第4卷第4期
1993年8月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 4, No. 4
Aug. 1993

一个求图的连通分支的并行算法

唐策善　　梁维发

（中国科技大学计算机系，合肥 230027）

A PARALLEL ALGORITHM FOR COMPUTING CONNECTED COMPONENTS OF GRAPHS

Tang Ceshan and Liang Weifa

(*Department of Computer Science, University of Science and Technology of China, Hefei* 230027)

Abstract　Given an undirected graph G(V,E), |V|=n, |E|=m. Based on SIMD shared memory model, a new parallel algorithm for computing connected components of graphs is proposed by using fast data transmission principle. As a result, the algorithm requires $O(\log^2 n)$ time and $O(n^2/\log n)$ processors on SIMD−CREW shared memory model. But on SIMD−CRCW shared memory model, the algorithm only requires $O(\log n)$ time and $O(n^2)$ processors. To compare our algorithm with known Hirschberg's algorithm, there exists some differences. The major differences are identified as: 1) the way to solve this problem is different; 2) proposed algorithm is simple and easy to understand; 3) proposed algorithm's implementation on some practical networks such as mesh−of−tree has better time complexity.

摘要　已知一个无向图 G(V,E), |V|=n, |E|=m. 本文基于 SIMD 共享存贮模型，运用数据在图中快速传播原理，建议了一个新的求图的连通分支算法. 具体来讲，在 SIMD−CREW 共享存贮模型上，求图的连通分支需 $O(\log^2 n)$ 时间、$O(n^2/\log n)$ 处理器；而在 SIMD−CRCW 共享存贮模型上需 $O(\log n)$ 时间、$O(n^2)$ 处理器. 建议的算法同著名的 Hirschberg 算法相比，其主要差别表现在：1) 采用的求解方法不同；2) 建议的算法简单易懂；3) 在某些实际网络如树网上，实现建议的算法有较好的时间复杂性.

§1. 引　言

近年来，图论问题的并行算法已日益受到人们的重视，尤其对无向图的连通分支这类图论基本问题的并行化，引起了人们极大的兴趣. 在这方面已做了大量的工作. 基于

本文 1990 年 7 月 20 日收到，1991 年 5 月 10 定稿. 此课题受国家 863 计划资金资助. 作者**唐策善**，教授，主要研究领域为并行处理，算法设计与分析，智能搜索算法，决策支持系统软件等. **梁维发**，博士生，1989 年硕士毕业于中国科技大学，主要研究领域为并行和分布式处理，图论算法，神经网求解组合优化问题等.

SIMD—CREW 共享存贮模型，Hirschberg 等人曾给出了 $O(\log^2 n)$* 时间、$O(n^2/\log n)$处理器的算法，即著名的 Hirschberg 算法[1]. 其后许多学者对这个算法进行了改进或移植，Chin 等人基于 SIMD—CREW 共享模型及 Vishkin 基于 SIMD—CRCW 共享存贮模型分别给出了 $O(\log^2 n)$时间、$O(n^2/\log^2 n)$处理器算法[2,3]；Nassimi 等人基于网孔模型给出了 $O(d \cdot n\log n)$时间、$O(n^2)$处理器的算法(d 为图的直径)[4]；Yeh 等人基于完全二叉树机给出了一个 $O(n^2/p)$时间、$O(p)$个处理器的算法[5]，$(1 \leqslant p \leqslant n)$；Ullman 在树网上给出了一个 $O(\log^3 n)$时间、$O(n^2)$处理器的算法[6]；Miller 等人在金字塔模型上给出了 $O(\sqrt{n})$时间、$O(n^2)$处理器的算法[7]；Awerbuch 等人在混洗网络上给出了 $O(\log^2 n)$时间、$O(n^2)$处理器的算法[8]；文[10]在超立方模型上给出了 $O(n^2/p)$时间、$O(p)$个处理器算法 $(1 \leqslant p \& p\log p \leqslant n)$. 上述算法的基本思想都是基于 Hirschberg 的顶点倒塌法，即将图顶点归约为“超顶点”，再将超顶点归约为新的超顶点，直到每个连通分支成为一个超顶点时算法终止. 本文建议的算法是从数据传播角度出发，利用并行处理的路径折叠技术，使数据在图中快速传播，从而快速求得图的连通分支. 具体地讲，因为同一连通分支内的顶点具有相同标识，所以我们把每个连通分支内最小标号顶点作为该连通分支标识. 本文研究怎样快速地把每个连通分支最小标号顶点(即所在连通分支标识)传播到该连通分支的每个顶点上，最后同一连通分支的顶点将具有相同标识. 本文的算法同 Hirschberg 算法相比，具有几个明显的不同特点：①采用不同的求解技术；②算法简单易懂；③在某些实际网络上实现时具有较好的时间复杂性.

