 主 页
标 题 页
第 89 页 共 134 页
返 回
全 屏 显 示
关 闭
退 出

```
! Maximize valve of assignments;
  max = @sum(Assign: c*x);
  @for(Flight(i):
!    Each i must be assigned to some j;
     @sum(Flight(j): x(i,j)) = 1;
!    Each i must receive an assignment;
     @sum(Flight(j): x(j,i)) = 1;
  );
end
```

无法做的工作用一个较大的负数表示。在程序中不用加$0-1$约束，因为由指派的理论可知, 其最优解只能取0或1.

访问主页
标题页
第 90 页 共 134 页
返回
全屏显示
关闭
退出

用LINGO软件求解得到：

$$x_{11}=1,\ x_{23}=1,\ x_{32}=1,$$

$$x_{44}=1,\ x_{56}=1,\ x_{65}=1,$$

$$f^*=135.$$

即第一个人做第1项工作，第二个人做第3项工作，第三个人做第2项工作，第四个人做第4项工作，第五个人做第6项工作，第六个人做第5项工作. 总效益值为135.

6.3. 最优连线问题

对于n个城市, 我们要建立一个通讯网络. 现已知任意两城市间的距离, 要求连接这n个城市的最优通讯网络, 即求完全加权图的最小生成树.

已知六大城市Atlanta，Chicago，Cincinnati，Houston，LA，Montreal之间的距离如表6.6.3. 求它们之间的最小生成树。

表 6.6.3 六大城市之间的距离表

城市	Atl	Chi	Cin	Hou	LA	Mon
Atl	0	702	454	842	2396	1196
Chi	702	0	324	1093	2136	764
Cin	454	324	0	1137	2180	798
Hou	842	1093	1137	0	1616	1857
LA	2396	2136	2180	1616	0	2900
Mon	1196	764	798	1857	2900	0

下面写出相应的的LINGO语句

```
MODEL:
  SETS:
    CITY/1..6/:U;!U( I)=level of city I;
                 ! U( 1) = 0;
    LINK(CITY,CITY):
      DIST,  !The distance matrix;
         X;  !X(I,J)=1 if we use link I,J;
  ENDSETS
```

```
DATA:!Distance matrix need not be symmetric;
     !However, city 1 is base of the tree;
 !to:Atl  Chi  Cin  Hou   LA  Mon;
 DIST= 0  702  454  842 2396 1196 !from Atl;
     702    0  324 1093 2136  764 !from Chi;
     454  324    0 1137 2180  798 !from Cin;
     842 1093 1137    0 1616 1857 !from Hou;
    2396 2136 2180 1616    0 2900 !from LA;
    1196  764  798 1857 2900    0;!from Mon;
ENDDATA
```

```
! The model size: Warning, may be slow for N>=8;
  N = @SIZE( CITY);
 ! Minimize total distance of the links;
  MIN = @SUM( LINK: DIST * X);
 ! For city K, except the base, ... ;
  @FOR(CITY(K)|K #GT# 1:
    ! It must be entered;
      @SUM(CITY(I)|I #NE# K: X(I,K))=1;
```

```
   ! If there are 2 disjoint tours from 1 city
     to another, we can remove a link without
     breaking connections. Note: These are
     not very powerful for large problems;
     @FOR(CITY(J)|J #GT# 1 #AND# J #NE# K:
        U(J) >= U( K) + X ( K, J) -
        ( N - 2) * ( 1 - X( K, J)) +
        ( N - 3) * X( J, K); );
 );
```


访问主页
标题页
第 97 页 共 134 页
返 回
全屏显示
关 闭
退 出

```
 ! There must be an arc out of city 1;
  @SUM(CITY(J)|J #GT# 1: X(1,J)) >= 1;

 ! Make the X’s 0/1;
  @FOR( LINK: @BIN( X); );

 ! The level of a city except the base
   is at least 1 but no more than N-1,
   and is 1 if it links to the base;
   @FOR( CITY( K)| K #GT# 1:
      @BND( 1, U( K), 999999);
      U(K) <= N-1 - (N - 2)* X(1, K); );
END
```

用LINGO软件求解得到：连接六大城市的最小生成树为

Atlanta$\rightarrow$ Cincinnati， Atlanta$\rightarrow$ Houston, Chicago$\rightarrow$ Montreal， Cincinnati$\rightarrow$ Chicago, Houston$\rightarrow$ LA.

