A* Wars: The Fight for Improving A* Search for Troubleshooting with Dependent Actions

Thorsten J. Ottosen and Finn V. Jensen
Department of Computer Science
Aalborg University
9220 Aalborg, Denmark

Abstract

Decision theoretic troubleshooting combines Bayesian networks and cost estimates to obtain optimal or near optimal decisions in domains with inherent uncertainty. We use the well-known A* algorithm to find optimal solutions in troubleshooting models where different actions may fix the same fault. We prove that a heuristic function proposed by Vomlelová and Vomlel is monotone in models without questions, and we report on recent work on pruning. Furthermore, we experimentally investigate a hybrid approach where A* is combined with a method that sometimes avoid branching. The method is based on an analysis of the dependency between actions as suggested by Koca and Bilgiç.

1 Introduction

Imagine that you have a device which has been running well up to now, but suddenly it is malfunctioning. A set of *faults* $\mathcal{F}$ describes the possible causes to the problem. To fix the problem you have a set $\mathcal{A}$ of *actions*, which may fix the problem and a set $\mathcal{Q}$ of *questions*, which may help identifying the problem. Each action or question S has a positive *cost* $\mathrm{C_S}(\boldsymbol{\varepsilon})$ possibly depending on evidence $\boldsymbol{\varepsilon}$. Your task is to fix the problem as cheaply as possible. In this paper we do not consider questions.

When actions in a model can remedy sets of faults that overlap, we say that the model has *dependent actions*. Finding an optimal solution in models with dependent actions is of great practical importance since dependent actions can be expected to occur in many non-trivial domains. However, all non-trivial troubleshooting scenarios have been shown to be NP-hard—this includes models with dependent actions (Vomlelová, 2003).

Two different approaches have previously been used for finding optimal strategies: (Jensen et al., 2001) describes a branch & bound algorithm whereas (Vomlelová and Vomlel, 2003) describes an AO* algorithm. The AO* algorithm can be used for models with questions, but since a model without questions does not lead to AND-nodes in the search tree, we only need to consider the simpler A* algorithm (Hart et al., 1968) (Dechter and Pearl, 1985) for models with dependent actions.

We can summarize our troubleshooting assumptions as follows:

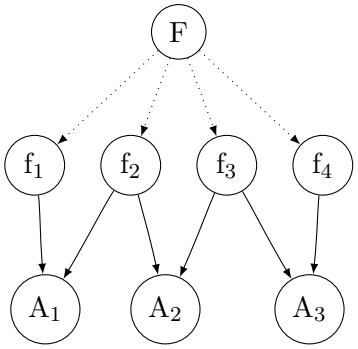
Assumption 1. *There are no questions.*

Assumption 2. *There is a single fault present when troubleshooting begins.* (This implies that we can have a single fault-node F with states $\mathrm{f}_i \in \mathcal{F}$.)

Assumption 3. *Actions are conditionally independent given evidence on the fault node.*

Assumption 4. *The cost of actions* $\mathrm{C_A}(\boldsymbol{\varepsilon})$ *is independent from evidence* $\boldsymbol{\varepsilon}$.

We use the following notation. The model provides for all $\mathrm{A} \in \mathcal{A}$ and $\mathrm{f} \in \mathcal{F}$ probabilities $\mathrm{P}(\mathrm{f}\,|\,\boldsymbol{\varepsilon})$, $\mathrm{P}(\mathrm{A}\,|\,\boldsymbol{\varepsilon})$ and $\mathrm{P}(\mathrm{A}\,|\,\mathrm{f})$, where $\boldsymbol{\varepsilon}$ is evidence. In Figure 1 is shown a simple model with dependent actions. We have some initial evidence $\boldsymbol{\varepsilon}^0$, and in the course of executing actions we collect further evidence. We write $\boldsymbol{\varepsilon}^i$ to denote that the first i actions have failed ($\boldsymbol{\varepsilon}^0 \subseteq \boldsymbol{\varepsilon}^i$), and we have by assumption $\mathrm{P}\big(\boldsymbol{\varepsilon}^0\big) = 1$ because



	f_1	f_2	f_3	f_4
$P(a_1 \mid F)$	1	1	0	0
$P(a_2 \mid F)$	0	1	1	0
$P(a_3 \mid F)$	0	0	1	1
$P(F)$	0.20	0.25	0.40	0.15
$C_{A_1} = C_{A_2} = C_{A_3} = 1$				

Figure 1: Left: a simple model for a troubleshooting scenario with dependent actions. The dotted lines indicate that the faults f_1 to f_4 are states in a single fault node F. A_1, A_2 and A_3 represent actions, and parents of an action node A are faults which may be fixed by A. Right: the quantitative part of the model.

