Troubleshooting under Uncertainty

David Heckerman **John S. Breese** **Koos Rommelse**

Microsoft Research
One Microsoft Way
Redmond, WA, 98052-6399
<heckerma|breese|koosr@microsoft.com>

Abstract

We develop a series of approximations for decision-theoretic troubleshooting under uncertainty. Our approach generates troubleshooting plans in the face of uncertainty in the relationships among components and device status, observations, as well as the affect of actions on device status. Included in our approach is a Bayesian-network representation of these relationships. We have applied our technique successfully to troubleshooting problems with printing, photocopier feeders, automobiles, and gas turbines. We report empirical findings demonstrating the high quality of plans produced by our approach.

1 Introduction

The purpose of troubleshooting is to generate a low cost plan for the repair of a device, where a plan consists of observations and component-repair actions.[1] In this paper, we develop a series of approximations for decision-theoretic troubleshooting under uncertainty. Our approach generates troubleshooting plans in the face of uncertainty in (1) the relationships among components and device status, (2) observations of device behavior, and (3) the affect of actions on device status. Included in our approach is a Bayesian-network representation of these relationships.

Let $\mathcal{C}$ be the set of variables $\{c_1, c_2, \ldots, c_n\}$ representing the components of the system that we are troubleshooting. We shall assume that each component or fault c_i is in exactly one of a finite set of states.[2] By convention, we use $c_i = \texttt{Normal}$ to denote the event that component c_i is functioning properly. Let $\mathcal{O}$ be the set of variables $\{o_1, o_2, \ldots o_m\}$ representing the set of observations that one can potentially make about the system in question, with each o_i taking on exactly one of r_i possible states $\{o_{i1}, o_{i2}, \ldots, o_{ir_i}\}$. We write $o_i = k$ to indicate that observation o_i takes on state k. In this paper, we assume there is a distinguished observation e that denotes the functional status of the device. We use $e = \texttt{Normal}$ to denote the event that the device is functioning normally.

Both repairs and observations can have nonzero cost. Optimal troubleshooting is the development of a troubleshooting plan with minimum expected cost. In this framework, an observation is valuable only if the expected cost of repair given that observation is less than the expected cost of repair in the absence of that observation. This type of analysis is called *value-of-information analysis* in the field of decision analysis, because we are explicitly calculating the expected value of various observations in terms of their subsequent effect on actions.

A decision tree for optimal troubleshooting of a simple two-component device is shown in Figure 1. The values at the end of the tree are the cumulative costs of observations and costs of repairs along the path from the root. We use C_i^o and C_i^r to denote the cost of observing and repairing component c_i, respectively. The expression $\Pr(e = \texttt{Normal} | \mathit{Repair}(c_i), o_j)$ is the probability that the device will work properly, given that we have repaired component c_i and have observed o_j before repair. We call such probabilities *repair probabilities*. In drawing this model, we have assumed that the costs of repair are independent, that the device will be fixed if one repairs both components, and that only component c_1 can be observed.

The expected cost of a repair sequence defines the (negative) value of that sequence. In Figure 1, the expected cost of repair for the sequence with no obser-

[1] A repair action may be a simple replacement of the component.

[2] Our approach can be generalized to continuous variables, but we do not do so here.

