DECIDING 1-SOLVABILITY OF DISTRIBUTED TASK IS NP-HARD

(Extended Abstract)

Ofer Biran, Shlomo Moran and Shmuel Zaks

Department of Computer Science
Technion, Haifa, Israel 32000

ABSTRACT

We present a use of graph-theoretic concepts in fault-tolerant distributed systems.

The question of which distributed tasks can be solved by asynchronous protocols in the presence of crash failures has been investigated in recent years. Some of these studies investigate the solvability of specific tasks, providing both positive and negative results.

Given a distributed task, we associate with it certain graphs, that enable us to determine whether this task can be solved in the presence of one faulty processor ([BMZ]), and in this paper we further use this approach to study the following problem: Given a task $T$ for $N$ processors, how difficult is it to determine whether $T$ can be solved in the presence of one faulty processor? We show that this problem is NP-hard for any fixed $N \geq 3$.

---

1 This research was supported in part by Technion V.P.R. Funds - Wellner Research Fund and Loewengart Research Fund, by the I. Goldberg Fund for Electronics Research, and by the Foundation for Research in Electronics, Computers and Communications, administrated by the Israel Academy of Sciences and Humanities.
1. INTRODUCTION

We present a use of graph-theoretic concepts in fault-tolerant distributed systems.

The question of which distributed tasks can be solved by asynchronous protocols in the presence of crash failures has been widely investigated in the recent years. In [FLP] it was shown that the consensus task cannot be solved in the presence of one fail-stop failure in the asynchronous model. In [DLPSW] it was shown that the approximate consensus is solvable in the presence of multiple such failures, and in [ABDKPR] two versions of the renaming task were studied. In [MW] a class of tasks were proven not to be solvable when one processor is faulty. A complete characterization of the tasks which are solvable in the presence of one faulty processor (1-solvable) was given in [BMZ]. This characterization used certain graphs associated with a given task, and by examining the properties of these graphs it is determined whether the task can be solved in the presence of one faulty processor; in case of a positive answer, an algorithm which use traversing of these graphs was given. Thus, determining whether a task is 1-solvable is reduced to testing certain combinatorial properties. Verifying whether these combinatorial properties hold for each of the tasks studied in the references above is relatively straightforward. This raises the question of whether there is an efficient algorithm that receives the specification of a distributed task for \( N \) processors as an input, and determines whether this task is 1-solvable. In this paper we show that this problem is NP-hard, and hence that no such algorithm exists unless P=NP. The problem remains NP-hard even when restricted to tasks for \( N \) processors, where \( N \) is any fixed integer \( \geq 3 \).

In the proof we consider only finite tasks, which are tasks with finitely many possible inputs and outputs (the consensus task for a fixed number of processors is finite, while the approximate consensus is not). A specification of a finite tasks \( T \) is given by a (finite) list of legal input vectors, and a (finite) list \( T(\bar{x}) \) of legal decision vectors for each input vector \( \bar{x} \). In fact, deciding 1-solvability for such tasks is shown to be NP-complete.

The NP-hardness proof is based on polynomial time reduction from the following problem:

**Input:** A \( k \)-partite graph \( H_k \), together with a partition of the vertices of \( H_k \) into \( k \) independent sets.

**Property:** \( H_k \) contains a clique of size \( k \) (in short, a \( k \)-size clique).

This problem is NP-complete by the original reduction from the Satisfiability problem to the Maximum Clique problem in [K].