第 6 卷第 1 期
1995 年 1 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 1
Jan. 1995

高效异步并行图算法及 ADA 的实现*

马 军 马绍汉

(山东大学计算机系,济南 250100)

摘要 本文给出了计算图的所有顶点间的距离矩阵 D 及最短路径矩阵 P 的一串行和异步并行算法.利用上述结果,又得到其他图论问题的高效异步并行算法,并介绍了用 ADA 语言对异步并行算法实现的主要步骤.

关键词 并行图算法,异步算法.

在共享存储器的多处理机系统(MIMD_SM)上设计的并行算法称为异步算法.一个异步算法是由一些并行的进程组成.设计实际高效的异步算法,应让并行的进程个数恰与系统允许的运行最大并行的进程个数相同.假设可确定整数 p,系统分配 p 个进程的时间可忽略不计,并且 p 个进程可并行运行,对共享存储器的读写方式为 CREW,下面对图论中若干基本问题,讨论求解的高效异步算法.因本文用到计算 n 个元素最小元的异步算法.首先给出计算 n 个元素最小元的异步算法.不失一般性,设 n/p 为一整数.

算法 1. 输入:$A[1,n]$;输出:A 中最小元.

main:

1. creates p parallel processes numbered from 1 to p;
2. sends $(1, n/p)$ to process 1;
3. waits until can take the minimum value from process 1;
4. output the minimum value.

process i:

1. waits until receive a pair (x,d);
2. if $2x \leqslant p$ then the sends $(2x,d)$ to process $2x$;
3. if $2x+1 \leqslant p$ then the sends $(2x+1,d)$ to process $2x+1$;
4. selects the minimum $t1$ from $A[(x-1)*d+1,\cdots,x*d]$;
5. if $2x \leqslant p$ then waits until can take $t2$ from process $2x$ else $t2:=\infty$;
6. if $2x+1 \leqslant p$ then waits until can take $t3$ from process $2x+1$ else $t3:=\infty$;
7. waits until father process(or main) takes the $\min\{t1,t2,t3\}$;

引理 1. 在允许有 p 个进程可并行运行的 MIMD_SM 系统上,算法 1 可在 $O(n/p+\log p)$时间内求出 n 个元素的最小元.

* 本文 1992－06－04 收到,1992－11－30 定稿

本文是山东省自然科学基金资助项目.作者**马军**,1956 年生,讲师,主要研究领域为人工智能,并行算法,算法设计与分析.**马绍汉**,1938 年生,教授,主要研究领域为算法设计与分析,人工智能,并行算法.

本文通讯联系人:马军,济南 250100,山东大学计算机系

证明：为方便理解算法1和对时间复杂性的分析，假定若进程 i 与进程 j 有信息传递，则 i 与 j 有一条边，这样得到的图称为进程图. 显然算法1的进程图为一 p 个顶点的完全二叉树，树根顶点为 process 1. 在算法1中，首先进程 i 被通知在 $A[(i-1)*n/p+1\cdots i*n/p]$ 上求出最小元，$i=1,\cdots,p$. 然后每个子进程再把自己的最小元与从左，右儿子进程得到的最小元中的最小者让父进程取走. 因 p 个顶点的完全二叉树的高为 $O(\log p)$，故通知和返回最小值的时间均为 $O(\log p)$. 加上进程求最小值的时间为 $O(n/p)$，因此算法1的执行时间为 $O(n/p+\log p)$.

把取最小运算换成更一般二元运算有：

定理1. 设 $*$ 是一可结合的，计算时间为常数的二元运算符，在允许有 p 个进程可并行运行的 MIMD_SM 系统上，表达式 $a_1 * a_2 * \cdots * a_n$ 可在 $O(n/p+\log p)$ 时间内被计算完毕.

