Troubleshooting with Simultaneous Models

Jiří Vomlel[1] and Claus Skaanning[2]

[1] Department of Computer Science, Aalborg University
Fredrik Bajers Vej 7E, DK-9220 Aalborg, Denmark
jirka@cs.auc.dk

[2] Hewlett-Packard, Customer Support R&D
Fredrik Bajers Vej 7E, DK-9220 Aalborg, Denmark
claus_skaanning@hp.com

Abstract. The goal of decision-theoretic troubleshooting is to find a sequence of actions that minimizes the expected cost of repair of a device. If the device is complex then it is convenient to create several Bayesian Networks, each designed to solve a particular problem. At the beginning of a troubleshooting process, it is often necessary to help the user to select the proper model. Complications arise if the user is able to give only a vague description of the problem. In such a case we need to work simultaneously with many troubleshooting models. In this paper we show how models that were originally designed as independent models can be used together while memory space and computational time are kept low. We allow models to be overlapping, i.e., two or more models may contain equivalent troubleshooting steps and/or equivalent problem causes (device faults). We propose a troubleshooting procedure that can be used with many simultaneous models at once. The key that enables us to join the models together is the single fault assumption, which means that there is only one fault causing a device malfunction at a time.

1 SACSO Troubleshooting Approach

We start with a review of the SACSO troubleshooting approach proposed for troubleshooting with a single model. The approach was implemented in the HP BATS troubleshooter [2]. The goal of a troubleshooting task is to find and remove the cause of a device malfunction. In case of a complex device, such as for example a laser printer, it is convenient to create several models each designed to solve a particular problem. All original troubleshooting models $M_i, i = 1, 2, \ldots, N$ have similar structure. Each model M_i describes relations between a set of repair actions $\mathcal{A}_i$, a set of observations $\mathcal{O}_i$, and a set of causes $\mathcal{C}_i$ that can be solved within model M_i. Repair actions are actions that can directly solve the problem, while observations can not solve the problem directly, but may help identify the problem cause.

It is assumed that only one cause from $\mathcal{C}_i$ can be the cause of a device malfunction at a time. It is often referred to as the single fault assumption. This assumption is reasonable when troubleshooting printing systems and similar

man-made devices. Therefore, each cause can be represented as a state $c_i \in \mathcal{C}_i$ of a single cause variable CM_i. The state space of each variable CM_i is extended by an additional state *n.a.* This state corresponds to the case when the true cause of the problem is not addressed in model M_i. In other words, $CM_i = n.a.$ corresponds to the situation when model M_i does not solve the problem.

It is also assumed that actions and observations are independent given the cause. This assumption implies that if the cause of the problem is known then neither the fact that an action failed to solve the problem nor an outcome of a made observation affect the probability of any other action solving the problem. In Fig. 1 an example of two SACSO troubleshooting models is shown.

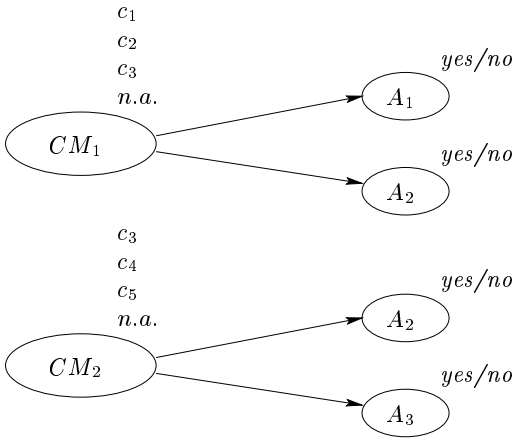

Fig. 1. Two SACSO troubleshooting models.

Remark 1. In the HP BATS troubleshooter both repair actions and observations are included. To keep the exposition simple we stick to actions only. However, the approach of this paper is suitable also for models containing both repair actions and observations.

For every action A and for each cause $c \in \mathcal{C}_i$ included in the model M_i the conditional probabilities $P_i(A = yes \mid CM_i = c)$ are provided by a domain expert and it is assumed that $P_i(A = yes \mid CM_i = n.a.) = 0$. We say that an action A can solve a cause c in a model M_i if $P_i(A = yes \mid CM_i = c) > 0$. The set of causes that can be solved by an action A in a model M_i is denoted by $pa_i(A)$, $pa_i(A) \subseteq \mathcal{C}_i$. The set of actions that can solve a cause c in a model M_i is denoted by $ch_i(c)$, $ch_i(c) \subseteq \mathcal{A}_i$.

