When To Test?

Troubleshooting with Postponed System Test

Ottosen, Thorsten Jørgen; Jensen, Finn V.

Publication date:
2010

Document Version
Early version, also known as pre-print

Link to publication from Aalborg University

Citation for published version (APA):
Ottosen, T. J., & Jensen, F. V. (2010). *When To Test? Troubleshooting with Postponed System Test*. Department of Computer Science, Aalborg University.

When To Test?

Troubleshooting with Postponed System Test.

Thorsten J. Ottosen and Finn V. Jensen

Technical Report 10-001
Department of Computer Science
Aalborg University, Denmark

October 19, 2010

Abstract.

Troubleshooting is about computing a cost-efficient way to repair a faulty device. In this paper we investigate the most simple action-based troubleshooting scenario with a single extension: the overall system test is not required to be performed after each action, but it may be postponed until several actions have been performed. As shown by Kadane and Simon, the simple troubleshooting scenario is easily solvable in polynomial time. However, we conjecture that the new troubleshooting scenario is NP-hard and therefore describe an $\Theta(n^3)$ time heuristic. The new heuristic guarantees optimality for a class of models, and for other classes of models we benchmark it against the optimal solution and other simpler greedy heuristics. The benchmark is performed on artificial models as well as a real-world troubleshooting model, and they suggest that the heuristics are close to optimal.

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 by repairing one or more faults. An action, α, has a positive cost C_α and it has two possible outcomes, namely "α = yes" (the problem was fixed) and "α = no" (the action failed to fix the problem). Informally, our task is to determine the best sequence of actions until the problem is fixed or until no more actions remains to be carried out.

For an overview of state of the art of decision theoretic troubleshooting we refer the reader to (Jensen et al., 2001), (Langseth and Jensen, 2001), (Skaanning and Vomlel, 2001), (Langseth and Jensen, 2003), (Koca and Bilgiç, 2004a), (Koca and Bilgiç, 2004b), and (Warnquist et al., 2008). Basically all non-trivial troubleshooting domains are NP-hard (Vomlelová, 2003). The scenario investigated in this article is similar to troubleshooting with conditional costs, but the proof for NP-hardness does not carry through in this simple scenario. Nevertheless we conjucture that our scenario is NP-hard.

In traditional troubleshooting it has been assumed that each action is followed by a test that determines whether the entire system D is working. Furthermore, it has generally been assumed that the cost of this test, C_D, is so low that it should not even be modelled. We show that this assumption is incorrect in general and develop terminology and algorithms for this new scenario.

In this paper we shall examine troubleshooting problems where the following model parameters are given:

1. An overall problem D that we need to solve (e.g. "my printer does not work").

2. A set of faults $\mathcal{F} = \{f_1, \ldots, f_m\}$ describing the possible causes to the problem (e.g. "toner cartridge empty").

3. For each fault $f \in \mathcal{F}$, a probability $P(f)$ describing how likely it is that f is present when troubleshooting begins.

4. A set of actions $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$ that can potentially solve D (e.g. "change the toner cartridge").

5. For each action $\alpha \in \mathcal{A}$, a cost C_α describing the resources required to perform the action.

6. For each action $\alpha \in \mathcal{A}$, a success probability $P(\alpha = yes\,|\,f)$ describing the likelihood of solving D by performing the action when f is present.

7. A cost of testing if the problem D remains, $\mathrm{C}_{\mathrm{D}} > 0$.

Furthermore, we have the following simplifying assumptions about the model:

Assumption 1. *The single fault assumption: Initially the problem* D *is known to exist and it is due to the presence of a single fault from* $\mathcal{F}$*.*

Assumption 2. *The action result assumption: repeating a failed action will not fix the problem.*

Assumption 3. *The carefulness assumption: by performing an action or testing the system, we never introduce new faults.*

Assumption 4. *The independent actions assumption: Each action repairs a distinct set of faults that may be causing the problem* D*.*

We uphold these assumptions throughout this paper.

Remark. Assumption 1 implies

$$\sum_{\mathrm{f} \in \mathcal{F}} \mathrm{P}(\mathrm{f}) \quad = \quad 1$$

and by Assumption 4 the repair probability of an action can be computed as

$$\mathrm{P}(\alpha = yes) = \sum_{\mathrm{f} \in \mathcal{F}} \mathrm{P}(\alpha = yes \,|\, \mathrm{f}) \cdot \mathrm{P}(\mathrm{f}) \,.$$

However, since actions may be imperfect ($\mathrm{P}(\alpha = yes \,|\, \mathrm{f}) < 1$), Assumption 4 does not imply

$$\sum_{\alpha \in \mathcal{A}} \mathrm{P}(\alpha = yes) = 1 \,.$$

Furthermore, the above equation can also be violated by the fact that not all faults may be addressed by an action.

Before we can define our optimization problem formally, we need some notation and terminology. We write ε^i to denote that the first i actions have failed ($\varepsilon^0 \subseteq \varepsilon^i$), and we have by assumption $\mathrm{P}\big(\varepsilon^0\big) = 1$ because the device is faulty. We call $\mathrm{P}(\alpha = yes)$ for the *repair probability* of α, and we shall abbreviate it with P_α. To make it clear that the cost of an action includes the cost of testing the system C_{D}, we write $\mathrm{C}_{\mathrm{A+D}}$ for $\mathrm{C}_{\mathrm{A}} + \mathrm{C}_{\mathrm{D}}$ and so C_{A} never includes the cost of testing the system. For each $V \subseteq \mathcal{A}$ we can form a *compound action* A with the following properties

$$\mathrm{P}_{\mathrm{A}} = \sum_{\alpha \in V} \mathrm{P}_\alpha, \quad \mathrm{C}_{\mathrm{A}} = \sum_{\alpha \in V} \mathrm{C}_\alpha, \quad |\mathrm{A}| = |V| \,.$$

To make the distinction clear, we call actions $\alpha \in \mathcal{A}$ for *atomic actions*. We can see that a partition of $\mathcal{A}$ into k subsets $\{V_1, \ldots, \mathrm{V}_k\}$ induces a new model with independent actions $\{\mathrm{A}_1, \ldots, \mathrm{A}_k\}$. We shall also say that $\{\mathrm{A}_1, \ldots, \mathrm{A}_k\}$ is a partition of $\mathcal{A}$ whenever the corresponding sets of atomic actions form a partition of $\mathcal{A}$. From now on we shall not distinguish between the set of actions and the compound action; that is, we use A to mean a set of atomic actions and write $\mathrm{A} = \{\alpha_1, \ldots, \alpha_{|\mathrm{A}|}\}$. A *troubleshooting sequence* is a sequence of compound actions $\mathrm{s} = \langle \mathrm{A}_1, \ldots, \mathrm{A}_\ell \rangle$ prescribing the process of repeatedly performing the next action until the problem is fixed or the last action has been performed and such that $\{\mathrm{A}_1, \ldots, \mathrm{A}_\ell\}$ is a partition of $\mathcal{A}$. When each compound action contains only one action, we say that the sequence is *atomic*. A subsequence of s, $\mathrm{s}[k, m] = \langle \mathrm{A}_{k+1}, \ldots, \mathrm{A}_m \rangle$ with $k \leq m$, is called a *partial* troubleshooting sequence if $k \geq 1$ or $m < \ell$. If $k = 0$, we may simply write $\mathrm{s}[m]$.

Definition 1. Let $\mathrm{s} = \langle \mathrm{A}_1, \ldots, \mathrm{A}_\ell \rangle$ be a troubleshooting sequence. Then the *expected cost of repair* (ECR) of s is the mean of the cost until an action succeeds or all actions have been performed:

$$\mathrm{ECR}\,(\mathrm{s}) \quad = \quad \sum_{i=1}^{\ell} \mathrm{C}_{\mathrm{A}_i + \mathrm{D}} \cdot \mathrm{P}\big(\varepsilon^{i-1}\big) \ .$$

Formally, the problem is then to find a troubleshooting sequence with minimal $\mathrm{ECR}\,(\cdot)$ over all possible sequences. We call such a sequence for *optimal*.

