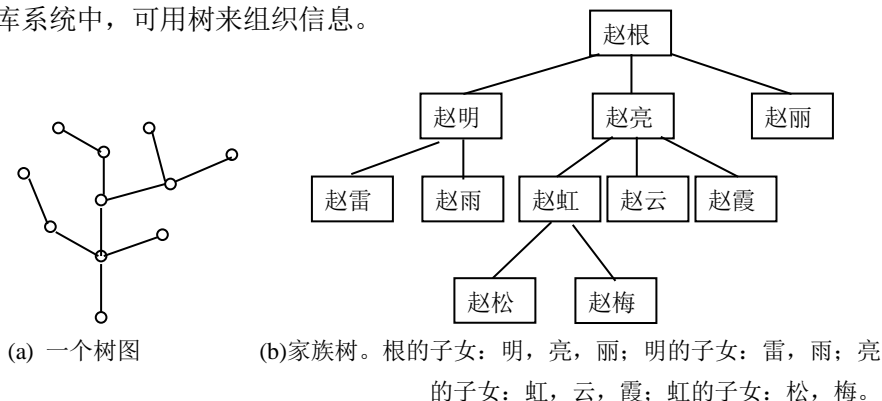


第十三章 树算法

§ 13.1 导言

树 (tree) 在图论中是相当重要的一类图, 它非常类似于自然界中的树, 图 13.1 就是一个树图的例子。树的性质非常好, 应用相当广泛。例如, 可用树图来形象地表示家族 (见图 13.1(b)), 行政组织机构; 可用树图来列举排列; 用树来分析游戏中的策略; 计算机用树来描述运算顺序, 用树来组织其拥有的资源以便于查找; 在编译程序中, 用树来表示源程序语法结构; 在数据库系统中, 可用树来组织信息。



(a) 一个树图

(b) 家族树。根的子女: 明, 亮, 丽; 明的子女: 雷, 雨; 亮的子女: 虹, 云, 霞; 虹的子女: 松, 梅。

图 13.1 树的例子

本章不讨论上述这些应用, 将主要讨论在设计具有某种最优性的网络方面起重要作用的最小生成树及其算法。图论中的算法设计与分析是一个非常引人入胜的领域, 是一个数学激发洞察力的领域。在这一章将会给出许多特色各异的算法, 你将会看到 Prim 算法和 Krusal 算法的历史; Prim 算法和 Krusal 算法的各种不同应用; 穷举法, 贪婪法, 试探法, 这些算法设计基本方法的灵活应用; 如何修改算法以适应于改变了的问题。

§ 13.2 引例: 计算机网络的线路设计

城市电信局有许多业务如收费, 营业, 112, 114 等, 希望在全市范围实

现计算机联网服务, 共享各种资源, 比如, 数据库资源。一个主要关心的问题是: 用数据通讯线把一组站点联结起来, 而不允许通讯线在非站点处连接, 如何连接可使通讯线的花费最少? 任意两个站点可经过若干中介站点, 取得联系。

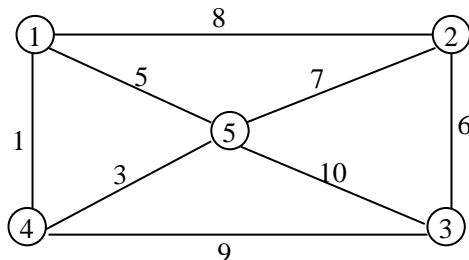


图 13.2

为简便, 以一个小型的问题为例。假设各站点间可以铺设通讯线路进行连接的情况如图 13.2 所示, 顶点为站点, 边为连接两站点之间的通讯线, 边的权为其费用。

我们的目的是要找到图 13.2 的一个连接所有顶点的具有最小总权数的连通子图。先介绍几个术语。

连通图(connected graph): 其中任意两点之间都有路径的图。如, 图 13.2 就是一个连通图。

圈(cycle): 当一条路径的起点和终点是同一顶点时, 称这条路径为圈。如图 13.2 中的路线 1—2—5—4—1 就是一个圈。

事实上, 在最经济的网络中, 不应该有任何圈, 否则, 去掉圈上的一条边, 这圈上任意两个顶点仍能取得联系, 正如在橡筋圈上剪一刀后, 仍然是一个整段。

树(tree): 没有圈的连通图称为树。如, 图 13.1 中的两个图都是树。



提示



思考

1) 树具有许多非常好的性质, 这些性质包括

- A) 树中任意两点间有唯一路径。
- B) 树的边数恰好等于顶点数减 1。
- C) 在树中任意去掉一条边, 将会不连通。
- D) 树中任意两个不相邻顶点间添一边后, 就恰好含一个圈。

