Troubleshooting: NP-hardness and solution methods

Marta Vomlelová* and Jiří Vomlel*
marta@cs.auc.dk, jirka@cs.auc.dk

Research Unit of Decision Support Systems,
Department of Computer Science, Aalborg University,
Fredrik Bajers Vej 7E, DK-9220 Aalborg, Denmark

Abstract

Troubleshooting is one of the areas where Bayesian Networks are successfully applied [8]. In the present paper we show that the generally defined troubleshooting task is NP-hard. Heuristic algorithms that find an optimal solution are described. A new method approximating the optimal solution within polynomial time is proposed. Approximations obtained through two different methods are compared with the optimal solutions. The comparisons are performed on models for troubleshooting of laser printers. We observed that the suboptimal solutions provided by the polynomial methods were not far from the optimal ones.

1 Introduction

We model the troubleshooting problem by a Bayesian Network encoding relations among the following variables: *faults* of the device $F \in \mathcal{F}$, *actions* $A \in \mathcal{A}$ - troubleshooting steps that may solve the problem, and *questions* $Q \in \mathcal{Q}$ - troubleshooting steps that help to identify the fault. Every action and question has a cost assigned, c_A denotes the cost of an action A and c_Q the cost of a question Q. When there is no risk of confusion we will abbreviate c_{A_i} to c_i. Before we introduce formal definitions we will start with a simple example.

1.1 Light print example

Suppose a page that came out of your printer is too light. The possible faults of the printer in the case of a light print problem are listed in Table 1. Please

*Both authors are members of the Laboratory for Intelligent Systems (LISp) at the University of Economics, Prague. This paper is based on the research performed during the authors' stay at Aalborg University. The research was supported by the National Centre for IT Research through grant #87.2. In the Czech Republic the authors were supported through grant MSMT VS96008 of the Ministry of Education, Youth and Sports of the Czech Republic.

note, that in a real model of light print designed by the domain experts consists of 22 faults. Let us regard a simplified model with four faults only.

Fault	$p(F_i)$
F_1: `Distribution problem`	0.4
F_2: `Defective toner`	0.3
F_3: `Corrupted dataflow`	0.2
F_4: `Wrong driver setting`	0.1

Table 1: Possible faults of the printer in the case of light print

There are several actions that may solve these faults (see Table 2). For example, the action `Try another toner` solves the `Distribution problem` and `Defective toner` with the probability 0.9 (i.e. $p(A_2 = yes \mid F_1) = p(A_2 = yes \mid F_2) = 0.9$), but it does not solve `Wrong driver setting` at all (i.e. $p(A_2 = yes \mid F_4) = 0$.

Actions and questions	c_i
A_1: `Remove, shake and reseat toner`	5
A_2: `Try another toner`	15
A_3: `Cycle power`	1
Q_1: `Is the printer configuration page printed light?`	2

Table 2: Actions and one question in the light print example

Generally, for every action an expert provides us with a table $p(A_i = yes \mid F_j)$. The conditional probability table of $p(A_2 = yes \mid F_j)$ is defined in Table 3.

F_j	$p(A_2 = yes \mid F_j)$
F_1	0.9
F_2	0.9
F_3	0
F_4	0

F_j	$p(Q_1 = yes \mid F_j)$
F_1	1
F_2	1
F_3	0
F_4	0

Table 3: Conditional probability tables

During the troubleshooting session it is often advisable to ask the user to answer some questions. The answers may help solve the problem faster by identifying the fault of the device. For instance if the answer to question Q_1 "`Is the` printer configuration page printed light?" is negative then the faults: Distribution problem and Defective toner are eliminated and the remaining faults are Corrupt data flow and Wrong driver setting. Generally, we have a table $p(Q_i = yes \mid F_j)$ for every question Q_i. See Table 3 for the definition of the conditional probability table of $p(Q_1 = yes \mid F_j)$.

If two assumptions are accepted: *conditional independence of actions and questions* given a fault and the *single fault assumption*, then the model corresponds to the Bayesian Network in Figure 1.

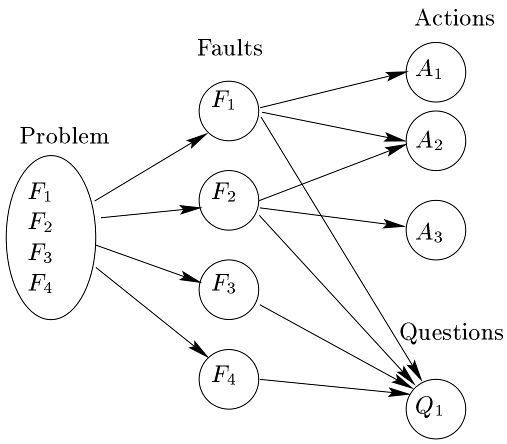

Figure 1: Bayesian Network model

There are many possible troubleshooting strategies. For example, first try to A_3 : Cycle power and then stop troubleshooting. Another strategy is: A_3 : Cycle power and in case this is unsuccessful, A_2 : Try another toner. A third strategy may require the user first to answer question Q : Is the printer configuration page printed light? If the answer is *yes*, then A_1 : Remove shake and reseat toner. If the answer is *no*, then try A_3 : Cycle power. The expected cost of repair (ECR) enables us to compare strategies. Table 4 shows the calculations of ECR for the three examples presented above, where c_{A_i} is the cost of action A_i and c_{CS} is a penalty for not solving the problem. Please note that real life troubleshooting strategies contain substantially more actions and questions than the presented ones.

1.2 Troubleshooting task specification

If only actions are considered, then every troubleshooting strategy can be described as a sequence of actions that are performed until the problem is solved. When questions are part of troubleshooting, then the solution of a troubleshooting task need not to be a sequence anymore. For every answer to a question a strategy may be to perform different troubleshooting steps (or the same steps in different order). Thus a troubleshooting strategy has generally the form of a

Strategy	**ECR**
A_3	$c_{A_3} + p(A_3 = no) \cdot c_{CS}$
A_3, A_2	$c_{A_3} + p(A_3 = no) \cdot\ c_{A_2} + p(A_2 = no, A_3 = no) \cdot c_{CS}$
$Q \begin{cases} A_1 \\ A_3 \end{cases}$	$c_Q + \begin{cases} + p(Q = yes) \cdot\ c_{A_1} + p(A_1 = no, Q = yes) \cdot c_{CS} \\ + p(Q = no)\ \cdot\ c_{A_3} + p(A_3 = no, Q = no) \cdot c_{CS} \end{cases}$

Table 4: ECR calculations

directed tree where branching may occur after every question. Figure 2 provides an example of a troubleshooting strategy.