1 图的所有顶点对之间的最短距离计算

假定图 G 的 n 个顶点被编号为 $0,\cdots,n-1$，我们稍微修改一下 Floyd 计算图的所有顶点对之间的最短距离算法[1]，使其不但可以计算出图的所有顶点对之间的最短距离并且也计算出对应的最短通路，我们用 P 表通路矩阵，其表示方法为：$P(i,j)$ 为从 i 到 j 的最短路上顶点 i 的后继顶点，故 i 到 j 的最短路为 $p=(i,P(i,j),P(P(i,j),j),\cdots,j)$. 修改后的 Floyd 算法如下：

算法2. 输入：$COST$，即图的费用矩阵.

输出：D,P，最短距离和最短通路矩阵.

```
main:
  1. D:=COST; P(i,j):=j; 0≤i,j≤n-1.
  2. for 0≤i,k,j≤n-1 do
       IF D(i,j)>(D(i,k)+D(k,j)) THEN
         {D(i,j):=D(i,k)+D(k,j);P(i,j):=P(i,k) }
```

$P(i,j)$ 的初值为 j，若在步骤2中，从顶点 i 经过 k 到 j 的路径短于目前路径，则通过修改 i 到 j 后继为 i 到 k 的后继的方法修正 $P(i,j)$ 的值. 算法的正确性是显然的，复杂性也没有改变.

我们给出算法2的异步算法如下：

算法3.

```
main:
  1. inputs n,p ; inputs COST;
  2. creates p parallel process i;
  3. sends (1, n,p) to process 1;
  4. waits until all process finish their initialization.
  5. for k = 0 to n - 1 do
       {sends the pair (1, k) to process 1;waits until all processes finish their computition;}
  6. outputs D,P.
  process i ;
     loacl integer veriable t,x,i,j, bound;
  procedure comput1
```

```
    begin for all i,j , such that
          ( i * n + j ) mod p = i − 1 do
          { D[i,j]:=COST[i,j]; P[i,j]:=j;}
    end;
procedure comput2
    begin for all i,j , such that
          ( i * n + j ) mod p= i − 1 do
          if D[i,j]>D[i,k]+D[k,j] then
          {D[i,j] :=D[i,k]+D[k,j];
          P[i,j]:= P[i,k]; }
    end;
−Steps
1. waits until receives an integer x ;
2. if 2x + 1 ≤ p then
     {sends 2x + 1 to process 2x + 1 ;
     sends 2x to process 2x ;
     calls comput1;
     waits until its two subprocess
     finish comput1;}
   else if 2x ≤ p then
       {sends 2x to process 2x;
            calls comput1;
            waits until its left son process
            finish comput1 }
            else calls comput1;
3. informs father process its subprocess
          finished the comput1.
4. for t := 0 to n − 1 do
       {waits until receive an integer k ;
        if 2x + 1 ≤ p then
          {sends 2x + 1 to process 2x + 1 ;
          sends 2x to process 2x ;
          calls comput2;
          waits until its left and right
          son finish comput2;}
         else if 2x ≤ p then
             {sends 2x to process 2x ;
             calls comput2;
             waits until its left son
             process finish comput2}
           else calls comput2;
5. informs father process its
   subprocess finish comput2;};
```

定理2. 在允许有 p 个进程可并行运行的 MIMD_SM 系统上,算法3在 $O(n(n^2/p+\log p))$的时间内正确地计算了 D 和 P.

证明：修改后的Floyd算法仍分两步，第一步为D,P赋初值，第二步循环n次更新D和P.算法思想是让p个进程并行地执行这两步并且每个进程处理约n^2/p个元素，其方法是通过1－1函数：$x=in+j$, $0\leqslant i,j\leqslant n$,变换二维坐标为一维[2]，让进程$i$只处理其下标$i,j$满足$(i*n+j) \bmod p=i-1$的$D$,$P$的元素.在算法3中的第一步，让每个进程接到初始化信号后，调用过程comput1完成对D,P部分元素的初始赋值；第二步每个进程接到一维顶点下标k后，调用comput2完成对D和P部分元素的比较和赋值运算，文献[2]已证明，对D的元素在计算时间上有差异，并不影响对D计算的正确性，这也保证了矩阵P计算的正确性.故算法正确.因算法3的进程图仍为完全二叉树，两进程的通信时间最多为$O(\log p)$，两过程的执行时间均为$O(n^2/p)$，因此算法3的时间复杂性为$O(n(n^2/p+\log p))$. □

因Warshall计算图传递闭包算法[3]和Floyd最短路算法形式一样，同样的方法可得到异步的Washall算法.

