On the time complexity of minimum and maximum global snapshot problems

Loon-Been Chen¹, I-Chen Wu *

Department of Computer Science and Information Engineering, National Chiao Tung University, 1001 Ta-Hsueh Road, Hsinchu, Taiwan

Received 1 June 1997; received in revised form 28 April 1998
Communicated by T. Asano

Abstract

Deriving the minimum and maximum global snapshots is very useful for some error detection problems in distributed programs. Several researchers, e.g., Groselj, Chen and Wu, have shown that the minimum and maximum global snapshot problems are linear-time reducible to the maximum constant-ratio network flow (MCNF) problem, here defined as the well-known maximum network flow problem with \( m = \Theta(n) \), where \( m \) is the number of edges and \( n \) is the number of vertices in the given flow network. In this paper we show in a reverse way that the MCNF problem is also linear-time reducible to these global snapshot problems. Thus, we can conclude that the global snapshot problems are "as difficult as" the MCNF problem in terms of time complexity. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Distributed systems; Computational complexity; Error detection; Global snapshot

1. Introduction

Error detection and debugging are critical tasks in developing distributed programs. Many research results [3,4,8] suggest that a distributed program usually adheres to certain invariant conditions to function properly. For example, in a distributed program, several tokens may be distributed over the processors (e.g., the token may represent the number of resources and critical sections), and the number of these tokens is bounded in a range at any snapshot, regardless of how tokens are moved over different processors. The conventional means of detecting whether or not the above condition holds are to derive the minimum or maximum number of tokens for all possible snapshots.

This problem is referred to as the minimum or maximum global snapshot problem [6].

Groselj [6] recently proposed an interesting method to derive the minimum global snapshot. In this method, he reduced this snapshot problem to a maximum constant-ratio network flow (MCNF) problem. The MCNF problem is defined herein as the well-known maximum network flow problem with \( m = \Theta(n) \), where \( m \) is the number of edges and \( n \) is the number of vertices in the given flow network. Chase and Garg [1] independently obtained similar results. Later, Chen and Wu [2] proposed a general technique, called normalization, for deriving both minimum and maximum global snapshots. According to their results, these snapshot problems are linear-time reducible to the MCNF problem.

Above investigations only point out that the time complexities of the minimum and maximum global
snapshot problems will not be higher than those of the MCNF problem. However, whether or not the time complexities for the above snapshot problems can be lower than those of the MCNF problem remains unknown. To resolve this question, this paper shows in a reverse manner that the MCNF problem is also linear-time reducible to these global snapshot problems. Thus, we can conclude that the above global snapshot problems are "as difficult as" the MCNF problem in terms of time complexity.

The rest of this paper is organized as follows. Section 2 describes our model and the notations used in this paper. Sections 3 and 4 discuss the time complexities of the minimum and maximum global snapshot problems, respectively. Concluding remarks are finally made in Section 5.

2. Model and notations

A distributed program consists of processes communicating via a network. These processes share no memory and no global clock. Pairs of processes must communicate via network channels. In addition, the state of such a program is distributed over these processes and channels in each snapshot.

Events

The states of processes change only when events [7] (atomic actions) are executed. Three kinds of events on each process are of relevant concerned:

1. **internal event**: performs a local computation;
2. **send event**: sends a message to another process via the channel; and
3. **receive event**: receives a message from another process via the channel.

Note that each process should start with an initial internal event and end with a final internal event.

Next, the chronological order of events is defined by assuming that the event $e_i$ happens before event $e_j$, denoted by $e_i \rightarrow e_j$, if and only if one of the following conditions holds [7]:

1. events $e_i$ and $e_j$ occur in the same process and $e_i$ occurs before $e_j$;
2. event $e_i$ is the sent event of a message and event $e_j$ is the receive event of the same message; and
3. another event $e_k$ exists such that $e_i \rightarrow e_k$ and $e_k \rightarrow e_j$.