the device is faulty. When action A has failed, we write $A = \neg a$ whereas $A = a$ means that it has succeeded. We often abbreviate $P(A = \neg a)$ as $P(\neg a)$. The presence of the fault f is written $F = f$, but we often abbreviate the event simply as f. Furthermore, we write $P(\boldsymbol{\varepsilon} \cup f)$ when we really had to write $P(\boldsymbol{\varepsilon} \cup \{f\})$. The set of faults that can be repaired by an action A is denoted $\mathrm{fa}(A)$. For example, in Figure 1 we have $\mathrm{fa}(A_2) = \{f_2, f_3\}$. In models where actions can have $P(a \mid \boldsymbol{\varepsilon}) = 1$, $\mathrm{fa}(\cdot)$ is a dynamic entity which we indicate by writing $\mathrm{fa}(\cdot \mid \boldsymbol{\varepsilon})$. The set of remaining actions is denoted $\mathcal{A}(\boldsymbol{\varepsilon})$, and $\mathcal{A}(f \mid \boldsymbol{\varepsilon}) \subseteq \mathcal{A}(\boldsymbol{\varepsilon})$ is the set of remaining actions that can fix f.

When there are no questions, a *troubleshooting strategy* is a sequence of actions $s = \langle A_1, \ldots, A_n \rangle$ prescribing the process of repeatedly performing the next action until the problem is fixed or the last action has been performed. To compare sequences we use the following definition:

Definition 1. The *expected cost of repair* (ECR) of a troubleshooting sequence $s = \langle A_1, \ldots, A_n \rangle$ with costs C_{A_i} is the mean of the costs until an action succeeds or all actions have been performed:

$$\mathrm{ECR}(s) \;=\; \sum_{i=1}^{n} C_{A_i}(\boldsymbol{\varepsilon}^{i-1}) \cdot P\big(\boldsymbol{\varepsilon}^{i-1}\big) \;.$$

We then define an *optimal sequence* as a sequence with minimal ECR. Also, $\mathrm{ECR}^*(\boldsymbol{\varepsilon})$ is the ECR for an optimal sequence of the actions $\mathcal{A}(\boldsymbol{\varepsilon})$ given $\boldsymbol{\varepsilon}$.

Example 1. Consider a sequence for the model in Figure 1:

$$\begin{aligned}
&\mathrm{ECR}(\langle A_2, A_3, A_1 \rangle) = \\
&\qquad C_{A_2} + P(\neg a_2) \cdot C_{A_3} + P(\neg a_2, \neg a_3) \cdot C_{A_1} \\
&= \; C_{A_2} + P(\neg a_2) \cdot C_{A_3} \\
&\qquad + P(\neg a_2) \cdot P(\neg a_3 \mid \neg a_2) \cdot C_{A_1} \\
&= \; 1 + \frac{7}{20} \cdot 1 + \frac{7}{20} \cdot \frac{4}{7} \cdot 1 = 1.55 \;.
\end{aligned}$$

A crucial definition is that of efficiency:

Definition 2. The *efficiency* of an action A given evidence $\boldsymbol{\varepsilon}$ is

$$\mathrm{ef}(A \mid \boldsymbol{\varepsilon}) \;=\; \frac{P(A = a)}{C_A(\boldsymbol{\varepsilon})} \;.$$

2 Monotonicity of the heuristic function ECR

A* (and AO*) is a best-first search algorithm that works by continuously expanding a frontier node n for which the value of the *evaluation function*

$$f(n) \;=\; g(n) + h(n), \qquad (1)$$

is minimal until finally a goal node t is expanded. The cost between two nodes n and m (m reachable from n) is denoted $c(n, m)$, and the function $g(n)$ is the cost from the start node s to n whereas $h(n)$ is the *heuristic function* that guides (or misguides) the search by estimating the cost from n to a goal node t.

If $h(n) \equiv 0$, A* degenerates to Dijkstra's algorithm. The cost of the shortest path from s to n is denoted $g^*(n)$, and from n to t it is denoted $h^*(n)$. Finally, the evidence gathered about actions from s to n is denoted $\boldsymbol{\varepsilon}^n$ ($\boldsymbol{\varepsilon}^0 \subseteq \boldsymbol{\varepsilon}^n$).

Definition 3. A heuristic function $h(\cdot)$ is *admissible* if

$$h(n) \quad \leq \quad h^*(n) \quad \forall n \ .$$

When A* is guided by an admissible heuristic function, it is guaranteed to find an optimal solution (Hart et al., 1968).