Each action A has associated a cost $cost(A)$. It may correspond to the time needed to perform action A, money spent when performing action A, a combination of time and money, or another criteria. The troubleshooting task is to

find a sequence of actions that minimizes the expected cost of repair, i.e., the expected total cost of all actions performed until the problem is solved.

It has been shown that in the case of actions $A \in \mathcal{A}$ with disjoint $pa_i(A)$ it suffices to order them decreasingly according to the ratio $P_i(A = yes)/cost(A)$ (see [4]). However, in [7] it was shown that in case of overlapping $pa_i(A)$ the troubleshooting task is NP-hard.[1] The heuristic algorithm that is the essence of the SACSO troubleshooting procedure [2, 5] consists of three basic steps that are repeated until the problem is solved:

1. Select a repair action of the highest efficiency

$$eff(A) = \frac{P_i(A = yes \mid h)}{cost(A)} \ ,$$

   where h denotes evidence introduced by the troubleshooting history. This corresponds to the evidence that all performed actions failed to solve the problem.
2. Perform the chosen action and observe the result.
3. If the performed action did not solve the problem then enter the outcome of the troubleshooting step into the model and update the model.

The reader interested in details or in other approximate methods used to find a troubleshooting strategy is referred to [2, 5, 7].

In the rest of this paper we discuss how troubleshooting with simultaneous models can be performed. In Sec. 2 we describe how single troubleshooting models can be joined together. In Sec. 3 we apply the SACSO troubleshooting procedure to the troubleshooting with many simultaneous models. In Sec. 4 we propose how probabilities of causes can be initiated when only a vague description of a problem is provided.

2 Simultaneous Models

The current approach to multiple models, implemented in the HP BATS troubleshooter [2, 5], is, first, to use the authoring tool [6]. This creates dozens of models, with each model related to a particular problem. When troubleshooting with the HP BATS troubleshooter, the user selects one model with the help of a selection tree. Then she performs troubleshooting with the chosen model as described above. The problem of this approach is that as the user may not know what the problem exactly is, it may not be clear which model to select. Therefore we need a way to work with several models simultaneously.

In Fig. 2 a scheme for troubleshooting with simultaneous models is displayed. The troubleshooter consists of troubleshooting models $M_1, M_2, \ldots, M_N$ and a *supermodel*. The supermodel reflects dependencies between causes and problems. It uses the user's problem description and answers to certain questions that may help identify the problem, it communicates with the troubleshooting models,

[1] If $\forall A \in \mathcal{A} : |pa_i(A)| \leq 2$ then the complexity is still undecided.

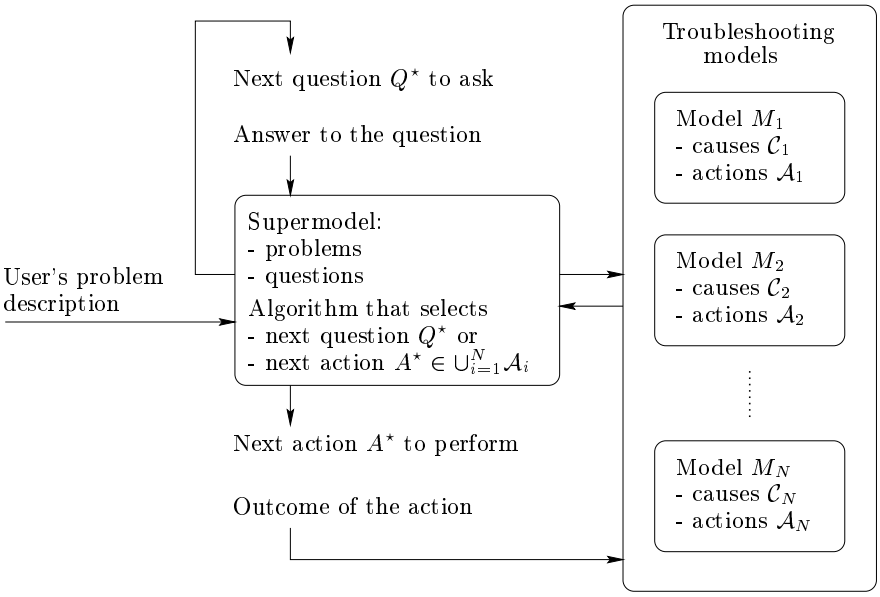

Fig. 2. A basic scheme for troubleshooting with simultaneous models

and it realizes the troubleshooting algorithm, i.e., it selects a best next action to perform and updates the troubleshooting models by the observed outcomes.

