Implementing Troubleshooting with Batch Repair

Roni Stern, Meir Kalech, Hilla Shinitzky
Ben Gurion University of the Negev
Be'er Sheva, Israel

Abstract

Recent work has raised the challenge of efficient automated troubleshooting in domains where repairing a set of components in a single repair action is cheaper than repairing each of them separately. This corresponds to cases where there is a non-negligible overhead to initiating a repair action and to testing the system after a repair action. In this work we propose several algorithms for choosing which batch of components to repair, so as to minimize the overall repair costs. Experimentally, we show the benefit of these algorithms over repairing components one at a time.

Introduction

Troubleshooting algorithms, in general, plan a sequence of actions that are intended to fix an abnormally behaving system. Fixing a system includes repairing faulty components. Such repair actions incur a cost. These costs can be partitioned into two types of repair cost. The first, referred to as the *component repair cost*, is the cost of repairing a component. The second, referred to as the *repair overhead*, is the cost of preparing the system to perform repair actions (e.g., halting the system) and the cost of testing the system after performing a repair action.

This paper considers the case where the repair overhead is not negligible and is potentially more expensive than a component repair cost (of a single component). Therefore, it may be more efficient to repair a batch of components in a single repair action. We call the problem of choosing which batch of components to repair the Batch Repair Problem (BRP). BRP is an optimization problem, where the task is to minimize the *total repair costs*, which is the sum of component repair costs and costs due to repair overheads incurred by repair actions performed until the system is fixed.

Note that in this paper we use the term "fix" when referring to the entire system and term "repair" for a single or a set of components. Thus, to *fix* the system one needs to *repair* components, and a system is only fixed if it returned to its nominal behavior.

Most previous work assumed that components are repaired one at a time (Heckerman, Breese, and Rommelse 1995; Friedrich and Nejdl 1992; Pernestål, Nyberg, and Warnquist 2012; Torta, Anselma, and Dupré 2014). This approach can be wasteful for BRP. For example, if a diagnosis engine infers that multiple faulty components need to be repaired to fix the system, then it would be wasteful to repair these components one at a time since each repair action incurring its repair overhead. Instead, an efficient BRP algorithm would repair all the faulty components in a single repair action. More generally, we expect an intelligent BRP algorithm to consider to repair overheads and the component repair costs when deciding which components to repair.

Due to the repair overhead, repairing a single component, even if it is the component most likely to be faulty, can be wasteful. This is especially wasteful in cases where all the found diagnoses consist of multiple faulty components, thus suggesting that repairing a single component would not fix the problem. Alternatively, one may choose to repair the components in the single most likely diagnosis. This may also be wasteful, especially if there are several diagnoses which have similar likelihood. It might be worthwhile to repair, in a single repair action, a set of components that "covers" more than a single diagnosis. This may reduce the number of repair actions until the system is fixed, thus saving repair overhead costs. The downside in this approach is that healthy components may be repaired, increasing the overall repair costs.

For example, consider the boiler tank system scheme made by Warren Controls, Inc. described in Figure 1. When demand for water reduces the liquid level in the tank, the float cage (A) opens the lever valves (B) to supply intake water to the tank and closes it when the water reaches the desired level. Component C is an overflow trap which collects and relieves condensate overflow. Component D includes two vacuum breakers which are opened to relieve the tank with outside air to prevent vacuum pressures in the tank.

Assume additional water is required but the level of the water does not increase. There are two possible diagnoses: either the float cage A is faulty or the lever valve B is faulty. Assume that the probabilities of A and B to be faulty are given by the manufacturer and are 0.06 and 0.04, respectively. Since only these diagnoses can explain the problem, the normalized probability of A to cause the problem is 0.6 and that of B is 0.4. There are three possible repair actions: to repair A, to repair B, and to repair A and B. Assume the repair cost of each component is 5$ but the repair over-

