ELSEVIER

Reliability Engineering and System Safety 80 (2003) 49–62

RELIABILITY ENGINEERING & SYSTEM SAFETY

www.elsevier.com/locate/ress

Decision theoretic troubleshooting of coherent systems

Helge Langseth*, Finn V. Jensen

Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg, Denmark

Received 5 June 2002; accepted 8 October 2002

Abstract

We present an approach to efficiently generating an inspection strategy for fault diagnosis. We extend the traditional troubleshooting framework to model non-perfect repair actions, and we include questions. Questions are troubleshooting steps that do not aim at repairing the device, but merely are performed to capture information about the failed equipment, and thereby ease the identification and repair of the fault. We show how Vesely and Fussell's measure of component importance extends to this situation, and focus on its applicability to compare troubleshooting steps. We give an approximate algorithm for generating a 'good' troubleshooting strategy in cases where the assumptions underlying Vesely and Fussell's component importance are violated, and discuss how to incorporate questions into this troubleshooting strategy. Finally, we utilize certain properties of the domain to propose a fast calculation scheme.
© 2003 Elsevier Science Ltd. All rights reserved.

Keywords: Repair strategies; Bayesian networks; Fault diagnosis; Vesely and Fussell component importance

1. Introduction

This paper describes a troubleshooting system which has been developed in the SACSO[1] project, and which is partly implemented in the BATS[2] tool. This is a troubleshooting (TS) system for performing efficient troubleshooting of electro-mechanical equipment, and it is currently employed in the printer domain. It is important to notice that the BATS tool is created to offer *printer users* a web-based interface to a decision-theoretic TS-system; it is not intended exclusively for maintenance personnel who are trained to handle the equipment that is to be repaired. The goal is that any user, however, inexperienced, should be able to repair the failed equipment on his own instead of relying on professional help. By design, the TS-system we describe, therefore, differs from other TS-systems [1–7] in several aspects. Most importantly the users of the TS-system may be inexperienced with handling and repairing the failed equipment. Hence, they may fail to repair broken components, e.g. by seating a new network card incorrectly. Furthermore, this may even happen without the user realizing the mistake. It is, therefore, crucial for the TS-system to explicitly include the possibility that users perform prescribed repair actions incorrectly in the TS-model.

Secondly, the users are expected to have limited knowledge about (and interest in) the design of the malfunctioning equipment. They cannot be expected to be interested in finding the *cause* of a problem; they merely want to *repair* it. Focusing on the identification of the faulty minimal cutset, as in Refs. [4–7], is therefore not expected to be relevant for the foreseen group of users. The troubleshooting will thus be terminated as soon as the equipment is repaired; that is, we assume that the user is satisfied with a *minimal repair* of the failed equipment. *Perfect repair* is not necessarily accomplished by using our TS-system (and not by the methods in Refs. [4–7] either), but may be considered using other means.

Finally, as the faulty device can be located under a variety of external conditions, the TS-system can pose *questions* in order to survey the faulty equipment's surroundings. Although these questions initially increase the cost of the troubleshooting, they may shed light on the situation, and ultimately decrease the overall cost of repairing the equipment.

* Corresponding author. Present address. Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway.

E-mail addresses: helgel@math.ntnu.no (H. Langseth), fvj@cs.auc.dk (F.V. Jensen).

[1] The SACSO (Systems for Automated Customer Support Operations) project constitutes joint work between the Research Unit for Decision Support Systems at Aalborg University and Customer Support R&D at Hewlett-Packard.

[2] BATS (Bayesian Automated Troubleshooting System) is available from Dezide over the internet: http://www.dezide.dk/

0951-8320/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
PII: S0951-8320(02)00202-8

To formalize, let the faulty equipment consist of K components $\chi = \{X_1, \ldots, X_K\}$. Each component is either *faulty* ($X_i =$ faulty) or *operating* ($X_i =$ ok). The equipment consists of R minimal cut sets (MCSs), and we use $\mathscr{C} = \{\mathscr{C}_1, \ldots, \mathscr{C}_R\}$ for the collection of these. A MCS is *faulty* ($\mathscr{C}_i =$ faulty) if all its members are faulty. Otherwise it is *operating* ($\mathscr{C}_i =$ ok). The equipment is assumed to be faulty at the time when troubleshooting starts; troubleshooting is terminated as soon as the equipment is brought back to operating modus. We will assume that only one MCS is in its faulty state, and use $\mathscr{C}_F$ to denote the faulty MCS (named *the actual MCS* in Ref. [4]). This assumption is common for most TS-systems, and it is usually justified by considering systems that are used almost continuously, and thus (like a printer) tested frequently. It is unlikely that several components should fail approximately, simultaneously. Common cause failures (due to, e.g. stroke of lightning, pouring coffee into the printer, etc.) are easily detected, and are handled separately. Note that if more than one MCS is faulty the proposed method will still repair the equipment, although not necessarily in an optimal fashion.[3] The TS-system may choose from a set of N possible actions $\mathscr{A} = \{A_1, \ldots, A_N\}$ to remedy the problem. There are also M predefined questions $\mathscr{Q} = \{Q_1, \ldots, Q_M\}$ that may be posed. The goal of a TS-system is to provide a 'good' *TS-strategy*. Formally, a strategy $\mathscr{S}$ is called a TS-strategy if it describes the process of repeatedly performing the next step in the strategy until the equipment is repaired or all steps have been performed. A *TS-step* is a step in a TS-strategy, either a repair step (termed action) or an information-gathering step (termed question). To each TS-step B_i the associated cost is denoted by C_i. The system is informed about the outcome of each TS-step after it has been performed.

Any TS-strategy can be represented by a strategy tree, see Fig. 1, for an example. The internal nodes in the strategy tree (depicted as ovals) represent chance nodes; TS-steps that we do not know the outcome initially. Each possible outcome of a chance node corresponds to a unique sub-tree in the strategy tree, which is found by selecting the edge labelled with that particular outcome. If, e.g. $Q_S = q$, the strategy tree in Fig. 1 prescribes to perform action A_2; if $Q_S = \neg q$, then the question Q_K should be posed. The terminal nodes (depicted as diamonds) signify that the troubleshooting strategy has ended, either because the problem is solved or because the set of actions has been exhausted.

How 'good' a TS-strategy is, is judged by its *expected cost of repair* (ECR). This is in-line with the decision-theoretic formulation of the troubleshooting task: one should balance the cost of a TS-step with the likelihood of the step to be beneficial, so that the optimal TS-strategy can be found [3]. Breese and Heckerman [9] used Bayesian networks (BNs) to model the troubleshooting domain, and Jensen et al. [10] report extensions to that framework. In Refs. [9,10], the domains under study were restricted to be serial systems, i.e. systems where all cutsets were singletons. In this paper we will extend these frameworks to work with any coherent system (represented by its cutsets).

The printer industry spends millions of dollars every year on customers support; mainly to provide telephone-support and on-site troubleshooting. This has sparked an interest for building automated troubleshooting systems which can resolve some of the printer users' problems without requiring support from call agents. A printing system consists of several components: The application from which the printing command is sent, the printer driver, the network connection, the server controlling the printer, the printer itself, etc. It typically has about 40 different failure-modes, e.g. *Light print*. Each failure-mode can be caused by several component failures, and we have one TS-system for each of them.[4] The typical size of these TS-models is about 30 actions and 15 questions. We will not describe the printer model in further detail, as the TS-system we propose is general in nature; the interested reader is referred to Refs. [10,11].

The rest of the paper is outlined as follows. In Section 2, we describe the basic system model, and the formal language used to describe it. Section 3 is devoted to how the TS-system sequences actions, and handling of questions are described in Section 4. The calculation scheme is described in detail in Section 5, and we conclude in Section 6.

2. The troubleshooting model

In this section, we will describe the troubleshooting model, and in particular focus on the modelling assumptions that we make. To do so, we start by introducing BNs, which constitute the representation language we employ. We then give a detailed description of how we generate a BN-representation of the troubleshooting domain.

2.1. Bayesian networks

Our system represents the TS-domain by a BN [12,13]. BNs have a long history of usage in the reliability and safety sciences, ranging from the early works [14,15] to the more recent contributions [8–11,16–21]. BNs offer a flexible language to describe the TS-model, and we utilize this to make a realistic model of the interactions one can have with the failed equipment; specifically we can define repair steps

[3] Srinivas [8] presents a modified algorithm to handle troubleshooting in serial systems where more than one component (and hence more than one MCS) may be faulty; it turns out that optimal troubleshooting in this case requires a balance between cost and probability of successful repair that is different from what is optimal in our situation. When presented with a system where more than one MCS is in its faulty state, our system may thus perform sub-optimally: The user may be asked to perform the repair in a way more expensive than had been required if we had not made this assumption.

[4] The first information the user enters into the system is the failure-mode he wants to troubleshoot. If some failure-modes are not easily distinguishable we have joined them into one TS-model.

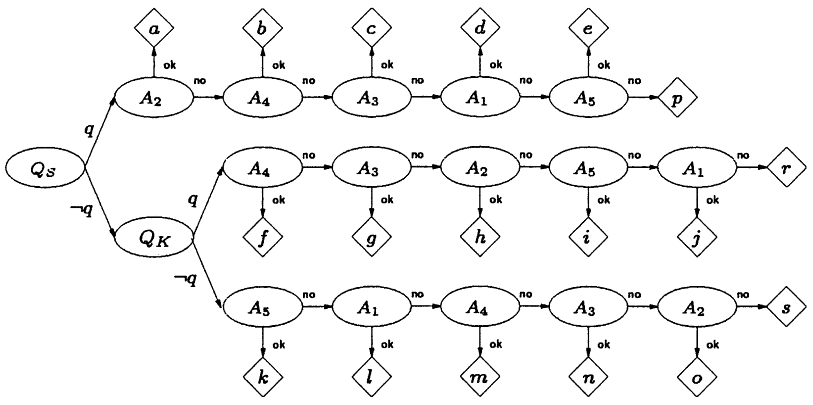

Fig. 1. A TS-strategy represented by a strategy tree.

including non-perfect repair, as well as information-gathering steps.