推论1. 当$p\ll n$时，在$O(n^3/p)$的时间内，图的最短路径和图的传递闭包的计算可被完成.

2 其他图问题的并行算法

因无向图可以看成有向图的一特例，故算法3对无向图也适用.下面不论对无向图还是有向图我们都用D和P分别表示图的距离和路径矩阵.设D,P已知，我们可以利用上面的结果去解决许多其他的图问题，下面给出3个图问题的并行算法的主要步骤.

a.确定一有向图的中心：

1. $E(j)=\max_{0\leqslant i\leqslant n-1}\{D(i,j)\}$；$0\leqslant j\leqslant n-1$.

2. $E(k)=\min_{0\leqslant j\leqslant n-1}\{E(j)\}$.

顶点k为图G的中心.

b.计算无向图的直径距离和对应路径：

1. $d=\max\ 0\leqslant i,j\leqslant n-1\ \{D(i,j)\}$

2.通过P和编号i,j输出d对应的路径.

c.寻找有向图中具有最短(最长)长度的有向圈：

1. $D(i,i):=(-\infty)$; $0\leqslant i\leqslant n-1$.

2. $D(i',j')=\min(\max)_{0\leqslant i,j\leqslant n-1}\{D(i,j)+D(j,i)\mid D(i,j)\neq\infty(-\infty)$ and $D(j,i)\neq\infty(-\infty),0\leqslant i,j\leqslant n-1\}$;

3.通过P和编号i',j',得G中最短(最长)长度的有向圈为i'到j'的最短(最长)路径和从j'到i'的最短(最长)路径.

如果不考虑输入，输出的时间，根据定理1，及我们在算法3中给出下标处理技术，在p个进程可并行的MIMD_SM系统上，以上三问题的处理时间均不超过计算D和P的时间，故有定理3.

定理3. 在允许有p个进程可并行运行的MIMD_SM系统上，以上三问题均可在$O(n(n^2/p+\log p))$时间内被计算；当$p\ll n$，计算时间为$O(n^3/p)$.

并行程序设计语言ADA多任务并行机制可以实现本文提出的所有异步算法.我们这

里只给出实现的关键步骤.

利用 ADA 可分层定义的特点，在第一层输入 n，p；第二层定义所用矩阵并完成输入；第三层产生 p 个任务，因本文算法的进程图均为完全二叉树，可用一维数组静态定义 p 个并行任务或利用链表动态产生. 主程序执行本文算法中的 main 部分，任务之间的通信利用 ADA 会合功能实现，这种实现方法效率比较高.

参考文献

1 Floyd R W. Algorithm 97. shortest path. CACM, 1962, **5(6)**: 345.

2 Ma Jun, Tadao Takaoka. An $O(n(n^2/p+\log n)$ parallel algorithm to compute the all pair shortest paths and the transitive closure. J. of Information Processing, Japan, 1989, **2(12)**: 119—124.

3 Warshall S. A theorem on Boolean matrices. J. ACM, 1962, **9(1)**: 11—12.

EFFICIENT ASYNCHRONIZED GRAPH ALGORITHMS AND ADA CODING

Ma Jun　Ma Shaohan

(*Department of Computer Science, Shandong University, Jinan* 250100)

Abstract　In this paper, both a sequential and an asynchronised algorithm to compute the all pair shortest distance matrix D and the path matrix P are given. This result is applied to develop other asynchronized graph algorithms. The main steps to code these asynchronised algorithms in ADA are introduced.

Key words　Parallel graph algorithms, asynchronised algorithms.