An important definition is that of efficiency:

Definition 2. The *efficiency* of an action A is

$$\mathrm{ef}(\mathrm{A}) \quad = \quad \frac{\mathrm{P_A}}{\mathrm{C_{A+D}}} \ .$$

Its importance stems from the following results.

Theorem 1 (Kadane and Simon (1977))**.** *Let* $\mathrm{s} = \langle \alpha_1, \ldots, \alpha_\mathrm{n} \rangle$ *be an atomic troubleshooting sequences, and let* $\mathrm{C_D} = 0$*. Then* s *is optimal if and only if*

$$\mathrm{ef}(\alpha_\mathrm{i}) \geq \mathrm{ef}(\alpha_{\mathrm{i}+1}) \qquad \text{for } i \in \{1, \ldots, n-1\} \ .$$

Theorem 2 (Jensen et al. (2001))**.** *Let* $\mathrm{s} = \langle \alpha_1, \ldots, \alpha_\mathrm{n} \rangle$ *be an atomic troubleshooting sequence. Then a necessary condition for* s *to be optimal, even when* $\mathrm{C_D} > 0$*, is that*

$$\mathrm{ef}(\alpha_\mathrm{i}) \geq \mathrm{ef}(\alpha_{\mathrm{i}+1}) \qquad \text{for } i \in \{1, \ldots, n-1\} \ .$$

Corollary 1. *Theorem 2 holds for arbitrary troubleshooting sequences.*

Proof. Because a troubleshooting sequence partitions $\mathcal{A}$, the compound actions can be seen as virtual actions which are also independent, thus satisfying Assumption 3. All other assumptions cannot be invalidated. □

We may also compute the ECR of any atomic troubleshooting sequence:

Theorem 3 (Jensen et al. (2001)). *Let* $\mathrm{s} = \langle \alpha_1, \ldots, \alpha_\mathrm{n} \rangle$ *be an atomic troubleshooting sequence. Then the ECR of* s *may be computed as*

$$\mathrm{ECR}\left(\mathrm{s}\right) = \sum_{i=1}^{n} \mathrm{C}_{\alpha_\mathrm{i}+\mathrm{D}} \cdot \left(1 - \sum_{j=1}^{i-1} \mathrm{P}_{\alpha_\mathrm{j}}\right) ,$$

where $1 - \sum_{j=1}^{i-1} \mathrm{P}_{\alpha_\mathrm{j}} = \mathrm{P}\left(\varepsilon^{i-1}\right)$.

Corollary 2. *Theorem 3 holds for arbitrary troubleshooting sequences.*

Proof. Same argument as for Corollary 1. □

Thus, due to Assumption 1-4, we may completely ignore the $\mathcal{F}$, $\mathrm{P}(\mathrm{f})$, and $\mathrm{P}(\alpha = yes|\mathrm{f})$ once the repair probabilities have been computed. Therefore we only use P_α in the rest of this paper.

The following example shows that Theorem 1 does not extend to arbitrary troubleshooting sequences when $\mathrm{C}_\mathrm{D} > 0$.

Example 1. Consider a model with four atomic actions α_1, α_2, α_3 and α_4. The other properties of the model are as follows:

	P_α	C_α	$\mathrm{C}_{\alpha+\mathrm{D}}$	$\mathrm{ef}(\alpha)$
α_1	0.24	1	2	0.12
α_2	0.42	3	4	0.105
α_3	0.2	1	2	0.1
α_4	0.14	19	20	0.007

So $\mathrm{C}_\mathrm{D} = 1$. We have

$$\begin{aligned}
\mathrm{ECR}\left(\langle \alpha_1, \alpha_2, \alpha_3, \alpha_4 \rangle\right) &= 2 + 0.76 \cdot 4 + 0.34 \cdot 2 + 0.14 \cdot 20 = 8.52 \\
\mathrm{ECR}\left(\langle \{\alpha_1, \alpha_2\}, \alpha_3, \alpha_4 \rangle\right) &= 5 + 0.34 \cdot 2 + 0.14 \cdot 20 = 8.48
\end{aligned}$$

thus proving that it is more efficient to form the compound action $\mathrm{A} = \{\alpha_1, \alpha_2\}$.

To be able to test heuristic algorithms it is necessary with an algorithm that is guaranteed to find an optimal value. Exhaustive searches are always able to do this, albeit the algorithm might take exponential or super-exponential time. Algorithm 1 shows an exhaustive search with pruning (line 11, applying Theorem 2) and memoization (line 5-8).

Algorithm 1 Exhaustive Search With Compound Actions

```
function GenerateAllSequences($\mathcal{A}$, $\&\mathcal{S}$)
    Let $all = \emptyset$
    for $i = 1$ to $|\mathcal{A}|$ do
        for all $\binom{|\mathcal{A}|}{i}$ ways to construct the first compound action $\mathrm{A}$ do
            if $\mathrm{A} \notin \mathcal{S}$ then
                Compute $\mathrm{A}$ from scratch
                Set $\mathcal{S} = \mathcal{S} \cup \{\mathrm{A}\}$
            end if
            Let $after$ = GenerateAllSequences($\mathcal{A} \setminus \mathrm{A}, \mathcal{S}$)
            for $\mathrm{s} = \langle \mathrm{A}_1, \ldots, \mathrm{A}_\ell \rangle \in after$ do
                if $\mathrm{ef}(\mathrm{A}_1) \leq \mathrm{ef}(\mathrm{A})$ then
                    Let $\mathrm{s}' = \langle \mathrm{A}, \mathrm{A}_1, \ldots, \mathrm{A}_\ell \rangle$
                    Set $all = all \cup \{\mathrm{s}'\}$
                end if
            end for
        end for
    end for
    return $all$
end function
function OptimalCompoundActionSequence($\mathcal{A}$)
    Let $\mathcal{S} = \emptyset$
    Let $all$ = GenerateAllSequences($\mathcal{A}, \mathcal{S}$)
    return $\arg\min_{\mathrm{s} \in all} \mathrm{ECR}(\mathrm{s})$
end function
```

Let us briefly discuss the complexity of this algorithm. The algorithm recursively generates all possible sequences of compound actions. The number of such sequences is $n! \cdot 2^{n-1}$ because for each of the $n!$ permutations of the n atomic actions we can form 2^{n-1} different partitions. Thus the running time and memory requirement of the algorithm is $O(n! \cdot 2^n)$. Note, however, that this is not a tight upper bound since a troubleshooting sequence $\mathrm{s} = \langle \mathrm{A}_1, \ldots, \mathrm{A}_\ell \rangle$ can be constructed by $\prod_{i=1}^{\ell} |\mathrm{A}_i|!$ different orderings. Later we show that the complexity of an exhaustive search can be reduced to $\Theta(n^3 \cdot \mathrm{n}!)$.

Two Simple Greedy Heuristics

In this section we shall develop two simple greedy algorithms. Furthermore, we also give examples showing that none of the algorithms are guaranteed to find an optimal troubleshooting sequence.

For the first algorithm we consider how the ECR of a troubleshooting sequence can be improved by merging two neighbouring actions into one compound action. We have the following result.