2) 要将 n 个点连接起来至少需要 $n-1$ 条边。

图 G 的子图(subgraph): 由 G 的一些边和一些顶点组成, 它是 G 的一个

部分图，且必须满足：当一条边在子图时，这条边的两个端点也要在子图中。

生成树或支撑树(spanning tree): G 的树的子图，其顶点集等于 G 的顶点集；如：图 13.3 中的两个图均是图 13.2 的生成树。

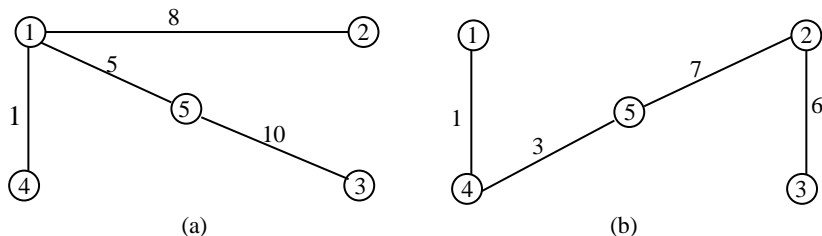


图 13.3

显然，每个生成树对应的线路费用都低于原图对应的线路费用，且生成树中没有多余边，随意去掉其中的一条边都会破坏其连通性。

因此，确定应在哪些站点之间铺设通讯线路，就可看作是在相应的加权图中构造最小费用的生成树的问题。

一般地，定义生成树的权为其上所有边权之和。

最小生成树(minimum-weight spanning tree): 在一个加权连通图 G 中，权最小的那棵生成树称为 G 的最小生成树。

最大生成树(maximum-weight spanning tree): 在一个加权连通图 G 中，权最大的那棵生成树称为 G 的最大生成树。



一个简单连通图只要不是树，其生成树就不唯一，甚至非常多。如：10 个顶点的完全图，其不同的生成树就有一亿棵。一般地， n 个顶点的完全图，其不同的生成树个数为 n^{n-2} 。

要求出最小生成树，一般不能用穷举法。30 个顶点的完全图就有 30^{28} 个生成树， 30^{28} 有 42 位，即使应用最现代的计算机，在我们有生之年也是无法穷举的，穷举法是无效算法，坏算法。必须寻求其他的有效算法，这将在下一节介绍。

如果允许通讯线可以在规定的站点以外的其他地方连接，得到的问题为 Steiner 树问题，它比最小生成树问题更复杂，这将在后面介绍。

与计算机网络设计相类似的问题还有许多，如：有线电视网，电线网，石油传输管网以及其他的管道网，铁路，公路的建设等等。

§ 13.3 最小生成树算法

在这一节，我们将介绍求最小生成树的两个算法：Prim 算法和 Kruskal 算法，它们都蕴涵了贪婪法的思想。



提示

贪婪法是一种可被用于各种各样问题的处理，它把解看成是由若干个部件构成，每一步求出解的一个部件，但这不是从整体或长远的角度去考虑的，只是局部或当前的最好选择。求出的一个个部件组合而作为最终的解。



注意

贪婪法只是一种试探法，计算上简便，有效，可提供正确解的一个近似。但一般情况下，不能保证输出的解是正确的。其正确性需要证明，这往往比较困难。

13.3.1 Kruskal 算法

假设给定了一个加权连通图 G ， G 的边集合为 E ，顶点个数为 n 。Kruskal 于 1956 年证明了，按下述贪婪法总可得到 G 的一棵最小生成树 T 。

Kruskal 算法的粗略描述：

为直观，假设 T 中的边和顶点均涂成红色，其余边为白色。一开始 G 中的边均为白色。

- 1) 将所有顶点涂成红色；
- 2) 在白色边中，挑选一条权最小的边，使其与红色边不形成圈，将该白色边涂红；
- 3) 重复 2) 直到有 $n-1$ 条红色边，这 $n-1$ 条红色边便构成最小生成树 T 的边集合。



做

用上述方法在图上手工操作，求出图 13.2 的最小生成树。

手工操作时，第 2 步中，判断是否形成圈一眼就可看出，但计算机实现就不是那么直接。

注意到在算法执行过程中，红色顶点和红色边会形成一个或者一个以上的连通分支，它们都是 G 的子树。一条边与红色边形成圈当且仅当这条边的两个端点属于同一个子树。因此判定一条边是否与红色边形成圈，只需判断

这条边的两端点是否属于同一个子树。

上述判断可这样实现：给每个子树一个不同的编号，对每一个顶点引入一个标记 t ，表示这个顶点所在的子树编号。当加入一条红色边，就会使该边两端点所在的两个子树连接起来，成为一个子树，从而两个子树中的顶点标记要改变成一样。综上，可将 Kruskal 算法细化使其更容易计算机实现。

变量说明：

c : 生成树的费用；

T : 生成树的边集合；

j : 迭代次数；

k : 记录已经被选入生成树的边数。

Kruskal 算法：

输入加权连通图 G 的边权矩阵 $[b(i,j)]_{m \times 3}$ ，顶点数 n 。

1) 整理边权矩阵

将 $[b(i,j)]_{m \times 3}$ 按第三行由小到大的次序重新排列，得到新的边权矩阵 $[B(i,j)]_{m \times 3}$ ；

2) 初始化

$j \leftarrow 0, T \leftarrow \emptyset, c \leftarrow 0, k \leftarrow 0$ ；对所有 $i, t(i) \leftarrow i$ 。

3) 更新 $T, c, t(i)$

$j \leftarrow j+1$ ，若 $t(B(1,j))=t(B(2,j))$ ，则转 4)；否则，若 $t(B(1,j)) \neq t(B(2,j))$ ，则 $T \leftarrow T \cup (B(1,j), B(2,j))$ ， $c \leftarrow c+B(3,j)$ ， $k \leftarrow k+1$ ，对所有 i ，

