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
http://dx.doi.org/10.1016/j.ijar.2013.07.003

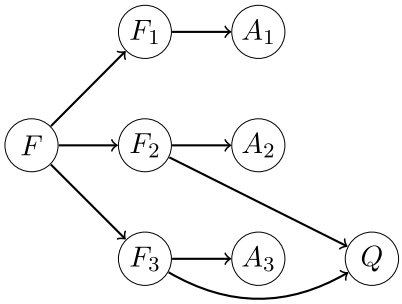

Fig. 1. Bayesian network for a troubleshooting model with single fault assumption and actions conditionally independent given the faults. There are three faults – F_1, F_2, F_3. To enforce the single fault assumption, we use a fault variable F with states $\{1, 2, 3\}$ and define the probability tables 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 $\mathbf{e}_\ell = \bigcup_{e \in E(\vartheta,\ell)} \text{outcome}(e)$. 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.

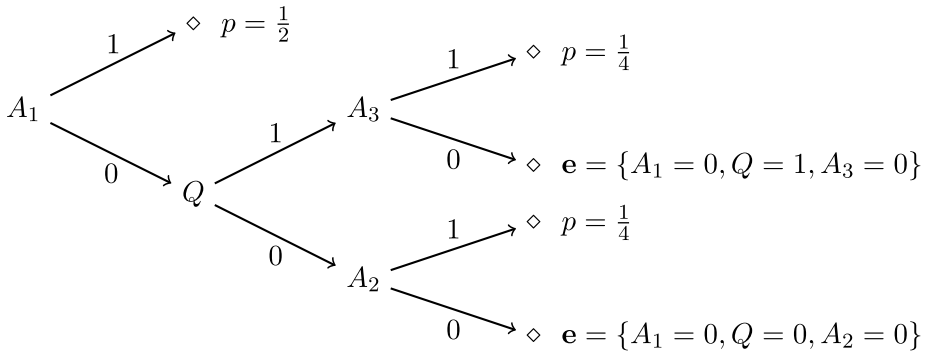

Fig. 2. A troubleshooting strategy for the model in Fig. 1. It contains actions A_1, A_2, A_3 and a question Q. Nodes are labeled by actions and questions; edges are labeled by action or question outcomes. According to this strategy, each troubleshooting session will begin with action A_1. If it fails ($A_1 = 0$), we use question Q. Depending on the outcome of Q, we perform either A_2 or A_3. Assume that all the A_i's and Q have unit cost, and the fault probabilities are $\mathcal{P}(F = 1) = \frac{1}{2}$, $\mathcal{P}(F = 2) = \frac{1}{4}$, $\mathcal{P}(F = 3) = \frac{1}{4}$. Further, assume that each action A_i solves the respective fault F_i with certainty. Then, summing over the terminal nodes with positive probability (the p's in the figure), we get $ECR = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 3 + \frac{1}{4} \cdot 3 = 2$. We display evidence $\mathbf{e}$ for two of the failure nodes.

Assumption 1. The available actions $A_1, \ldots, A_n$ are sufficient to fix the fault, and we never terminate the troubleshooting process before the system is fixed.[1]

With Assumption 1, we can ignore the penalty term of (1), and the *ECR* of a sequence $[A_1, \ldots, A_n]$ can be computed by

$$ECR(A_1, \ldots, A_n) = \sum_{i=1}^{n} \mathcal{P}\left(\bigcup_{j<i} \{A_j = 0\} \cup \{A_i = 1\} \right) \cdot \sum_{j \leqslant i} c(A_i), \tag{2}$$

that is, we multiply the cost of the first i actions by the probability of fixing the fault with the i-th action.

2.1.1. Basic troubleshooting

Finding optimal strategies in polynomial time is known to be possible under quite restrictive assumptions [11]:

Assumption 2. There is exactly one fault present in the system (the *single fault assumption*).

Assumption 3. The actions are conditionally independent given the faults, and each action addresses exactly one fault.

Assumption 4. The action costs are constant over time.

Assumption 5. There are no questions.

Scenarios conforming to these assumptions are called *basic troubleshooting*. Optimal strategy for a basic troubleshooting scenario is computed by ordering the actions so that the sequence of ratios $\frac{\mathcal{P}(A_i=1)}{c(A_i)}$ is non-increasing.[2] Due to the single fault assumption, Formula (2) becomes

$$ECR(A_1, \ldots, A_n) = \sum_{i=1}^{n} \mathcal{P}(A_i = 1) \cdot \sum_{j \leqslant i} c(A_j).$$

2.1.2. Cost clusters

The requirement for constant action costs, mentioned in Section 2.1.1, can be violated when some of the troubleshooting actions require common initialization or preparatory work. For example, it is often necessary to disassemble the machine to perform certain actions when troubleshooting a large piece of machinery. To model such situations, Langseth and Jensen [14] 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 $\mathcal{K}_i$, we have to pay an additional cost to *open* the cluster, $c(\mathcal{K}_i)$. Once a cluster $\mathcal{K}_i$ is opened, we can use actions from $\mathcal{K}_i$ at any time, possibly mixed with actions from other open clusters. Ottosen and Jensen [19] 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 [18] suggested a further generalization of

[1] Note that this is a simplification. In real life, we quite often decide to buy a new device before we have tried everything possible to fix the old one.
[2] This simple observation goes back to Bellman [2] and Smith [20].