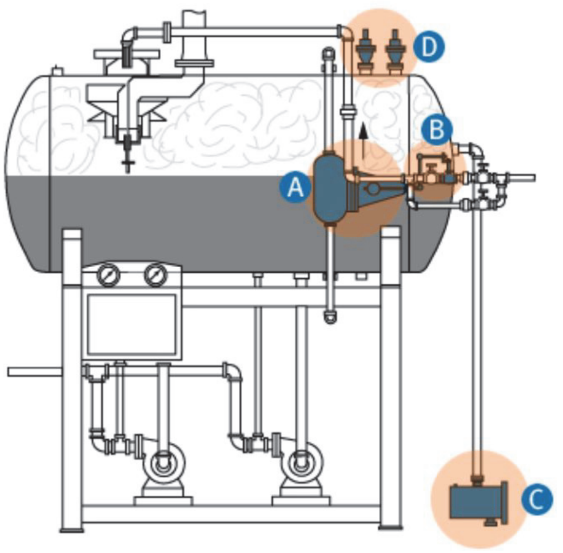

Figure 1: An example where repairing components one at a time is wasteful.

head is 50$, due to the cost of opening the tank and boiling the water. If A is repaired, there is a 0.4 chance that the system would not be fixed and another repair action would be needed (repairing B). Thus, the expected total repair cost of repairing A first is $0.4 \cdot (5 + 50) + 5 + 50 = 77$. Similarly, the total repair cost for repairing B first is $0.6 \cdot (5 + 50) + 5 + 50 = 88$. The best option is thus to repair A and B together in a single repair action, incurring a total repair cost of $5 + 5 + 50 = 60$.

In this work we model BRP as a combinatorial optimization problem, searching in the combinatorial space of possible repair actions for the best repair action. There are two challenges in implementing this approach: (1) how to measure the quality of a repair action, and (2) how to efficiently search for the repair action that maximizes this measure. There are many efficient heuristic search algorithms in the literature, and thus we focus on the first challenge — developing intelligent heuristics for estimating the merit of a repair action.

The contributions of this work are practical. A range of heuristic objective functions are proposed and analyzed, and we evaluate their effectiveness experimentally on a standard benchmark. A clear observation from the results is that indeed considering batch repair actions can save repair cost significantly and that intelligent heuristics are crucial in saving repair costs.

Problem Definition

Next, we provide background required for defining the batch repair problem we address. Following standard model-based diagnosis (MBD) terminology, we denote by *COMPS* and *OBS* the components in the system and the observed system behavior, respectively. *SD* describes the behavior of the diagnosed system, and in particular the behavior of each component. The term *behavior mode* of a component refers to a state of the component that affects its behavior. *SD* describes for every component one or more behavior modes. For every component, at least one of the behavior modes must represent the nominal behavior of the component. The normal mode is often described by the clause $h(C_i) \rightarrow \varphi_{C_i}$, where $C_i \in \mathit{COMPS}$, $h(C_i)$ is a predicate stating that C_i is healthy, and φ_{C_i} describes the nominal behavior of C_i. For instance, the nominal behavior of the lever valve (component B in Figure 1) is to be opened once the float cage opens it, while an abnormal behavior can be stuck open or close.

A *batch repair problem* (BRP) arises when the assumption that all components are normal is not consistent with the system description and observations. Formally,

$$\mathit{SD} \land \mathit{OBS} \land \bigwedge_{C \in \mathit{COMPS}} h(C) \quad \text{is not consistent}$$

A mode assignment ω is an assignment of *behavior modes* to components. Let $\omega^{(+)}$ be the set of components assigned a nominal (i.e., normal) behavior mode and $\omega^{(-)}$ be the set of components assigned one of the other modes.

Definition 1 (Diagnosis). *A mode assignment ω is called a diagnosis if $\omega \land \mathit{OBS} \land \mathit{SD}$ is satisfiable.*

In the example shown in Figure 1, assuming that all components are healthy under the observation that the water level is decreased is not consistent. Then a possible diagnosis is that components C and D are healthy, while either A or B are in an abnormal mode. For instance, the lever valve (B) is stuck close.
In such a case, at least one component must be repaired.

Definition 2 (Repair Action). *A repair action can be applied to any subset of components and results in these components becoming normal. Applying a repair action to a set of components γ is denoted by Repair(γ).*

Definition 2 assumes that repair actions always succeed, i.e., a component is normal after it is repaired.

After a repair action, the system is tested to check if it has been fixed. We assume that the system inputs in this test are the same as in the original observations (*OBS*). The observed system outputs are then compared to the expected system outputs of a healthy system. Thus, the result of a repair action is either that the system is fixed, or a new observation that may help choosing future repair actions.

