求最大匹配的一个新算法*

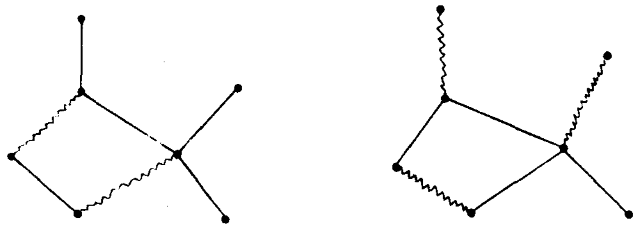
戴　一　奇

（清华大学，北京）

简单无向图G的最大匹配问题分为二部图和一般图的最大匹配两类．前者主要采用可增广路的思想解决，〔1〕中已经详述；本文主要讨论有关后者的算法．

定义　设G=（V，E）是简单无向图，在G的所有匹配M′中，若 |M|= max |M′|，则称M是G的一个最大匹配．

例如图1（b）是该图的一个最大匹配（〰〰表示匹配边），（a）则不是．



（a）　　　　　　（b）

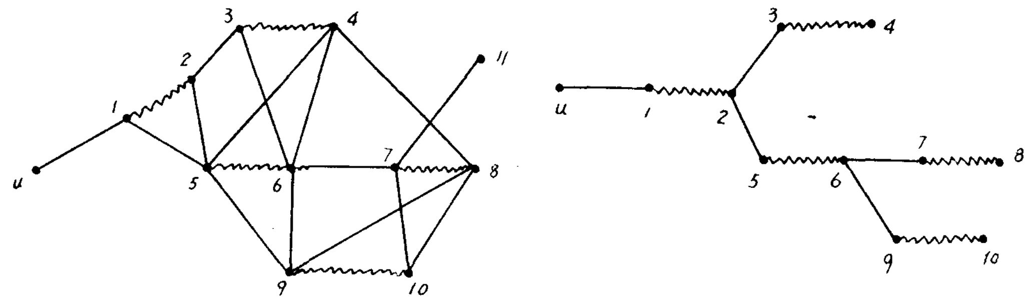
图　1

在求一般图的最大匹配时，可以借用二部图中找增广路的思想，但不能简单套用，否则会使算法无效．到目前为止，除了采用Edmonds提出的树花（blossom）理论，其它各种尝试都已证明无效．

一、已有算法介绍　已有算法中具有代表性的是Edmonds（1965）和Gabow（1976）算法．因为每个算法都很复杂，所以只准备简单介绍其中有关交互树T的生长与树花B的处理两个核心部分．详细内容可参阅〔2〕,〔3〕,〔4〕．

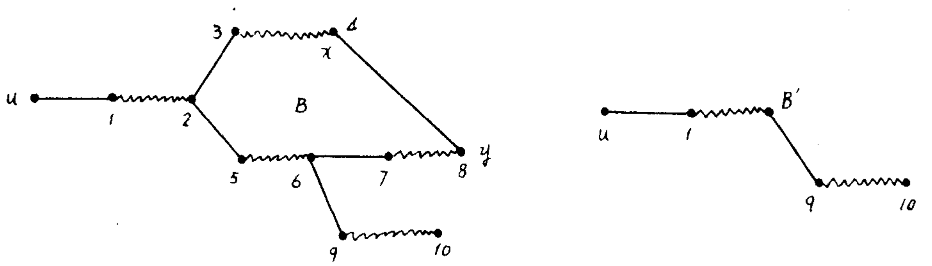
任给一初始匹配M_0，从非饱和点u开始寻找交互道，其搜索过程将是树形结构，称之为交互树T，如图2．T中到根u的路径长度为偶数的顶点称为外点，如2，4，8；否则为内点，如3，7，9．交互树T中任意顶点v到根u只有唯一交互道P(v，u)，而且仅当非饱和点$u'(\neq u)$与T的外点相邻时才有增广路P(u'，u)．更为重要的是T中的内点并非固定不变，当两个外点x，y相邻时，加入边（x，y）后，T中将构成一个唯一回路，该回路中的所有内点i亦具有经过边（x，y）的唯一偶长路P（i，u），因此i也变成为外点．这种回路就叫做树花，记为B．例如图2（b）中加入边（4，8）便形成树花，如图3(a)．花中离根u最近的顶点称为花基（或者说是P（y，u）与P（x，u）的公共顶点中与u距