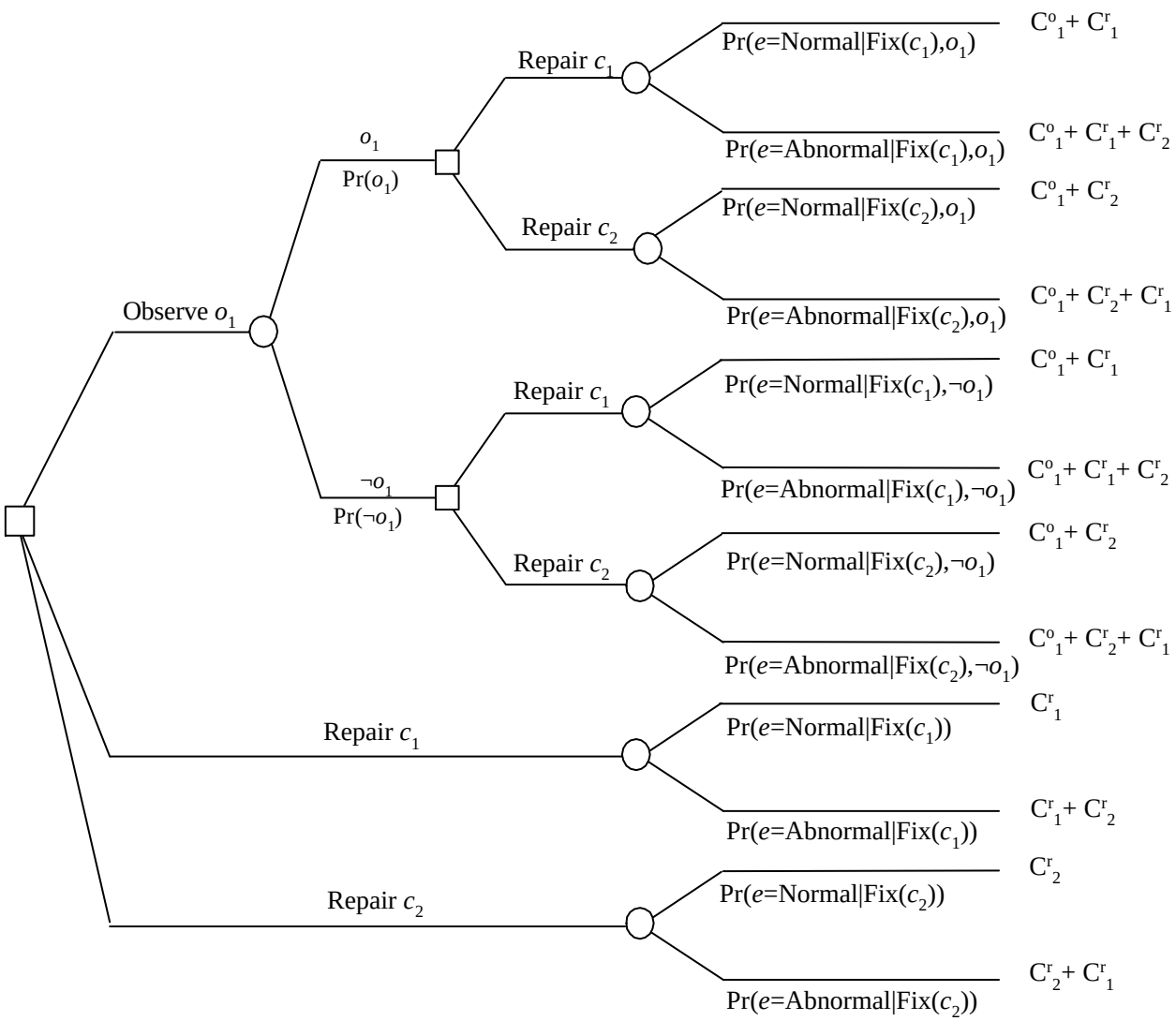

Figure 1: A decision tree for optimal troubleshooting of a two-component device.

vations and repairing c_1 first is

$$\Pr(e = \texttt{Normal}|Repair(c_1))C_1^r + (1 - \Pr(e = \texttt{Normal}|Repair(c_1)))(C_1^r + C_2^r)$$

where the repair cost is C_1^r if the repair of c_1 succeeds in fixing the problem, and $C_1^r + C_2^r$ if we must also repair c_2. Again, this expression assumes that repairing both c_1 and c_2 is guaranteed to fix the problem.

In general, a sequential troubleshooting plan involves the following steps:

1. If the device is working properly, terminate.
2. If not, then either
   (a) Select a component c_i to replace or repair,
   (b) Select an unobserved variable o_j for observation, or
   (c) Call service.
3. Go to 1.

In addition to being able to observe or repair a component at each stage, we have the option of calling service. The idea is that at any level of troubleshooting a device it is possible to "promote" the problem to a higher level of expertise that is guaranteed to be able to repair the device. In the worst case, a service call may correspond to completely replacing a device. For example, if an airplane mechanic is troubleshooting an engine on a jet on the runway, a service call may correspond to replacing the faulty engine and sending it to the maintenance shop for repair.

Development of an optimal solution to the general troubleshooting problem requires an off-line analysis of all possible mixed observation-repair-service sequences using dynamic programming. As we increase the number of reparable components and possible observations, the decision tree grows exponentially. For example, a troubleshooting problem of this form with 5 components and 3 observations would generate a decision tree with nearly 340,000 endpoints. Our strategy in this paper is to generate a series of approximations that more efficiently selects either an observation, a repair action, or the service-call action at each stage of the troubleshooting process.

Our approach incorporates the Bayesian-network representation, an annotated directed graph that encodes probabilistic dependencies among distinctions [Howard and Matheson, 1981, Pearl, 1988]. The representation rigorously describes probabilistic relationships, yet includes a human-oriented qualitative structure that facilitates communication between the user and the probabilistic model. A Bayesian network for a set of variables $\{x_1, \ldots, x_n\}$ represents a joint probability distribution over those variables. The Bayesian network represents the joint distribution by encoding assertions of conditional independence as well as a collection of probability distributions. From the chain rule of probability, we know

$$\Pr(x_1, \ldots, x_n) = \prod_{i=1}^{n} \Pr(x_i|x_1, \ldots, x_{i-1}) \quad (1)$$

For each variable x_i, let $\Pi_i \subseteq \{x_1, \ldots, x_{i-1}\}$ be a set of variables that renders x_i and $\{x_1, \ldots, x_{i-1}\}$ conditionally independent. That is,

$$\Pr(x_i|x_1, \ldots, x_{i-1}) = \Pr(x_i|\Pi_i) \quad (2)$$

A Bayesian network consists of (1) a Bayesian-network structure that encodes the assertions of conditional independence in Equation 2, and (2) a set of probability distributions corresponding to that structure. In particular, the Bayesian-network structure is a directed acyclic graph such that each variable $x_1, \ldots, x_n$ corresponds to a node in that structure, and the parents of the node corresponding to x_i are the nodes corresponding to the variables in Π_i. (In the remainder of this paper, we use x_i to refer to both the variable and its corresponding node in a graph.) Associated with each node x_i are the probability distributions $\Pr(x_i|\Pi_i)$—one probability distribution for each instance of Π_i. Combining Equations 1 and 2, we see that any Bayesian network for $\{x_1, \ldots, x_n\}$ uniquely determines a joint probability distribution for those variables. That is,

$$\Pr(x_1, \ldots, x_n) = \prod_{i=1}^{n} \Pr(x_i|\Pi_i) \quad (3)$$

Algorithms exist for calculating any conditional probability of interest from this implicit joint probability distribution [Pearl, 1988, Jensen et al., 1990]. For most real-world problems, these algorithms are efficient.