A BN describing a multivariate random variable $\mathbf{X}$ is a compact representation of the distribution function $P(\mathbf{X} = \mathbf{x})$. A BN consists of a qualitative part; a directed acyclic graph, and a quantitative part; a set of conditional probability distributions. More formally, a BN representing the distribution function of a stochastic vector $\mathbf{X}$ is a 2-tuple $(\mathcal{G}, \mathcal{P})$. $\mathcal{G}$ is a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes in the graph and $\mathcal{E}$ is the set of directed edges. There is bijection between the random variables $\mathbf{X}$ and the nodes in $\mathcal{V}$, and the edges are used to represent dependence between the variables of $\mathbf{X}$ is annotated with $\mathcal{P}$, a set of conditional probability distributions over the domain $\mathcal{V}$. In the TS-domain we only work with *discrete* BNs, where each node $V \in \mathcal{V}$ takes on values from a finite state-space denoted $\mathrm{sp}(V)$. We define the *parent set of* V, $\mathrm{pa}(V)$, as the set of nodes having outgoing edges directed into V. The graph is annotated with the probability distributions $\mathcal{P}$ s.t. each node $V \in \mathcal{V}$ is associated with a conditional probability table $P(V|pa(V))$. The full joint distribution over the variables $\mathcal{V}$ can now be calculated as $P(\mathcal{V}) = \prod_{V\in\mathcal{V}} P(V|pa(V))$.

The essential property of the distribution function that is utilized in the BN representation of $P(\mathbf{X} = \mathbf{x})$ is the set of *conditional independencies* encoded in the distribution function: If $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{W}$ are vectors of random variables with joint probability distribution $P(\mathbf{Y}, \mathbf{Z}, \mathbf{W})$, then we say that $\mathbf{Y}$ is conditionally independent of $\mathbf{Z}$ given $\mathbf{W}$, written $\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{W}$, if $P(\mathbf{Y}|\mathbf{Z}, \mathbf{W} = \mathbf{w}) = P(\mathbf{Y}|\mathbf{W} = \mathbf{w})$ for all $\mathbf{w}$ where $P(\mathbf{W} = \mathbf{w}) > 0$. If $\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\{\emptyset\}$, then $\mathbf{Y}$ and $\mathbf{Z}$ are (marginally) independent (written $\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}$ for short).

An example of conditional independence from our domain is as follows. If the toner is low, then this can be detected in at least two ways: (i) There may be an error message on the control panel, and (ii) the last page may be printed lightly. There is a slight possibility for the error message not to show up, and for the last page not be visibly light-printed, even when the toner is low. If we learn that the last page was printed lightly, we may assume this was because the toner is low, and that will in turn increase our belief in finding the error message on the control panel; hence these two events are not (marginally) independent. On the other hand, if we know that the toner is low, then information about a message on the control panel will not change our belief regarding the last page being light-printed. The two events are conditionally independent given the toner's status.

2.2. The basic troubleshooting model

The faulty equipment and the effect of interactions between the repair personnel and this equipment are modelled in a BN. As our starting point we use a BN model of the system generated from the MCS representation (see Ref. [21] for how this translation can be done). This part of the BN is denoted the *system layer* in Fig. 2; the system layer is the part of the BN that mimics the fault tree in Fig. 3. Note that we have introduced a *constraint node*[5] L to enforce the assumption that exactly one MCS is in its faulty state. Next, the MCSs are modelled by logical functions, such that $\mathcal{C}_i = \text{faulty}$ if and only if all the components in the MCS are in the faulty state. Hence, $\mathrm{pa}(\mathcal{C}_i)$ are exactly those components that are members of the cutset $\mathcal{C}_i$, and $P(\mathcal{C}_i|\mathrm{pa}(\mathcal{C}_i))$ is used to encode this deterministic relationship. Note that the cutset nodes of the system layer are not really required to encode the equipment model; the probabilistic relationship could have been encoded in the constraint node L. There are, however, at least two reasons to include the cutset nodes in the model: firstly, reliability engineers are used to working with the notion of cutsets, and including the cutsets explicitly makes the model more understandable and easier to build. Secondly, including the cutset nodes in the model typically

[5] A constraint node is a node which is used to enforce other variables into specific configurations. In the example model we use L to enforce that exactly one cutset is faulty. This is done by defining $L = \text{yes}$ if exactly one of the cutsets is faulty and $L = \text{no}$ otherwise. The evidence $\{L = \text{yes}\}$ is entered into the system before the calculations are performed, and the cutset nodes are thereby constrained s.t. the MCS assumption is fulfilled.

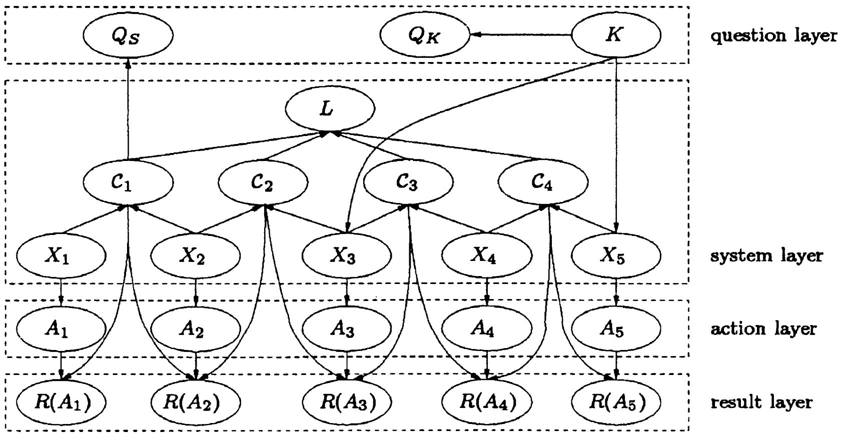

Fig. 2. The BN representation of the example model. Note that this model is extremely simple; more complex models in which, e.g. an action A_i can repair more than one component can easily be defined.

makes the overall model more compact (i.e. the total number of required parameters is reduced).[6] $P(X_\ell = \text{faulty})$ is given as the a priori probability for the component to have failed, i.e. the probability unconditioned on the equipment failure. After a *propagation* in the BN (see Ref. [22] for a description of how this is done) the posterior probability for a component failure given that the equipment is faulty (enforced by using the constraint node) can be read off the node representing that component in the BN, and the probability for each MCS to be the actual MCS can be found in the corresponding nodes.

Next, the system model is extended by an explicit model of the effect of the interaction between the equipment and the repair personnel. These interactions are limited to the predefined sets of actions $\mathcal{A}$ and questions $\mathcal{Q}$. First, we look at how the actions are modelled (see the *action layer* in Fig. 2).

Actions are connected to the system layer by making them children of the components they can repair, that is, $\text{pa}(A_i) \subseteq \mathcal{X}$. We explicitly describe the joint effect an action A has on all the components it can repair. This is done by extending the state space of A. For the state space we use the notation $+rX$ for the event that A repairs X and $-rX$ otherwise; note that this notation is unconditioned on the state of X. For example, see Fig. 4, where action A can repair the components X_k and X_ℓ. Then, $\text{pa}(A) = \{X_k, X_\ell\}$, and the state-space of A is $\text{sp}(A) = \{+rX_k + rX_\ell, +rX_k - rX_\ell, -rX_k + rX_\ell, -rX_k - rX_\ell\}$. Without referring to $\text{sp}(A)$ we use the notation $\{A^{\downarrow X} = \text{yes}\}$ for the event that A repairs X, and $\{A^{\downarrow X} = \text{no}\}$ otherwise. Thus, in the current example the shorthand $\{A^{\downarrow X_k} = \text{yes}\}$ denotes the event $\{A = +rX_k + rX_\ell \vee A = +rX_k - rX_\ell\}$.

We make a number of assumptions about the TS-domain. Some are made to simplify the model definition, whereas others turn out to be beneficial when we perform calculations in the BN:

- We disregard component failure induced by trouble-shooting personnel;[7] note that this is related to the assumption that only one MCS is faulty.
- By construction of the model it is made sure that an action only can repair components in its parent set, $P(A^{\downarrow X} = \text{yes} | X = \text{faulty}) = 0$ whenever $X \notin \text{pa}(A)$.
- The state of a component X_ℓ does not influence the user's ability to repair component X_k, $A^{\downarrow X_k} \perp\!\!\!\perp X_\ell | X_k, k \neq \ell$. That is, we assume, for instance, that it is not more difficult to replace an MIO card when the toner cartridge is faulty than it would have been had the toner cartridge been operating.
- If we were to receive information about a user's failure to perform one repair action, then this would not influence our beliefs about his ability to perform other actions. Thus, when the user fails to repair some component we assume it is due to 'bad luck' and not 'clumsiness'. Formally, we write $A_i^{\downarrow X_k} \perp\!\!\!\perp A_j^{\downarrow X_\ell} | \{X_k, X_\ell\}$ whenever $i \neq j$. This assumption requires that the group of users is homogeneous. In real-world applications, where we typically have 'novice' and 'expert' users, it can be beneficial to maintain two TS-systems; one for the 'novices' and one for the 'experts'.
- We use the convention that an action cannot repair a component that is already operating, $P(A^{\downarrow X} = \text{yes} | X = \text{ok}) = 0$. This may seem counterintuitive, but we use $A^{\downarrow X} = \text{yes}$ to denote the event that the user has *improved* the system, it is not used to describe the *state* of the system.

These assumptions suffice for the TS-system to be operational, and for the calculation scheme (Section 5) to work. For simplicity we may also make the additional

[6] If we choose not to include the cutset nodes in the BN representation we can, at the cost of a larger model, relax the binary system-model we employ. This can be utilized to create multi-state systems to, e.g. model 'degrees of failure'. We have nevertheless chosen to work with the MCS representation, primarily to render the fast calculation scheme of Section 5 possible.

[7] In the models underlying the BATS tool we have increased the cost of an action to partly reflect the *risk* of performing it. If the probability of introducing new component failures into the domain is high, then the risk is high, and the cost will be increased to reflect this potential danger.

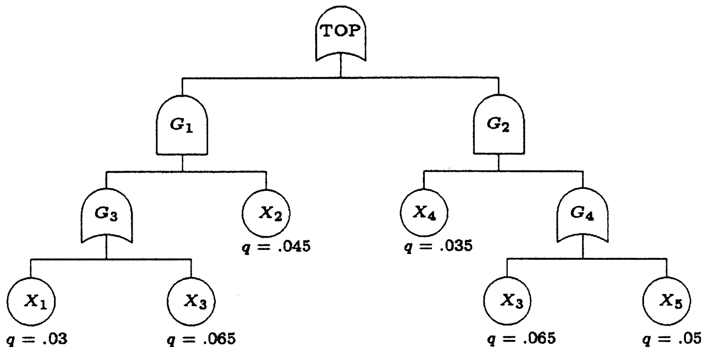

Fig. 3. A fault tree describing our example model.

assumption that $A^{\downarrow X_k} \perp\!\!\!\perp A^{\downarrow X_\ell} | \{X_k, X_\ell\}$ whenever $k \neq \ell$. This means that a conditional probability $P(A|\text{pa}(A))$ is fully specified by the collection of probabilities $\{P(A^{\downarrow X_k} = \text{yes}|X_k = \text{faulty}) : X_k \in \text{pa}(A)\}$; this is often referred to as restrictive *independence of causal influence* [23]. Hence, if A can repair t components, then it is enough to enter only t conditional probabilities to describe $P(A|\text{pa}(A))$. This should be compared to the 2^t numbers needed if this independence assumptions had not been made. Note that we do not require the repair actions to be perfect; non-perfect repair is modelled by $P(A^{\downarrow X_\ell} = \text{yes}|X_\ell = \text{faulty}) = \gamma$, $0 \leq \gamma < 1$.

