Decision-Theoretic Troubleshooting: Hardness of Approximation

Václav Lín
Faculty of Management, University of Economics, Prague
and
Institute of Information Theory and Automation of the ASCR
lin@utia.cas.cz

Abstract
Troubleshooting is one of the application areas of Bayesian networks. Given a probabilistic model of a malfunctioning device, the task is to find the repair strategy with minimal expected cost. Except for simple cases, finding an optimal strategy is NP-hard. We show that optimal troubleshooting strategies are also hard to approximate.

1 Introduction

In decision-theoretic troubleshooting, 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 – the oldest works known to the author are (Johnson, 1956; Gluss, 1959).

Troubleshooting has become one of the application areas of Bayesian networks and as such it has received considerable attention over the last two decades. We refer to (Breese and Heckerman, 1996; Jensen et al., 2001; Ottosen, 2012) for discussion, references and survey of the most important results. The problem is known to be solvable in polynomial time under quite restrictive assumptions (to be discussed in Section 2). Otherwise, the problem is NP-hard (Vomlelová, 2003; Lín, 2011).

In Section 2, we recall several troubleshooting scenarios proposed in the literature. In Section 3 we identify a combinatorial optimization problem that captures their difficulty and show that computing approximate solutions with performance guaranteed within certain bounds is NP-hard.

Our Contribution. We solve an open problem suggested by Ottosen (2012) – we show that troubleshooting with cost clusters forming an acyclic directed graph (see below) is both NP-complete and NP-hard to approximate.

We strengthen known NP-completeness results due to Vomlelová (2003) by showing hardness of approximation for troubleshooting scenarios containing either multiple dependent faults or dependent actions.

All the troubleshooting scenarios that we consider are closely related to a single combinatorial problem – Min-sum Set Cover (Feige et al., 2004). This can be used to develop novel troubleshooting algorithms in the future (see also Section 4).

2 Troubleshooting Models and Strategies

Bayesian networks for troubleshooting contain variables representing faults, repair actions, called simply actions, and diagnostic actions, called questions. We take several assumptions common in earlier literature (Jensen et al., 2001; Vomlelová and Vomlel, 2003):

Actions have only two possible outcomes – either the system is fixed after the action has been performed, or it remains in a faulty state.
Figure 1: Bayesian network from Example 1. A troubleshooting model with single fault assumption and actions conditionally independent given the faults.

We assume that by performing the actions, we cannot introduce any new faults, 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. We take the assumption that these costs do not change over time.

Often, we take the single fault assumption – there can be but a single fault present in the system at any moment of time.

**Example 1.** A simple troubleshooting model is shown in Figure 1. 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 $F = i$. There are actions $A_1$, $A_2$, $A_3$, each addressing one of the faults. There is a question $Q$ that can be used to discriminate between $F_2$ and $F_3$.

Troubleshooting strategy is a policy governing the troubleshooting process. In general, it is a rooted directed tree with internal nodes labeled by actions and questions. Edges are labeled by outcomes of the actions and questions. The outdegree of nodes labeled by repair actions is exactly two, since we assume that each action has only two possible outcomes: “1” (system fixed) and “0” (system still in faulty state).

An example of a troubleshooting strategy is in Figure 2. Let $s$ be a troubleshooting strategy.

![Bayesian network](image1)

![Troubleshooting strategy](image2)

Each path from the root of $s$ to one of the leaves corresponds to a possible troubleshooting session starting in the root and terminating in the leaf. Failure nodes are all the terminal nodes $l$ of $s$ for which the corresponding troubleshooting session fails to fix the system. For a strategy $s$, we use this notation:

- $L(s)$ – the set of terminal nodes,
- $t(l)$ – the cost of performing all the actions and questions on the path from the root of $s$ to a terminal node $l \in L(s)$,
- $P(l)$ – the probability of reaching node $l$,
- $L^{-}(s) \subset L(s)$ – the set of failure nodes,
- $c_P$ – the penalty for not fixing the system.