A model-based diagnosis engine (MBDE) accepts as input *SD*, *OBS*, and *COMPS* and outputs a set of diagnoses Ω. Although a diagnosis is consistent with *SD* and *OBS*, it may be incorrect. A diagnosis ω is *correct* if by repairing the set of components in $\omega^{(-)}$ the system is fixed. Some diagnosis algorithms return, in addition to Ω, a measure of the likelihood that each diagnosis is *correct* (Williams and Ragno 2007; Abreu, Zoeteweij, and van Gemund 2011). Let $p : \Omega \rightarrow [0, 1]$ denote this likelihood measure. We assume that $p(\omega)$ is normalized so that $\sum_{\omega \in \Omega} p(\omega) = 1$ and use it to approximate the probability that ω is correct.

A common way to estimate the likelihood of diagnoses, assumes that each component has a prior on the likelihood that it would fail and component failures are independent.

Therefore, if $p(c)$ represents the likelihood that a component c would fail then diagnosis likelihood can be computed as

$$p(\omega) = \frac{\prod_{c\in\omega^-} p(c)}{\sum_{\omega'\in\Omega}\prod_{c\in\omega'^-} p(c)} \quad (1)$$

where the denominator is a normalizing factor. We assume in the rest of this paper that diagnoses likelihoods are computed according to Equation 1. Other methods for computing likelihood of diagnoses also exist (Mengshoel et al. 2010).

Repairing a set of components incurs a cost, composed of a repair overhead and component repair costs. The repair overhead is denoted by $cost_{repair}$, and the component repair cost of a component $c \in COMPS$ is denoted by $cost_c$.

Definition 3 (Repair Costs)**.** *Given a set of components $\gamma \subseteq$ COMPS, applying a repair action Repair(γ) incurs a cost:*

$$cost(Repair(\gamma)) = cost_{repair} + \sum_{c\in\gamma} cost_c$$

We assume that all repair costs are positive and non-zero, i.e., $cost_{repair} > 0$ and $cost_c > 0$ for every component $c \in$ *COMPS*. As defined earlier, the task in BRP is to fix a system with minimum total repair cost.

As shown in Figure 1, an efficient BRP solver should consider the possibility of repairing a set of components in a single repair action. Thus, the potential number of repair actions is $2^{|COMPS|}$. Therefore, from a complexity point of view BRP is an extremely hard problem.

System Repair Likelihood

If the MBDE returns a single diagnosis ω that is guaranteed to be correct, then the optimal solution to BRP would be to perform a single repair action: Repair(ω^-). This, however, is rarely the case, and more often a possibly a very large set of diagnoses is returned by diagnosis algorithms. This introduces uncertainty as to whether a repair action would actually fix the system. We define this uncertainty as follows:

Definition 4 (System Repair Likelihood)**.** *The System Repair Likelihood of a set of components $\gamma \subseteq$ COMPS, denoted* SystemRepair(γ)*, is the probability that Repair(γ) would fix the system.*

Consider the relation between $p(\omega)$ and SystemRepair(ω). If ω is correct, then repairing all components that are faulty, meaning $\omega^{(-)}$, would fix the system. Therefore, the likelihood of repairing $\omega^{(-)}$ causing the system to be fixed is at least $p(\omega)$, i.e.,

$$\texttt{SystemRepair}(\omega^{(-)}) \geq p(\omega)$$

Moreover, if ω is correct then repairing any superset of $\omega^{(-)}$ would also fix the system. Thus, SystemRepair($\omega^{(-)}$) may be larger than $p(\omega)$. On the other hand, repairing any set of components that is not a superset of $\omega^{(-)}$, as there would still be faulty components in the system. Therefore, a repair action Repair(*COMPS*$'$) would fix the system if and only if $\omega^{*(-)} \subseteq$ *COMPS*$'$, where ω^* is the correct diagnosis. While we do not know ω^*, we can compute SystemRepair(γ) from Ω and $p(\cdot)$:

$$\texttt{SystemRepair}(\gamma) = \sum_{\omega\in\Omega\wedge\omega\subseteq\gamma} p(\omega)$$

For example, in the boiler tank depicted in Figure 1, there are two diagnoses, $\{A\}$ and $\{B\}$, such that $p(\{A\}) = 0.6$ and $p(\{B\}) = 0.4$. Thus, SystemRepair($\{$A$\}$)=0.6, SystemRepair($\{$B$\}$)=0.4, and SystemRepair($\{$A, B$\}$)=$p(\{A\})$+$p(\{B\})$=1.