6.4. 旅行商问题

现已知n个城市之间的距离，从某个城市出发，经过所有的城市，最后再回到出发点。这就是旅行商问题。即在完全加权图中，找最优Hamilton回路。

求上述六大城市Atlanta, Chicago, Cincinnati, Houston, LA, Montreal的最优Hamilton回路。

相应的LINGO语句

```
MODEL:
 ! Traveling Salesman Problem for the
 cities of Atlanta, Chicago, Cincinnati,
 Houston, LA, Montreal;
 SETS:
  CITY/1..6/:U; !U(I)=sequence no. of city;
  LINK(CITY, CITY):
     DIST, !The distance matrix;
        X; !X(I,J)=1 if we use link I, J;
 ENDSETS
```

```
DATA:!Distance matrix, it need not be symmetric;
 DIST =    0  702  454  842 2396 1196
         702    0  324 1093 2136  764
         454  324    0 1137 2180  798
         842 1093 1137    0 1616 1857
        2396 2136 2180 1616    0 2900
        1196  764  798 1857 2900    0;
ENDDATA
```

```
N = @SIZE( CITY);
MIN = @SUM( LINK: DIST * X);
@FOR( CITY( K):
!  It must be entered;
 @SUM(CITY(I)|I #NE# K: X(I,K))=1;
!  It must be departed;
 @SUM(CITY(J)|J #NE# K: X(K,J))=1;
```


访 问 主 页
标 题 页

返 回
全 屏 显 示
关 闭
退 出

```
! Weak form of the subtour breaking constraints;
! These are not very powerful for large problems;
 @FOR( CITY( J)| J #GT# 1 #AND# J #NE# K:
     U( J) >= U( K) + X ( K, J) -
     ( N - 2) * ( 1 - X( K, J)) +
     ( N - 3) * X( J, K)
 );
);
```

```
 ! Make the X's 0/1;
 @FOR( LINK: @BIN( X));
 ! For the first and last stop we know...;
 @FOR( CITY( K)| K #GT# 1:
  U( K) <= N - 1 - ( N - 2) * X( 1, K);
  U( K) >= 1  + ( N - 2) * X( K, 1)
 );
END
```

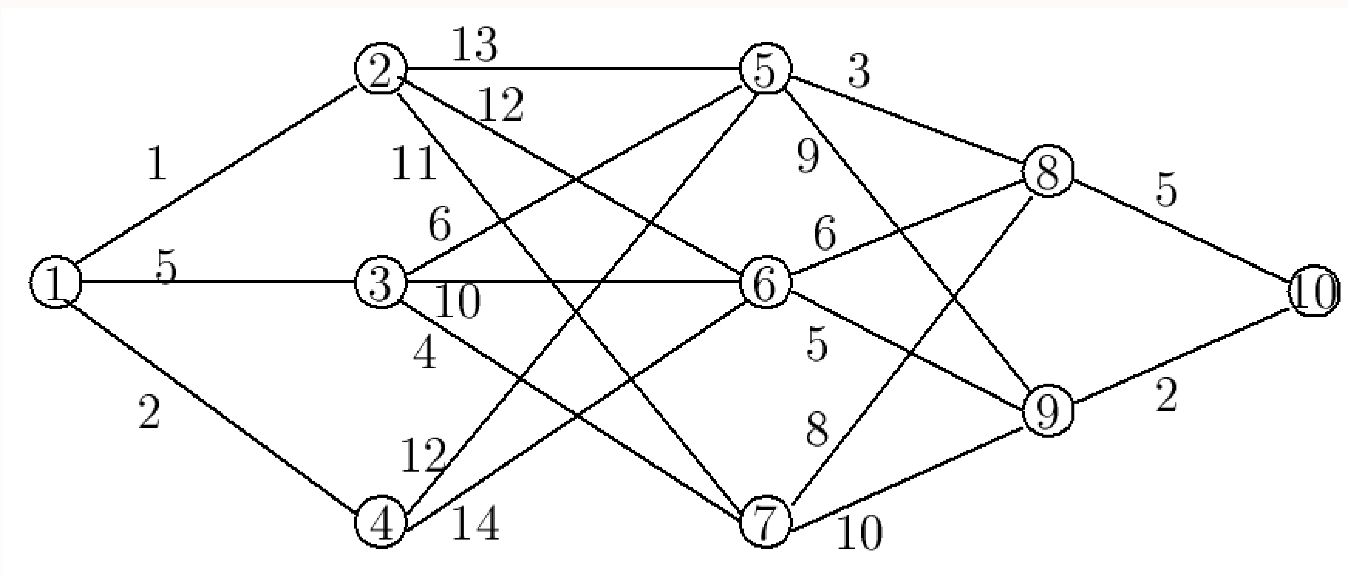
用LINGO软件求解得到：六大城市的最优最优Hamilton回路为

Atlanta→Houston→LA→Chicago

→Montreal→Cincinnati →Atlanta.

