Information Processing Letters 112 (2012) 823–828

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Topological ordering algorithm for LDAG

GuiPing Wang [a,*], ShuYu Chen [b], XiaoQin Zhang [a], Zhen Zhou [a]

[a] College of Computer Science, Chongqing University, Chongqing 400044, China
[b] College of Software Engineering, Chongqing University, Chongqing 400044, China

ARTICLE INFO

Article history:
Received 5 January 2012
Received in revised form 10 July 2012
Accepted 26 July 2012
Available online 1 August 2012
Communicated by Jinhui Xu

Keywords:
DAG
Topological order
Level
LDAG
Computational complexity

ABSTRACT

Directed Acyclic Graph (DAG) is an important tool for workflow modeling and data provenance management. In these applications, DAG usually performs well. Yet for some workflow applications, except data or control dependencies between atomic tasks, there exists another requirement that each atomic task should be accomplished at an expected stage. Therefore, this paper proposes an improved DAG model – LDAG, in which each vertex has a level. Three cases of the level of vertices are discussed. For a reasonable one of these cases, this paper proposes a topological ordering algorithm and proves its correctness. In addition, it discusses the complexity of the algorithm and some other relevant problems.

1. Introduction

Directed Acyclic Graph (DAG) is widely used in various areas, such as workflow modeling [1], data provenance management [2], etc. An important operation on a DAG is topological ordering, which can be used to gain a topological order of all vertices and judge whether there exist directed circuits.

A topological order T of a given DAG $D = (V, A)$ is a linear order of its all vertices in which for all directed paths from vertex x to $y (x \neq y)$, $T(x) < T(y)$ holds.

Topological ordering can be implemented by two means:

1) DFS-based means: utilizing DFS algorithm to compute finishing times $f[v]$ for each vertex v; as each vertex is finished, inserting it into the front of a linked list.
2) BFS-analogous means: at each step, choosing a vertex v without incoming arc, deleting v and its all outgoing arcs, and then inserting v into the tail of a linked list.

For DFS-based means, some well-known algorithms are proposed, which compute the topological order of a DAG in $O(m+n)$ time [3–5] ($n = |V|$ and $m = |A|$). A detailed description of the algorithms can be found in [6].

For BFS-analogous means, a proposition and a corresponding algorithm can be found in [7]. Similarly, the algorithm can be performed in $O(m+n)$ time.

Several variants of topological ordering problem are proposed. For example, an online variant of this problem, which is called online incremental topological ordering, is discussed in [8,9]. In this online variant, the arcs of the DAG are unknown in advance but are given one at a time. An $O(n^{2.75})$ time algorithm is proposed in [9], which is independent of the number m of arcs inserted.

In most situations of workflow modeling, DAG performs well. Yet for some workflow applications, except data or control dependencies between atomic tasks, there exists another requirement that each atomic task should be accomplished at an expected stage. For example, due to insufficiency of computing resources, a large and sophisticated job has to be executed in several stages. Another situation is that the job inherently contains several stages. In these situations, the workflow has to be partitioned into several stages, and each task corresponds to a given stage called level.

* Corresponding author.
E-mail address: w_guiping@163.com (G.P. Wang).

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2012.07.015

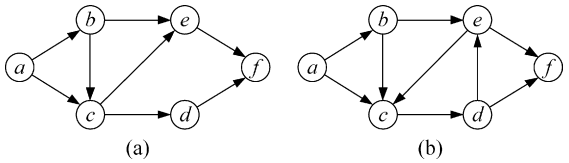

Fig. 1. Two digraphs.

For these situations, in this paper, we propose an improved DAG model – LDAG. In LDAG, each atomic task has a level. The level of an atomic task may be an inherent stage of the whole job or an expected stage to be executed. We analyse three cases of the level of atomic tasks. For a reasonable one of these cases, we propose a topological ordering algorithm based on BFS-analogous algorithm [7]. The algorithm consists of two phases, namely *Level Adjusting* and *Topological Ordering*. We prove the correctness of the algorithm.