2 Determination of the Repair Sequence

Under particular assumptions, it is possible to generate an optimal sequence of repair actions under uncertainty without enumerating a decision tree or equivalent search procedure. Let $p_i = \Pr(e = \texttt{Normal}|Repair(c_i))$ be the repair probability that the device will be functioning properly given c_i is repaired; and let C_i^r be the cost of repairing component c_i. In our simple decision tree in Figure 1, ignoring the observation o_1, we repair c_1 first if and only if

$$p_1 C_1^r + (1 - p_1)(C_1^r + C_2^r) < p_2 C_2^r + (1 - p_2)(C_1^r + C_2^r)$$

or equivalently when

$$\frac{p_1}{C_1^r} > \frac{p_2}{C_2^r}$$

For example, if repairs have the same cost, we choose the one with the higher probability of solving the problem; and if the repairs are equally likely to succeed, then we choose the one with lower cost. Kalagnanam and Henrion (1988) have shown that this result generalizes to generating a sequence of n repairable components under the following assumptions:

A1 There is a single fault—that is, one and only one of the components is responsible for the failure of the device. This assumption implies $\sum_{i=1}^{n} p_i = 1$.

A2 Costs of repair are independent.

A3 There are no observations interleaved between repair actions, except for the observation of whether or not the device is functioning.

Under these conditions, the optimal repair sequence is generated by sorting the components for repair such that

$$\frac{p_1}{C_1^r} \geq \frac{p_2}{C_2^r} \geq \ldots \geq \frac{p_n}{C_n^r} \tag{4}$$

and the expected cost of the repair sequence is given by

$$\begin{aligned} \text{ECR} &= C_1^r + (1-p_1)C_2^r + \ldots + (1-\sum_{j=1}^{n-1} p_j)C_n^r \\ &= \sum_{i=1}^{n}(1-\sum_{j=1}^{i-1} p_j)C_i^r \end{aligned} \tag{5}$$

The preceding equation says the expected cost of repair consists of the cost of repairing the first component, and if that repair is unsuccessful (with probability $1-p_1$), then we incur the cost of the second component, and so on.

In real-world problems, Assumptions A1–A3 are sometimes unreasonable. For example, if the cost of observing a component is significantly less than the cost of repairing the component, then it may be useful to observe that component before repairing it. As another example, there may be multiple faults. To address these possibilities, we use the following approximations of the previous approach for determining repair sequence.

We begin by labeling each component as observable or unobservable. If a component is observable, then we require that it be observed immediately prior to its repair. We call this sequence of an observation followed by a repair an *observation–repair action*. Conversely, if a component is unobservable, then we do not allow its observation prior to repair (or at any time). For convenience, we also call this simple repair action an observation–repair action. Let $\mathtt{E}$ denote the current information state of the troubleshooter. $\mathtt{E}$ may include information about previous observations as well as repairs. The expected cost of the observation–repair action for an observable component c_i given information state $\mathtt{E}$, denoted $C_i^{or}(\mathtt{E})$, is given by

$$C_i^{or}(\mathtt{E}) = C_i^o + \Pr(c_i \neq \mathtt{Normal}|\mathtt{E}) \cdot C_i^r \tag{6}$$

For unobservable components, we have $C_i^{or}(\mathtt{E}) = C_i^r$.

With observation–repair actions and costs so defined, we can still use Equation 4 with C_i^r replaced by $C_i^{or}(\mathtt{E}_i)$, where $\mathtt{E}_i$ denotes the state of information after $i-1$ observation–repair actions have taken place. In addition, when multiple faults are possible, this equation remains valid, provided we replace p_i with $p_i(\mathtt{E}_i) = \Pr(e = \mathtt{Normal}|Repair(c_i), \mathtt{E}_i)$. Because the repair probabilities and costs are no longer constant, the repair sequence specified by Equation 5 is invalid; and the problem of finding a repair sequence that minimizes ECR becomes intractable. Nonetheless, we have found the following approximation to be useful. First, we compute all repair probabilities $p_i(\mathtt{E}_1)$ and costs $C_i^{or}(\mathtt{E}_1)$ under the assumption that no observation–repair actions have taken place. Then, we identify the component with the highest ratio $p_i(\mathtt{E}_1)/C_i^{or}(\mathtt{E}_1)$ as the first component to be observed/repaired. Next, we set this component to $\mathtt{Normal}$, update the information state to $\mathtt{E}_2$, recompute repair probabilities and costs, and identify the second component to be repaired. We iterate this procedure, generating a full sequence.

We can include the special repair action of a service call easily in this procedure. Namely, let C^s denote the cost of a service call. Then, we identify a service call as the next action in the repair sequence when $1/C^s$ is greater than all ratios $p_i(\mathtt{E})/C_i^{or}(\mathtt{E})$.

3 A Myopic Approximation for Observation Planning

In the previous section, we considered two special classes of observations: (1) the observation of the device after a repair is made, and (2) the observation of a component before a repair is made (as part of an observation-repair action). We refer to these observations as *base observations*. In this section, we describe a method for making more general observations. To do so, we use a *myopic* approximation. In this approximation, we pretend that we can make at most one nonbase observation before executing a plan consisting of only observation–repair actions and a service call. Then, we determine which nonbase observation if any should

be made, and make the observation if appropriate. Finally, we iterate this procedure, possibly making additional observations. The procedure is called myopic, because we may make additional nonbase observations in the troubleshooting sequence, but we ignore these possible actions when selecting the next nonbase observation.

