

文章编号: 1000-2022(2002) 05-0711-04

一种有向权图的拓扑排序算法及其应用

王顺凤

(南京气象学院 数学系, 江苏 南京 210044)

摘 要: 提出一种有向权图的拓扑排序算法, 并给出实例说明其应用。

关键词: 拓扑排序; 算法; AOV-网

中图分类号: O233 **文献标识码:** A

有向无环图(Directed Acycline Graph)^[1]是描述一项工程或系统进行过程的有效工具。拓扑排序^[2]是有向无环图的一种重要运算。对有向无环图进行拓扑排序, 可以得到对应于工程(或系统)的各项子工程(或系统的分过程)的一个线性序列。这对工程的计划管理、系统的统筹调度有着重要的指导意义。

工程项目计划管理问题常常受紧前活动约束^[3], 有时要求各活动的等待时间之和最小, 这一问题目前尚未得到解决。文献[4]给出了一种优化模型, 并用分支定界法求解, 该方法在计算机上实现较为困难, 且为一种枚举搜索算法, 属于 NP 问题。本文提出一种有向权图的拓扑排序算法及其实现, 解决了该问题, 并用一实例说明其应用。

1 概 念

若集合 X 上的关系 R 是自反的、反对称的和传递的, 则称 R 是集合 X 上的偏序(关系)。设 R 是集合 X 上的偏序, 如果对每个 $x, y \in X$, 必有 xRy 或 yRx , 则称 R 是集合 X 上的全序(关系)。由某个集合上的一个偏序得到该集合上的一个全序, 这个操作称为拓扑排序。

一个偏序的有向图可用来表示一个流程图, 该流程图可以是一个施工流程图, 或者是一个产品生产的流程图, 或者是一个数据流程图(每个顶点表示一个过程)。图中每一条有向边表示两个子工程之间的次序关系。这样的图常称为活动结点网。

用顶点表示活动, 用有向边表示活动间的优先关系的有向图称为活动结点网^[3](AOV-网, Activity On Vertex Network)。在 AOV-网中, 若 $\langle i, j \rangle$ 是网中一条有向边, 则活动 i 必须在活动 j 之前完成。

在 AOV-网中, 不应该出现有向环, 否则, 对应的活动便无法进行。任何有向无环图的结点都可以安排在一个拓扑序列中^[2]。

以下算法基于 AOV-网, 且在任一顶点的出边上附加一权, 表示该活动所需进行的时间。

收稿日期: 2002-01-04; 改回日期: 2002-05-16

基金项目: 上海市信息化专项资金项目

作者简介: 王顺凤(1965-), 女, 江苏宜兴人, 讲师, 硕士生, 主要从事数值预报及数值模拟方面的研究。

2 带权拓扑排序算法

设有向权图 $G = \langle V, E \rangle$, $V = \{v_1, v_2, \dots, v_n\}$, $E = \{e_1, e_2, \dots, e_n\}$, $W = \{w_1, w_2, \dots, w_n\}$ 。有向边 $e_k = \langle v_i, v_j \rangle$ 表示活动 v_i 必须在活动 v_j 前完成, 边 $e_k = \langle v_i, v_j \rangle$ 上的权 w_i 表示活动 v_i 所需进行的时间。

一个有向图的拓扑序列如果存在, 则它未必是唯一的。带权拓扑序列就是求各拓扑序列中各活动的等待时间之和最小的拓扑序列。如果各活动对应于工程, 其现实意义是各子工程的完工时间之和最小。

拓扑排序方法为: 1) 在有向图中选择一个入度为 0 且其出边之权最小的顶点, 输出之; 2) 从有向图中删掉此顶点及其所有出边。

重复上述两步, 直到全部顶点均已输出, 或者当前图中不存在没有入度为 0 的顶点。后一种情况说明有向图中存在环, 拓扑排序不能继续。以下讨论该拓扑排序方法分别在邻接矩阵和邻接表上的实现。

2.1 邻接矩阵上的拓扑排序算法

设 A 为有向权图 G 的邻接矩阵, $A = (a_{ij})_{n \times n}$, 其中

$$a_{ij} = \begin{cases} 0, & \langle v_i, v_j \rangle \notin E; \\ w_i, & \langle v_i, v_j \rangle \in E. \end{cases}$$

称 w_i 为 A 的第 i 行的行值。

找图 G 中入度为 0 的顶点就是在 A 中找全为 0 的列, 删除某个顶点的所有出边就是把邻接矩阵中对应于该列的行置成全 0。

算法 1——邻接矩阵上的拓扑排序算法: 1) 取 1 作为第一个新序号。2) 找出 A 的全部尚未得到新序号的全 0 的矩阵列。如果没有, 则停止。此时若矩阵所有列都已得到新序号, 则拓扑排序完成。否则说明是有环图。3) 找出与全 0 列对应的全部行中行值最小的行, 将新序号赋给与之相应的列。4) 将步骤 3) 中行值最小的行置成全 0。5) 新序号增 1, 转步骤 2)。

算法结束时, 若拓扑排序能够完成, 则矩阵的每一列得到一个新序号, 它就是该列所对应的顶点在拓扑排序序列中的序号。

算法中步骤 2)、3) 对邻接矩阵的行和列分别扫描一遍, 所以其时间复杂度为 $O(n^2)$ 。存储量为一个 $n \times n$ 矩阵和一个 n 维向量。

2.2 邻接表上的拓扑排序算法

设有向图用邻接表存储, 保存 1 个顺序存储的结点表和 n 个链接存储的边表, 结构为:

结点表

data	indegree	weight	out
------	----------	--------	-----

边表

vex_no	link
--------	------

```
Struct node {
    datatype data; // 顶点信息
    int indegree; // 顶点入度
    float weight; // 顶点之权, 顶点活动所需时间
    struct node * out; // 指向与该顶点相邻的第一个顶点
} nodelist[ n ]; // 结点表
```

```
struct edge {
    int vex_no; // 顶点序号
    struct edge * link; // 指向与同一顶点相邻的下一个顶点
} // 边表

struct edge * head; // 存储当前入度为 0 的顶点的单链表, 其中顶点以 weight 的不减顺序排列
```

算法 2——邻接表上的带权拓扑排序算法:

```
Status TopologicalSort(void)
// 有向图采用邻接表存储结构
// 若 G 中无环, 则输出 G 的顶点的一个带权拓扑序列并返回 OK, 否则, 返回 ERROR
{
    for (i= 0; i< n; i+ ){ // 建 0 入度顶点的单链表
        if (! nodelist[i]. indegree) InsertLinkTable(head, i); // 入度为 0 的顶点插入到单链表
    }
    count= 0; // 对输出顶点计数
    while (! head){
        i= head -> vex_no; head= head -> link; // 取出 0 入度顶点单链表中第一个顶点
        printf (i, nodelist[i]. data); + + count; // 输出 i 号顶点, 并计数
        for (p= nodelist[i]. out; p; p= p -> link){
            k= p -> vex_no;
            if (! (-- nodelist[k]. indegree)) InsertLinkTable(head, k);
        } // 对 i 号顶点的每一个邻接顶点的入度减 1, 若入度为 0, 则插入有序链表 head 中
    }
    if (count < n) return ERROR; // 图中有环
    else return OK;
}
```

其中: InsertLinkTable(head, i) 在 head 所指的单链表中插入顶点号为 i 结点, 并要求按结点的 weight 字段值的不减顺序插入。算法从略。

对于 (n, m) 权图, 该算法需存储结点表中 n 个结点和边表中的 m 个结点, 存储单元为 $n + m$ 。算法需扫描结点表和边表中的所有结点, 所以, 其时间复杂度为 $O(m + n)$ 。可见其效率远远高于邻接矩阵上的拓扑排序算法。

3 应用实例

1991 年上海市大学生数学模型竞赛试题 B^[5]为: 现有 14 件工件等待在一台机床上加工, 某些工件的加工必须安排在另一些工件完工以后才能开始, 第 j 号工件的加工时间 t_j 及先期必须完成的工件号 i 由表 1 给出。

表 1 工件加工时间和约束
Table 1 Restriction and time lasting of workpieces machining

	工件号 j													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
t_j	20	28	25	16	42	12	32	10	24	20	40	24	36	16
前期工件号 i	3, 4	5, 7, 8	5, 9		10, 11	3, 8, 9	4	3, 5, 7	4		4, 7	6, 7, 14	5, 12	1, 2, 6

若给出一个加工顺序, 则确定了每个工件的完工时间(包括等待与加工两个阶段)。试设计

一个满足条件的加工顺序, 使各个工件的完工时间之和最小。

解: 将每个工件的加工看作一个活动, 得如图 1 所示的 AOV-网。

满足问题要求的加工顺序就是该 AOV-网的带权拓扑序列。由上述算法计算得拓扑序列为 4、10、9、7、11、5、3、8、6、1、2、14、12、13。

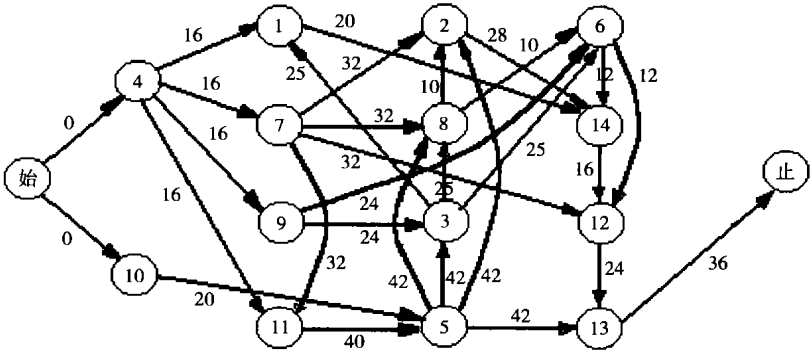


图 1 工件加工的 AOV-网

Fig. 1 The AOV-net of workpieces machining

4 结 论

算法 1、2 给出了一种有向权图的拓扑排序问题在两种存储结构上的实现, 算法 2 在时间效率上远优于算法 1。它们可在一类排序问题, 特别是在有紧前工序约束下的工程计划问题上有着重要的应用。文献[5] 指出在有紧前工序的约束下, 上例中的问题不能有效解决。算法 1、2 给出了这一问题的有效解决途径。

参考文献:

[1] 耿素云, 屈婉玲. 离散数学基础[M]. 北京: 北京大学出版社, 1994: 254.
[2] 许卓群, 张乃孝, 杨冬青, 等. 数据结构[M]. 北京: 高等教育出版社, 1987: 229-233.
[3] 杜端甫. 运筹图论[M]. 北京: 北京航空航天大学出版社, 1990: 276-277.
[4] 谭永基, 俞文敏. 数学模型[M]. 上海: 复旦大学出版社, 1997: 49-53.
[5] 李尚志, 王树禾, 苏 淳, 等. 数学建模竞赛教程[M]. 南京: 江苏教育出版社, 1996: 308; 312.

An Algorithm of Topological Sequencing for Weighted Directed Graph and Its Application

WANG Sun-feng

(Department of Mathematics, NIM, Nanjing 210044, China)

Abstract: This Paper provides an algorithm of topological sequencing for weighted directed graph, and illustrates its application with an example.
Key words: topological sequencing; algorithm; AOV-net