* 1986年7月5日收到．



G　　　　　　　　T

图 2



（a）　　　　　　　（b）

图 3

离最远的顶点），记为 b，例如顶点 2．可以证明，树花中任意外点 i 到该花基的偶长交互道必定是它到非饱和点 u 的偶长交互道的一部分．由于树花中各顶点都成为外点，因此可以将 B 收缩成一个伪点 B′，然后继续生长交互树，如图3(b)．这种收缩可以多次进行，同时新的交互树不含任何回路，从而保证树 T 中的任一点 i 都有唯一交互道 P(i, u)．

求最大匹配的算法就是不断由外点生长交互树 T，遇花收缩，然后继续生长．如遇到非饱和点 u' 则找到增广路，沿该路改善匹配．如此反复进行，直至不存在任何新的增广路．

已有算法的主要理论依据有

引理 1　以 u 为根的交互树 T 在生长中只有三种可能结果：a）形成匈牙利树，即不存在关于 u 的增广路；b）找到一条增广路；c）形成树花 B．

引理 2　若以 u 为根的交互树 T 是（G, M）的一棵匈牙利树，则以非饱和点 u' 为一端点的任意一条增广路 P 都不包含 T 上的任一顶点．

因此若 T 是匈牙利树，令 G←G—T，只需对新的图 G 进行计算就可以了．以上两个引理，也同样适用于新算法．

最大匹配算法的关键在于：（1）如何处理树花，（2）如何确定唯一增广路．Edmonds 算法和 Gabw 算法都使用广探法生长树 T，前者将树花 B 真正进行了收缩，而后者采用了一些巧妙的处理方法，使之较为简单．它们的复杂性分别是 $O(n^4)$ 和 $O(n^3)$．本文借鉴了 Gabow 算法的一些技巧，提出了一种以深探法为基础的新算法．

二、新算法的主要数据结构及说明

算法的粗框如下：

A．主程序（PRO—A）．用深探法生长交互树T，由外点k(最初是非饱和点u）生长，如果找到新的外点i，则由i继续生长．

B．树花处理子程序（PRO—B）．T在生长中构成树花B时调用此过程，它确定花基、将B的内点变为外点并确定唯一偶长路．

C．确定增广路子程序（PRO—C）．T在生长中找到增广路后调本过程．它确定增广路，改善匹配并为生长新的交互树做准备．

为了便于理解算法，有必要先介绍若干数组及其使用方法．设G有n个顶点m条边，顶点编号分别为$1,2,\cdots,n$；边编号为$1,2,\cdots,m$．设置的主要数组是：

1．Mate(1：n)

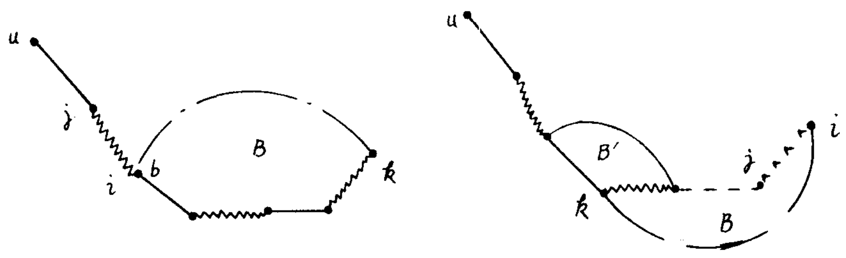
$$m(i)=\begin{cases} j & 边(i,\ j)\in M \\ 0 & 其它 \end{cases}$$

它的作用是：a）若$m(i)=j$，则$m(j)=i$，即$(i,\ j)\in M$．b）当k是外点且$(i,\ k)\in G$时，若$m(i)=0$，则$P(i,\ u)$是可增广路；若$m(i)=j\neq 0$，则i是内点，j是T的新外点，从而使T得到生长．c）修改Mate的值可以记录新匹配．