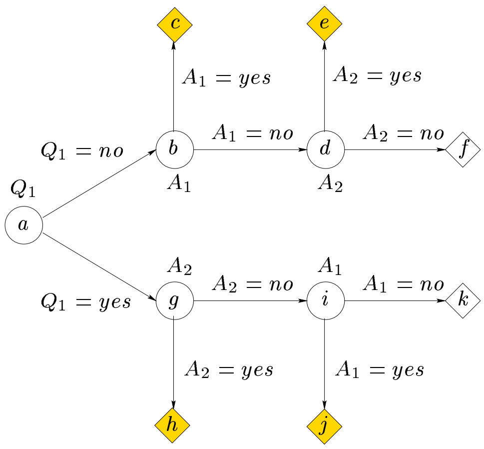

Figure 2: Troubleshooting strategy

There are two types of nodes in the tree - *chance nodes* and *terminal nodes*. In Figure 2 circles are used to denote chance nodes and the diamonds to denote terminal nodes. The label of a *chance node* assigns a troubleshooting step (action or question) to that node, e.g. the label of node a is Q_1 in Figure 2 . Every edge coming out from a chance node is labeled by an outcome of the troubleshooting step corresponding to that node, $label(edge)$ will denote the function that provides the *edge* labels.

In every *terminal node* the troubleshooting strategy terminates. There are two ways for a troubleshooting strategy to terminate, either by solving the problem or by giving up. Thereupon two types of *terminal nodes* are defined: (1) *Success terminal nodes* correspond to solving the problem. (2) *Failure terminal nodes* correspond to giving up the troubleshooting. Success terminal nodes are filled while failure terminal nodes are not in Figure 2.

Definition 1 *Troubleshooting strategy* is a labeled directed tree that describes the process of performing actions and questions until the process terminates.

The set of all terminal nodes of a strategy $\mathbf{s}$ will be denoted by $\mathcal{L}(\mathbf{s})$. Please note that all leaves of a troubleshooting strategy are terminal nodes. The root node of all strategies will be denoted by ϑ.

Let $path(m,n)$ be the sequence of edges constituting the path from node m to node n in the troubleshooting strategy, then $\mathbf{e}_n = \bigcup_{e \in path(\vartheta,n)} label(e)$ defines the evidence compiled through the performance of actions and questions in $path(\vartheta,n)$.

Further let $p(\mathbf{e}_n \mid \mathbf{e}_m)$ for $\mathbf{e}_m \subseteq \mathbf{e}_n$ denote conditional probability of evidence $\mathbf{e}_n$ given evidence $\mathbf{e}_m$, i.e. the probability of getting to node n from node m. Please note that since $\mathbf{e}_s = \emptyset$, the probability of getting to node n from the root is $p(\mathbf{e}_n)$.

Finally let $t(m,n)$ denote total the total cost of actions and questions in the path from node m to node n. For example, in Figure 2, the evidence corresponding to node b labeled by A_1 is $Q_1 = no$, the probability of getting there is $p(Q_1 = no)$, and the total cost of getting there is c_{Q_1}.

A penalty function $c(\mathbf{e}_\ell)$ applies for every terminal node ℓ. The penalty is defined to be zero if the problem is solved, i.e. for all success terminal nodes. The penalty may be interpreted as a cost of calling service. If the penalty is constant for all failure terminal nodes then it will be denoted by c_{CS}.

Next, we shall define the expected cost of repair of a troubleshooting strategy.

Definition 2 *Expected cost of repair* (ECR) of a troubleshooting strategy $\mathbf{s}$ is defined as

$$ECR(\mathbf{s}) = \sum_{\ell \in \mathcal{L}(\mathbf{s})} p(\mathbf{e}_\ell) \cdot [\ t(\vartheta,\ell) + c(\mathbf{e}_\ell)\ ]\,. \tag{1}$$

Remark. If troubleshooting strategy $\mathbf{s}$ is a sequence of actions $A_1, A_2, \ldots, A_n$, then its ECR can be computed by

$$\begin{aligned}
ECR(\mathbf{s}) &= p(A_1 = yes) \cdot c_1 + p(A_1 = no, A_2 = yes) \cdot (c_1 + c_2) + \ldots \\
&\quad \ldots + p(A_1 = no, \ldots, A_{n-1} = no, A_n = yes) \cdot (c_1 + \ldots + c_n) \\
&\quad + p(A_1 = no, \ldots, A_i = no) \cdot c(A_1 = no, \ldots, A_n = no) \\
&= c_1 + \sum_{i=2}^{n} p(A_1 = no, \ldots, A_{i-1} = no) \cdot c_i \\
&\quad + p(A_1 = no, \ldots, A_n = no) \cdot c(A_1 = no, \ldots, A_n = no).
\end{aligned} \tag{2}$$

For a strategy $\mathbf{s}$ and a node m of this strategy, let symbol $\mathbf{s}_m$ denote a sub-strategy of $\mathbf{s}$ such that m is the root of $\mathbf{s}_m$ and all successors of m in $\mathbf{s}$ are contained in $\mathbf{s}_m$ as well. Please note that $\mathbf{s} = \mathbf{s}_\vartheta$.

It will be useful to have Definition 2 generalized so that the expected cost of repair of a troubleshooting sub-strategy $\mathbf{s}_m$ given an evidence $\mathbf{e}_m$ corresponding to root m of $\mathbf{s}_m$ is defined. It can be further extended to the case where an additional evidence $\mathbf{e}'$ is entered into the Bayesian Network. Please observe that for $m = \vartheta$ and $\mathbf{e}' = \emptyset$ we get Definition 2.

Definition 3 Let $\mathbf{e}'$ correspond to additional evidence compiled in the Bayesian Network. *Expected cost of repair* of a troubleshooting sub-strategy $\mathbf{s}_m$ given the evidence $\mathbf{e} = \mathbf{e}_m \cup \mathbf{e}'$ is defined as

$$ECR(\mathbf{s}_m \mid \mathbf{e}) \quad = \quad \sum_{\ell \in \mathcal{L}(\mathbf{s}_m)} p(\mathbf{e}_\ell \mid \mathbf{e}) \cdot [\ t(m,\ell) + c(\mathbf{e}_\ell)\ ] \,. \tag{3}$$

The following lemma provides directions for a recursive computation of ECR.

Lemma 1 *If $\mathbf{e} \subseteq \mathbf{e}_m$, then $ECR(\mathbf{s}_m \mid \mathbf{e})$ can be computed by the following recurrent formula:*

- *If m is a terminal node, then $ECR(\mathbf{s}_m \mid \mathbf{e}) = c(\mathbf{e})$.*
- *If m is a chance node corresponding to a troubleshooting step S with outcomes $s_1, s_2, \ldots, s_r$ and $m_1, m_2, \ldots, m_r$ are children of node m, then*

$$ECR(\mathbf{s}_m \mid \mathbf{e}) \quad = \quad c_S + \sum_{i=1}^{r} p(S = s_i \mid \mathbf{e}) \cdot ECR(\mathbf{s}_{m_i} \mid \mathbf{e} \cup S = s_i). \tag{4}$$

The lemma is simple to prove using induction over the tree structure of a troubleshooting strategy (starting from the leaves of strategy $\mathbf{s}$).

An optimal strategy given evidence $\mathbf{e}$ will be denoted by $\mathbf{s}^\star(\mathbf{e})$. Please note that the order in which the previous troubleshooting steps are performed (so that an evidence $\mathbf{e}$ is achieved) has no impact on the optimal strategy $\mathbf{s}^\star(\mathbf{e})$. We will abbreviate $ECR(\mathbf{s}^\star(\mathbf{e}) \mid \mathbf{e})$ to $ECR^\star(\mathbf{e})$. If $\mathbf{e} = \emptyset$, then we will simply write $\mathbf{s}^\star$.