It will turn out that under assumptions discussed in this section we can join together all SACSO models and create a single Bayesian network which we will call the *joint model*. Then we can simply apply the algorithm used for troubleshooting with a single SACSO model to troubleshooting with simultaneous models. The supermodel is discussed in detail in Sec. 4.

Generally, there can be identical problem causes and actions that appear in more than one SACSO model. For example, "`Media out of specification`" can be a cause of "`Paper Jam`", "`Spots`", or "`Temporary problem solvable by cycling power`". For each of these three problems a single SACSO model is designed. If it is possible, we identify equivalent causes across all models automatically. Otherwise we need to consult a domain expert. We assign the same index to equivalent causes. Similarly, an automatic program or a domain expert should identify equivalent troubleshooting actions across all models. Again, we will assign the same index to equivalent actions. The joint model contains all problem causes and all troubleshooting actions from the individual SACSO models, i.e.,

$$\mathcal{C} = \bigcup_{i=1}^{N} \mathcal{C}_i \setminus n.a. \quad \text{and} \quad \mathcal{A} = \bigcup_{i=1}^{N} \mathcal{A}_i \ .$$

A question arises whether we can declare two causes $c \in \mathcal{C}_i, c' \in \mathcal{C}_j$ identical if $ch_i(c) \neq ch_j(c')$. For example in Fig. 1, cause c_3 of model M_1 is solved by

action A_1 but action A_1 is not present in model M_2 where cause c_3 is present. A reason for this situation can be that an expert did not want to include the action that is part of the first model in the context of the second model, e.g., because the model would be too complex for the audience. We allow such situations and treat two identical causes solved by different actions as one cause solved by all actions that can solve that cause in any SACSO model. It means that for each cause $c \in \mathcal{C}$ the set of its children in the joint model

$$ch(c) = \bigcup_{i=1}^{N} ch_i(c) \ .$$

A similar question is whether two actions $A \in M_i, A' \in M_j$ can be declared identical if $pa_i(A) \neq pa_j(A')$. For example, in Fig. 1, action A_2 can solve cause c_4 in model M_2 but not in model M_1 since this model does not contain this cause. This situation will appear quite naturally, since no expert would like to list all possible causes that can be solved by an action no matter what the problem is. We treat two identical actions solving different causes as one action solving all causes solved in any SACSO model. It means that for each action $A \in \mathcal{A}$ the set of its parents in the joint model

$$pa(A) = \bigcup_{i=1}^{N} pa_i(A) \ .$$

Let us use the example of the two models from Fig. 1 to explain how to define the conditional distributions attached to actions present in more than one model. Action A_2 solves cause c_2 with probability $P_1(A_2 = yes \mid CM_1 = c_2)$ and c_3 with probability $P_1(A_2 = yes \mid CM_1 = c_3)$ in model M_1. In model M_2 it solves cause c_3 with probability $P_2(A_2 = yes \mid CM_2 = c_3)$ and cause c_4 with probability $P_2(A_2 = yes \mid CM_2 = c_4)$. It is natural to have the same conditional probabilities in the joint model. A question is what to do if for example $P_1(A_2 = yes \mid CM_1 = c_3) \neq P_2(A_2 = yes \mid CM_2 = c_3)$. We believe that the only reason for the difference can be that it is difficult for an expert to be 100% consistent. Therefore we either ask an expert to resolve this inconsistency or we simply take the average of the two numbers. In the rest of this paper we assume that for any two models $M_i \neq M_j$ containing an action A and a cause $c \in pa_i(A)$ and $c \in pa_j(A)$ it holds that

$$P_i(A \mid CM_i = c) = P_j(A \mid CM_j = c) \ .$$

In the individual SACSO models the basic assumption used is the single fault assumption. It is reasonable to keep this assumption in the joint model as well since it is a characteristics of the device and it can not be influenced by the fact that a user does not know what the problem is. The single fault assumption is encoded in the joint model using the node CA that has all possible causes as its states. We also keep the conditional independence of actions given the cause in the joint model.