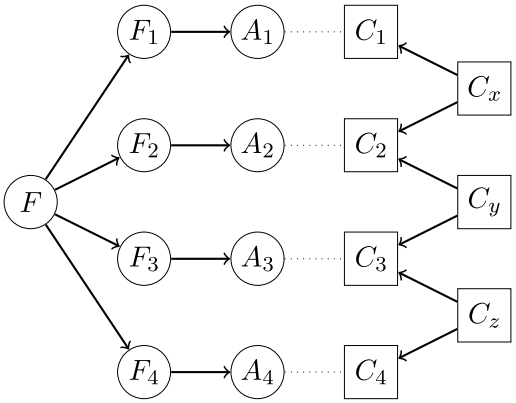

Fig. 3. Troubleshooting with cost clusters. At the left side is a Bayesian network with faults F_i and actions A_i. At the right side is the graph of cost clusters. To access, say, action A_2, we have to open clusters C_y and C_2 (or C_x and C_2). In general, a cost cluster may contain more than one action.

the problem, where the cost clusters are allowed to form an acyclic directed graph (DAG). A simple model with cost cluster graph is shown in Fig. 3. We study models with cost cluster DAGs in Section 3.

The cost cluster scenarios discussed so far differ from the scenario where there are cost clusters "*without inside information*" [14]. Without inside information, we have to close the most recently opened cluster when we need to check the outcome of troubleshooting actions. The cost $c(\mathcal{K}_i)$ is paid when we open $\mathcal{K}_i$ again. Troubleshooting cost clusters without inside information is *NP*-hard [15].

3. Hardness of approximation

The results of this paper pertain to troubleshooting scenarios that do not satisfy the assumptions of *basic troubleshooting* listed in Section 2.1.1. In Section 3.1, we formally define four simple scenarios, each of them violating exactly one of the assumptions of *basic troubleshooting*. We proceed to prove hardness of approximation for each of these scenarios in Section 3.2.

Preliminaries. We will review here some terminology and notation used later on. An introduction to the computational complexity theory can be found in standard textbooks such as [7,1].

Let L be a minimization problem, let x be an instance of the problem L, and let A be an algorithm for problem L. By $A(x)$ we denote the objective value returned by algorithm A when applied to instance x. By $opt(x)$ we denote the optimum value of x, and by $|x|$ we denote the size of instance x. Given a function $\rho : \mathbb{N} \to \mathbb{R}$ with $\rho(n) > 1$ for all n, we say that algorithm A is a *polynomial $\rho(n)$-approximation algorithm* if for all instances x of problem L we have $A(x) \leqslant \rho(|x|) \cdot opt(x)$, and algorithm A operates in time polynomial in $|x|$.

We say that it is *NP*-hard to approximate problem L within factor $\rho(n)$, if there is a polynomial-time reduction ϕ from *3SAT* to L, such that ϕ combined with a hypothetical $\rho(n)$-approximation algorithm for L would make it possible to decide *3SAT* in polynomial time. Due to *NP*-completeness of *3SAT*, this would imply $P = NP$. Equivalently, we say that problem L has no polynomial $\rho(n)$-approximation algorithm unless $P = NP$.

For some problems encountered in this paper, ρ is just a constant. For other problems, we do not state the function ρ explicitly but only bound it using the 'big-Ω' notation.

3.1. Statement of results

We state the results first and postpone the proofs until Section 3.2.

3.1.1. Troubleshooting scenarios without questions

We formally define simple troubleshooting scenarios, each of them breaking one of the requirements of *basic troubleshooting* listed in Section 2.1.1. Each scenario isolates a single property that makes it hard to approximate.

- Troubleshooting with dependent actions (Definition 1), breaking Assumption 3.
- Troubleshooting with multiple dependent faults (Definition 2), breaking Assumption 2.
- Troubleshooting with a single fault, independent actions and cost clusters forming a DAG (Definition 3), breaking Assumption 4.

We discuss properties of the scenarios in further detail in the remarks following the definitions.

Definition 1 *(Troubleshooting with dependent actions (TSDA)).*
Input: A random variable F with values (*faults*) $f_1, \ldots, f_n$, uniform probability distribution $\mathcal{P}(F)$, and a set of *actions* $\{A_1, \ldots, A_k\}$. Each action fixes a subset of faults $F(A_i) \subseteq \{f_1, \ldots, f_n\}$ with certainty and does not fix any other faults. Each fault is fixed by at least one action.
Objective: Find a linear ordering of actions minimizing the expected cost of repair.

Remark. In troubleshooting with dependent actions, Assumption 3 of *basic troubleshooting* is violated. This happens when the actions solve multiple faults, and some faults are solved by several actions.

Remark. In real situations, the probability distribution $\mathcal{P}(F)$ is not necessarily uniform and the actions may fail, that is, they need not fix faults with certainty. This comment also applies to Definitions 2 and 3.

Theorem 1. *The TSDA problem has no polynomial* $(4-\varepsilon)$*-approximation algorithm for any* $\varepsilon > 0$ *unless* $P = NP$.

Remark. Consider a greedy algorithm for troubleshooting with dependent faults described by Jensen et al. [11]. The algorithm builds the action sequence by iteratively picking a previously unused action with the highest ratio $\frac{\mathcal{P}(A|\mathbf{e})}{c(A)}$, where $\mathcal{P}(A|\mathbf{e})$ is the probability that the action A fixes the fault given that all preceding actions have failed. The algorithm is called *Updating P-over-C* in [18] because the probabilities $\mathcal{P}(A|\mathbf{e})$ of unused actions need to be updated in every step. Theorem 2.3 in [13] implies that the greedy algorithm is in fact a polynomial time 4-approximation algorithm for the following generalization of *TSDA*:

- The costs of actions are fixed but arbitrary.
- We accept the single fault assumption, but the fault distribution $\mathcal{P}(F)$ is not necessarily uniform.
- The distributions $\mathcal{P}(A_i|F(A_i))$ are arbitrary except that the probability $\mathcal{P}(A_i = 1|F(A_i) = \mathbf{0})$ is zero.

The result stated in Theorem 1 is therefore tight.