$t(i) \leftarrow \min\{t(B(1,j)), t(B(2,j))\}$ ，

4) 若 $k=n-1$ 或 $j=n$ ，则终止，输出 T, c ；否则返回 3)。

1) Kruskal 算法最终输出的必定是最小生成树，是最优解。



2) Kruskal 算法的时间复杂度为 $O(m \log_2 m)$ 。

例 13.1 一个编程例子

借助 MATLAB 软件，用 Kruskal 算法求图 13.2 所示的加权图的最小生成树。

加权图的存储结构采用边权矩阵 $[b(i, j)]_{m \times 3}$ ，编制的 MATLAB 程序 Kruskal.m 如下

```
% Kruskal's algorithm
```

```
b=[1 1 1 2 2 3 3 4;2 4 5 3 5 4 5 5;8 1 5 6 7 9 10 3];
```

```
[B,i]=sortrows(b',3);B=B'; %按边权由小到大重新排列矩阵b的列
```

```

m=size(b,2); n=5;
t=1:n; k=0; T=[ ]; c=0;
for i=1:m
    if t(B(1,i))~=t(B(2,i)) %判断第i条边是否与树中的边形成圈
        k=k+1; T(k,1:2)=B(1:2,i),c=c+B(3,i)
        tmin=min(t(B(1,i)),t(B(2,i)));
        tmax=max(t(B(1,i)),t(B(2,i)));
        for j=1:n
            if t(j)==tmax
                t(j)=tmin;
            end
        end
    end
end
if k==n-1
    break;
end
end
T,c

```

程序运行结果:

```

T =
     1     4
     4     5
     2     3
     2     5

c =
    17

```

因此,图 13.2 的最小生成树的边集合为 $\{(1,4),(4,5),(2,3),(2,5)\}$,费用为 17。

13. 3. 2 Prim 算法

1. 历史

1956 年,美国 AT&T 利用最小生成树来计算对几家商业客户的索价。这几家客户想把一些站点与一个称为 **Private Wire Service** 的专用网络联结起来,要求最小生成树。为了找出这样的树,工作人员作过各种可能的模型。他们把一张大比例的美国地图铺在地板上,开始寻找联结所有站点的网络。

为了找出连线总长度最小的网络，要求对任意一对站点始终不出现两种联结方式。以这种方式，比较聪明的人确实能得到很好的结果。但这种能用手工（并且跪着）操作的方式完成的问题是很有局限的。

那时，Kruskal 算法刚刚发表，它的第一步是要整理所有站点对之间的距离表。当有 n 个站点时，共有 $\binom{n}{2} = \frac{n(n-1)}{2}$ 个站点对。对于一些小问题，

Kruskal 算法很不错，但 AT & T 需要为一个客户找到一个由 500 个站点组成的最小生成树。没有一个人会用手工整理这样一个网络的 $500 \times 499 / 2 = 124750$ 条边，而那时的计算机也不具有处理这样大规模数据集的能力。在这种情况下，人们需要另一种算法。

1957 年，领导着贝尔实验室数学研究室的 Prim，得到了他的算法。Prim 算法优于 Kruskal 算法之处是 Prim 算法一次处理的数据不超过 n ，因此 Prim 算法所需的存储器要比 Kruskal 算法小。

2. 算法

Prim 算法也是一种贪婪法，其基本思路是：任选一个顶点 v_1 ，将其涂红，其余顶点为白点；在一个端点为红色，另一个端点为白色的边中，找一条权最小的边涂红，把该边的白端点也涂成红色；如此，每次将一条边和一个顶点涂成红色，直到所有顶点都成红色为止。最终的红色边和顶点便是最小生成树，上面的描述就是最小生成树的逐步生长过程。



用上述方法在图上手工操作，求出图 13.2 的最小生成树。

显然，Prim 算法的关键是如何找出连接红点与白点的具有最小权的边。若把 G 看成完全图，当前有 k 个红点，则有 $k(n-k)$ 条连接红，白点的白边。从如此多的白边中选取最短边显然费时。可构造一个较小的候选边集，只要保证最短边在里面即可。事实上，对每个白点，从该点到各红点的边中，必存在一条最短的白边，只要将所有 $n-k$ 个白点所关联的最短白边作为候选集，就必定能保证所有 $k(n-k)$ 条连接红，白点的白边中最短的白边属于该候选集。

Prim 算法：

输入加权图的带权邻接矩阵 $[a(i,j)]_{n \times n}$ 。

- 1) 建立初始候选边表, $T \leftarrow \phi$;
- 2) 从候选边中选取最短边 (u,v) , $T \leftarrow T \cup (u,v)$;
- 3) 调整候选边集;

