第五章　支配集与独立集

5.1　求支配集

一、问题及其分析

要在 $V_1, V_2, \cdots, V_n$ 这 N 个城镇建立一个通迅系统，为此从这 n 个城镇中选定几座城镇，在那里建立中心台站，要求它们与其它各城镇相邻，同时为降低造价，要使中心台站数目最少，有时还会提其它要求。例如在造价最低的条件下，需要造两套(或更多套)通讯中心，以备一套出故障时启用另一套。

例如图 5-1 看作是一个通讯系统。$V_1, V_2, \cdots, V_6$ 是一些城镇，仅当两城之间有直通通讯线时，相应的两顶点连一条边。在讲这个问题的数学模型之前，先讲一个定义：

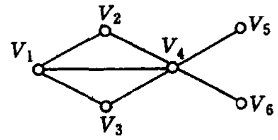

图　5-1

D 是图 G 的一个顶点子集，对于 G 的任一顶点 U，要么 U 是 D 集合的一个顶点元素，要么与 D 中的一个顶点相邻，那么 D 称为图 G 的一个支配集。若在 D 集中去除任何元素后 D 不再是支配集，则支配集 D 是极小支配集。称 G 的所有支配集中顶点个数最少的支配集为最小支配集 D_0，记 $\gamma(G)=D_0$ 中的顶点个数，称作 G 的支配数。

由上述定义可知，凡最小支配集一定是极小支配集；任何一个支配集以一个极小支配集为其子集；G 所含的极小支配集可能有两个以上，而且其顶点数也可以不一致，但支配数 $\gamma(G)$ 是唯一的。

显然中心台站的选址问题，实际上是求一个能与 G 中其它顶点相邻的顶点集合，且这个顶点集合所含的顶点元素必须最少，即求图的最小支配集。若建两套，则从一切极小支配集 $D_1, D_2, \cdots, D_d$ 中选取 D_m 和 D_n，使得 $D_m \cap D_n = \varphi$(φ 为支配数，愈小愈好)，即其中一套出故障时不影响另一套工作。并且 $|D_m \cup D_n| = \min\{|D_i| + |D_j| \mid 1 \leqslant i, j \leqslant d, D_i \cap D_j = \varphi\}$，即两套互不干扰的通讯中心所选定的城镇数最少。

例如在图 5-1 所示的 6 个城镇中建中心台站，方案有 $\{V_1, V_5\}$，$\{V_1, V_6\}$，$\{V_4\}$，$\{V_2, V_3, V_5\}$，$\{V_2, V_3, V_6\}$。这些顶点集合为极小支配集。若建一套通讯中心，只需建立在 V_4 城；若建两套，则 V_4 建一套，$\{V_1, V_5\}$ 或者 $\{V_1, V_6\}$ 建第 2 套。

有许多实际问题可以转化成上述这种数学模型来处理。

支配集有哪些性质呢？

1. 若 G 中无零次顶点($d(v)=0$)，则存在一个支配集 D，使得 G 中除 D 外的所有顶点也组成一个支配集；

2. 若 G 中无零次顶点($d(v)=0$)，D_1 为极小支配集，则 G 中除 D_1 外的所有顶点组成一个支配集。

求所有极小支配集的计算，包括 5.2 节中的求所有极小覆盖集的计算都是采用逻辑

运算的。

设 X,Y,Z 三条指令：

规定："要么执行 X，要么执行 Y" 记作 $X+Y$（和运算）

"X 与 Y 同时执行" 记作 XY （与运算）

上述逻辑运算有以下运算定律：

1. 交换律

$X+Y=Y+X$；$XY=YX$

2. 结合律

$(X+Y)+Z=X+(Y+Z)$；$(XY)Z=X(YZ)$

3. 分配律

$X(Y+Z)=XY+XZ$；$(Y+Z)X=XY+XZ$、

4. 吸收律

$X+X=X$；$XX=X$；$X+XY=X$

上述定律尤其是吸收律，在求所有极小支配集和极小覆盖集的两个公式中用处颇大。求所有极小支配集的公式：

$$\varphi(V_1,V_2,\cdots,V_n)=\prod_{i=1}^{n}\Big(V_i+\sum_{U\in N(V_i)}U\Big)$$

一个顶点同与它相邻的所有顶点进行加法运算组成一个因子项，几个因子项再连乘。连乘过程中根据上述运算规律展开成积之和形式。每一积项给出一个极小支配集，所有积项给出了一切极小支配集，其中最小者为最小支配集。