Using the procedure described in the previous section, we obtain the expected cost of repair under information state $\mathtt{E}$:

$$\mathrm{ECR}(\mathtt{E}) = \sum_{i=1}^{n_f}(1-\sum_{j=1}^{i-1}p_j(\mathtt{E}_j))C_i^{or}(\mathtt{E}_i)) + (1-\sum_{j=1}^{n_f}p_j(\mathtt{E}_j))C^s$$

where n_f is the number of components fixed before a service call. Recall that $\mathtt{E}_1 = \mathtt{E}$ by definition. Similarly, the expected cost of making nonbase observation o_i under the current state of information $\mathtt{E}$ is given by

$$\mathrm{ECO}(\mathtt{E}, o_i) = C_i^o + \sum_{k=1}^{r_i} ECR(\mathtt{E} \cup \{o_i = k\}) \Pr(o_i = k|\mathtt{E}) \quad (7)$$

Equation 7 says that the expected cost of a plan starting with a nonbase observation is the cost of observation (C_i^o, incurred with certainty) plus the expected cost of the repair sequences that would result under each possible state of the observation variable. We note that the repair sequence may be different for every possible outcome of the observation.

If $\mathrm{ECR}(\mathtt{E}) < \mathrm{ECO}(\mathtt{E}, o_i)$ for any nonbase observation o_i, then we choose not to make an additional nonbase observation, but rather to observe and possibly repair that component with the maximum probability-to-cost ratio. Otherwise, we choose to observe that o_i with the lowest ECO. Once a repair or nonbase observation has been carried out, we update the information state $\mathtt{E}$, and repeat the cycle. When we update the information state, if a repair action has been carried out, we must remove all observations that correspond to nodes that are descendants of the repair-action node, because the repair action could have changed these observations. The net result of this procedure is a troubleshooting plan consisting of interleaved observation–repair actions and nonbase observations.

4 Calculation of the Probability of Repair

Bayesian networks represent uncertain relationships among observations, and inference algorithms for Bayesian networks allow us to compute the probability of component faults and observations given evidence. When we move to a troubleshooting model, we must also be able to represent actions in the world and the ensuing changes in the state of the device. Furthermore, we must be able to predict (under uncertainty) the state of the system given some action, before we take that action. In our simple model, for example, we must be able to compute $\Pr(e = \mathtt{Normal}|Repair(c_i), \mathtt{E})$, the probability that the device will perform properly after we repair component c_i, given that we know $\mathtt{E}$ before we fix c_i. In this section, we address these issues.

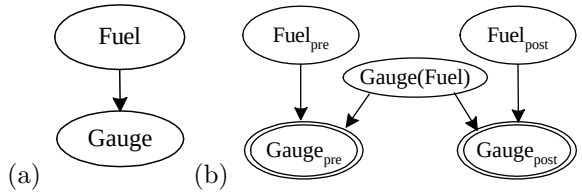

Figure 2: (a) A Bayesian network for the interaction between fuel in tank and fuel gauge. (b) A response network for determining the probability that gauge will read not empty after we fill the tank with gas, given that the gauge currently reads empty.

4.1 Response Networks

Consider the simple causal relationship between the fuel in a gas tank ("Fuel"), which we model as having states empty ($\mathtt{e}$) or not empty ($\neg\mathtt{e}$), and the reading on a fuel gauge ("Gauge"), which we also model as having states empty and not empty. The two nodes are dependent as depicted in the Bayesian network in Figure 2a.

Now suppose we observe the gauge to be empty, and we want to determine the probability that the gauge will read not empty, after we fill the tank with gas. We can do so, using the Bayesian network shown in Figure 2b. In this network, "Fuel$_{pre}$" and "Fuel$_{post}$" represent whether or not the tank is empty before and after we fill the tank, respectively (thus, "Fuel$_{post}$" = $\neg\mathtt{e}$). Similarly, "Gauge$_{pre}$" and "Gauge$_{post}$" represent whether or not the gauge reads empty before and after we fill the tank, respectively (thus, "Gauge$_{pre}$" = $\mathtt{e}$). The node "Gauge(Fuel)", called a *mapping variable*, represents all of the possible mappings from "Fuel" to "Gauge". As shown in Table 1, this node has four states: (1) `ok`, where the gauge reads empty if and only if there is no fuel, (2) `stuck on empty`, where the gauge always reads empty, (3) `stuck on not empty`, where the gauge always reads not empty, and (4) `backwards`, where the gauge reads empty if and only if there is fuel. The node "Gauge$_{pre}$" is a deterministic function of its cause "Fuel$_{pre}$" and the mapping variable "Gauge(Fuel)",

Table 1: The four possible mappings between "Fuel" and "Gauge".

	ok		stuck on empty		stuck on not empty		backwards	
Fuel	e	¬e	e	¬e	e	¬e	e	¬e
Gauge	e	¬e	e	e	¬e	¬e	¬e	e

as indicated by the double ovals around the node "Gauge$_{pre}$". For example, if "Fuel$_{pre}$" is empty and "Gauge(Fuel)" is `backwards`, then "Gauge$_{pre}$" will be not empty. The node "Gauge$_{post}$" is the same deterministic function of cause "Fuel$_{post}$" and the node "Gauge(Fuel)". The uncertainty in the relationship between "Fuel" and "Gauge", formerly associated with the variable "Gauge", now is associated with the variable "Gauge(Fuel)". In effect, we have extracted the uncertainty in the relationship between these two variables, and moved this uncertainty to the mapping variable. The probabilities associated with the node "Gauge(Fuel)" are constrained by, but not necessarily determined by, the probabilities in the original Bayesian network. We return to this issue in the following section.