6.5. 最短路问题

给定n个城市，已知连接城市i到城市j的道路长度为w_{ij}, 现求一条从城市1到城市n的最短路。下面给出求城市1到城市10的最短路.



访问主页
标题页
◀◀ ▶▶
◀ ▶
第 107 页 共 134 页
返回
全屏显示
关闭
退出

相应的LINGO语句

```
MODEL:
SETS:
! We have a network of 10 cities.  We want
  to find the length of the shortest route
  from city 1 to city 10.;

! Here is our primitive set of ten cities,
  where F( i) represents the shortest path
  distance from city i to the last city;
 CITIES /1..10/: F;

```

```
! The derived set ROADS lists the roads
  that exist between the cities (note: not
  all city pairs are directly linked by a
  road, and roads are assumed to be one way.);
```

```
16]  ROADS( CITIES, CITIES)/
17]   1,2  1,3  1,4
18]   2,5  2,6  2,7
19]   3,5  3,6  3,7
20]   4,5  4,6
21]   5,8  5,9
22]   6,8  6,9
23]   7,8  7,9
24]   8,10
25]   9,10/: D;
26] ! D( i, j) is the distance from city i to j;
27]ENDSETS
28]
```

第16 ~ 26行是定义连接城市的边.

```
DATA:
! Here are the distances that correspond
  to the above links;
```

```
32]  D =
33]      1    5    2
34]     13   12   11
35]      6   10    4
36]     12   14
37]      3    9
38]      6    5
39]      8   10
40]      5
41]      2;
42]ENDDATA
```

第$32\sim41$行是相应的权（道路长度）。

```
! If you are already in City 10, then the cost to
  travel to City 10 is 0;
F( @SIZE( CITIES)) = 0;
```

```
! The following is the classic dynamic programming
 recursion. In words, the shortest distance
 from City i to City 10 is the minimum over
 all cities j reachable from i of the sum
 of the distance from i to j plus the minimal
 distance from j to City 10;
@FOR( CITIES(i)| i #LT# @SIZE( CITIES):
 F(i) = @MIN(ROADS(i,j): D(i, j) + F(j))
);
END
```

用LINGO软件求解得到：城市1～城市9到城市10的最短路为

19，19，14，20，8，7，12，5，2.

7 实例分析— 通讯网络的最小生成树

通讯网络的最小生成树问题是1991年美国大学生数学建模竞赛B题, 我们以对它的求解来作为本章的结束, 同时也是对本章内容的应用和检验.

7.1. 通讯网络的最小生成树

两个通讯站间通讯线路的费用与线路长度成正比. 通过引入若干“虚设站”并构造一个新的Steiner树就可以降低由一组站生成的传统的最小生成树所需的费用. 用这种方法可降低费用多达$13.4\%(1-\frac{\sqrt{3}}{2})$, 如图6.7.1所示. 而且为构造一个有$n$个站的网络的费用最低的Steiner树决不需要多于$(n-2)$个虚设站. 下面有两个简单的例子.

