# AOE 网的并行算法*

唐策善　梁维发
(中国科技大学计算机系)

# PARALLEL ALGORITHMS FOR AOE NETWORKS

Tang Ce-shan Liang We-fa
(*China University of Science and Technology*)

**Abstract**

Parallel algorithms for AOE networks are proposed, which include acyclicity test, topological sorting and computing critical paths. All these algorithms work on a SIMD-CREW-PRAM and require $O(\log^2 n)$ time and $O(n^{\log 7}/\log n)$ processors.

## §1. 引　言

在并行图论算法中,有向图 $G(V, E)$ 的重要应用之一是边表示活动的网(即 AOE 网). 本文研究 AOE 网的并行算法. 假定 AOE 网是一个带权的有向无环图,其中顶点 $i \in V$ 表示事件,有向边 $\langle i,j\rangle \in E$ 表示活动,权 $w(i, j)$ 表示活动的持续时间. 为不失一般性,进一步假定: $V = \{1, 2, \cdots, n\}$, 起始点 $s = 1$, 终止点 $t = n$.

本文是在单指令流多数据流 (SIMD) 机器上研究并行算法. 假定机器有一个无限大的共享主存贮器,有 $f(n)$ 个处理器(其中 $f(n)$ 是 $n$ 的多项式),所有处理器可同时读取主存中某个单元的内容, 但不允许同时写入某个单元; 即假定的机器模型是 SIMD-CREW-PRAM.

对 AOE 问题的并行算法, Dekel 等人建议: 在均匀洗牌及超立方这类互连网络的 SIMD 机器上的 AOE 网并行拓扑排序算法[1],Chaudhuri 在多指令流多数据流(MIMD)机器上提供了分布式算法[2],以及 Chaudhuri 等人在 SIMD-CRCW-PRAM 上 (所有处理器可以同时读、写主存单元)研究的并行算法[3]. 与后者相比, 我们的模型要弱些,需要的处理器个数也少些. 但算法更清晰简洁, 几乎可不加修改地移植到他们的模型上去实现,并具有相同的复杂性.

* 1988 年 4 月 19 日收到.

本文提供了有向图的无环性测试以及 AOE 网拓扑排序和关键路径计算等问题的并行算法，且举例说明这些并行算法的执行过程.

## §2. AOE 问题的算法

**2.1. 预备知识** 对于含有 $n$ 个顶点的有向图 $G(V, E)$，我们用 $n$ 阶邻接矩阵表示.

**定义 1.** $G$ 的邻接矩阵 $A$ 是

$$A(i, j)=\begin{cases}1, & \text{若 } \langle i, j\rangle \in E;\\ 0, & \text{否则.}\end{cases}$$

**定义 2.** $G$ 的邻接矩阵 $A$ 的传递闭包 $A^*$ 是

$$A^*(i,j)=\begin{cases}1, & \text{若从 } i \text{ 到 } j \text{ 有路径;}\\ 0, & \text{否则.}\end{cases}$$

**引理 1.** 在 SIMD-CREW-PRAM 上，计算两个 $n$ 阶矩阵乘积需要 $O(\log n)$ 时间和 $O(n^{\log 7}/\log n)$ 个处理器.

证明. 见[4].

**引理 2.** 在 SIMD-CREW-PRAM 上，将 $n$ 个元素排成有序序列，需要 $O(\log n)$ 时间和 $O(n\log n)$ 个处理器.

证明. 见[5].

**2.2. 有向图的无环性测试.** 判定一个有向图 $G(V, E)$ 是否存在有向环，是决定其为 AOE 网的首要条件. 仅当 $G$ 不包含有向环时，$G$ 才是 AOE 网. 本文的测试算法就是判别 $G$ 是否存在有向环.

**算法 TEST-CYCLE.**

用途. 测试 $G$ 是否包含有向环.

输入. $n$ 阶邻接矩阵 $A$.

输出. '有环'或'无环'.