By using a single node "Gauge(Fuel)" to represent the mappings from "Fuel" to "Gauge" both before and after the action is taken, we have assumed that these mappings are not affected by the action(s) that affect "Fuel". We say that "Gauge(Fuel)" is *unresponsive* to the actions that affect "Fuel". In general, a domain or mapping variable is said to be unresponsive to a set of actions if the outcome of that variable can not be affected by those actions.

Suppose we have a Bayesian network for a set of domain variables $\mathcal{U} = \{x_1, \ldots, x_n\}$. Further, suppose that we want to answer questions of the form: "What would be the probability of $\mathcal{X} \subseteq \mathcal{U}$ if we were to take some action, given that we now observe $\mathcal{Y} \subseteq \mathcal{U}$." To answer such questions, we construct a new Bayesian network as we did in our example. First, we label each domain variable as being either responsive or unresponsive to the action. Second, we introduce a mapping node $x_i(\Pi_i)$ for each responsive domain variable x_i that represents the possible mappings from the parents of x_i to x_i itself. Provided Π_i are causes for x_i, each such mapping variable will be unresponsive to the action. Third, we assess the probabilistic relationships among all the unresponsive variables, including the mapping variables. Fourth, we copy every responsive variable; the first and second instance of each such variable represents that variable before and after the action is taken, respectively. Fifth, we make both copies of the responsive variable x_i the same deterministic function of its old parents Π_i and the mapping node $x_i(\Pi_i)$, as in our example. Finally, we identify

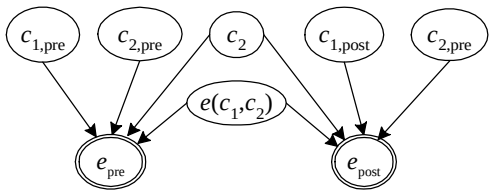

Figure 3: A response network for troubleshooting a two-component device.

those domain nodes in the post-action network that are affected directly by our action, break the arcs from their parents, and set the states of these nodes to their values as determined by our action.

We refer to Bayesian networks constructed in this manner as *response networks.* Figure 3 shows a response network for troubleshooting a device with output e consisting of two components c_1 and c_2. The Bayesian network in the figure allows us to compute the probability that the device will function after we repair c_1, given that the device is observed not to function before we do the repair—that is, $\Pr(e_{post} = \texttt{Normal} | Repair(c_1), e_{pre} = \neg\texttt{Normal})$. Note that there is only one copy of the variable c_2 in this network, reflecting the assertion that c_2 is unresponsive to the repair of c_1.

This transformation is described in more detail in Balke and Pearl (1994) and Heckerman and Shachter (1994). We emphasize that, in both works, the authors require that the original Bayesian network reflect causal interaction, not simply probabilistic dependence.

4.2 Causal Independence

A problem with constructing response networks is that the number of states of each mapping node m_i can be large, making the assessment of probabilities for these nodes intractable. In particular, if x_i has r_i states, and if the parents of x_i have q_i states, then the number of states of m_i will be $r_i^{q_i}$. In this section, we present a simplification that substantially reduces and sometimes eliminates entirely the need for these assessments.

When a set of variables $\{c_1, \ldots, c_n\}$ are the cause of an effect e, it is often reasonable to apply a particular form of conditional independence to this interaction. Heckerman (1993) introduces this form of conditional independence, which he calls *causal independence.* To define causal independence, we associate a set of variables indexed by time with each cause and with the effect. We use c_{jt} to denote the variable associated with cause c_j at time t, and e_t to denote the variable

associated with the effect at time t. For all times t and t', we require the variables c_{jt} and $c_{jt'}$ to have the same set of instances. Finally, for each cause, we designate some state of its associated variables to be *distinguished.* For most real-world models, this state will be the normal state, but we do not require this association. We use $*$ to denote the distinguished state for c_{jt}. Under these conditions, we say that $c_1, \ldots, c_n$ *are causally independent with respect to* e if the following set of conditional-independent assertions hold:

$$\forall t < t' \; \forall c_j \;\; (e_{t'} \perp c_{1t}, \ldots, c_{j-1,t}, c_{j+1,t}, \ldots, c_{nt} \mid e_t, c_{jt} = *, c_{jt'}, c_{kt} = c_{kt'} \text{ for } k \neq j)$$

hold, where $(X \perp Y|Z)$ denotes the conditional-independence assertion "the sets of variables X and Y are independent, given Z." These assertions state that if cause c_j makes a transition from its distinguished state at time t ($c_{jt} = *$) to some (possibly different) state at time t' ($c_{jt'}$), and if no other cause makes a transition during this time interval, then the probability distribution over the effect at time t' depends only on the state of the effect at time t and on the transition made by c_j; the distribution does not depend on the states of the other causes. We note that causal independence refers to the relationships between a set of causes and an effect; it does not refer to the relationships among the set of causes. In fact, causal independence may hold even when the causes themselves are dependent.