Theorem 4. *Let the two troubleshooting sequences* s *and* s' *be given as*

$$\begin{aligned} \mathrm{s} &= \langle \ldots, \mathrm{A}_x, \mathrm{A}_{x+1}, \ldots \rangle \\ \mathrm{s}' &= \langle \ldots, \mathrm{A}, \ldots \rangle \end{aligned}$$

with $\mathrm{A} = \{\mathrm{A}_x, \mathrm{A}_{x+1}\}$ *and everything else being the same. Then* s' *has a strictly lower ECR than* s *if and only if*

$$\mathrm{C}_\mathrm{D} > \frac{\mathrm{C}_{\mathrm{A}_{x+1}} \cdot \mathrm{P}_{\alpha_\mathrm{x}}}{1 - \sum_{j=1}^{x} \mathrm{P}_{\alpha_\mathrm{j}}} .$$

Proof. When the sequence with the new compound action is better, we must have

$$\mathrm{ECR}\left(\mathrm{s}'\right) - \mathrm{ECR}\left(\mathrm{s}\right) \leq 0 .$$

We observe that most of the terms cancel each other out and we are left with

$$\mathrm{C}_{\mathrm{A+D}} \cdot \mathrm{P}\big(\boldsymbol{\varepsilon}^{x-1}\big) - \big[\mathrm{C}_{\mathrm{A}_x+\mathrm{D}} \cdot \mathrm{P}\big(\boldsymbol{\varepsilon}^{x-1}\big) + \mathrm{C}_{\mathrm{A}_{x+1}+\mathrm{D}} \cdot \mathrm{P}(\boldsymbol{\varepsilon}^{x})\big] < 0$$

$$\Updownarrow$$

$$\mathrm{C}_{\mathrm{A+D}} < \mathrm{C}_{\mathrm{A}_x} + \mathrm{C}_{\mathrm{D}} + (\mathrm{C}_{\mathrm{A}_{x+1}} + \mathrm{C}_{\mathrm{D}}) \cdot \frac{\mathrm{P}(\boldsymbol{\varepsilon}^{x})}{\mathrm{P}(\boldsymbol{\varepsilon}^{x-1})}$$

$$\Updownarrow$$

$$\mathrm{C}_{\mathrm{A}_{x+1}} \cdot \left(1 - \frac{\mathrm{P}(\boldsymbol{\varepsilon}^{x})}{\mathrm{P}(\boldsymbol{\varepsilon}^{x-1})}\right) < \mathrm{C}_{\mathrm{D}} \cdot \frac{\mathrm{P}(\boldsymbol{\varepsilon}^{x})}{\mathrm{P}(\boldsymbol{\varepsilon}^{x-1})}$$

$$\Updownarrow$$

$$\mathrm{C}_{\mathrm{D}} > \mathrm{C}_{\mathrm{A}_{x+1}} \cdot \left(\frac{\mathrm{P}\big(\boldsymbol{\varepsilon}^{x-1}\big)}{\mathrm{P}(\boldsymbol{\varepsilon}^{x})} - 1\right)$$

$$\Updownarrow$$

$$\mathrm{C}_{\mathrm{D}} > \mathrm{C}_{\mathrm{A}_{x+1}} \cdot \left(\frac{\mathrm{P}_{\alpha_{\mathrm{x}}}}{1 - \sum_{j=1}^{x} \mathrm{P}_{\alpha_{\mathrm{j}}}}\right)$$

□

Theorem 4 immediately suggests a procedure where we greedily merge adjacent actions until the ECR is no longer improved. We start with an atomic troubleshooting sequence and form compound actions by repeated application of Theorem 4. Justified by Theorem 1, there are two obvious choices of the initial sorting: with respect to descending efficiency or descending $\frac{\mathrm{P}}{\mathrm{C}}$-value. We have summarized this procedure in Algorithm 2. (The special function $1\mathrm{st}\arg(\cdot)$ returns the first index for which the boolean argument returns true.) As specified, the algorithm runs in $\Theta(n^2)$ time. However, if the sum in line 5 in `CompoundAction`$(\cdot)$ are precomputed , the total cost of all calls to `CompoundAction`$(\cdot)$ can be made linear. The running time of the algorithm is then dominated by the initial sorting of the actions leading to an $\Theta(n \cdot \lg n)$ worst case running time.
Is Algorithm 2 guaranteed to find an optimal troubleshooting sequence? The example below show it is not.

Example 2. We consider the following model:

	P_α	C_α	$\mathrm{C}_{\alpha+\mathrm{D}}$	$\mathrm{ef}(\alpha)$	$\frac{\mathrm{P}_\alpha}{\mathrm{C}_\alpha}$
α_1	0.61	1	11	0.055	0.61
α_2	0.21	5	15	0.014	0.042
α_3	0.18	3	13	0.013	0.06

So $\mathrm{C}_{\mathrm{D}} = 10$. Using the $\mathrm{ef}(\cdot)$ order Algorithm 2 first computes

$$10 < \frac{5 \cdot 0.61}{1 - 0.61} \approx 7.8\ .$$

Algorithm 2 Computing sequence with compound actions greedily

```
1: function CompoundAction($\langle \alpha_1, \ldots, \alpha_{\mathrm{n}} \rangle, x$)
2:     if $x = n$ then
3:         return $\{\alpha_{\mathrm{n}}\}$
4:     end if
5:     Let $m = \underset{i=x}{\overset{n}{\mathrm{1st\,arg}}} \left( \mathrm{C_D} \leq \frac{\mathrm{C}_{\alpha_{\mathrm{i+1}}} \cdot \mathrm{P}_{\alpha_{\mathrm{i}}}}{1 - \sum_{j=1}^{i} \mathrm{P}_{\alpha_{\mathrm{j}}}} \right)$
6:     return $\{\alpha_{\mathrm{x}}, \ldots, \alpha_{\mathrm{m}}\}$
7: end function
8: function GreedyCompoundActionSequence($\mathcal{A}$)
9:     Let $\mathrm{s}' = \langle \alpha_1, \ldots, \alpha_{\mathrm{n}} \rangle$ such that $\mathrm{ef}(\alpha_{\mathrm{i}}) \geq \mathrm{ef}(\alpha_{\mathrm{i+1}})$ or $\frac{\mathrm{P}_{\alpha_{\mathrm{i}}}}{\mathrm{C}_{\alpha_{\mathrm{i}}}} \geq \frac{\mathrm{P}_{\alpha_{\mathrm{i+1}}}}{\mathrm{C}_{\alpha_{\mathrm{i+1}}}}$.
10:    Let $\mathrm{s} = \langle\rangle$
11:    Let $i = 1$
12:    while $i \leq n$ do
13:        Let $\mathrm{A} = $ CompoundAction($\mathrm{s}', i$)
14:        Set $s = \mathrm{s} + \mathrm{A}$
15:        Set $i = i + |\mathrm{A}|$
16:    end while
17:    return $\mathrm{s}$
18: end function
```

Since this is false, it continues with the second test

$$10 < \frac{3 \cdot 0.21}{1 - 0.61 - 0.21} = 3.5$$

which means that the algorithm suggest the sequence consisting of a single compound action $\{\alpha_1, \alpha_2, \alpha_3\}$. If we repeat the calculations using the $\frac{\mathrm{P}}{\mathrm{C}}$-order, we get the same result. The ECR is easily seen to be 19, but

$$\mathrm{ECR}\left(\langle \alpha_1, \{\alpha_2, \alpha_3\}\rangle\right) = 11 + 0.39 \cdot 18 = 18.02$$

so the algorithm is not optimal.

Let us now consider another greedy heuristic. The algorithm is based on the idea that we should repeatedly form the most efficient compound action until all atomic actions have been assigned a compound action. To do this, we need a simple way of computing the most efficient compound action from a set of atomic actions. We first need a lemma.

Lemma 1. *Let a, b, c, and d be strictly positive numbers. Then*

$$\frac{a+b}{c+d} \geq \frac{a}{c} \quad \text{if and only if} \quad \frac{b}{d} \geq \frac{a}{c}$$

Proof.

$$\frac{a+b}{c+d} \geq \frac{a}{c} \Leftrightarrow a \cdot c + b \cdot c \geq a \cdot c + a \cdot d \Leftrightarrow \frac{b}{d} \geq \frac{a}{c}$$

□

Theorem 5. *Let $V \subseteq \mathcal{A}$ be given. Let A be a compound action consisting of actions in V with $\mathrm{ef}(\mathrm{A})$ maximal and with $|\mathrm{A}|$ minimal over those actions with maximal efficiency. Then A can be found by the following procedure: define $\mathrm{ef}(\mathrm{A}) = 0$ when $\mathrm{A} = \emptyset$; then repeatedly add an action $\alpha \in V$ to A with the highest $\frac{\mathrm{P}_\alpha}{\mathrm{C}_\alpha}$ value until $\mathrm{ef}(\mathrm{A})$ does not increase.*