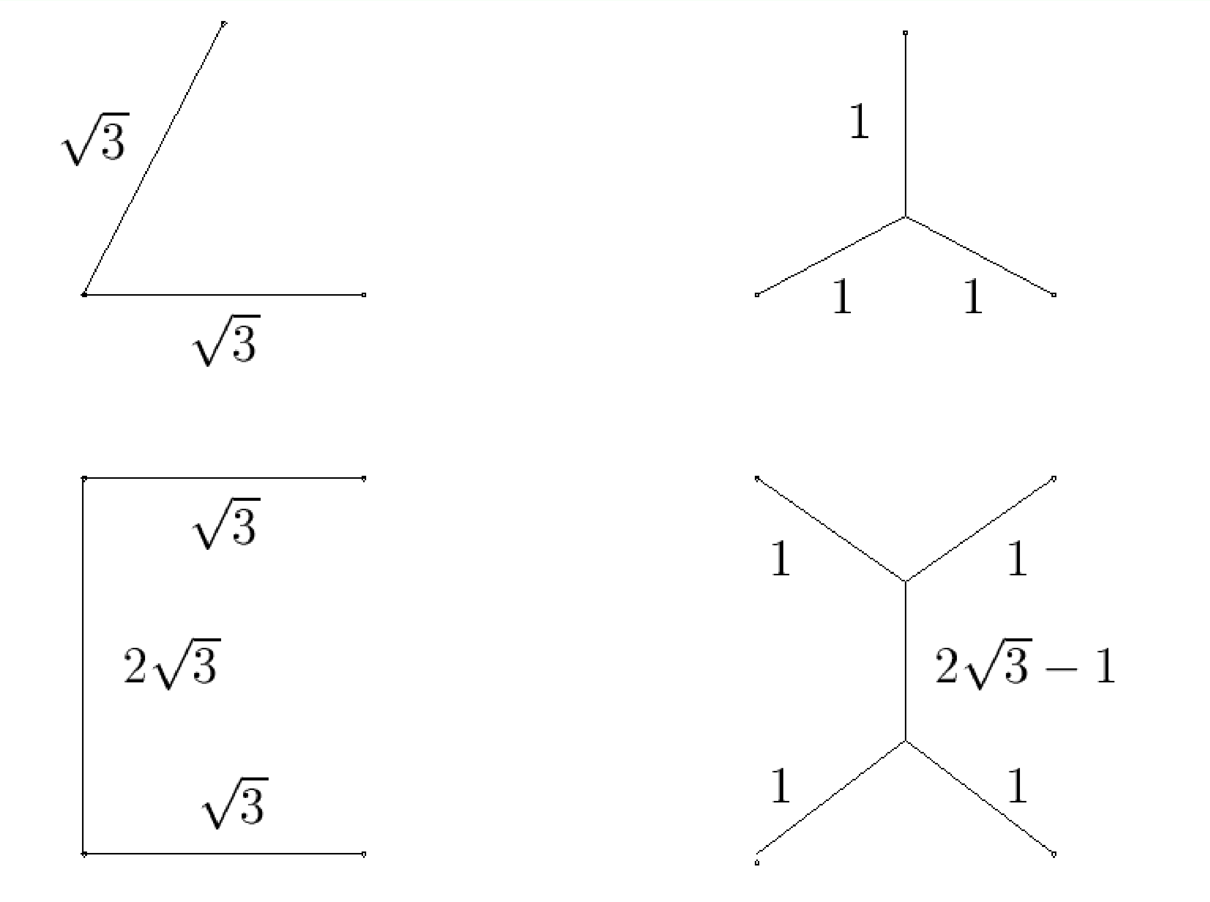


图6.7.1 增加虚设站节省费用的情况，节省量达$1-\frac{\sqrt{3}}{2}$

对于局部网络而言，常有必要用直折线距离或“棋盘”距离来代替欧氏距离. 对于这种尺度可以计算距离如图6.7.2所示.

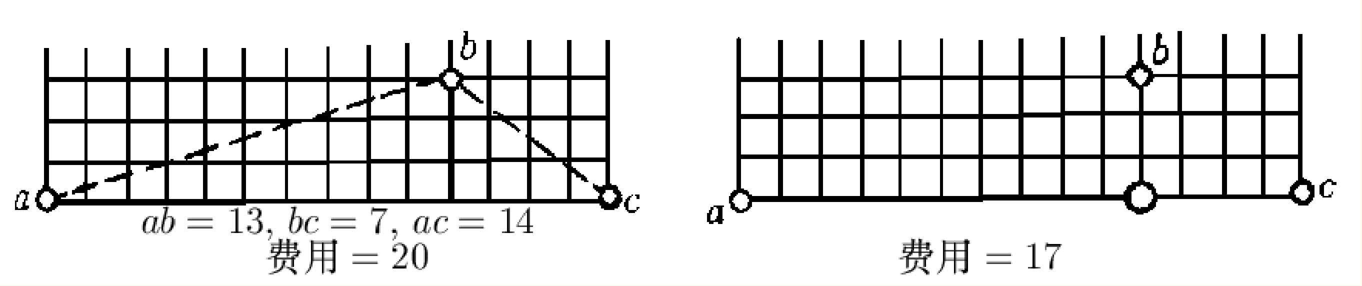
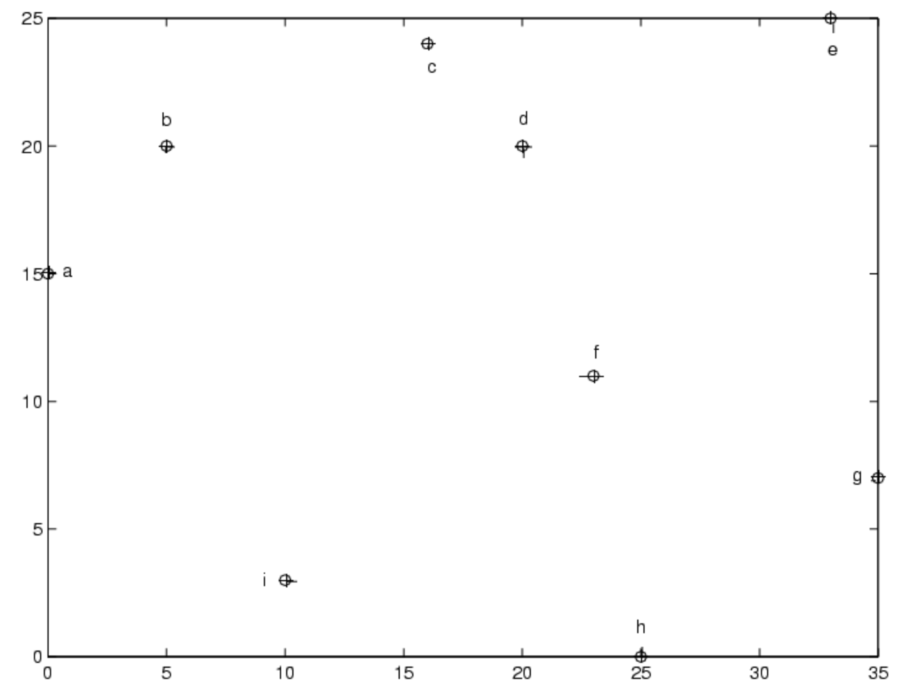


