The Lexicographically First Topologicai Order Problem is NLOG-Complete

Shoudai, Takayoshi
Department of Mathematics, Kyushu University

出版情報：RIFIS Technical Report. 8, 1988-12-26. Research Institute of Fundamental Information Science, Kyushu University
バージョン：
権利関係：

RIFIS-TR-CS-8

RIFIS Technical Report

The Lexicographically First Topological Order Problem is NLOG-Complete

Takayoshi Shoudai

December 26, 1988
(Revised June 30, 1989)

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan
E-mail: shoudai@rifis1.sci.kyushu-u.junet Phone: 092(641)1101 Ext.4484

THE LEXICOGRAPHICALLY FIRST TOPOLOGICAL ORDER PROBLEM IS NLOG-COMPLETE

Takayoshi SHOUDAI
Department of Mathematics, Kyushu University 33
Fukuoka 812, Japan

Abstract

We show that the lexicographically first topological order problem is NLOG-complete. The algorithm presented here uses the result that NLOG is closed under complementation.
Keywords: Computational complexity, topological sort, NLOG-complete

1. Introduction

The topological sorting problem is, given a directed acyclic graph $G = (V, E)$, to find a total ordering of its vertices such that if (v, w) is an edge then v is ordered before w. It has important applications for analyzing programs and arranging the words in the glossary [5]. Moreover, it is used in designing many efficient sequential algorithms, for example, the maximum flow problem [12].

Some techniques for sorting a directed acyclic graph topologically have been developed. The algorithm by Knuth [5] runs in time $O(|E|)$. This algorithm is essential to the lexicographically first topological order. Tarjan [12] also devised an $O(|E|)$ time algorithm by employing the depth-first search method. Dekel, Nassimi and Sahni [3] showed a parallel algorithm using the parallel matrix multiplication technique. Ruzzo also devised a simple NL^*-algorithm as is stated in [2]. Hence this problem is in NC^2. However, any completeness result does not seem to be known as to the exact complexity of topological sorting algorithms.

In this paper, we will consider the lexicographically first topological order, abbreviated to the lf topological order. Immerman [4] and Szelepcsényi [11] showed that $NLOG$ is closed under complementation. By using this result, we can give an $NLOG$-algorithm deciding this topological order. It should be noted that the "lexicographically first" property often makes some problems P-complete [1],[2],[6]-[10]. But the topological sorting with this property remains in $NLOG$.

2. Main result

Let $G = (V, E)$ be a directed acyclic graph, where $|V| = n$ and $|E| = m$, and we assume that a linear order is given to V. We use "$\prec$" to represent this order. The topological order of G is the sequence $v_1, v_2, \cdots, v_n$ of vertices such that if $i < j$ then there is no path from v_j to v_i. The *lf topological order* of G is the lexicographically first sequence with respect to $\prec$ such that it is a topological order of G. We use $u <_{lf} v$ to represent that u is ordered before v (u is located to the left of v) in the lf topological order. We consider the following problem in the same way as Reif [10].

LFTOP-ORDER

INSTANCE: A directed acyclic graph $G = (V, E)$ with a linear order on V, and two distinguished vertices s and t.

QUESTION: Is $s <_{lf} t$?

We can define the lexicographically first topological sorting algorithm as a special version of the one by Knuth [5]. The Knuth's algorithm works by repeatedly deleting a vertex of in-degree zero. It generates the lf topological order if the smallest vertex of in-degree zero is always deleted. Thus, we can say that LFTOP-ORDER is to determine the order generated by the Knuth's algorithm.

The main result in this paper is the following theorem.

Theorem 2.1. *LFTOP-ORDER is log space complete for $NLOG$.*

This theorem follows from the $NLOG$-hardness and $NLOG$-algorithm.

2.1. NLOG-hardness

For $u, v \in V$, we denote $u \rightarrow^* v$ (resp. $u \rightarrow^+ v$) if there is a path (resp. of length greater than zero) from u to v.

Lemma 2.2. *For any $u, v \in V$, if $u <_{lf} v$, then either (1) $u \rightarrow^+ v$ or (2) there exists a vertex w such that $u <_{lf} w$, $w \rightarrow^* v$ and $u \prec w$.*

Proof. Suppose $u <_{lf} v$ and $u \not\rightarrow^+ v$. Let w be the smallest vertex with respect to $<_{lf}$ such that $u <_{lf} w$, $w \rightarrow^* v$. Then $w \neq u$ and the order made by moving w just to the left of u is also a topological order. Therefore $u \prec w$ since $<_{lf}$ is the lf topological order.□

