International Journal of Approximate Reasoning 56 (2015) 87–107

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Scheduling results applicable to decision-theoretic troubleshooting ☆

Václav Lín [a,b,*]

[a] Institute of Information Theory and Automation of the AS CR Prague, Czech Republic
[b] Faculty of Management, Prague University of Economics, Czech Republic

ARTICLE INFO

Article history:
Received 9 April 2014
Received in revised form 7 August 2014
Accepted 12 August 2014
Available online 19 August 2014

Keywords:
Decision-theoretic troubleshooting
Single machine scheduling with weighted flowtime
Algorithms
Computational complexity

ABSTRACT

In decision-theoretic troubleshooting, we are given a Bayesian network model of a malfunctioning device and our task is to find a repair strategy with minimal expected cost. The troubleshooting problem has received considerable attention over the past two decades. We show that several troubleshooting scenarios proposed in the literature are equivalent to well-studied machine scheduling problems. This immediately yields new complexity-theoretic and algorithmic results for troubleshooting. We also apply scheduling results to multi-agent troubleshooting. Further, we examine the so-called *call service action* which is often used in troubleshooting but has no natural counterpart in machine scheduling. We show that adding the *call service* action to basic troubleshooting models does not make the problem intractable.

> Nothing under the sun is new, neither is any man able to say: Behold this is new: for it hath already gone before in the ages that were before us.
>
> *Holy Bible, Ecclesiastes 1:10* (Douay-Rheims transl.)

1. Introduction

Sequencing repair and diagnostic actions in order to fix a system at the lowest expected cost is a very natural problem that has been studied since the advent of computing in 1950s. The oldest formulation known to the author is due to Johnson [22]:

> "A problem of importance to the Air Force is that of *troubleshooting* to find a malfunctioning part of a complex piece of electronic equipment. [...] A complicated machine may break down because of the failure of some of its components. In what sequence should its components be tested and repaired in order to minimize the expected delay time?"

Similarly in *decision-theoretic troubleshooting* [19], a malfunctioning man-made system described by a Bayesian network is to be repaired by a sequence of troubleshooting steps designed to identify and eliminate the cause of malfunction at

☆ This work was supported by the Czech Science Foundation through Grant 13-20012S, and by the Institutional Research Support of the Faculty of Management, Prague University of Economics. A preliminary version of this paper has appeared in the Proceedings of ECSQARU 2011 [30], and some of the presented material has evolved from [31,32].

* Correspondence to: Institute of Information Theory and Automation of the AS CR, Pod Vodárenskou věží 4, CZ-182 08, Prague, Czech Republic.

http://dx.doi.org/10.1016/j.ijar.2014.08.004
0888-613X/

minimal expected cost. This research area has been drawing interest from 1990s to recent years [43,56,40]. The purpose of this paper is to show equivalence of certain machine scheduling problems to the combinatorial problems that lie at heart of several recently studied troubleshooting scenarios.

The relation of automated diagnosis and machine scheduling problems has been noted before, for example by Kadane and Simon [23], Monma and Sidney [37] and recently by Nobibon et al. [39]. However, it seems to have been overlooked in the newer literature on troubleshooting.

Contribution of the paper. The presented results enhance our understanding of the borderline between troubleshooting problems solvable in polynomial time and $\mathcal{NP}$-hard troubleshooting problems. We study several troubleshooting scenarios proposed by Jensen et al. [21], Langseth and Jensen [27], Ottosen and Jensen [42], and we identify equivalent scheduling problems. Since machine scheduling has been studied intensively for decades (see the surveys by Lawler et al. [29], Brucker [7]), we get strong results pertaining to troubleshooting almost "for free". Thus, the paper helps troubleshooting researchers to find relevant algorithms and complexity-theoretic results in the vast scheduling literature.

Troubleshooting models often contain the so-called *call-service action* [19,21], or a penalty for not fixing the fault. Call-service action does not have a natural counterpart in machine scheduling. It has been conjectured that using this kind of action increases computational complexity of troubleshooting [21]. Somewhat surprisingly, we prove that in basic troubleshooting scenarios, this is not true. We shall see that presence of the call-service action in troubleshooting models does not imply intractability – a positive result.

Areas not covered. First results concerning complexity of troubleshooting were given by Vomlelová and Vomlel [54] and Vomlelová [53] who have identified several sources of complexity that render troubleshooting problems $\mathcal{NP}$-hard ("dependent actions", "dependent faults" and other). This paper complements their work in focusing on other sources of complexity ("precedence constraints", "call service", and other discussed below) that have appeared in troubleshooting literature. We always study a simplistic troubleshooting scenario extended with a single source of complexity. This enables us to get valid results but it also poses a limitation since:

- realistic troubleshooting problems may contain several sources of complexity at the same time,
- we do not consider the sources of complexity studied in [54,53].

A distinguishing feature of decision-theoretic troubleshooting following the work of Heckerman et al. [19] is the use of Bayesian networks for modeling of the system under consideration. We use a very simple Bayesian network model in Section 3, however the knowledge-modeling aspects of troubleshooting are not treated in this paper. A treatment of Bayesian network modeling, including applications to troubleshooting and diagnosis, may be found in [20].

Structure of the paper. In the next Section 2 we overview basic scheduling concepts used later on. Section 3 treats *basic troubleshooting* scenario and relates it to a simple scheduling problem. We also derive the result concerning *call service* there, in Section 3.1. In Section 4, we overview several extensions of basic troubleshooting proposed in earlier troubleshooting literature, relate them to suitable scheduling problems, and state our results. We also extend the troubleshooting-scheduling analogy to a multi-agent setting, where there are multiple repairmen working in parallel to minimize the repair time. We give an account of several problems similar to the ones studied in this paper in Section 5.

To keep the paper self-contained, we include Appendix A with a review of terminology pertaining to partial orders. It is assumed that the reader is familiar with basic notions of computational complexity theory. For an introduction to the theory the reader may consult any standard textbook such as [15].

2. Scheduling concepts

The material in this section is drawn mainly from surveys [29,44] and book [7].[1] While in this paper we restrict ourself to using few scheduling concepts and results, the cited publications can give the interested reader a more thorough overview of the field.

In deterministic machine scheduling, we are given a predetermined set of production activities, called *jobs*, to be processed on one or more *machines*. In the simplest model sufficient for this paper, any machine can perform at most one job at a time, and any job can be processed on at most one machine at a time. Once a job processing is started, it must continue without interruption until the processing is complete. Unless stated otherwise, we shall always assume that there is only one machine to process the jobs. A *schedule* is an allocation of time intervals on the machine to the jobs. A schedule is *feasible* if time intervals allocated to distinct jobs do not overlap. Additionally, there may be problem-specific restrictions to be met by a schedule to be considered feasible. The machine time is typically scarce and the task is to construct a feasible schedule minimizing some suitable objective, such as the completion time of the last processed job.

[1] The author gladly acknowledges using two very useful webpages to aid the bibliographic search of the vast scheduling literature: Dürr [13], Brucker and Knust [8].