计算机科学2000Vol. 27№. 2

图的极小顶剖的有效枚举算法*)

Efficient Algorithms for Enumerating all Minimal Separators in a Graph

许胤龙　万颖瑜　陈国良

(中国科学技术大学计算机系　国家高性能计算中心　合肥230027)

Abstract　Enumerating all minimal separators of a graph is important in reliability analysis for networks and other areas. Given an undirected connected simple graph $G=(V,E)$ and two non-adjacent vertices a and b, this paper presents two efficient algorithms for enumerating all minimal (a,b)separators. All these two improve the known best results. Moreover, for two non-adjacent vertex sets A and B of G, we propose an algorithm to enumerating all minimal(A,B) separators.

Keywords　Graph theory, Minimal separator, Enumeration algorithm

设$G=(V,E)$是无向连通单图，S为G的一个顶子集，$G[S]$为S的导出子图。若$G[V-S]$不连通，则称S为G的一个顶剖；若S是G的顶剖，而S的任意真子集都不是G的顶剖，则称S为G的一个极小顶剖。a,b为G中任意两个不相邻的顶，若a,b分别处在$G[V-S]$的不同连通片中，则称S是G的一个(a,b)顶剖；若S是G的(a,b)顶剖，而S的任意真子集都不是G的(a,b)顶剖，则称S为G的一个极小的(a,b)顶剖。

枚举图中所有极小(a,b)顶剖和所有极小顶剖是图论中的一个基本枚举问题，这个问题在网络的可靠性分析和运筹学等方面有着极大的应用价值[1~3]，已经有很多研究人员对此进行了研究[2~5]。在[5]中Kloks等提出了一个$O(n^3R_{ab})$时间的枚举所有极小(a,b)顶剖的算法，和一个$O(n^5R)$时间的枚举所有极小顶剖的算法，其中$n=|V|$，R_{ab}为所有极小(a,b)顶剖的数目，R为所有极小顶剖的数目。这是目前已知的最好结果。

本文在Kloks算法的基础上采用一种有效的数据结构保存枚举过程的结果，在这种结构上查询一个极小顶剖的时间只需$O(n)$，由此得到了一个$O(nmR_{ab})$时间的枚举所有极小(a,b)顶剖的算法，其中$m=|E|$。这个算法改进了Kloks的结果。采用这种数据结构的另一个好处就是大大地减少了所需的空间。利用这个算法作为子过程，我们提出了一个枚举所有极小顶剖的算法，这个算法的时间复杂度为$O(nmR_{\Sigma})$，其中$R_{\Sigma}=\sum_{1\leqslant v_i\neq v_j\leqslant n,(v_i,v_j)\in E}R_{v_iv_j}$，由于$R\leqslant R_{\Sigma}\leqslant n^2R$，所以这个算法也改进了Kloks的结果。对于$G$中两个不相邻的顶子集$A$和$B$，若要枚举$G$的所有极小$(A,B)$顶剖，我们首先将$A$和$B$退化成两个顶$a$和$b$，然后在新得到的图上枚举所有极小$(a,b)$顶剖，这些极小$(a,b)$顶剖就是要求的$G$的所有极小$(A,B)$顶剖，从而得到了一个时间复杂度为$O(m(n-n_A-n_B)R_{AB})$的枚举所有极小$(A,B)$顶剖的算法，其中$n_A=|A|$，$n_B=|B|$，$R_{AB}$为所有极小$(A,B)$顶剖的数目。

1. 预备知识

本节给出一些相关定义和Kloks算法的思想。

1.1　相关定义

本小节中的定义均沿用文[5]中的定义。

设$G=(V,E)$是一无向连通单图，$X\subset V$是G的一个顶子集，我们使用$G[X]$表示X的导出子图。

定义1　$S\subset V$，若$G[V-S]$不连通，则称S为G的一个顶剖；若S是G的顶剖，而S的任意真子集都不是G的顶剖，则称S为G的一个极小顶剖。

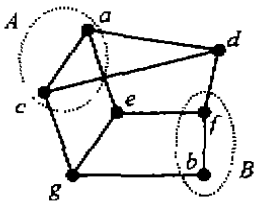
定义2　$S\subset V$，顶$a,b\in V-S$且a,b不相邻，若a,b分别在$G[V-S]$的不同连通片中，则称S为G的一个(a,b)顶剖；若S是G的(a,b)顶剖，而S的任意真子集都不是G的(a,b)顶剖，则称S为G的一个极小(a,b)顶剖；若S是G的极小(a,b)顶剖，且S中只含a的邻顶，则称S为G中a的极小(a,b)顶剖，并记作S_a。