Definition 4 The *troubleshooting task* is to find a troubleshooting strategy $\mathbf{s}^\star$ such that for all possible strategies s it holds that

$$ECR(\mathbf{s}^\star) \quad \leq \quad ECR(\mathbf{s}).$$

1.3 Various setups of troubleshooting

It is often reasonable to assume that if the device is malfunctioning, then there is exactly one fault in the device (*single fault assumption*) [3]. Otherwise the situation is referred to as *multiple faults*. When the faults are independent, we speak about *independent faults* [9]. A solution of the *troubleshooting task* can be easily found in the case of *independent actions* when every action solves just one fault and all actions are pairwise independent.

In the case of *independent actions* with *single fault assumption* it suffices to order actions decreasingly according to the ratio $\frac{p(A_i = yes)}{c_{A_i}}$ (see [4]). In the case of *independent actions* with *independent faults* an ordering according to $\frac{p(A_i=yes)}{c_{A_i}\cdot(1-p(A_i=yes))}$ leads to an optimal sequence [9].

The task becomes harder if some actions may solve more than one fault. This case is referred to as *dependent actions*. In any case we assume that all actions and questions are pairwise independent given the faults. Please note that these assumptions can be expressed by different Bayesian Network structures. The *single fault assumption* leads to a model with one single root node, the *problem node*, having all faults as its states (see Figure 1), while the model of *independent faults* has omitted the problem node.

In the next section we shall analyze the complexity of troubleshooting. We will prove the NP-hardness of general troubleshooting with dependent actions under either single fault assumption or independent faults. If every action solves at most two causes, we conjecture the troubleshooting is also NP-hard even if we show a polynomially solvable special case.

2 Complexity of troubleshooting

To prove that the troubleshooting with dependent actions is NP-hard we reduce the *exact cover by 3-sets* to troubleshooting.

> **Exact cover by 3-sets:** We are given a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of subsets of a set U, such that $|U| = 3m$ for any integer m, and $|S_i| = 3$ for all i. We are asked if there are m sets in $\mathcal{S}$ that are disjoint and have U as their union. The proof of NP-completeness can be found for example [5].

2.1 Reduction

Let us have an *exact cover by 3-sets* input $\langle \mathcal{S}, U \rangle$. We construct a troubleshooting task as follows:

- For each element i in U we construct a fault F_i in the troubleshooter.
- For each set $S_i \in \mathcal{S}$ we assign $\mathcal{F}_i$, a set of faults corresponding to the elements of S_i. We construct an action A_i that solves faults in $\mathcal{F}_i$ with probability 1 and others with probability 0.
- All faults are equally probable; the probability of any fault is $p(F_i) = \frac{1}{3\cdot m}$.
- All actions have cost 1.
- The cost of the Call Service action CS is high enough to ensure it is not used in any good strategy. We set it to $c_{CS} = 3 \cdot m \cdot (m + 1)$. We will see later that this is high enough.

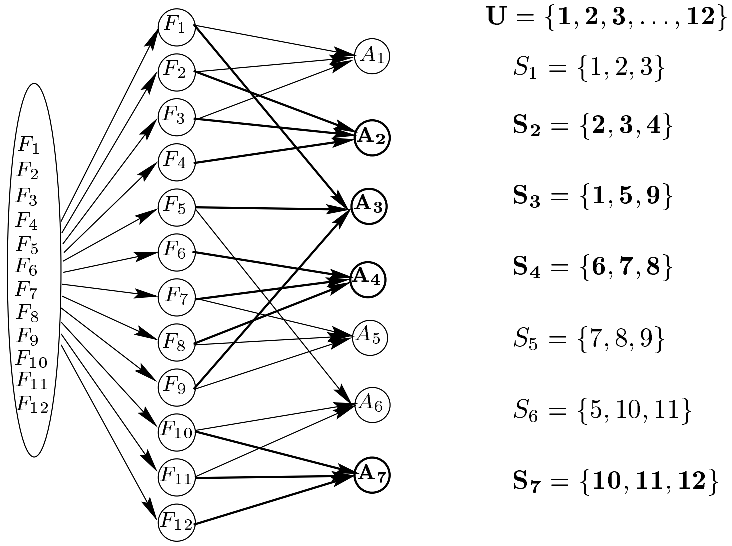

Figure 3: Troubleshooting model of the *exact 3-sets cover*. Every action corresponds to one set, the bold-face sets are the exact cover and corresponding actions belong to the best troubleshooting sequence.

2.2 Single fault

We aim to prove that the *exact cover by 3-sets* exists if and only if there is a troubleshooting sequence $\mathbf{s}$ with $ECR(\mathbf{s}) \leq \frac{m+1}{2}$. We have to prove three lemmas.

Lemma 2 (Basic properties) *The single fault assumption implies*

$$p(F_1 = yes \vee \ldots \vee F_r = yes) = \sum_{i=1}^{r} p(F_i = yes) = \frac{r}{3m},$$

therefore $p(F_1 = no, \ldots, F_r = no) = 1 - \sum_{i=1}^{r} p(F_i = yes) = \frac{3m-r}{3m}$. *Since* $p(A_i \mid F_j \in \mathcal{F}_i) = 1$ *we have*

$$p(A_1 = no, \ldots, A_r = no) = 1 - \sum_{F \in \cup_{i=1}^{r} \mathcal{F}_i} p(F = yes) = 1 - \frac{|\bigcup_{i=1}^{r} \mathcal{F}_i|}{3m} \quad (5)$$

Proof. The proof comes from the basic probability calculus. □

Lemma 3 *If we have an* exact cover by 3-sets $V = \{S_{j_1}, \ldots, S_{j_m}\}$, *then the corresponding action sequence* $A_{j_1}, \ldots, A_{j_m}$ *(in any order) has*

$$ECR(A_{j_1}, \ldots, A_{j_m}) = \frac{m+1}{2}.$$

Proof. No two actions solve the same fault and each action solves 3 faults, therefore $\frac{|\bigcup_{i=1}^{r} \mathcal{F}_i|}{3m} = \frac{3r}{3m}$. Substituting this to (5) we get:

$$p(A_1 = no, \ldots, A_r = no) = 1 - \frac{3r}{3m} = \frac{m-r}{m}$$

Now we calculate the ECR using formula 2 (the penalty does not apply since the sequence of actions solves the problem for sure):

$$ECR(A_{j_1}, \ldots, A_{j_m}) = \sum_{i=1}^{m} p(A_{j_1} = no, \ldots, A_{j_{(i-1)}} = no) \cdot c_i.$$

Please, recall that all actions have cost 1. Therefore:

$$ECR(A_{j_1}, \ldots, A_{j_m}) = \sum_{i=1}^{m} \left(\frac{m-(i-1)}{m} \right) \cdot 1 = \frac{(m+1) \cdot m}{2 \cdot m} = \frac{m+1}{2}.$$

□

Lemma 4 *If we have a troubleshooting sequence* $\mathbf{s}$ *solving the troubleshooting task defined by reduction 2.1, then* $ECR(\mathbf{s}) \geq \frac{m+1}{2}$. *If there exists an action* A_b *addressing only two or fewer unsolved faults (i.e. violating the disjunction of the corresponding 3-sets) then the* $ECR(A_1, \ldots, A_b, \ldots, A_r)$, *is strictly greater than* $\frac{m+1}{2}$.