(Vomlelová and Vomlel, 2003) have suggested the following heuristic function for use in troubleshooting:

Definition 4. Let $\mathcal{E}$ denote the set containing all possible evidence. The function $\underline{\mathrm{ECR}} : \mathcal{E} \mapsto \mathcal{R}^+$ is defined for each $\boldsymbol{\varepsilon}^n \in \mathcal{E}$ as

$$\underline{\mathrm{ECR}}(\boldsymbol{\varepsilon}^n) = \mathrm{P}(\boldsymbol{\varepsilon}^n) \cdot \sum_{\mathrm{f} \in \mathcal{F}} \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f}) \ .$$

Remark. In (Vomlelová and Vomlel, 2003) the factor $\mathrm{P}(\boldsymbol{\varepsilon}^n)$ is left out. However, the factor ensures that the decomposition in Equation 1 takes the simple form

$$f(n) \quad = \quad \underbrace{\mathrm{ECR}\,(\boldsymbol{\varepsilon}^{\mathrm{n}})}_{\mathrm{g(n)}} + \underbrace{\underline{\mathrm{ECR}}(\boldsymbol{\varepsilon}^n)}_{\mathrm{h(n)}} \ ,$$

where $\mathrm{ECR}\,(\boldsymbol{\varepsilon}^{\mathrm{n}})$ is the ECR for the sequence defined by the path up to n. We also define

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}) \quad = \quad \sum_{\mathrm{f} \in \mathcal{F}} \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon} \cup \mathrm{f}) \ .$$

The optimal cost $\mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f})$ is easy to calculate under Assumption 2-4: the optimal sequence is found by ordering the actions in $\mathcal{A}\,(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^{\mathrm{n}})$ with respect to descending efficiency (Kadane and Simon, 1977).

Example 2. Assume the fault f can be repaired by two actions A_1 and A_2 and that $\mathrm{P}(\mathrm{a}_1 \,|\, \mathrm{f}) = 0.9$ and $\mathrm{P}(\mathrm{a}_2 \,|\, \mathrm{f}) = 0.8$. Furthermore, let both actions have cost 1. Since instantiating the fault node renders the actions conditionally independent, $\mathrm{P}(\mathrm{a} \,|\, \boldsymbol{\varepsilon} \cup \mathrm{f}) = \mathrm{P}(\mathrm{a} \,|\, \mathrm{f})$ and the efficiencies of the two actions are 0.9 and 0.8, respectively. We get

$$\begin{aligned} \mathrm{ECR}^*(\boldsymbol{\varepsilon} \cup \mathrm{f}) &= \mathrm{ECR}\,(\langle \mathrm{A}_1, \mathrm{A}_2 \rangle) \\ &= \mathrm{C}_{\mathrm{A}_1} + \mathrm{P}(\neg \mathrm{a}_1 \,|\, \mathrm{f}) \cdot \mathrm{C}_{\mathrm{A}_2} \\ &= 1 + 0.1 \cdot 1 = 1.1 \ . \end{aligned}$$

Not only is $\underline{\mathrm{ECR}}(\cdot)$ easy to compute, it also has the following property (Vomlelová and Vomlel, 2003):

Theorem 1. *Under Assumption 2-4 the heuristic function* $\underline{\mathrm{ECR}}(\cdot)$ *is admissible, that is,*

$$\underline{\mathrm{ECR}}(\boldsymbol{\varepsilon}^n) \quad \leq \quad \mathrm{ECR}^*(\boldsymbol{\varepsilon}^n) \quad \forall \boldsymbol{\varepsilon}^n \in \mathcal{E}.$$

For a class of heuristic functions, A* is guaranteed to have found the optimal path to a node when the node is expanded (Hart et al., 1968):

Definition 5. A heuristic function $h(\cdot)$ is *monotone* if

$$h(n) \quad \leq \quad c(n, m) + h(m),$$

whenever m is a successor node of n.

Remark. Monotonicity is equivalent to the often used and seemingly stronger *consistency* property: $h(n) \leq c^*(n, m) + h(m)\ \forall n, m$.

Henceforth we let A_n denote the performed action on the edge from a node n to a successor node m in the search graph.

Proposition 1. *For the heuristic function* $\underline{\mathrm{ECR}}(\cdot)$ *under Assumption 1 and 4 monotonicity is equivalent to*

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) \quad \leq \quad \mathrm{C}_{\mathrm{A}_n} + \mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m) \, .$$

Proof. We have $c(n, m) = \mathrm{P}(\boldsymbol{\varepsilon}^n) \cdot \mathrm{C}_{\mathrm{A}_n}$ and $\mathrm{P}(\boldsymbol{\varepsilon}^m) = \mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \mathrm{P}(\boldsymbol{\varepsilon}^n)$ and so the common factor $\mathrm{P}(\boldsymbol{\varepsilon}^n)$ cancels out. □

Theorem 2. *Under Assumption 1-4 the heuristic function* $\underline{\mathrm{ECR}}(\cdot)$ *is monotone.*

Proof. The idea is to express $\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m)$ in terms of $\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n)$. To do that we consider the complement of the set $\mathrm{fa}(\mathrm{A}_n)$ which is the set of all faults that A_n cannot fix. For each $\mathrm{f} \in \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)$ Bayes' rule (conditioned) yields

$$\mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^m) \quad = \quad \frac{1 \cdot \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n)}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)},$$

because $\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n \cup \mathrm{f}) \equiv 1$. If we restrict $\underline{\mathrm{ECR}}_h(\cdot)$ to a subset of faults X, we shall abuse notation and write it