定义3　$A,B\subset V$，且$A\cap B=\varnothing$，A中顶不与B中顶相邻，$S\subset V-(A\cup B)$，若在$G[V-S]$中A的顶与B的顶不连通，则称S为G的一个(A,B)顶剖；若S是G

*)本文得到国家教育部博士点基金资助。

的(A,B)顶剖，而S的任意真子集都不是G的(A,B)顶剖，则称S为G的一个极小(A,B)顶剖。

图1显示了一个图的所有极小(a,b)顶剖、极小(A,B)顶剖和极小顶剖，其中图中a的极小(a,b)顶剖为$S_a=\{c,e,d\}$。



$A=\{a,c\}$, $B=\{f,d\}$

极小(a,b)顶剖

$\{c,e,d\}$, $\{g,e,d\}$, $\{g,f\}$, $\{c,e,f\}$

极小(A,B)顶剖

$\{g,e,d\}$

极小顶剖

$\{c,e,d\}$, $\{g,e,d\}$, $\{g,f\}$, $\{c,e,f\}$, $\{e,b,d\}$, $\{a,d,g\}$, $\{a,f,g\}$, $\{a,c,f\}$, $\{c,e,b\}$

图1

在下面的叙述中如无特殊说明，我们分别用R、R_{ab}和R_{AB}表示图的所有极小顶剖、所有极小(a,b)顶剖和所有极小(A,B)顶剖的数目。

1.2 Kloks算法思想

给定一无向连通单图$G=(V,E)$和G中不相邻的两个顶a和b，下面的引理给出了确定一个极小(a,b)顶剖的充要条件。

引理1[5] 设S是G的一个(a,b)顶剖，则S是G的极小(a,b)顶剖当且仅当S的每个顶在$G[V-S]$的两个连通片C_a和C_b中均有邻顶，其中C_a和C_b分别为包含顶a和b的那个连通片。

若S是G的一个极小(a,b)顶剖，顶$x\in S$，且x不与a相邻，下面的引理给出了由S和x求下一个极小(a,b)顶剖的方法。

引理2[5] 设C_a是$G[V-S]$中包含a的连通片，Δ是C_a中x的极小(x,a)顶剖，C'_a是$G[C_a-\Delta]$中包含顶a的连通片，N是S中与C'_a不相邻的顶子集，则$S'=(S\cup\Delta)\setminus N$是$G$的一个极小$(a,b)$顶剖，且$S'\neq S$。

Kloks算法的思想如下：

1)求出G中b的极小(a,b)顶剖S_b；然后执行以下循环，其中第2步为第1次循环，第3步为第k次循环($k=2,3,\cdots$)。

2)对S_b中每个不与a顶相邻的顶，使用引理2求出一个新的极小(a,b)顶剖。

3)在第k次循环中，若第$k-1$次循环没有产生新的极小(a,b)顶剖，则算法结束；否则对第$k-1$次循环产生的每个极小(a,b)顶剖，及其中的每个不与a相邻的顶x，使用引理2求出一个新的极小(a,b)顶剖。

在文[5]中，Kloks等证明了以上算法可以枚举G的所有极小(a,b)顶剖，于是有：

引理3[5] 以上算法可以枚举G的所有极小(a,b)顶剖。

2. 数据结构和算法

本节介绍我们使用的一种数据结构和基于此得到的两个枚举算法。

2.1 数据结构

在Kloks算法的执行过程中可能会出现以下情况：在第k次循环中新产生一个极小(a,b)顶剖S，而实际上S在以前的某次循环中产生过，设为第l次循环($l<k$)，由算法知在第$l+1$次循环中已经对S及S中不与a相邻的顶考虑过，所以在第$k+1$次循环中不应该将S作为新的极小(a,b)顶剖予以考虑。这就要求在算法中保存所有已找到的极小(a,b)顶剖，然后每求出一个新的极小(a,b)顶剖，都要判断该顶剖是否以前被找到过，以防止重复的操作。Kloks算法对这种情况也进行了判断，但是由于存储结构不好，所以每次判断所花的时间都是$O(n^2)$。