Proof. If a strategy $\mathbf{s}$ contains a call service action at the end, then at least one fault F was not addressed since all addressed faults are solved with probability 1. Therefore:

$$ECR(\mathbf{s}) \geq p(\text{CS is performed}) \cdot c_{CS} \geq p(F) \cdot c_{CS} = \frac{1}{3m} \cdot 3 \cdot m \cdot (m+1) > \frac{m+1}{2}.$$

Thus we may consider only sequences that are certain to solve the problem without calling the service.

The probability of coming into the first step of troubleshooting is 1. Since this step solves at most 3 faults, the probability of coming into the second step is at least $1 - \frac{3}{3m} = 1 - \frac{1}{m}$. Similarly, the probability of coming into the i^{th} step is at least $1 - \frac{(i-1)}{m}$. Now we insert these estimates into the ECR calculation:

$$\begin{aligned} ECR(A_1, \ldots, A_r) &= \sum_{i=1}^{r} p\left(A_1 = no, \ldots, A_{i-1} = no\right) \cdot c_i \\ &\geq \sum_{i=1}^{m} \left(1 - \frac{(i-1)}{m}\right) = m - \frac{1}{m} \cdot \sum_{i=0}^{m-1} i = m - \frac{1}{m} \cdot \frac{(m-1) \cdot m}{2} = \frac{m+1}{2}. \end{aligned}$$

If there exists an action A_b solving only two or fewer faults, we need at least $m+1$ steps to be certain to solve the problem. The probability of coming into the step i, $i > b$ is greater than or equal to $\left(1 - \frac{2+3\cdot(i-2)}{3\cdot m}\right)$, so we conclude:

$$ECR(A_1, \ldots, A_b, \ldots, A_r) \geq \sum_{i=1}^{b}\left(1 - \frac{(i-1)}{m}\right) + \sum_{i=b+1}^{m+1}\left(1 - \frac{2+3\cdot(i-2)}{3\cdot m}\right)$$
$$= \frac{m+1}{2} + \frac{1}{3m} + \left(\frac{1}{3} - \frac{b+1}{3m}\right) \geq \frac{m+1}{2} + \frac{1}{3m} > \frac{m+1}{2}$$

since $m \geq b$. □

Theorem 1 *Suppose we are given a troubleshooting problem with dependent actions, single fault assumption, and a constant $K \in \Re^+$. The decision whether there exists a troubleshooting sequence* $\mathbf{s}$ *with $ECR(\mathbf{s}) \leq K$ is a NP-complete problem*[1].

Proof. The problem is NP: if we guess a good sequence $\mathbf{s}$, we calculate $ECR(\mathbf{s})$ and compare whether $ECR(\mathbf{s}) \leq K$. To calculate $ECR(\mathbf{s})$ of a sequence is polynomial time.

To prove the problem is NP-hard we reduce the *exact cover by 3-sets* to troubleshooting. Lemmas 3 and 4 show us that there exists a troubleshooting sequence with $ECR(\mathbf{s}) \leq \frac{m+1}{2}$ if and only if there exists an *exact cover by 3-sets*. Exact cover is a well known NP-hard problem so the troubleshooting is also NP-hard. □

2.3 Independent faults

If we consider that more faults can occur simultaneously, we must define a model of their dependencies. The simplest model is to assume the faults to be independent. As compared to troubleshooting with single fault assumption, this leads to different values of expected cost of repair but to similar complexity and solution methods as troubleshooting with single fault assumption.

We aim to prove the NP-hardness of troubleshooting with dependent actions and independent faults.

Notation: If we have a uniform distribution on $3m$ faults, the probability of any fault being present is $\frac{1}{3m}$, therefore the probability that the fault is not present is $p(\neg F) = \frac{3m-1}{3m}$.

Troubleshooting is initiated only if there is evidence of system failure. The probability of system failure is $p_0 = p(\text{system is faulty}) = 1 - p(\neg F)^{3m}$.

We denote ECR_M the expected cost of repair based on the independent faults assumption. It has different values from the ECR based on the single fault case.

The lemmas for independent faults look similar to the lemmas for a single fault.

[1] Of course, for $K < \min\{c_i\}$ or $K > \sum c_i$ the solution is trivial. The theorem says that any **general** algorithm solving these problems is NP-hard

Lemma 5 *If we have an* exact cover by 3-sets $V = \{S_{j_1}, \ldots, S_{j_m}\}$, *then the* ECR_M *of the corresponding action sequence* $A_{j_1}, \ldots, A_{j_m}$ *(in any order) is*

$$ECR_M = \frac{m}{1 - p(\neg F)^{3m}} - \frac{1}{p(\neg F)^{-3} - 1}.$$

Proof. The probability of coming into step $(k+1)$ is equal to one minus the probability of all not-checked faults not being present. Since no two actions solve the same fault and since in step k exactly $3k$ faults were checked, $3m - 3k$ faults remain. The probability of coming into step $(k+1)$ given system failure is $\frac{1-p(\neg F)^{3m-3k}}{p_0}$.

$$ECR_M(A_{j_1}, \ldots, A_{j_m}) = \sum_{i=1}^{m} p(A_{j_1} = no, \ldots, A_{j_{(i-1)}} = no) \cdot c_{j_i}$$

Therefore

$$\begin{aligned} ECR_M(A_{j_1}, \ldots, A_{j_m}) &= \sum_{k=0}^{m-1} \frac{1 - p(\neg F)^{3m-3k}}{p_0} \cdot 1 \\ &= \frac{m}{1 - p(\neg F)^{3m}} - \frac{1}{p(\neg F)^{-3} - 1}. \end{aligned}$$

□

Lemma 6 *If we have a troubleshooting sequence* $\mathbf{s}$ *solving the troubleshooting task defined by reduction 2.1, then* $ECR_M(\mathbf{s}) \geq \frac{m}{1-p(\neg F)^{3m}} - \frac{1}{p(\neg F)^{-3}-1}$. *If there exists an action* A_b *addressing only two or fewer unsolved faults, then* $ECR_M(A_1, \ldots, A_b, \ldots, A_r)$ *is strictly greater than* $\frac{m}{1-p(\neg F)^{3m}} - \frac{1}{p(\neg F)^{-3}-1}$.