4) 重复2), 3) 直到T含有 $n-1$ 条边。

例13. 2 一个编程例子

借助 MATLAB 软件, 用 Prim 算法求图 13.2 所示的加权图的最小生成树。

加权图的存储结构采用带权邻接矩阵 $[a(i,j)]_{n \times n}$ 。

MATLAB程序Prim.m:

```
% Prim's algorithm
a=[0   8   inf   1   5;
   8   0   6   inf   7;
  inf   6   0   9  10;
   1  inf   9   0   3;
   5   7  10   3   0];
T=[]; c=0; v=1; n=5; sb=2:n; % 1是第一个红点, sb是白点集
for j=2:n % 构造初始候选边集
    b(1,j-1)=1;
    b(2,j-1)=j;
    b(3,j-1)=a(1,j);
end
while size(T,2)<n-1
    [min,i]=min(b(3,:)); % 在候选边集中找最短边
    T(:,size(T,2)+1)=b(:,i);
    c=c+b(3,i);
    v=b(2,i); % v是新红点
    temp=find(sb==b(2,i));
    sb(temp)=[ ]; b(:,i)=[ ];
    for j=1:length(sb) % 调整候选边集
        d=a(v,b(2,j));
        if d<b(3,j)
            b(1,j)=v; b(3,j)=d;
        end
    end
end
end
T,c
```

程序运行结果:

```
T =
    1     4     5     2
```


4 5 2 3
 1 3 7 6
 c =
 17

因此, 图 13.2 的最小生成树的边集合为 $\{(1,4),(4,5),(5,2),(2,3)\}$, 费用为 17。



1) Prim 算法最终输出的必定是最小生成树, 是精确解。

2) Prim 算法的时间复杂度为 $O(n^2)$ 。

§ 13. 4 范例 1: 分组技术

13. 4. 1 问题

分组技术是设计制造系统的一种方法, 它把生产零件的机器分组, 相应地把需生产的零件分类, 使零件跨组加工的情形尽量少, 最理想的情况是使每个零件的加工, 都在组内完成。

假设有 13 种零件, 需在 9 台机器上加工。在各台机器上加工的零件号在表 13.1 中给出。

表 13.1

机器	1	2	3	4	5	6	7	8	9
加 工 的 零 件	2,3,7,8, 9, 12,13	2,7,8,1 1,12,	1,6	3,5,1 0	3,7,8,9, 12, 13	5	4,10	4,10	6

将这 9 台机器分为 3 组, 使零件跨组加工的情形尽量少, 并给出相应的零件分类。

13. 4. 2 建模

设用 M_i 表示需由机器 i 加工的零件集, 对任意两台机器 i, j , 定义 i 与 j 的相异度:

$$\omega(i, j) = \frac{|M_i \oplus M_j|}{|M_i \cup M_j|}.$$

其中 “ \oplus ” 表示对称差, 分子即为在机器 i 但不在机器 j 上加工, 或在机

器 j 但不在机器 i 上加工的零件数。分母为或在机器 i ，或在机器 j 上加工的零件数。显然 $0 \leq \omega \leq 1$ ， $\omega(i, j)=0$ 表示机器 i 与机器 j 加工的零件完全相同； $\omega(i, j)=1$ 表示机器 i 与机器 j 加工的零件没有一个相同。 ω 表达了两台不同机器加工零件的相异程度。

以机器为顶点，作一个完全图，每条边 (i, j) 被赋予权 $\omega(i, j)$ 。这个图的最小生成树是由那些相异度最小的边构成的连通图，或看成是去掉了相异度相对比较的边后余下的连通图。如果希望把机器分成 k 个组，就继续删去最小生成树上权最大的 $k-1$ 条边。于是得到 k 个分离的子树，每棵树的顶点集就构成各机器组。

13. 4. 3 模型求解及结果

对表 13.1 给出的数据，按上节的构图方式得到的加权图的边权矩阵如下

```
[1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 5 5 5 5 6 6 6 7 7 8;...
 2 3 4 5 6 7 8 9 3 4 5 6 7 8 9 4 5 6 7 8 9 5 6 7 8 9 6 7 8 9 7 8 9 8 9 9;...
0.5 1 0.89 0.14 1 1 1 1 1 1 0.62 1 1 1 1 1 1 1 1 1 1 0.5 0.87 0.67 0.75 ...
0.75 1 1 1 1 1 1 1 1 1 0 1 1]
```

用 Kruskal 算法可求出最小生成树，将前面例 13.1 给出的 MATLAB 程序中，边权矩阵 b 的值改为此处的边权矩阵，顶点数 n 改为 9。运行便得下述结果

T =

```
7      8
1      5
1      2
3      9
4      6
4      7
4      5
1      3
```

c =

4.4300

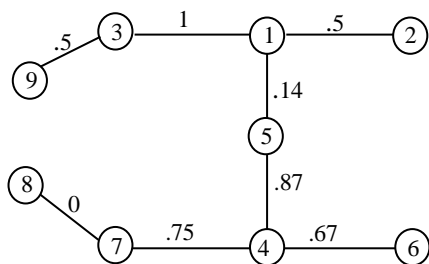


图 13.4

由此，作出生成树如图 13.4，将其中最大权的两条边去掉，得到三个分离树，它们的顶点集分别为 $\{3, 9\}$ ， $\{1, 2, 5\}$ ， $\{4, 6, 7, 8\}$ ，这也就是机器的分组。