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon} \,|\, X) = \sum_{\mathrm{f} \in X} \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon} \cup \mathrm{f}) \ .$$

In particular, we must have

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) = \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)) + \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathrm{fa}(\mathrm{A}_n)). \quad (2)$$

We can furthermore define

$$\begin{aligned} \Delta\mathcal{F} &= \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m \,|\, \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)) \\ &\quad - \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)) \ , \end{aligned}$$

which is an extra cost because all faults in $\mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)$ are more likely. Similarly

$$\Delta\mathrm{fa}(\mathrm{A}_n) = \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m \,|\, \mathrm{fa}(\mathrm{A}_n)) - \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathrm{fa}(\mathrm{A}_n)) \ ,$$

is the cost lost or gained because A_n has been performed and can no longer repair the faults $\mathrm{fa}(\mathrm{A}_n)$. We can then express $\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m)$ by

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m) = \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) + \Delta\mathrm{fa}(\mathrm{A}_n) + \Delta\mathcal{F}, \quad (3)$$

The constant $\mathrm{ECR}^*(\cdot)$ factors implies

$$\Delta\mathcal{F} = \sum_{\mathrm{f} \in \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)} [\mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^m) - \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n)] \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f}).$$

Exploiting Bayes' rule (as explained above) and Equation 2 we get

$$\Delta\mathcal{F} = \left[\frac{1}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)} - 1\right] \cdot \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathcal{F} \setminus \mathrm{fa}(\mathrm{A}_n)) = \left[\frac{1}{\mathrm{P}(\neg \mathrm{a}_\mathrm{n} \,|\, \boldsymbol{\varepsilon}^\mathrm{n})} - 1\right] \cdot \left[\underline{\mathrm{ECR}}_\mathrm{h}(\boldsymbol{\varepsilon}^\mathrm{n}) - \underline{\mathrm{ECR}}_\mathrm{h}(\boldsymbol{\varepsilon}^\mathrm{n} \,|\, \mathrm{fa}(\mathrm{A}_\mathrm{n}))\right].$$

Inserting into Equation 3 yields

$$\begin{aligned} \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m) &= \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) + \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m \,|\, \mathrm{fa}(\mathrm{A}_n)) - \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathrm{fa}(\mathrm{A}_n)) \\ &\quad + \left[\frac{1}{\mathrm{P}(\neg \mathrm{a}_\mathrm{n} \,|\, \boldsymbol{\varepsilon}^\mathrm{n})} - 1\right] \cdot \left[\underline{\mathrm{ECR}}_\mathrm{h}(\boldsymbol{\varepsilon}^\mathrm{n}) - \underline{\mathrm{ECR}}_\mathrm{h}(\boldsymbol{\varepsilon}^\mathrm{n} \,|\, \mathrm{fa}(\mathrm{A}_\mathrm{n}))\right] \\ &= \frac{\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n)}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)} + \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m \,|\, \mathrm{fa}(\mathrm{A}_n)) - \frac{1}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)} \cdot \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n \,|\, \mathrm{fa}(\mathrm{A}_n)), \end{aligned}$$

and we rearrange the equation into

$$\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n) = \mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \underline{\mathrm{ECR}}(\boldsymbol{\varepsilon}^m) + \underbrace{\underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^n | \mathrm{fa}(\mathrm{A}_n)) - \mathrm{P}(\neg \mathrm{a}_n | \boldsymbol{\varepsilon}^n) \cdot \underline{\mathrm{ECR}}_h(\boldsymbol{\varepsilon}^m | \mathrm{fa}(\mathrm{A}_n))}_{\Delta}.$$

By Proposition 1, we have to prove $\Delta \leq \mathrm{C}_{\mathrm{A}_n}$. Because of Bayes' rule and Assumption 3 we have

$$\begin{aligned} &\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^m) = \\ &\qquad \mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n) \cdot \frac{\mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n)}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \boldsymbol{\varepsilon}^n)} \\ &\quad = \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n) \ . \end{aligned}$$

So we get

$$\Delta = \sum_{\mathrm{f} \in \mathrm{fa}(\mathrm{A}_n)} \mathrm{P}(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^n) \cdot \underbrace{[\mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f}) - \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon}^m \cup \mathrm{f})]}_{\delta}.$$

Because of the single-fault assumption, we only need to prove that $\delta \leq \mathrm{C}_{\mathrm{A}_n}$. We now index the actions in $\mathcal{A}\,(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^\mathrm{n})$ as follows:

$$\frac{\mathrm{P}(\mathrm{B}_i = \mathrm{b}_i \,|\, \mathrm{f})}{\mathrm{C}_{\mathrm{B}_i}} \geq \frac{\mathrm{P}(\mathrm{B}_{i+1} = \mathrm{b}_{i+1} \,|\, \mathrm{f})}{\mathrm{C}_{\mathrm{B}_{i+1}}} \quad \forall i.$$