There is an important difference between what is modelled in the action layer and what is actually observed. The action layer describes the events $\{A^{\downarrow X} = \text{yes}|X = \text{faulty}\}$, however, we may only observe whether the equipment is repaired or not, i.e. if the event $\{A^{\downarrow X} = \text{yes} \wedge X \in \mathscr{C}_F\}$ occurs. To be able to work with the actual observations as evidence, we extend the model with a *result layer* consisting of a set of nodes $R(A)$, one for each $A \in \mathscr{A}$. $R(A)$, the *result of* A at the system level, is defined as $R(A) = \text{ok}$ if $A^{\downarrow X} = \text{yes}$ for some $X \in \mathscr{C}_F$ and $R(A) = \text{no}$ otherwise.

The probability that action A repairs the equipment, $P(R(A) = \text{ok})$, naturally extends Vesely and Fussell's measure of component importance[8] [24] when A can repair only one component X. Let $I^{\text{VF}}(X)$ be defined as the probability for X to be *critical*, i.e. $X \in \mathscr{C}_F$, given that the equipment is faulty. Then

$$P(R(A) = \text{ok}) = P(X \in \mathscr{C}_F)P(A^{\downarrow X} = \text{yes}|X = \text{faulty})$$
$$= I^{\text{VF}}(X)P(A^{\downarrow X} = \text{yes}|X = \text{faulty}). \quad (1)$$

When A can repair a set of components, we have (with a slight abuse of notation)

$$P(R(A) = \text{ok}) = P\left(\bigvee_{X \in \mathscr{X}} \{A^{\downarrow X} = \text{yes} \wedge X \in \mathscr{C}_F\}\right)$$
$$= \sum_{\mathscr{C}_\ell \in \mathscr{C}} I^{\text{VF}}(\mathscr{C}_\ell) \coprod_{X \in \mathscr{C}_\ell \cap \text{pa}(A)} P(A^{\downarrow X} = \text{yes}|X = \text{faulty}),$$

where $I^{\text{VF}}(\mathscr{C}_\ell)$ is the probability that *all* components in $\mathscr{C}_\ell$ are critical, i.e. $I^{\text{VF}}(\mathscr{C}_\ell)$ equals the probability that $\mathscr{C}_\ell$ is the actual cutset.

Regarding questions, we distinguish between *symptom questions* and *configuration questions*. Symptom questions are used to examine possible failure manifestations; an example from the printer-domain is '`Does the printer test-page print correctly?`' These questions are designed to shed light on the fault at the cutset level, e.g. by trying to replicate the equipment's faulty modus in other slightly different situations. (If the test-page prints correctly the problem is probably related to the application generating the print job.) Symptom questions are connected to the MCS nodes in the domain, see the node Q_S in Fig. 2. The edges are pointing in the direction of the causal influence, i.e. from the MCS nodes to the questions. The parent set of a symptom question Q_S, $\text{pa}(Q_S) \subseteq \mathscr{C}$, determines the set of MCSs that directly influences the likelihood of the different answers to the question.

Configuration questions are used to uncover the environment in which the equipment is embedded, by trying to reveal any configuration settings that are applied. An example from our domain is '`What operating system do you use?`'. Configuration settings does not directly relate to a given MCS, but may change the likelihood for components to be operating. (If the operating system is Linux, the printing problem is not related to the Windows printer drivers.) The edges connecting a configuration node to the system layer are therefore directed from the configuration node to the components, see K in Fig. 2. The user may be unable to correctly answer questions regarding the configuration settings. The answer to a configuration question is therefore modelled as a random variable, see Q_K in Fig. 2. That is, we will receive

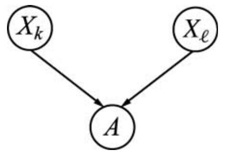

Fig. 4. Action A can repair both and X_k and X_ℓ.

[8] $I^{\text{VF}}(X)$ is defined as the probability that at least one minimal cutset which contains component X is faulty, given that the system is faulty. Under the assumptions in this paper this is simply $I^{\text{VF}}(X) = P(X \in \mathscr{C}_F)$.

information about Q_K (and not K directly) when the model is used, and Q_K is therefore needed explicitly in the model together with K.

2.3. Building the TS-models

The theme of this paper is to find a close to optimal TS-strategy in a *given* TS-model, but we will close this section by briefly touching upon the knowledge acquisition process used to generate the TS-model.[9] Building BNs has traditionally been considered such a complex task that BN experts would have to be deeply involved in the process. The BATS system consists of about 40 separate BN models, each representing a specific failure-mode. Some models are quite small, but the largest contains about 80 actions and 40 questions. To build these models we solely relied on a team of 6–7 experts. The number of models made it necessary to build a special tool for knowledge acquisition [25]. This tool, which is termed *BATS Author*, is designed to ensure that no knowledge about BNs is required to build the TS-models. The information required to generate the models can be specified in a terminology close to the experts' own, and the conditional probabilities can be expressed in the direction most natural for the expert. The BN structure is made s.t. the conditional independence statements encoded in the graph are easily verified. Skaanning [25] reports that all the models required to describe the failure-modes for another printer was built and validated in one man-month using this tool.

3. Action sequences

In this section we look at the situation where the only available troubleshooting steps are actions. In this case the TS-strategy is simply a *TS-sequence*, i.e. a string of actions performed one after another until the equipment is repaired. Let $\boldsymbol{\epsilon}$ denote arbitrary evidence collected so far during troubleshooting, i.e. a list of actions that all have failed to repair the equipment. To be more specific, we use $\mathbf{e}_j$ to denote the evidence that the first j actions in the sequence $\mathcal{S} = \langle A_1, \ldots, A_N \rangle$ have all failed to repair the equipment, $\mathbf{e}_j = \{R(A_i) = \text{no};\ i = 1, \ldots, j\}$. If A_k solves the problem with certainty, then $P(\mathbf{e}_k) = 0$, which reflects the fact that the TS-sequence is terminated after the kth step. Note that $\mathbf{e}_0 = \{\emptyset\}$, and $P(\mathbf{e}_0) = 1$ as the equipment is assumed to be faulty at the beginning of the troubleshooting.

The ECR of a troubleshooting sequence $\mathcal{S} = \langle A_1, \ldots, A_n \rangle$, where action A_i is allocated the cost C_i, the mean cost until an action succeeds or all actions have been performed:

$$\mathrm{ECR}(\mathcal{S}) = \sum_{i=1}^{N} C_i P(\mathbf{e}_{i-1}). \qquad (2)$$

A TS-sequence is said to be *optimal* if it achieves the minimum ECR of all TS-sequences. Note that it might be slightly misleading to use the term ECR as we consider a situation where a repair sequence may fail to repair the equipment (since some actions may be imperfect, and therefore fail to fix the critical components). Thus, a repair sequence may leave the equipment faulty, and the ECR is in this case the expected cost of *performing the sequence* and not of *repairing the equipment* (see the terminal nodes p, r and s in Fig. 1). The probability of a sequence failing to repair the equipment is however determined by the set $\mathcal{A}$ only, and does not depend on the sequencing of the actions. Hence, as we are only interested in finding the *cheapest sequence*, we will disregard this slight twist.[10] In this paper we focus our attention towards the cost of *performing* the TS-strategy, and we will continue to call this cost the ECR.

3.1. The greedy approach

Vesely and Fussell's component importance is commonly regarded as the best search heuristic when each component is repaired by a perfect action, and all repair actions have the same cost. Furthermore, when the costs are unequal the Vesely and Fussell's component importance can be scaled by the action's cost. The idea of using $I^{\mathrm{VF}}(\cdot)$ to sequence the actions generalizes to our situation, see Eq. (1), and we therefore define an action's *efficiency* in the following way:

Definition 1. Let $A \in \mathcal{A}$ be a repair action, let C_A be the cost of performing A, and let $\boldsymbol{\epsilon}$ be the evidence compiled so far during troubleshooting. The efficiency of A given $\boldsymbol{\epsilon}$ is defined as

$$\mathrm{ef}(A|\boldsymbol{\epsilon}) = \frac{P(R(A) = \text{ok}|\boldsymbol{\epsilon})}{C_A}.$$

The efficiency has an important property when verifying that a TS-sequence $\mathcal{S}$ is sub-optimal:

Proposition 2. *Let $\mathcal{S} = \langle A_1, \ldots, A_N \rangle$ be an optimal TS-sequence of actions for which the cost of each action is*

[9] This outline is based on Skaanning [25] and Jensen et al. [10]; further details can be found in those papers.

[10] If, on the other hand, we were interested in the monetary value of the *expected cost* of the cheapest sequence, our approach would be misleading. To work in such situations, Breese and Heckerman [9] propose to introduce a new action named *Call Service* as the final act in a TS-sequence. Performing this action will put the equipment back in operating modus, but presumably at a high cost since external personnel is involved in fixing the problem.

independent of the other actions taken. Then it must hold that $\mathrm{ef}(A_i|\mathbf{e}_{i-1}) \geq \mathrm{ef}(A_{i+1}|\mathbf{e}_{i-1})$.