BRP as a Combinatorial Search

As mentioned in the introduction, the approach for solving BRP that we pursue in this paper formulates BRP as a combinatorial search problem. The search space is the space of possible repair actions, i.e., every subset of the set of components there were not repaired yet. The search problem is to find the repair action that maximizes a utility evaluation function $u(\cdot)$ that maps a repair action to a real value that estimates its merit.

The effectiveness of this search-based approach for BRP depends on the search algorithm used and how the $u(\cdot)$ utility function is defined. There are many existing heuristic search algorithm for searching large combinatorial search spaces (Russell and Norvig 2010; Edelkamp and Schroedl 2011). Thus, in this work we propose and evaluate a set of possible utility functions. Note that for some of the utility functions described next it is possible to find the best repair action without searching the entire search space of possible actions, while others are more computationally intensive.

k Highest Probability

A key source of information for all the utility functions described below is the set of diagnoses Ω and their likelihoods ($p(\cdot)$). We assume that this information is obtained by using a diagnosis engine over the observations of the current state of the system. The set of returned diagnoses may be very large. The first utility function we propose is based on the system's *health state*, which has been recently proposed as a method for aggregating information from a set of diagnoses (Stern et al. 2015).

Definition 5 (Health State)**.** *A health state is a mapping F : $COMPS \rightarrow [0, 1]$ where*

$$F(C) = \sum_{\omega\in\Omega s.t. C\in\omega} p(\omega)$$

$F(C)$ is an estimate of the likelihood that component C is faulty given a set of diagnoses Ω and their likelihoods. Based on the system's health state, we propose the following utility function, denoted u_{HP}:

$$u_{HP}(\gamma) = \sum_{C\in\gamma} F(C)$$

where γ is a subset of $COMPS$ that were not repaired yet.

The repair action that maximizes u_{HP} is trivial — repair all components. This would result in the system being repairs, but of course, may repair many components that are likely to be healthy. To mitigate this effect, we propose the k *highest probability* repair algorithm (k-HP), which limits the number of components that can be repaired in a single repair action to k, where k is a user-defined parameter. Note that computing k-HP does not need any exhaustive search: simply sort the health state in descending order of $F(\cdot)$ values and repair the first k components.

The k-HP repair algorithm has two clear disadvantages. First, the user needs to define k. Second, k-HP does not consider repair costs (neither component repair costs nor overhead costs). The next set of utility functions and corresponding repair algorithms address these disadvantages.

Wasted Costs Utilities

Repairing a system requires performing repair actions. Some repair costs are inevitable. These are the repair overhead of a single repair action, and the component repair costs that repair the faulty components. We propose a family of utility functions that try to estimate the expected total repair costs beyond these inevitable costs. We refer to these costs as *wasted costs* and to utility functions of this family as *wasted cost functions*. We model these wasted costs as being composed of two parts.

- **False positive costs ($cost_{FP}$).** These are the costs incurred by repairing components that are not really faulty.
- **False negative costs ($cost_{FN}$).** These are the overhead costs incurred by future repair actions.

It is clear why the false positive costs are wasted costs — these are repair costs incurred on repairing healthy components. The false negative costs are wasted costs because if one knew upfront which components are faulty, then the optimal repair algorithm would repair all these components in a single batch repair action, incurring no further overhead costs. Thus, future overhead costs represent wasted costs.

We borrow the terminology of false positive and false negative from the machine learning literature, but use it in a somewhat different manner. To explain this choice of terminology, assume that positive and negative mean faulty and healthy components respectively. Choosing to repair a faulty component is regarded as a true positive, and not repairing a healthy component is regarded as a true negative. Thus, the wasted costs incurred by repairing healthy components are costs incurred due to false positives, and the wasted costs incurred by not repairing a faulty component are overhead costs incurred due to false negatives. While this is not a perfect match in terminology, we belief that it helps clarify the underlying intention of $cost_{FP}$ and $cost_{FN}$.