In this ordering, we have $\mathrm{A}_n = \mathrm{B}_x$. The inequalities generalizes to

$$\mathrm{C}_{\mathrm{B}_i} \leq \frac{\mathrm{P}(\mathrm{B}_i = \mathrm{b}_i \,|\, \mathrm{f})}{\mathrm{P}(\mathrm{B}_j = \mathrm{b}_j \,|\, \mathrm{f})} \cdot \mathrm{C}_{\mathrm{B}_j} \quad \forall j > i \ . \quad (4)$$

In particular, this is true for $j = x$ which we shall exploit later.

Assume we have N dependent actions in $\mathcal{A}\,(\mathrm{f} \,|\, \boldsymbol{\varepsilon}^\mathrm{n})$. The first term of δ is then

$$\begin{aligned} \mathrm{ECR}^*(\boldsymbol{\varepsilon}^n \cup \mathrm{f}) &= \mathrm{ECR}^*(\langle \mathrm{B}_1, \ldots, \mathrm{B}_N \rangle) \\ &= \mathrm{C}_{\mathrm{B}_1} + \sum_{i=2}^{N} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \ . \end{aligned} \quad (5)$$

Assume that $x > 1$ (we shall deal with $x = 1$ later), then the second term of δ is

$$\begin{aligned} &\mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{ECR}^*(\boldsymbol{\varepsilon}^m \cup \mathrm{f}) = \\ &\qquad \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{ECR}^*(\langle \ldots, \mathrm{B}_{x-1}, \mathrm{B}_{x+1}, \ldots \rangle) \\ &\quad = \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \\ &\qquad \left[\mathrm{C}_{\mathrm{B}_1} + \sum_{i=2}^{x-1} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) + \frac{\sum_{i=x+1}^{N} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f})}{\mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f})}\right] . \end{aligned}$$

We see that the last term is also represented in Equation 5 and therefore cancels out. We get

$$\begin{aligned}\delta = {} & \mathrm{C}_{\mathrm{B}_1} \cdot [1 - \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f})] + \\ & [1 - \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f})] \cdot \sum_{i=2}^{x-1} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ & + \mathrm{C}_{\mathrm{A}_n} \cdot \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}),\end{aligned}$$

where the last term is a leftover from Equation 5. Using $\mathrm{P}(\neg \mathrm{a} \,|\, \boldsymbol{\varepsilon}) = 1 - \mathrm{P}(\mathrm{a} \,|\, \boldsymbol{\varepsilon})$ and Equation 4 we get

$$\begin{aligned}\delta = {} & \mathrm{C}_{\mathrm{B}_1} \cdot \mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f}) + \\ & \mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f}) \cdot \sum_{i=2}^{x-1} \mathrm{C}_{\mathrm{B}_i} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ & + \mathrm{C}_{\mathrm{A}_n} \cdot \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ \leq {} & \frac{\mathrm{P}(\mathrm{b}_1 \,|\, \mathrm{f})}{\mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f})} \cdot \mathrm{C}_{\mathrm{A}_n} \cdot \mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f}) + \\ & \mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f}) \cdot \sum_{i=2}^{x-1} \frac{\mathrm{P}(\mathrm{b}_i \,|\, \mathrm{f})}{\mathrm{P}(\mathrm{a}_n \,|\, \mathrm{f})} \cdot \mathrm{C}_{\mathrm{A}_n} \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ & + \mathrm{C}_{\mathrm{A}_n} \cdot \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ = {} & \mathrm{C}_{\mathrm{A}_n} \cdot \Big[\mathrm{P}(\mathrm{b}_1 \,|\, \mathrm{f}) + \\ & \sum_{i=2}^{x-1} \mathrm{P}(\mathrm{b}_i \,|\, \mathrm{f}) \cdot \prod_{j=1}^{i-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \\ & + \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \Big] \qquad (6) \\ = {} & \mathrm{C}_{\mathrm{A}_n} \cdot \Big[1 - \mathrm{P}(\neg \mathrm{b}_1 \,|\, \mathrm{f}) + \\ & (1 - \mathrm{P}(\neg \mathrm{b}_2 \,|\, \mathrm{f})) \cdot \mathrm{P}(\neg \mathrm{b}_1 \,|\, \mathrm{f}) \\ & + \cdots + \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \Big] \\ = {} & \mathrm{C}_{\mathrm{A}_n} \cdot \Big[1 - \mathrm{P}(\neg \mathrm{b}_2 \,|\, \mathrm{f}) \cdot \mathrm{P}(\neg \mathrm{b}_1 \,|\, \mathrm{f}) \\ & + \cdots + \prod_{j=1}^{x-1} \mathrm{P}(\neg \mathrm{b}_j \,|\, \mathrm{f}) \Big] \\ = {} & \mathrm{C}_{\mathrm{A}_n} \cdot 1,\end{aligned}$$

as required. This is not surprising if we look at the expression inside the parenthesis of Equation 6: the corresponding events are "B_1 fixes f, B_2 fixes f if B_1 did not fix f" etc. up to "none of the actions fixed f". These events form a sample space.

