Generating all the Acyclic Orientations of an Undirected Graph

Valmir C. Barbosa[1]
Universidade Federal do Rio de Janeiro
COPPE - Programa Eng. Sistemas e Computação
Caixa Postal 68511, 21945-970 Rio de Janeiro, RJ, Brasil
valmir@cos.ufrj.br

Jayme L. Szwarcfiter
Universidade Federal do Rio de Janeiro
Instituto de Matemática e NCE
Caixa Postal 2324, 20001-970 Rio de Janeiro, RJ, Brasil
jayme@nce.ufrj.br

KEY WORDS: acyclic orientations, algorithms, graphs

ABSTRACT

Let G be an undirected graph with n vertices, m edges and α acyclic orientations. We describe an algorithm for finding all these orientations in overall time $O((n+m)\alpha)$ and delay complexity $O(n(n+m))$. The space required is $O(n+m)$.

[1]Participated in this research while visiting the International Computer Science Institute, Berkeley, CA, USA

1 Introduction

G denotes a finite undirected graph with no loops nor multiple edges. $V(G)$ and $E(G)$ are its vertex and edge sets, respectively with $n = |V(G)|$ and $m = |E(G)|$. An **orientation** of G is a digraph obtained by directing the edges of G. The orientation is **acyclic** if it contains no directed cycles. Acyclic orientations of a graph have been considered in [4, 5], where it has been shown that their cardinality is closely related to the chromatic polynomial of the graph. We consider the problem of generating all the acyclic orientations of the undirected graph G. Let α be the total number of such orientations. We describe an algorithm which finds all the α orientations in overall complexity $O((n+m)\alpha)$ and delay complexity $O(n(n+m))$. The space required is $O(n+m)$.

The problem of generating all the acyclic orientations of a graph G has been motivated by a distributed scheduling mechanism with applications to resource sharing and to the parallel simulation of models of some complex systems [1, 2]. In such applications, a vertex of G represents a process that must from time to time compute on local information and on data received from other vertices. The edges of G are such that processes that must communicate with each other correspond to adjacent vertices. What these applications have in common is that the processes corresponding to any two adjacent vertices must not compute concurrently. In order to enforce this constraint, a subset of processes is allowed to compute concurrently only if their corresponding vertices are sinks in some orientation of G. The distributed scheduling mechanism starts at any acyclic orientation of G, and employs a very simple rule to generate another orientation, once the processes that correspond to sinks in the current acyclic orientation have finished their computation. The new orientation is guaranteed to be acyclic.

From the perspective of this scheduling mechanism, the acyclic orientations of G are partially ordered by a measure of concurrency. However, no efficient algorithm is likely to exist for the problem of optimizing concurrency, which is a central issue when selecting the initial acyclic orientation. Generating all the acyclic orientations of relatively small graphs is expected to yield insights into heuristic strategies to this optimization problem.

The approach of using sinks of acyclic orientations of G has been shown to be advantageous in relation to employing other possible independent subsets of vertices of G [1].

Let G be an undirected graph. By $N(v)$ we represent the set of vertices adjacent to v in G. Let $v_1, \ldots, v_n$ be a sequence of the vertices of G. Denote by G_i the subgraph induced in G by $\{v_1, \ldots, v_i\}$, and for $v_j \in V(G)$, $N_i(v_j) = N(v_j) \cap \{v_1, \ldots, v_i\}$. An orientation of G is represented by $\vec{G}$. For $v, w \in V(\vec{G})$, if $\vec{G}$ contains a path from v to w then v is an **ancestor** of w in $\vec{G}$. $A_i(v_j)$ denotes the set of ancestors of v_j in $\vec{G}_i$. Let $S \subseteq V(\vec{G})$. A **topological ordering** of S in $\vec{G}$ is a sequence W of the vertices of S, such that all edges of $\vec{G}$ between pairs of vertices of S are from left to right in W. By $\vec{G} - v$ we mean the digraph obtained from $\vec{G}$ by removing v and all edges entering or leaving it. Similarly, $\vec{G} + v$ is the digraph which results by adding to $\vec{G}$ the isolated vertex v. Finally, we use the same notation $(v, w) \in E(G)$ and $(v, w) \in E(\vec{G})$, the meaning (and possible direction of the edge) being clear from the context.

In Section 2 we describe the concepts and proofs in which the algorithm is based. The algorithm itself is given in Section 3.