The arguments provided above leads to a unique way of combining the SACSO models to a *joint model*. The joint model has a Naïve Bayes structure, whose conditional probability distributions are given by the following definition.

Definition 1. *Let $m(c \to A)$ define a function such that for any cause $c \in \mathcal{C}$ and any action $A \in \mathcal{A}$*

$$m(c \to A) = \begin{cases} i \text{ if } \exists M_i,\ A \in M_i \ \wedge\ c \in pa_i(A), \\ 0 \textit{ otherwise.} \end{cases}$$

In the joint model the conditional probability of an action $A \in \mathcal{A}$ given a cause $c \in \mathcal{C}$ is defined as

$$P(A = yes \mid CA = c) = \begin{cases} 0 & \text{if } \ m(c \to A) = 0\ , \\ P_m(A = yes \mid CM_m = c) & \text{if } \ m = m(c \to A) \neq 0\ . \end{cases}$$

In Fig. 3 we give an example of a joint model, which is composed from two SACSO models of Fig. 1.

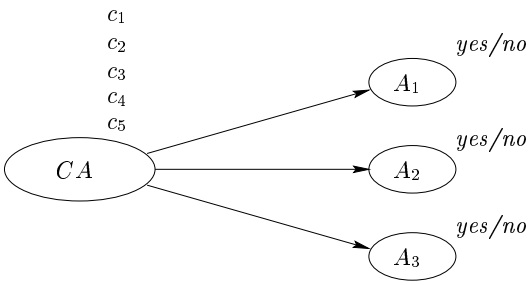

Fig. 3. The two SACSO troubleshooting models from Fig. 1 joined together.

3 A Troubleshooting Procedure

In Sec. 1 we sketched the SACSO troubleshooting procedure for an individual SACSO model. In Sec. 2 we described how the joint model is created from the SACSO models. In this section we propose an algorithm for troubleshooting with simultaneous models that is an application of the SACSO troubleshooting procedure to the joint model. We show that we need not even create the joint model since all information can be stored in and read from the SACSO models. We also demonstrate that a simple naïve approach to troubleshooting with simultaneous models ignoring the fact that different models may contain identical causes and actions may provide erroneous results.

We require $P_i(A = yes \mid CM_i = c)$ being equivalent for all models including both A and c and with $c \in pa_i(A)$. Therefore it does not matter which of

these models is chosen by the function $m(c \to A)$. In fact, also after inserting a troubleshooting history h as evidence the function $m(c \to A)$ can be used to select a model for reading $P(A \mid CA = c, h)$.

Lemma 1. *For the conditional probability $P(A \mid CA = c, h)$ of A given a cause $c \in pa(A)$ and a troubleshooting history h in the joint model it holds that*

$$P(A \mid CA = c, h) = P_{m(c \to A)}(A \mid CM_{m(c \to A)} = c) \ .$$

Proof. By the conditional independence of actions given the cause in the joint model we get

$$P(A \mid CA = c, h) = P(A \mid CA = c) \ .$$

From Definition 1 we read that

$$P(A \mid CA = c) = P_{m(c \to A)}(A \mid CM_{m(c \to A)} = c) \ ,$$

which proves the assertion of the Lemma. □

The following lemma underlies the simple propagation scheme for Naïve Bayes models. It will be used in the troubleshooting procedure.

Lemma 2. *Consider the joint model with a troubleshooting history h. Let $L = |\mathcal{C}|$. Then the probabilities of causes $c \in \mathcal{C}$ of the joint model can be updated in the light of a new evidence $e = (A = no)$ by the following formula*

$$P(CA = c \mid e, h) := \frac{P(e \mid CA = c) \cdot P(CA = c \mid h)}{P(e \mid h)} \ ,$$

where $P(e \mid h) = \sum_{\ell=1}^{L} P(e \mid CA = c_\ell) \cdot P(CA = c_\ell \mid h)$.