Proof. Examine the two TS-sequences $\mathcal{S} = \langle A_1,\dots,A_i,A_{i+1},\dots,A_N\rangle$ and $\mathcal{S}' = \langle A_1,\dots,A_{i+1},A_i,\dots,A_N\rangle$. From Eq. (2) we get

$$\mathrm{ECR}(\mathcal{S}) - \mathrm{ECR}(\mathcal{S}') = (C_i P(\mathbf{e}_{i-1}) + C_{i+1}P(\mathbf{e}_{i-1}, R(A_i) = \mathrm{no})) - (C_{i+1}P(\mathbf{e}_{i-1}) + C_i P(\mathbf{e}_{i-1}, R(A_{i+1}) = \mathrm{no})),$$

hence, $\mathrm{ECR}(\mathcal{S}) - \mathrm{ECR}(\mathcal{S}') \leq 0$ iff

$$\frac{P(R(A_i) = \mathrm{ok}|\mathbf{e}_{i-1})}{C_i} \geq \frac{P(R(A_{i+1}) = \mathrm{ok}|\mathbf{e}_{i-1})}{C_{i+1}}. \qquad \square$$

Note that Proposition 2 can in general not be used to decide whether a TS-sequence $\mathcal{S}$ is *optimal*, it is merely a characterization of some sub-optimal sequences.

A direct corollary of Proposition 2 is that if action A_i has the highest efficiency amongst all remaining actions given the aggregated evidence $\boldsymbol{\epsilon}$, and no evidence $\boldsymbol{\epsilon}' \supset \boldsymbol{\epsilon}$ excluding A_i exists such that this changes, then it is optimal to perform A_i before any other action. Some situations where this formulation is useful is given in Proposition 3, which is a simple reformulation of [10, Proposition 1]:

Proposition 3.

(1) *The equipment has N components and N actions.*
(2) *There are no questions.*
(3) *Exactly one MCS is faulty.*
(4) *Each action has a specific probability of repairing the components. It is given by* $P(A_i^{\downarrow X_i} = \mathrm{yes}|X_i = \mathrm{faulty}) > 0$, $P(A_i^{\downarrow X_j}|X_j = \mathrm{faulty}) = 0$ *for* $i \neq j$.
(5) *The cost* C_i *of action* A_i *does not depend on the sequencing of the actions.*
(6) *The equipment is designed as a serial system, i.e. the MCSs are singletons:* $\mathcal{C}_i = \{X_i\}$, $i = 1,\dots,N$.

Then we have: If $\mathrm{ef}(A_j|\mathbf{e}_0) \leq \mathrm{ef}(A_k|\mathbf{e}_0)$ *then* $\mathrm{ef}(A_j|\boldsymbol{\epsilon}) \leq \mathrm{ef}(A_k|\boldsymbol{\epsilon})$, *where* $\boldsymbol{\epsilon}$ *is any evidence of the type 'Actions* $\mathcal{A}' \subseteq \mathcal{A}\setminus\{A_j,A_k\}$ *have failed'.*

Propositions 2 and 3 motivate the *greedy approach*:

Algorithm 1 (Greedy approach).

```
(1) For all $A_j \in \mathcal{A}$ Calculate $\mathrm{ef}(A_j|\mathbf{e}_0)$;
(2) Let $\mathcal{S}$ be the list of actions ordered according to $\mathrm{ef}(\cdot|\mathbf{e}_0)$;
(3) Return $\mathcal{S}$;
```

It follows that the greedy approach is optimal under the assumptions of Proposition 3. Note that it is not always optimal to sequence the actions based on the efficiencies. A counter-example is given below:

Example 4. Consider the domain described in Fig. 2 (with failure data from Fig. 3). We assume perfect repair actions, let $C_i = 1$ for all actions, and disregard the questions Q_S and Q_K. The greedy approach selects the sequence $\langle A_3,A_2,A_4\rangle$ with $\mathrm{ECR} = 1.58$. The optimal sequence found by exhaustive search is $\langle A_2,A_4\rangle$, with $\mathrm{ECR} = 1.47$. (Note that this result is not contradictory to Proposition 2; the efficiencies are calculated as $\mathrm{ef}(A_2|\mathbf{e}_0) = .529$, $\mathrm{ef}(A_3|\mathbf{e}_0) = .624$ and $\mathrm{ef}(A_4|\mathbf{e}_0) = .486$, hence it is in accordance with Proposition 2 to start with A_2 as long as it is not followed by A_3.)

An obvious attempt to improve the results of Example 4 is to recalculate the efficiencies each time new evidence comes in. In this way we make sure that all information available when the ith step is to be chosen is actually taken into account; recall that we use B_j to denote the jth step in the strategy $\mathcal{S}$.

Algorithm 2 (Greedy approach with recalculations).

```
(1) $\boldsymbol{\epsilon} \leftarrow \{\emptyset\}$; $\mathcal{A}' \leftarrow \{A_1,\dots,A_N\}$; $\mathcal{S} = \langle\cdot\rangle$;
(2) For $i = 1$ to $N$
    (a) For all $A_j \in \mathcal{A}'$ Calculate $\mathrm{ef}(A_j|\boldsymbol{\epsilon})$;
    (b) Select $A_k \in \mathcal{A}'$ s.t. $\mathrm{ef}(A_k|\boldsymbol{\epsilon})$ is maximized;
    (c) $B_i \leftarrow A_k$; $\mathcal{A}' \leftarrow \mathcal{A}'\setminus\{A_k\}$; $\boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon} \cup \{R(A_k) = \mathrm{no}\}$.
(3) Return $\mathcal{S}$;
```

Applied to the model in Example 4 this algorithm generates the sequence $\mathcal{S} = \langle A_3,A_4,A_2\rangle$ with $\mathrm{ECR} = 1.53$. This is better than the greedy approach, but still not optimal. A result similar to Proposition 3 can be shown for arbitrary sized but disjoint MCSs if we assume that all actions are perfect.

Proposition 5. *Let* $\mathcal{S} = \langle A_1,\dots,A_n\rangle$ *be a repair sequence for a troubleshooting problem fulfilling conditions* 1–5 *in Proposition 3. The MCSs are disjoint,* $\mathcal{C}_i \cap \mathcal{C}_j = \{\emptyset\}$, $i \neq j$, *and all repair actions are perfect, i.e.* $P(A_i^{\downarrow X_i} = \mathrm{ok}|X_i = \mathrm{faulty}) = 1$ *for* $i = 1,\dots,N$. *Let* $\mathcal{S}$ *be the output of Algorithm 2. Then* $\mathcal{S}$ *is an optimal repair sequence.*

It should be emphasized that the actions are assumed to be perfect in Proposition 5. When the actions are non-perfect, optimality is no longer assured, as can be seen from Example 6.

Example 6. Consider a TS-model with two cutsets $\mathcal{C}_1 = \{X_1,X_2\}$ and $\mathcal{C}_2 = \{X_3,X_4,X_5\}$. Let $P(X_i = \mathrm{faulty}) = 3\times10^{-6}$ for $i = 1,3,4,5$ and $P(X_2 = \mathrm{faulty}) = 7\times10^{-6}$. Each component X_i is repaired by a dedicated action A_i. Let the cost of the actions be $C_1 = 9$, $C_2 = 12$, and $C_i = 10$ for $i = 3,4,5$. Finally, $P(X_1^{\downarrow A_1} = \mathrm{ok}|X_1 = \mathrm{faulty}) = .9$, $P(X_i^{\downarrow A_i} = \mathrm{ok}|X_i = \mathrm{faulty}) = .98$ for $i = 2,3$, and $P(X_i^{\downarrow A_i} = \mathrm{ok}|X_i = \mathrm{faulty}) = .95$ for $i = 4,5$. Then Algorithm 2 returns $\mathcal{S}_1 = \langle A_5,A_1,A_3,A_4,A_2\rangle$ with $\mathrm{ECR}(\mathcal{S}_1) = 14.95$, whereas the optimal sequence is $\mathscr{S}_2 = \langle A_5, A_3, A_4, A_1, A_2 \rangle$ with $\mathrm{ECR}(\mathscr{S}_2) = 14.84$.

3.2. Dependent actions

The crucial step when optimality is proven in the setting of Proposition 3 is the fact that no evidence obtained during troubleshooting can change the ordering of the remaining actions under consideration; the residual probability mass, i.e. the probability $P(R(A_i) = \mathrm{ok} | \mathbf{e}_{i-1})$, is absorbed uniformly by all these actions. Hence, the initial ordering of two actions, $A_i \prec A_j$, say, cannot change when some new evidence $R(A_k) = \mathrm{no}$, $A_k \notin \{A_i, A_j\}$ arrives. In the general case, however, the ordering of a subset of actions $\mathscr{A}' \subset \mathscr{A}$ may depend on what evidence $\boldsymbol{\epsilon}$ is collected, even if $\boldsymbol{\epsilon}$ does not contain explicit information about any of the actions in $\mathscr{A}'$. We call this situation *dependent actions* [26].

A domain for which the cost of an action does not depend on the sequence of actions taken is said to have dependent actions whenever there exists actions A_i, A_j and A_k s.t.

$$\frac{\mathrm{ef}(A_i|\emptyset)}{\mathrm{ef}(A_j|\emptyset)} \neq \frac{\mathrm{ef}(A_i|R(A_k) = \mathrm{no})}{\mathrm{ef}(A_j|R(A_k) = \mathrm{no})}.$$

A domain has dependent actions if there exists two actions A_i and A_j s.t. $\mathrm{pa}(A_i) \cap \mathrm{pa}(A_j) \neq \emptyset$ or there exists two actions A_i and A_j, two components $X_k \in \mathrm{pa}(A_i)$ and $X_\ell \in \mathrm{pa}(A_j)$, and an MCS $\mathscr{C}_m$ s.t. $\{X_k, X_\ell\} \subseteq \mathscr{C}_m$. An example from the printer domain is the action-pair '`Reseat toner cartridge`' and '`Change toner cartridge`' as both may solve problems related to bad seating of the cartridge.

Examples 4 and 6 showed that Vesely and Fussell's component importance is not optimal, in general, when the domain has dependent actions. This is hardly a surprise, since the problem of finding an optimal troubleshooting strategy is known to be NP-hard in this case [27]. To try to improve a sub-optimal strategy we employ an adapted version of a standard algorithm for combinatorial optimization (similar to the algorithm presented by Norstrøm et al. [7]). This algorithm starts from an initial seed, and iteratively improves this sequence until it converges to a local optimum. Note that $B_k^{(i)}$ (Step 2a) denotes the kth TS-step in the action sequence $\mathscr{S}$ when starting the ith step of the iteration. Note also that the algorithm is said to converge (Step 3) when the ECR of the found sequence is not lower than the ECR of the sequence found previously.