When $x = 1$, then $\delta = \mathrm{C}_{\mathrm{A}_n} - \mathrm{P}(\neg \mathrm{a}_n \,|\, \mathrm{f}) \cdot \mathrm{C}_{\mathrm{A}_n}$, so in all cases $\delta \leq \mathrm{C}_{\mathrm{A}_n}$ which completes the proof. □

Remark. It is quite straightforward to show that $\underline{\mathrm{ECR}}(\cdot)$ is not monotone when the model includes questions.

3 Pruning based on efficiency and ECR

We recently investigated the effect of a pruning method based on efficiency (Ottosen and Jensen, 2008). By considering two adjacent actions in an optimal troubleshooting sequence, the following has been proved about the efficiency (Jensen et al., 2001):

Theorem 3. *Let* $\mathrm{s} = \langle \mathrm{A}_1, \ldots, \mathrm{A}_n \rangle$ *be an optimal sequence of actions with independent costs. Then it must hold that*

$$\mathrm{ef}(\mathrm{A}_i \,|\, \boldsymbol{\varepsilon}^{i-1}) \quad \geq \quad \mathrm{ef}(\mathrm{A}_{i+1} \,|\, \boldsymbol{\varepsilon}^{i-1}),$$

for all $i \in \{1, \ldots, n-1\}$.

In Figure 2 it is illustrated how the theorem can be used for pruning. If we have the order $\mathrm{ef}(\mathrm{A}_1) > \mathrm{ef}(\mathrm{A}_2) > \mathrm{ef}(\mathrm{A}_3)$ at the root, we know that A_3 should never be the first action. Furthermore, after performing A_2, we know that A_1 should never be the second action. We call this *efficiency-based pruning*.

In summary, the theorem was very easy to exploit during the expansion of a node by keeping the actions sorted with respect to efficiency and by passing that information in the parent node. However, the results where a bit disappointing since it only gave a speed-up of a factor of 2-4.

We have since then tried to extend the idea by considering three adjacent actions instead of two. We call this *ECR-based pruning*. Figure 2 shows an overview of the pruning process. If we consider an arbitrary subset of three actions A_1, A_2, and A_3, we would normally need to compare six different sequences. However, if we have

calculated the efficiencies of the three actions at the local root node with evidence $\boldsymbol{\varepsilon}$, Theorem 3 leaves us with only three possible candidates. After the three sequences are expanded, the paths are coalesced into a single node in the search graph.

Now imagine that A* is about to expand A_3 in the sequence $\langle A_1, A_2, A_3\rangle$. We determine if the current node expansion is optimal by comparing it with the ECR of the sequence $\langle A_2, A_3, A_1\rangle$. (There is no need for comparing $\langle A_1, A_2, A_3\rangle$ with $\langle A_1, A_3, A_2\rangle$ since Theorem 3 has pruned the latter.) If we expand the sequence $\langle A_2, A_3, A_1\rangle$ first, the analysis is similar and we compare with the best of the two other sequences (again, the best sequence is found by applying Theorem 3).

There is no way to avoid calculating the full ECR of both sequences, and we have to traverse the search graph down to the root and up to the first node of the second path. Furthermore, this traversal means that we have to store child pointers in all nodes, and we also need to keep all expanded nodes in memory. This more than doubles the memory requirement.

In conclusion the method turned out to slow down A*. In a model with 19 action and 19 faults, Theorem 3 pruned 989k nodes whereas the ECR-based pruning prevented a mere 3556 expansions out of about 41k possible.

Theorem 2 also explains why the effect of the pruning is so small: if A* is guided by a monotone heuristics and expands a node, then the optimal path to that node has been found (Hart et al., 1968). This means that the sub-trees outlined with dotted edges in Figure 2 are never explored unless we really need to explore them. Should we discover a non-optimal sequence first, that path is not explored further until coalescing happens.