2．First(1：n)

$$f(i)=\begin{cases} 0 & 初始 \\ j & j是P(i,u)中离外点i最近的内点 \\ -j & 同上，但此时i是深探时已从T中退出的外点 \end{cases}$$

该数组主要用于收缩树花及确定花基．a）当形成树花B后，必定$f(i)=j,\forall i\in B$，j是离花基b最近的内点．所以由$f(i)$可将B处理得象一个伪点．b）从外点k深探到另一外点i时，



（a）i是花基　　　　（b）k是花基

图 4

若$f(i)=j>0$，则i是树花的基；若$f(i)=-j$，则k是花基，如图4．

3．Edge(1:m，1:2)

$$\left.\begin{aligned} e(l,1)&=x \\ e(l,2)&=y \end{aligned}\right.\quad 第l条边的顶点序是(x,\ y)$$

它的作用有：a）深探中若是从y找到邻点x构成树花时，将x与y的位置对换．b）与数组Label配合使用．

4．Label(1 : n)

$$l(i)=\begin{cases}0 & i\notin T\\ j & (0<j\le n)\ i\text{是外点，}j\text{是}P(i,u)\text{中离}i\text{最近的外点}\\ -1 & i\text{是内点}\\ l+n & i\text{是花中内点，}i\text{处于交互路}P(x,u)\text{上}\\ -(l+n) & i\text{是花中内点，}i\text{处于交互路}P(y,u)\text{上.}\end{cases}$$

其中最后两式表示加入边 (x,y)（它处于数组Edge的第 l 行）形成树花B以后，花中各内点的Label值．

下面介绍它们在一些关键步骤中的使用．

（一）交互树T的生长

从 u 开始找一邻点 v_1，若 $m(v_1)=0$，则 $P(v_1,u)$ 是增广路．否则置 $l(v_1)=-1$，$lm(v_1)=\text{Label}(\text{Mate}(v_1))=u$，$fm(v_1)=v_1$．然后从 $m(v_1)$ 深探．例如图5(b) 及表 1 表示树T的生长情况．

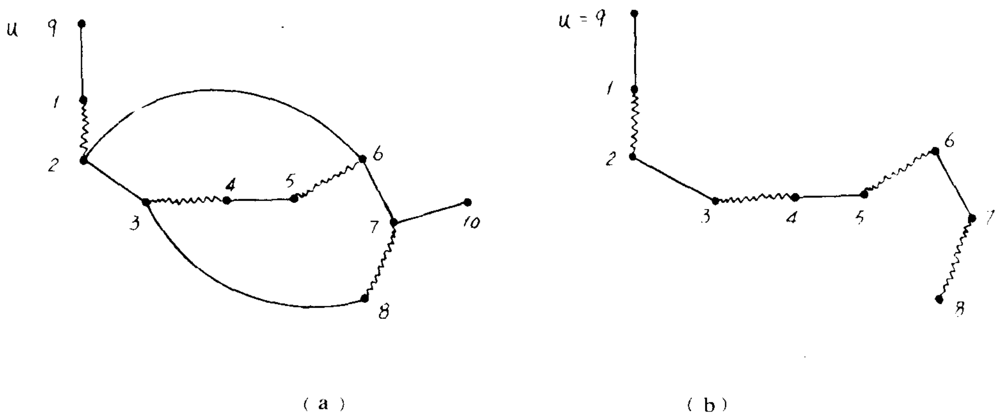


图 5

Edge

l		
$l=1$	9	1
2	1	2
3	2	3
4	2	6
5	3	4
6	3	8
7	4	5
8	5	6
9	6	7
10	7	8
11	7	10

i	Mate	Label	First
$u=9$	0	0	0
1	2	−1	0
2	1	9	1
3	4	−1	0
4	3	2	3
5	6	−1	0
6	5	4	5
7	8	−1	0
8	7	6	7