Proof. Let A consist of actions in V such that $\mathrm{ef}(\mathrm{A})$ is maximized and $|\mathrm{A}|$ is minimal. Then $\mathrm{ef}(\mathrm{A})$ equals

$$\frac{\sum_{\alpha \in \mathrm{A}} \mathrm{P}_\alpha}{\mathrm{C_D} + \sum_{\alpha \in A} \mathrm{C}_\alpha} = \frac{\sum_{\alpha \in \mathrm{A} \setminus \{\beta\}} \mathrm{P}_\alpha + \mathrm{P}_\beta}{\mathrm{C_D} + \sum_{\alpha \in \mathrm{A} \setminus \{\beta\}} \mathrm{C}_\alpha + \mathrm{C}_\beta} = \frac{S_\mathrm{P} + \mathrm{P}_\beta}{S_\mathrm{C} + \mathrm{C}_\beta} = \frac{\mathrm{P}}{\mathrm{C}}$$

where β is chosen arbitrarily. Let furthermore $\alpha \in V \setminus \mathrm{A}$. We shall prove

$$\frac{\mathrm{P}_\beta}{\mathrm{C}_\beta} \geq \frac{\mathrm{P}}{\mathrm{C}} \geq \frac{\mathrm{P}_\alpha}{\mathrm{C}_\alpha}$$

which implies the theorem. We first prove

$$\frac{\mathrm{P}_\beta}{\mathrm{C}_\beta} \geq \frac{\mathrm{P}}{\mathrm{C}} \qquad (i).$$

Because $\mathrm{ef}(\mathrm{A})$ is maximal and A has minimal size, Lemma 1 gives

$$\frac{S_\mathrm{P} + \mathrm{P}_\beta}{S_\mathrm{C} + \mathrm{C}_\beta} > \frac{S_\mathrm{P}}{S_\mathrm{C}} \Leftrightarrow \frac{\mathrm{P}_\beta}{\mathrm{C}_\beta} > \frac{S_\mathrm{P}}{S_\mathrm{C}} \qquad (ii).$$

Then assume Equation (i) is false; then by Lemma 1

$$\frac{S_\mathrm{P} + \mathrm{P}_\beta}{S_\mathrm{C} + \mathrm{C}_\beta} \geq \frac{\mathrm{P}_\beta}{\mathrm{C}_\beta} \Leftrightarrow \frac{S_\mathrm{P}}{S_\mathrm{C}} \geq \frac{\mathrm{P}_\beta}{\mathrm{C}_\beta}$$

which is a contradiction with Equation (ii). We then consider the second inequality. Since $\mathrm{ef}(\mathrm{A})$ is maximized, Lemma 1 implies

$$\frac{\mathrm{P}}{\mathrm{C}} \geq \frac{\mathrm{P} + \mathrm{P}_\alpha}{\mathrm{C} + \mathrm{C}_\alpha} \Leftrightarrow \frac{\mathrm{P}}{\mathrm{C}} \geq \frac{\mathrm{P}_\alpha}{\mathrm{C}_\alpha}$$

which completes the proof. □

Remark. In (Langseth and Jensen, 2001) a similar heuristic procedure is reached albeit for a slightly different problem.

Algorithm 3 Greedy algorithm with max-efficient actions

```
1: function GREEDYCOMPOUNDACTIONSEQUENCE2($\mathcal{A}$)
2:     Let $s' = \langle \alpha_1, \ldots, \alpha_n \rangle$ such that $\frac{P_{\alpha_i}}{C_{\alpha_i}} \geq \frac{P_{\alpha_{i+1}}}{C_{\alpha_{i+1}}}$
3:     Let $s = \langle \rangle$
4:     Repeatedly add the most efficient compound action to $s$
5:     return $s$
6: end function
```

Algorithm 3 shows a greedy algorithm based on Theorem 5. Like Algorithm 2 we can precompute the sums used in the efficiency and get a running time of $\Theta(n \cdot \lg n)$.

The following example shows that Algorithm 3 is not guaranteed to find an optimal troubleshooting sequence.

Example 3. Consider the following model

	P_α	C_α	$C_{\alpha+D}$	$\frac{P_\alpha}{C_\alpha}$
α_1	0.15	1	2	0.150
α_2	0.35	2	3	0.175
α_3	0.50	3	4	0.167

where the ordering considered is $\langle \alpha_2, \alpha_3, \alpha_1 \rangle$. Algorithm 3 computes the following values to determine the first compound action:

$$\begin{aligned} \text{ef}(\{\alpha_2\}) &= \frac{0.35}{3} = 0.117 \\ \text{ef}(\{\alpha_2, \alpha_3\}) &= \frac{0.35 + 0.5}{6} = 0.1416 \\ \text{ef}(\{\alpha_2, \alpha_3, \alpha_1\}) &= \frac{1}{7} = 0.1429 \end{aligned}$$

Thus the suggested sequence is one single compound action with ECR 7. However,

$$\text{ECR}\left(\langle \{\alpha_2, \alpha_3\}, \alpha_1 \rangle\right) = 6 + 0.15 \cdot 2 = 6.3$$

which shows the algorithm is not optimal.

Having established the heuristic nature of both greedy algorithms, we would like to know if they at least find a local optimum. We use the following definition to define a local optimum.

Definition 3. Let $s = \langle \alpha_1, \ldots, \alpha_n \rangle$ be a sequence of atomic actions, and let $s^c = \langle B_1, \ldots, B_m \rangle$ be a compound action sequence. Then s^c is said to be **consistent** with s if $\forall\, k < l$

$$\alpha_i \in B_k \text{ and } \alpha_j \in B_l \implies i < j.$$

In other words, $\mathrm{s^c}$ can be seen as an ordered partition of s.

If an atomic action sequence s is given, we call the set of all compound action sequences consistent with s for $\mathcal{C}(\mathrm{s})$. A local optimum is then a compound action sequence $\mathrm{s^c} \in \mathcal{C}(\mathrm{s})$ with minimal ECR. We then say the (ordered) partition induced by $\mathrm{s^c}$ is *optimal*.

From Example 2 and 3 it follows that neither Algorithm 2 nor Algorithm 3 guarantees an optimal partition.

An Algorithm For Optimal Partitioning

We have now seen how the greedy algorithms fail to provide optimal sequences even under the restriction that the solution must be consistent with a single seed-sequence s of atomic actions. However, since there are 2^{n-1} ways to partition the actions of s, a brute force approach is quite impractical even for models of moderate size. By exploiting Theorem 6 below we can construct an $O(n^3)$ dynamic programming algorithm.

Lemma 2. *Let $k \geq 0$ and let $\mathrm{s^c} = \langle \mathrm{A}_1, \ldots, \mathrm{A}_\ell \rangle$ be a partial troubleshooting sequence that is performed after the first k actions have been performed. Then the contribution of $\mathrm{s^c}$ to the total ECR is given by*

$$\mathrm{ECR_k}(\mathrm{s^c}) \quad = \quad \sum_{i=1}^{\ell} \mathrm{P}\Big(\boldsymbol{\varepsilon}^{k+i-1}\Big) \cdot \mathrm{C}_{\mathrm{A}_i+\mathrm{D}}$$

where

$$\mathrm{P}\Big(\boldsymbol{\varepsilon}^{k+i-1}\Big) \quad = \quad 1 - \sum_{j=1}^{k} \mathrm{P}_{\alpha_\mathrm{j}} - \sum_{j=1}^{i-1} \mathrm{P}_{\mathrm{A}_j} \ .$$

Proof. Follows directly from Corollary 2. □

The Lemma ensures that the following definition is sound.