图6.7.2 在“棋盘”距离下，
增加虚设站节省费用的情况

假定你希望设计一个有9个站的局部网络的最低造价生成树. 这9个站的直角坐标是$a(0,15)$, $b(5,20)$, $c(16,24)$, $d(20,20)$, $e(33,25)$, $f(23,11)$, $g(35,7)$, $h(25,0)$, $i(10,3)$.



限定你只能用直线, 而且所有的虚设站必需位于格点上(即坐标是整数). 每条直线的造价是其长度值.

1. 求该网络的一个最小生成费用树.
2. 假定每个站的费用为$d^{\frac{3}{2}} \cdot w$, 其中$d =$通讯站的度, 若$w = 1.2$, 求最小费用树.
3. 试推广本问题.

7.2. 问题的分析与求解

先考虑简单的问题, 即只考虑在原来点上的最小费用树问题.

已知九个点的坐标是$a(0,15)$, $b(5,20)$, $c(16,24)$, $d(20,20)$, $e(33,25)$, $f(23,11)$, $g(35,7)$, $h(25,0)$, $i(10,3)$. 构造各边的权值,

$$w(i,j)=|x_i-x_j|+|y_i-y_j|,\ i\neq j,\ i,j=1,2,\cdots,9,$$

其中(x_i,y_i)是第i个点的坐标.

用Lingo软件，我们得到最小生成费用树，如图6.7.3所示. 若不计算站的费用, 其最小费用为110. 若计算站的费用, 其最小费用为136.43.