Algorithm 3 (Discrete optimization).

```
(1) Initialization: $\mathscr{S} \leftarrow \langle B_1, \ldots, B_N \rangle$ for some ordering of $\mathscr{A}$;
(2) For $i = 1$ to $N$
    (a) For $j = i$ to $N$
            $\mathscr{R}_j \leftarrow \langle B_1^{(i)}, \ldots, B_{i-1}^{(i)}, B_j^{(i)}, B_i^{(i)}, \ldots, B_{j-1}^{(i)}, B_{j+1}^{(i)}, \ldots, B_N^{(i)} \rangle$;
    (b) Select $j_0 \in [i, \ldots, N]$ s.t. $\mathrm{ECR}(\mathscr{R}_{j_0})$ is minimized;
    (c) $\mathscr{S} \leftarrow \mathscr{R}_{j_0}$;
(3) If not converged then goto 2;
(4) Return $\mathscr{S}$;
```

A sequence $\mathscr{S} = \langle A_1, A_2, \ldots, A_i, \ldots, A_j, \ldots, A_N \rangle$ is a local optimum if, whenever we insert A_j before A_i ($j > i$) in $\mathscr{S}$ to obtain $\mathscr{S}' = \langle A_1, A_2, \ldots, A_{i-1}, A_j, A_i, \ldots, A_{j-1}, A_{j+1}, \ldots, A_N \rangle$, then $\mathrm{ECR}(\mathscr{S}) \leq \mathrm{ECR}(\mathscr{S}')$. It is obvious that Algorithm 3 converges to a local optimum since $\mathrm{ECR}(\mathscr{S})$ is guaranteed to be non-increasing after each loop of the algorithm (the algorithm can decide to stay put by selecting j_0 s.t. $\mathscr{R}_{j_0} = \mathscr{S}$ in Step 2b. It is, however, not guaranteed that the algorithm converges to the globally optimal sequence. The crucial choice to be made in Algorithm 3 is the initialization of $\mathscr{S}$ in Step 1. To ensure quick convergence to an approximately optimal solution, it can be beneficial to select a seed sequence that is close to the optimum. A natural choice is to initialize $\mathscr{S}$ as found by Algorithm 2. It is, however, easy to see that this sequence is a local optimum itself (confer Proposition 2), and it will, therefore, not be improved by Algorithm 3. Instead, we suggest to initialize the action sequence by ordering w.r.t. the *observation-based efficiency* (obef). We outline the derivation of the observation-based efficiency [26] below.

Consider a situation where the evidence $\boldsymbol{\epsilon}$ has been collected and it has been decided that the next action to perform is A. To calculate the observation-based efficiency, the TS-system should consider what information can be gained about the failed equipment by just getting to know that A does not solve the problem, and more importantly, the *value* of this information. It is natural to quantify this value as the difference in ECR between two degenerate models: (i) The TS-system where the collected evidence is $\boldsymbol{\epsilon}' = \{\boldsymbol{\epsilon}, R(A = \mathrm{no})\}$ and (ii) The TS-system where A has been made unavailable, but where the collected evidence is $\boldsymbol{\epsilon}'' = \boldsymbol{\epsilon}$. Assume that the sequence of remaining actions when given evidence $\boldsymbol{\epsilon}'$ is $\mathscr{S}(\boldsymbol{\epsilon}')$ and that the sequence of the actions when given evidence $\boldsymbol{\epsilon}''$ and A is unavailable is $\mathscr{S}(\boldsymbol{\epsilon}'')$. We define the *conditional* ECR *of the sequence* $\mathscr{S} = \langle A_1, \ldots, A_N \rangle$ given $\boldsymbol{\epsilon}'$ as $\mathrm{ECR}(\mathscr{S}|\boldsymbol{\epsilon}') = \sum_{j=1}^{N} C_j P(\mathbf{e}_{j-1}|\boldsymbol{\epsilon}')$. Finally, we define the value of the information contained in the event that $R(A) = \mathrm{no}$ given the current evidence $\boldsymbol{\epsilon}$ as

$$\mathrm{VOI}(R(A) = \mathrm{no}|\boldsymbol{\epsilon}) = \mathrm{ECR}(\mathscr{S}(\boldsymbol{\epsilon}')|\boldsymbol{\epsilon}') - \mathrm{ECR}(\mathscr{S}(\boldsymbol{\epsilon}'')|\boldsymbol{\epsilon}'),$$

i.e. $\mathrm{VOI}(R(A) = \mathrm{no}|\boldsymbol{\epsilon})$ is the difference of the expected cost of the strategies $\mathscr{S}(\boldsymbol{\epsilon}')$ and $\mathscr{S}(\boldsymbol{\epsilon}'')$. Note that both expected costs are calculated conditioned on $\boldsymbol{\epsilon}'$, the evidence actually collected as the two strategies are considered to be employed.

To recapitulate, we want to consider the value of information an action *that fails* has to offer when we determine how to sequence the actions. This amount is calculated as $\mathrm{VOI}(R(A) = \mathrm{no}|\boldsymbol{\epsilon})$, and we receive this gain with probability $P(R(A) = \mathrm{no}|\boldsymbol{\epsilon})$. If we regard this amount as a refund, it is natural to approximate the 'real' cost of action A as

$$\tilde{C}_A = C_A - P(R(A) = \text{no}|\boldsymbol{\epsilon})\text{VOI}(R(A) = \text{no}|\boldsymbol{\epsilon}),$$

where $\tilde{C}_A$ is the cost we 'spend' by performing A; $C_A - \tilde{C}_A$ is the expected reduction in ECR of the remaining sequence of action, which is obtained by learning that A fails. It is argued by Langseth and Jensen [26] that if one couples Definition 1 with Algorithm 2, one implicitly assumes that $\text{VOI}(R(A) = \text{no}|\boldsymbol{\epsilon}) = 0$. On the other hand, if $\tilde{C}_A$ is used as the cost of A in the efficiency calculation, this will change the troubleshooting strategy in a way that attempts to incorporate the actual value of the information we receive. This leads to the definition of the observation-based efficiency:

Definition 7. Let $A \in \mathscr{A}$ be a repair action, let the cost of A be C_A, and let $\boldsymbol{\epsilon}$ be the evidence compiled so far during troubleshooting (i.e. not containing A). Let $\text{VOI}(R(A) = \text{no}|\boldsymbol{\epsilon})$ be the value of information A will have if it fails (by altering the sequencing of the remaining actions). Then the observation-based efficiency of A given $\boldsymbol{\epsilon}$ is:

$$\text{obef}(A|\boldsymbol{\epsilon}) = \frac{P(R(A) = \text{ok}|\boldsymbol{\epsilon})}{C_A - P(R(A) = \text{no}|\boldsymbol{\epsilon})\text{VOI}(R(A) = \text{no}|\boldsymbol{\epsilon})}.$$

An algorithm that orders the actions according to the observation-based efficiency does in general not offer an optimal solution; a sequence ordered in this way may even violate the optimality check of Proposition 2. This is, however, of minor importance, as we only use the sequence as a seed to Algorithm 3 and do not regard it as a final solution on its own. Note however, that the probability update is proportional under the assumptions in Proposition 3, which means that $\text{VOI}(R(A) = \text{no}|\boldsymbol{\epsilon}) = 0$ in this case. The observation-based efficiency is therefore exact under the assumptions of Proposition 3. '`Cycle power`' is an example of an action from our domain which has high value of information. Power cycling repairs many temporal problems, and ruling these out can be very beneficial for the future troubleshooting.

A problem with Definition 7 is that $\text{VOI}(R(A) = \text{no}|\boldsymbol{\epsilon})$ cannot be calculated unless one is able to correctly sequence all remaining actions (after performing A) in order to calculate $\text{ECR}(\mathscr{S}(\boldsymbol{\epsilon}')|\boldsymbol{\epsilon}')$ and $\text{ECR}(\mathscr{S}(\epsilon'')|\boldsymbol{\epsilon}')$; a computationally prohibitive task. Langseth and Jensen discuss two approximations of $\text{VOI}(R(A) = \text{no}|\boldsymbol{\epsilon})$: one based on the Shannon entropy of the efficiencies of the remaining actions, and the computationally simpler approach to use the *myopic* ordering of actions (i.e. based on Definition 1), see Ref. [26] for details.

Table 1 shows results of a small simulation study. Three troubleshooting models have been used: the example model of Fig. 3 (with $N = 5$ actions and $R = 4$ cutsets), the CPQRA model [28] ($N = 25$, $R = 20$) and Norstrøm et al.'s example [7] ($N = 6$, $R = 4$). For each model the actions' costs and the failure probabilities of the components have been randomized. Additionally, the probability of an action to successfully repair a component in its parent set was randomly selected in the interval [0.9,1.0]. Then Algorithms 2 and 3 were run, and compared by difference in ECR.[11] The simulations were run for 500 iterations. The reported numbers give the relative number of times Algorithm 2 found a result inferior to that of Algorithm 3, the average relative difference in ECR in those runs (Avg. rel. diff.), and the maximum relative difference in ECR (Max. rel. diff.).

Table 1
Algorithms 2 and 3 are compared through a small simulation study

	Rel. num. (%)	Avg. rel. diff. (%)	Max. rel. diff. (%)
Example model of Fig. 3	8.2	4.0	7.5
The CPQRA model [28]	9.4	4.2	8.2
Norstrøm et al.'s example [7]	4.0	4.9	9.2

The results in Table 1 show that even for the relatively small models we have considered, a strategy generated by the Vesely and Fussell's component importance (Algorithm 2) fails fairly frequently, and the additional cost of following an inferior sequence may be considerable.

As it is NP-hard to find the optimal repair sequence, Algorithm 3 is not infallible; it may sometimes be stuck in sub-optimal solutions. This did, for instance, happen for the CPQRA model (Table 1), where Algorithm 3 even was inferior to Algorithm 2 in 1.2% of the simulations, with maximum relative cost difference equal to 2.1%.

4. Questions

When we add questions to our TS-model, the strategy is represented by a strategy tree, see Fig. 1. Note that the ECR cannot be calculated by Eq. (1) in this case, instead we use a recursive calculation scheme to compute the expected cost of repair.

Proposition 8. *Let $\mathscr{S}$ be a TS-strategy which starts with the step $B_{(1)}$ and then continues with the strategy conditioned on the possible outcomes of $B_{(1)}$. Then the* ECR *of $\mathscr{S}$ can be calculated recursively as*:

$$\text{ECR}(\mathscr{S}) = C_{(1)} + \sum_{b_{(1)} \in \text{sp}(B_{(1)})} P(B_{(1)} = b_{(1)})\text{ECR}(\mathscr{S}|B_{(1)} = b_{(1)}), \tag{3}$$

where $C_{(1)}$ is the cost of step $B_{(1)}$ and $\text{ECR}(\mathscr{S}|B_{(1)} = b_{(1)})$ *is the ECR of the sub-tree $\mathscr{S}$ following the branch for which $B_{(1)} = b_{(1)}$. The recursion is terminated by* $\text{ECR}(\emptyset|\cdot) = \text{ECR}(\cdot|R(A_j) = \text{ok}) = 0$.