你能给出对应于该机器分组的零件分类吗？

§ 13. 5 范例 2: 通讯网络的最佳 Steiner 树

13. 5. 1 问题

给定平面上若干通讯站，两通讯站之间的线路长度为两点间的直角折线距离，即

$$d=|x_1-x_2|+|y_1-y_2|$$

两点间的线路费用正比于线路的长度。如何布线使连接通信站的线网费用最低。通过加入若干“虚设站”后，构造出由原站点和虚设站生成的最小生成树（称为 Steiner 树），若虚设站设置得恰当，就可降低由原站点生成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%。如何构造最小 Steiner 树，即最低费用的 Steiner 树。



提示

- 1) Steiner 树允许线路在通讯站点以外连接，这种连接点即为虚设站。
- 2) 为构造一个有 n 个站的网络，最低费用的 Steiner 树最多只需 $n-2$ 个虚设站，这些虚设站称为 Steiner 点。Steiner 点位于给定通信站点的 x 坐标线， y 坐标线形成的格点上。

假定要设计一个有 9 个通讯站点的局部网络，使其造价最低。这 9 个站的直角坐标为

$a(0,15)$, $b(5,20)$, $c(16,24)$, $d(20,20)$, $e(33,25)$, $f(23,11)$, $g(35,7)$, $h(25,0)$, $i(10,3)$ 。

求出联结这 9 个通讯站的最小 Steiner 树。

13. 5. 2 假设

- 1) 通信站点集合 V_0 是整数坐标的平面点集；
- 2) 两点间的距离为直角折线距离，线路费用正比于线路长度；
- 3) 允许通讯线在非站点处连接。

13.5.3 问题分析及模型

给定 n 个通讯站点, 用通讯线把这些站点联结起来, 允许通讯线在非站点处连接, 如何连接, 可使连接通信站的线网费用最低? 这个问题不同于前面的最小生成树问题, 在那里, 不允许通讯线在非站点处连接, 这一限制使问题变得简单。在这里取消了这个限制, 而允许通讯线在站点以外的点 (即“虚设站”或 Steiner 点) 连接, 这样可以使线网费用降低, 但问题要复杂得多。

例如, 有三个通讯站, 直角坐标分别为 $a(0,0)$, $b(4,3)$, $c(6,0)$ 。两点间的距离为直角折线距离, 以这三个站为顶点, 距离为边权的加权完全图见图 13.5。

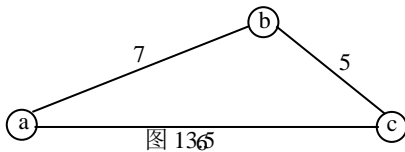


图 13.5

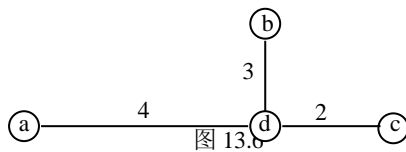


图 13.6

若不许可通讯线在非站点处连接, 则图 13.5 的最小生成树即代表最小费用的线网, 其长度为 11。但若允许通讯线在非站点处连接, 即可引入“虚设站”, 则“虚设站”的个数和位置将是解决问题的关键。若在 $(4,0)$ 处设置一个“虚设站” d , 则 a, b, c, d 四个点的完全图的最小生成树是图 13.6, 其长度为 9。小于不加“虚设站”时的长度 11。

“虚设站” (即 Steiner 点) 的个数和位置是解决问题的关键,

根据提示“Steiner 点位于给定通信站点的 x 坐标线, y 坐标线形成的格点上”, 可知最多有 $n^2 - n = n(n-1)$ 个 Steiner 点的可能位置, 这些位置就是 Steiner 点的候选点。当 $n=9$ 时, 有 $n(n-1)=72$ 个 Steiner 点的可能位置。

V_0 : 给定的 n 个通信站点的集合;

V_p : Steiner 点的候选点集合, 设其点数为 p , $V_p \cap V_0 = \emptyset$;

以 $V = V_p \cup V_0$ 为顶点集作一个加权完全图 K_{p+n} , 其中的边 (u, v) 的权取为点 u 与 v 之间的直角折线距离。我们的问题就是: 求加权完全图 K_{p+n} 中包含 V_0 (也允许包含 V 中的其他点) 的权最小的子树。此即求加权完全图 K_{p+n} 中, V_0 的最小 Steiner 树问题。

求 V_0 的最小 Steiner 树可分解为两个问题:

1) 求 Steiner 点; 2) 求最小生成树。

根据提示“最小 Steiner 树最多只需 $n-2$ 个虚设站 (Steiner 点)”,

V_s : 表示 V_p 中任意 s 个点的集合。

对满足 $0 \leq s \leq n-2$ 的整数 s 和点集 $V_s \subseteq V_p$, 以 $V = V_s \cup V_0$ 为顶点集的加权完全图 K_{s+n} 的最小生成树记为 T_s , 所有 T_s 中权最小者记为 T^* , T^* 即为所要求的最小 Steiner 树。



1) 所有的 T_{vs} 共有多少? 当 $n=9$ 时, 依次用求最小生成树的算法求出一个个的 T_{vs} 是否可行? 如何解决该问题? 比如, 穷举法, 贪婪法。

2) 当 $n=9$ 时, 有 $n(n-1)=72$ 个 Steiner 点的可能位置, 72 可否再减少?