Definition 4. Let $\mathrm{s} = \langle \alpha_1, \ldots, \alpha_\mathrm{n} \rangle$ be an atomic troubleshooting sequence. For any partial sequence $\mathrm{s}[k,m]$ with $k \leq m$ the ECR of an optimal compound action sequence consistent with $\mathrm{s}[k,m]$ is defined by

$$\mathrm{ECR}^*(\mathrm{s}[k,m]) = \begin{cases} \displaystyle\min_{\mathrm{s^c} \in \mathcal{C}(\mathrm{s[k,m]})} \mathrm{ECR_k}(\mathrm{s^c}) & \text{if } k < m \\ 0 & \text{if } k = m \end{cases} .$$

Theorem 6. *Let $\mathrm{s} = \langle \alpha_1, \ldots, \alpha_\mathrm{n} \rangle$ be an atomic troubleshooting sequence. Then for $k < m$, $\mathrm{ECR}^*(\mathrm{s}[k,m])$ can be calculate as*

$$\mathrm{ECR}^*(\mathrm{s}[k,m]) \quad = \quad \min_{i=k+1}^{m} \Big(\mathrm{ECR_k}(\langle \{\alpha_\mathrm{k+1}, \ldots, \alpha_\mathrm{i}\} \rangle) + \mathrm{ECR}^*(\mathrm{s}[i,m]) \Big)$$

Proof. We use induction in $m - k$. Basis step: $m - k = 1$. Since there is only one compound sequence consistent with $\langle \alpha_{\mathrm{m}} \rangle$, we get

$$\mathrm{ECR}^*(\mathrm{s}[k, m]) = \min(\mathrm{ECR}_{\mathrm{k}}(\langle\{\alpha_{\mathrm{m}}\}\rangle) + 0) = \mathrm{ECR}_{\mathrm{k}}(\langle\{\alpha_{\mathrm{m}}\}\rangle)$$

so the theorem holds in this case. Induction step: assume the theorem holds for $m - k \leq x$ and let $m - k = x + 1$. Then let

$$\mathrm{s}^* = \underset{\mathrm{s^c} \in \mathcal{C}(\mathrm{s[k,m]})}{\arg\min}\ \mathrm{ECR}_{\mathrm{k}}(\mathrm{s^c})$$

for which we must prove

$$\mathrm{ECR}^*(\mathrm{s}^*) \quad = \quad \min_{i=k+1}^{m} \left(\mathrm{ECR}_{\mathrm{k}}(\langle\{\alpha_{\mathrm{k+1}}, \ldots, \alpha_{\mathrm{i}}\}\rangle) + \mathrm{ECR}^*(\mathrm{s}[i, m]) \right) .$$

By Lemma 2 it is correct to split the ECR into two terms corresponding to an ordered partition of s^*. Furthermore, by the induction hypothesis we can compute $\mathrm{ECR}^*(\mathrm{s}[i, m])$ for any i since $m - i \leq x$. Then consider that s^* must start with a compound action consisting of $y \in \{1, 2, \ldots, x + 1\}$ actions. These choices of the first action correspond exactly to the expressions that we minimize over, and the result follows. □

Theorem 6 shows that the problem of finding an optimal partition of a seed sequence s requires optimal partitions of smaller partial sequences (the optimal-substructure property) and that these smaller partial sequences are needed by the computation of several bigger sequences (the overlapping subproblems property). These two properties enables a *dynamic programming algorithm*, that is, an algorithm that only computes solutions to subproblems once by storing the results in a table. Algorithm 4 uses this approach to find the optimal partition of a seed-sequence s. The algorithm runs in $\Theta(n^3)$ time and uses $\Theta(n^2)$ space. The correctness of the algorithm follows from the above theorem. Algorithm 4 now also means that we can make a more efficient exhaustive search for the best troubleshooting sequence. The algorithm runs in $\Theta(n^3 \cdot n!)$ time and is shown in Algorithm 6.

Unfortunately, Algorithm 5 is not guaranteed to find an optimal troubleshooting sequence, as the following example shows.

Example 4. Consider the model from Example 3, but take $\mathrm{C_D} = 2$:

	P_α	C_α	$\mathrm{C}_{\alpha+\mathrm{D}}$	$\mathrm{ef}(\alpha)$	$\frac{\mathrm{P}_\alpha}{\mathrm{C}_\alpha}$
α_1	0.15	1	3	0.05	0.150
α_2	0.35	2	4	0.0875	0.175
α_3	0.50	3	5	0.1	0.167

Algorithm 4 Optimal Partition into Compound Actions

```
 1: procedure FILLCELL($row, col, \mathrm{s}, \mathrm{C_D}, \&ecrMatrix, \&prevMatrix$)
Require: $row > col$
 2:     Let $N = row - col$
 3:     Let $\mathrm{A} = \langle \alpha_{\mathrm{col}}, \dots, \alpha_{\mathrm{col+N}} \rangle$ from $\mathrm{s}$
 4:     Let $ecr = \mathrm{P}\big(\varepsilon^{col-1}\big) \cdot \mathrm{C_A}$
 5:     Set $ecrMatrix[row, col] = ecr$
 6:     Set $prevMatrix[row, col] = col$
 7:     for $prev = col + 1$ to $row - 1$ do
 8:         Let $ecr' = ecrMatrix[prev, col] + ecrMatrix[row, prev]$
 9:         if $ecr' < ecr$ then
10:             Set $ecrMatrix[row, col] = ecr'$
11:             Set $prevMatrix[row, col] = prevMatrix[row, prev]$
12:         end if
13:     end for
14: end procedure
15: function OPTIMALPARTITION($\mathrm{s} = \langle \alpha_1, \dots, \alpha_{\mathrm{n}} \rangle$, $\mathrm{C_D}$)
16:     Let $N = |\mathrm{s}|$
17:     Let $ecrMatrix = \{0\}$
18:     Let $prevMatrix = \{0\}$
19:     for $row = 1$ to $N$ do
20:         for $col = row - 1$ down to 1 do
21:             FILLCELL($row, col, \mathrm{s}, \mathrm{C_D}, ecrMatrix, prevMatrix$)
22:         end for
23:     end for
24:     Let $\mathrm{s}^* = \langle\rangle$
25:     Let $prev = N$
26:     repeat
27:         Let $i = prevMatrix[prev, 1]$
28:         Let $\mathrm{A} = \langle \alpha_{\mathrm{i+1}}, \dots, \alpha_{\mathrm{prev}} \rangle$
29:         Set $\mathrm{s}^* = \langle \mathrm{A}, \mathrm{s}^* \rangle$
30:     until $prev < 1$
31:     ASSERT(ECR$(\mathrm{s}^*) = ecrMatrix[N, 1]$ )
32:     return $\mathrm{s}^*$
33: end function
```

Algorithm 5 Compound Action Sequence with Optimal Partition

```
1: function COMPOUNDACTIONSEQUENCE($\mathcal{A}$)
2:     Let $\mathrm{s} = \langle \alpha_1, \dots, \alpha_{\mathrm{n}} \rangle$ such that $\mathrm{ef}(\alpha_{\mathrm{i}}) \geq \mathrm{ef}(\alpha_{\mathrm{i+1}})$ or $\frac{\mathrm{P}_{\alpha_{\mathrm{i}}}}{\mathrm{C}_{\alpha_{\mathrm{i}}}} \geq \frac{\mathrm{P}_{\alpha_{\mathrm{i+1}}}}{\mathrm{C}_{\alpha_{\mathrm{i+1}}}}$
3:     Let s = OPTIMALPARTITION(s)
4:     return s
5: end function
```

Algorithm 6 Optimized Exhaustive Search With Compound Actions

```
function OptimalCompoundActionSequence2($\mathcal{A}$)
    Let $bestEcr = \infty$
    Let $bestSeq = \langle\rangle$
    for all $n!$ permutations of $\mathcal{A}$ s do
        Let $s' =$ OptimalPartitioning(s)
        if $\text{ECR}(s') < bestEcr$ then
            Set $bestEcr = \text{ECR}(s')$
            Set $bestSeq = s'$
        end if
    end for
    return $s'$
end function
```

For the $\text{ef}(\cdot)$ ordering, the algorithm constructs the following tables—here visualized as one table with a pair of values $(\text{ECR}^*(\text{s}[k,m]), i)$ where $i-1$ is the split that minimized the (partial) ECR:

$$\begin{pmatrix} (0,0) & (0,0) & (0,0) & (0,0) \\ (5,0) & (0,0) & (0,0) & (0,0) \\ (7,0) & (2,1) & (0,0) & (0,0) \\ (7.45,2) & (2.45,2) & (0.45,2) & (0,0) \end{pmatrix}$$

Looking at the bottom left entry, we get that the last action is $\{\alpha_1\}$ and then looking at the entry above it, that the first action is $\{\alpha_3, \alpha_2\}$ and $\langle\{\alpha_3, \alpha_2\}, \alpha_1\rangle$ has ECR 7.45. Even though the initial $\frac{\text{P}}{\text{C}}$ ordering is different, the algorithm returns the same troubleshooting sequence. However,

$$\text{ECR}\left(\langle\{\alpha_1, \alpha_3\}, \alpha_2\rangle\right) = 6 + 0.35 \cdot 4 = 7.4$$

which shows that the algorithm does not lead to an optimal sequence.