The Wasted Cost Utility Function For a given set of components γ, we denote by $cost_{FP}(\gamma)$ and $cost_{FN}(\gamma)$ the false positive costs and false negative costs, respectively, incurred by performing a batch repair action of repairing all the components in γ. Given $cost_{FP}(\gamma)$ and $cost_{FN}(\gamma)$, we propose the following general formula for computing the expected wasted costs, denoted by C_{WC}.

$$C_{WC} = cost_{FP}(\gamma) + (1 - \texttt{SystemRepair}(\gamma)) \cdot cost_{FN}(\gamma)$$

The left hand side of the formula is the false positive costs. The right hand side of the formula is the false negative costs, multiplied by the probability that the system will not be fixed by repairing the components in γ. Thus, the formula gives the total expected wasted costs. We define $U_{WC} = -C_{WC}$ as the *wasted cost utility function*.

The wasted cost utility function is a theoretical utility function, since one does not know upfront the values of $cost_{FP}$ and $cost_{FN}$. Next, we propose several ways to estimate U_{WC} by proposing ways to estimate $cost_{FP}$ and $cost_{FN}$.

Estimating the False Positives Cost: We propose to estimate the false positive costs by considering the system's health state (Definition 5), as follows.

$$\widehat{cost}_{FP}(\gamma) = \sum_{C \in \gamma} (1 - F(C)) \cdot cost(C)$$

This estimate of the false positive costs can be understood as an expectation over the false positive costs. The cost of a repaired component $C \in \gamma$ is part of the false positive costs only if C is in fact healthy. The probability of this occurring is $(1 - F(C))$. Thus, $(1 - F(C)) \cdot cost(C)$ is the expected false positive cost due to repairing component C.

Estimating the False Negatives Cost: Correctly estimating $cost_{FN}$ is more problematic than $cost_{FP}$, as it requires considering the future actions of the repair algorithm. In the best case, only one additional repair action would be needed. This would incur a single additional overhead cost. We call this **the optimistic** $cost_{FN}$, or simply $cost^{o}_{FN}$, which is equal to $cost_{repair}$. The other extreme assumes that every component not repaired so far would be repaired by a single repair action, and correspondingly an incurred overhead cost. We experimented with a slightly less extreme estimate, in which we assume that only faulty components will be repaired in the future, but each will be repaired in a single repair action, incurring one $cost_{repair}$ per faulty component. Since we do not know the number of faulty components, we use the expected number of faulty components according to the health state: $\sum_{c \notin \gamma} F(c)$. The resulting estimate is referred to as **the pessimistic** $cost_{FN}$, denoted by $cost^{p}_{FN}$, is thus computed as:

$$cost^{p}_{FN}(\gamma) = cost_{repair} \cdot \sum_{c \notin \gamma} F(c)$$

To summarize, we propose two utility functions from the wasted cost utility function family. A pessimistic wasted cost function, that uses $\widehat{cost}_{FP}$ and $cost^{p}_{FN}$ to estimate $cost_{FP}$ and $cost_{FN}$, and an optimistic wasted cost function that uses $\widehat{cost}_{FP}$ and $cost^{o}_{FN}$. The corresponding repair algorithms search in the combinatorial space of all possible sets of components to find the set of components that maximizes U_{WC}.

Handling the Computational Complexity

The search space is very large — the size of the power set of all components that were not repaired so far. We explored two simple ways to handle this. The first approach is to only consider subset of components with up to k components,

Name	$\vert$COMPS$\vert$	in	out	#observations
74181	65	14	8	26
74182	19	9	5	25
74283	36	9	5	22
c432	160	36	7	23
c499	202	41	32	22
c880	383	60	26	30

Table 1: The Benchmark suite: systems 74XXX and ISCAS-85, and scenarios Feldman.

where k is a parameter. We set the components in a decreasing order of their health state (Definition 5) and choose the first k components. Thus we increase the probability of repairing faulty components. This approach is referred to as **Powerset-based search**.