Proof. The probability of coming into the first step of troubleshooting is 1. Since this step solves at most 3 faults, the probability of coming into the second step is at least $\frac{1}{p_0}(1 - p(\neg F)^{3m-3})$. Similarly, the probability of coming into the i^{th} step is at least $\frac{1}{p_0}(1 - p(\neg F)^{3m-3(i-1)})$. Now we insert these estimates into the ECR_M calculation:

$$\begin{aligned} ECR_M(A_1, \ldots, A_r) \geq & \sum_{i=1}^{r} p\left(A_1 = no, \ldots, A_{i-1} = no\right) \cdot c_i \\ = & \frac{1}{p_0} \cdot \sum_{k=1}^{m} \left(1 - p(\neg F)^{3m-3(k-1)}\right) \\ = & \frac{m}{1 - p(\neg F)^{3m}} - \frac{1}{p(\neg F)^{-3} - 1}. \end{aligned}$$

If there is an action A_b solving only two or fewer faults, we need at least $m+1$ steps to be certain to solve the problem. The probability of coming into step i, $i > b$ is greater than or equal to $\frac{1}{p_0}(1 - p(\neg F)^{3m-3(i-2)-2})$ so we conclude:

$$ECR_M \quad (A_1, \ldots, A_b, \ldots, A_r) \geq$$

$$
\begin{aligned}
&\geq \quad \frac{1}{p_0}\cdot\left[\sum_{k=1}^{b}\left(1-p(\neg F)^{3m-3(k-1)}\right)+\sum_{k=b+1}^{m+1}\left(1-p(\neg F)^{3m-3(k-2)-2}\right)\right]\\
&> \quad \frac{m}{1-p(\neg F)^{3m}}-\frac{1}{p(\neg F)^{-3}-1}.
\end{aligned}
$$

□

Theorem 2 *Given a troubleshooting problem with dependent actions and independent faults assumption and a constant $K \in \Re^+$, the decision whether there exists a troubleshooting sequence* $\mathbf{s}$ *with $ECR(\mathbf{s}) \leq K$ is a NP-complete problem.*

Proof. Again, we check the correct sequence in polynomial time.

Lemmas 5 and 6 show us that any *exact cover by 3-sets* problem may be reduced to troubleshooting. □

2.4 Polynomial problems

Troubleshooting with one fault per action is known to be polynomially solvable for both single fault assumption and independent faults [10]. There are other troubleshooting tasks solvable in polynomial time, too.

Theorem 3 *Let us assume a troubleshooting with n faults $F_1, \ldots, F_n$ and m actions $A_1, \ldots, A_m$. Each action can solve* **one or two** *faults with probability* 1*, the rest with probability* 0*. We have either single fault assumption or independent faults. All actions have equal cost* 1*, all faults have equal prior probability. There is a polynomial $O(n^5)$ algorithm to find the optimal sequence with minimal ECR (either ECR or ECR_M).*

We reduce this task to the *maximal matching* problem. The *maximal matching* can be solved in the time $O(n^5)$ (i.e. polynomial).

> **Maximal matching problem:** A matching in a graph G is a subset of its edges such that no two edges meet the same vertex. The task is in a given graph to find a matching of maximum cardinality. The algorithm is described in [2].

For a given troubleshooting problem we construct a *maximal matching* problem as follows (see Figure 4). We create a node for every fault, for each action that solves two faults we construct an edge between the nodes representing these faults. The *maximal matching* solution gives us a list of edges, we choose actions corresponding to these edges and for every unsolved fault we choose one (any) action solving this fault. This is a troubleshooting sequence with minimal ECR for both single fault and independent faults assumption. The proof is similar to the proofs of lemmas 3,4,5, and 6, therefore it is omitted.

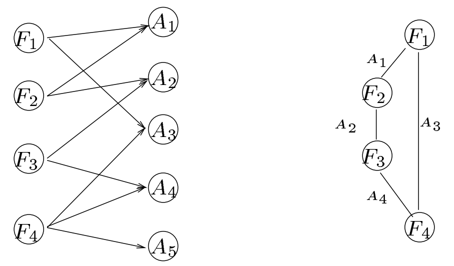

Figure 4: Max. two faults per action troubleshooting task and corresponding *maximal matching* problem.

2.5 Two faults per action troubleshooting

Let as assume each action to solve at most two actions and any distribution on faults. This case is more difficult. The straightforward reduction to the *optimal (weighted) matching* does not work.

> **Optimal matching problem:** Let $G = (V, E)$ be a complete graph K_{2n} with non-negative weight function w on the edges. The task is to find a matching M with the maximal sum of weights of the edges in M. This problem is polynomially solvable.

2.5.1 Reduction

Let us try to define a reduction of the troubleshooting problem with one or two faults per action to optimal matching. For every fault in the troubleshooting we create a node; for every action solving two faults we create an edge between the nodes corresponding to its faults. We search for a function mapping the probabilities of faults to the weights on edges.

Lemma 7 *There does not exist any function $\Re^2 \to \Re$ that would extend reduction 2.5.1 to a reduction mapping an optimal troubleshooting sequence to the best optimal matching for all 2-troubleshooting tasks.*

Proof. The example in Figure 5 shows a troubleshooting model with two different probability distributions of faults. The second distribution has the same values for the first four faults; but it has $F_5 = 0.01$, $F_7 = 0.04$ instead of $F_5 = 0.04$, $F_7 = 0.01$. Each distribution leads to a different optimal troubleshooting sequence. The action A_2 is left out in the first sequence and is present in the second one even though the faults solved by A_2 have exactly the same probabilities in both cases. $\square$

We proved that troubleshooting with uniform distribution on faults, one or two faults per action and the probabilities zero or one is polynomially solvable. We conjecture the general troubleshooting with one or two faults per action is NP-hard.

There is a similar couple of problems already known. The 2SAT problem is known to be polynomial whereas MAX2SAT is NP-hard [5].

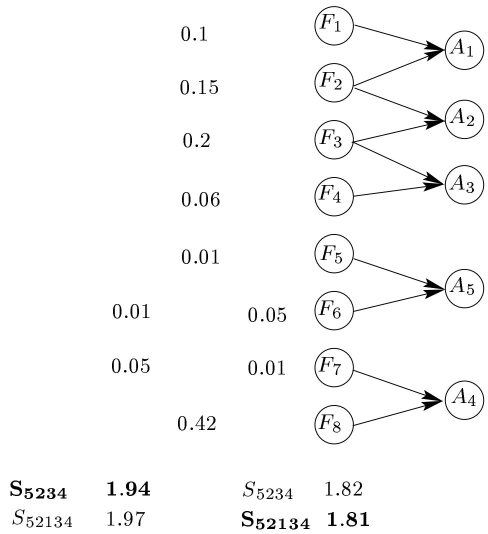

Figure 5: Two troubleshooting models with different best sequences but the same corresponding maximal matching problem. The numbers in the index of sequence $\mathbf{s}$ refer to actions applied in the given order, i.e. $\mathbf{s}_{5234}$ denotes sequence of actions $\{A_5, A_2, A_3, A_4\}$ and $\mathbf{s_{52134}}$ sequence $\{A_5, A_2, A_1, A_3, A_4\}$.

3 Search for an optimal strategy

In the previous section we proved that the general troubleshooting task is NP-hard, which means that there is no polynomial algorithm providing an optimal strategy. Nevertheless, some heuristics may direct the search so that we get an optimal strategy within a reasonable time. This section summarizes our efforts in this direction.

Every troubleshooting problem can be represented by a *decision tree*. Since the order in which troubleshooting steps are performed does not matter, identical subtrees can be collapsed, so that we get a *coalesced decision tree*. All decision trees in the following text will be coalesced. The coalesced decision tree for the Light Print Example is given in Figure 6. For simplicity, action A_3 is omitted so that the troubleshooting problem consists of two actions A_1, A_2 and one question Q_1 only.