The remainder of this paper is organized as follows. Section 2 introduces formal definitions and preliminaries. Section 3 presents LDAG model, and discusses three cases of the level of tasks. In Section 4, we propose a topological ordering algorithm for a reasonable case of LDAG.

2. DAG model and preliminaries

For clear illustration, we introduce formal definitions relevant to DAG and topological order in this section.

2.1. DAG as a model for workflow

Definition 2.1 *(Directed graph or digraph).* A directed graph D consists of a non-empty finite set of elements called vertices (or nodes) and a finite set of ordered pairs of distinct vertices called arcs. Usually, $D = (V, A)$ or $D(V, A)$ represents a digraph, while $V(D)$ and $A(D)$ represent vertex set and arc set respectively.

For example, in the digraph shown in Fig. 1(a), $V = \{a, b, c, d, e, f\}$, $A = \{\langle a, b\rangle, \langle a, c\rangle, \langle b, c\rangle, \langle b, e\rangle, \langle c, d\rangle, \langle c, e\rangle, \langle d, f\rangle, \langle e, f\rangle\}$.

Definition 2.2 *(Directed circuit).* In a given digraph, if a directed path, $uu_1u_2 \ldots u_nv$, starts from and ends in a same vertex, that is, $u = v$, then this path is called a directed circuit.

Definition 2.3 *(DAG).* A digraph without directed circuit is called Directed Acyclic Graph or DAG for short.

Based on this definition, for the two diagraphs in Fig. 1, Fig. 1(a) is a DAG, while Fig. 1(b) is not since *cdec* is a directed circuit.

Due to data or control dependencies between atomic tasks, and the requirement for no directed circuit, DAG is a natural tool for workflow modeling, in which vertices represent atomic tasks and directed arcs represent data or control dependencies between tasks. The terms *vertex*, *task* and *atomic task* are interchangeable in the remainder of this paper.

Therefore, a workflow job can be represented by a DAG $D(V, A)$, where $V = \{t_1, t_2, \ldots, t_n\}$ is the set of atomic tasks (n is the total number of tasks), and A is the set of arcs indicating the dependency and precedence constraints between tasks.

Definition 2.4 *(Predecessor and successor).* In a DAG $D(V, A)$, if $\langle u, v\rangle$ is an arc in A, then the task u is a direct predecessor of task v, and v is a direct successor of u. If there exists a directed path from u to v, such as $uu_1u_2 \ldots u_nv$, then u is a predecessor of v, and v is a successor of u.

Definition 2.5 *(Task strictness).* In a DAG, each task is strict with respect to both its predecessors and successors. In other words, a task can start to be executed only when all of its predecessor tasks have been accomplished.

After modeling workflow with a DAG, the foremost work is to judge whether there exist directed circuits. For a DAG workflow without circuits, the tasks can be scheduled according to their topological order. Otherwise, they cannot be scheduled and executed.

2.2. Topological order

Definition 2.6 *(Topological order).* A topological order of a DAG is a linear order of all vertices that satisfy all the predecessor and successor relations in the DAG. Another definition has been shown in Section 1. The topological ordering operation on a DAG is to arrange all vertices into a topological order.

For example, one of the topological orders of Fig. 1(a) is *abcedf*, which satisfies the condition that an arc always leads from an anterior vertex to a posterior one in this order. *abcdef* is another topological order, while *acbdef* is not because the precedence relation of $\langle b, c\rangle$ is not satisfied.

3. LDAG: each vertex has a level

In some real applications, each atomic task corresponds to a given stage, i.e. level. We propose an improved model, LDAG, for the modeling of these workflow applications. We then discuss three cases of the level of vertices.

Definition 3.1 *(Level).* In some DAG workflows, each vertex has a weight called level. The level of a task may be an inherent stage of the task in a whole job or an expected stage to be executed.

Definition 3.2 *(LDAG).* In a DAG workflow, if each vertex has a level, the model is called LDAG in this paper.

For example, a LDAG is shown in Fig. 2, the number beside each vertex represents its level.

LDAG can be widely used in workflow modeling. Consequently, it is important to study this model and some relevant problems, such as topological ordering, scheduling and optimizing. In this paper, we focus on the first problem.