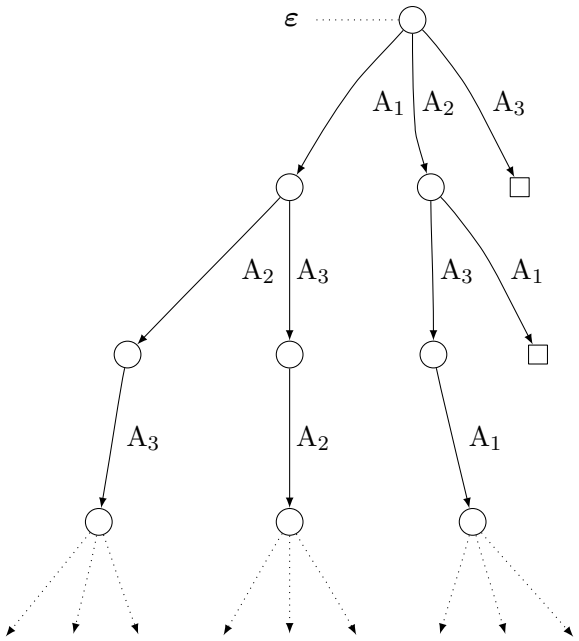


Figure 2: An overview of the pruning process for any subset of three actions. At the root of the subtree we have evidence $\boldsymbol{\varepsilon}$ and the actions are sorted with respect to efficiency, and we have $\mathrm{ef}(A_1|\boldsymbol{\varepsilon}) > \mathrm{ef}(A_2|\boldsymbol{\varepsilon}) > \mathrm{ef}(A_3|\boldsymbol{\varepsilon})$. Theorem 3 implies that we can prune the nodes ending in a square, and so we are left with only three possible sequences ($\langle A_1, A_2, A_3\rangle$, $\langle A_1, A_3, A_2\rangle$, and $\langle A_2, A_3, A_1\rangle$). After A* has discovered the last node in these three sequences, the three paths are subject to coalescing.

4 An A* hybrid approach

Troubleshooting with dependent actions was proved NP-hard by reduction from the exact cover by 3-sets problem (Vomlelová, 2003). Therefore, troubleshooting in models where each action can repair three or more faults is NP-hard. However, A* still has difficulties in finding a solution for models with much lower average dependency (that is, the average size of $\mathrm{fa}(\cdot)$ over all actions). Remarkably enough, we found that when the average dependency went down from 3 towards 1, then the running time increased many times. In the following we describe a hybrid method for models with an average dependency well below 3.

Figure 3 shows an example of the search tree explored by A* in our hybrid approach. Near the root node, A* is often forced to create a branch for each successor node. However, as we get closer to the goal nodes, branching is more likely to be avoided. The branching can be avoided

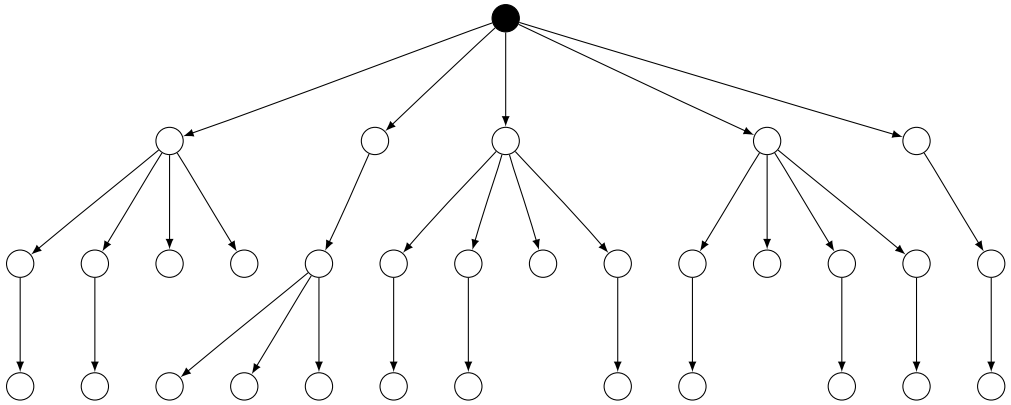


Figure 3: An example of what the search tree looks like in our hybrid approach. For some nodes, the normal A* branching is avoided, and near goal nodes this branching is almost avoided for all nodes. We can see that it might happen that the algorithm has to investigate all successors of a node even though the path down to that node was explored without branching.

because we are able to determine the optimal next step of the remaining sequence (see below).

This leads us to the following definitions:

Definition 6. A *dependency graph* for a troubleshooting model given evidence $\boldsymbol{\varepsilon}$ is the undirected graph with a vertex for each action $\mathrm{A} \in \mathcal{A}(\boldsymbol{\varepsilon})$ and an edge between two vertices A_1 and A_2 if $\mathrm{fa}(\mathrm{A}_1 \,|\, \boldsymbol{\varepsilon}) \cap \mathrm{fa}(\mathrm{A}_2 \,|\, \boldsymbol{\varepsilon}) \neq \emptyset$.

Definition 7. A *dependency set* for a troubleshooting model given evidence $\boldsymbol{\varepsilon}$ is a connectivity component in the dependency graph given $\boldsymbol{\varepsilon}$.

Definition 8. A *dependency set leader* for a troubleshooting model given evidence $\boldsymbol{\varepsilon}$ is the first action of an optimal sequence in a dependency set given $\boldsymbol{\varepsilon}$.