表 1

（二）树花处理

当从外点 x 探到外点 y 时将构成树花B．若 $f(y)>0$，则 $y\in P(x,u)$ 且 y 是花基；否则

$x \in P(y, u)$ 且为花基．这样可以从x(或y) 依序列

$x, f(x), lmf(x), flmf(x), (lmf)^2(x), \cdots, y, f(y), \cdots$(或 $y, f^*(y), lmf^*(y), f^*lmf^*(y), (lmf^*)^2(y), \cdots, x, f(x), \cdots$其中$f^*(y)$ 表示 $|f(y)|$)

有偶长交互道经由花基退回到u．如果仅需确定树花，只要退到y(或x) 即可．这时将该树花中各点的First值置$f(y)$，各内点的Label值置$l+n$；如果由y退回，则分别是$f(x)$ 和 $-(l+n)$．

例如图5中通过栈由外点v_8退回到v_6，再由外点v_6探到另一个外点v_2($f(v_2)>0$) 时，形成树花B_1，v_2是花基．所以$v_2 \in P(v_6, u)$，应将$v_3 \sim v_6$的First值改为1，v_3，v_5的Label值改为$l+n=14$，同时要修改Edge(4) 的值为(6，2)，以保证顶点序一致．这时各数组元素的值如表2所示．

i	Mate	Label	First
$u=9$	0	0	0
1	2	-1	0
2	1	9	1
3	4	14	1
4	3	2	1
5	6	14	1
6	5	4	1
7	8	-1	0
8	7	6	-7

表 2

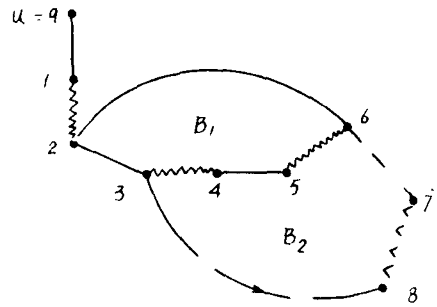


图 6

当再由栈退回到v_3(此时v_3已是外点) 时，v_3有一邻点是已退栈外点v_8($f(v_8)=-7$)，又形成新的树花B_2，如图6．这时v_3是花基．所以顶点v_7，v_8的l，f数组值修改为

i	Mate	Label	First
7	8	-16	1
8	7	6	1

因此树花的确定十分方便．这种处理方法的根据是

引理3 当从外点x深探到T中外点y($f(y)>0$) 形成树花时，y是该树花的基．

证明 因为树T由深探法生长得到，因此y先于x进入T．设T的根是u，那么交互道$P(y, u)$ 与$P(x, u)$ 至少有一个公共顶点u，假定b是离u最远的一个公共顶点，若$y \notin P(x, u)$，显然$b \neq y$，又因为$b \in P(y, u)$，可见T的生长过程一定是由y退回到b，再由b推进到x而得，但此时y已不在T中，与题意矛盾．因此$y \in P(x, u)$，由花基定义，$b=y$．

引理4 当从外点x深探到已不在T上的外点y($f(y)<0$) 时，x是该树花的基．

证明 设$x \notin P(y, u)$，则有花基$b \neq x$，因为$y \notin T$，x一定是由y退回到b，再由b深探而得．但从y退回是由于y或没有邻点，或邻点全部是内点而致，这与当时$x \notin T$相矛盾(否则可以从y生长到x)．因此$x \in P(y, u)$ 且是花基．

这两个引理保证了确定树花时，只需从一个顶点x(或y) 退回即可，且交互道$P(x, y)$

恰好通过了树花的全部顶点．而Gabow等算法需要分别从x，y退回到根u，确定两条路径序列，然后再逐一推进判断序列中对应值是否相等来决定花基和树花．比较起来新算法更为方便．