13. 5. 4 问题求解

求最小 Steiner 树问题是 NP 难题, 点数较小的问题可用穷举法, 但若规模较大, 应寻求近似算法。

1. 穷举法

由于费用最少的 Steiner 树 T^* 上最多只需引入 $n-2$ 个虚设点, 因此可从 $m \leq n(n-1)$ 个可能的 Steiner 点位置中任取 s 个点, $s=0, 1, 2, \dots, n-2$, 连同给定的 n 个点一起, 用 Kruskal 算法, 求由这 $n+s$ 个点确定的赋权完全图 (图中边权取为两点间的直角折线距离) 的最小生成树 T_s 。由于从 m 个点中任取 s 个点的取法有 $\binom{m}{s}$, 因此共有 $\binom{m}{0} + \binom{m}{1} + \dots + \binom{m}{s}$ 个可能的 Steiner 点集。每次

迭代, 用 Kruskal 算法对每个可能的 Steiner 点集与给定的 n 个点确定的完全

图的最小生成树需用多项式时间, 共需进行 $\binom{m}{0} + \binom{m}{1} + \dots + \binom{m}{s}$ 次迭代。

若 m 不大, 此法可行, 否则若 m 大, 此法将无效。



n 个通讯站点所在的最小长方形区域的四个拐角区域也不可能有 Steiner 点, 即: 设 $V_0 = \{v_i(x_i, y_i) : i=1, 2, \dots, n\}$, 对每个 $y_k, k=1, 2, \dots, n$, 记

$$x_{00}(k) = \min_{(x_i, y_i) \in V_0, y_i < y_k} \{x_i\}, \quad x_{01}(k) = \min_{(x_i, y_i) \in V_0, y_i > y_k} \{x_i\},$$



$$x_{10}(k) = \max_{(x_i, y_i) \in V_0, y_i < y_k} \{x_i\}, \quad x_{11}(k) = \max_{(x_i, y_i) \in V_0, y_i > y_k} \{x_i\}.$$

则在下述四类区域中不含 Steiner 点

$$D_1 = \{(x, y) \mid x < x_{00}(k), y < y_k\}; D_2 = \{(x, y) \mid x < x_{01}(k), y > y_k\};$$

$$D_3 = \{(x, y) \mid x > x_{10}(k), y < y_k\}; D_4 = \{(x, y) \mid x > x_{11}(k), y > y_k\}.$$

如图 13.7, 星号点是给定的 9 个通讯站点。共有 $n(n-1)=72$ 个 Steiner 点的可能位置, 它们位于过 9 个点的水平线与垂直线的交点上。且由于区域 D_1, D_2, D_3 和 D_4 内不含 Steiner 点, 72 个可能的 Steiner 点位置可减少到 31 个 (图 13.7 中小圆圈所示的 31 个位置)。

$m=31$, 迭代次数减少到 $\binom{31}{0} + \binom{31}{1} + \cdots + \binom{31}{7} = 3572224$ 次。假设每次迭代只

需用 1/60 秒的时间, 3572224 次迭代需要大约 17 个小时。

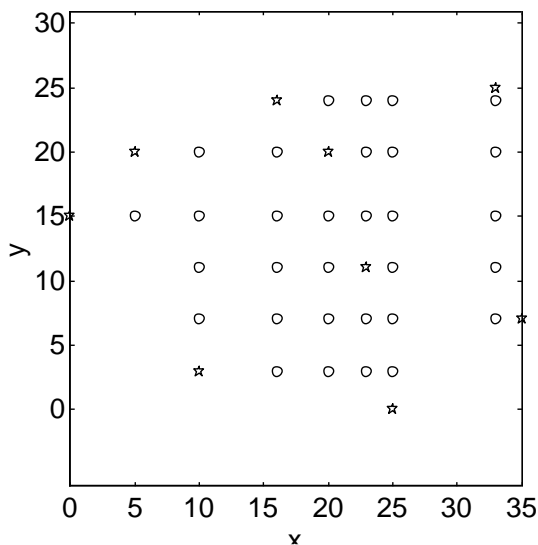
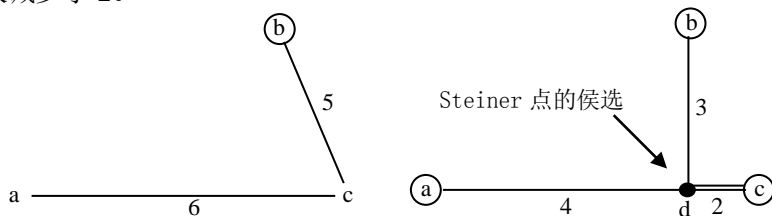


图 13.7 9 个给定点和 31 个可能的 Steiner 点

2. 贪婪试探算法

图 13.5 的最小生成树如图 13.8 (a), 顶点按直角坐标之定位放置, 右边的图是把边画成直角折线, 该图称为其三个点的最小直角折线支撑树。若将图 13.8 的右边图中重边的端点 d 作为虚设站加入, 则 4 个点的最小生成树的权减少了 2。





(a)



(b)

图 13.8

因此, 一般情况下, 可把最小直角折线支撑树中重边的端点作为 Steiner 点的候选点。结合贪婪法的思想, 可构造出如下的试探法, 能否得到正确解, 需要证明。