The decision tree of a troubleshooting problem represents all possible troubleshooting strategies. As compared with a single strategy there is one more node type - the *decision node*. Every decision node corresponds to the decision that is made when the next troubleshooting step is being chosen. Decision nodes

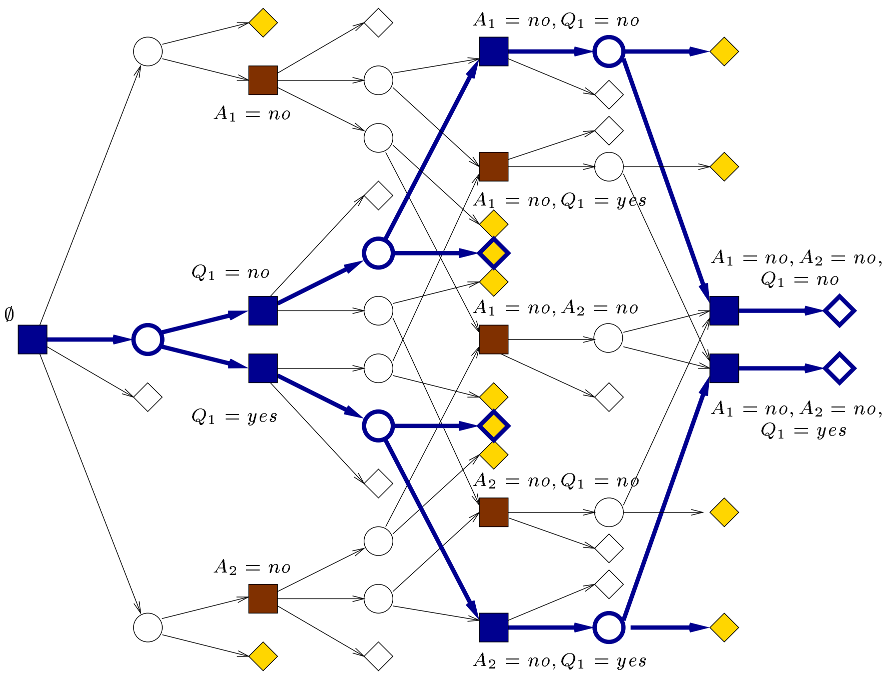

Figure 6: Coalesced decision tree corresponding to the troubleshooting problem consisting of two actions and one question. One troubleshooting strategy is highlighted

are denoted by squares. In Figure 6, one possible troubleshooting strategy is highlighted. There are 28 different troubleshooting strategies represented by the decision tree in Figure 6. We have omitted the chance node labels in the figure. Decision nodes are labeled by corresponding evidence.

3.1 Correspondence to AND/OR graphs

Every decision tree of a troubleshooting problem can be interpreted as an *AND/OR graph*. An AND/OR graph of a troubleshooting problem has the same nodes and edges as the corresponding decision tree. The only difference is the interpretation assigned to the nodes. Again there are three types of nodes: *OR nodes*, *AND nodes*, and *terminal nodes* in an AND/OR graph. The chance nodes of the decision tree correspond to the AND nodes, since all their children must be included in a troubleshooting strategy. The decision nodes correspond to the OR nodes, since exactly one of their children must be included in a troubleshooting strategy. Finally, terminal nodes have equivalent interpretation in both graph types.

Proposition 1 *A subgraph* $\mathbf{s}$ *of the AND/OR graph corresponds to a trou-*

bleshooting strategy if it fulfills the following conditions:

- $\mathbf{s}$ *is a tree with the same root* ϑ *as the AND/OR graph*
- *If* n *is an OR node of* $\mathbf{s}$*, then exactly one of its children belongs to* $\mathbf{s}$.
- *If* n *is an AND node of* $\mathbf{s}$*, then all its children belong to* $\mathbf{s}$ *as well.*
- *Every leaf* n *of* $\mathbf{s}$ *is a terminal node.*

In [6] the conditions listed above define the *solution tree* of an AND/OR graph. Consequently, if the AND/OR graph corresponds to the decision tree, then there is one-to-one correspondence between the troubleshooting strategies and the solution tree. Therefore algorithms used for solving AND/OR graphs can be used to find optimal troubleshooting strategies. Three algorithms will be discussed in Section 3.

The fact that every troubleshooting strategy is equivalent to a solution tree of the AND/OR graph corresponding to the troubleshooting problem enables us to use a straightforward application of algorithms for solving problems represented by AND/OR graphs. In particular, the search for an optimal troubleshooting strategy is equivalent to the search for the cheapest solution tree of the AND/OR graph corresponding to a troubleshooting problem.

3.2 Depth-first search

We have implemented two versions of the *depth-first search* algorithm - with and without memory. The depth-first search algorithm without memory corresponds to the search in the not coalesced decision tree. In this case equivalent subtrees are searched through many times.

If the search is performed in the coalesced decision tree and the minimal values of $ECR^\star(\mathbf{e}_n)$ are stored for every expanded node n, then the complexity of the search is substantially reduced. Please note that when using the recurrent formula 4 to compute $ECR^\star(\mathbf{e}_n)$ for any node n, we need to know the $ECR^\star$ values of the children of n in the optimal strategy $\mathbf{s}_n^\star$ only. See Table 5 where the number of explored nodes for runs on three troubleshooting problems for the two versions of the depth-first search are compared.

depth-first search	$\lvert\mathcal{A}\rvert = 6, \lvert\mathcal{Q}\rvert = 2$	$\lvert\mathcal{A}\rvert = 9, \lvert\mathcal{Q}\rvert = 3$	$\lvert\mathcal{A}\rvert = 12, \lvert\mathcal{Q}\rvert = 3$
without memory	5,060	130,328	476,191
with memory	374	4,354	16,881

Table 5: Number of expanded nodes for the depth-first search algorithms with and without memory

Another option is to reverse the search from *top-down* to *bottom-up* and systematically evaluate all nodes using the recurrent formula 4 again. Such an algorithm may be understood as an application of *dynamic programming* [1].

In spite of the substantial reduction due to the storage of the $ECR^\star$ values of the explored subtrees, the complexity of the depth-first search algorithm with memory is still very high, $\mathcal{O}(2^{|\mathcal{A}|+|\mathcal{Q}|})$. The question arises whether we can successfully apply classical heuristic search algorithms. Next, we will discuss two algorithms: a branch & bound algorithm and the $A^\star$ algorithm generalized for AND/OR graphs (in [6] it is called $AO^\star$ algorithm). For both algorithms a heuristic function, namely an estimate of $ECR^\star$, is essential.

3.3 Heuristic function

We propose a heuristic function that exploits the conditional independence of all actions and questions given the fault of the device. This is the assumption already used when building the Bayesian Network. Please recall that for every $F \in \mathcal{F}$ strategy $\mathbf{s}^\star(\mathbf{e} \cup F = yes)$ denotes an optimal strategy given the evidence $\mathbf{e} \cup F = yes$ and $ECR^\star(\mathbf{e} \cup F = yes)$ provides the ECR of this strategy.

Definition 5
Let $\mathbf{e}$ denote the evidence compiled so far. The function $\underline{ECR} : \mathbf{e} \mapsto \Re^+$ is defined by

$$\underline{ECR}(\mathbf{e}) \quad = \quad \sum_{F \in \mathcal{F}} p(F = yes \mid \mathbf{e}) \cdot ECR^\star(\mathbf{e} \cup F = yes). \tag{6}$$

The following theorem claims that the estimate is optimistic, i.e. its values are always lower than the true values of $ECR^\star$. This fact allows the application of the estimate into the heuristic search methods.