在 SIMD—CREW 共享存贮模型及 SIMD—CRCW 共享存贮模型上，本文算法分别需$O(\log^2 n)$时间、$O(n^2/\log n)$处理器及 $O(\log n)$时间、$O(n^2)$处理器.

以下各节安排如下：在第二节我们将对计算模型作一简单介绍，然后介绍新的算法，同时讨论算法在上述两模型上的实现及正确性证明、复杂性分析. 最后给出结论.

§2. 共享存贮模型上的连通分支算法

2.1 并行计算模型

本文采用一种常见的并行计算模型——共享存贮模型. 此模型假定存在一个容量无限大的共享存贮器，所有处理器之间的通讯都是通过共享存贮变量实现的. 由于所有处理器对存贮器的读、写约束不同，形成了以下几种不同的计算模型：EREW——每次仅允许一个处理器读或写一个存贮单元；CREW——每次允许多个处理器同时读一共享存贮单元内容，但仅允许一个处理器写共享存贮单元；CRCW——允许任意个处理器同时读或同时写一个共享存贮单元. 对于同时写“同一单元”引起的写“冲突”，Kucera 曾建议了四条解决方法[9]：(1)同时写“1”成功；(2)同时写相同值成功；(3)优先权最高的处理器写成功. 本文的“写冲突”，采用优先权法. 这里处理器优先权定义为：处理器编号越低，优先权则越高. 假定处理器编号用二维数组表示，且以行主方式对所有处理器编号.

2.2 连通分支算法

对于任意一个顶点 $i \in V$，令 D(i)表示顶点 i 所在连通分支标识，即 i 所在连通分支的

* 本文"log"均以 2 为底

最小标号顶点是D(i).$D^{(t)}(i)$表示在第t次迭代时,顶点i所在的连通分支标识(t=0时,约定$D^{(0)}(i)=i$).无向图按邻接矩阵A输入.算法分为四个基本步.其中第一步:初始化工作;第二步:找每个与顶点$i\in V$有边关联的另一端点所在连通分支标识,将最小标号顶点标识顶点i;第三步:若顶点i目前的连通分支标识已更新,即$D^{(t)}(i)<D^{(t-1)}(i)$,则对以前所有由$D^{(t-1)}(i)$标识的顶点$j(D^{(t-1)}(j)=D^{(t-1)}(i))$更新后标识为$D^{(t)}(j)$,将最小的$D^{(t)}(k)$(满足$D^{(t-1)}(k)=D^{(t-1)}(i)$)送入$D^{(t)}(i)$中,同时也送入所有$j(D^{(t-1)}(j)=D^{(t-1)}(k))$中;第四步:路径折叠,加速数据传播.算法形式化描述如下:

```
Procedure Connected_Components;
(1)  ∀ i∈V, D^(0)(i)←i;   A(i,i)←1; t←1;
(2)  ∀ i∈V, D^(t)(i)←min{D^(t-1)(j) | A(i,j)=1, ∀ j∈V};   B(i)←n+1;
(3)  ∀ i∈V, if D^(t)(i)≠D^(t-1)(i)
                then B(D^(t-1)(i))←D^(t)(i), 且最小 D^(t)(i)值写成功;
                     if B(D^(t-1)(i))<D^(t)(i)
                       then D^(t)(i)←B(D^(t-1)(i))
                     endif
             endif;
(4)  ∀ i∈V, D(i)←D^(t)(D^(t)(i));   D^(t)(i)←D(i);
(5)  ∃ j∈V, if D^(t)(j)≠D^(t-1)(j)  then t←t+1;   goto (2)
                                    else STOP
             endif
end;
```