在此我们引入一种树形结构来保存所有已找到的极小(a,b)顶剖，在这种结构上进行判断所花的时间只有$O(n)$。首先对G的顶集V的所有顶编号，设$n=|V|$，则顶的编号分别为$1,2,\cdots,n$。这种树形结构为二叉树，树中所有叶子的深度都是$n+1$，每个叶子对应一个极小(a,b)顶剖，树根到该叶子的路径唯一地确定了这个极小(a,b)顶剖，确定规则如下：在这条路径上若深度为i的结点是深度为$i-1$的结点的右儿子，则说明该极小(a,b)顶剖中含有G中编号为$i-1$的顶；否则不含这个顶。对图1中的图，设顶a,b,c,d,e,f,g的编号分别为1,2,3,4,5,6,7，则图1中的所有极小(a,b)顶剖对应的二叉树如图2所示。

在这种结构上查询一个顶剖S是否存在，只需从树根开始按上面的规则找S对应的叶子，若叶子存在，则表明S在该二叉树中存在，否则不存在。例如：若$S=\{f,g\}$，在图2中的树中从根开始可以找到对应的叶子，说明S存在树中；若$S=\{b,c,e\}$，按照上面的规则在图2的树中查找S，当找到深度为2的结点时，找不到深度为3的右儿子，说明S对应的叶子在树中不存在，也就是S在树中不存在。类似的方法可以将S插入该二叉树中。由于树高为$n+1$，所以这两个操作

的时间均为$O(n)$，于是有：

引理4 在以上二叉树中查询一个顶剖S是否存在和插入S的操作均只需$O(n)$时间。

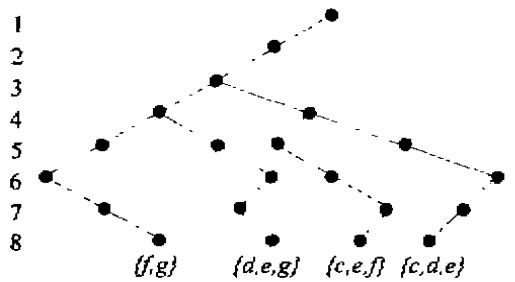


图2

在Kloks算法中每一个顶剖都是分开存放的，若有x个顶剖，则需xn的存储空间来存放这些顶剖，而使用上面的二叉树结构顶剖之间可以共享一部分信息（即树中的一些结点），对于x个顶剖，所需存储空间少于xn，这说明使用这种二叉树还可以减少算法所需的空间。

2.2 算法

基于以上的数据结构和Kloks的思想，我们得到一个枚举G的所有极小(a,b)顶剖的算法，如图3所示。

```
Procedure separator(G,a,b,T)
输入：图G和G中两个不相邻的顶a,b
输出：二叉树T,包含G的所有极小(a,b)顶剖
Begin
   1  计算G中b的极小(a,b)顶剖S_b
   2  L:={S_b},L':=∅
   3  将S_b插入T
   4  while L≠∅ do
        for L中每个极小(a,b)顶剖S do
            4.1  计算G[V-S]中含顶a的连通片C_a
            4.2  for S中每个不与a相邻的顶x do
                 4.2.1  计算C_a中x的极小(x,a)顶剖Δ
                 4.2.2  计算G[C_a-Δ]中含a的连通片
                        C'_a
                 4.2.3  计算S中与C'_a不相邻的顶子集
                        N
                 4.2.4  S*:=(S∪Δ)\N
                 4.2.5  if S*∉T then L':=L'∪{S*},
                        并将S*插入T
                 end for
        end for
        L:=L'
      end while
End
```

图3 枚举G的所有极小(a,b)

算法开始时，$T=\varnothing$，由引理3可知以上算法可以枚举G的所有极小(a,b)顶剖。

算法第1步计算S_b，由引理1知S_b由顶b的邻顶去掉与C_a不相邻的顶而得到，所花时间和计算C_a的时间相同，为$O(n+m)=O(m)$。同理第4.2.1步的时间也为$O(m)$。第2步的时间为$O(n)$。第3步和第4.2.5步的时间由引理4知为$O(n)$，第4.1步和第4.2.2步求连通片的时间为$O(m)$。第4.2.3步求N的时间为$O(m)$，第4.2.4步为顶集上的操作，时间为$O(n)$。算法中$L:=L'$可以在常数时间内实现（如通过指针赋值），算法最内层的*for*循环对每个S最多执行$|S|$次，又因为$|S|\leqslant n-2$，所以最多执行$O(n)$次，由于对每个极小(a,b)顶剖只考虑一次，所以算法最外层的两个循环（*while*和*for*循环）最多执行R_m次。于是得到以下定理：