Theorem 4 *The function $\underline{ECR}(\mathbf{e})$ is an optimistic estimate of ECR of an optimal troubleshooting strategy given evidence* $\mathbf{e}$.

Proof. Let the root of $\mathbf{s}^\star(\mathbf{e})$ be n, i.e. $\mathbf{e}_n = \mathbf{e}$ and $\mathcal{L}(\mathbf{e}_n)$ be an abbreviation of $\mathcal{L}(\mathbf{s}^\star(\mathbf{e}_n))$ denoting the set of leaves of strategy $\mathbf{s}^\star(\mathbf{e}_n)$. Using Definition 3 we can write:

$$\begin{aligned}
ECR^\star(\mathbf{e}) \quad &= \quad ECR(\mathbf{s}^\star(\mathbf{e}_n) \mid \mathbf{e}_n) \quad = \\
&= \sum_{\ell \in \mathcal{L}(\mathbf{e}_n)} p(\mathbf{e}_\ell \mid \mathbf{e}_n) \cdot (c(\mathbf{e}_\ell) + t(n,l)) \\
&= \sum_{\ell \in \mathcal{L}(\mathbf{e}_n)} \left(\sum_{F \in \mathcal{F}} p(\mathbf{e}_\ell \mid \mathbf{e}_n \cup F = yes) \cdot p(F = yes \mid \mathbf{e}_n) \right) \cdot (c(\mathbf{e}_\ell) + t(n,l)) \\
&= \sum_{F \in \mathcal{F}} p(F = yes \mid \mathbf{e}_n) \cdot \sum_{\ell \in \mathcal{L}(\mathbf{e}_n)} p(\mathbf{e}_\ell \mid \mathbf{e}_n \cup F = yes) \cdot (c(\mathbf{e}_\ell) + t(n,l)) \\
&= \sum_{F \in \mathcal{F}} p(F = yes \mid \mathbf{e}_n) \cdot ECR(\mathbf{s}^\star(\mathbf{e}_n) \mid \mathbf{e}_n \cup F = yes).
\end{aligned}$$

In the previous formula $ECR(\mathbf{s}^\star(\mathbf{e}_n) \mid \mathbf{e}_n \cup F = yes)$ provides the ECR of strategy $\mathbf{s}^\star(\mathbf{e}_n)$ which need not be optimal when having the given the evidence $\mathbf{e}_n \cup F = yes$. Let $\mathbf{s}^\star(\mathbf{e}_n \cup F = yes)$ be the optimal strategy given evidence $\mathbf{e}_n \cup F = yes$. It is obvious that for $F \in \mathcal{F}$

$$ECR(\mathbf{s}^\star(\mathbf{e}_n) \mid \mathbf{e}_n \cup F = yes) \quad \geq \quad ECR(\mathbf{s}^\star(\mathbf{e}_n \cup F = yes) \mid \mathbf{e}_n \cup F = yes)$$

and consequently the following inequality holds:

$$ECR^\star(\mathbf{e}) \quad \geq \quad \sum_{F \in \mathcal{F}} p(F = yes \mid \mathbf{e}) \cdot ECR^\star(\mathbf{e} \cup F = yes) \quad = \quad \underline{ECR}(\mathbf{e}),$$

which corresponds to the assertion of the theorem. $\square$

A basic advantage of this estimate is that computation of $ECR^\star(\mathbf{e} \cup F = yes)$ does not require expensive operations if the actions are independent given the fault of the device. If the fault is known then there are usually only few actions that may solve the problem, i.e. $p(A = yes \mid F = yes) > 0$. For every fault $F \in \mathcal{F}$ the actions that are not contained in the evidence $\mathbf{e}$ are ordered according to $p(A = yes \mid F = yes)/c_A$ which gives an optimal strategy for the given evidence $\mathbf{e} \cup F = yes$. Please note that the value of the ECR estimate is computed using the conditional probabilities $p(A = yes \mid F = yes)$ that are available already from the original model. In the light of the new evidence $\mathbf{e}$ it is necessary to update the probabilities of the faults $p(F)$ only.

We have performed experiments on nine different models to see how far the values of the lower bound were from the optimal values of $ECR^\star(\mathbf{e})$. The quality of the estimate was measured by relative difference with respect to the optimal value of ECR

$$\varrho_k = 100 \cdot \frac{ECR^\star(\mathbf{e}_k) - \underline{ECR}(\mathbf{e}_k)}{ECR^\star(\mathbf{e}_k)}.$$

For every tested model we computed the average value $\bar{\varrho}$ over all nodes in the decision tree. The estimate was reasonably good for most models, the average relative difference was $\bar{\varrho} = 10\%$. For one tested model the value of $\bar{\varrho}$ was very good, 2.5%, and for two models it was quite poor, $\bar{\varrho} = 45\%$ and $\bar{\varrho} = 43\%$.

3.4 Branch & bound algorithm

Our implementation of branch & bound algorithm performs depth first search with pruning. The temporarily best $ECR'(\mathbf{e}_n)$ is stored for every expanded decision node n. Pruning of an edge coming from n is performed as soon as it is certain that any strategy that include the edge can not be the optimal one. Next, we describe the pruning formally.

Let n be a decision node, and m one of its children in the decision tree. Let $children(m) = \{m_1, \ldots, m_r\}$ be the set of children of node m corresponding to evidence $\mathbf{e}_{m_i} = \mathbf{e}_m \cup S = s_i$ where S is a troubleshooting step with outcomes $\{s_1, \ldots, s_r\}$. Further assume that the value of $ECR^\star(\mathbf{e}_m)$ has already been computed for the children $m_1, \ldots, m_q$ and for the remaining children $m_{q+1}, \ldots, m_r$

the value of $\underline{ECR}(\mathbf{e}_m)$ is provided. The edge from n to m is pruned immediately after it is realized that

$$\begin{aligned} ECR'(\mathbf{e}_n) \quad < \quad c_S \quad & + \sum_{i=1}^{q} P(S = s_i \mid \mathbf{e}_n) \cdot ECR^\star(\mathbf{e}_{m_i}) \\ & + \sum_{i=q+1}^{r} P(S = s_i \mid \mathbf{e}_n) \cdot \underline{ECR}(\mathbf{e}_{m_i}). \end{aligned}$$

Since the applied function $\underline{ECR}$ is an optimistic estimate of $ECR^\star$, the optimum can not be missed.

3.5 AO* algorithm

The $AO^\star$ algorithm starts in the root ϑ and successively marks a single new node. Set of marked nodes constitute a tree, by $\mathcal{M}$ we will denote the leaves of the marked tree. For each node $n \in \mathcal{M}$ we compute an optimistic estimate of the ECR for the remaining troubleshooting strategy. In our implementation the function $\underline{ECR}$ is used. By $children(\mathcal{M})$ we will denote the set of all non-marked children of marked nodes from $\mathcal{M}$. The algorithm exploits the function $\underline{ECR}$ to decide which of the nodes in $children(\mathcal{M})$ to mark next:

1. All nodes $n \in children(\mathcal{M})$ are evaluated using $\underline{ECR}(\mathbf{e}_n)$
2. Estimates of all considered strategies are updated using the new values of $\underline{ECR}$
3. Among all considered strategies, the cheapest one $\mathbf{s}'$ is chosen.
4. A node $m \in children(\mathcal{M})$ belonging to $\mathbf{s}'$ is marked.