定理1:上述算法能正确地求出无向图G(V,E)的连通分支.

证明:由算法的第(3)、(4)步可知,在最小标号S_1的传播路径上(相对这段路径来讲,S_1标号最小),标号为S_1的顶点u传播标号S_1到另一标识为S_2的顶点v且$S_1<S_2$,则将S_1标识顶点v,同时S_1标识所有S_2标识的顶点;若顶点v存在多条这样的路径,即有多个满足$S_1<S_2$的标号S_1存在,此时v将由最小的S_1标识,同时所有由S_2标识的顶点将由S_1标识.又在一连通分支中,两顶点间最长距离为cn(0<c<1),用路径折叠法将一连通分支内最小标号顶点传播到此连通分支所有其它顶点,至多需$\lceil\log n\rceil$时间,也即第(5)步至多执行$\lceil\log n\rceil$次.现在来证明,若$\forall i\in V$,均有$D^{(t)}(i)\equiv D^{(t+1)}(i)$,则$D^{(t+1)}(i)$就是所求i的连通分支标识D(i).我们采用反证法.假定$\forall i\in V$满足$D^{(t)}(i)\equiv D^{(t+1)}(i)$,设一个顶点$v\in V$有$D^{(t+1)}(v)=k$,但v所在连通分支的最小标号顶点为j,如图1所示.若在第t次迭代时,j已标识了顶点u到j这段路径上所有顶点,由假定知$D^{(t)}(u)=k$.又$j<k$,由第(3)步知$D^{(t+1)}(u)=j$,执行第(4)步后,所有标记为k的顶点都将标识为j,即$D^{(t+1)}(v)=j$,与条件$D^{(t)}(v)=D^{(t+1)}(v)$矛盾,故算法能正确地将每个连通分支的最小标号顶点传播到连通分支的所有其它顶点上.值得指出的是:我们采用第(5)步而不用$\lceil\log n\rceil$次迭代.理由如下:若图的直径为d,则第(5)步只需$\lceil\log d\rceil$次迭代即可,这比用$\lceil\log n\rceil$次迭代节省了$\lceil\log n\rceil-\lceil\log d\rceil$次迭代.下面给出一个具体例子来验证算法的正确性,如图2所示.

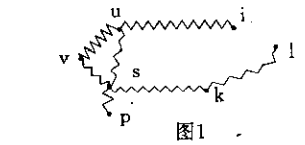
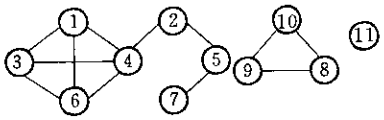


图1



V	1	2	3	4	5	6	7	8	9	10	11
$D^{(0)}(\cdot)$	1	2	3	4	5	6	7	8	9	10	11
$D^{(1)}(\cdot)$	1	2	1	1	2	1	2	8	8	8	11
$D^{(2)}(\cdot)$	1	1	1	1	1	1	1	8	8	8	11
$D^{(3)}(\cdot)$	1	1	1	1	1	1	1	8	8	8	11

(注:$D^{(i)}(\cdot)$表示第i次迭代时的连通分支标识,·代表V中任一顶点)