A Heuristic for The General Problem

We have seen that by changing the ordering of the atomic actions, we could improve the ECR compared to keeping the ordering fixed. We shall therefore consider the orthogonal task of optimizing the ordering given that the partitioning is fixed.

Definition 5. Let $\text{s}^c = \langle \text{A}_1, \ldots, \text{A}_\ell \rangle$ and $\text{s}^d = \langle \text{B}_1, \ldots, \text{B}_\ell \rangle$ be two compound action sequences of the same size. Then s^c and s^d are *partition equivalent* if

$$|\text{A}_i| = |\text{B}_i| \;\; \forall i \in \{1, 2, \ldots, \ell\}.$$

Lemma 3. *Let* $\mathrm{s^c} = \langle \dots, \mathrm{A}_x, \dots, \mathrm{A}_y, \dots \rangle$ *and* $\mathrm{s^d} = \langle \dots, \mathrm{B}_x, \dots, \mathrm{B}_y, \dots \rangle$ *be two partition equivalent compound action sequences. Furthermore, let the only difference between* $\mathrm{s^c}$ *and* $\mathrm{s^d}$ *be that two actions* $\alpha \in \mathrm{A}_x$ *and* $\beta \in \mathrm{A}_y$ *have been swapped s.t.* $\beta \in \mathrm{B}_x$ *and* $\alpha \in \mathrm{B}_y$*. Then*

$$\begin{aligned}\mathrm{ECR}\,(\mathrm{s^c}) - \mathrm{ECR}(\mathrm{s^d}) \quad &= \quad (\mathrm{C}_\alpha - \mathrm{C}_\beta) \cdot \left[\mathrm{P}(\beta) - \mathrm{P}(\alpha) + \sum_{i=x}^{y-1} \mathrm{P}(\mathrm{A}_i) \right] \\ &\quad + [\mathrm{P}(\beta) - \mathrm{P}(\alpha)] \cdot \sum_{i=x+1}^{y} \mathrm{C}_{\mathrm{A}_i + \mathrm{D}} \, .\end{aligned}$$

Proof. Let $\mathrm{s^c}$ and $\mathrm{s^d}$ be given as above. We then have

$$\mathrm{ECR}(\mathrm{s^c}) = \sum_{i=1}^{\ell} \mathrm{C}_{\mathrm{A}_i+\mathrm{D}} \cdot (1 - \sum_{j=1}^{i-1} \mathrm{P}(\mathrm{A}_j)) = \sum_{i=1}^{\ell} \mathrm{C}_{\mathrm{A}_i} \cdot a_i$$
$$\mathrm{ECR}(\mathrm{s^d}) = \sum_{i=1}^{\ell} \mathrm{C}_{\mathrm{B}_i+\mathrm{D}} \cdot (1 - \sum_{j=1}^{i-1} \mathrm{P}(\mathrm{B}_j)) = \sum_{i=1}^{\ell} \mathrm{C}_{\mathrm{B}_i} \cdot b_i \, .$$

The difference between these two numbers can be expressed as

$$\mathrm{ECR}(\mathrm{s^c}) - \mathrm{ECR}(\mathrm{s^d}) \quad = \quad \Delta X + \Delta I + \Delta Y \qquad (i)$$

where

- ΔX is the difference caused by changes to $\mathrm{C}_{\mathrm{A}_x}$,
- ΔI is the difference caused by a_i changing into b_i for all compound actions between A_x and A_y, and
- ΔY is the difference causes by changes to $\mathrm{C}_{\mathrm{A}_y}$ and a_y.

For $i \in \{1, \dots, x-1, y+1, \dots, \ell\}$ the terms cancel out. We have

$$\Delta X = a_x \cdot (\mathrm{C}_{\mathrm{A}_x+\mathrm{D}} - \mathrm{C}_{\mathrm{B}_x+\mathrm{D}}) = a_x \cdot (\mathrm{C}_\alpha - \mathrm{C}_\beta) \, .$$

Furthermore, since $b_i = a_i + \mathrm{P}(\alpha) - \mathrm{P}(\beta) \, \forall \, i \in \{x+1, \dots, y\}$:

$$\Delta I = \sum_{i=x+1}^{y-1} (a_i - b_i) \cdot \mathrm{C}_{\mathrm{A}_i+\mathrm{D}} = [\mathrm{P}(\beta) - \mathrm{P}(\alpha)] \cdot \sum_{i=x+1}^{y-1} \mathrm{C}_{\mathrm{A}_i+\mathrm{D}}$$

Finally we have

$$\begin{aligned}\Delta Y \quad &= \quad a_y \cdot \mathrm{C}_{\mathrm{A}_y+\mathrm{D}} - b_y \cdot \mathrm{C}_{\mathrm{B}_y+\mathrm{D}} \\ &= \quad a_y \cdot \mathrm{C}_{\mathrm{A}_y+\mathrm{D}} - b_y \cdot (\mathrm{C}_{\mathrm{A}_y+\mathrm{D}} - \mathrm{C}_\beta + \mathrm{C}_\alpha) \\ &= \quad (a_y - b_y) \cdot \mathrm{C}_{\mathrm{A}_y+\mathrm{D}} - b_y \cdot (\mathrm{C}_\alpha - \mathrm{C}_\beta) \\ &= \quad [\mathrm{P}(\beta) - \mathrm{P}(\alpha)] \cdot \mathrm{C}_{\mathrm{A}_y+\mathrm{D}} - b_y \cdot (\mathrm{C}_\alpha - \mathrm{C}_\beta) \, .\end{aligned}$$

Now observe that

$$\begin{aligned} b_y &= a_y - [\mathrm{P}(\beta) - \mathrm{P}(\alpha)] \\ &= \left(a_x - \sum_{i=x}^{y-1} \mathrm{P}(\mathrm{A}_i) \right) - [\mathrm{P}(\beta) - \mathrm{P}(\alpha)] \end{aligned}$$

which implies that

$$\begin{aligned} \Delta Y &= [\mathrm{P}(\beta) - \mathrm{P}(\alpha)] \cdot \mathrm{C}_{\mathrm{A}_y + \mathrm{D}} \\ &\quad - \left[a_x - \mathrm{P}(\beta) + \mathrm{P}(\alpha) - \sum_{i=x}^{y-1} \mathrm{P}(\mathrm{A}_i) \right] \cdot (\mathrm{C}_\alpha - \mathrm{C}_\beta) \, . \end{aligned}$$

Inserting these three results into Equation (i) yields

$$\begin{aligned} \mathrm{ECR}(\mathrm{s^c}) - \mathrm{ECR}(\mathrm{s^d}) &= (\mathrm{C}_\alpha - \mathrm{C}_\beta) \cdot \left[\mathrm{P}(\beta) - \mathrm{P}(\alpha) + \sum_{i=x}^{y-1} \mathrm{P}(\mathrm{A}_i) \right] \\ &\quad + [\mathrm{P}(\beta) - \mathrm{P}(\alpha)] \cdot \sum_{i=x+1}^{y} \mathrm{C}_{\mathrm{A}_i + \mathrm{D}} \end{aligned}$$

as required. □

Theorem 7. *Let* $\mathrm{s^c}$*,* α *and* β *be given as in Lemma 3. Then the ECR of* $\mathrm{s^c}$ *can be improved by swapping* α *and* β *if and only if*

$$(\mathrm{C}_\alpha - \mathrm{C}_\beta) \cdot \left[\mathrm{P}(\beta) - \mathrm{P}(\alpha) + \sum_{i=x}^{y-1} \mathrm{P}(\mathrm{A}_i) \right] > [\mathrm{P}(\alpha) - \mathrm{P}(\beta)] \cdot \sum_{i=x+1}^{y} \mathrm{C}_{\mathrm{A}_i + \mathrm{D}}$$

Proof. We should swap the two actions if

$$\begin{gathered} \mathrm{ECR}(\mathrm{s^c}) - \mathrm{ECR}(\mathrm{s^d}) > 0 \\ \Updownarrow \\ (\mathrm{C}_\alpha - \mathrm{C}_\beta) \cdot \left[\mathrm{P}(\beta) - \mathrm{P}(\alpha) + \sum_{i=x}^{y-1} \mathrm{P}(\mathrm{A}_i) \right] + [\mathrm{P}(\beta) - \mathrm{P}(\alpha)] \cdot \sum_{i=x+1}^{y} \mathrm{C}_{\mathrm{A}_i + \mathrm{D}} > 0 \\ \Updownarrow \\ (\mathrm{C}_\alpha - \mathrm{C}_\beta) \cdot \left[\mathrm{P}(\beta) - \mathrm{P}(\alpha) + \sum_{i=x}^{y-1} \mathrm{P}(\mathrm{A}_i) \right] > [\mathrm{P}(\alpha) - \mathrm{P}(\beta)] \cdot \sum_{i=x+1}^{y} \mathrm{C}_{\mathrm{A}_i + \mathrm{D}} \, . \end{gathered}$$

□

It turns out that in certain cases we can avoid the full analysis of Theorem 7.