Global snapshot

Consider a possible run for a distributed program. The system can proceed from one state to another by executing events. A set $E_C$ of events are said to be consistent if event $e \in E_C$ and event $e' \rightarrow e$ imply that $e' \in E_C$. A global snapshot is a collection of states, one from each process and channel, immediately after executing a consistent set of events. Herein, of relevant concern is the total number of tokens. Therefore, a state value can actually be represented by its token number. In addition, the minimum (maximum) global snapshot is the global snapshot with the minimum (maximum) number of tokens among all possible snapshots.

Event graph

An event graph is used to represent a run for a distributed program under the following conditions:

- (1) a vertex denotes an event, and
- (2) if event $e_i \rightarrow e_j$ and no event $e_k$ exists such that $e_i \rightarrow e_k$ and $e_k \rightarrow e_j$, there is a corresponding arc, denoted by $(e_i, e_j)$;

For example, Fig. 1 illustrates an event graph. An arc $(e_i, e_j)$ is referred to as a message arc if it corresponds to an in-transit message from event $e_i$ to $e_j$. Otherwise, the arc is referred to as an internal arc because it corresponds to an internal state transition inside a process. For each message arc $a$, $S_a$ denotes the number of message tokens; for each internal arc $a = (e_i, e_j)$, $S_a$ denotes the token number of the corresponding process after execution of the event $e_i$ and before the event $e_j$. In addition,
the vertex corresponding to an initial (final) internal event is called an \textit{i-vertex} (\textit{f-vertex}). Moreover, the arc incident to \textit{i-vertex} (\textit{f-vertex}) is called an \textit{i-arc} (\textit{f-arc}). Clearly, each process has an \textit{internal path} from its \textit{i-vertex} to its \textit{f-vertex} without going through any message arcs.

\textbf{Cuts}

A cut in an event graph \(H\) partitions the vertex set \(U\) into two disjoint sets such that one, called the \textit{source part} and denoted by \(U_s\), contains all the \textit{i-vertices}. Meanwhile, the other, called the \textit{sink part} and denoted by \(U_f\), contains all the \textit{f-vertices}. The cost of a cut \(C\) is

\[
\text{cost}(C) = \sum_{\forall u: u=(u, v), u \in U_s, v \in U_f} S_a.
\]

For example, in Fig. 1, the costs of cuts \(C_1\), \(C_2\) and \(C_3\) are 14, 12 and 16, respectively.

A cut of the event graph is \textit{consistent} if and only if the set of all events in the source part are consistent. From this definition, we can infer that each consistent cut corresponds to a global snapshot and the cut cost is actually the number of tokens of the global snapshot. Obviously, for each consistent cut \(C\), each arc in \(C\) must be from the source part to the sink part.

The minimum (maximum) consistent cut is the consistent cut with the least (largest) cost among all consistent cuts. Also, the minimum (maximum) consistent cut cost is the cost of the minimum (maximum) consistent cut. Clearly, the minimum (maximum) consistent cut corresponds to the minimum (maximum) global snapshot.

\textit{Global snapshot model}

In the proposed model, the event graph with arc costs is assumed to be given in advance. Garg and Waldecker [4] suggested that in practice, for each event in each process \(P_i\), \(P_i\) sends its token number to a process, called the \textit{checker process}. This process runs an algorithm for deriving the minimum or maximum global snapshot.

\section{Minimum global snapshot}

This section shows that the MCNF problem is linear-time reducible to the minimum global snapshot problem. Given a flow network \(N\) (see Definition 1 below), the linear-time reduction algorithm given below constructs an event graph \(H\) such that the min-cut capacity of \(N\) equals the minimum consistent cut cost of \(H\). Fig. 2 presents an example of the reduction.

\textbf{Definition 1.} A flow network \(N = (V, E)\) is a directed graph in which each edge \((u, v) \in E\) has a non-negative capacity \(c(u, v) \geq 0\). One node \(s\) is designated as the source and another node \(t\) is designated as the sink. A cut is a set of arcs all incident to two disjoint vertex sets partitioned from \(V\), where one set
with node $s$ is referred to as the source set, and the other with node $t$ is referred to as the sink set. The capacity of a cut $C$, denoted by $\text{capacity}(C)$, is the total capacity of all arcs (on $C$) from the source set to the sink set. A minimum cut of the flow network is the cut with the least capacity. The least capacity is also called the min-cut capacity.