Definition 2 *(Troubleshooting with dependent faults (TSDF)).*
Input: A Bayesian network representing probability distribution $\mathcal{P}(\mathcal{F})$, binary random variables $F_1, \ldots, F_n \in \mathcal{F}$ (*faults*), and a set of *actions* $\{A_1, \ldots, A_n\}$. For each fault, there is exactly one action that fixes it with certainty. Each action fixes just one fault.
Objective: Find a linear ordering π of actions that minimizes the *ECR*.

Remark. In troubleshooting with dependent faults, there can be multiple faults present in the system at the same time. That possibility violates Assumption 2 of *basic troubleshooting*. If faults occur independently, the problem is solvable in polynomial time [21]. If faults do not occur independently, the problem is *NP*-hard [22].

Theorem 2. *The TSDF problem has no polynomial* $(4-\varepsilon)$*-approximation algorithm for any* $\varepsilon > 0$ *unless* $P = NP$.

Remark. In Definition 2, there is no restriction on $\mathcal{P}(\mathcal{F})$ except that it is represented by a Bayesian network. Bayesian network inference is a hard problem in itself [5]. In the proof of Theorem 2 we construct a Bayesian network for which the inference is easy, yet, the troubleshooting problem is hard.

Definition 3 *(Troubleshooting with DAG cost clusters (TSCC)).*
Input: A random variable F with values (*faults*) $f_1, \ldots, f_n$, uniform probability distribution $\mathcal{P}(F)$, and a set of *actions* $\{A_1, \ldots, A_n\}$. For each fault, there is exactly one action that fixes it with certainty. Each action fixes just one fault. There is an acyclic directed graph (V, E) of cost clusters. Vertices $v \in V$ represent cost clusters and an edge $u \to v$ shows that the cluster v can be accessed from cluster u. Each action A_i is assigned to exactly one cluster $v \in V$, and each cluster contains *zero or more* actions. The cost of opening cluster v is $c(v) \geqslant 0$.
Objective: Find a schedule of cluster opening and troubleshooting actions minimizing the expected cost of repair.

Remark. As discussed in Section 2.1.2, Assumption 4 of *basic troubleshooting* is violated when there are cost clusters of actions. When the clusters are allowed to form a DAG, we get a hard problem. When the cost clusters form a tree, the problem is solvable in polynomial time [19].

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

Theorem 4. *TSCC is NP-complete.*

3.1.2. Troubleshooting with questions

As in Section 3.1.1, we formally define a simple scenario that captures the complexity of using questions in troubleshooting. In this case, the probability distribution on faults is not required to be uniform. The actions and questions are deterministic.

Definition 4 *(Troubleshooting with questions (TSQ)).*
Input: A random variable F with values (*faults*) $f_1, \ldots, f_n$, a probability distribution $\mathcal{P}(F)$, and a set of *actions* $\{A_1, \ldots, A_n\}$. For each fault, there is exactly one action that fixes it with certainty. There are m binary questions $Q_j : \{f_1, \ldots, f_n\} \to \{0, 1\}$.
Objective: Find a troubleshooting strategy minimizing the expected cost of repair.

Remark. Troubleshooting with questions violates Assumption 5 of *basic troubleshooting*.

Remark. In realistic situations, we could have questions with more than two possible values. The questions could also be nondeterministic.

Theorem 5. *The TSQ problem has no polynomial $\Omega(\log n)$-approximation algorithm, where n is the number of faults, unless $P = NP$. For the special case of TSQ with a uniform probability distribution $\mathcal{P}(F)$, there is no polynomial $(4 - \varepsilon)$-approximation algorithm for any $\varepsilon > 0$, unless $P = NP$.*

3.2. Proofs of theorems

In all of the proofs the sets are finite, the random variables are discrete, and the probabilities are rational numbers.

3.2.1. Troubleshooting without questions

The starting point of all reductions in this section is the *Min-sum set cover* – a combinatorial problem that combines covering and sequencing. Easy reductions are possible since the probability distributions and costs considered in Definitions 1, 2, and 3 are uniform.

Definition 5 *(Min-sum set cover (MSSC)).*
Input: A finite set U, and a collection $\mathcal{C}$ of subsets of U, that is $\mathcal{C} \subseteq \{S\colon\ S \subseteq U\}$.
Objective: Find a linear ordering π of $\mathcal{C}$ minimizing the function

$$\sigma_{U,\mathcal{C}}(\pi) = \sum_{u \in U} i_\pi(u),$$

where $i_\pi(u)$ is the index of the first set $S \in \mathcal{C}$ covering element u under the ordering π.

Theorem 6. *(See Feige et al. [6].) MSSC has no polynomial $(4 - \varepsilon)$-approximation algorithm for any $\varepsilon > 0$ unless $P = NP$.*

Remark. In the proofs of Theorems 1, 2, and 3, we will consider only *MSCC* instances $(U, \mathcal{C})$ for which a set cover exists, that is $U = \bigcup_{S \in \mathcal{C}} S$. We can make such an assumption without a loss of generality, since the proof of Theorem 6 works with set systems for which set covers are guaranteed to exist [6].