Proposition 1. *Let* $\mathrm{s^c}$, α *and* β *be given as in Lemma 3. If*

$$\begin{array}{lll} \mathrm{P}(\beta) > \mathrm{P}(\alpha) & \text{and} & \mathrm{C}_\alpha \geq \mathrm{C}_\beta, \ \text{or} \\ \mathrm{P}(\beta) = \mathrm{P}(\alpha) & \text{and} & \mathrm{C}_\alpha > \mathrm{C}_\beta \end{array}$$

then $\mathrm{ECR}(\mathrm{s^c})$ *can be improved by swapping* α *and* β.

Proof. If $\mathrm{P}(\beta) > \mathrm{P}(\alpha)$ and $\mathrm{C}_\alpha \geq \mathrm{C}_\beta$, then $\mathrm{P}(\beta) - \mathrm{P}(\alpha) < 0$, but $\mathrm{C}_\alpha - \mathrm{C}_\beta \geq 0$. In Theorem 7 the two other factors are strictly positive, so the left hand side becomes non-negative while the right hand side becomes negative making the inequality true. If $\mathrm{P}(\beta) = \mathrm{P}(\alpha)$ and $\mathrm{C}_\alpha > \mathrm{C}_\beta$, then the left hand side is strictly positive while the right hand side is zero which also makes the inequality true. □

Based on the above results, we can easily search for a better partition equivalent troubleshooting sequence. The procedure is given in Algorithm 7. Arguably, after the algorithm has terminated, it could be that more actions could be swapped to improve the ECR. However, we have been unsuccessful in proving the number of possible swaps. Therefore we prefer an algorithm that terminates deterministically. Furthermore, we tried to run the procedure until no more swaps could be made on our test models, and we found that it did not improve the ECR compared to Algorithm 7.

Algorithm 7

```
1:  procedure OptimizePartition(& $\mathrm{s^c}=\langle \mathrm{A}_1, \ldots, \mathrm{A}_n \rangle$)
2:      for $x = 1$ to $n$ do
3:          for $\alpha \in \mathrm{A}_x$ do
4:              for $y = x + 1$ to n do
5:                  for $\beta \in \mathrm{A}_y$ do
6:                      if $(\mathrm{C}_\alpha - \mathrm{C}_\beta) \cdot \left[\mathrm{P}(\beta) - \mathrm{P}(\alpha) + \sum_{i=x}^{y-1} \mathrm{P}(\mathrm{A}_i)\right] >$
7:                          $[\mathrm{P}(\alpha) - \mathrm{P}(\beta)] \cdot \sum_{i=x+1}^{y} \mathrm{C}_{\mathrm{A}_i + \mathrm{D}}$
8:                      then
9:                          Swap($\alpha, \beta$)
10:                     end if
11:                 end for
12:             end for
13:         end for
14:     end for
15: end procedure
```

It should be clear that the algorithm runs in $\Theta(n^3)$ where n is the number of atomic actions. The reason it cannot run in $\Theta(n^2)$ time is that checking Theorem 7 takes linear time on average, albeit when Proposition 1 applies it only takes constant time. Because any swapping invalidates the involved sums, it is not possible to precompute these as with the greedy approaches we saw earlier. We can also formulate the improved heuristic described in Algorithm 8.

Algorithm 8 Compound Action Sequence with Optimized Partition

```
1: function COMPOUNDACTIONSEQUENCE($\mathcal{A}$)
2:     Let $s = \langle \alpha_1, \ldots, \alpha_n \rangle$ such that $\text{ef}(\alpha_i) \geq \text{ef}(\alpha_{i+1})$ or $\frac{P_{\alpha_i}}{C_{\alpha_i}} \geq \frac{P_{\alpha_{i+1}}}{C_{\alpha_{i+1}}}$
3:     Let s = OPTIMALPARTITION(s)
4:     OPTIMIZEPARTITION(s)
5:     return s
6: end function
```

We have by means of a computer constructed examples that show that the algorithm is not guaranteed to find optimal troubleshooting sequences. However, for certain models we can prove that optimality is ensured. Consider a model where the actions can be ordered into the sequence $s = \langle \alpha_1, \ldots, \alpha_n \rangle$ such that $P_{\alpha_i} \geq P_{\alpha_{i+1}}$ and $C_{\alpha_i} \leq C_{\alpha_{i+1}}$ for $i \in \{1, 2, \ldots, n-1\}$; we call such models for *non-reordable* models.

Theorem 8. *Let* $s = \langle \alpha_1, \ldots, \alpha_n \rangle$ *be an atomic troubleshooting sequence in a non-reordable model such that* $\text{ef}(\alpha_i) \geq \text{ef}(\alpha_{i+1})$. *Then there exists an optimal troubleshooting sequence* s^c *consistent with* s.

Proof. Let $s^d = \langle B_1, \ldots, B_\ell \rangle$ and $s^c = \langle A_1, \ldots, A_\ell \rangle$ be partition equivalent compound troubleshooting sequences where s^d is optimal and s^c is consistent with s. Now assume s^d is not consistent with s. Since $s^c \neq s^d$, we can find the index of the first pair of compound actions A_x and B_x where the two sequences differ. Then at least one action $\beta \in A_x$, but $\beta \notin B_x$ and at least one action $\alpha \in B_x$, but $\alpha \notin A_x$. Furthermore, α must be found in A_y with $y > x$. Since s^c is consistent with s and since the model is non-reordable

$$P_\beta \geq P_\alpha \text{ and } C_\beta \leq C_\alpha \ (i).$$

We now consider two cases.

Case 1: at least one of the inequalities in (i) is strict. Then if we apply Theorem 7 on s^d we find that we can improve the ECR of s^d by swapping α and β. However, this is a contradiction to s^d being optimal. Hence in this case any optimal sequence must be consistent with s.

Case 2: we have equality in Equation (i). In that case we can safely swap α and β in s^d without altering the ECR. If we now have $s^d = s^c$, then we are done. If not, we can repeat the procedure until $s^d = s^c$. Since each swap puts at least one atomic action into its rightful compound action, the procedure terminates in a finite number of steps, thus proving that there exists an optimal sequence consistent with s. □

Corollary 3. *Algorithm 5 and 8 find an optimal troubleshooting sequence in non-reorderable models.*

Proof. Both algorithms calls `OptimalPartition`$(\cdot)$ after having sorted the actions with respect to descending efficiency. $\square$

Results

In this section we shall investigate the performance of the heuristic algorithms. In particular, we investigate how the following two model parameters affect the precision:

1. The closeness of the initial efficiency of actions.
2. The closeness of the cost of the actions.

Due to the difficulty of finding an optimal solution, we could not investigate models with more than 8 actions. In the models below the repair probabilities have not been normalized to make the specification accurate (but they are of course normalized at runtime) and the efficiency stated is the efficiency rounded to three decimals when $C_D = 0$.