**Reduction Algorithm.**
1. For each vertex $v$ in $N$, create the corresponding internal path $P_v$. This operation takes $\Theta(n)$ time.
2. For each pair of vertices $(u, v)$, if $(u, v)$ or $(v, u)$ is in $N$, create the corresponding internal path $P_{(u,v)}$ (or, equivalently, $P_{(v,u)}$). This operation takes $O(m)$ time.
3. For each internal path, add two internal events such that the path is divided into three parts: the i-arc, the middle part, and the f-arc. All message arcs added below must have their vertices incident to the middle parts. This operation takes $O(m + n)$ time.
4. For each internal path $P_{(u,v)}$, create a message arc $a_{(u,v)}$ from $P_{(u,v)}$ to $P_u$ and a message arc $a_{(v,u)}$ from $P_{(u,v)}$ to $P_v$. This operation takes $\Theta(m)$ time.
5. Set the cost of f-arc of $P_s$ and i-arc of $P_t$ to $\infty$, where $s$ is the source and $t$ is the sink of $N$. Set the costs of other i-arcs and f-arcs to 0. Set the costs of all other internal arcs (in the middle parts) to $\infty$. This operation takes $\Theta(m + n)$ time.
6. For each message arc $a_{(u,v)}$, if $(u, v)$ is in $N$, set $S_{a_{(u,v)}} = c(u, v)$; otherwise, set $S_{a_{(u,v)}} = 0$. This operation takes $\Theta(m)$ time.

In the above reduction algorithm, each consistent cut $C_H$ in $H$ is said to correspond to a cut $C_N$ in $N$, denoted by $C_H \leftrightarrow C_N$, if and only if the following property holds:

**A1.** For each vertex $v$ in $N$, $v$ is in the source (sink) part of cut $C_N$ if and only if $C_H$ cuts across the i-arc (f-arc) of the internal path $P_v$.

For example, in Fig. 2, $C_H \leftrightarrow C_N$. Note that since the source $s$ (the sink $t$) is always in the source (sink) part, $C_H$ must always cut across the i-arc of $P_s$ (the f-arc of $P_t$). Theorem 2 proves that the minimum consistent cut cost of $H$ equals the min-cut capacity in $N$.

**Theorem 2.** For a given flow network, its min-cut capacity equals the minimum consistent cut cost in the event graph constructed from the above reduction algorithm.

**Proof.** Let $N$ be the given flow network and $H$ be the reduced event graph. In addition, let $K_H$ be the set of all the consistent cuts in $H$, $K_{H\mapsto N}$ be the set of all cuts in $K_H$, that correspond to some cut in $N$, and $\bar{K}_{H\mapsto N}$ be $K_H - K_{H\mapsto N}$. In this proof, it obviously suffices to prove that the following two properties hold:

**B1.** For each consistent cut $C_H$ in $\bar{K}_{H\mapsto N}$, $\text{cost}(C_H) = \infty$.

**B2.** Among all consistent cuts $C_H$ in $K_{H\mapsto N}$,

$$\min_{C_H \in K_{H\mapsto N}} \text{cost}(C_H) = \min_{C_N \in N} \text{capacity}(C_N).$$

This holds if the following two properties hold:

**B21.** For each cut $C_N$, there exists at least one $C_H$ in $K_{H\mapsto N}$ such that $C_H \leftrightarrow C_N$.

**B22.** For each cut $C_N$,

$$\text{capacity}(C_N) = \min_{C_H \in K_{H\mapsto N}} \text{cost}(C_H).$$

Properties B21 and B22 imply Property B2 owing to the following reason:

$$\min_{C_N} \left( \min_{C_H \in K_{H\mapsto N}} \text{cost}(C_H) \right) = \min_{C_H \in K_{H\mapsto N}} \left( \min_{C_N} \text{cost}(C_H) \right).$$

Initially, Property B1 must be proven. From Property A1, for each consistent cut in $K_{H\mapsto N}$, the cut must cut across i-arcs or f-arcs except for the f-arc of $P_s$ and the i-arc of $P_t$ (s and t are the source and sink vertices of $N$). On the other hand, for each consistent cut $C_H$ in $\bar{K}_{H\mapsto N}$, the cut $C_H$ must cut across some internal arc in the middle parts or the f-arc of $P_s$ or the i-arc of $P_t$. Since these arcs have costs $\infty$ (see the Reduction Algorithm), we obtain $\text{cost}(C_H) = \infty$.