Proof. Using Bayes' rule we can write

$$P(CA = c \mid e, h) = \frac{P(e, CA = c \mid h)}{P(e \mid h)} \ . \tag{1}$$

Since actions are independent given a cause in the joint model we get

$$P(e, CA = c \mid h) = P(e \mid CA = c) \cdot P(CA = c \mid h) \ . \tag{2}$$

The probability $P(e \mid h)$ can be computed as

$$P(e \mid h) = \sum_{\ell=1}^{L} P(e \mid CA = c_\ell) \cdot P(CA = c_\ell \mid h) \ . \tag{3}$$

Substituting $P(e \mid h)$ from (3) and $P(e, CA = c \mid h)$ from (2) into (1) we get the assertion of the lemma. □

Table 1. A troubleshooting procedure

1. Initiate h, $\mathcal{A}(h) := \mathcal{A}$, and $P(CA = c), c \in \mathcal{C}$.
2. For each action $A \in \mathcal{A}(h)$ compute

$$P(A = yes \mid h) := \sum_{c \in pa(A)} P(A = yes \mid CA = c) \cdot P(CA = c \mid h)$$

$$\mathit{eff}(A \mid h) := \frac{P(A = yes \mid h)}{cost(A)} \ .$$

3. Perform action $A^\star := \arg\max_{A \in \mathcal{A}(h)} \mathit{eff}(A \mid h)$.
4. If $A^\star$ solves the problem then quit
   otherwise set $e := \{A^\star = no\}$ and continue with step 5.
5. For each $c \in pa(A^\star)$ compute

$$f(c, e \mid h) := P(e \mid CA = c) \cdot P(CA = c \mid h) \ .$$

6. For each $c \in \mathcal{C} \setminus pa(A^\star)$ set $f(c, e \mid h) := P(CA = c \mid h)$.
7. Compute the normalization constant

$$P(e \mid h) := \sum_{c \in \mathcal{C}} f(c, e \mid h) \ .$$

8. Normalize, i.e., for each $c \in \mathcal{C}$

$$P(CA = c \mid e, h) := \frac{f(c, e \mid h)}{P(e)} \ .$$

   Update $h := h \cup \{A^\star = no\}$ and $\mathcal{A}(h) := \mathcal{A}(h) \setminus A^\star$. If $\mathcal{A}(h) = \emptyset$ then quit. Otherwise, go to step 2.

Using Lemma 2 we establish a fast updating scheme. The full troubleshooting procedure based on this updating scheme is described in Table 1. We note that the procedure corresponds to the SACSO troubleshooting procedure [2, 5] applied to the joint model. This procedure does not provide optimal troubleshooting strategies. However, when troubleshooting printers, it was shown that the strategies provided by the SACSO troubleshooting procedure are very close to optimal strategies [2] .

Proposition 1. *The troubleshooting procedure described in Table 1 can be performed using the original SACSO models only, i.e., we need not create the joint model.*

Proof. Observe that $P(A = yes \mid CA = c)$, needed in step 2, can be read from model $M_i, i = m(c \to A)$ as $P_i(A = yes \mid CM_i = c)$ and $P(e \mid CA = c)$, needed in step 5, can be read from model $M_j, j = m(c \to A^\star)$ as $P_j(A^\star = no \mid CM_j = c)$

(Lemma 1). We only need to store repeatedly updated probabilities of causes $P(CA = c_\ell \mid h)$ somewhere. A convenient location can be the variables CM_i in the models $M_i, i = 1, 2, \ldots, N$, where, having new evidence e, we update the probability distribution $P_i(CM_i = c_\ell \mid h)$. Thus we keep all models updated and we can read the probability $P(CA = c_\ell \mid h)$ from any model containing the cause c_ℓ. □

The complexity of the proposed troubleshooting procedure if used to provide a full troubleshooting sequence is only $\mathcal{O}(|\mathcal{A}|^2 \times |\mathcal{C}|)$, where $|\mathcal{A}|$ is the number of different actions and $|\mathcal{C}|$ is the number of different causes over all models. Furthermore, we may substantially speed up the computations if we check for causes having $P(CA = c \mid h)$ equal or close to zero and disqualify such causes from further computations. Consequently we need not work with a great many models at the same time, since all models M_i having $P_i(CM_i = n.a. \mid h)$ equal or close to one can be disqualified from further computations.

In the following remark we show that if the probability of an action solving the problem was computed as the total sum over all SACSO models we would get erroneous results.

Remark 2. Note that each cause $c \in pa(A)$ is included in the sum of step 2 only once, which is generally different from the sum

$$\sum_{j \in \{1,2,\ldots N\}} P_j(A = yes \mid h) = \sum_{j \in \{1,2,\ldots N\}} \sum_{c \in pa_j(A)} \begin{array}{c} P_j(A = yes \mid CM_j = c) \\ \cdot P_j(CM_j = c \mid h) \end{array} ,$$

where each cause appear as many times as is the number of models it is contained in. Therefore, if this formula was used to estimate $P(A = yes \mid h)$, it would disproportionally favour actions solving causes that are contained in more models.