Model 1: close efficiencies + close costs								
	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
P_α	0.21	0.17	0.19	0.155	0.185	0.139	0.17	0.135
C_α	1.8	1.4	1.7	1.3	1.6	1.2	1.5	1.1
$\text{ef}(\alpha)$	0.086	0.090	0.083	0.088	0.085	0.085	0.084	0.091

Model 2: close efficiencies + non-close costs								
	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
P_α	0.36	0.205	0.32	0.155	0.28	0.11	0.25	0.05
C_α	8	4	7	3	6	2	5	1
$\text{ef}(\alpha)$	0.026	0.030	0.026	0.030	0.027	0.031	0.029	0.029

Model 3: non-close efficiencies + close costs								
	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
P_α	0.4	0.105	0.52	0.055	0.78	0.13	0.6	0.02
C_α	1.8	1.4	1.7	1.3	1.6	1.2	1.5	1.1
$\text{ef}(\alpha)$	0.085	0.029	0.117	0.016	0.187	0.042	0.153	0.007

Model 4: non-close efficiencies + non-close costs								
	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
P_α	0.16	0.15	0.05	0.165	0.165	0.159	0.07	0.165
C_α	8	4	7	3	6	2	5	1
$\text{ef}(\alpha)$	0.018	0.035	0.007	0.051	0.025	0.073	0.013	0.152

For each of the four initial models we run all the algorithms for increasing values of C_D. The following observation shows when it does not make sense to increase C_D anymore.

Proposition 2. *Once* C_D *is made so high that the optimal sequence consists of one compound action, then increasing* C_D *can never change the optimal sequence.*

The increment in C_D was determined as one permille of the largest cost of any action in the model. Starting from $C_D = 0$ we kept sampling until the optimal sequence consists of only one compound action. The tables below summarize the result for the four models where each column describes the results of a given algorithm. The first four rows summarize the percentwise deviation from the optimal ECR, and the last row shows the percent of cases where the algorithm was optimal. The number of increments to C_D is given in the parenthesis after the model name. The first algorithm column is the simple $\mathrm{ef}(\alpha_i) \geq \mathrm{ef}(\alpha_{i+1})$ sorting. For Algorithm 2, 5, and 8 we give two numbers. The first corresponds to an initial $\mathrm{ef}(\cdot)$ ordering and the second corresponds to an initial $\frac{P}{C}$ ordering.

Model 1: close efficiencies + close costs (5828)					
	$\mathrm{ef}(\alpha_i)$ ord.	Alg. 3	Alg. 2	Alg. 5	Alg. 8
min	0	0	0/0	0/0	0/0
max	128.26	45.56	4.28/2.83	1.47/0.68	1.08/0.63
mean	73.14	10.05	1.66/0.66	0.77/0.07	0.15/0.02
median	79.37	5.97	1.52/0.37	0.79/0	0.11/0
% opt.	1.48	0.05	1.49/26.56	1.49/62.06	39.88/84.80

Model 2: close efficiencies + non-close costs (4201)					
	$\mathrm{ef}(\alpha_i)$ ord.	Alg. 3	Alg. 2	Alg. 5	Alg. 8
min	0	0	0/0	0 /0	0/0
max	97.16	38.90	6.17/2.34	5.05/0.48	4.17/0.48
mean	54.71	9.01	2.75/0.33	1.87/0.06	1.09/0.03
median	58.87	5.71	3.15/0.07	1.77/0.01	1.17/0
% opt.	0.33	0.05	8.40/32.85	9.50/46.04	18.64/63.91

Model 3: non-close efficiencies + close costs (79145)					
	$\mathrm{ef}(\alpha_i)$ ord.	Alg. 3	Alg. 2	Alg. 5	Alg. 8
min	0	0	0/0	0/0	0/0
max	149.50	3.93	2.80/2.80	0/0	0/0
mean	124.27	0.08	0.16/0.16	0/0	0/0
median	137.14	0	0/0	0/0	0/0
% opt.	0.45	76.27	73.89/73.89	100/100	100/100

Model 4: non-close efficiencies + non-close costs (18085)					
	$\mathrm{ef}(\alpha_i)$ ord.	Alg. 3	Alg. 2	Alg. 5	Alg. 8
min	0	0	0/0	0/0	0/0
max	219.13	4.87	2.62/2.62	0/0	0/0
mean	162.80	0.35	0.24/0.24	0/0	0/0
median	180.95	0	0/0	0/0	0/0
% opt.	0.25	53.08	76.08/76.08	100/100	100/100

We can summarize the results as follows:

1. Algorithm 8 is by far the best heuristic. Algorithm 3 is the worst of the new heuristics. Algorithm 2 performs surprisingly well, almost as good as Algorithm 5 and would thus be considered the preferred choice for a system under real-time constraints.

2. Models with non-close efficiencies are much easier to solve than models with close efficiencies. Models with non-close costs seem to induce larger deviations from the optimal ECR than models with close costs. We consider none of these findings surprising.

We have also tested the algorithms on a real-world model with 32 actions. The results are summarized in the table below where the statistics are computed as the percent-wise deviation from the overall best algorithm (Algorithm 8 with $\frac{P}{C}$ ordering). The -0 indicates that a small percentage of the cases where better than Algorithm 8. Also notice that the apparent conflict between "median" and "% best" in column two is due to rounding of the median.

Model 5: 32 actions (33145)					
	$\mathrm{ef}(\alpha_i)$ ord.	Alg. 3	Alg. 2	Alg. 5	Alg. 8
min	0	0	0/0	-0/0	-0
max	667.30	3.79	25.30/19.05	10.68/0.05	6.07
mean	555.34	0.14	5.94/3.14	3.25/0	1.51
median	598.70	0	3.73/1.32	3.59/0	1.30
% best	0.01	48.15	0.01/0	0.01/99.43	0.01

The conclusion here is the same with one noticeable difference: Algorithm 3 is now the third best algorithm, only beaten by Algorithm 5 and 8 with $\frac{P}{C}$ ordering and far better than any other heuristic. The fact that Algorithm 5 and 8 are very close to each other also suggests that we are quite close to the optimal value.

Conclusion And Further Research

We have extended a classical polynomial troubleshooting scenario with postponed system test. Albeit this is a somewhat simple extension, we conjecture that the new scenario is NP-hard. We have identified a class of non-reordable models where the best of the described heuristics are guaranteed optimal. For reorable models, we found that the suggested heuristics provide quite close approximations to an optimal solution—even for heuristics with an $O(n \cdot \lg n)$ complexity. Apart from proving or disproving NP-hardness, future research includes finding guarantees on the performance of the heuristics as well as guarantees about local optimality when optimizing a given partition.

Acknowledgements

We would like to thank Sven Skyum for the discussions that lead to Theorem 5. We are also grateful to Dezide Aps for providing the real-world model used in the experiments. Finally, we would like to thank our collegues of the Machine Intelligence group at Aalborg University for providing such a fruitful research environment.

References

F. V. Jensen, U. Kjærulff, B. Kristiansen, C. Skaanning, J. Vomlel, and M. Vomlelová. The sacso methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15:321–333, 2001.

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

E. Koca and T. Bilgiç. Troubleshooting with questions and conditional costs. *Proceedings of the 13th Turkish Symposium on Artificial Intelligence and Artificial Neural Networks*, pages 271–280, 2004a.

E. Koca and T. Bilgiç. A troubleshooting approach with dependent actions. In R. L. de Mántaras and L. Saitta, editors, *ECAI 2004: 16th European Conference on Artificial Intelligence*, pages 1043–1044. IOS Press, 2004b. ISBN 1-58603-452-9.

H. Langseth and F. V. Jensen. Decision theoretic troubleshooting of coherent systems . *Reliability Engineering & System Safety*, pages 49–62, 2003.

H. Langseth and F. V. Jensen. Heuristics for two extensions of basic troubleshooting. In *Proceedings of the Seventh Scandinavian Conference on Artificial Intelligence*, pages 80–89. IOS Press, 2001. ISBN 1-58603-161-9.

C. Skaanning and J. Vomlel. Troubleshooting with simultaneous models. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 6th*, 2001.

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

H. Warnquist, M. Nyberg, and P. Säby. Troubleshooting when action costs are dependent with application to a truck engine. In *Proceeding of the 2008 conference on Tenth Scandinavian Conference on Artificial Intelligence*, pages 68–75, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press. ISBN 978-1-58603-867-0.