We record two simple lemmas for later use:

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

$$\sigma_{U,\mathcal{C}}(\pi) = \sum_{i=1}^{|\mathcal{C}|} |U_{\pi(i)}| \cdot i. \tag{3}$$

Proof. Straight from Definition 5. □

Lemma 2. *Assume $U = \bigcup_{S \in \mathcal{C}} S$. Then $|U| \leqslant \sigma_{U,\mathcal{C}}(\pi)$ for any ordering π.*

Proof. A consequence of Lemma 1. □

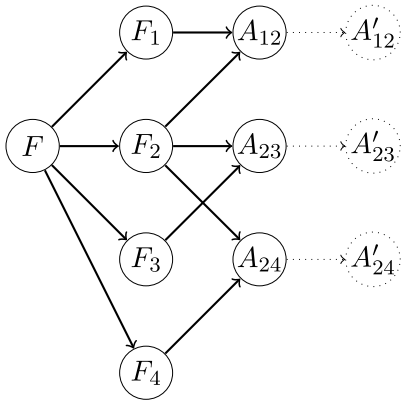

Fig. 4. Model with dependent actions for $U = \{1, 2, 3, 4\}$ and $\mathcal{C} = \{\{1, 2\}, \{2, 3\}, \{2, 4\}\}$. The dotted edges and vertices show extension to a model with dependent faults.

In this section, we work with scenarios without questions and use Formula (2) for computation of the *ECR*. For a linear ordering π of actions $\{A_1, \ldots, A_n\}$, we define

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

Formula (2) becomes

$$ECR(A_{\pi(1)}, \ldots, A_{\pi(n)}) = \sum_{i=1}^{n} p_{\pi(i)} \cdot \sum_{j \leqslant i} c(A_{\pi(i)}). \tag{4}$$

Theorem 1. *The TSDA problem has no polynomial* $(4 - \varepsilon)$*-approximation algorithm for any* $\varepsilon > 0$ *unless* $P = NP$.

Proof. The idea of the proof is to show that a special case of troubleshooting with dependent actions is equivalent to the min-sum set cover.

An instance $(U, \mathcal{C})$ of the *MSCC* problem is reduced to an instance of *TSDA* as follows. There is a fault variable F with a set of values $\{f_u\colon\ u \in U\}$. Distribution $\mathcal{P}(F)$ is uniform. For each $S \in \mathcal{C}$ there is an action A_S. Action A_S fixes fault f_u with probability one if and only if $u \in S$. All the actions have unit costs. For ease of notation, the actions are indexed by integers $1, \ldots, k$, where $k = |\mathcal{C}|$. 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. Using (4), we compute the expected cost of repair

$$ECR(A_{\pi(1)}, \ldots, A_{\pi(n)}) = \sum_{i=1}^{k} p_{\pi(i)} \cdot i = \sum_{i=1}^{k} i \cdot |F_{\pi(i)}|/|U|.$$

From Lemma 1, we know that the *ECR* multiplied by $|U|$ is equal to $\sigma_{U,\mathcal{C}}(\pi)$. Therefore any approximation algorithm for *TSDA* could be used to approximate $(U, \mathcal{C})$ with the same approximation ratio. □

Theorem 2. *The TSDF problem has no polynomial* $(4 - \varepsilon)$*-approximation algorithm for any* $\varepsilon > 0$ *unless* $P = NP$.

Proof. We use an idea by Vomlelová [22] to transform the *TSDA* instance constructed in the proof of Theorem 1 to an instance of *TSDF*. First, construct a Bayesian network with the vertex set

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

and the edge set

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

An example of such a network is shown in the left side of Fig. 4. Variable F has states $\{f_u\}_{u \in U}$ and a uniform probability distribution. All the other variables have deterministic distributions of probability:

- The probability $\mathcal{P}(F_u = 1|F)$ equals one if $F = f_u$; otherwise, it equals zero.
- The probability $\mathcal{P}(A_S = 1|\{F_u\}_{u\in S})$ equals one if at least one parent F_u of A_S has value 1; otherwise, the probability equals zero.

We use the network to create an instance of the *TSDF* problem. Add to the network a new vertex A'_S and a new edge $A_S \to A'_S$ for each $S \in \mathcal{C}$ (the dotted part of Fig. 4). The new vertices are *actions* of the *TSDF* instance, and the original actions A_S are now *faults*. The cost of each new action is one. The optimal *ECR* of the original *TSDA* problem is the same as the optimal *ECR* of the constructed *TSDF* problem. □

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

Proof. We again reduce the min-sum set cover.

For each element $u \in U$, create a fault f_u with probability $\mathcal{P}(F = f_u) = 1/|U|$ and an action A_u, exclusively solving f_u. Each action is contained in an associated cost cluster, C_u. The cost of opening C_u and performing A_u equals one.

For each set $S \in \mathcal{C}$, create an empty cluster C_S with cost c (c is a large positive constant to be discussed later). Create directed edges from the C_S's to the C_u's according to set membership – there is an edge $C_S \to C_u$ if and only if $S \ni u$. An example of a cost cluster model created this way is shown in Fig. 3.

In any optimal schedule, each cluster C_u is opened right before performing A_u; therefore, we shall not mention opening of the C_u's in the rest of the proof. Any troubleshooting sequence has to contain all the actions A_u. Their order is arbitrary, since their costs and probabilities of success are uniform. Thus the *ECR* can be decomposed into a sum of the expected cost of actions and the expected cost of opening "top-level" clusters C_S. Assume the clusters $\{C_S\}_{S\in\mathcal{C}}$ are indexed by integers $1,\dots,k$ and are opened in sequence $C_1,\dots,C_k$. Using (4), we get

$$ECR = \underbrace{\sum_{i=1}^{n}\frac{i}{n}}_{\text{actions}} + \underbrace{\sum_{j=1}^{k}\mathcal{P}(C_j)\cdot j\cdot c}_{\text{top-level clusters}} = \frac{n+1}{2} + c\cdot\sum_{j=1}^{k}\mathcal{P}(C_j)\cdot j,$$

where $n = |U|$, $k = |\mathcal{C}|$ and $\mathcal{P}(C_j)$ is the probability that by opening C_j we make accessible an action that fixes the fault. In such a case, C_j is the last cluster of the sequence that needs to be open. We assume that once a cluster C_S is opened, we perform all the actions accessible from C_S except for those that have already been performed. Indeed, if we did not perform the actions greedily after opening each cost cluster, the *ECR* would increase, because some of the cost clusters could be opened needlessly. With this assumption, $\mathcal{P}(C_j) = |F_{\pi(j)}|/n$, where $F_{\pi(j)}$ is the set of actions first made available by opening the j-th cluster. Let π be an ordering of $\mathcal{C}$ and let $\sigma_{U,\mathcal{C}}(\pi)$ be the value given by (3). Then there is a corresponding troubleshooting sequence specified by ordering π with

$$ECR(\pi) = \frac{n+1}{2} + c\cdot\frac{\sigma_{U,\mathcal{C}}(\pi)}{n}, \tag{5}$$

and the correspondence of *MSSC* solutions and *TSCC* solutions is one to one.