The goal is to construct a strategy $s$ minimizing the expected cost of repair

$$ECR(s) = \sum_{l \in L(s)} P(l) \cdot t(l) + \sum_{l \in L^{-}(s)} P(l) \cdot c_P.$$  \hspace{1cm} (1)

Heuristic search algorithms for computing optimal troubleshooting strategies are described in (Vomlelová and Vomlel, 2003).
Example 2. We continue with Example 1, by showing how to evaluate the $ECR$ of the strategy shown in Figure 2. Assume that the probabilities specified for the Bayesian network shown in Figure 1 are:

- $P(F = 1) = \frac{1}{2}$,
- $P(F = 2) = P(F = 3) = \frac{1}{4}$,
- for $i = 1, 2, 3$, $F_i = 1$ if and only if $F = i$,
- for $i = 1, 2, 3$, $A_i = 1$ if and only if $F_i = 1$,
- $Q = 1$ if and only if $F_3 = 1$.

Failure terminal nodes of the strategy have zero probability, therefore we can disregard the penalty term $c_P$ of Formula 1. Assume the cost of $Q$ and all $A_i$’s is one. Summing over the terminal nodes with positive probability (see the $p$’s in Figure 2), we get

$$ECR = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 2 = \frac{3}{2}.$$  

Troubleshooting without Questions. When there are no questions, the troubleshooting strategy is just a sequence of actions. The actions are performed in order and the troubleshooting session continues until the fault is fixed or all the actions have been used.¹ When we assume that all the actions of a sequence $A_1, \ldots, A_n$ are sufficient to fix the fault, we can ignore the penalty term of Formula 1. Then we can use Formula 2 to compute the $ECR$:

$$\sum_{i=1}^{n} P(\bigcup_{j<i} A_j = 0 \cup \{A_i = 1\}) \cdot \sum_{j\leq i} c(A_i),$$

i.e., we multiply the cost of first $i$ actions by the probability of fixing the fault with the $i$-th action. Finding optimal strategies in polynomial time is known to be possible under quite restrictive assumptions (Jensen et al., 2001):

- There is exactly one fault present.
- The actions are conditionally independent given the faults and each action addresses exactly one fault.
- The action costs are constant over time and there are no questions.

Scenarios conforming to these assumptions are called basic troubleshooting. Optimal troubleshooting sequence is computed by ordering the actions so that the ratios $P(A_j = 1)/c(A_j)$ are non-increasing.²

Cost Clusters. When troubleshooting a large piece of machinery (such as the car engine), it is often necessary to disassemble the machine to perform certain actions. In general, some of the troubleshooting actions may require common initialization or preparatory work. To model such situations, Langseth and Jensen (2001) proposed an extension of the basic troubleshooting model, where the set of actions is partitioned into disjoint subsets, called cost clusters. To access actions within a cluster $K_i$, we have to pay additional cost, $c(K_i)$. Once $c(K_i)$ is paid, we can use actions from $K_i$ at any time, possibly mixed with actions from other clusters. Ottosen and Jensen (2010) have generalized the cost cluster scenario by allowing the cost clusters to form a tree and gave a polynomial-time algorithm for finding optimal troubleshooting sequences. Ottosen (2012) suggested a further generalization of the problem, where the cost clusters are allowed to form an acyclic directed graph. A simple model with cost cluster graph is shown in Figure 3.

¹ Note that this is a simplification. We could consider models where it is possible to terminate the troubleshooting process before we have tried all the available actions. Instead of continued troubleshooting, we would pay some fixed penalty. In the real life, this happens – we quite often decide to buy a new device before we have tried everything possible to fix an old one.

² In different contexts, this observation has already been made in the 1950’s – by Bellman (1957) in Dynamic Programming and by Smith (1956) in Scheduling.
c(K_i) is paid whenever we open K_i again. Troubleshooting cost clusters without inside information is \textit{NP-hard} (Lín, 2011).

\textbf{Modifications of the Cost Cluster Model.}

Gluss (1959) described a problem which is similar to the troubleshooting scenarios discussed in previous paragraph. Its complexity is unknown. Assume that there are independent faults and each of them is addressed by single action. Any number of faults can occur at the same time. The set of actions is partitioned into disjoint cost clusters (“modules” in (Gluss, 1959)). When we decide to perform an action from a particular cluster K_i, we pay the cluster opening cost c(K_i). If we continue performing actions from K_i, no additional cost is incurred. However, a cluster opening cost is incurred \textit{whenever} we switch between clusters. Outcome of an action is known immediately after it is performed (contrary to the cost cluster scenario without inside information).

When we take the single fault assumption, we can use observations made in (Lín, 2011) and show that this troubleshooting scenario is equivalent to a scheduling problem called “scheduling job families with sequence-independent setup times on a single machine to minimize total weighted completion time”, denoted 1|s_f|\sum w_j C_j in the Scheduling notation. This problem is mentioned by Potts and Kovalyov (2000). They give references to branch-and-bound algorithms for the problem but report that its complexity status is unresolved. To our knowledge, this is still true today.