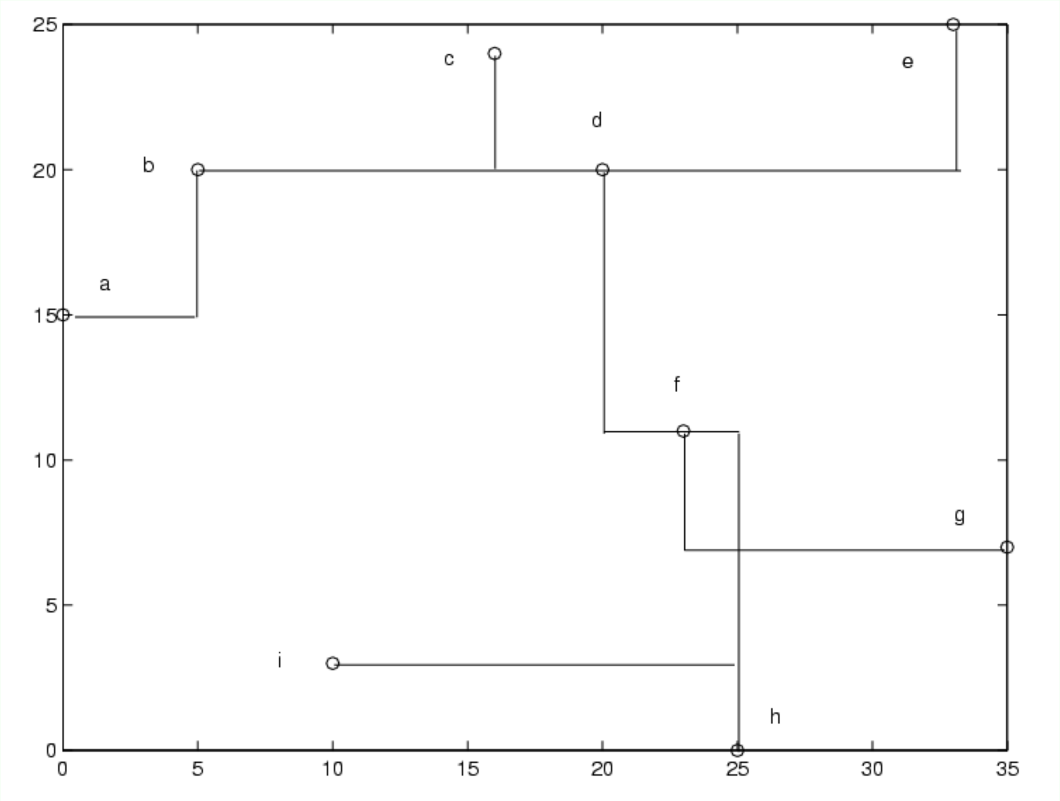


图6.7.3 九个点(无虚设站)的最小费用树

现考虑增加虚设站的情况, 全面考虑这个问题较为麻烦, 按照题目给出的结果, 虚设站的个数最多可以增加到$7 = 9 - 2$, 而按照题意, 所有的虚设站必需位于格点上, 因此可以增加虚设站的位置共有$936 = 26 \times 36$个. 若用穷举法, 共有

$$C_{936}^1+C_{936}^2+C_{936}^3+C_{936}^4+C_{936}^5+C_{936}^6+C_{936}^7 = 1.2303\times10^{17}$$

种情况, 在三天时间内找出它们的最优解是绝对不可能的. 因此, 我们需要对问题作进一步的分析.

为了减少长度, 增加的点只能在各个点位置的交线上, 因此可设虚设站的位置共有$72 = 8 \times 9$个. 因此, 共有

$$C_{72}^1 + C_{72}^2 + C_{72}^3 + C_{72}^4 + C_{72}^5 + C_{72}^6 + C_{72}^7 = 1.6444 \times 10^9$$

种情况. 即使这样, 在三天内计算出全部结果的最优解也是不可能的. 需要再作简化.

除了增加的点只能在各个点位置的交线上外, 增加的点还只能在这九个点所围区域的内部, 且原有的九个点上不用考虑是否要增加点, 因此, 可设虚设站的位置共有$31 = 6 \times 7 - 5$(已有的点)-6(边界上的点)个. 因此, 共有

$$C_{31}^1 + C_{31}^2 + C_{31}^3 + C_{31}^4 + C_{31}^5 + C_{31}^6 + C_{31}^7 = 3572223$$

种情况. 如果按每秒种得到一个最优生成树来计算, 也需要41.3452天算完, 这当然也是不允许的. 因此, 我们不考虑整体最优, 而考虑得到一个较好的策略即可.

一种考虑问题的方法是从图6.7.3出发，由图可以容易看出，在图中增加虚设站的位置为$(16,20),(25,7),(25,3)$，可以减少生成树的长度，经计算得到其最小费用树，如图6.7.4所示。