We conclude the proof by showing that for large values of c, the ratio $ECR(\pi)/ECR(\pi^*) \leqslant 4-\varepsilon$ implies

$$\frac{\sigma_{U,\mathcal{C}}(\pi)}{\sigma_{U,\mathcal{C}}(\pi^*)} \leqslant 4-\varepsilon. \tag{6}$$

Due to Theorem 6, this would imply $P = NP$. However, we have to make sure that the representation of c is not larger than a polynomial in n.

Let us denote any lower bound of $ECR(\pi^*)$ by $\underline{ECR}(\pi^*)$, and use (5) to express $\sigma_{U,\mathcal{C}}$ as $(ECR(\pi) - \frac{n+1}{2})\cdot\frac{n}{c}$. With a little algebra, one can rewrite (6) and verify that it is implied by

$$\frac{ECR(\pi)}{ECR(\pi^*)} \leqslant 4-\varepsilon-\frac{2(n+1)}{\underline{ECR}(\pi^*)}. \tag{7}$$

We set $c = (2k-\frac{1}{2})(n+1)$, $k > 0$, and use (5) and Lemma 2 to obtain $\underline{ECR}(\pi^*) = \frac{n+1}{2} + c = 2k(n+1)$. We use the $\underline{ECR}$ in inequality (7) and get the implication

$$\frac{ECR(\pi)}{ECR(\pi^*)} \leqslant 4-\varepsilon-\frac{1}{k} \quad\Longrightarrow\quad \frac{\sigma_{U,\mathcal{C}}(\pi)}{\sigma_{U,\mathcal{C}}(\pi^*)} \leqslant 4-\varepsilon.$$

Now we are almost done – all we need to show is that $\frac{1}{k}$ can be arbitrarily close to zero and yet, the number of bits needed to encode c as a binary number is polynomial in n. Let k equal $2^{\alpha(n)}$, where $\alpha(n)$ is an arbitrary polynomial in n, and let $|c|$ be the length of c in bits. We check that $|c|$ remains polynomial in n:

$$
\begin{aligned}
|c| &= \left\lceil \log_2\left(2k - \frac{1}{2}\right)(n+1) \right\rceil \\
&= O(\log_2 k + \log_2 n) \\
&= O\big(\alpha(n)\big) \quad \text{when } k = 2^{\alpha(n)}. \qquad \square
\end{aligned}
$$

Proof of Theorem 4. *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 $opt(x) \leqslant K$? This decision problem belongs to class *NP* – once we guess the strategy $s(x)$, it is easy to compute the *ECR* in polynomial time and check $ECR(s(x)) \leqslant K$. To show *NP*-hardness, we can use the same reduction as in the proof of Theorem 3 (ending with Eq. (5)). □

3.2.2. Troubleshooting with questions

We reduce the *Decision tree* problem to *TSQ* (Definition 4).

Definition 6 (*Binary decision tree (DT)*. (See [7].)
Input: A set $\mathcal{E} = \{e_1, \ldots, e_n\}$ of *entities* with a probability distribution $\mathcal{P}(\mathcal{E})$; we assume that all the probabilities are nonzero. A set $\mathcal{T} = \{T_1, \ldots, T_m\}$ of functions $T_j : \mathcal{E} \to \{0, 1\}$ called *tests*.
Objective: Construct a decision tree for $\mathcal{E}$ using the tests in $\mathcal{T}$ with minimal weighted external path length (defined below).

A *decision tree* is a rooted binary tree with leaves labeled by $e_i \in \mathcal{E}$ and non-leaf vertices labeled by $T_j \in \mathcal{T}$. If an entity passes a test, $T_j(e_i) = 1$, it follows the right branch, otherwise it follows the left branch. It is assumed that for each pair of distinct entities e_r, e_s, there exists a test T_j with $T_j(e_r) = 1$ and $T_j(e_s) = 0$. Therefore, the tests can be used to correctly identify each entity, and we require that a path from the root of the decision tree to the leaf uniquely determines the item e_i that labels the leaf.

Weighted external path length of tree $\mathbf{T}$ is $W(\mathbf{T}) = \sum_{i=1}^{n} \mathcal{P}(e_i) d(e_i)$, where $d(e_i)$ is the length of the path from the root of $\mathbf{T}$ to the leaf labeled by e_i.