\section{3 Hardness of Approximation}

We assume the reader is familiar with basic concepts of computational complexity theory. Therefore we only summarize some basic terminology used later on. For an introduction to the theory, we refer to textbooks such as (Garey and Johnson, 1979; Arora and Barak, 2009).

Given a minimization problem L, constant \(\rho > 1\), and a polynomial-time algorithm \(A\) for L, we say that \(A\) is a \textit{polynomial \(\rho\)-approximation algorithm} if for all instances \(x\) of problem \(L\) we have \(\rho \geq A(x)/\text{opt}(x)\). By \(A(x)\) we denote the objective value returned by algorithm \(A\) when applied to instance \(x\), and by \(\text{opt}(x)\) we denote the optimum of \(x\). We say that it is \textit{NP-hard} to approximate \(L\) within ratio \(\rho\), if there is a polynomial-time reduction \(\phi\) from 3SAT to \(L\), such that \(\phi\) combined with a hypothetical \(\rho\)-approximation algorithm for \(L\) would make it possible to decide 3SAT in polynomial time. By \textit{NP-completeness} of 3SAT, this would imply \(P=NP\). Inapproximability results are now known for many problems (Johnson, 2006).

\subsection{3.1 Reductions}

In this section, we treat troubleshooting scenarios without questions. We isolate several features of troubleshooting scenarios and show that each of them is sufficient to make troubleshooting hard to approximate. These features are: multiple dependent faults, dependent actions, acyclic directed graph of cost clusters.

The complexity of the troubleshooting scenarios studied in this paper is captured by single combinatorial problem that combines covering and sequencing – Min-sum Set Cover.

\textbf{Definition 1} (Min-sum Set Cover \((MSSC)\)). Input: A finite set \(U\), a collection of subsets \(\mathcal{C} = \{S \subseteq U\}\).
Objective: Find a linear ordering $\pi$ of $C$ minimizing the function

$$\sigma(U, C, \pi) = \sum_{u \in U} \pi(u),$$

where $\pi(u)$ is the index of the first set $S \in C$ covering $u$ under the ordering $\pi$.

**Theorem 1** (Feige et al. (2004)). MSSC has no polynomial $(4 - \epsilon)$-approximation algorithm for any $\epsilon > 0$ unless $P = NP$.

**Remark.** In the proofs of Theorems 2, 3, 4, we will assume that any MSSC instance $(U, C)$ used in the reductions is such that $U = \bigcup \{S \in C\}$, i.e., the set cover exists. We can make such an assumption without a loss of generality, since the hardness-of-approximation proof in (Feige et al., 2004) works with set systems for which set covers exist.

For later use, we record two simple lemmas. The first one follows from the definition. The second lemma is a consequence of the first one.

**Lemma 1.** Consider an MSSC instance $(U, C)$ and an ordering $\pi$ of $C$. Denote by $U_{\pi(i)}$ the set of $u \in U$ first covered by the $i$-th set of $\pi$. Then

$$\sigma(U, C, \pi) = \sum_{i=1}^{\lvert C \rvert} \lvert U_{\pi(i)} \rvert \cdot i. \quad (3)$$

**Lemma 2.** When $U = \bigcup \{S \in C\}$ and $\pi^*$ is an optimal ordering of $C$, then

$$\lvert U \rvert \leq \sigma(U, C, \pi^*) \leq \sum_{i=1}^{\lvert U \rvert} i. \quad (4)$$

We will reduce MSSC to several troubleshooting scenarios. In this section, we will work with scenarios without questions and will use Formula 2 for computation of the ECR. Let $\pi$ be a linear ordering of actions $\{A_i\}_{i=1}^n$ and define

$$p_{\pi(i)} = \mathcal{P}(\bigcup_{j<i} \{A_{\pi(j)} = 0\} \cup \{A_{\pi(i)} = 1\}).$$

Formula 2 then becomes

$$ECR = \sum_{i=1}^n p_{\pi(i)} \cdot \sum_{j \leq i} c(A_{\pi(i)}). \quad (5)$$

In definitions 2, 3, 4 we formally define simple troubleshooting scenarios, each of them isolating one property:

- single fault and dependent actions (Definition 2),
- multiple dependent faults (Definition 3),
- single fault, independent actions and cost clusters forming a DAG (Definition 4).