There are several versions of the $AO^\star$ algorithm. They differ in the criteria for selection in step 4. For example, the node n having the biggest difference of the lower and upper bounds for $ECR^\star(\mathbf{e}_n)$ can be chosen since its expansion may bring the most information to restrict the interval for the true value of ECR.

It is a consequence of a well-known property of $A^\star$ algorithms that if an optimistic estimate of ECR is used, then the first fully marked strategy is optimal. No matter the efficiency of the $AO^\star$ algorithm, it only enabled us to determine optimal strategies for domains with less than 20 troubleshooting steps.

4 Approximative methods

In Section 2 we showed that the search of an optimal troubleshooting strategy is NP-hard. In Section 3 we have discussed several heuristic algorithms that turned out to be applicable only if the domain is sufficiently small. Therefore methods that provide reasonably good troubleshooting strategies in real-time seem to be necessary. The basic idea of both methods discussed below is that a local computation is performed whenever a new troubleshooting step has to be chosen. For both methods it is assumed that the single fault assumption holds.

4.1 BATS troubleshooter

The Bayesian Automated Troubleshooting System (BATS) was developed in the Laboratory for Normative Systems, within a joint project of the Hewlett-Packard Company and Aalborg University. The approach, which we are not going to discuss in detail here, exploits several heuristics based on the p/c ratio. For further information, see [8].

4.2 LB method

The *Lower Bound method (LB method)* is an algorithm that selects as its next troubleshooting step a step corresponding to node n that minimizes the lower bound estimate of $ECR^\star(\mathbf{e}_n)$, as defined in Definition 5. Formally speaking, a suboptimal troubleshooting strategy $\mathbf{s}$ is being constructed by use of a recursive procedure *expand* described in Table 6. Initially, the procedure is called with the root node ϑ of the decision tree. The resulting strategy is the tree defined by the set of nodes V and the set of edges E.

$K := children(node)$;
$m := \arg\min_{k \in K} \underline{ECR}(\mathbf{e}_k)$;
$V := V \cup m$;
If node m is not a terminal node, then
    for every node $\ell \in children(m)$:
        $edge := m \rightarrow \ell$;
        $expand(\ell)$;
        $E := E \cup edge$;

Table 6: Procedure $expand(node)$

The procedure is simple and does not require any expensive computation since the value of $\underline{ECR}(\mathbf{e}_k)$ can be provided efficiently as proposed in Section 3. The quality of the suboptimal strategies is increasing with decreasing values of $ECR^\star(\mathbf{e}_k) - \underline{ECR}(\mathbf{e}_k)$ decreasing. If for all expanded nodes k would hold that $ECR^\star(\mathbf{e}_k) = \underline{ECR}(\mathbf{e}_k)$, then we have the optimal strategy.

4.3 Comparison of approximations with optimal solution

Strategies obtained by use of the suboptimal methods described above were compared with the optimal solutions. The comparisons were performed for nine models designed by domain experts for troubleshooting of laser printers. The algorithm used to find the optimal solutions was depth-first search with memory, discussed in Section 3. All algorithms were implemented in C^{++}, partly by us and partly by a DAT3 student group within a student project at Aalborg University [7].

In Table 7 the ECR of the optimal troubleshooting strategy is compared with strategies provided by: the BATS troubleshooter, the LB method, and the sequence of actions ordered according to step by step updated p/c. Please note that some models are designed so that a given question must be performed first, which we have respected within all four approaches. Details are provided in [11].

Problem\Method	optimum	BATS	LB	P/C
53	433.24	442.39	478.79	444.54
Tray	129.21	129.21	132.34	155.10
Overrun	106.20	112.35	120.02	116.80
Load	38.38	38.42	41.25	43.05
PaperJam	124.32	124.37	131.39	300.85
Scatter	115.41	115.86	129.92	236.58
NotDupl	70.67	75.03	77.06	121.10
Spots	161.38	162.25	187.95	286.75
MIO1	250.45	253.31	343.26	479.96
Relative difference with respect to optimal ECR		1.81%	12.69%	59.16%

Table 7: Summary comparison of ECR for different approaches.

The ECR values of the strategies provided by the BATS troubleshooter are very close to the optimal values. The strategies of the LB method were poorer but still reasonably good. Thus, in the case of the troubleshooting of laser printers it turns out that good troubleshooting strategies can be provided in real time despite the fact that the search for an optimal strategy is NP-hard.

5 Conclusions

In this paper we have proved that the general troubleshooting task is NP-hard. We plan to study the complexity of different troubleshooting problems in detail. We have already proved that the troubleshooting problem is NP-hard even if the actions are independent while the costs are not fixed but functions of previously performed actions. The troubleshooting with questions and independent actions is NP-hard as well. Even though the precise border between polynomial and NP-hard troubleshooting problems is not known, we were able to narrow the uncertain area.

The proposed heuristic methods used to find an optimal strategy are applicable only to domains with a limited number of actions and questions. On the other hand, the suboptimal strategies provided by the two tested methods (BATS and LB) are not far from the optimum.

Acknowledgments

We would like to thank Finn Verner Jensen for inspiring us to work on the discussed problem and for many valuable comments on this paper. We are grateful to Claus Skaanning for the detailed explanation of the BATS troubleshooter approach and to Regitze Larsen for the careful reading of the paper.

References

[1] R. Bellman and S. Dreyfus. *Applied Dynamic Programming.* Princeton University Press, Princeton, N.J., 1962.

[2] Jack Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[3] D. Heckerman, J. S. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57, 1995.

[4] J. Kalagnanam and M. Henrion. A comparison of decision analysis and expert rules for sequential analysis. In P. Besnard and S. Hanks, editors, *The Fourth Conference on Uncertainty in Artificial Intelligence*, pages 271–281, New York, 1988.

[5] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, 1994.

[6] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Publishing Company, Inc., 1984.

[7] E.B. Rasmussen, M.H. Pedersen, M.K. Christiansen, S.T. Pedersen, and T. Glaesner. How to pull a solution tree out of a Bayesian Network (DAT3 report). Aalborg University, Department of Computer Science, 1999.

[8] C. Skaaning, F. V. Jensen, and U. Kjærulff. Printer troubleshooting using Bayesian Networks. In *the Thirteenth International Conference on Industrial & Engineering Applications of AI & Expert Systems*, 2000.

[9] S. Srinivas. A polynomial algorithm for computing the optimal repair strategy in a system with independent component failures. In *Proc. of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 515–522, Montréal, Canada, 1995.

[10] S. Srinivas. A polynomial algorithm for computing the optimal repair strategy in a system with independent component failures. In *Proc. of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 515–522, Montréal, Canada, 1995.

[11] J. Vomlel. On quality of BATS troubleshooter and other approximative methods. Technical report, The Research Unit of Decision Support Systems at Aalborg University, 2000.