The second approach considers only supersets of the diagnoses in Ω. This has the intuitive reasoning that at least one of these diagnoses is supposed to be true (according to the known observation), and thus a repair algorithm should try to aim for fixing the problem in the next repair action. Thus, in this approach, we considered in the search for the best repair action every set of components that are unions of at most k diagnoses, where k is a parameter. We set the diagnoses in a decreasing order of their likelihood (Equation 1) and choose the first k diagnoses. Thus we increase the probability of repairing faulty components. This approach is referred to as the **Union-based search**.

For both powerset-based search and union-based search, increasing k results in a larger search space and consequently higher computational complexity. On the other hand, a large search space increases the range of repair actions considered, and thus higher k can potentially find better repair actions. This provides an often desired tradeoff of computation vs. solution quality. This trend is observable in our experimental results below.

Experimental Results

We evaluated the proposed batch selection algorithms on standard Boolean circuit systems. Figure 2 presents a logic diagram of one of these systems, a known MSI chip called the 74181. It is an arithmetic logic unit (ALU) that provides thirty-two functions of two 4-bit variables. *COMPS* in this example include the Boolean gates in the ALU. *SD* is the behavior description of the components, for instance, the healthy behavior of an *OR* gate implies the "*OR*" behavior while abnormal behaviors can be stuck at 1 or at 0. *OBS* includes the inputs and outputs of the ALU. A diagnosis states which gates are healthy and which are in a faulty mode. The batch repair algorithms propose a set of gates to fix.

The standard Boolean circuits we used in our experiments are presented in Table 1. The systems 74XXX (Hansen, Yalcin, and Hayes 1999) are described in the first three rows, and additional three systems of ISCAS-85 (Brglez, Bryan, and Kozminski 1989) are described in the following three rows. Observations were selected randomly from Feldman et al.'s (2010) known benchmark. To adapt these benchmark systems and observations to be an experimental infrastructure for batch repair algorithms we set the prior probability of each gate to be faulty to 0.01 and chose a single diagnosis for each observation to serve as the injected faults. This is needed to decide when the system is fixed. Note that this "true" diagnosis was chosen with probability proportional to its likelihood of being correct, computed according to the priors mentioned above under the standard assumption of fault independence. The component repair cost was set to 5, and we experimented with repair overhead ($cost_{repair}$) costs of 10, 15, 20, and 25.

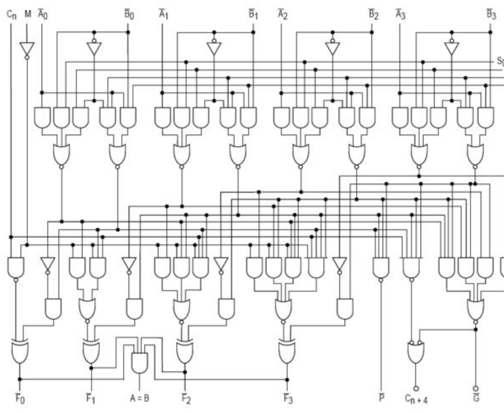

Figure 2: A logic diagram of ALU 74181.

All batch repair algorithms used a simple MBDE based on exhaustive search to generate diagnoses. Diagnoses were generated in order of increasing cardinality, and halted after either all subset minimal diagnoses were found or a timeout of 15 minutes was reached. We did not use a more sophisticated MBDE as it is not the focus of this work. Note that the batch repair algorithms are applicable to any MBDE.

Baseline Repair Algorithms

The main hypothesis of this line of work is that performing a batch repair action can save repair costs. To evaluate if the proposed batch repair algorithms are able to do so we compare them with 1-HP, in which the component that is most likely to be faulty is repaired. A similar approach was used by previous work on test planning (Zamir, Stern, and Kalech 2014). Another baseline repair algorithm we evaluated experimentally is to repair all components of the most likely diagnosis in a single batch repair action denoted *Batch Best Diagnosis* (hereinafter, BD-batch).