4 Initial probabilities of causes

The reader has probably realized that we have not discussed how the probabilities of the causes are defined when creating the joint model. Since this task requires a deeper discussion we have left it for an independent section.

In order to be able to reflect dependencies between causes and problems we create a Bayesian network model, which we call the *supermodel*. This model can use user's problem description and answers to certain questions that may help identify the problem. The problem variable PR has all possible problems $pr_1, pr_2, \ldots, pr_K$ as its states. The supermodel is connected to the joint model through variable CA. Expert knowledge of dependence between problems and causes is encoded in the conditional probability distribution

$$P(CA = c_\ell \mid PR = pr_k), \quad k = 1, 2, \ldots, K, \quad \ell = 1, 2, \ldots, L \ .$$

This means that for each problem pr_k the expert distributes the probability mass between the causes.

Remark 3. When building a single SACSO model M_i a domain expert provided the initial probabilities of causes assuming the problem being solved in that model, i.e. she provided the conditional probabilities $P_i(CM_i = c \mid problem_i)$, where $problem_i$ is the problem solved in model M_i (see [6] for details). If there is a one-to-one correspondence between models and problems, i.e., $problem_k \sim pr_k$, for $k = 1, 2, \ldots, K$ and $K = N$, then we can use the initial probabilities of causes from the SACSO models to define the conditional probability distribution $P(CA \mid PR)$, so that for $k = 1, 2, \ldots, K$

$$P(CA = c_\ell \mid PR = pr_k) = P_k(CM_k = c_\ell) \text{ if } c_\ell \in \mathcal{C}_k \ ,$$
$$P(CA = c_\ell \mid PR = pr_k) = 0 \text{ otherwise.}$$

When starting with a particular troubleshooting process the model should reflect all the prior knowledge. Prior knowledge is summarized by means of the prior probability distribution of the variable PR:

$$P(PR = pr_1), \quad P(PR = pr_2), \quad \ldots, \quad P(PR = pr_K).$$

For example, these probabilities can be a result of a text mining task performed on the user's description of problem and observations the user made.

At the beginning of a troubleshooting process or during the process the user can be asked certain questions $Q_1, Q_2, \ldots, Q_J$ which may help identify the problem. For each question Q_j and for each problem pr_k, a conditional probability distribution $P(Q_j \mid PR = pr_k)$ is given. It is assumed that question Q_i is independent of Q_j given PR for any $i \neq j, i, j \in \{1, \ldots, L\}$. An example of a supermodel connected to a joint model is given in Fig. 4.

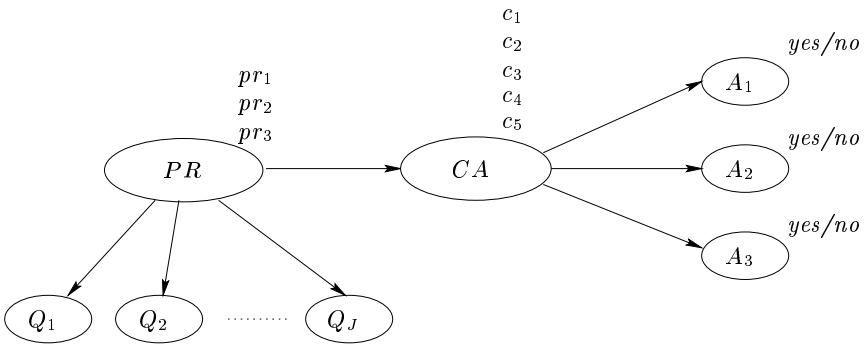

Fig. 4. The supermodel and the joint model joined together

In order to select the most informative question given a current probability distribution over problems $P(PR \mid h)$ we can use standard methods for value of information [1]. An overview of these methods can be found, e.g., in [3]. Methods

used to select questions in the full version of the SACSO algorithm [2, 5], i.e., the probability of a question identifying the problem and the expected cost of observation, are suitable here as well. These methods can be also used to decide whether we will ask the user more questions. In Table 2 the initialization of probabilities of causes is formally described.

Table 2. Initialization of the probabilities of causes.