We show that already these very simple scenarios are hard to approximate.

In the proofs to follow, all the sets, collections of variables etc. are finite. All random variables are discrete.

**Definition 2** (Troubleshooting with Dependent Actions (TSDA)). Input: A random variable $F$ with values (faults) $f_1, \ldots, f_n$, a probability distribution $\mathcal{P}(F)$, a set of actions $\{A_i\}_{i=1}^k$. Each action fixes a subset of faults $F(A_i) \subseteq \{f_j\}_{j=1}^n$ with certainty and no other faults. Each fault is fixed by at least one action.

Objective: Find a linear ordering of actions minimizing the expected cost of repair.

**Theorem 2.** TSDA has no polynomial $(4 - \epsilon)$-approximation algorithm for any $\epsilon > 0$ unless $P = NP$.

**Proof.** We show that a special case of Troubleshooting with Dependent Actions is equivalent to Min-sum Set Cover. We reduce an MSSC instance $(U, C)$ to a troubleshooting problem as follows. Create a fault variable $f_u$ with a set of values $\{f_u\}_{u \in U}$. Distribution $\mathcal{P}(F)$ is uniform. For each $S \in C$ create an action $A_S$ with cost one. $A_S$ fixes $f_u$ with probability one if and only if $f_u \in S$. Let $k = \lvert C \rvert$. Given an arbitrary linear ordering $A_{\pi(1)}, \ldots, A_{\pi(k)}$ of the actions, denote by $F_{\pi(i)}$ the set of faults first covered by the $i$-th action of the ordering. The expected cost of repair (see Formula 5) is

$$\sum_{i=1}^k p_{\pi(i)} \cdot i = \sum_{i=1}^k i \cdot \lvert F_{\pi(i)} \rvert / \lvert U \rvert.$$

Using Lemma 1, we see that the $ECR$ multiplied by $\lvert U \rvert$ is equal to $\sigma(U, C, \pi)$. Therefore any approximation algorithm for TSDA could be used to approximate $(U, C)$ with the same approximation ratio. \qed
**Definition 3** (Troubleshooting with Dependent Faults (TSDF)). Input: A Bayesian network representing probability distribution $\mathcal{P}(F)$, binary random variables $F_1, \ldots, F_n \in F$ (faults), a set of actions $\{A_i\}_{i=1}^n$. For each fault, there is exactly one action that fixes it with certainty. Objective: Find a linear ordering $\pi$ of actions that minimizes the ECR.

**Theorem 3.** TSDF has no polynomial $(4-\epsilon)$-approximation algorithm for any $\epsilon > 0$ unless P=NP.

**Proof.** We use an idea by Vomlelová (2003) to transform the TSDA instance constructed in proof of Theorem 2 to TSDF. First, construct a Bayesian network for the TSDA with vertex set

$$\{F\} \cup \{F_u\}_{u \in U} \cup \{A_S\}_{S \in C}$$

and edge set

$$\{F \rightarrow F_u\}_{u \in U} \cup \{F_u \rightarrow A_S\}_{u \in S}.$$  

Variable $F$ has states $\{f_u\}_{u \in U}$ and an uniform probability distribution. All the other variables have deterministic distributions of probability: $F_u = 1$ if and only if $F = f_u$, and $A_S = 1$ if and only if at least one parent $F_u$ of $A_S$ has value 1. An example of such a network is shown in the left side of Figure 4 (for now, ignore the dotted part). We use the network to create a TSDF problem. Add to the network $k$ new vertices $A'_S$ and edges $A_S \rightarrow A'_S$ (see the dotted part of Figure 4). The new vertices are actions of the TSDF instance, the original actions $A_S$ are now faults. Cost of the new actions is one. The optimal ECR of the original TSDA problem is the same as the optimal ECR of the constructed TSDF problem.

**Remark.** In Definition 3, we place no restriction on $\mathcal{P}(F)$ except that it is represented by a Bayesian network. Bayesian network inference is in general a hard problem in itself (see e.g. Kwisthout et al. (2010) for references). In our proof of Theorem 3 we have constructed a Bayesian network for which the inference is easy and yet, the Troubleshooting problem is hard.