The *DT* problem is known to be *NP*-hard [10]. As for inaproximability, we have a recently proven theorem:

Theorem 7. *(See Chakaravarthy et al. [4].) The DT problem has no polynomial $\Omega(\log n)$-approximation algorithm, where n is the number of entities in the input, unless $P = NP$. For the special case of DT with a uniform probability distribution $\mathcal{P}(\mathcal{E})$, there is no polynomial $(4 - \varepsilon)$-approximation algorithm for any $\varepsilon > 0$, unless $P = NP$.*

Troubleshooting with questions is at least as hard to approximate as *DT*:

Theorem 5. *The TSQ problem has no polynomial $\Omega(\log n)$-approximation algorithm, where n is the number of faults, unless $P = NP$. For the special case of TSQ with a uniform probability distribution $\mathcal{P}(F)$, there is no polynomial $(4 - \varepsilon)$-approximation algorithm for any $\varepsilon > 0$, unless $P = NP$.*

Proof. We reduce *DT* to *TSQ* as follows:

- For each entity $e_i \in \mathcal{E}$, there is a corresponding fault f_i, $\mathcal{P}(f_i) = \mathcal{P}(e_i)$.
- There are questions $Q_1, \ldots, Q_m$ corresponding to the tests, $\mathcal{P}(Q_j = 1 | f_i = 1) = T_j(e_i)$.
- The questions have unit costs.
- We accept the single fault assumption, and for each fault f_i, there is exactly one perfect action A_i solving the fault with uniform cost c_A (a large enough constant to be specified later).

The idea of the reduction is to set the cost of action c_A high enough so that it is never profitable to perform an action before a fault is completely identified by the questions. By Lemma 3 below, a cost of action with this property is $c_A = 1/\min_{e \in \mathcal{E}} \mathcal{P}(e)$. Note that the size of representation of c_A is not an issue here, since $\mathcal{P}(e)$ is part of the input parameters representing the *DT* problem. Using Lemma 3, we can restrict our attention to troubleshooting strategies that first identify the fault using a decision tree for $(\mathcal{E}, \mathcal{T})$ and perform a single action afterwards. For the decision tree $\mathbf{T}$, the *ECR* of its corresponding troubleshooting strategy is

$$ECR(\mathbf{T}) = W(\mathbf{T}) + c_A.$$

Unless $P = NP$, Theorem 7 implies that for any approximation algorithm, there will be input instances of the *Decision tree* problem with approximation ratio

$$\underbrace{\frac{W(\mathbf{T})}{W(\mathbf{T}^*)}}_{opt.} \geqslant r \cdot \log n$$

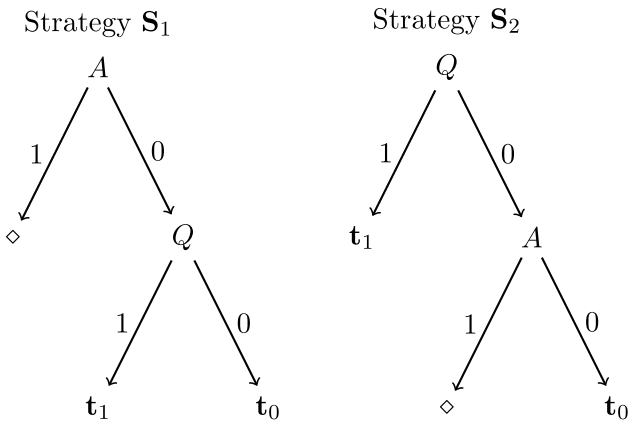

Fig. 5. The two strategies from the proof of Lemma 3. Strategy $\mathbf{S}_2$ is constructed from strategy $\mathbf{S}_1$ by interchanging action A and question Q.

for some (fixed) constant r and large n. A little rearrangement of the above inequality shows that any approximation algorithm for *TSQ* will have an approximation ratio equal to $ECR(\mathbf{T})/ECR(\mathbf{T}^*) \geqslant r' \cdot \log n$, for some $r' < r$:

$$W(\mathbf{T}) + c_A \geqslant W(\mathbf{T}^*) \cdot r \cdot \log n,$$
$$\frac{W(\mathbf{T}) + c_A}{W(\mathbf{T}^*) + c_A} \geqslant \frac{W(\mathbf{T}^*)}{W(\mathbf{T}^*) + c_A} \cdot r \cdot \log n = \frac{1}{1 + \frac{c_A}{W(\mathbf{T}^*)}} \cdot r \cdot \log n,$$
$$\frac{ECR(\mathbf{T})}{ECR(\mathbf{T}^*)} \geqslant \frac{1}{1 + c_A} \cdot r \cdot \log n, \quad \text{since } W(\mathbf{T}^*) \geqslant 1. \qquad \square$$

The proof of Theorem 5 relies on Lemma 3.

Lemma 3. *Consider the instance of TSQ constructed in the proof of Theorem 5 above, and denote by* $\mathbf{e}$ *the evidence at any particular step of a troubleshooting strategy. Then*:

- *When there is an action A with probability of success $0 < \mathcal{P}(A = 1|\mathbf{e}) < 1$, there is also at least one question Q with probability $0 < \mathcal{P}(Q = 1|\mathbf{e}) < 1$.*
- *Moreover, if the uniform cost of action is $c_A = 1/\min_{i=1}^{n} \mathcal{P}(f_i)$, then performing the question Q immediately before the action A leads to a strategy with a lower expected cost than performing A immediately before Q.*

Proof. The reduction from the *DT* problem to *TSQ* ensures that when there is an action A with the probability of success $\mathcal{P}(A = 1|\mathbf{e}) < 1$, then:

- There are at least two distinct faults f_i and f_j, $i \neq j$, with nonzero probability values.
- Action A solves exactly one of the faults f_i and f_j.
- There is a question Q that can "distinguish" between the two faults, that is $\mathcal{P}(Q = 1|\mathbf{e}, f_i)$ is either 0 or 1, and $\mathcal{P}(Q = 1|\mathbf{e}, f_j)$ equals $1 - \mathcal{P}(Q = 1|\mathbf{e}, f_i)$.

We want to show that it is always better to perform Q before A. Consider two strategies:

- Strategy $\mathbf{S}_1$ performs action A first, and if it fails, it performs question Q.
- Strategy $\mathbf{S}_2$ performs Q first and then it performs A in one of its branches.