Dependency sets are important because the order of actions in the same dependency set does not change when actions outside the set are performed. This property has been exploited in the following theorem (Koca and Bilgiç, 2004):

Theorem 4. *Suppose we are able to calculate the dependency set leaders. Then the globally optimal sequence is given by the following algorithm:*

1. *Construct the dependency sets and retrieve the set leaders.*
2. *Calculate* $\mathrm{ef}(\cdot)$ *for all set leaders.*
3. *Select the set leader with the highest* $\mathrm{ef}(\cdot)$ *and perform it.*
4. *If it fails, update the probabilities, and continue in step (2).*

Our hybrid approach then simply works by finding the optimal sequence in dependency sets of a fairly small size. For this work we have restricted us to sets of a size < 4. At any point before expanding a node, if the most efficient action belongs to a dependency set of such a small size, we find the first action in that dependency set. If the dependency set consists of one or two actions, this calculation is trivial. If the dependency set has three actions, we find the first by comparing the three candidate sequences as we discussed in Section 3. Otherwise we simply expand the node as usual by inspecting all successors.

Table 4 shows the results of three versions of A*. We can see that the hybrid approach is somewhat slower for models with an average dependency between 2 and 3. This is because the hybrid approach spends time investigating the size of the dependency set of the most efficient action, but it rarely gets to exploit the benefits of a small dependency set. For an average dependency between 2.1 and 1.6 the hybrid approach becomes superior, and below 1.6 it becomes very fast.

Table 1: Results for the hybrid approach in models with 20 actions and 20 faults. The average dependency ranges between 3 and 1, and each action is usually associated with 1 to 3 faults. The time is measured in seconds. "A* " is A* with coalescing, "pruning-A* " is "A* " plus efficiency-based pruning and "hybrid-A* " is the hybrid approach based on "pruning-A* ". Already at an average dependency around 2.1 we see that the hybrid method wins.

Method	A*	pruning-A*	hybrid-A*
Dep.		*Time*	
3.0	33.56	11.48	12.27
2.9	42.11	12.42	21.97
2.8	62.14	15.08	27.52
2.7	45.03	14.38	21.61
2.6	29.86	10.20	12.39
2.5	86.52	22.20	29.61
2.4	31.55	12.39	12.00
2.3	65.19	19.11	21.28
2.2	80.56	21.28	29.38
2.1	50.28	18.72	9.78
2.0	83.75	27.70	20.05
1.9	62.88	16.77	10.64
1.8	127.59	35.09	18.72
1.7	102.17	25.36	14.42
1.6	133.17	39.14	25.41
1.5	122.08	27.59	0.92
1.4	164.84	41.16	4.25
1.3	139.89	39.44	0.13
1.2	168.42	38.13	0.00
1.1	160.42	39.42	0.00
1.0	159.95	38.08	0.00

5 Discussion

We originally investigated Theorem 2 because monotonicity plays an important role in promising bidirectional A* methods (Kaindl and Kainz, 1997). However, a bidirectional A* for troubleshooting is very difficult because there is no apparent way to start a search from the goal nodes using $\underline{\mathrm{ECR}}(\cdot)$.

The hybrid A* approach seems very promising. We still need to determine how large a dependency set that it pays off to solve. We expect that it will be most beneficial to solve small dependency sets by brute-force whereas dependency sets of medium size can be solved by a recursive call to hybrid-A* .

Acknowledgements

We would like to thank the reviewers for their valuable and detailed feedback.

References

Rina Dechter and Judea Pearl. 1985. Generalized best-first search strategies and the optimality af a*. *J. ACM*, 32(3):505–536.

P. E. Hart, N. J. Nilsson, and B. Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, SSC-4(2):100–7.

Finn V. Jensen, Uffe Kjærulff, Brian Kristiansen, Claus Skaanning, Jiri Vomlel, and Marta Vomlelová. 2001. The sacso methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15:321–333.

J. Kadane and H. Simon. 1977. Optimal strategies for a class of constrained sequential problems. *The Annals of Statistics*, 5:237–255.

Hermann Kaindl and Gerhard Kainz. 1997. Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research*, 7:283–317.

Eylem Koca and Taner Bilgiç. 2004. A troubleshooting approach with dependent actions. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI 2004: 16th European Conference on Artificial Intelligence*, pages 1043–1044. IOS Press.

Thorsten J. Ottosen and Finn V. Jensen. 2008. Better safe than sorry—optimal troubleshooting through A* search with efficiency-based pruning. In *Proceedings of the Tenth Scandinavian Conference on Artificial Intelligence*, pages 92–97. IOS Press.

M. Vomlelová and J. Vomlel. 2003. Troubleshooting: Np-hardness and solution methods. *Soft Computing Journal, Volume 7, Number 5*, pages 357–368.

Marta Vomlelová. 2003. Complexity of decision-theoretic troubleshooting. *Int. J. Intell. Syst.*, 18(2):267–277.