Heckerman and Breese (1994a,1994b) show that when causal independence holds for the interaction between $\{c_1, \ldots, c_n\}$ and e, and when e_0 (e when all causes take on their distinguished state) is a constant, then we can express the relationship between the causes and the effect atemporally as shown in Figure 4. In the figure, node e'_i represents e when all nodes but c_i take on their distinguished state. Each node e'_i depends only on c_i, and e is a deterministic function (f) of nodes e'_1, ..., e'_n. We sometimes refer to e'_i as the *mediator* of cause c_i. Heckerman and Breese also show that the mapping variables associated with the variables e' in a response network must be independent.

If e_0 is not a constant, we can introduce a dummy cause c_l that is always instantiated to a nondistinguished state. In this case, we can express the uncertainty in e_0 as uncertainty in e'_l, leaving e_0 a constant in the mathematical formalism. Henrion (1989) calls c_l a *leak* cause for e.

An example of causal independence is the *noisy max* relationship [Heckerman, 1993]. In this special case, each cause c_i is binary, and the effect e and all variables e'_i take on ordered states $e0 < e1 < \ldots < em$, and $e_0 = e0$. The function f returns the state of e that is the maximum of its inputs. Another example of causal

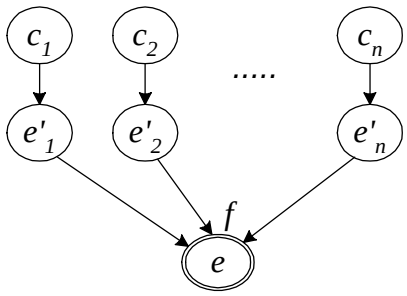

Figure 4: A Bayesian network for mediator independence.

independence is the *noisy adder* [Heckerman, 1993]. In this case, each cause c_i is binary, each variable e'_i takes on integer states $0 < 1 < \ldots < m$, and $e_0 = 0$. The function f returns the sum of its inputs. With n causes, e may vary from $-nm$ to nm.

Heckerman and Breese (1994a) have shown that the assumption of causal independence places constraints on the function f, but these constraints are not important for this discussion. We refer to the general interaction where (1) the mediator variables are independent, (2) the mapping variables $e'_i(c_i)$ associated with the mediator variables are independent, and (3) e_0 is a constant as *mediator independence*. In the remainder of this paper, we consider this general class of cause-effect interaction.

If all interactions in a domain satisfy mediator independence, then the knowledge-acquisition burden is significantly reduced. In particular, the only domain variables that require the introduction of mapping variables are causes and the variables e'_i, which have only one parent. Furthermore, by definition of e_0 and e'_i, and from our assumption that e_0 is a constant, it follows that $e'_i = e_0$, whenever c_i takes on its distinguished state. Thus, if c_i is a binary node, as is the case for the noisy max and noisy adder relationships, e'_i and its associated mapping node $e'_i(c_i)$ have the same number of states. Consequently, because the mapping variables are independent, the probabilities for these are determined uniquely by the probabilities encoded in the original Bayesian network; and the transformation from a Bayesian network to a response network can be automated.[3]

[3]Even when the component c_i is not binary, we can automate the transformation, because we consider only actions on c_i that take it from an abnormal state to the normal (distinguished) state.

4.3 Single-Copy Approximation

Although the use of mediator independence facilitates knowledge acquisition, we are still left with the problem that a response network often contains many undirected cycles, making inference computationally expensive. In this section, we describe an additional approximation wherein the repair probabilities can be computed without copying the original Bayesian network.

Suppose our device satisfies the following conditions: (1) the interaction between the n components of the device $c_1, \ldots, c_n$ and the output of the device e satisfy mediator independence (with $e_0 = \texttt{Normal}$, a constant); and (2) the function f has the property that the output is normal if all inputs are normal. For example, the noisy max relationship (with e_0 corresponding to `Normal`), and the noisy adder relationship (with 0 corresponding to `Normal`) satisfy these conditions.

Let `sf` denote the event that a single component is responsible for any observed failure of the device. That is, `sf` denotes the event that if e is not `Normal`, then exactly one e'_i is not `Normal`. Under conditions 1 and 2 and the assumption that `sf` is true, if e is observed to be some abnormal state k, then we have

$$\Pr(e_{post} = \texttt{Normal} | Repair(c_i), e_{pre} = k, \texttt{sf}) = \Pr(e'_{i,pre} \neq \texttt{Normal} | e_{pre} = k, \texttt{sf}) \quad (8)$$

To see this fact, first note that given the single-fault assumption `sf` and the observation $e_{pre} = k$, we know that exactly one node e'_i is abnormal. As discussed in the previous section, when c_i is normal, so is e'_i. Thus, we know c_i is abnormal. If we repair this node, then c_i and hence e'_i will become normal. Also, all nodes $e'_j, j \neq i$ are unresponsive to the repair action and will remain normal. Thus, all e'_i will be normal, and by condition 2, the device will be normal. Conversely, if we repair node $c_j, j \neq i$, e'_i will remain abnormal, and so will e. Thus, given the events `sf` and $e_{pre} = k$, the event $e_{post} = \texttt{Normal}$ given the repair of c_i is equivalent to the event $e'_{i,pre} \neq \texttt{Normal}$, which establishes Equation 8.

In practice, the single-fault assumption `sf` may not be true. Nonetheless, for many troubleshooting domains, we have used the right-hand-side of Equation 8 in place of the left-hand-side, as an approximation, because the right-hand-side may be computed without copying the Bayesian network for a device. In so doing, we rescale the probabilities for the nodes e'_i, so that the sum over the probabilities of all non-normal states of all nodes e'_i is equal to one.

For the domains we have considered, we have found empirically that the approximation is a good one.