例如求图 5-1 通讯网的一切极小支配集及支配数。

$\varphi(V_1,V_2,V_3,V_4,V_5,V_6)$

$=(V_1+V_2+V_3+V_4)(V_2+V_1+V_4)(V_3+V_1+V_4)(V_4+V_1+V_2+V_3+V_5+V_6)$

$(V_5+V_4+V_6)(V_6+V_4+V_5)$

$=(1+2+3+4)(2+1+4)(3+1+4)(4+1+2+3+5+6)(5+4+6)(6+4+5)$

$=15+16+4+235+236$

故得所有极小支配集如下：

$\{V_1,V_5\},\{V_1,V_6\},\{V_4\},\{V_2,V_3,V_5\},\{V_2,V_3,V_6\}$

$\gamma(G)=1$

二、求极小支配集和支配数的程序

下面根据公式和运算定律，给出求所有极小支配集和支配数的程序。

```
Program ji_xiao_zhi_pei_ji;

const
  maxn          =   30;

type
  ghtype        =   array [1..maxn,1..maxn] of integer;
```

```
  settype        =    set of 1..maxn;
  ltype          =    array [1..maxn] of settype;

var
  g              :    ghtype;      { 邻接矩阵 }
  n,l            :    integer;     { 顶点数,极小支配集的个数 }
  f              :    text;        { 文件变量 }
  lt             :    ltype;       { 极小支配集 }

procedure read_graph;
  var
    str : string;
    i,j : integer;
  begin
    write('Graph file = '); { 输入文件名,并与文件变量连接 }
    readln(str);
    assign(f,str);
    reset(f);
    readln(f,n);           { 输入顶点数 }
    for i : =1 to n do
      for j : =1 to n do read(f,g[i,j]); { 输入图的邻接矩阵 }
    close(f);
  end;

procedure reduce(s : settype);
{ 根据吸收律求 s+lt[1]+lt[2]+…+lt[l]的结果 }
  var
    i,j : integer;
  begin
    i : =1;
    while i<=l do { 检查所有支配集 }
      begin
        if s * lt[i]=lt[i] then exit;
        { 若第 i 个支配集是 s 的子集,则退出过程 }
        if s * lt[i]=s
        { 若 s 是第 i 个支配集的子集,则删除该极小支配集 }
           then begin
                  for j : =i+1 to l do lt[j-1] : =lt[j];
                  dec(l)
                end
           else inc(i)
        { 否则检查下一个支配集,即删除所有含 s 子集的支配集 }
        { 直至某支配集作为 s 的子集或所有支配集与 s 非子集关系为止 }
      end;
    inc(l);lt[l] : =s { s 作为最后一个支配集 }
  end;

procedure think;
  var
```

```
    tl,i,j,k : integer;
    t : ltype;
  begin
    l : =0;
    for i : =1 to n do if (i=1) or (g[1,i]>0) then reduce([i]);
    { 建立 v1+ ∑(u∈N(v1)) u,各项存入 lt }
    for i : =2 to n do
      begin
        t : =lt;tl : =l; { 暂存所有支配集 lt 和支配集个数 }
        l : =0;
        for j : =1 to n do
        { 求(v2+ ∑(u∈N(v2)) u) (v3+ ∑(u∈N(v3)) u)…(vi+ ∑(u∈N(vi)) u)的所有乘积项 lt[1]…lt[l] }
          if (i=j) or (g[i,j]>0)
             then for k : =1 to tl do
                     reduce(t[k]+[j])
      end;
    lt : =t;l : =tl { 求出 L 个极小支配集 lt }
  end;

procedure print; { 打印 l 个极小支配集中的所有元素 }
  var
    i,j : integer;
  begin
    for i : =1 to l do
      begin
        write(i : 3);
        for j : =1 to n do if j in lt[i] then write(j : 3);
        writeln
      end
  end;

begin
  read_graph;  { 输入图 }
  think;       { 计算极小支配集 }
  print        { 输出结果 }
end.
```

5.2 求独立集

一、问题及其分析

$S=\{S_1,S_2,\cdots,S_n\}$为信息传送的基本信号集合。

可以把信号 S_i 设想成汉字或拉丁字母。统计规律表明哪些信号与哪些信号易于发生错乱。例如输入 S_i,输出应该是 S_i $(1\leqslant i\leqslant 5)$。但由于干扰发生了错乱,$S_1$ 可能和 S_2 错乱,S_1 还可能和 S_5 错乱等。例如已知错乱可能性如图 5-2(a)所示。