Results

Table 2 shows the average repair costs incurred until the system was fixed for the systems and problem instances described above. The first column lists the name of the compared algorithms, where $Opt.(\cdot)$ and $Pes.(\cdot)$ denote the union-based search using either pessimistic or the optimistic wasted costs utility functions, i.e., where $cost^{p}_{FN}$ and $cost^{o}_{FN}$ are used to estimate $cost_{FN}$, respectively, and the number in brackets (either 1 or 2) is the value of k. We did not experiment with larger values of k due to com-

System	74182				74283				74181				c432				c499				c880			
Overhead	10	15	20	25	10	15	20	25	10	15	20	25	10	15	20	25	10	15	20	25	10	15	20	25
1-HP	82	110	137	164	106	142	177	213	119	159	199	239	60	80	100	120	41	55	69	83	144	191	239	287
BD	59	73	88	103	81	101	122	142	92	115	137	160	83	109	135	161	**28**	36	45	53	95	123	151	178
2-HP	62	78	94	109	85	107	128	149	85	107	128	149	47	59	71	83	32	40	49	51	90	109	127	145
3-HP	57	69	81	93	77	93	108	124	83	99	116	132	**44**	**53**	**62**	72	30	37	44	46	**87**	**102**	**117**	**131**
4-HP	58	69	79	90	78	91	104	117	82	96	110	124	47	55	63	72	**28**	**34**	**40**	62	117	151	186	220
Opt. (1)	56	69	83	97	70	89	108	125	76	95	113	131	50	65	79	94	33	43	52	62	117	151	186	220
Opt. (2)	**54**	65	71	82	68	82	95	103	**75**	91	107	121	51	63	73	84	33	41	49	52	114	147	179	207
Pes. (1)	58	69	81	97	68	89	109	128	77	95	111	129	51	65	76	88	33	43	52	62	118	153	187	225
Pes. (2)	56	**58**	**64**	**73**	**65**	**73**	**83**	**91**	76	**90**	**102**	**110**	51	61	63	**69**	32	40	45	**49**	118	149	178	205

Table 2: Average repair costs until system is fixed.

putational complexity. The powerset-based search approach yielded substantially worse results compared to the union-based results so we do not display it in Table 2. The other columns in Table 2 show the results for different overhead costs – 10, 15,20, and 25. For every system and column, we marked in bold the best performing algorithm for every combination of repair overhead cost and system.

The first trend we highlight is that in all cases, using a batch repair algorithm resulted in significantly less costs compared to the 1-HP, in which a single component is repaired in each round. This supports our main claim that reasoning about the possibility of batch repair is important. For example, in the 74181 system when the repair overhead is 25, 1-HP required an average cost of 239 while Pes.(2) needed only 110.

Increasing the repair overhead causes all algorithms to require more cost to fix the system. However, the advantage of batch repair algorithms over 1-HP increase as repair overhead costs increases, demonstrating that the importance of batch repair is greater when overhead costs are higher.

Also, in most cases the trivial BD-batch algorithm did not perform well, suggesting non-trivial algorithms are needed for intelligent use of batch repair. For example, in the c499 system with repair overhead of 25, BD-batch required an average cost of 161 while Pes.(2) needed only 69. No clear winner was observed when comparing the non-baseline approaches (k-HP, Opt.(k), and Pes.(k)), and in general most of these algorithms performed well. However, we do observe that in general Pes.(2) is more robust in most system, being either the best performing or close to it in all systems except c880.

Related Work

BRP is a troubleshooting problem, where the goal is to perform repair actions to fix a system. Algorithms for automated troubleshooting were proposed in previous works. Heckerman et al. (1995) proposed the *decision-theoretic troubleshooting* (DTT) algorithm, that uses a decision theoretic approach for deciding which components to observe in order to identify the faulty component. Later work also applied a decision theoretic approach that integrated planning and diagnosis to a real world troubleshooting application (Pernestål, Nyberg, and Warnquist 2012; Warnquist, Kvarnström, and Doherty 2009). Torta et al. (2014) proposed using model abstractions for troubleshooting while taking into account the cost of repair actions. All these works did not consider the possibility of repairing a set of components together, allowing only repair actions that repair a single component at a time.

Our current paper do not consider applying further diagnostic actions such as probing and testing, which are considered by previous troubleshooting algorithms. Thus, our work on BRP could be integrated in previous troubleshooting frameworks so as to consider both batch repair actions and diagnostic actions. This is left to future work.