图2 算法的执行过程

现在讨论在SIMD—CRCW共享存贮模型上如何实现上述算法.约定PE(i,j)表示编号为i*n+j—1处理器.注意在算法的第(2)、(3)步存在写"冲突".首先第(2)步写冲突解决如下:

(2.1) ∀ i,j: PE(i,j)置 $C(i,j)\leftarrow n+1$;

(2.2) ∀ i,j;**if** $A(i,j)=1$ **then**

PE(i,j)向 $C(i,D^{(t-1)}(j))$写"1",仅最小编号PE写成功

endif;

(2.3) ∀ i,j;**if** $C(i,j)=1$ **then**

PE(i,j)向 $D^{(t)}(i)$单元写内容j,仅当最小编号PE写成功

endif;

第(3)步写"冲突"解决如下:

(3.1) ∀ i, j: PE(i,j)置 $C(i,j)\leftarrow n+1$; PE(i,1)置 $B(i)\leftarrow n+1$;

(3.2) ∀ i;**if** $D^{(t)}(i)\neq D^{(t-1)}(i)$ **then**

PE(i,1)向 $C(D^{(t)}(i), D^{(t-1)}(i))$写"1",仅最小编号PE写成功

endif;

(3.3) ∀ i,j: **if** $C(i,j)=1$ **then**

PE(i,j)向 $B(D^{(t-1)}(j))$写"1",仅最小编号PE写成功

endif;

(3.4) ∀ i: **if** $B(D^{(t-1)}(i))<D^{(t)}(i)$ **then**

$D^{(t)}(i)\leftarrow B(D^{(t-1)}(i))$

endif;

最后,第(5)步"写冲突"解决如下:

(5.1) **if** $D^{(t)}(i)\neq D^{(t-1)}(i)$ # **then**

PE(i,1)写"1"到变量FLAG中,仅最小编号PE写成功

endif;

(5.2) PE(i,1)判别FLAG=1?

if FLAG=1 **then** FLAG←0; **goto**(2) **else** STOP **endif**;

定理2:在SIMD—CRCW共享存贮模型上,求一无向图G(V,E),|V|=n的连通分支需$O(\log n)$时间、$O(n^2)$处理器.

证明:算法第(1)步需O(1)时间、O(n)处理器;第(2)步需O(1)时间、$O(n^2)$处理器;

第(4)、(5)步需 O(1)时间、O(n)处理器，而迭代至多为 $\lceil \log n \rceil$ 次，故整个算法需 O(logn)时间、$O(n^2)$处理器.

下面叙述算法在 SIMD—CREW 共享存贮模型上的实现. 也就是在不允许有"写冲突"的前提下，如何解决算法(2)、(3)、(5)步的写冲突. 具体解决办法是：

第(2)步实现如下：

(2.1) $\forall$ i,j: **if** A(i,j)=1 **then** E(i,j)←$D^{(t-1)}$(j) **else** E(i,j)←n+1 **endif**;

(2.2) 对矩阵 E 的每行用 O(n)处理器合作求最小值 $C_{min}(i)=\mathbf{min}\{E(i,j)\mid \forall\ j\in V\}$;
if $C_{min}(i)\neq n+1$ **then** $D^{(t)}(i)\leftarrow C_{min}(i)$ **endif**;

第(3)步实现如下：

(3.1) $\forall$ i,j: PE(i,j)置 C(i,j)←n+1; PE(i,1)置 B(i)←n+1;

(3.2) $\forall$ i: **if** $D^{(t)}(i)\neq D^{(t-1)}(i)$ **then** $C(i, D^{(t-1)}(i))\leftarrow D^{(t)}(i)$ **endif**;

(3.3) 对矩阵 C 的每列 j 用 O(n)处理器合作求最小值，$C_{min}(j)=\mathbf{min}\{C(j,j_1)\mid j_1\in V\}$;
if $C_{min}(j)\neq n+1$ **then** $B(j)\leftarrow C_{min}(j)$ **endif**;