[11] Algorithm 3 was initialized by the sequence obtained when the actions were ordered according to the observation-based efficiency; the value of information was approximated by employing a myopic strategy.

The obvious way to decide whether it pays to pose a question Q, is to calculate the value of information for that particular question. Let the strategy be defined as $\langle Q, \mathcal{S}\rangle$, where $\mathcal{S}$ is the optimal strategy conditioned on the answer to the question Q, and let $\mathcal{S}'$ be the optimal strategy when we are refused to pose Q. We define VOI(Q) as:

$$\mathrm{VOI}(Q) = \mathrm{ECR}(\mathcal{S}') - \sum_{q\in \mathrm{sp}(Q)} P(Q=q)\mathrm{ECR}(\mathcal{S}|Q=q).$$

The system should pose the question if $\mathrm{VOI}(Q) > C_Q$.

A problem with this approach is that we must correctly position all other questions in the strategy before we can calculate $\mathrm{ECR}(\mathcal{S}')$ and $\mathrm{ECR}(\mathcal{S}|Q=q)$; this will lead to a too expensive recursion. Breese and Heckerman [9] propose to use a *myopic* approach to this problem: Assume that it is sufficient to sequence only *actions* when VOI(Q) is to be calculated, i.e. one should consider the effect of the question Q only on the sequencing of actions, and disregard the effect of the other questions. The two action sequences $\mathcal{S}$ and $\mathcal{S}'$ are then approximated by ordering the actions according to their efficiencies by Algorithm 2. In Ref. [10] it is argued that this approach will over-rate the effect of the question, because one in this case only compares the effect of asking the question *now*, with $\mathrm{ECR}_{\mathrm{now}} = C_Q + \sum_{q\in\mathrm{sp}(Q)} P(Q=q)\mathrm{ECR}(\mathcal{S}|Q=q)$, or *never*, $\mathrm{ECR}_{\mathrm{never}} = \mathrm{ECR}(\mathcal{S}')$. The decision rule is to pose the question iff

$$\mathrm{ECR}_{\mathrm{now}} < \mathrm{ECR}_{\mathrm{never}}. \tag{4}$$

Jensen et al. [10] argue that one should also compare $\mathrm{ECR}_{\mathrm{now}}$ to the ECR of a strategy starting with what appears to be the best action, followed by Q, and thereafter a TS-sequence $\mathcal{S}''$, which depends on the outcome of Q. This approach has ECR given by

$$\mathrm{ECR}_{A,Q}(A,Q,\mathcal{S}'') = C_A + P(R(A)=\mathrm{no})C_Q + \sum_{q\in\mathrm{sp}(Q)} P(Q=q,R(A)=\mathrm{no})\mathrm{ECR}(\mathcal{S}''|Q=q,R(A)=\mathrm{no}).$$

The question should be posed iff

$$\mathrm{ECR}_{\mathrm{now}} < \min\{\mathrm{ECR}_{A,Q}(A,Q,\mathcal{S}''), \mathrm{ECR}_{\mathrm{never}}\}. \tag{5}$$

To emphasize the importance of including questions in the troubleshooter system we reproduce and extend a set of experimental results from Vomlel [29]. We have examined nine of the troubleshooter models included in the BATS tool; for each of them we calculated the ECR of the optimal TS-strategy, the ECR of the TS-strategy produced when combining Algorithm 2 with Eq. (5), the ECR of the TS-strategy produced when combining Algorithm 2 with Eq. (4), and finally the ECR of the TS-sequence generated from Algorithm 2 when questions were disregarded. The models we used for testing were moderately sized with N (number of actions) ranging from 6 to 16 and M (number of questions) in the interval from 2 to 10. The results clearly show how important the questions are in these real-life TS-models, and they also indicate that the approximations made in Algorithm 2 combined with the decision rule of Eq. (5) may be quite reasonable (Table 2).

Table 2
Empirical comparison of the effect of including questions into nine of the BATS TS-models. The results are extended from those reported by Vomlel [29]

N	M	Optimal	Algorithm 2 + Eq. (5)	Algorithm 2 + Eq. (4)	Algorithm 2
6	2	433.24	442.39	442.43	444.54
9	3	129.21	129.21	205.54	155.10
11	3	106.20	108.07	111.75	116.80
12	3	38.38	40.01	52.86	43.05
13	4	124.32	125.56	125.94	300.85
14	4	115.41	115.86	116.74	236.58
9	9	70.67	77.67	76.53	121.10
16	5	161.38	162.25	162.49	286.75
10	10	250.45	256.96	445.93	479.96
Avg. rel. diff. from opt.			2.51%	21.5%	59.16%

5. Calculation scheme

In this section we will consider how to perform the required calculation in the model. As the TS-system will continuously interact with the user, it is important that the system can perform its calculations in 'real time' (that is, the calculations should be performed using an amount of time that seems negligible to the user). The important point to make is that performing calculations is in principle of time complexity exponential in the number of components in the model. It is therefore crucial to identify the 'idle time' of the system (i.e. the time when the user is not interacting with it), and use those points in time to perform the calculations. Idle time is available before the system is put into use, and at times when the user is busy performing an action or trying to find information to answer a question. Before the system is ready to be used it has to go through an initialization phase, which basically amounts to calculating the initial probabilities for each component to be in its faulty state given that the equipment is faulty, the probabilities for the actions to be successful, and the initial beliefs regarding possible answers to the different questions. These calculations can be performed off-line and are thus not subject to speed requirements. In the following we will therefore focus on how to incorporate information from the performed TS-steps into the system, that is, how to update the probability distributions when the compiled evidence $\boldsymbol{\epsilon}$ is extended.

5.1. Action sequences

First we will look at TS-systems that only consists of actions, and describe a method for calculating $P(R(A) =$

ok$|\boldsymbol{\epsilon}$) for an action $A \in \mathscr{A}$ where is some evidence not involving A. Next, we will describe a method to calculate $P(\boldsymbol{\epsilon})$ (required by the ECR-calculations, see Eq. (2)). Note that the evidence $\boldsymbol{\epsilon}$ will contain only a list of *failed* actions, i.e. $\boldsymbol{\epsilon} = \{R(A) = \text{no} : A \in \mathscr{A}'\}$. If an action is successful, the troubleshooting ends, and there is no need to incorporate that evidence into the system.

The key point during the calculations is that of conditional independence. Let nd(V) be the *non-descendants* of V in a directed graph $\mathscr{G}$; $Y \in \text{nd}(X)$ iff there is no directed path $X \to \cdots \to Y$ in $\mathscr{G}$. An important result we shall use frequently is that $V \perp\!\!\!\perp \text{nd}(V)|\text{pa}(V)$ for any variable $V \in \mathscr{V}$.

The backbone of our calculating scheme is the observation that *if we know that $\mathscr{C}_i$ is the actual cutset*, then it is easy to calculate the success probabilities given the evidence $\boldsymbol{\epsilon}$. It turns out that $P(R(A) = \text{ok}|\mathscr{C}_i = \text{faulty}, \boldsymbol{\epsilon}) = P(R(A) = \text{ok}|\mathscr{C}_i = \text{faulty})$, see Lemma 9 below. Since the actual cutset is not known during troubleshooting, we use

$$P(R(A) = \text{ok}|\boldsymbol{\epsilon}) = \sum_{\mathscr{C}_\ell \in \mathscr{C}} P(R(A) = \text{ok}|\mathscr{C}_\ell = \text{faulty}, \boldsymbol{\epsilon}) \qquad P(\mathscr{C}_\ell = \text{faulty}|\boldsymbol{\epsilon}) = \sum_{\mathscr{C}_\ell \in \mathscr{C}} P(R(A) = \text{ok}|\mathscr{C}_\ell = \text{faulty}) \qquad P(\mathscr{C}_\ell = \text{faulty}|\boldsymbol{\epsilon}),$$

to calculate $P(R(A) = \text{ok}|\boldsymbol{\epsilon})$. Next, we formalize the above statement:

Lemma 9. *Let $A \in \mathscr{A}$ be a repair action, and let evidence compiled during troubleshooting be denoted by $\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} = \{R(A_i) = \text{no} : A_i \in \mathscr{A}'\}$ $(A \notin \mathscr{A}')$. Assume that the user's ability to repair one component X does not depend on the state of the other components, $A^{\downarrow X} \perp\!\!\!\perp X'|X$ for all $X' \in \mathscr{X}\backslash\{X\}$, and that information about the user failing to perform one repair action will not influence our beliefs about his ability to perform other actions, $A_i^{\downarrow X_k} \perp\!\!\!\perp A_j^{\downarrow X_\ell}|\{X_k, X_\ell\}$ whenever $i \neq j$. Then $P(R(A)|\mathscr{C}_m = \text{faulty}, \boldsymbol{\epsilon}) = P(R(A)|\mathscr{C}_m = \text{faulty})$. That is, the evidence $\boldsymbol{\epsilon}$ does not influence $R(A)$ when conditioning on the actual MCS.*

Proof. First, notice that if $P(R(A) = \text{ok}|\mathscr{C}_m = \text{faulty}) = 0$, then no evidence $\boldsymbol{\epsilon}$ can change this belief. Hence, $P(R(A)|\mathscr{C}_m = \text{faulty}, \boldsymbol{\epsilon}) = P(R(A)|\mathscr{C}_m = \text{faulty})$ if A cannot repair any component in $\mathscr{C}_m$. Next, assume that the action A can repair components in only one MCS, $\mathscr{C}_\ell$. If $\mathscr{C}_\ell = \text{faulty}$, then all components $X_j \in \mathscr{C}_\ell$ are in their faulty state. Hence, we have evidence on the set pa(A), and since $\boldsymbol{\epsilon}$ only contains non-descendant of A by construction of the domain model, $A \perp\!\!\!\perp \boldsymbol{\epsilon}|\{\mathscr{C}_\ell = \text{faulty}\}$. It follows that $R(A) \perp\!\!\!\perp \boldsymbol{\epsilon}|\{\mathscr{C}_\ell = \text{faulty}\}$, and therefore $P(R(A)|\mathscr{C}_\ell = \text{faulty}, \boldsymbol{\epsilon}) = P(R(A)|\mathscr{C}_\ell = \text{faulty})$. (See A_1 in Fig. 5; the probability for A_1 to repair the equipment is determined by the state of $\mathscr{C}_1$ only. If $\mathscr{C}_1$ is the actual MCS then A_1 repairs the equipment with probability $P(A_1^{\downarrow X_1} = \text{yes}|X_1 = \text{faulty})$ no matter what actions have earlier been performed; if $\mathscr{C}_1$ is not faulty, then A_1 can never repair the equipment.)