Lemma 2.3. *LFTOP-ORDER is log space hard for NLOG.*

Proof. We will give a reduction from the monotone graph accessibility problem (MGAP) to LFTOP-ORDER. It is known that MGAP is *NLOG*-complete and this is described as follows: Given a directed acyclic graph G and vertices s and t, then determine whether t is reachable from s. For an instance (G, s, t) of MGAP, we assign a linear order to the vertices so that s is the largest vertex with respect to this order. If $s \rightarrow^+ t$, s must be ordered before t in any topological order. Conversely, if $s <_{lf} t$, Lemma 2.1 implies $s \rightarrow^+ t$ since s is the largest vertex. It is clear that this reduction is computable in log space.□

2.2. NLOG-algorithm

The class *NLOG* contains the complement of MGAP (co-MGAP) since it is closed under complementation [4,11]. We use the MGAP and its complement to see reachabilities between vertices.

For $u \in V$, $R(u)$ denotes the set $\{v \in V | v \rightarrow^* u\}$. We assume that $s \not\rightarrow^+ t$, $t \not\rightarrow^+ s$ and $R(s) \cap R(t) \neq \phi$ since in other cases the order between s and t is easily determined in *NLOG*. We consider a sequence $u_1, u_2, \cdots, u_p$ of vertices defined as follows:

$$\begin{aligned} u_1 &:= \max R(s) \cap R(t), \\ u_i &:= \max\{v \in R(s) \cap R(t) | u_{i-1} \rightarrow^+ v\} \ (i \geq 2), \end{aligned}$$

where max returns the largest vertex with respect to $\prec$. The vertex u_p is defined to be the first vertex such that $\{v \in R(s) \cap R(t) | u_p \rightarrow^+ v\} = \phi$.

Lemma 2.4. *The procedure Lftop_Order solves LFTOP-ORDER.*

```
procedure Lftop_Order(s,t)
    begin
        U := V;
        while true do
            begin
                v_s := max(R(s) − R(t)) ∩ U;
                v_t := max(R(t) − R(s)) ∩ U;
                if R(s) ∩ R(t) ∩ U ≠ φ then
                    begin
                        u := max R(s) ∩ R(t) ∩ U;
                        U := {v ∈ V | u →+ v};
                        if u ≺ v_s or u ≺ v_t then break
                    end
                else
                    break
            end;
        if v_s ≺ v_t then accept else reject
    end.
```

Proof. After the kth iteration of the **while** loop, the variable u is set to u_k. We use $v_{s,k}$, $v_{t,k}$ to denote the values v_s, v_t just after the kth iteration. After at most $p+1$ iterations, the **while** loop breaks.

Fact 1. For any $v \in R(s) \cap R(t)$, if $u_k <_{lf} v$, then $u_k \rightarrow^+ v$ $(1 \leq k \leq p)$.

The proof is by induction on k. In the case of $k = 1$, the proof is similar to the case $k \geq 2$ excluding the induction step. Suppose $k \geq 2$. Unless $u_k \rightarrow^+ v$, by Lemma 2.2 there exists w such that $u_k <_{lf} w$, $w \rightarrow^* v$ and $u_k \prec w$. This implies $u_{k-1} <_{lf} w$ and $w \in R(s) \cap R(t)$. Thus, by the induction hypothesis, $u_{k-1} \rightarrow^+ w$. This contradicts the definition of u_k.

Note that Fact 1 implies that if $u_p <_{lf} v$ then $v \notin R(s) \cap R(t)$. The following fact can also be shown in a similar way.

Fact 2. Suppose that the kth iteration of **while** loop does not break ($1 \leq k \leq p$). For any $v \in (R(s) - R(t)) \cup (R(t) - R(s))$, if $u_k <_{lf} v$, then $u_k \rightarrow^+ v$.

Fact 3. Suppose that the kth iteration of **while** loop breaks ($1 \leq k \leq p+1$). If $v_{s,k} \prec v_{t,k}$, then

$$\max\{v \in R(s) - R(t) | u_p <_{lf} v\} \preceq v_{s,k},$$
$$\max\{v \in R(t) - R(s) | u_p <_{lf} v\} = v_{t,k}.$$

We only give a proof for the case of $2 \leq k \leq p$. Since the $(k-1)$st iteration does not break, $v_{s,k} = \max\{v \in R(s) - R(t) | u_{k-1} <_{lf} v\}$ by Fact 2. Therefore, $\max\{v \in R(s) - R(t) | u_p <_{lf} v\} \preceq v_{s,k}$. In a similar way, $\max\{v \in R(t) - R(s) | u_p <_{lf} v\} \preceq v_{t,k}$. If $v_{t,k} <_{lf} u_p$, there exists w such that $v_{t,k} <_{lf} w$, $w \rightarrow^* u_p$ and $v_{t,k} \prec w$ since $v_{t,k} \not\rightarrow^* u_p$. Then, $u_{k-1} \rightarrow^+ v_{t,k}$ implies $u_{k-1} <_{lf} w$. Since $w \in R(s) \cap R(t)$, by Fact 1, $u_{k-1} \rightarrow^+ w$. This contradicts the definition of u_k since $u_k \prec v_{t,k} \prec w$.