Second, both Properties B21 and B22 must be proven. Given a cut $C_N$ that partitions the vertices
of $N$ into $V_s$ and $V_t$, a consistent cut $C_H$ (in $H$) can be constructed, having the minimum cost among all consistent cuts in $K_{H \rightarrow N}$, as follows. First, to let $C_H \mapsto C_N$, $C_H$ must cut across each $P_u$ as in Property A1. Then, we need to verify the consistency and examine the minimum cost among each set of $P_u$, $P_v$, and $P_{(u,v)}$:

1. If $C_H$ cuts across i-arc of both $P_u$ and $P_v$ ($u \in V_s$ and $v \in V_s$), as shown in Fig. 3(a), a consistent cut in $K_{H \rightarrow N}$ can cut across either i-arc or f-arc of $P_{(u,v)}$. We let $C_H$ cut across the i-arc because the arc will have a smaller cost. In this case, no edge contributes cost to $C_N$ and no message arc contributes cost to $C_H$.

2. If $C_H$ cuts across i-arc of $P_u$ and f-arc of $P_v$ ($u \in V_s$ and $v \in V_t$), as shown in Fig. 3(b), $C_H$ must cut across f-arc of $P_{(u,v)}$. Otherwise, the cut will be inconsistent. In this case, edge $(u, v)$ contributes $c(u, v)$ to $C_N$. Corresponding message arc $a_{(u,v)}$ contributes $S_{a_{(u,v)}}$ to $C_H$.

3. If $C_H$ cuts across f-arc of $P_u$ and i-arc of $P_v$ ($u \in V_t$ and $v \in V_s$), $C_H$ must cut across f-arc of $P_{(u,v)}$. This case is similar to case (2).

4. If $C_H$ cuts across f-arc of both $P_u$ and $P_v$ ($u \in V_t$ and $v \in V_t$), as shown in Fig. 3(d), $C_H$ must cut across f-arc of $P_{(u,v)}$. In this case, no edge contributes cost to $C_N$ and no message arc contributes cost to $C_H$.

From above discussion, cut $C_H$ is consistent due to the consistency among each set of $P_{(u,v)}$, $P_u$, and $P_v$. In addition, it is obvious from above that

$$
capacity(C_N) = cost(C_H) = \min_{C_H, C_H' \mapsto C_N} \left( cost(C_H') \right).
$$

Thus, both Properties B21 and B22 hold.

4. Maximum global snapshot

Herein, both minimum and maximum global snapshot problems can be reduced to each other in the following manner: (1) based on the normalization technique [2], reset the cost of each message arc to zero, while not changing any consistent cut cost; and (2) change the cost $S$ of each internal arc to $M - S$, where $M$ is the maximum cost among all arcs. Thus, we can conclude that the maximum global snapshot problem is also "as difficult as" the maximum network flow problem in terms of time complexity.

5. Conclusion

From the investigations of Chase and Garg [1], Chen and Wu [2], and Groselj [6], the minimum and maximum global snapshot problems are linear-time reducible to the MCNF problem. In this paper, we show in a reverse manner that the MCNF problem is also linear-time reducible to these global snapshot problems. Thus, we can conclude that the global snapshot problems are "as difficult as" the MCNF problem in terms of time complexity. The fact that
$O(n^2 \log n)$ has been the optimal time complexity for the MCNF problem for many years [5] implies the difficulty in improving the $O(n^2 \log n)$ global snapshot algorithms (as well as that for the MCNF problem).

Acknowledgments

The authors would like to thank the National Science Council of the Republic of China for financially supporting this research under Contract No. NSC-86-2213-E-009-32.

References