（三）确定增广路

确定增广路$P(u',u)$并改善匹配主要使用Label和Mate数组，当从离u'最近的外点i退回时，

（a）若i不在树花中（$0<l(i)\leq n$），则

$$P(i, u)=(i, m(i))\cup(m(i), l(i))\cup P(l(i), u)$$

并置$(m(i),l(i))$为匹配边，$(i, m(i))$为非匹配边，再令$i\leftarrow l(i)$，可以递推进行．

（b）若i在树花B中

1．若i是外点（$0<l(i)\leq n$），处理同（a）．

2．若i是内点（$|l(i)|>n$），如果$l(i)>0$，则

$$P(i, u)=P(i, x)\cup(x, y)\cup P(y, u)$$

否则

$$P(i, u)=P(i, y)\cup(x, y)\cup P(x, u)$$

其中$P(i, x)$，$P(i, y)$恰是该树花中$P(x, i)$，$P(y, i)$的倒向路径，它们的判断过程亦与（a）类似．

例如图6中假定有边（v_7，u'），则$P(u', u)$是增广路，它的回退过程是

$$P(u', u)=(u', v_7)\cup P(v_7, u)$$
$$P(v_7, u)=P(v_7, v_8)\cup(v_8, v_3)\cup P(v_3, u)$$
$$P(v_3, u)=P(v_3, v_6)\cup(v_6, v_2)\cup P(v_2, u)$$

因此寻找增广路的复杂性仅与$P(u', u)$中的顶点数成正比，亦较Gabow等算法方便．

至此已对新算法及其数据结构进行了较详尽的说明．下面给出其具体步骤．

三、求最大匹匹配的新算法

（一）生长交互树（PRO—A）

1．初始化有关数组，任给一初始匹配M．

2．若G的全部顶点都已饱和，则$\cup M_{T_i}\cup M$是最大匹配（最初$\cup M_{T_i}=\phi$）．结束．

3．确定一非饱和点u，$k\leftarrow u$，生长交互树T．

4．对外点k，寻k的尚未判断的邻点i，

4.1 若i是非饱和点（$m(i)=0$），则有增广路，调PRO—C；转2．

4.2 若$i\notin T\wedge m(i)=j$，则$l(i)=-1$，$l(j)=k$，$f(j)=i$，k入栈，$k\leftarrow j$转4．

4.3 若i是生长T时曾探过的外点（$l(i)>0\vee l(i)<-n$），则构成树花，调PRO-B；转4．

4.4 若k无新邻点且栈空，则T是匈牙利树．记下T中匹配M的边集M_{T_i}，令$G\leftarrow G-T$，转2；若栈非空，令$f(k)=-f(k)$，退栈，$k\leftarrow$栈顶元素，转4．

（二）PRO—B

（从外点k探到外点i，设边（k，i）的编号为l，即$e(l, 1)=k$，$e(l, 2)=i$）

1.1 若$f(i)>0$，则$i\in P(k, u)$，沿$P(k, u)$依次退至i．令$P(k, i)$中各点j的

$f(j)=f(i)$，各内点j_0的$l(j_0)=l+n$.

1.2 若$f(i)<0$，则$k\in P(i, u)$，沿$P(i, u)$依次退至k. 令$P(i, k)$中各点j的$f(j)=f(k)$，各内点j_0的$l(j_0)=-(l+n)$；$e(l, 1)$与$e(l, 2)$的内容对换.

2. 花中各内点j_0逐一入栈（已成为新外点）.

3. 返回.

（三）PRO—C

（由外点k退回找增广路）

1.

1.1 若$k=u$，则$m(u')=k$，$m(k)=u'$，转2.

1.2 若$l(k)=j\ (0<j\leq n)$，则

$$P(k, u)=(k, m(k))\cup(m(k), l(k))\cup P(l(k), u)$$

令$m(m(k))=l(k)$，$m(l(k))=m(k)$，$k\leftarrow l(k)$，转1.

1.3 若$l(k)=l+n$，令$r\leftarrow e(l, 1)$，$s\leftarrow e(l, 2)$；若$l(k)=-(l+n)$，令$r\leftarrow e(l, 2)$，$s\leftarrow e(l, 1)$；执行

$$P(k, u)=P(k, r)\cup(r, s)\cup P(s, u)$$

其中$P(k, r)$是$P(r, k)$的倒向路，处理过程类似1.2；最后$k\leftarrow s$，转1.

2. 初始化First, Label数组，返回主程序.