步骤 1° $R^0 \leftarrow A + I$ // 置初始值，$I$ 为单位矩阵//

步骤 2° $R^* \leftarrow (\cdots((R^0)^2)^2\cdots)^2$ // 计算传递闭包，$\lceil \log n \rceil$ 次矩阵乘法//

步骤 3° 计算转置矩阵 $A^T$.

步骤 4° 对 $G$ 中任意 $i$，$j \in V$，若存在 $R^*(j, i) \wedge A^T(i,j) \neq 0$，则'有环'；否则'无环'.

算法中的步骤 2°，在计算 $n$ 阶矩阵的传递闭包时，包含 $\lceil \log n \rceil$ 次矩阵乘法[1]. 由引理 1，在 SIMD-CREW-PRAM 上用 $O(n^{\log 7}/\log n)$ 个处理器和 $O(\log^2 n)$ 时间，可以完成计算传递闭包 $R^*$. 算法中的步骤 1°，3° 和 4°，在 $O(n^2)$ 个处理器时，仅需 $O(1)$ 时间，故算法的时间复杂性为 $O(\log^2 n)$，所需处理器个数是 $O(n^{\log 7}/\log n)$. 由步骤 4° 可知，若存在某两个顶点 $i$，$j \in V$，且 $R^*(i, j) \wedge A^T(i, j) \neq 0$，则 $A^T(i, j) \neq 0$，亦即

1) 指逻辑乘、逻辑加意义下的矩阵乘法.

$A(j, i) \neq 0$. 这表示从 $j$ 到 $i$ 有一条有向边 $l$ 存在;同样 $R^*(i,j) \neq 0$，亦即从 $i$ 到 $j$ 有一条有向路径 $p$ 存在,故有向边 $l$ 与有向路径 $p$ 形成一个有向环 $\gamma$. 若算法执行结果包含有向环,则表示此环上的活动永远不得终止，此与事实不符. 不含有向环的有向图，才能是 AOE 网. 这样,定理 1 得证.

**定理 1.** 在 SIMD-CREW-PRAM 上，$n$ 个顶点的有向图 $G$，可在 $O(\log^2 n)$ 时间内使用 $O(n^{\log 7}/\log n)$ 个处理器测试，是否包含环.

**2.3. AOE 网的拓扑排序.** AOE 网的顶点集是一个偏序集 $(V, \prec)$,仅能在集合的部份顶点间比较前后次序，现在通过拓扑排序使得它的顶点集 $V$ 构成一个全序集，AOE 网的所有顶点都可以比较前后次序. 即对任意的 $i, j \in V$，若 $i \prec j$，则 $i$ 拓扑排序后排在 $j$ 的前面. 本文基于传递闭包算法以及一般排序算法，给出了一个新的拓扑排序算法如下:

**算法 TOPOL-SORT.**

用途. 求 AOE 网顶点集 $V$ 的拓扑次序.

输入. $n$ 阶邻接矩阵 $A$.

输出. 顶点集 $V$ 的拓扑序号 No.$(i)$，$1 \leqslant i \leqslant n$.

步骤 1°. $R^0 \leftarrow A + I$ // 初始化//.

步骤 2°. $R^* \leftarrow (\cdots((R^0)^2)^2\cdots)^2$ //计算传递闭包，$\lceil \log n \rceil$ 次矩阵乘法//.

步骤 3°. $\mathrm{Out}(i) \leftarrow \sum_{j=1}^{n} R^*(i, j)$，$i = 1, \cdots, n$. // 从顶点 $i$ 可达的顶点数//.

步骤 4°. $\mathrm{In}(i) \leftarrow \sum_{j=1}^{n} R^*(j, i)$，$i = 1, \cdots, n$. // 可达顶点 $i$ 的顶点数//.

步骤 5°. 生成记录数组 $\mathrm{CRA}(i) \leftarrow (\mathrm{Out}(i), \mathrm{In}(i), i)$，$i = 1, \cdots, n$.