(3.4) **if** $B(D^{(t-1)}(i))<D^{(t)}(i)$ **then** $D^{(t)}(i)\leftarrow B(D^{(t-1)}(i))$ **endif**;

第(5)步实现如下：

(5.1) $\forall\ i\in V$: PE(i,1) 做：
if $D^{(t)}(i)\neq D^{(t-1)}(i)$ **then** B(i)←1 **else** B(i)←0 **endif**;

(5.2) **if** $\sum_{i=1}^{n} B(i)\neq 0$ **then** t←t+1; **goto** (2)
else STOP
endif;

因为在 SIMD—CREW 共享存贮模型上，算法的每步实现至多需 O(logn)时间、$O(n^2)$处理器. 若采用分组技术，即每个处理器不是赋予一个元素而是 $\lceil \log n \rceil$ 个元素，则求 n 个元素最小值需O(logn)时间、O(n/logn)处理器，这样算法的第(2)、(3)步只需 O(logn)时间、$O(n^2/\log n)$处理器就可完成. 又算法第(5)步至多执行 $\lceil \log n \rceil$，故得：

定理3：在 SIMD—CREW 共享存贮模型上，求一无向图 $G(V,E)$，$|V|=n$ 的连通分支需$O(\log^2 n)$时间、$O(n^2/\log n)$处理器.

结　论：本文从数据在图中快速传播角度出发，建议了一个新的求图的连通分支算法. 在 SIMD—CREW 共享存贮模型上，此算法需 $O(\log^2 n)$时间、$O(n^2/\log n)$处理器，而在 SIMD—CRCW 共享存贮模型上，此算法只需 O(logn)时间、$O(n^2)$处理器. 同著名的 Hirschberg 算法相比，除了采用的方法不同以及本算法更简单易懂之外，在某些实际网络上实现起来较 Hirschberg 算法更为优越，其原因在于：该算法每次求超顶点需 $\lceil \log n \rceil$ 次迭代，而这种迭代在整个算法实现过程中占有支配地位，耗费时间，而本文算法则较好地克服了这一点.

参考文献

1 D. S. Hirschberg, A. K. Chanda and D. V. Sarwate, Computing Connected Components on Parallel Computers, CACM, Vol. 22, 1979, 461—464.

2 F. Y. Chin, J. Lam and I. N. Chen, Efficient Parallel Algorithms for Some Graph Problems, CACM, Vol. 25, 1982, 659—665.

3 U. Vishkin, An Optimal Parallel Connectivity Algorithm, Distrete Appl. Math., Vol. 9, 1984, 197—207.

4 D. Nassimi and S. Sahni, Finding Connected Components and Connected Ones on a Mesh—Connected Parallel Computer, SIAM J. Comput., Vol. 9, 1980, 744—757.

5 D. Y. Yeh and D. T. Lee, Graph Algorithms on a Tree Structured Parallel Computer, BIT, Vol. 24, 1984, 333—340.

6 J. D. Ullman, Computational Aspects of VLSI, Computer Sci. Press, 1984.

7 R. Miller and Q. F. Stout, Data Movement Techniques for the Pyramid Computer, SIAM J. Comput., Vol. 16, 1987, 38—60.

8 B. Awerbuch and Y. Shiloach, New Connectivity and MSF Algorithms for Shuffle—— Exchange Networks and PRAM, IEEE Trans. Comput., Vol. C—36, 1987, 1258—1263.

9 L. Kucera, Parallel Computation and Conflicts in Memory Access, Inform. Proc. Lett., Vol. 14, 1982, 93—96.

10 梁维发、陈国良，Hypercube 多处理器上图的最优算法，计算机学报，第 14 卷第 9 期，1991 年 9 月，641—650.

11 唐策善、梁维发，并行图论算法，合肥，中国科技大学出版社，1991.