为了确切地从输出信号得知输入信号,我们不能选用 $S_1,S_2,\cdots,S_n$ 中的每一信号,只

能从中选一部分用于输出。那么选哪一部分信号作输入源呢？我们作另外一张图，顶点表示信号，若信号源 S_i 可能输出 S_j，则在 S_i 与 S_j 之间连一条边，如图 5-2(b)。为了使可用于输出的信号最多，我们在图 5-2(b)上求这样一个顶点集合 I，I 中任两个顶点不相邻且这些独立的顶点数最多。例如可以选$\{S_1,S_3\}$作为输出信号；或选$\{S_1,S_4\}$或$\{S_2,S_4\}$或$\{S_2,S_5\}$或$\{S_3,S_5\}$做输出信号。这个数学模型就是所谓的求最大独立集问题。

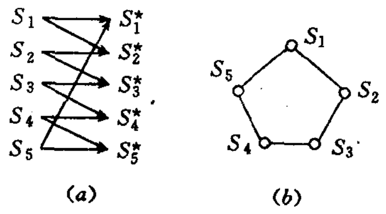

图 5-2

在实际生活中，求最大独立集的应用实例很多，如 8 皇后问题。我们将棋盘的每个方格看作一个顶点，放置的皇后所在格与她能攻击的格看成是有边连接，从而组成 64 个顶点的图。若要设计一个算法，放置尽可能少的皇后，使她们控制棋盘上全部格，这显然是求最小支配集问题；再设计一个算法，要放置尽可能多的皇后，而相互不攻击地共处，这个问题就是求最大独立集问题了。

在讲求最大独立集的定义之前，我们先引出与独立集密切相关的覆盖集的概念：K 是 G 图的一个顶点子集且 G 的每一边至少有一个端点属于 K，则称 K 是 G 的一个覆盖。这里覆盖一词的含义是顶点覆盖全体边。若在 K 集中去除任一顶点后将不再是覆盖，则称 K 为极小覆盖。在图 G 的所有极小覆盖集中含顶点数最少的极小覆盖称作最小覆盖。用 $\alpha(G)$表示最小覆盖的顶点数，又称 G 的覆盖数。

例如图 5-3 中的黑色顶点是一个极小覆盖，同时也是一个最小覆盖，$\alpha(G)=4$，$K=\{V_1,V_3,V_4,V_6\}$。下面给出独立集的概念：

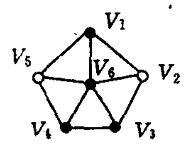

图 5-3

I 是 G 的一个顶点子集，I 中任两个顶点不相邻，则称 I 是图 G 的一个独立集。若独立集 I 中增加任一个除 I 集外的顶点后 I 不是独立集，则称 I 是极大独立集。在图 G 的所有极大独立集中含顶点数最多的极大独立集称作最大独立集。用 $\beta(G)$表示最大独立集的顶点数，又称 G 的独立数。

那么独立集又有什么性质呢？我们从独立集、覆盖集、支配集之间关系中论述：

1. 一个独立集是极大独立集，当且仅当它是一个支配集；

2. I 是独立集，当且仅当 G 中除 I 集外的所有顶点是一个覆盖集；

I 是极大(最大)独立集，当且仅当 G 中除 I 集外的所有顶点是一个极小(最小)覆盖集，即

$$\alpha(G)+\beta(G)=G\text{ 的顶点数}$$

3. 极大独立集必为极小支配集；但是极小支配集未必是极大独立集。

例如一个含边数为 4 的圈上的两个相邻顶点是极小支配集，但不能为极大独立集，连独立集也不是。

例如图 5-3 中$\{V_1,V_3,V_4,V_6\}$是一个最小覆盖集，$V-K=\{V_2,V_5\}$是一个最大独立集同时又是一个极小支配集。注意图 5-3 中的最小支配集是$\{V_6\}$不是独立集。

由于极大独立集与极小覆盖集有互补性，我们这里仅给出求所有极小覆盖集的计算

公式，读者可以由此推出极大独立集：

$$\varphi(V_1,V_2,\cdots,V_n)=\prod_{i=1}^{n}\Big(V_i+\prod_{U\in N(V_i)}U\Big)$$

某顶点的所有相邻顶点进行积运算后再与该顶点进行和运算，组成一个因子项。几个因子项连乘，并根据逻辑运算定律展开成积之和形式。每一积项给出一个极小覆盖集，所有积项给出了一切极小覆盖集，其中最小者为最小覆盖集。

