International Journal of Approximate Reasoning 55 (2014) 977–988

Contents lists available at SciVerse ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Decision-theoretic troubleshooting: Hardness of approximation [☆]

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:
Available online 22 July 2013

Keywords:
Decision-theoretic troubleshooting
Hardness of approximation
NP-completeness
Min-sum set cover
Decision tree

ABSTRACT

Decision-theoretic troubleshooting is one of the areas to which Bayesian networks can be applied. Given a probabilistic model of a malfunctioning man-made device, the task is to construct a repair strategy with minimal expected cost. The problem has received considerable attention over the past two decades. Efficient solution algorithms have been found for simple cases, whereas other variants have been proven *NP*-complete. We study several variants of the problem found in literature, and prove that computing *approximate* troubleshooting strategies is *NP*-hard. In the proofs, we exploit a close connection to set-covering problems.

1. Introduction

In decision-theoretic troubleshooting [3], we are given a probabilistic model of a man-made device. The model describes faults, repair actions and diagnostic actions addressing the faults. Knowing that the modeled device is in a faulty state, the task is to find the most cost-efficient strategy for fixing the device with available repair and diagnostic actions. This is a natural optimization problem that has been studied independently in various contexts since the early days of computing [12,2,8].

Troubleshooting has become one of the areas to which we apply Bayesian networks [3,11] with interesting algorithmic problems and results [18]. The troubleshooting problem is known to be solvable in polynomial time under quite restrictive assumptions (to be discussed in Section 2). When these assumptions are relaxed, the problem is *NP*-hard [22]. Efficient heuristics exist that yield close-to-optimal results in practice [11,9]. Search algorithms for computing optimal troubleshooting strategies are described in [23]. However, we provide negative results showing that approximating the optimal solutions is, in general, a difficult problem.

Our contribution. We solve an open problem suggested by Ottosen [18] and show that troubleshooting with cost clusters (to be defined below) forming an acyclic directed graph is *NP*-complete, and it is *NP*-hard to approximate. We improve upon known *NP*-completeness results [22] by showing hardness of approximation for troubleshooting scenarios containing multiple dependent faults, dependent actions or questions.

Organization of the paper. In Section 2, we provide an overview of the various troubleshooting problem setups. Precise statements of our results are in Section 3.1. The proofs are presented in Section 3.2. These proofs utilize reductions from the *Min-sum set cover* problem [6] and the *Decision tree* problem [7,4].

☆ 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.

* Correspondence to: Institute of Information Theory and Automation of the AS CR Prague, Czech Republic.
E-mail address: lin@utia.cas.cz.

0888-613X/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ijar.2013.07.003

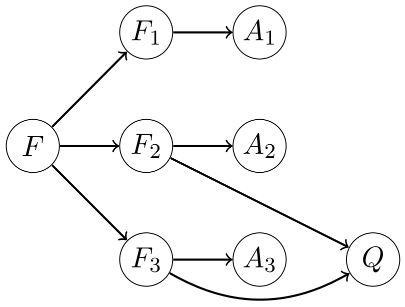

Fig. 1. Bayesian network for a troubleshooting model with single fault assumption and actions conditionally independent given the faults. There are three faults – F_1, F_2, F_3. To enforce the single fault assumption, we use a fault variable F with states $\{1, 2, 3\}$ and define the probability tables for all F_i so that $F_i = 1$ if and only if $F = i$. Actions A_1, A_2, A_3 each address one of the faults. A question Q can be used to discriminate between F_2 and F_3.

2. Troubleshooting models and strategies

Bayesian networks for troubleshooting [3,11] contain variables representing

- *faults*,
- repair actions, called simply *actions*, and
- diagnostic actions, called *questions*.

Actions have only two possible outcomes – either the system is fixed after the action has been performed, or it remains in a faulty state. We assume that we cannot introduce any new faults by performing the actions, and we know the outcome of any action immediately after its execution. *Questions* do not alter the state of the system, but may give useful information to direct the troubleshooting process.

Each action or question has an associated *cost*. These costs do not change over time except when stated otherwise. The actions and questions are *idempotent* [18] in the sense that repeating a failed action does not fix the system, and repeating a question provides the same answer as the first time the question was asked.

Under the *single fault assumption*, there can be at most one fault present in the system at any moment of time. A simple troubleshooting model with the single fault assumption is shown in Fig. 1.

Troubleshooting strategy [23] is a policy governing the troubleshooting process. An example of a troubleshooting strategy is shown in Fig. 2. In general, troubleshooting strategy is a rooted directed tree with internal nodes labeled by actions and questions. Edges are labeled by outcomes of the actions and questions. Each path from the root of the strategy to one of the leaves corresponds to a possible troubleshooting session starting at the root and terminating in the leaf. *Failure nodes* are all the leaf nodes for which the corresponding troubleshooting session fails to fix the system. For a strategy $\mathbf{S}$, we use this notation:

$\mathcal{L}(\mathbf{S})$	The set of leaves, also called *terminal nodes*.
$\mathcal{L}^-(\mathbf{S})$	The set of failure nodes, $\mathcal{L}^-(\mathbf{S}) \subset \mathcal{L}(\mathbf{S})$.
$\mathbf{e}_\ell$	The *evidence* (the outcomes of all actions and questions) compiled along the path from the root ϑ of strategy $\mathbf{S}$ to node ℓ. Let $E(\vartheta, \ell)$ be the set of all the edges constituting the path from ϑ to ℓ. Then $\mathbf{e}_\ell = \bigcup_{e \in E(\vartheta,\ell)} \mathrm{outcome}(e)$. An example is shown in Fig. 2.
$\mathcal{P}(\mathbf{e}_\ell)$	The probability of reaching node ℓ.
$t(\ell)$	The cost of performing all the actions and questions on the path from the root of $\mathbf{S}$ to node ℓ.
c_P	The penalty for not fixing the system.

Since we assume that each repair action has only two possible outcomes, "1" (system fixed) and "0" (system still in faulty state), the outdegree of all nodes labeled by repair actions is exactly two, with the edge labeled by "1" always leading to a terminal node in $\mathcal{L}(\mathbf{S}) \setminus \mathcal{L}^-(\mathbf{S})$. Our goal is to construct a strategy $\mathbf{S}$ minimizing the *expected cost of repair*

$$ECR(\mathbf{S}) = \sum_{\ell \in \mathcal{L}(\mathbf{S})} \mathcal{P}(\mathbf{e}_\ell) \cdot t(\ell) + \sum_{\ell \in \mathcal{L}^-(\mathbf{S})} \mathcal{P}(\mathbf{e}_\ell) \cdot c_P. \tag{1}$$

Fig. 2 gives an example of *ECR* evaluation.

2.1. Troubleshooting without questions

When there are no questions, the troubleshooting strategy is just a sequence of repair actions $A_1, \ldots, A_n$. The actions are performed in the given order and the troubleshooting session continues until the fault is fixed or all the actions have been used. In this paper, we assume the following.