定理1 对无向连通单图$G=(V,E)$中的两个不相邻的顶a和b，可在$O(nmR_m)$时间内枚举出所有极小(a,b)顶剖，其中$n=|V|$，$m=|E|$，R_m为所有极小(a,b)顶剖的数目。

若要枚举G的所有极小顶剖，可以对G的每个不相邻顶对(a,b)使用上面的算法求出所有的极小(a,b)顶剖，然后将这些顶剖合并到一起，即得到G的所有极小顶剖。算法如图4所示。

```
Procedure all-separator(G,T)
输入：图G
输出：二叉树T,包含G的所有极小顶剖
Begin
    for i:=1 to n-1 do
        for j:=i+1 to n do
            if(v_i,v_j)∉E then
                separator(G,v_i,v_j,T)
End
```

图4 枚举G的所有顶剖

算法开始时$T=\varnothing$，结束后T中包含了G的所有极小顶剖。在整个算法过程中每次对算法separator的调用都使用同一棵二叉树T，保存求得的极小顶剖，这样可能会导致以下情况的出现：设a,b和c,d是G中两个不相邻的顶对，某个顶剖S既是极小(a,b)顶剖，又是极小(c,d)顶剖，在调用separator(G,a,b,T)时，将S插入树T中，然后在调用separator(G,c,d,T)的过程中，第一次产生S后，在树T中查询到S，会认为S以前被考虑过，在第4.2.5步不会将S插入集合L'，从而在下一次执行外层的*for*循环时对S不予考虑，这样有可能导致某些极小(c,d)顶剖被漏掉。所以我们对数据结构和算法稍作修改。首先树T的每个叶子增加一个域以保存该叶子代表的极小顶剖是由哪个顶对求得的。然后在算法Separator中利用这个域来防止上述情况的出现，只需将算法separator的第3步和第4.2.5步相应地修改为：

3 将S_b插入T，并置对应叶子的域为当前顶对(a,b)

4.2.5 *if* S'在T或对应叶子的域所记录的顶对不是当前顶对(a,b)
then $L':=L'\cup\{S'\}$，将S'插入T并置相应的域为当前顶对(a,b)

（下转第35页）

不仅具有一般的多层 Client. Server 分布式应用系统的优点，而且 Borland 公司针对多层分布式应用的开发还提供了最完美的支持。例如，通过浏览器和 Web 服务器也能在客户端没有任何数据库工具的情况下观察远程机器上的数据集；通过分布式数据集可大大缩减网络通信量；提供访问数据库约束条件，当从服务器上卸载数据时，可以同时卸载一套自动执行的约束条件；Borland 多层计算的一个重要功能是将数据库的负载分散到多个服务器上；Remote DataBroker，其结构的精髓是让每一个客户端不再需要 BDE，取而代之的是一个中央化的 BDE，以集中管理的方式降低每一个客户在 BDE 上所需的开销和复杂度；Constraint Broker，它所扮演的角色是保证所有客户数据的一致性及数据的完整性；BusinessObjectBroker，其目的是给一些关键性的商业应用程序提供一个快速且可信赖的使用环境，为了满足这种高层次的要求，Business Object-Broker 会自动地将应用程序做适当的划分，并复制重要的业务规则到每一个区间，以达到速度的要求。

参 考 文 献

1 Microsoft DCOM Technical Overview. Available at: http://www.microsoft.com

2 Building Scalable Application with Windows NT and COM. Available at: http://www.borland.com/midas/dcom

3 徐新华. Delphi 4核心编程技术. 希望电脑公司出品，1998.11

4 Developer's Guide-Borland Delphi4 for Windows95 and WindowsNT. INPRISE Coporation, 100 Enterprise Way Scotts, Valley, CA 95066-3249