算法复杂性分析

交互树T生长时，如果不产生树花，则每一条边都不需重复探测，其复杂性是$O(m)$，如果产生树花B，设它包含t个顶点，确定它只需回退t条边. 由于数组First将树花处理得象一个顶点，所以当产生新的树花B′时，如果它包含了原树花B中若干顶点，由于已将B视为一个伪点，所以在确定B′时，必不涉及B中的那些边. 因此即使产生树花，也没有一条边需要探测二次以上. 所以生长交互树的复杂性是$O(m)$.

增广路的确定只与$P(u', u)$中所含的顶点数相关，其复杂性为$O(n)$.

综上，确定一棵可增广的交互树T的复杂性是$O(m)$.

对n个顶点的图G来说，最多有$\frac{n}{2}$棵这样的交互树. 因此新算法的总时间复杂性是$O(m\cdot n)$.

最后再举一例说明该算法的应用.

设图7(a)以u为根的交互树T的生长如(b)所示，它的深探过程是

u, 1, 2, 3, 4, 5, 6, 7, 8, 7, 6*, 5, 3**, 7, 9, 10, u'

*该处加入边$l_1=(6, 2)$形成$B_1=\{2,3,4,5,6\}$，

**该处加入边$l_2=(8, 3)$形成$B_2=\{7,8,B_1\}$.

当找到增广路$P(u', u)$后

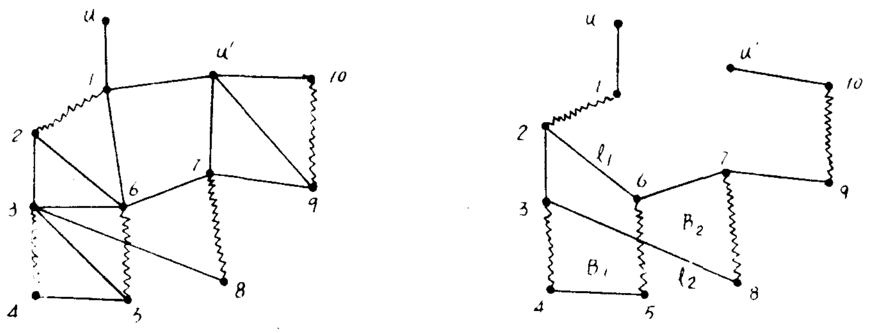
$$P(u', u)=(u', 10)\cup(10, 9)\cup(9, 7)\cup P(7, u)$$

7是B_2的内点，由$l(7)=-(l_2+n)$知$7\in P(8, u)$，故

$$P(7, u)=P(7, 8)\cup(8, 3)\cup P(3, u)$$

3是B_1的内点，由$l(3)=l_1+n$知$3\in P(6, u)$，故

$$P(3, u)=P(3, 6)\cup(6, 2)\cup P(2, u)$$



（a） （b）

图 7

因此增广路是（u, 1, 2, 6, 5, 4, 3, 8, 7, 9, 10, u'）. 改善匹配使 u 和 u' 成为饱和点.

参 考 文 献

〔1〕卢开澄，图论及其应用，清华大学出版社，1981.

〔2〕J. Edmonds, Paths, Trees and Flowers, Canad. J. Math., Vol 17, 449—467, 1965.

〔3〕山东数学会，图与网络算法，1980.

〔4〕H.N·Gabow, An efficient implementation of Edmoncls' algorithm for maximum matching on graphs, J. ACM, Vol 23, 221—234, 1976.

〔5〕M·Swamy and K·Thulasiraman, Graphs, Networks and Algorithms, John Wiley Sons. Inc., 1981.

〔6〕卢开澄，组合数学——算法与分析，清华大学出版社，1983.

A New Algorithm for Constructing A Maximum Matching on a Graph

Dai Yiqi

(Tsinghua University, Beijing)

Abstract

A new algorithm is proposed for constructing a maximum matching in a simple undirected graph. It uses DFS method and proper datastructure to find a alternating tree. In this way, the treatment of blosson and the determination of augmementing path become very simple. The complexity of the algorithm is $O(mn)$.