例如求图 5-4 中的一切极小覆盖集和覆盖数 $\alpha(G)$ 以及一切极大独立集和独立数 $\beta(G)$。

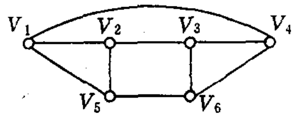

图 5-4

$\varphi(V_1,V_2,\cdots,V_6)$

$=(1+246)(2+136)(3+246)(4+135)(5+346)(6+125)$

$=2456+1356+1346+2346+1245+1235$

即得一切极小覆盖集如下

$$\{V_2,V_4,V_5,V_6\},\{V_1,V_3,V_5,V_6\},$$
$$\{V_1,V_3,V_4,V_6\},\{V_2,V_3,V_4,V_6\},$$
$$\{V_1,V_2,V_4,V_5\},\{V_1,V_2,V_3,V_5\}。$$
$$\alpha(G)=4$$

由于 G 的除极小覆盖集外的顶点可组成极大独立集，由此可以得出一切极大独立集为

$$V-\{V_2,V_4,V_5,V_6\}=\{V_1,V_3\};$$
$$V-\{V_1,V_3,V_5,V_6\}=\{V_2,V_4\};$$
$$V-\{V_1,V_3,V_4,V_6\}=\{V_2,V_5\};$$
$$V-\{V_2,V_3,V_4,V_6\}=\{V_1,V_5\};$$
$$V-\{V_1,V_2,V_4,V_5\}=\{V_3,V_6\};$$
$$V-\{V_1,V_2,V_3,V_5\}=\{V_4,V_6\}。$$
$$\beta(G)=2$$

可见求最大独立集，首要的是求极小覆盖集。

二、求所有极小覆盖集的程序

下面我们给出求所有极小覆盖集的程序。

```
program ji_xiao_fu_gai_ji;

const
  maxn        =   30;

type
  ghtype      =   array [1..maxn,1..maxn] of integer;
  settype     =   set of 1..maxn;
  ltype       =   array [1..maxn] of settype;

var
  g           :   ghtype;     { 邻接矩阵 }
```

```
  n,l          :  integer;    { 顶点数,极小覆盖集个数 }
  f            :  text;       { 文件变量 }
  lt           :  ltype;      { 极小覆盖集 lt[i]—第 i 个极小覆盖集 1≤i≤l }
procedure read_graph;
  var
    str : string;
    i,j : integer;
  begin
    write('Graph file = ');{输入文件名并与文件变量连接 }
    readln(str);
    assign(f,str);
    reset(f);
    readln(f,n); { 输入顶点数和图矩阵 }
    for i : =1 to n do
      for j : =1 to n do read(f,g[i,j]);
    close(f);
  end;

procedure reduce(s : settype); { 根据吸收律求 s+lt[1]+…+lt[l]的结果 }
  var
    i,j : integer;
  begin
    i : =1;
    while i<=l do
      begin
        if s * lt[i]=lt[i] then exit;
        if s * lt[i]=s
           then begin
                   for j : =i+1 to l do lt[j-1] : =lt[j];
                   dec(l)
                end
           else inc(i)
      end;
    inc(l);lt[l] : =s
  end;

procedure think;
  var
    tl,i,j,k : integer;
    t : ltype;
    s : settype;
  begin
    lt[1] : =[1];lt[2] : =[];l : =2;
    for i : =2 to n do if g[1,i]>0 then lt[2] : =lt[2]+[i]; { 建立 v1 + ∏_{u∈N(v1)} u }

      for i : =2 to n do { 求 ∏_{i=1}^{n} (vi + ∏_{u∈N(vi)} u) }
```

```
      begin
        t := lt;tl := l;
        l := 0;s := [];
        for j := 1 to n do if g[i,j]>0 then s := s+[j];
        { 求所有与顶点 i 相连的顶点集合 s }
        for k := 1 to tl do
          begin
          reduce(t[k]+s);reduce(t[k]+[i])
            { 根据分配律求(t[1]+…+t[l])(s+[i])的所有乘积项 }
          end
      end;
      lt := t;l := tl
  end;

procedure print;
  var
    i,j : integer;
  begin
    for i := 1 to l do { 打印每个乘积项(极小覆盖集)的元素 }
      begin
        write(i : 3);
        for j := 1 to n do if j in lt[i] then write(j : 3);
        writeln
      end
  end;

begin
  read_graph;  { 输入图 }
  think;       { 计算极小覆盖集 }
  print        { 输出结果 }
end.
```