步骤 6°. 按数组 CRA 的 Out$(i)$ 分量按升序排序，得到新的数组 $\mathrm{CRA}_1(i') \leftarrow (\mathrm{Out}(i'), \mathrm{In}(i'), i')$，且 $\mathrm{Out}(i'_1) \leqslant \mathrm{Out}(j'_1)$，$i'_1 < j'_1$，$i'$，$i'_1$，$j'_1 \in \{1, \cdots, n\}$.

步骤 7°. 对数组 $\mathrm{CRA}_1$ 中 Out$(i')$ 分量相同的记录,再按 In$(i')$ 非增序排序,得到排序后的数组为 $\mathrm{CRA}_2(i'') \leftarrow (\mathrm{Out}(i''), \mathrm{In}(i''), i'')$，$1 \leqslant i'' \leqslant n$.

步骤 8°. 若处理器 $i$ 包含数组元素 $\mathrm{CRA}_2(i'')$，则 No.$(i'') \leftarrow n - i + 1$.

上述算法中的步骤 7°,与实现步骤 6° 没有本质的区别,只需将 Preparata 的排序算法[5]中任意两个元素 $a_i$，$a_j$ 的比较改写如下:

若记录 $\mathrm{CRA}_1(i)$ 第一分量同 $\mathrm{CRA}_1(j)$ 的第一分量相等,则做比较之类的动作；否则不作任何动作.

**定理 2.** 在 SIMD-CREW-PRAM 上，$n$ 个顶点的 AOE 网的拓扑排序算法 TOPOL-SORT，需要 $O(\log^2 n)$ 时间和 $O(n^{\log 7}/\log n)$ 个处理器.

证明. 步骤 1° 和 2°，与定理 1 的证明相同，至多需要 $O(\log^2 n)$ 时间和 $O(n^{\log 7}/\log n)$ 个处理器；步骤 3°，4° 和 5° 显然需要 $O(\log n)$ 时间和 $O(n^2/\log n)$ 个处理器；由引理 2 知,步骤 6° 需 $O(\log n)$ 时间和 $O(n \log n)$ 个处理器;步骤 7° 同步骤 6° 一样；步骤 8° 需要 $O(1)$ 时间和 $O(n)$ 个处理器,故共需时间 $O(\log^2 n)$ 和处理器数 $O(n^{\log 7}/\log n)$. 定理证毕.

下面证明算法 TOPOL-SORT 的正确性. 根据拓扑排序定义,证明如下: 若 $i$ 是 $j$ 的前趋,则 $j$ 能到达的顶点 $i$ 也同样可以到达,故 $\mathrm{Out}(i) > \mathrm{Out}(j)$, 在步骤 6° 和 7° 的排序过程中, $i$ 所在的记录将排在 $j$ 所在记录的后面,不妨设排序后 $i$ 的序号为 $\delta$, $j$ 的序号为 $\lambda$, 则由步骤 8° 得 $n-\delta < n-\lambda$, 即在拓扑排序后 $i$ 的拓扑序号在 $j$ 的拓扑序号之前; 若 $i$ 与 $j$ 不存在前趋关系, 则有两种情况考虑: (1) $\mathrm{Out}(i) > \mathrm{Out}(j)$, 方法同前; (2) $\mathrm{Out}(i) = \mathrm{Out}(j)$, 但 $\mathrm{In}(i) \neq \mathrm{In}(j)$, 即 $i$, $j$ 不在同一条有向路径上,但需构成拓扑有序,故在 Out 值相同部分,按 In 值的降序重排,不妨设 $\mathrm{In}(i) > \mathrm{In}(j)$,经过步骤 7° 后, $j$ 所在记录排在 $i$ 所在记录的前面,则 $n-\delta < n-\lambda$, 即在拓扑排序后, $i$ 排在 $j$ 的前面. $i$ 是 $j$ 的后继, $j$ 就是 $i$ 的前趋. 方法同上. 拓扑排序算法同拓扑排序定义是一致的,故能正确地完成拓扑排序.