2 Correctness

Let $v_1, \ldots, v_n$ be an arbitrary sequence of the vertices of G, and $\vec{G}_{i-1}$ an acyclic orientation of G_{i-1}, $1 < i \leq n$. A **direction assignment for** $\vec{G}_{i-1}$ is a function

$$d_{i-1} : N_i(v_i) \rightarrow \{0, 1\}.$$

If $d_{i-1}(v_p) = 0$ for all $v_p \in N_i(v_i)$ then d_{i-1} is called **null**. Similarly, d_{i-1} is **unit** when $d_{i-1}(v_p) = 1$, for all $v_p \in N_i(v_i)$. A direction assignment is **legal** when

$$v_p, v_q \in N_i(v_i) \text{ and } v_q \in A_{i-1}(v_p) \Rightarrow \text{if } d_{i-1}(v_q) = 1 \text{ then } d_{i-1}(v_p) = 1.$$

The **extension of $\vec{G}_{i-1}$ induced by** d_{i-1} is an orientation $\vec{G}_i$ of G_i, such that

$$\vec{G}_{i-1} = \vec{G}_i - v_i, \text{ and}$$

$$v_p \in N_i(v_i) \Rightarrow \begin{cases} (v_p, v_i) \in E(\vec{G}_i), & \text{if } d_{i-1}(v_p) = 0 \\ (v_i, v_p) \in E(\vec{G}_i), & \text{otherwise.} \end{cases}$$

The lemma below relates acyclic orientations of G_i to those of G_{i-1}.

Lemma 1: There exists a one-to-one correspondence between acyclic orientations $\vec{G}_i$ of G_i and pairs $(\vec{G}_{i-1}, d_{i-1})$, where d_{i-1} is a legal direction assignment for the acyclic orientation $\vec{G}_{i-1}$ of G_{i-1}.

Proof: Define a function f from the set of all pairs $(\vec{G}_{i-1}, d_{i-1})$ to that of all acyclic orientations $\vec{G}_i$, as follows: $f[(\vec{G}_{i-1}, d_{i-1})] = \vec{G}_i$, where $\vec{G}_i$ is the extension of $\vec{G}_{i-1}$ induced by d_{i-1}.

First, we need to prove that $f[(\vec{G}_{i-1}, d_{i-1})]$ is acyclic. By hypothesis, $\vec{G}_{i-1}$ is acyclic and d_{i-1} is a legal assignment for $\vec{G}_{i-1}$. Suppose the assertion false. Then $\vec{G}_{i-1}$ contains a cycle C. Because $\vec{G}_{i-1}$ is an induced subdigraph of $\vec{G}_i$, it follows that C must contain v_i. Since every edge of G_i is assigned to exactly one direction in $\vec{G}_i$, C has at least 3 vertices. Hence C contains distinct vertices $v_p, v_q \in N_i(v_i)$, entering and leaving v_i in C, respectively. That is, $d_{i-1}(v_p) = 0$ and $d_{i-1}(v_q) = 1$. Also, $C - v_i$ is a path from v_q to v_p in $\vec{G}_{i-1}$, i.e. $v_q \in A_{i-1}(v_p)$. Consequently, d_{i-1} is not legal, a contradiction. Hence $f[(\vec{G}_{i-1}, d_{i-1})]$ is acyclic.

Conversely, let $\vec{G}_i$ be an acyclic orientation of G_i. We prove that there exists an acyclic orientation $\vec{G}_{i-1}$ and a legal assignment d_{i-1} for it, such that $\vec{G}_i = f[(\vec{G}_{i-1}, d_{i-1})]$. Define $\vec{G}_{i-1} = \vec{G}_i - v_i$ and d_{i-1} to be as follows.

$$v_p \in N_i(v_i) \Rightarrow d_{i-1}(v_p) = \begin{cases} 0, & \text{if } (v_p, v_i) \in E(\vec{G}_i) \\ 1, & \text{if } (v_i, v_p) \in E(\vec{G}_i) \end{cases}$$

We have to show that d_{i-1} is legal. Suppose it is not. Then there exist $v_p, v_q \in N_i(v_i)$ such that $d_{i-1}(v_p) = 0$, $d_{i-1}(v_q) = 1$ and $v_q \in A_{i-1}(v_p)$. The

latter implies $v_q \in A_i(v_p)$. In this case, a path from v_q to v_p in G_i together with the edges $(v_p, v_i), (v_i, v_q) \in E(\vec{G}_i)$ forms a cycle in $\vec{G}_i$, a contradiction. Hence d_{i-1} is a legal assignment for the acyclic orientation $\vec{G}_{i-1}$ of G_{i-1}. That is, $\vec{G}_i$ is precisely the extension of $\vec{G}_{i-1}$ induced by d_{i-1}, meaning $\vec{G}_i = f[(\vec{G}_{i-1}, d_{i-1})]$.