These two strategies are shown in Fig. 5. Since action A addresses just one fault, there are two cases to consider. Either

(a) $\mathcal{P}(A = 1, Q = 1|\mathbf{e}) = 0$, or
(b) $\mathcal{P}(A = 1, Q = 0|\mathbf{e}) = 0$.

In case (a), strategy $\mathbf{S}_2$ schedules the action A in the 0-branch emanating from Q as shown at the right hand side of Fig. 5. In case (b), strategy $\mathbf{S}_2$ schedules the action A in the 1-branch. In the rest of the proof, we consider case (a), but case (b) is symmetric. We denote the cost of Q by c_Q, and the sub-trees of strategies by $\mathbf{t}_0$ and $\mathbf{t}_1$. We use the notation introduced in Section 2: For a leaf node ℓ, we denote by $\mathbf{e}_\ell$ the associated evidence. By $t(\ell)$ we denote the cost of reaching ℓ from the root of $\mathbf{t}_0$ (or $\mathbf{t}_1$). The expected costs of the two strategies $\mathbf{S}_1$ and $\mathbf{S}_2$ are

Table 1
Summary of results.

Troubleshooting scenario	Hard to approximate within
Dependent actions (*TSDA*)	$4-\varepsilon$ for all $\varepsilon > 0$
Dependent faults (*TSDF*)	$4-\varepsilon$ for all $\varepsilon > 0$
DAG Cost clusters (*TSCC*)	$4-\varepsilon$ for all $\varepsilon > 0$
Questions (*TSQ*)	$\Omega(\log n)$

$$ECR(\mathbf{S}_1|\mathbf{e}) = \mathcal{P}(A=1|\mathbf{e})\cdot c_A + \sum_{\ell\in\mathcal{L}(\mathbf{t}_1)} \mathcal{P}(\mathbf{e}_\ell, A=0, Q=1|\mathbf{e})\cdot\big[c_A + c_Q + t(\ell)\big] + \sum_{\ell\in\mathcal{L}(\mathbf{t}_0)} \mathcal{P}(\mathbf{e}_\ell, A=0, Q=0|\mathbf{e})\cdot\big[c_A + c_Q + t(\ell)\big],$$

$$ECR(\mathbf{S}_2|\mathbf{e}) = \sum_{\ell\in\mathcal{L}(\mathbf{t}_1)} \mathcal{P}(\mathbf{e}_\ell, Q=1|\mathbf{e})\cdot\big[c_Q + t(\ell)\big] + \mathcal{P}(Q=0, A=1|\mathbf{e})\cdot[c_Q + c_A] + \sum_{\ell\in\mathcal{L}(\mathbf{t}_0)} \mathcal{P}(\mathbf{e}_\ell, A=0, Q=0|\mathbf{e})\cdot\big[c_A + c_Q + t(\ell)\big].$$

To have strategy $\mathbf{S}_2$ more efficient than strategy $\mathbf{S}_1$, we need $ECR(\mathbf{S}_2|\mathbf{e}) - ECR(\mathbf{S}_1|\mathbf{e}) \leqslant 0$ to hold. Since the probability $\mathcal{P}(A=1, Q=1|\mathbf{e})$ is zero, most of the terms conveniently cancel out when we perform the subtraction:

$$\begin{aligned}
0 &\geqslant ECR(\mathbf{S}_2|\mathbf{e}) - ECR(\mathbf{S}_1|\mathbf{e}),\\
0 &\geqslant c_A\cdot\underbrace{\big[\mathcal{P}(Q=0, A=1|\mathbf{e}) - \mathcal{P}(A=1|\mathbf{e})\big]}_{\mathcal{P}(A=1,Q=1|\mathbf{e})=0} + c_Q\cdot\underbrace{\mathcal{P}(Q=0, A=1|\mathbf{e})}_{\mathcal{P}(A=1|\mathbf{e})}\\
&\quad + \sum_{\ell\in\mathcal{L}(\mathbf{t}_1)} \underbrace{\big[\mathcal{P}(\mathbf{e}_\ell, Q=1|\mathbf{e}) - \mathcal{P}(\mathbf{e}_\ell, A=0, Q=1|\mathbf{e})\big]}_{\mathcal{P}(\mathbf{e}_\ell, A=1, Q=1|\mathbf{e})=0}\cdot\big(c_Q + t(\ell)\big) - c_A\cdot\underbrace{\sum_{\ell\in\mathcal{L}(\mathbf{t}_1)} \mathcal{P}(\mathbf{e}_\ell, A=0, Q=1|\mathbf{e})}_{\mathcal{P}(Q=1|\mathbf{e})},\\
0 &\geqslant c_Q\cdot\mathcal{P}(A=1|\mathbf{e}) - c_A\cdot\mathcal{P}(Q=1|\mathbf{e}). \qquad (8)
\end{aligned}$$

To finish the proof, we claim that the value $c_A = 1/\min_{i=1}^n \mathcal{P}(f_i)$ satisfies inequality (8) since:

- As we have argued, $\mathcal{P}(Q=1|\mathbf{e}) \geqslant 0$. By the construction in the proof of Theorem 5, we have $\mathcal{P}(Q=1|\mathbf{e}) \geqslant \min_{i=1}^n \mathcal{P}(f_i)$.
- By the construction in the proof of Theorem 5, we also have $c_Q = 1$.
- We assume $\mathcal{P}(A=1|\mathbf{e}) < 1$. □

4. Summary and discussion

The results are summarized in Table 1. As discussed in Section 3.1.1, the result for troubleshooting with dependent actions is tight for a restricted, but relevant, variant of the problem. For other scenarios, no approximation algorithms have been designed so far. Therefore the results reported might not be tight.