**2.4. 关键路径算法.** 在 AOE 网中,令 $L(i,j) = \sum_{\langle i,j\rangle \in P} w(i,j)$ 表示从 $i$ 到 $j$ 的路径 $P = i, i_1, i_2, \cdots, j$ 的长度. 定义它的最大路径长度矩阵如下:

**定义 3.** AOE 网的最大路径长度矩阵 $M(i, j)$ 是

$$M(i,j) = \begin{cases} \max\limits_{\text{所有}P} \{L(i,j)\}, & \text{若 } i \text{ 可达 } j, \text{ 且 } i \neq j; \\ 0, & \text{若 } i = j; \\ -\infty, & \text{否则.} \end{cases}$$

特别地,

$$M^0(i,j) = \begin{cases} w(i,j), & \text{若 } \langle i, j\rangle \in E, \text{ 且 } i \neq j; \\ 0, & \text{若 } i = j; \\ -\infty, & \text{否则.} \end{cases}$$

与 **2.2** 类似,可通过计算 $M^2$, $M^4$, $\cdots$, $M^{2^{\lceil \log n \rceil}}$ 得到传递闭包 $M^*$. 注意,这里的乘积项 $M(i,j)$ 由定义应为

$$M(i,j) = \max_{1 \leqslant k \leqslant n} \{M(i,j), M(i,k) + M(k,j)\}.$$

设一活动 $\langle i, j\rangle \in E$ 的最早开始时间为 $\mathrm{EST}(i, j)$, 定义 $\mathrm{EST}(i,j) = M(s, i)$; 最迟开始时间为 $\mathrm{LST}(i, j)$, 其定义为 $\mathrm{LST}(i, j) = M(s, t) - M(j, t) - w(i, j)$. 所谓 AOE 网的关键路径是指从起点 $s$ 到终点 $t$ 的所有路径中的一条路径 $P_0$, 其上的边 $\langle i,j\rangle \in P_0$ 满足 $\mathrm{EST}(i, j) = \mathrm{LST}(i, j)$. 定义 AOE 网的松弛矩阵 $s$ 的元素 $s(i,j)$ 如下: 若 $\langle i,j\rangle \in E$, 则 $s(i,j) = \mathrm{LST}(i, j) - \mathrm{EST}(i,j)$; 否则 $s(i, j) = \phi$. Chaudhuri 等人基于树归并的基本思想,给出了求最大权路径的算法[3],但算法冗长. 本文给出一个利用传递闭包求最大权路径,然后给出求关键路径算法.

**算法 CRITICAL-PATH.**

用途. 求 AOE 网的关键路径.

输入. $n$ 阶带权的邻接矩阵 $w$ 及邻接矩阵 $A$.

输出. $n$ 阶松弛矩阵 $S$.

步骤 1°. $M^0 \leftarrow W$ // 初始化//.

步骤 2°. $M^* \leftarrow (\cdots((M^0)^2)^2\cdots)^2$ // 计算传递闭包,$\lceil \log n \rceil$ 次乘法运算//.

步骤 3°. 若 $A(i,j) = 1$, 则 $\mathrm{EST}(i, j) \leftarrow M^*(s, i)$ // 置最早开始时间// $\mathrm{LST}(i,$

$j) \leftarrow M^*(s,t) - M^*(j,t) - w(i,j)$ // 计算最迟开始时间//；否则 $EST(i,j) \leftarrow \phi$ //$\phi$为任意符号，表示不作考虑之列//，$LST(i,j) \leftarrow \phi$

步骤 4°. 若 $A(i,j) = 1$，则 $S(i,j) \leftarrow LST(i, j) - EST(i,j)$；否则 $S(i,j) \leftarrow \phi$ //计算松弛矩阵//.

松弛矩阵 $S$ 为 0 的元素是 AOE 网的关键活动，大于 0 的正数则是松弛时间. 由关键路径定义可知，当且仅当从 $s$ 到 $t$ 的路径上，$\langle i, j\rangle \in E$ 满足 $EST(i, j) = LST(i, j)$ 的边$\langle i, j\rangle$ 才表示关键活动. 本文算法正确地计算了关键路径.