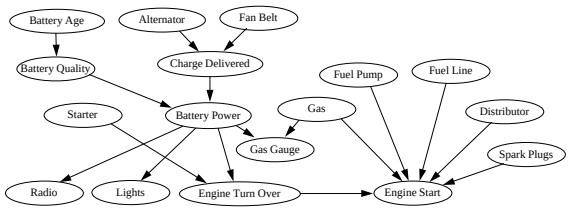

Figure 5: A Bayesian network for an automobile troubleshooting problem.

There are two explanations for this observation. First, given the observation that e is abnormal, the single-fault assumption is likely to be true, because it is unlikely that two components of a device will fail at roughly the same time. Second, we employ this approximation in the context of our other approximations, which can themselves introduce substantial errors.

We note that conditions 1 and 2 are not as strong as they first appear to be. In particular, it is common to decompose a device into subsystems. For example, Figure 5 shows a Bayesian network for an automobile troubleshooting problem. The components are "Spark Plugs," "Starter," "Battery," "Fuel Pump," "Fuel Line," "Fuel," and the device node e is "Engine Starts." In this network, "Engine Starts" is decomposed into subsystems "Engine Turns Over" and "Fuel System." Both subsystems exhibit a noisy max relationship with their parent components. In addition, "Engine Starts" exhibits a noisy max relationship with its parents, where "Engine Starts" is `Abnormal` with certainty, if either subsystem is `Abnormal`. Consequently, the effective or marginal relationship between the components and "Engine Starts" satisfy conditions 1 and 2. In general, Bayesian networks resulting from device decomposition, including recursive decomposition, will often satisfy our conditions.

The automobile troubleshooting example is somewhat more general than the model we have described. In particular, there are several nonbase observation variables—"Lights", "Fuel Gauge", and "Battery Age"—that can affect the probabilities of component failure. Nonetheless, given that some or all of these variables are instantiated, Equation 8 is still valid under conditions 1 and 2, provided we replace $e_{pre} = k$ with all observed evidence `E`. Also, if the fuel gauge is electric, then we should add an arc from "Battery" to "Fuel Gauge." Although "Battery" and "Fuel" become dependent when "Fuel Gauge" is observed, this fact does not invalidate condition 1.

5 Application of the Method

In this section, we summarize the approach described in the previous sections, with some operational details. We describe the procedure consistent with the single-copy approximation described in Section 4.3.

5.1 Representation of the Problem as a Bayesian Network

We begin the construction of a troubleshooting model by building a Bayesian network, indicating the causal relationships among components, observations, and subsystems (if any). A sample Bayesian network for automobile starting problems in shown in Figure 5. Next, under the assumption that all nondeterministic parent-child relationships satisfy mediator independence, we introduce mediator nodes e'_i and causal leaks as necessary. This expansion can be automated, and we have developed software for this purpose.

Next, we partition the nodes of the original Bayesian network into the following classes:

Problem-Defining Node:
: This node represents whether or not the device is working properly. It is the single output node e described in Section 1. In Figure 5, the node labeled "Engine Starts" is the problem defining node.

Repairable Nodes:
: These nodes should be direct or indirect predecessors of the problem-defining node. In Figure 5, the repairable nodes are shown in boldface type. For each repairable node, we assess C_i^r—the cost of repairing component x_i—and, if appropriate, C_i^o the cost of observing whether or not that component actually is faulty, before a repair is attempted.

Unrepairable Nodes:
: These nodes are similar to repairable nodes in that they directly contribute to the behavior of the device, and should be direct or indirect predecessors of the problem-defining node. The nodes are repairable in principle, but are deemed unrepairable by the current troubleshooter. Unrepairable nodes in Figure 5 are "Fuel Pump" and "Fuel Line."

Observation Nodes:
: These nodes represent nonbase observations that we can make during the course of troubleshooting. These nodes are not eligible for repair or other forms of intervention. Observational nodes may include subsystem nodes. In Figure 5, the nodes labeled "Lights," "Fuel Gauge," "Battery Age," and "Engine Turns Over" are observation nodes. For each such node, we assess the cost of observation.

Finally, we need to designate a cost of service, C^s. The service cost is interpreted as a fixed cost that leads to a functioning device with certainty.

5.2 Generation of Recommendations at Runtime

A troubleshooting session is started by observing the problem defining node to be abnormal. We then evaluate various observation–repair actions and nonbase observations as described in Section 3. As mentioned in Section 4.3, in order to apply Equation 8, we renormalize the probabilities of the nodes e'_i, given each new observation or action.

6 Empirical Results

We have applied our approach to troubleshooting printing problems, automobile startup problems, copier feeder systems, and gas turbines. The results have been satisfying along a number of dimensions. The models have been easy to build and assess. In addition, the plans generated have been reasonable. In the remainder of this section, we discuss experiments that measure the performance of our decision-theoretic approach.

We have developed a Monte-Carlo technique for estimating troubleshooting costs for a given planner and domain. The basic idea is to use a Bayesian network for a given device to generate a relatively large set of problem instances where one or more faults are known to have occurred. We then apply the planning method to each case, recording the sum of costs of each observation and repair action. A histogram of these total costs then provides a good estimate of the distribution of troubleshooting costs.

The method relies on an *oracle Bayesian network* to generate sample problems and to predict the effects of repair actions. This Bayesian network is a response network where (1) every variable that is responsive to any repair action or any combination of actions has a corresponding mapping variable, and (2) there is only one copy of every responsive variable. The oracle Bayesian network in these experiments is identical to the response networks used in the decision-theoretic planner, insuring that the planner has the "correct" model. This assumption could be relaxed in future experiments.