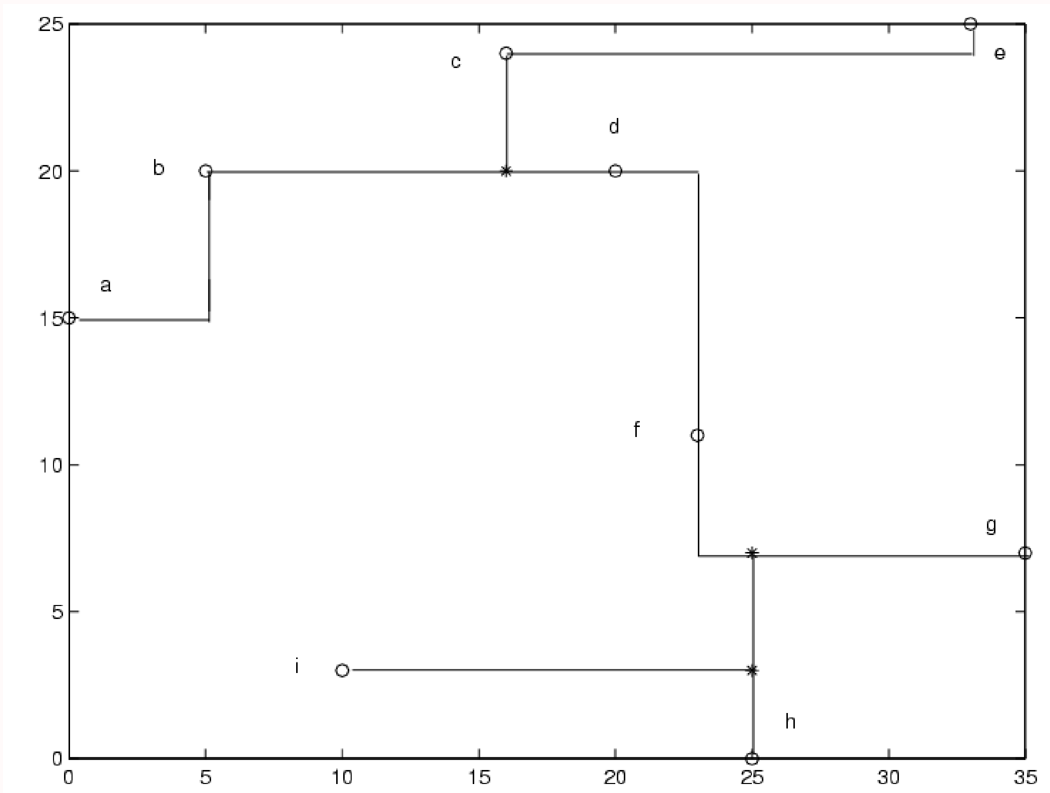


图6.7.4 十二个点(三个虚设站)的最小费用树

在不计算站的费用的情况, 其最小费用为97. 在计算站的费用情况下, 其最小费用为135.28.

增加四个虚设站$(16,20)$, $(25,7)$, $(25,3)$和$(23,20)$的最小费用树，如图6.7.5所示。

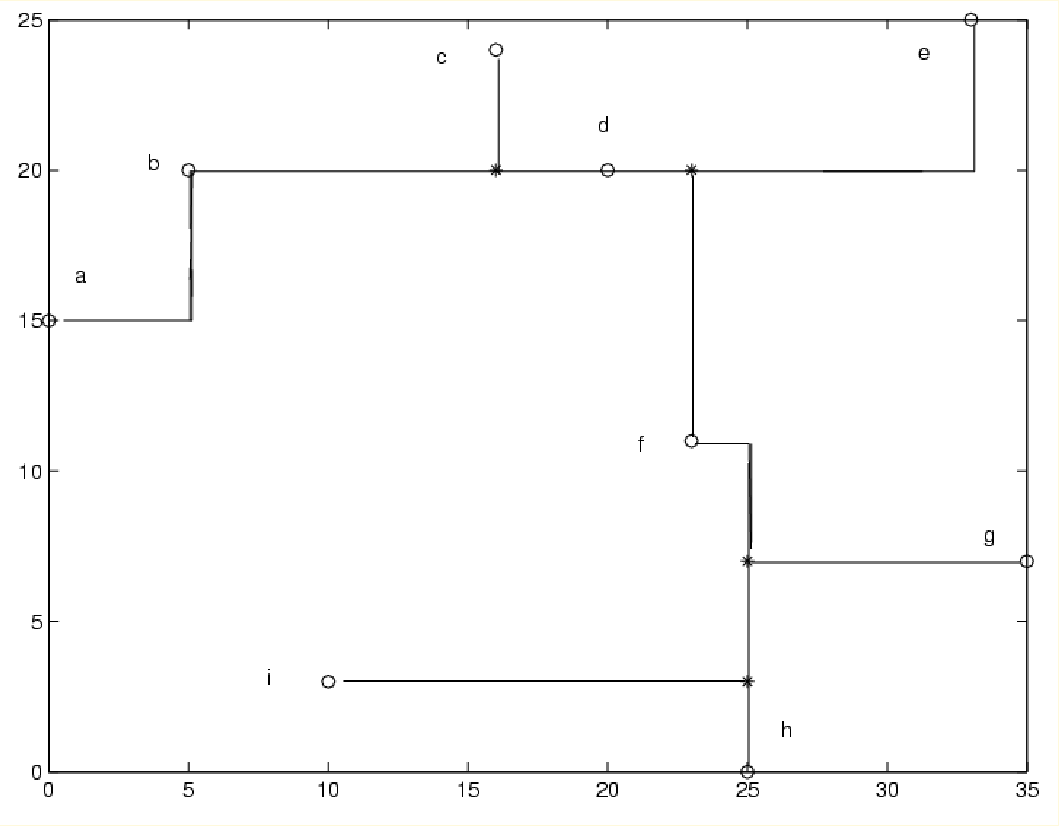


图6.7.5 十三个点(四个虚设站)的最小费用树

在不计算站的费用的情况, 其最小费用为94. 在计算站的费用情况下, 其最小费用为136.32.