**定理 3.** 若 AOE 网任意两个顶点之间存在路径，则步骤 2° 能正确计算两顶点之间的最大路径长度.

证明. 用归纳法对矩阵乘法执行次数 $t'$ 进行归纳证明，约定 $M_{t'}(i,j)$ 表示 $i$ 到 $j$ 之间至多经过 $2^{t'}$ 个中间顶点的最大路径长度. 当 $t' = 0$ 时，$M_0(i, j) \leftarrow w(i, j)$. 若 $\langle i, j\rangle \in E$，定理显然成立. 设 $t' = r - 1$ 时 $M_{r-1}(i, j)$ 表示任意两顶点 $i, j$ 间至多有 $2^{r-1}$ 个中间顶点的最大路径长度，现在证明：当 $t' = r$ 时，$M_r(i, j)$ 也是 $i$ 到 $j$ 之间至多有 $2^r$ 个中间顶点的最大路径长度. 倘若 $M_r(i, j)$ 不是 $i$ 到 $j$ 之间至多有 $2^r$ 个中间顶点的最大路径长度，则从 $i$ 到 $j$ 的路径上经过了中间顶点

$$k_1, k_2, \cdots, k_p, \quad 1 \leqslant p \leqslant n. \tag{1}$$

设从 $i$ 到 $j$ 间经过至多 $2^r$ 个中间顶点的最大路径为 $P_0$，$P_0$ 的中间顶点为

$$m_1, m_2, \cdots, m_q, \quad 1 \leqslant q \leqslant n. \tag{2}$$

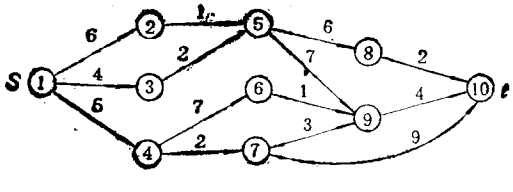
令第 $l + 1$ 个中间顶点序列(1)和(2)不相同，$l = \min(p, q)$，即 $k_1 = m_1, \cdots, k_l = m_l, k_{l+1} \neq m_{l+1}$，由步骤 2° 的乘法定义，$M_r(i, j) = \max\{M_{r-1}(i, j), M_{r-1}(i, k) + M_{r-1}(k, j)\}$，其中 $k = k_l = m_l$，由此推得 $M_{r-1}(k, j)$ 不是 $k_l$ 到 $j$ 间至多经过 $2^{r-1}$ 个中间顶点的最大长度路径. 这与归纳假设矛盾，故步骤 2° 算得的 $M_r(i, j)$ 能正确计算 $i$ 到 $j$ 间至多有 $2^r$ 个中间顶点的最大长度路径. 证毕.

**定理 4.** 在 SIMD-CREW-PRAM 上，$n$ 个顶点的 AOE 网的关键路径，可经 $O(\log^2 n)$ 时间并使用 $O(n^{\log 7}/\log n)$ 个处理器求得.

证明. 算法 CRITICAL-PATH 的步骤 1°，3° 和 4°，仅需 $O(1)$ 时间和 $O(n^2)$ 个处理器；步骤 2° 需要 $O(\log^2 n)$ 时间和 $O(n^{\log 7}/\log n)$ 个处理器，故求得关键路径共需 $O(\log^2 n)$ 时间和 $O(n^{\log 7}/\log n)$ 个处理器. 证毕.