1. For each problem $pr_k \in \mathcal{PR}$ and for each cause $c_\ell \in \mathcal{C}$, ask a domain expert to provide you $P(CA' = c_\ell \mid PR = pr_k)$.
2. Ask the domain expert to propose set of questions $\mathcal{Q}$. For all answers q_j to all questions $Q_j \in \mathcal{Q}$ and for all possible problems $pr_k \in \mathcal{PR}$ ask the domain expert to provide $P(Q_j = q_j \mid PR = pr_k)$. Use the knowledge acquisition methodology described in [6].

3. Set $h := \emptyset$.
4. Derive $P(PR = pr_k), k \in \{1, 2, \ldots, K\}$ from the user's problem description.
5. Ask the user the most informative question $Q_j \in \mathcal{Q}$ given the history h. Record the answer q_j.
6. For each $pr_k, k \in \{1, 2, \ldots, K\}$ compute

$$P(pr_k \mid h \cup \{Q_j = q_j\}) := \frac{P(Q_j = q_j \mid PR = pr_k) \cdot P(pr_k \mid h)}{\sum_{k=1}^{K} P(Q_j = q_j \mid PR = pr_k) \cdot P(pr_k \mid h)} \text{ and}$$

   set $h := h \cup \{Q_j = q_j\}$.
7. Use a criteria to decide whether you will ask the user more questions. If yes, go to step 5, else initialize $P(CA = c_\ell \mid h)$ for all $c \in \mathcal{C}$ as

$$P(CA = c \mid h) = \sum_{k=1}^{K} P(CA = c \mid PR = pr_k) \cdot P(PR = pr_k \mid h)$$

   and initiate $P_i(CM_i = c \mid h) = P(CA = c \mid h)$ in the individual SACSO models $M_i, i = 1, 2, \ldots, N$.

If we prefer to allow general questions from $\mathcal{Q}$ to be asked during a troubleshooting session then we need to communicate the probability distribution $P(CA \mid h)$ between the SACSO models and the supermodel. When necessary we can return to the supermodel and update it, i.e. in the supermodel we propagate

$$P(CA = c \mid h) = P_m(CM_m = c \mid h), \text{ for all } c \in \mathcal{C} \ ,$$

where m is the index of any model containing cause c. Then we use the algorithm of Table 2 starting with step 5 and using the updated probabilities to decide which questions we ask. Finally, when we decide to return back to the troubleshooting procedure we simply replace the values of $P_i(CM_i = c \mid h)$, $i = 1, 2, \ldots, N$ by the values computed in the supermodel as proposed in step 7.

5 Conclusions

We have presented a fast method that can be used to combine information from thousands of troubleshooting models that may share certain problem causes and solution actions. The single fault assumption, allowed us to derive a simple scheme for updating probabilities of causes in the models. Thus we were able to use the same troubleshooting methods as if we were working with a single joint model.

Acknowledgments

We would like to thank Finn V. Jensen, Helge Langseth, Thomas Nielsen, Kristian G. Olesen, Marta Vomlelová, and Ole-Christoffer Granmo for helpful comments and all other members of the Decision Support Group and Hewlett-Packard Laboratory for Normative Systems at Aalborg University for the friendly working environment they created.

References

1. R. A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, pages 22–26, 1966.
2. Finn V. Jensen, Uffe Kjærulff, Brian Kristiansen, Helge Langseth, Claus Skaanning, Jiří Vomlel, and Marta Vomlelová. The SACSO methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (to appear)*, 2001.
3. F.V. Jensen. *An Introduction to Bayesian Networks*. Springer Verlag, 1996.
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. Claus Skaaning, Finn V. Jensen, and Uffe Kjærulff. Printer troubleshooting using Bayesian Networks. In *the Thirteenth International Conference on Industrial & Engineering Applications of AI & Expert Systems*, 2000.
6. Claus Skaanning. A knowledge acquisition tool for bayesian-network troubleshooters. In C. Boutilier and M. Goldszmidt, editors, *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 549–557, San Francisco, 2000. Morgan Kaufmann Publisher.
7. Marta Sochorová and Jiří Vomlel. Troubleshooting: NP-hardness and solution methods. In *Proceedings of the Fifth Workshop on Uncertainty Processing WUPES'2000, Jindřichův Hradec, Czech Republic*, pages 198–212. University of Economics, Prague, 20-24th June 2000.