![Figure 4: Model with dependent actions for $U = \{1, 2, 3, 4\}$ and $C = \{\{1, 2\}, \{2, 3\}, \{2, 4\}\}$. The dotted edges and vertices show extension to a model with dependent faults.](image)

**Definition 4** (Troubleshooting with DAG Cost Clusters (TSCC)). Input: A random variable $F$ with values (faults) $f_1, \ldots, f_n$, a probability distribution $\mathcal{P}(F)$, a set of actions $\{A_i\}_{i=1}^n$. For each fault, there is exactly one action that fixes it with certainty. There is an acyclic directed graph $(V, E)$ of cost clusters. Vertices $v \in V$ represent cost clusters and edges $u \rightarrow v$ show that cluster $v$ can be accessed from $u$. Each action $A_i$ is assigned to some cluster $u \rightarrow v$ of the constructed TSDF problem. Cost of opening cluster $v$ is $c(v) \geq 0$.

Objective: Find a schedule of cluster opening and troubleshooting actions minimizing the expected cost of repair.

**Theorem 4.** TSCC has no polynomial $(3-\epsilon)$-approximation algorithm for any $\epsilon > 0$ unless P=NP. The result holds even when the cost cluster graph is bipartite.

**Proof.** We again reduce Min-sum Set Cover. For each item $u \in U$, create a fault $f_u$ with probability $P(F = f_u) = 1/|U|$ and an action $A_u$, solving exclusively $f_u$. Each action is contained in an associated cost cluster, $C_u$. The cost of opening $C_u$ and performing $A_u$ is one. For each set $S \in C$, create an empty cluster $C_S$ with cost $c$ ($c$ is a positive constant to be discussed later). Create directed edges from the
By Theorem 1, this would imply $P=NP$. Using Formula 6, we can write inequality
\[
\frac{\sigma(U, C, \pi)}{\sigma(U, C, \pi^*)} = \frac{ECR(\pi) - (n+1)/2}{ECR(\pi^*) - (n+1)/2} < 4 \quad (8)
\]
which is equivalent to
\[
\frac{ECR(\pi)}{ECR(\pi^*)} < 4 - \frac{\frac{3}{2}(n+1)}{ECR(\pi^*)} \quad (9)
\]
Now, for any lower bound $ECR(\pi^*)$ of $ECR(\pi^*)$, the inequality
\[
\frac{ECR(\pi)}{ECR(\pi^*)} < 4 - \frac{\frac{3}{2}(n+1)}{ECR(\pi^*)} \quad (10)
\]
implies inequality 9. To get the estimate $ECR$, we use Formula 6 and Lemma 2:
\[
ECR(\pi^*) = \frac{n+1}{2} + c \cdot n = \frac{n+1}{2} + c.
\]
When we set $c = n+1$ and use $ECR$ in inequality 10, we finally prove implication 7 via inequalities 9 and 8.

**Remark.** Note that had we used a greater value for the constant $c$ in the proof, we would get an approximation ratio closer to 4, yielding a stronger theorem.

**Corollary 1.** Troubleshooting with DAG Cost Clusters is NP-complete.

**Proof.** NP-completeness is a concept defined for decision problems. Therefore we have to consider the decision variant of TSCC: given an arbitrary positive constant $K$ and an instance $x$ of TSCC, is it true that $\text{opt}(x) \leq K$? This decision problem clearly belongs to class $NP$ – once we guess the cluster opening / action schedule $s(x)$, it is easy to compute the $ECR$ in polynomial time and check $ECR(s(x)) \leq K$. NP-hardness is implied by Theorem 4.

**4 Summary and Future Work**

In the proofs, we have seen that the three troubleshooting scenarios are closely related to a well studied combinatorial problem Min-sum Set Cover. The natural next step is to have
a look at the algorithms developed for the mentioned problem and see how well do they apply to troubleshooting. In particular, a simple greedy algorithm for MSSC achieves approximation ratio 4, and hence the bound given by Theorem 1 is tight (Feige et al., 2004).

An open problem is the hardness of approximation of troubleshooting with questions. The problem has been shown to be NP-hard by Vomlelová (2003). The NP-hardness can be shown by a simple reduction from the Decision Tree Problem (Garey and Johnson, 1979), which has recently been shown to be hard to approximate by Adler and Heeringa (2012). However, it is not obvious how to extend the reduction to prove also hardness of approximation for troubleshooting with questions.

The Troubleshooting community grew in the 1990’s without knowing about the very early works such as (Johnson, 1956) and (Gluss, 1959). It is worthwhile to perform further bibliographic research to see whether their work has been carried on in some communities other than our own.

Acknowledgments
I thank Jirka Vomlel for useful discussions.

References