## §3. 实　例

给出有向图 $G(V, E)$



它的邻接矩阵 $A$ 和带权邻接矩阵 $W$ 如下：

$$A=\begin{bmatrix}0&1&1&1&0&&&&\\&0&0&0&1&&0&&\\&&0&0&1&&&&\\&&&0&0&1&1&0&0&0\\&&&&0&0&0&1&1&0\\&&&&&0&0&0&1&0\\&&&&&&0&0&1&1\\&&0&&&&&0&0&1\\&&&&&&&&0&1\end{bmatrix},\quad W=\begin{bmatrix}0&6&4&5&0&&&&\\&0&0&0&1&&0&&\\&&0&0&2&&&&\\&&&0&0&7&2&0&0&0\\&&&&0&0&0&6&7&0\\&&&&&0&0&0&0&1&0\\&&&&&&0&0&0&3&9\\&&0&&&&&0&0&0&2\\&&&&&&&&0&0&4\\&&&&&&&&&&0\end{bmatrix}$$

下面分别给出三个算法各个步骤的计算结果：

**1.** 无环性测试算法 TEST-CYCLE

步骤 1°

$$R^0=\begin{bmatrix}1&1&1&1&0&&&&\\&0&1&0&0&1&&0&\\&&1&0&1&&&&\\&&&1&0&1&1&0&0&0\\&&&&1&0&0&1&1&0\\&&&&&1&0&0&1&0\\&&&&&&1&0&1&1\\&&0&&&&&1&0&1\\&&&&&&&&1&0\\&&&&&&&&&1\end{bmatrix}$$

步骤 2°

$$R^*=\begin{bmatrix}1&1&1&1&1&1&1&1&1&1\\&0&1&0&0&1&0&0&1&1&1\\&&1&0&1&0&0&1&1&1\\&&&1&0&1&1&0&1&1\\&&&&1&0&0&1&1&1\\&&&&&1&0&0&1&1\\&&&&&&1&0&1&1\\&&&&&&&1&0&1\\&&&&&&&&1&1\\&&&&&&&&&1\end{bmatrix}$$

步骤 3° 即将 $A$ 的行列互换得出 $A^T$. 在步骤 4° 中，$R^*\wedge A^T=\mathbf{0}$，故有向图 $G(V,E)$ 不存在有向环，是一个 AOE 网.

**2.** 拓扑排序算法 TOPOL-SORT

步骤 1° 与步骤 2° 与 TEST-CYCLE 相同. 步骤 3° 与步骤 4°，包含在步骤 5° 中.

**步骤 5°** 至步骤 8°，见下面的表。

| 步 骤 5° | | | 步 骤 6° | | | 步 骤 7° | | | 步 骤 8° | |
|---|---|---|---|---|---|---|---|---|---|---|
| CRA | | | $CRA_1$ | | | $CRA_2$ | | | PE | No. |
| Out | In | $i$ | Out | In | $i'$ | Out | In | $i''$ | | |
| 10 | 1 | 1 | 1 | 10 | 10 | 1 | 10 | 10 | 1 | 10 |
| 5 | 2 | 2 | 2 | 5 | 8 | 2 | 8 | 9 | 2 | 9 |
| 5 | 2 | 3 | 2 | 8 | 9 | 2 | 5 | 8 | 3 | 8 |
| 5 | 2 | 4 | 3 | 3 | 6 | 3 | 3 | 6 | 4 | 7 |
| 4 | 4 | 5 | 3 | 3 | 7 | 3 | 3 | 7 | 5 | 6 |
| 3 | 3 | 6 | 4 | 4 | 5 | 4 | 4 | 5 | 6 | 5 |
| 3 | 3 | 7 | 5 | 2 | 2 | 5 | 2 | 2 | 7 | 4 |
| 2 | 5 | 8 | 5 | 2 | 3 | 5 | 2 | 3 | 8 | 3 |
| 2 | 8 | 9 | 5 | 2 | 4 | 5 | 2 | 4 | 9 | 2 |
| 1 | 10 | 10 | 10 | 1 | 1 | 10 | 1 | 1 | 10 | 1 |

即上述 AOE 网顶点的拓扑排序序列为 1,4,3,2,5,7,6,8,9,10.

**3.** 关键路径算法 CRITICAL-PATH

步骤 1°. 输入带权邻接矩阵$W$到 $M^0$ 中.