Last, we prove that f is injective. Let $\vec{G}_{i-1}, \vec{G}'_{i-1}$ be acyclic orientations of G_{i-1}. Also, let d_{i-1}, d'_{i-1} be legal assignments for $\vec{G}_{i-1}, \vec{G}'_{i-1}$, respectively. Let $\vec{G}_i = f[(\vec{G}_{i-1}, d_{i-1})]$ and $\vec{G}'_i = f[(\vec{G}'_{i-1}, d'_{i-1})]$. By hypothesis, $(\vec{G}_{i-1}, d_{i-1}) \neq (\vec{G}'_{i-1}, d'_{i-1})$. If $\vec{G}_{i-1} \neq \vec{G}'_{i-1}$ then $\vec{G}_i \neq \vec{G}'_i$, because $\vec{G}_i$ is an extension of $\vec{G}_{i-1}$, and $\vec{G}'_i$ an extension of $\vec{G}'_{i-1}$, implying the assertion. The second case to examine is $d_{i-1} \neq d'_{i-1}$. It means that there is $v_p \in N_i(v_i)$, such that $d_{i-1}(v_p) \neq d'_{i-1}(v_p)$. Hence the edge (v_p, v_i) of G_i is assigned to opposite directions in $\vec{G}_i$ and $\vec{G}'_i$. That is, $\vec{G}_i \neq \vec{G}'_i$, completing the proof. □

Denote by $\vec{G}_{i-1}$ an acyclic orientation of G_{i-1}. Let $w_1, \ldots, w_\ell$, $\ell = |N_i(v_i)|$, be a sequence W of the vertices of $N_i(v_i)$. Denote by d_{i-1}, d'_{i-1} two direction assignments for $\vec{G}_{i-1}$. Call d_{i-1} **lexicographically smaller than** d'_{i-1} **relative to** W, when there exists an index j such that $d_{i-1}(w_p) = d'_{i-1}(w_p)$, $1 \leq p < j$, while $d_{i-1}(w_j) < d'_{i-1}(w_j)$. When $d_{i-1} \neq unit$, denote by $(d_{i-1} + 1)_W$ the direction assignment which follows d_{i-1} in the increasing lexicographical ordering relative to W, of all direction assignments for $\vec{G}_{i-1}$. Finally, denote by $closure(d_{i-1})$ the direction assignment obtained from d_{i-1} by setting to 1 all values $d_{i-1}(v_p)$ such that $v_q \in A_{i-1}(v_p)$ and $d_{i-1}(v_q) = 1$, for all $v_p, v_q \in N_i(v_i)$. It follows that $closure(d_{i-1})$ is necessarily legal, for any assignment d_{i-1}.

The next lemma describes the legal direction assignments for a given orientation, in increasing lexicographical ordering.

Lemma 2: Let $\vec{G}_{i-1}$ be an acyclic orientation of G_{i-1}. Let W be a topological ordering of $N_i(v_i)$ in $\vec{G}_{i-1}$, and $d_{i-1} \neq unit$ a legal direction assignment for $\vec{G}_{i-1}$. In the increasing lexicographical ordering of all legal assignments for $\vec{G}_{i-1}$ relative to W, the assignment which immediatly follows d_{i-1} is $closure[(d_{i-1} + 1)_W]$.

Proof: Denote by d' the assignment $(d_{i-1}+1)_W$ and $d'' = closure(d')$. Let j be the rightmost index in W such that $d_{i-1}(w_j) = 0$. Then $d'(w_p) = d_{i-1}(w_p)$, $1 \leq p < j$, $d'(w_j) = 1$ and $d'(w_q) = 0$, $j < q \leq \ell$. Also, because d_{i-1} is legal and W a topological ordering it follows that $d''(w_p) = d'(w_p)$, $1 \leq p \leq j$. Suppose the lemma not to be true. Then there exists a legal assignment d^* for $\vec{G}_{i-1}$, such that $d_{i-1} < d^* < d''$. Consequently, there is an index $t > j$ satisfying $d''(w_p) = d^*(w_p)$, $1 \leq p < t$, $d''(w_t) = 1$ and $d^*(w_t) = 0$. However, $d'(w_t) = 0$. Thus, applying the definition of closure we conclude that there exists $w_q \in A_{i-1}(w_t)$, such that $d'(w_q) = 1$. Since W is a topologocal ordering, it follows $q < t$. Hence $d^*(w_q) = 1$, meaning that d^* is not legal, a contradiction. □

3 The Algorithm

The algorithm is based upon Lemmas 1 and 2, by an inductive argument. Suppose the method correctly enumerates all acyclic orientations $\vec{G}_{i-1}$ of G_{i-1}. By Lemma 1, each acyclic orientation $\vec{G}_i$ is an extension of some $\vec{G}_{i-1}$, induced by a convenient legal assignment d_{i-1}, and conversely. The algorithm proceeds as follows. For each acyclic orientation $\vec{G}_{i-1}$ of G_{i-1} and for each legal assignment d_{i-1} for $\vec{G}_{i-1}$ construct the extension $\vec{G}_i$ of $\vec{G}_{i-1}$ induced by d_{i-1}. The orientations so obtained form exactly the set of all acyclic orientations of G_i.