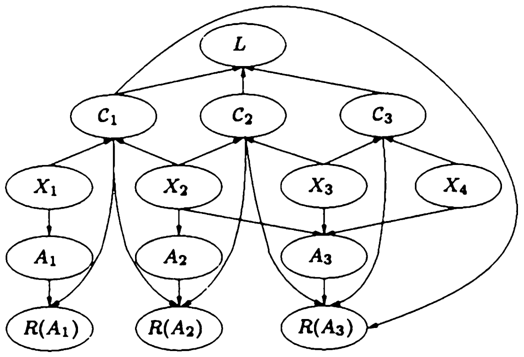

Fig. 5. Example TS-model to exemplify the proof of Lemma 9.

In the general case action A can repair more than one MCS. To see that the Lemma holds also in this case, we introduce the random variable $\zeta(\mathscr{C}_\ell)$, which is defined s.t. $\zeta(\mathscr{C}_\ell) = \text{yes}$ if $\{X_i = \text{faulty} : X_i \in \mathscr{C}_\ell \wedge X_j = \text{ok} : X_j \notin \mathscr{C}_\ell\}$; $\zeta(\mathscr{C}_\ell) = \text{no}$ otherwise. Notice that the effect of conditioning on the event $\zeta(\mathscr{C}_\ell) = \text{yes}$ is that all $X \in \mathscr{X}$ are given evidence, and by construction of the domain model, the set pa$(A) \subseteq \mathscr{X}$ is instantiated. Hence $P(R(A) = \text{ok}|\zeta(\mathscr{C}_\ell) = \text{yes}, \boldsymbol{\epsilon}) = P(R(A) = \text{ok}|\zeta(\mathscr{C}_\ell) = \text{yes})$. Since $A^{\downarrow X} \perp\!\!\!\perp X'X$ for $X' \in \mathscr{X}\backslash\{X\}$, we have $P(R(A)|\mathscr{C}_\ell = \text{faulty}, \boldsymbol{\epsilon}) = P(R(A)|\zeta(\mathscr{C}_\ell) = \text{yes}, \boldsymbol{\epsilon})$. Finally, it follows that $P(R(A)|\mathscr{C}_\ell = \text{faulty}, \boldsymbol{\epsilon}) = P(R(A)|\mathscr{C}_\ell = \text{faulty})$. (Look at action A_3 in Fig. 5, and assume that $\mathscr{C}_3$ is known to be faulty, which means that $X_3 = X_4 = \text{faulty}$. The event $\{R(A_3) = \text{ok}\}$ is in this case equivalent to $\{A_3^{\downarrow X_3} = \text{yes} \vee A_3^{\downarrow X_4} = \text{yes}\}$. So far we only have observations on X_3 and X_4; X_1 and X_2 are not instantiated. Hence, information may flow from $R(A_1)$ to $R(A_3)$, and thereby break the required independence (which is problematic if $\{R(A_1) = \text{no}\}$ has been observed). The assumption $A_i^{\downarrow X_j} \perp\!\!\!\perp X_k|X_j$ does however justify that we may set $X_1 = X_2 = \text{ok}$ without changing the required probability $P(R(A_3)|\mathscr{C}_3 = \text{faulty}, \boldsymbol{\epsilon})$. All flow of information from any compiled evidence $\boldsymbol{\epsilon}$ to $R(A_3)$ is blocked when these stochastic variables are instantiated, and the desired conditional independence follows.) $\square$

We utilize Lemma 9 to calculate the probability that an action $A \in \mathscr{A}$ repairs the equipment:

$$P(R(A)|\boldsymbol{\epsilon}) = \sum_{\mathscr{C}_\ell \in \mathscr{C}} P(R(A)|\mathscr{C}_\ell = \text{faulty}) P(\mathscr{C}_\ell = \text{faulty}|\boldsymbol{\epsilon}). \quad (6)$$

That is, calculating $P(R(A)|\boldsymbol{\epsilon})$ amounts to finding $P(R(A)|\mathscr{C}_\ell = \text{faulty})$ and $P(\mathscr{C}_\ell = \text{faulty}|\boldsymbol{\epsilon})$ for all $\mathscr{C}_\ell \in \mathscr{C}$. The values of $P(R(A)|\mathscr{C}_\ell = \text{faulty})$ can easily be calculated from the model description before the troubleshooting starts, whereas $P(\mathscr{C}_\ell = \text{faulty}|\boldsymbol{\epsilon})$ must be calculated in each case.

We now show that Lemma 9 can be used also to calculate $P(\mathscr{C}_\ell = \text{faulty}|\mathbf{e}_i)$ rather efficiently; recall that $\mathbf{e}_i$ is used to denote the evidence that the first i actions in the sequence $\mathcal{S} = \langle A_1, \ldots, A_N \rangle$ have all failed to repair the equipment. We first use Bayes' rule to investigate how to update this probability when new evidence $\{R(A_i) = \text{no}\}$ is received and appended to the compiled knowledge $\mathbf{e}_{i-1}$:

$$\begin{aligned} P(\mathcal{C}_\ell = \text{faulty}|\mathbf{e}_i) &= P(\mathcal{C}_\ell = \text{faulty}|\mathbf{e}_{i-1}, R(A_i) = \text{no}) \\ &= \frac{P(R(A_i) = \text{no}|\mathcal{C}_\ell = \text{faulty}, \mathbf{e}_{i-1})P(\mathcal{C}_\ell = \text{faulty}|\mathbf{e}_{i-1})}{P(R(A_i) = \text{no}|\mathbf{e}_{i-1})} \\ &= \frac{P(R(A_i) = \text{no}|\mathcal{C}_\ell = \text{faulty})P(\mathcal{C}_\ell = \text{faulty}|\mathbf{e}_{i-1})}{P(R(A_i) = \text{no}|\mathbf{e}_{i-1})}, \end{aligned} \tag{7}$$

$P(R(A_i) = \text{no}|\mathbf{e}_{i-1})$ is just a normalization constant in this calculation, which can be found by

$$\begin{aligned} &P(R(A_i) = \text{no}|\mathbf{e}_{i-1}) \\ &= \sum_{\mathcal{C}_k \in \mathcal{C}} P(R(A_i) = \text{no}|\mathcal{C}_k = \text{faulty})P(\mathcal{C}_k = \text{faulty}|\mathbf{e}_{-1}). \end{aligned}$$

Hence $P(\mathcal{C}_\ell = \text{faulty}|\mathbf{e}_i)$ can be calculated by expanding the evidence iteratively. The first step of this procedure requires the a priori distribution over the MCSs, $P(\mathcal{C}_\ell = \text{faulty}|\mathbf{e}_0)$. This distribution should be calculated by a full propagation in the BN, see Ref. [22]; remember that this propagation can be performed off-line (i.e. before troubleshooting starts). The evidence $\mathbf{e}_i$ is then incorporated by using Eq. (7) until we obtain $P(\mathcal{C}_\ell = \text{faulty}|\mathbf{e}_i)$. This means that calculating $P(R(A)|\mathbf{e}_i)$ is of complexity $\mathcal{O}(R)$, where R is the number of MCSs in the domain if we have stored $P(\mathcal{C}_\ell = \text{faulty}|\mathbf{e}_{i-1})$. As a consequence, the complexity of Algorithm 1 is $\mathcal{O}(NR + N\log(N))$ and the complexity of Algorithm 2 is $\mathcal{O}(N(NR + N) = \mathcal{O}(N^2R)$.

Next, we look at how to calculate $P(\mathbf{e}_i)$; a number required by the ECR calculations, see Eq. (2). This can be done by using the identity $P(\mathbf{e}_i) = P(R(A_i) = \text{no}|\mathbf{e}_{i-1})P(\mathbf{e}_{i-1})$ and make the calculations iteratively; $P(R(A_i) = \text{no}|\mathbf{e}_{i-1})$ is given by Eq. (6); $P(\mathbf{e}_0) = 1$ by convention. Calculating $P(\mathbf{e}_i)$ is therefore of complexity $\mathcal{O}(R)$ if we store the values $P(\mathbf{e}_{i-1})$. In total, the calculation of ECR is thus of time complexity $\mathcal{O}(NR)$.

The time complexity of generating a full action sequence based on the observation-based efficiency (Definition 7) is dominated by the expensive calculations required to find VOI$(\cdot|\boldsymbol{\epsilon})$. If this value is approximated by calculating the ECR of the sequence generated by Algorithm 2, then the time complexity of generating a complete action sequence by the observation-based efficiency is $\mathcal{O}(N^3R)$. If one settles for the cruder approximation offered by Algorithm 1 the time complexity of generating the sequence is reduced to $\mathcal{O}(N^2(\log(N) + R))$.

The time complexity of Algorithm 3 is given by the complexity of the initialization and the cost of $\mathcal{O}(N^2)$ calculations of ECR. This means that the total complexity of Algorithm 3 when initialized according to the obef-sequence is $\mathcal{O}(N^3R)$. This should be compared to the corresponding calculations performed in a fault tree, which Norstrøm et al. [7] report to be $\mathcal{O}(N^2 3^N)$.

5.2. Questions

In this section we consider the cost of belief updating when the TS-model is extended to incorporate questions.