因而在计算站的费用情况下, 增加过多虚设站, 并不一定降低费用, 反而使费用上升.

我们考虑增加适当的虚设站, 例如考虑增加2个虚设站$(16, 20), (25, 7)$, 得到最小生成树(见图6.7.6).

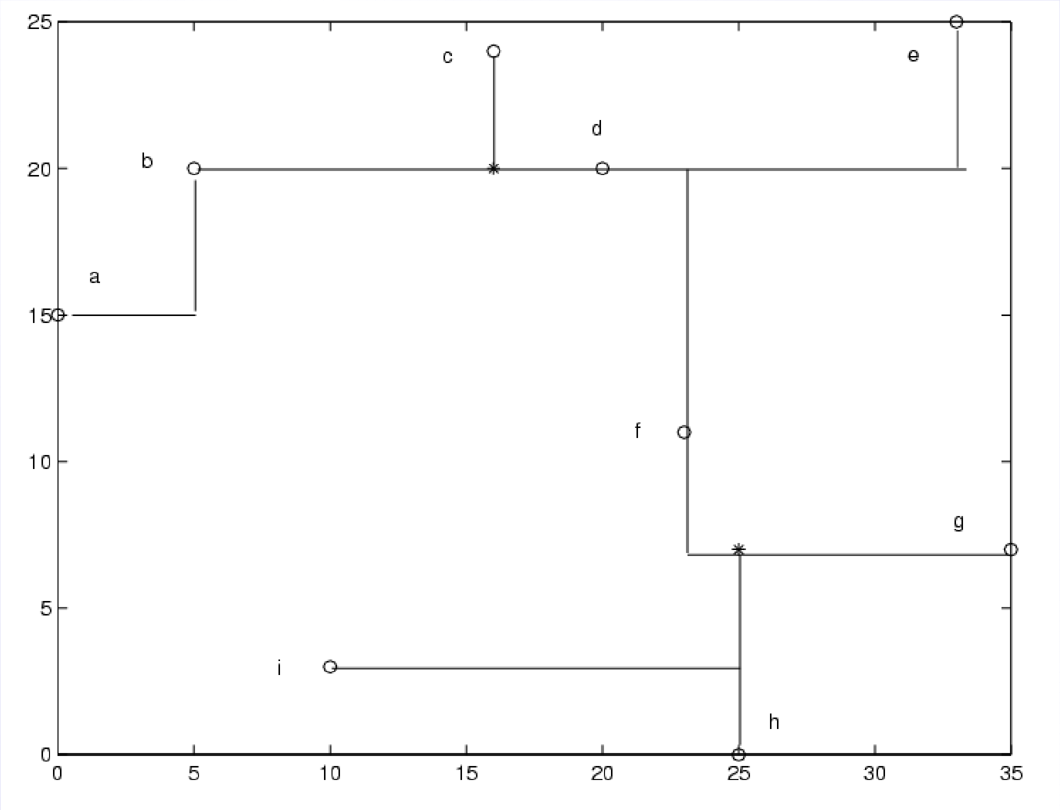


图6.6.6 十一个点(二个虚设站)的最小费用树

在不计算站的费用的情况, 其最小费用为100. 在计算站的费用情况下, 其最小费用为134.89. 但这种作法的缺点是: 需要对具体情况具体分析, 不便于推广.

另一种较为可行的方法是: 不考虑整体最优, 而考虑局部最优, 在可增加虚设站的31个位置上, 每次增加一个虚设站, 然后再计算出相应的最小费用树(不计算站的费用和计算站的费用两种情况), 比较其相应的费用值, 最终得到在该意义下的最小费用生成树. 这种方法的优点是计算量小, 最多仅有

$$31 + 30 + 29 + 28 + 27 + 26 + 25 = 196$$

情况.

这种方法的另一个优点, 便于推广, 在点数区域比较大的情况下, 仍然可以很快地计算出结果, 这种方法的缺点就是: 得到的解并不一定是整体最优解.