The problem that remains is to generate all legal assignments d_{i-1} for an orientation $\vec{G}_{i-1}$. This problem is equivalent to that of finding all distinct ways of labelling the vertices of an acyclic digraph, using the labels 0 and 1, so that no descendant of a vertex labelled 1 has label 0. A solution of it is given by Lemma 2. Let W be a topological ordering of $N_i(v_i)$ in $\vec{G}_{i-1}$. We obtain the legal assignments in increasing lexicographical ordering relative to W. If $N_i(v_i) = \emptyset$ there are no assignments. Otherwise, the first and last assignments are the *null* and *unit*, respectively. Let $d_{i-1} \neq unit$ be a legal assignment for $\vec{G}_{i-1}$. The legal assignment which follows d_{i-1} in the lexicographical ordering is $closure[(d_{i-1}+1)_W]$.

The recursive formulation below describes the process. The current legal

assignment is represented by d, while $v_1, \ldots, v_n$ is an arbitrary sequence of the vertices of G. By convention, $\vec{G}_0 = \emptyset$. The external call is $ACYCLIC(1, \emptyset)$.

proc $ACYCLIC(i, \vec{G}_{i-1})$
  if $i > n$ **then** enumerate acyclic orientation $\vec{G}_n$
  else if $N_i(v_i) = \emptyset$ **then** $ACYCLIC(i+1, \vec{G}_{i-1} + v_i)$
    else $W :=$ topological ordering of $N_i(v_i)$ in $\vec{G}_{i-1}$
      $d := null$; $last :=$ **false**
      repeat $\vec{G}_i :=$ extension of $\vec{G}_{i-1}$ induced by d
        $ACYCLIC(i+1, \vec{G}_i)$
        if $d = unit$ **then** $last :=$ **true**
        else $d := closure[(d+1)_W]$
      until $last =$ **true**

The computation of a topological ordering requires $O(n+m)$ steps [3], while finding an extension of a given orientation can be done in $O(n)$ steps. For an assignment d, $closure[(d+1)_W]$ can be easily obtained also in $O(n+m)$ steps. The work performed within each iteration of the **repeat** block is charged to the recursive call which occurs in this iteration. Consequently, the complexity per call of the procedure is $O(n+m)$ and the delay complexity of the algorithm is $O(n(n+m))$.

However, the average complexity of the algorithm is better. The recursion describes a tree T in which all the leaves are at the same level n. Each leaf corresponds to an acyclic orientation of G. The total number of steps performed by the algorithm can be obtained by adding the work done at the leaves of T with that corresponding to the internal nodes. The total computation at the leaves requires $O((n+m)\alpha)$ steps. There are two kinds of internal nodes. The set I_1 of nodes with exactly one child each, and those with more than one child, forming set I_2. Each internal node of I_1 corresponds to a computation of $ACYCLIC(i, \vec{G}_{i-1})$ in which $N_i(v_i) = \emptyset$. In this situation, the procedure is computed in a constant number of steps. Also, we know that $|I_1| < n\alpha$. That is, the total computation at the internal nodes of I_1 is $O(n\alpha)$. On the other hand, $|I_2| < \alpha$. In addition, the number of steps needed at each node of I_2 is $O(n+m)$. Therefore the total number of steps to enumerate all α acyclic orientations of G is $O((n+m)\alpha)$.

References

[1] V. C. Barbosa and E. Gafni, Concurrency in heavily loaded neighborhood-constrained systems, *ACM Transactions on Programming Languages and Systems* **11** (1989), 562–584.

[2] V. C. Barbosa, *Massively Parallel Models of Computation*, Ellis Horwood, Chichester, UK, 1993.

[3] D. E. Knuth, *The Art of Computer Programming 1: Fundamental Algorithms*, Addison-Wesley, Reading, Ma., 1969 (second edition 1973).

[4] R.P. Stanley, Acyclic orientations of graphs, *Discrete Mathematics* **5** (1973), 171–178.

[5] K.-P Vo, Graph colourings and acyclic orientations, *Linear and Multilinear Algebra* **22** (1987), 161–170.