5.2.1. Symptom questions

We start the treatment of questions by considering symptom questions. Recall that symptom questions are used to examine possible failure symptoms; they are connected to the system layer at the MCS level, with edges directed from problem causes to the questions, see Q_S in Fig. 2. By construction, the parent set of a symptom question Q_S in our BN representation is therefore restricted to the MCS nodes, $\text{pa}(Q_S) \subseteq \mathcal{C}$. Furthermore, symptom questions do not have descendants in the graph. It follows that $Q_S \perp\!\!\!\perp \mathcal{V}\setminus\{\mathcal{C}, Q_S\}|\mathcal{C}$. Therefore, to calculate the effect of a symptom question on the remaining strategy, it is only required to calculate the effect on the distribution over the MCSs, $P(\mathcal{C}_\ell = \text{faulty}|Q_S = q, \boldsymbol{\epsilon})$. This can be done by using Bayes' rule

$$\begin{aligned} &P(\mathcal{C}_\ell = \text{faulty}|Q_S = q, \boldsymbol{\epsilon}) \\ &\quad = \frac{P(Q_S = q|\mathcal{C}_\ell = \text{faulty}, \boldsymbol{\epsilon})P(\mathcal{C}_\ell = \text{faulty}|\boldsymbol{\epsilon})}{P(Q_S = q|\boldsymbol{\epsilon})} \\ &\quad = \frac{P(Q_S = q|\mathcal{C}_\ell = \text{faulty})P(\mathcal{C}_\ell = \text{faulty}|\boldsymbol{\epsilon})}{P(Q_S = q|\boldsymbol{\epsilon})}, \end{aligned} \tag{8}$$

where $P(Q_S = q|\boldsymbol{\epsilon}) = \sum_{\mathcal{C}_k \in \mathcal{C}} P(Q_S = q|\mathcal{C}_k = \text{faulty})\ P(\mathcal{C}_k = \text{faulty}|\boldsymbol{\epsilon})$. Hence, the complexity of calculating $P(\mathcal{C}_\ell = \text{faulty}|Q_S = q, \boldsymbol{\epsilon})$ from $P(\mathcal{C}_\ell = \text{faulty}|\boldsymbol{\epsilon})$ is $\mathcal{O}(R)$. If we assume that the ordering of actions needed to calculate the ECR values in the decision rule of Eq. (5) is based on Algorithm 2, then a question can be evaluated in time complexity $\mathcal{O}(N^2R)$. Note that the calculations will require the computation of ECR for several action sequences (described in Section 5.1); one for each possible answer to the question.

Note that $Q_S \perp\!\!\!\perp \mathcal{V}\setminus\{\mathcal{C}, Q_S\}|\mathcal{C}$ implies that symptom questions will not corrupt the calculations of $R(A|\boldsymbol{\epsilon})$ in Eq. (6); we can use that calculation scheme to calculate $R(A|\boldsymbol{\epsilon})$ even when the evidence $\boldsymbol{\epsilon}$ contains answers to symptom questions.

5.2.2. Configuration questions

Configuration questions are designed to highlight the likelihood of component failures by uncovering the environment in which the failed equipment is embedded. Configuration nodes are connected to the system layer via the component layer, with edges directed from question to components, see K in Fig. 2. The answer to the question is modelled as a random variable dependent on the configuration, see Q_K in Fig. 2.

As for symptom questions, we are interested in evaluating Q_K according to Eq. (5). First, however, we note that $R(A) \perp\!\!\!\perp \boldsymbol{\epsilon} | \{\mathscr{C}_\ell = \text{faulty}\}$ also when configuration questions have been posed, $\{Q_K = q\} \subseteq \boldsymbol{\epsilon}$. Recall that $P(R(A)|\boldsymbol{\epsilon}', \mathscr{C}_\ell = \text{faulty}) = P(R(A)|\mathscr{C}_\ell = \text{faulty})$ when $\boldsymbol{\epsilon}'$ is a list of actions (not containing A) that have failed. This result trivially extends to the case where $\boldsymbol{\epsilon}$ contain answers to questions because configuration questions are non-descendants of the actions' result nodes. We can therefore calculate the efficiency of an action using Eq. (6) also in the case when configuration questions have been answered. Similarly, we can calculate the ECR-values required to evaluate a configuration question Q_K (according to Eq. (5) efficiently by incorporating the effect of a question Q_K at the cutset nodes by using Eq. (8).

Special attention is however required for the case when one configuration question Q_{K_1} is evaluated, and the evidence $\boldsymbol{\epsilon}$ already contains the answer to another configuration question Q_{K_2} together with a list of failed actions $\boldsymbol{\epsilon}', \boldsymbol{\epsilon} = \{Q_{K_2} = q, \boldsymbol{\epsilon}'\}$. The answer to the two configuration questions Q_{K_1} and Q_{K_2} are not independent given the actual cutset; we have $P(Q_{K_1} = q, |\boldsymbol{\epsilon}, \mathscr{C}_\ell = \text{faulty}) = P(Q_{K_1} = q | Q_{K_2} = q, \mathscr{C}_\ell = \text{faulty})$. Hence, we must take the answers to all earlier configuration questions into account when we want to calculate $P(Q_{K_1} = q|\boldsymbol{\epsilon})$. A consequence of this conditional dependence is that the fast rules to incorporate new evidence into the system, see Eqs. (7) and (8), cannot be generalized to evidence containing configuration questions if the distribution of other configuration questions should be updated correctly. We therefore have to perform a propagation in the model as soon as a configuration question is *answered*; note that it is not required to perform any propagations as long as the TS-system just considers to *pose* the question. The complexity of *evaluating* a configuration question is therefore $\mathscr{O}(N^2R)$; the time complexity of incorporating the answer into the system is exponential in the number of components.

6. Concluding remarks

We have described a decision-theoretic troubleshooting system, which builds on a BN describing the faulty equipment and its surroundings. The expressive power of the BN framework outperforms that of more commonly used model description paradigms as, e.g. fault trees, see Ref. [21]. We utilized this to make a rich description of the troubleshooting domain, which may include, e.g. non-perfect actions and information-gathering troubleshooting steps. Finally, we showed how our BN models allow fast calculation of the probabilities required to generate a reasonable troubleshooting strategy.

Acknowledgements

We would like to thank our project coworkers, in particular Claus Skaanning, Jiří Vomlel, and Olav Bangsø, for interesting discussions. Jiří Vomlel also supplied the software used to generate the results in Table 2. An anonymous referee gave comments that helped improving the paper.

References

[1] Vesely WE. Fault tree handbook. Technical report. NUREG-0492. US Nuclear Regulatory Committee, Washington, DC; 1981.
[2] Zhang Q, Mei Q. A sequence of diagnosis and repair for a 2-state repairable system. IEEE Trans Reliab 1987;R-36(1):32–3.
[3] Kalagnanam J, Henrion M. A comparison of decision analysis and expert rules for sequential analysis. Uncertainty in artificial intelligence 4, New York: North-Holland; 1990. p. 271–81.
[4] Xiaozhong W. Fault tree diagnosis based on Shannon entropy. Reliab Engng Syst Safety 1991;34:143–67.
[5] Xiaozhong W, Cooke RM. Optimal inspection sequence in fault diagnosis. Reliab Engng Syst Safety 1992;37:207–10.
[6] Reinertsen R, Xiaozhong W. General inspection strategy for fault diagnosis-minimizing the inspection costs. Reliab Engng Syst Safety 1995;48(3):191–7.
[7] Norstrøm J, Cooke RM, Bedford TJ. Value of information based inspection-strategy of a fault-tree. Proceedings of the Tenth European Conference on Safety and Reliability; 1999. p. 621–6.
[8] Srinivas S. A polynomial algorithm for computing the optimal repair strategy in a system with independent component failures. Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, San Fransisco, CA; 1995. p. 515–22.
[9] Breese JS, Heckerman D. Decision-theoretic troubleshooting: a framework for repair and experiment. Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA: Morgan Kaufmann Publishers; 1996. p. 124–32.
[10] Jensen FV, Kjærulff U, Kristiansen B, Langseth H, Skaanning C, Vomlel J, Vomlelová M. The SACSO methodology for troubleshooting complex systems. Artif Intell Engng, Des, Anal Manufact 2001; 15(5):321–33.
[11] Skaanning C, Jensen FV, Kjærulff U, Pelletier P, Rostrup-Jensen L. Printing system diagnosis: a Bayesian network application. Workshop on Principles of Diagnosis, Cape God, MA; 2000.
[12] Pearl J. Probabilistic reasoning in intelligent systems: networks of plausible inference. San Mateo, CA: Morgan Kaufmann; 1988.
[13] Jensen FV. Bayesian networks and decision graphs. New York: Springer; 2001.
[14] Barlow RE. Using influence diagrams. In: Clarotti CA, Lindley DV, editors. Accelerated life testing and experts' opinions in reliability. 1988. p. 145–57.
[15] Call HJ, Miller WA. A comparison of approaches and implementations for automating decision analysis. Reliab Engng Syst Safety 1990;30:115–62.
[16] Torres-Toledano JG, Sucar LE. Bayesian networks for reliability analysis of complex systems. Lect Notes Artif Intell 1998;1484:195–206.
[17] Fenton NE, Littlewood B, Neil M, Strigini L, Sutcliffe A, Wright D. Assessing dependability of safety critical systems using diverse evidence. IEE Proc Software Engng 1998;145(1):35–9.
[18] Fenton NE, Neal M. Bayesian belief nets: a causal model for predicting defect rates and resource requirements. Software Test Qual Engng 2000;2(1):48–53.
[19] Dahll G, Gran BA. The use of Bayesian belief nets in safety assessment of software based systems. Int J Gen Syst 2000;29(2): 205–29.

[20] Ibargüengoytia PH, Sucar LE, Morales E, A probabilistic model approach for fault diagnosis. Eleventh International Workshop on Principles of Diagnosis; 2000. p. 79–86.

[21] Bobbio A, Portinale L, Minichino M, Ciancamerla E. Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. Reliab Engng Syst Safety 2001;71(3):249–60.

[22] Jensen FV, Lauritzen SL, Olesen KG. Bayesian updating in causal probabilistic networks by local computations. Comput Stat Q 1990;4: 269–82.

[23] Heckerman D, Breese JS. A new look at causal independence. Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence, San Francisco, CA: Morgan Kaufmann; 1994. p. 286–92.

[24] Vesely WE. A time-dependent methodology for fault tree evaluation. Nucl Engng Des 1970;13:339–60.

[25] Skaanning C. A knowledge acquisition tool for Bayesian-network troubleshooters. Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference, San Francisco, CA: Morgan Kaufmann; 2000. p. 549–57.

[26] Langseth H, Jensen FV. Heuristics for two extensions of basic troubleshooting. Seventh Scandinavian Conference on Artificial Intelligence, SCAI'01, Frontiers in Artificial Intelligence and Applications, Odense, Denmark: IOS Press; 2001. p. 80–9.

[27] Sochorová M, Vomlel J. Troubleshooting: NP-hardness and solution methods. The Proceedings of the Fifth Workshop on Uncertainty Processing, WUPES'2000, Jindřichův Hradec, Czech Republic; 2000. p. 198–212.

[28] Center for Chemical Process Safety, Guidelines for chemical process quantitative risk analysis. New York: American Institute of Chemical Engineers; 1989.

[29] Vomlel J. On quality of BATS troubleshooter and other approximative methods. Technical report. Department of Computer Science, Aalborg University, Denmark; 2000.