As we have shown in Section 3.2.1, the *Min-sum set cover* problem [6] is equivalent to a special case of troubleshooting with dependent actions. A generalized variant of the *min-sum set cover* is studied under the name *pipelined set cover* by Munagala et al. [17]. There, the sets (corresponding to faults) have associated weights (corresponding to probabilities of faults), and the elements (corresponding to repair actions) have associated costs. The problem is further generalized by Kaplan et al. [13]. All three papers [6,17,13] provide insights and results that are directly applicable to the greedy algorithm *Updating P-over-C* [11] discussed in this paper in Section 3.1.

Problems solvable in polynomial time. A special case of troubleshooting with questions has already been implicitly addressed by Johnson [12]. He gives an algorithm that runs in $O(n\log n)$ time and is optimal under reasonable assumptions. Likewise, a very restricted case of troubleshooting with dependent actions is solvable by graph matching [23] in $O(n\sqrt{m})$ time [16], where m is the number of faults and n is the number of actions. Identification of interesting special cases solvable in polynomial time is an open research area.

Acknowledgements

I thank the reviewers and Jirka Vomlel for useful comments.

References

[1] Sanjeev Arora, Boaz Barak, Computational Complexity – A Modern Approach, Cambridge University Press, ISBN 978-0-521-42426-4, 2009.
[2] Richard E. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.
[3] John S. Breese, David Heckerman, Decision-theoretic troubleshooting: A framework for repair and experiment, in: Eric Horvitz, Finn Verner Jensen (Eds.), UAI, Morgan Kaufmann, ISBN 1-55860-412-X, 1996, pp. 124–132.
[4] Venkatesan T. Chakaravarthy, Vinayaka Pandit, Sambuddha Roy, Pranjal Awasthi, Mukesh K. Mohania, Decision trees for entity identification: Approximation algorithms and hardness results, ACM Transactions on Algorithms (ISSN 1549-6325) 7 (2) (March 2011) 15:1–15:22.
[5] Gregory F. Cooper, The computational complexity of probabilistic inference using bayesian belief networks, Artificial Intelligence 42 (2) (1990) 393–405.
[6] Uriel Feige, László Lovász, Prasad Tetali, Approximating min sum set cover, Algorithmica 40 (4) (2004) 219–234.
[7] Michael R. Garey, David S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, ISBN 0-7167-1044-7, 1979.
[8] Brian Gluss, An optimum policy for detecting a fault in a complex system, Operations Research 7 (4) (1959) 468–477.
[9] Korhan Gökçay, Taner Bilgiç, Troubleshooting using probabilistic networks and value of information, International Journal of Approximate Reasoning 29 (2) (2002) 107–133.
[10] Laurent Hyafil, Ronald L. Rivest, Constructing optimal decision trees is NP-Complete, Information Processing Letters 5 (1) (1976).
[11] Finn Verner Jensen, Uffe Kjærulff, Brian Kristiansen, Helge Langseth, Claus Skaanning, Jiří Vomlel, Marta Vomlelová, The SACSO methodology for troubleshooting complex systems, AI EDAM 15 (4) (2001) 321–333.
[12] Selmer M. Johnson, Optimal Sequential Testing, Number RM1652, Rand Corporation, 1956.
[13] Haim Kaplan, Eyal Kushilevitz, Yishay Mansour, Learning with attribute costs, in: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, ACM, 2005, pp. 356–365.
[14] Helge Langseth, Finn Verner Jensen, Heuristics for two extensions of basic troubleshooting, in: Henrik Hautop Lund, Brian H. Mayoh, John W. Perram (Eds.), SCAI, volume 66, in: Frontiers in Artificial Intelligence and Applications, vol. 66, IOS Press, ISBN 1-58603-161-9, 2001, pp. 80–89.
[15] Václav Lín, Extensions of decision-theoretic troubleshooting: Cost clusters and precedence constraints, in: Weiru Liu (Ed.), ECSQARU, in: Lecture Notes in Computer Science, vol. 6717, Springer, ISBN 978-3-642-22151-4, 2011, pp. 206–216.
[16] Silvio Micali, Vijay V. Vazirani, An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs, in: 21st Annual Symposium on Foundations of Computer Science, 1980, IEEE, 1980, pp. 17–27.
[17] Kamesh Munagala, Shivnath Babu, Rajeev Motwani, Jennifer Widom, The pipelined set cover problem, in: Thomas Eiter, Leonid Libkin (Eds.), ICDT, in: Lecture Notes in Computer Science, vol. 3363, Springer, ISBN 3-540-24288-0, 2005, pp. 83–98.
[18] Thorsten J. Ottosen, Solutions and heuristics for troubleshooting with dependent actions and conditional costs, PhD thesis, Aalborg University, Denmark, 2012.
[19] Thorsten J. Ottosen, Finn V. Jensen, The cost of troubleshooting cost clusters with inside information, in: Peter Grünwald, Peter Spirtes (Eds.), UAI, AUAI Press, ISBN 978-0-9749039-6-5, 2010, pp. 409–416.
[20] Wayne E. Smith, Various optimizers for single-stage production, Naval Research Logistics Quarterly (ISSN 1931-9193) 3 (1–2) (1956) 59–66.
[21] Sampath Srinivas, A polynomial algorithm for computing the optimal repair strategy in a system with independent component failures, in: Philippe Besnard, Steve Hanks (Eds.), UAI, Morgan Kaufmann, ISBN 1-55860-385-9, 1995, pp. 515–522.
[22] Marta Vomlelová, Complexity of decision-theoretic troubleshooting, International Journal of Intelligent Systems 18 (2) (2003) 267–277.
[23] Marta Vomlelová, Jiří Vomlel, Troubleshooting: NP-hardness and solution methods, Soft Computing 7 (5) (2003) 357–368.