5 Borland 的 MIDAS 技术. Available at: http://bbs.tsinghua.edu.cn

6 徐新华，Delphi 3编程指南. 宇航出版社，1998.6

7 鲍泓，佟建新，尉林明，等. Windows NT 组网技术. 电子工业出版社，1998.4

（上接第96页）

经过上述修改后的算法能够保证不同顶对之间互不干扰，从而避免了上述情况的发生。

上述算法的时间复杂度为 $\sum_{(a,b)\in E}O(nmR_{ab})=O(nmR_{\sum})$，由此得到以下定理：

定理2 给定无向连通单图 $G=(V,E)$，可在 $O(nmR_{\sum})$ 时间内枚举 G 的所有极小顶割，其中 $n=|V|$，$m=|E|$，$R_{\sum}=\sum_{1\le i\le n,(v_i,v_j)\in E}R_{v_iv_j}$。

对于连通单图 G，有 $m<n^2$，当 G 为稀疏图时，$m=O(n)$，此时 $O(nmR_{ab})=O(n^2R_{ab})$。显然我们的第一个算法改进了 *Kloks* 的结果 $O(n^3R_{ab})$。又因为 $R\le R_{\sum}\le n^2R$，$O(nmR)\le O(nmR_{\sum})\le O(n^2mR)$，所以我们的第二个算法也改进了 *Kloks* 的结果 $O(n^5R)$。

3. 推广

给定 G 中两个不相邻的顶子集 A，B，若要求所有极小 (A,B) 顶割，可以先将顶子集 A 和 B 退化成两个顶 a 和 b，对于 $V-A-B$ 中的顶，若与 A（或 B）中某顶相邻，则在 a（或 b）和该顶之间连一条边，从而将图 G 变成图 $G'=(V',E')$，其中：

$$V'=(V-A-B)\cup\{a,b\}$$

$$E'=E(V-A-B)\cup\{(a,v)\mid v\in V-A-B,\text{且}\exists\ u\in A,\text{使}(u,v)\in E\}\cup\{(u,b)\mid u\in V-A-B,\text{且}\exists\ v\in B,\text{使}(u,v)\in E\}$$

然后在 G' 上使用上一节的算法 separator，求出 G' 中的所有极小 (a,b) 顶割，而这些极小 (a,b) 顶割就是 G 中的所有极小 (A,B) 顶割，所以通过上面的过程可以求得 G 中的所有极小 (A,B) 顶割。

现分析上述过程的时间复杂度如下：将 G 变成 G' 的过程中，计算 V' 所花时间为 $O(n)$，计算 E' 所花时间为 $O(m)$；由于 $|V'|=n-n_A-n_B+2$，$|E'|\le m$，所以在 G' 上求所有极小 (a,b) 顶割的时间为 $O(m(n-n_A-n_B)R_{AB})$。故总时间为 $O(m(n-n_A-n_B)R_{AB})$，于是得到以下定理：

定理3 给定无向连通单图 $G=(V,E)$，$A\subset V$，$B\subset V$，且 $A\cap B=\varnothing$，A 与 B 不相邻，则可在 $O(m(n-n_A-n_B)R_{AB})$ 的时间内枚举 G 的所有极小 (A,B) 顶割，其中 $n=|V|$，$m=|E|$，$n_A=|A|$，$n_B=|B|$，R_{AB} 为所有极小 (A,B) 顶割的数目。

最后，一个极富挑战性的问题是能否在 $O(n^2R_{ab})$ 时间内枚举所有极小 (a,b) 顶割，这也是我们下一步的工作。

参 考 文 献

1 Ariyoshi H. Cut-set graph and systematic generation of separating sets. IEEE Trans. Circuit Theory, 1972, CT-19: 233～240

2 Goldberg L. A. Efficient algorithms for listing combinatorial structures. Cambridge Univ. Press, Cambridge, 1993

3 Kanevsky A. On the number of minimum size separating vertex sets in a graph and how to find all of them. In: Proc. 1st Ann. ACM-SIAM Symp. Discrete Algorithms, 1990. 411～421

4 Arnberg S. Efficient algorithm for combinatorial problems on graphs with bounded decomposability a survey. BIT, 1985, 25: 2～23

5 Kloks T, Kratsch D. Listing all minimal separators of a graph. SIAM J. Comput., 1998, 27(3): 605～613