- 1) 输入给定的 n 个通信站点的坐标;
- 2) 计算最小直角折线支撑树;
- 3) 找重边, 则重边的端点便是 Steiner 点的候选点;
- 4) 分别计算出每个候选点作为 Steiner 点加入后所减少的费用, 该费用称为此点的价值;
- 5) 把最大价值的候选点也作为一个给定点, 重复 1) 直到没有正价值的候选点。



编写上面介绍的贪婪试探算法的 MATLAB 程序, 求给定的 9 个通讯站的最小 Steiner 树

3. 改进型试探算法

如果每次迭代, 都按照随意的顺序加入“虚设站”, 并使得到的最小生成树费用有所减少, 直到已加入 $n-2$ 个“虚设站”, 或加入任何一个剩余的可能的 Steiner 点都不能使费用减少为止。按步骤描述如下

- 1) 求给定的 n 个点的最小生成树, 记录其费用;
- 2) 取一个可能的 Steiner 点加入, 求最小生成树, 若该树的费用小于当前的费用, 则记录此树并更新费用;
- 3) 重复 2) 直到已有 $n-2$ 个 Steiner 点, 或任何剩余的 Steiner 点加入都不能减少费用。



编写上面介绍的改进型试探算法的 MATLAB 程序, 求给定的 9 个通讯站的最小 Steiner 树



贪婪试探算法和改进型试探算法都是近似算法, 对一般的问题

题未必能得到最优解。对本问题给定的 9 个通讯站的情况, 求出的解是否为最小 Steiner 树, 需要证明, 否则, 应分析解的近似程度。

4. 模拟退火法

这是一种通用的随机搜索法, 是解决 NPH 问题的比较有效的方法。

1) 给定点集连同一些虚设点一起构成点集 Z , 求 Z 的最小支撑树, 其费用记为 C , 置 $k=0$;

2) 产生新的点集 S

从以下几种方式中随机选择一种:

- 加入一个新的虚设点
- 去掉一个存在的虚设点
- 移动一个现有的虚设点到一个随机的允许位置

3) 确定新点集 S 的最小支撑树, 其费用记为 C_1 ,

若 $C_1 \leq C$, 则更新 C 为 C_1 , 更新当前点集 Z 为 S , 当 $k=M$ 时停止, 否则 $k=k+1$, 转 2);

若 $C_1 > C$, 则仅以一定的概率 (可取为 $\exp\{-(C_1-C)/T(k)\}$, 其中 T 为一控制参数, 称为温度, 随 k 的增大而减小, 比如取 $T(k)=T(0)/k$, 称为冷却方案) 接受 S 作为当前点集 Z , 转 2)。

用模拟退火法来求图 13.4 所给出的 9 个点的最小 Steiner 树, 在 25MHz 型 386 计算机上运行, 大约 1.5 分中便可得到了最优解。对应于随机数产生器的不同种子的不同运行, 可给出全部五个不同的最优解。在上百次运行中, 模拟退火法都总是收敛到五个最优解中的一个。计算时间方面的优越性表明当该问题的规模较大, 穷举法不可行时, 模拟退火法的价值。

5. 修正的 Prim 启发式算法

受到求最小生成树的 Prim 算法的启发, 根据 Prim 算法的思想, 构造出下面的求最小直角折线 Steiner 树的方法。

先引入一些记号:

Z : 给定的通讯站点集合;

G : 给定通信站点的 x 坐标线, y 坐标线形成的格点全体构成的集合;

T : 当前 Steiner 树的顶点集;

$S=G-Z$;

算法步骤:

1) 选取 Z 中距离最近的两点 $z_i=(x_i, y_i)$, $z_j=(x_j, y_j)$;

2) 这两点的 x 坐标或 y 坐标相同, 则将两点连接起来, 并把该路径

上所有在 G 中的点以及 z_i, z_j 加入 T ，否则

- a) 构造过 (x_i, y_j) 的连接 z_i, z_j 的直角折线路径 $path_1$, 将 $path_1$ 上所有属于 G 的点以及 z_i, z_j 加入 T ;
 - b) 在 $Z-T$ 中找到与当前树距离最近的顶点 z ，其距离记为 $dist1$ ，然后删掉树中的 $path_1$;
 - c) 构造过 (x_i, y_j) 的连接 z_i, z_j 的直角折线路径 $path_1$, 将 $path_1$ 上所有属于 G 的点以及 z_i, z_j 加入 T ;
 - d) 在 $Z-T$ 中找到与当前树距离最近的顶点 z ，其距离记为 $dist2$ ，然后删掉树中的 $path_2$;
 - e) 若 $dist1 < dist2$ ，则加入 $path_1$ ，
若 $dist2 < dist1$ ，则加入 $path_2$ ，
若 $dist1 = dist2$ ，则对下一个最近点重复(2)a 到(2)e, 直到 $dist1 \neq dist2$ ，或穷尽了 Z 中所有顶点（此时任意选择）;
- 3) 取 $z_i \in Z \cap (G-T), z_j \in T$, 使 z_i, z_j 尽可能近;
 - 4) 重复 2), 3) 直到 Z 中的顶点均在 T 中。