步骤 2°.

$$M^* = \begin{bmatrix} 0 & 6 & 4 & 5 & 7 & 12 & 7 & 13 & 14 & 18 \\ -\infty & -\infty & -\infty & -\infty & 1 & -\infty & -\infty & 7 & 8 & 12 \\ -\infty & -\infty & -\infty & -\infty & 2 & -\infty & -\infty & 8 & 9 & 13 \\ -\infty & -\infty & -\infty & -\infty & -\infty & 7 & 2 & -\infty & 8 & 12 \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 6 & 7 & 11 \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 1 & 5 \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 3 & 9 \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 2 \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 4 \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty \end{bmatrix}.$$

步骤 3°.

$$\mathrm{EST} = \begin{bmatrix} \phi & 0 & 0 & 0 & \phi & & & & & \\ \phi & \phi & \phi & 6 & & \phi & & & & \\ \phi & \phi & 4 & & & & & & & \\ \phi & \phi & 5 & 5 & \phi & \phi & \phi & & & \\ \phi & \phi & \phi & 7 & 7 & \phi & & & & \\ \phi & \phi & \phi & 12 & \phi & & & & & \\ \phi & & & & \phi & \phi & 7 & 7 & & \\ & & & & \phi & \phi & 13 & & & \\ & & & & & \phi & 14 & & & \\ & & & & & & \phi & & & \end{bmatrix}, \quad \mathrm{LST} = \begin{bmatrix} \phi & 0 & 1 & 1 & \phi & & & & & \\ \phi & \phi & \phi & 6 & & \phi & & & & \\ \phi & \phi & 5 & & & & & & & \\ \phi & \phi & 6 & 7 & \phi & \phi & \phi & & & \\ \phi & \phi & \phi & 10 & 7 & \phi & & & & \\ \phi & \phi & \phi & 13 & \phi & & & & & \\ \phi & & & & \phi & \phi & 11 & 9 & & \\ & & & & \phi & \phi & 16 & & & \\ & & & & & \phi & 14 & & & \\ & & & & & & \phi & & & \end{bmatrix}.$$

步骤 4°。

$$S=\begin{bmatrix}
\phi & 0 & 1 & 1 & \phi & & & & & \\
 & \phi & \phi & \phi & 0 & & \phi & & & \\
 & & \phi & \phi & 1 & & & & & \\
 & & & \phi & \phi & 1 & 2 & \phi & \phi & \phi \\
 & & & & \phi & \phi & \phi & 3 & 0 & \phi \\
 & & & & & \phi & \phi & \phi & 1 & \phi \\
 & & & & & & \phi & \phi & 4 & 2 \\
 & & & \phi & & & & \phi & \phi & 3 \\
 & & & & & & & & \phi & 0 \\
 & & & & & & & & & \phi
\end{bmatrix}.$$

由矩阵 $S$ 知，$s(i,j)$ 为 0 的边有$\langle 1,2\rangle$,$\langle 2,5\rangle$,$\langle 5,9\rangle$,$\langle 9,10\rangle$,由这些边构成了 AOE 网的一条关键路径,即 $s=1\to 2\to 5\to 9\to 10=t$。

## §4. 结 论

在 SIMD-CREW-PRAM 上,以矩阵乘法、排序等基本算法为基础，给出了测定一个有向图$G$是否为 AOE 网以及 AOE 网的拓扑排序、关键路径计算等并行算法，并可在 $O(\log^2 n)$ 时间内使用 $O(n^{\log 7}/\log n)$ 个处理器实现。

## 参 考 文 献


[1] E. Dekel, D. Nassimi, S. Sahni, Parallel matrix and graph algorithms, *SIAM J. Comput.*, 10(1981), 657—675.

[2] P. Chaudhuri, Distributed algorithm for analysing activity Networks, *Int. J. Electronics*, 60(1986), 603—607.

[3] P. Chaudhui, P. K. Ghosh, Parallel algorithms for analyzing activity networks," *BIT*, 26(1986), 418—429.

[4] A. K. Chandra, Maximal Parallelism in Matrix Multiplication, IBM Report, RC6193, Yorktown, 1976.

[5] F. P. Preparata, Parallelism in sorting, *Int. Conf. on Parallel Processing*, 1977.