Friedrich and Nedjl (1992) discussed the relation between diagnoses and repair, in an effort to minimize the *breakdown costs*. Breakdown costs roughly correspond to a penalty incurred for every faulty output in the system, for every time step until the system is fixed. In BRP, the goal is to minimize costs until the system if fixed, and there is no partial credit for repairing only some of the system outputs.

Cordier et. al. (2008) discuss self-healability that considers both the diagnosabilty as well as the repairabilty of the system. Repairability deals with the possibility that a system will be fixed by repairing a subset of the system which is equivalent to batch repair. The paper only lays the basic definitions but does not address the question of how to select the batch of components.

Conclusion and Future Work

We addressed the problem of troubleshooting with the possibility of performing a batch repair action — a repair action in which more than a single component is repaired. Batch repair makes sense only if repairing a set of components in a single repair action is cheaper than repairing each of them separately. We proposed several algorithms for selecting which batch of components to repair. Experimental results clearly show the benefit of batch repair over single repair actions, and the benefit of the algorithms we suggested for choosing these set of components to repair.

The computation of the proposed utility functions embodied several assumptions. First, components are assumed to fail independently (this is used in Equation 1). Second, we assume that a batch repair action always succeeds, i.e., all repaired components are healthy after it. Third, we assume that overhead cost do not depend on the components being repaired. In future work we will investigate how relaxing these assumptions. Additionally, an alternative approach to address the batch repair problem may consider BRP as a planning under uncertainty problem, model it as a Markov Decision Process (MDP) and solve it appropriately. Finally, we plan to evaluate the proposed approaches experimentally on a realistic domain.

References

Abreu, R.; Zoeteweij, P.; and van Gemund, A. J. C. 2011. Simultaneous debugging of software faults. *Journal of Systems and Software* 84(4):573–586.

Brglez, F.; Bryan, D.; and Kozminski, K. 1989. Combinatorial profiles of sequential benchmark circuits. In *IEEE International Symposium on Circuits and Systems*, 1929–1934.

Cordier, M.-O.; Pencolé, Y.; Travé-Massuyès, L.; and Vidal, T. 2008. Characterizing and checking self-healability. In *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, 789–790. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Edelkamp, S., and Schroedl, S. 2011. *Heuristic search: theory and applications*. Elsevier.

Feldman, A.; Provan, G.; and van Gemund, A. 2010. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research (JAIR)* 38:371.

Friedrich, G., and Nejdl, W. 1992. Choosing observations and actions in model-based diagnosis/repair systems. *KR* 92:489–498.

Hansen, M. C.; Yalcin, H.; and Hayes, J. P. 1999. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Des. Test* 16:72–80.

Heckerman, D.; Breese, J. S.; and Rommelse, K. 1995. Decision-theoretic troubleshooting. *Communications of the ACM* 38(3):49–57.

Mengshoel, O.; Chavira, M.; Cascio, K.; Poll, S.; Darwiche, A.; and Uckun, S. 2010. Probabilistic model-based diagnosis: An electrical power system case study. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 40(5):874–885.

Pernestål, A.; Nyberg, M.; and Warnquist, H. 2012. Modeling and inference for troubleshooting with interventions applied to a heavy truck auxiliary braking system. *Engineering Applications of Artificial Intelligence* 25(4):705–719.

Russell, S. J., and Norvig, P. 2010. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education.

Stern, R. T.; Kalech, M.; Rogov, S.; and Feldman, A. 2015. How many diagnoses do we need? In Bonet, B., and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 1618–1624. AAAI Press.

Torta, G.; Anselma, L.; and Dupré, D. T. 2014. Exploiting abstractions in cost-sensitive abductive problem solving with observations and actions. *AI Commun.* 27(3):245–262.

Warnquist, H.; Kvarnström, J.; and Doherty, P. 2009. Planning as heuristic search for incremental fault diagnosis and repair. In *Scheduling and Planning Applications Workshop (SPARK) at the International Conference on Automated Planning and Scheduling (ICAPS)*.

Williams, B. C., and Ragno, R. J. 2007. Conflict-directed A* and its role in model-based embedded systems. *Discrete Applied Mathematics* 155(12):1562–1595.

Zamir, T.; Stern, R. T.; and Kalech, M. 2014. Using model-based diagnosis to improve software testing. In Brodley, C. E., and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 1135–1141. AAAI Press.