To generate a case, we set the problem-defining node to `Abnormal`, and update the probabilities throughout the oracle network. We then choose one of the uncertain (unresponsive) nodes, sample its updated probability distribution, and set its state accordingly. We repeat this process until all fixable, unfixable, leak,

and mapping nodes are set. This procedure guarantees that the device fails in every case, and the samples obey the probability distributions in the oracle network.

When a planner posts a repair to the oracle, we set the corresponding fixable node in the oracle Bayesian network to `Normal`. When a planner queries the oracle Bayesian network for the state of an observation variable, we can compute this state from the oracle Bayesian network, because all uncertain nodes have been set.

In the results that follow, we compare our decision-theoretic planner, a random planner, and a static planner. The decision-theoretic planner posts a mixture of repairs and queries to the oracle Bayesian network—as we have described—until the oracle reports that the device is repaired. The random planner posts repairs at random (without repetition) until the oracle reports that the device is repaired. The static planner posts repairs in a static order: components with lower observation costs are repaired first, with ties broken by repair cost. Again, the static planner continues until the oracle reports that the device is repaired.

We generated 1000 troubleshooting cases using a version of the automobile Bayesian network shown in Figure 5. The average cost of the decision-theoretic, static, and random planners was \$154, \$298, and \$457, respectively. Figure 6 shows a histogram of the costs for each of the planners for the 1000 cases. We see that the decision-theoretic planner has significantly lower costs than the other planners on average. For comparison, the average cost of repair for a planner that knows the causes of failure with certainty was \$127. This figure sets a lower bound on the expected cost of an optimal planner. Our decision-theoretic planner is close to this lower bound.

7 Conclusions and Future Work

We have developed an approximate decision-theoretic approach for generating troubleshooting plans under uncertainty that interleaves both observations and repair actions. Despite our approximations, we have seen that our planner produces expected troubleshooting costs that are close to optimal and significantly lower than simple planners for a domain of automobile troubleshooting.

In future work, we shall relax the assumptions used for approximating the expected costs of repair for various scenarios, specifically examining extensions for dependent costs of repair. We shall also investigate the benefits of using response networks for calculating repair probabilities.

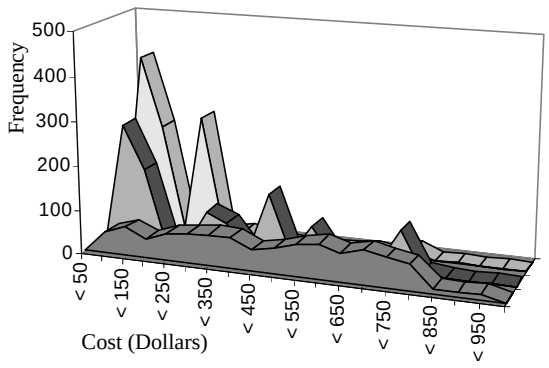

Figure 6: A histogram of costs for three planners. From front to back are the random, static, and decision-theoretic planners.

References

[Balke and Pearl, 1994] Balke, A. and Pearl, J. (1994). Probabilistic evaluation of counterfactual queries. In *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence,* Seattle, WA. Morgan Kaufmann.

[Breese et al., 1992] Breese, J., Horvitz, E., Peot, M., Gay, R., and Quentin, G. (1992). Automated decision-analytic diagnosis of thermal performance in gas turbines. In *Proceedings of the International Gas Turbine and Aeroengine Congress and Exposition,* Cologne, Germany, American Society of Mechanical Engineers.

[Heckerman, 1993] Heckerman, D. (1993). Causal independence for knowledge acquisition and inference. In *Proceedings of Ninth Conference on Uncertainty in Artificial Intelligence,* Washington, DC, pages 122–127. Morgan Kaufmann.

[Heckerman and Breese, 1994a] Heckerman, D. and Breese, J. (1994a). Causal independence for Bayesian-network knowledge acquisition and inference. Technical Report MSR-TR-94-08, Microsoft.

[Heckerman and Breese, 1994b] Heckerman, D. and Breese, J. (1994b). A new look at causal independence. In *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence,* Seattle, WA, pages 286–292. Morgan Kaufmann.

[Heckerman and Shachter, 1994a] Heckerman, D. and Shachter, R. (1994a). A decision-based view of causality. In *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence,* Seattle, WA, pages 302–310. Morgan Kaufmann.

[Henrion, 1989] Henrion, M. (1989). In Kanal, L., Levitt, T., and Lemmer, J., editors, *Uncertainty in Artificial Intelligence 3,* pages 161–174. North-Holland, New York, 1989.

[Howard and Matheson, 1981] Howard, R. and Matheson, J. (1981). Influence diagrams. In Howard, R. and Matheson, J., editors, *Readings on the Principles and Applications of Decision Analysis*, volume II, pages 721–762. Strategic Decisions Group, Menlo Park, CA.

[Jensen et al., 1990] Jensen, F., Lauritzen, S., and Olesen, K. (1990). Bayesian updating in recursive graphical models by local computations. *Computational Statisticals Quarterly*, 4:269–282.

[Kalagnanam and Henrion, 1990] Kalagnanam, J. and Henrion, M. (1990). In Shachter, R., Levitt, T., Kanal, L., and Lemmer, J., editors, *Uncertainty in Artificial Intelligence 4,* pages 271–281. North-Holland, New York, 1990.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.