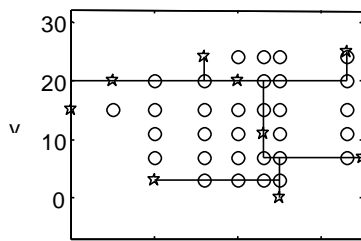
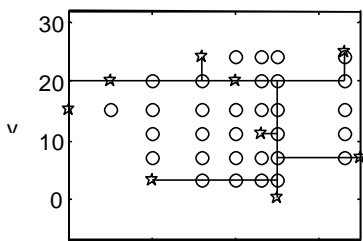
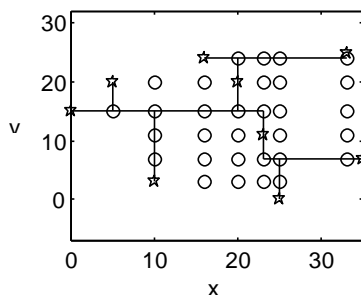
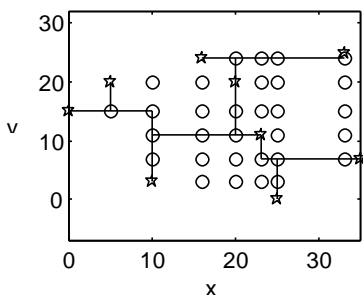
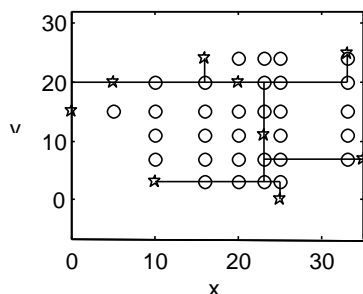


图 13.9



用手工操作的方式，用修正的 Prim 启发式算法求给定的 9 个通讯站的最小 Steiner 树，体会该算法的思想。



提示

最小 Steiner 树问题是 NP 难题。前面介绍的方法，除穷举法之外，每个算法都是有效算法，但不一定能得到最优解，一般需要对解的近似程度进行分析。对算法的好坏，可随机给出一些通讯站，用这些方法来求解，对算法给出的解进行比较，看哪个算法效果最佳。

13.5.5. 结果

用穷举法可得到给定的 9 个通讯站的最小 Steiner 树，共有 5 个，费用为 94。图 13.9 给出了这 5 个最小 Steiner 树，其中每个 Steiner 树都含 4 个或 5 个 Steiner 点。因此可按图 13.9 中任何一个 Steiner 树来设计给定 9 个站的费用最少的局部网络。

§ 13.6 实验

13.6.1 实验一：海底管道设计

某石油公司在墨西哥海湾拥有几个石油钻井平台，每个平台开采出的石油需要运往路易斯安娜的炼油厂。要在平台与路易斯安娜海岸之间建造一个管道网，使石油通过管道传输。这管道网该如何设计，才能使建造费用最低。

图 13.10 中，顶点代表钻井平台及炼油厂，边表示其两端点之间可以铺设管道，边上的权代表该段管道的建设费用。

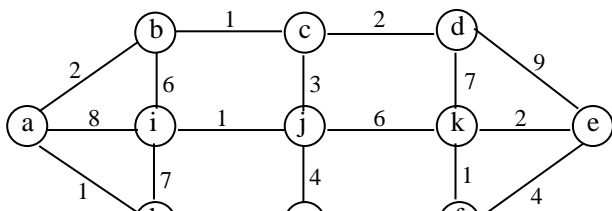


图 13.10

13. 6. 2 实验二：最大容量路径

在一个计算机通讯网络中,某一计算机(顶点)欲呼叫另一台计算机并进行数据传输,若传输数据量很大,又要求了传输速度,则通常需要沿容量最大的路径进行数据传输。

假设该通讯网络对应于图 13.11 的无向图 G , 其上每条边的权代表容量(带宽), 即通过该边的最大流量。求两个给定顶点之间容量最大的路径, 路径的容量为该路径上的最小边容量。

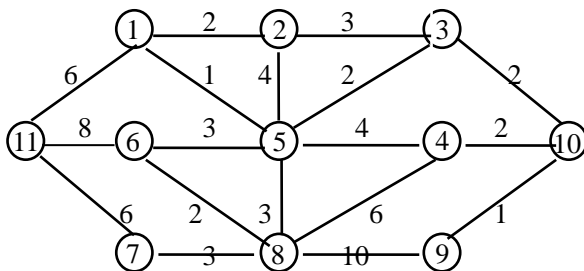



图 13.11



 **提示** G 的最大生成树中的路径均为最大容量路径。

参考文献

- [1] 龚飏, 图论与网络最优化算法, 重庆大学应用数学系, 1998.
- [2] COMAP 著, 申大维等译, 数学的原理与实践, 高等教育出版社, 1998.
- [3] T.McGrath, M.Menzies and C.Smith, Iterative and constructive models for minimal rectilinear Steiner trees. The UMAP Journal Vol.12, No.3, 1991.

- [4] P.J. Melody, H. L. Moore and M. Wood, The construction of a minimal-cost Steiner tree, The UMAP Journal Vol.12, No.3, 1991.
- [5] 李人厚等译, 精通 MATLAB 综合辅导与指南, 西安交通大学出版社, 1998。