Suppose that the kth iteration of **while** loop breaks and $v_{s,k} \prec v_{t,k}$ at that time. Then, by Fact 3, $u_p <_{lf} v_{t,k}$. If $v_{t,k} <_{lf} s$, there exists w such that $v_{t,k} <_{lf} w$, $w \rightarrow^* s$ and $v_{t,k} \prec w$. Then $w \in R(s) - R(t)$ by $u_p <_{lf} w$. Thus, $w \preceq v_{s,k}$ by Fact 3. It contradicts the assumption $v_{s,k} \prec v_{t,k}$. Therefore, $s <_{lf} v_{t,k}$. This implies $s <_{lf} t$. In a similar way, if $v_{t,k} \prec v_{s,k}$, then $t <_{lf} s$.□

This implies the following lemma.

Lemma 2.5. *LFTOP-ORDER is in NLOG.*

Proof. It is not difficult to see that the procedure Lftop_Order can be simulated in *NLOG* since the reachability between vertices is computed in *NLOG* by enumerating all vertices and employing the MGAP and co-MGAP. □

3. Concluding Remarks

The following topological sorting algorithms are also known.

Tarjan [12]: We assume an adjacency list which is ordered lexicographically. The depth-first search is executed to make a forest. Then we order the vertices in decreasing order as they are popped up.

Dekel, Nassimi and Sahni [3]: We assume an adjacency matrix. By matrix multiplications, we can compute the length of the longest path from any vertex with no predecessors to each vertex. Then vertices are sorted in nondecreasing order of their lengths.

Ruzzo (Cook [2]): We compute the transitive closure of an adjacency matrix. This gives the number of predecessors of each vertex by summing the columns. Then we sort vertices by these numbers.

These algorithms generate distinct topological sorted orders and each of them is different from the lf topological order.

Generally the problem TOP-ORDER(A) is defined as follows: Given a directed acyclic graph $G = (V, E)$ and two distinguished vertices s and t, determine whether s is ordered before t in the order generated by a given topological sorting algorithm A. We can show that the problems for the above three algorithms are all $NLOG$-complete. The proofs are similar to that of $NLOG$-completeness of LFTOP-ORDER. But it is not known whether all possible TOP-ORDER(A) problems are at least $NLOG$-hard. It is an interesting open question to determine the inherent complexity of topological sorting.

References

[1] R. Anderson and E.W. Mayr, Parallelism and the maximal path problem, *Inform. Process. Lett.* **24** (1987) 121-126.

[2] S.A. Cook, A taxonomy of problems with fast parallel algorithms, *Inform. Contr.* **64** (1985) 2-22.

[3] E. Dekel, D. Nassimi and S. Sahni, Parallel matrix and graph algorithms, *SIAM J. Comput.* **10** (4) (1981) 657-675.

[4] N. Immerman, Nondeterministic space is closed under complementation, *SIAM J. Comput.* **17** (5) (1988) 935-938.

[5] D. Knuth, "The Art of Computer Programming," Vol.1, Addison-Wesley, Reading, Mass. (1968).

[6] M. Luby, A simple parallel algorithm for the maximal independent set problem, *Proc. 17th ACM Symp. Theory of Comput.* (1985) 1-10.

[7] S. Miyano, The lexicographically first maximal subgraph problems: P-completeness and NC algorithms, *Proc. 13th ICALP* (Lecture Notes in Computer Science **267**) (1987) 425-434.

[8] S. Miyano. A parallelizable lexicographically first maximal edge-induced subgraph problem, *Inform. Process. Lett.* **27** (1988) 75-78.

[9] S. Miyano. Δ_2^p-complete lexicographically first maximal subgraph problems, *Proc. 13th MFCS* (Lecture Notes in Computer Science **317**) (1988) 454-462.

[10] J.H. Reif, Depth-first search is inherently sequential, *Inform. Process. Lett.* **20** (1985) 229-234.

[11] R. Szelepcsényi, The method of forcing for nondeterministic automata, *Bulletin of the EATCS* **33** (1987) 96-100.

[12] R.E. Tarjan, "Data Structures and Network Algorithms," CBMS-NSF Regional Conference Series in Applied Mathematics **44**, SIAM, Philadelphia, Pennsylvania (1983).