Troubleshooting with actions

A troubleshooting model consists of:
- A set of faults \( F \) (\( f \in F \)) that is potentially causing the problem.
- A set of actions \( A (A \in \mathcal{A}) \) that can fix the problem.
- A dynamic set of evidence \( e = \{ \Lambda \in \mathcal{A} | \Lambda \text{ failed to fix the problem (written } \Lambda = \neg a) \} \).
- A cost \( C(f, e) \) for each action \( A \), possibly depending on evidence \( e \).
- A Bayesian Network that provides \( P(A | e), P(A | i, e) \), and \( P(i | e) \).

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

\[
E(CR) = \frac{\sum_{i=1}^{n} C_i}{n-1}.
\]

**The goal is to determine a sequence with the lowest ECR.**

**Example (ECR calculation)**

Consider a sequence for the model in Figure 1:

\[
E(CR) = C_1 + P(\neg a_2 \mid A_1) \cdot C_2 + P(\neg a_3 \mid A_2) \cdot C_3 = 0.2 + 0.1 \cdot 0.0 + 0.2 \cdot 0.1 = 0.3.
\]

The set of faults that can be repaired by an action \( A \) is denoted \( fa(A) \). For example, in Figure 1 we have \( fa(A_1) = \{f_1, f_2\} \). In models where actions can have \( P(a_1 | e) = 1 \), \( fa(\cdot | e) \) is a dynamic entity which we indicate by writing \( fa(\cdot | e) \).

**Definition 2.** The efficiency of an action \( A \) given evidence \( e \) is the probability that the actions solves the problem divided by its cost, that is:

\[
eff(A | e) = \frac{P(A | e)}{C(A | e)}.
\]

A* and monotonicity of the function \( ECR \)

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

\[
f(n) = g(n) + h(n),
\]

is minimal until finally a goal node \( g \) is expanded (Hart et al., 1968). If node \( m \) is reachable from node \( n \), \( c(n, m) \) is the cost from \( n \) to \( m \). Then \( g(n) = c(s, n) \) where \( s \) is the start node, and \( h(n) \) is the heuristic function that guides (or misguides) the search by estimating the cost \( c(n, f) \). For Troubleshooting we have

\[
f(n) = ECR(e) + ECR(e) \cdot P(f | e).
\]

where ECR \( (e) \) is the ECR of the sequence defined by the path from \( s \) to \( n \).

**Example (ECR* calculation)**

Assume the fault \( f \) can be repaired by two actions \( A_1 \) and \( A_2 \) and that \( P(a_1 | f) = 0.9 \) and \( P(a_2 | f) = 0.8 \). Furthermore, let both actions have cost 1. Since instantiating the fault node renders the actions conditionally independent, \( P(f | e) = P(f) \) and the efficiencies of the two actions are 0.9 and 0.8, respectively. We get

\[
ECR(e | f) = ECR((A_1, A_2)) = C_1 + P(\neg a_2 | f) \cdot C_2 = 1 + 0.1 \cdot 0.1 = 1.01
\]

because the optimal sequence with independent actions is found by ordering the actions w.r.t. descending initial efficiency (Kadane and Simon, 1977).

**Definition 4.** A heuristic function \( h(i) \) is monotone if

\[
h(n) \leq h(n, m) + h(m),
\]

whenever \( m \) is a successor node of \( n \).

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

**Theorem 1.** Under the assumption of no questions, constant costs, a single initial fault, and conditional independence of actions given that the fault is known, then the heuristic function \( ECR(e) \) is monotone.

**Hybrid-A* Algorithm**

**Definition 5.** A dependency graph for a troubleshooting model given evidence \( e \) is the undirected graph with a vertex for each action \( A \in \mathcal{A}(e) \) and an edge between two vertices \( A_1 \) and \( A_2 \) if \( fa(A_1 | e) \cap fa(A_2 | e) \neq \emptyset \).

**Definition 6.** A dependency set leader for a troubleshooting model given evidence \( e \) is the first action of an optimal sequence in a connectivity component in the dependency graph given \( e \) (a dependency set).

**Theorem 2** (Koca and Biligiç, 2004). The globally optimal sequence is given by the following algorithm:

I. Construct the dependency sets and retrieve the set leaders.
II. Calculate \( e(i) \) for all set leaders.
III. Select the set leader with the highest \( e(i) \) and perform it.
IV. If it fails, update the probabilities, and continue in step (2).

**Hybrid-A*: We exploit Theorem 2 and avoid branching whenever the most efficient action belongs to a small dependency set (which is solved by brute-force).**

**Experimental results**

Figure 3: Comparison of normal \( A^* \) (Ottosen and Jensen, 2008) with the hybrid approach. The X-axis indicates average dependency of the model (that is, the average size of \( fa(\cdot | e) \) over all actions), and the Y-axis represents time in seconds. All models had 20 actions and 20 faults.
References


