Implementing Troubleshooting with Batch Repair

Roni Stern[1] and **Meir Kalech**[1] and **Hilla Shinitzky** [1]
[1]Ben Gurion University of the Negev
e-mail: roni.stern@gmail.com, kalech@bgu.ac.il, hillash@post.bgu.ac.il

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 (and not as a batch).

1 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 may be required), 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 the repair overheads and component repair costs incurred by all the repair actions performed until the system is fixed.

Note that in this paper we use the term "repair" for a single or a set of components and the term "fix" to refer to the entire system. Thus, *repairing* components eventually causes the system to be *fixed*, 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 [1; 2; 3; 4]. 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 weigh the cost of repairing batches of components as well as the repair overhead. Some discussion on repairing multiple components together was done in prior work on self healability [5].

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 consists 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 most likely diagnoses. This may also be wasteful, especially if there are several diagnoses which have similar likelihood. It might be worthwhile to repair by 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 the component repair costs can be high, as more healthy components may be repaired.

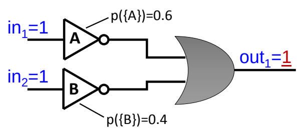

Figure 1: An example where repairing components one at a time is wasteful.

For example, consider the small system described in Figure 1. It is a logical circuit whose output is fault. Assume that the "OR" gate is known to be healthy and there are only two possible diagnoses: either A is faulty or B is faulty, where the probability that A and B are faulty is 0.6 and 0.4, respectively. There are three possible repair actions: to repair A, to repair B, and to repair A and B. Assume the repair overhead costs 10, and repairing a component costs 1. 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 15.4. Similarly, the total repair cost for repairing B first is 17.6. The best option is thus to repair A and B together in a single repair action, incurring a total repair cost of 12.

Recent work [6] proposed two high-level approaches to solve BRP: as a planning under uncertainty problem, or as a combinatorial optimization problem. When modeling BRP as a planning under uncertainty problem the task is to find a *repair policy*, mapping a state of the system to the repair action that minimizes the expected total repair costs. This approach, while attractive theoretically, quickly becomes not feasible in non-trivial scenarios.

In this work we focus on the second high-level approach proposed for BRP, in which BRP is modeled 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. First, how to measure the quality of a repair action and how to efficiently search for the repair action that maximizes this measure. There are many efficient heuristic search algorithms in the literature, and thus the main challenge addressed in this work is in proposing several 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. Moreover, the most effective heuristics provide a tunable tradeoff between computation time and resulting repair costs.

2 Problem Definition

A classical MBD input $\langle SD, COMPS, OBS\rangle$ is assumed, where SD is a model of the system, COMPS represents the components in the system, and OBS is the observed behavior of the system. Every component can be either normal or abnormal. The assumption that a component $c \in COMPS$ is abnormal is represented by the abnormal predicate $AB(c)$.
A *batch repair problem* (BRP) arises when the assumption that all components are normal is not consistent with the system description and observations. Formally,

$$SD \wedge OBS \wedge \bigwedge_{c\in COMPS} \neg AB(c) \quad \text{is not consistent}$$

In such a case, at least one component must be repaired.

Definition 1 (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 1 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.

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 2 (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.

3 Preliminaries

Next, we provide background and definitions required for describing the BRP algorithms we propose.

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.

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 3 (Diagnosis)**.** *A mode assignment ω is called a diagnosis if $\omega \wedge OBS \wedge SD$ is satisfiable.*

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* [7; 8]. 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

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

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 [9].

3.1 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 logical circuit 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.

4 BRP as a Combinatorial Search Problem

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 [10; 11]. 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.

4.1 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 [12].

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 any subset of $COMPS$ that has not been 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.

4.2 Wasted Costs Utilities

Before describing the next set of proposed utility functions we explain the over-arching reasoning behind it. 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 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 fast 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 wastes costs, denoted by 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 wastes 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.

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

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.

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 component 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 estimate of $cost_{FN}$, denoted by $cost^{p}_{FN}$, is thus computed as:

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

Summarizing all the above, 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}.

4.3 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, where k is a parameter. This approach is referred to as *Powerset-based search.*

The second approach we considered is to consider 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. 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. This means higher computational complexity, but also increases the range of repair actions considered, and thus using higher k can potentially find better repair actions than using lower k values. This provides an often desired tradeoff of computation vs. solution quality. Experimentally, we observed that the union-based search approach yields much better results and thus we only show results for it in the experimental results below.

5 Experimental Results

We evaluated the proposed batch selection algorithms on two standard Boolean circuits: 74283 and 74182. We experimented on 21 observations for system 74283 and 23 observations for system 74182. These observations were selected randomly from Feldman et al.'s [13] set of observations. For each observation, all subset minimal diagnoses were found using exhaustive search.

5.1 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 two repair algorithms that do not consider batch repair actions. These baseline repair algorithms, named "Best Diagnosis" (BD) and "Highest Probability" (HP), are inspired by previous work on test planning [14] and work as follows. BD chooses to repair a single component from the most preferred diagnosis in Ω (that with the highest $p(\cdot)$ value). From the set of components in the most probable diagnosis, BD chooses to repair the one with the lowest repair costs. The HP repair algorithm chooses to repair the component that is most likely to be faulty, as computed by the system's health state ($F[\cdot]$).

Another baseline repair algorithm we evaluated experimentally that serves as a baseline is to repair all components of the most likely diagnosis in a single batch repair action. Note that this algorithm, denoted *Batch Best Diagnosis*, ignores repair costs, and serves as an extreme alternative to the BD algorithm that repairs a single component from the most likely diagnosis.

Table 1 shows the average repair costs incurred until the system was fixed for the proposed repair algorithms. The average was over all the observations we used for system 74182. The rows labeled BD, HP, 2-HP, and 3-HP show the

Algorithm	Overhead cost 10	15	20	25
BD & HP	83.5	111.3	139.1	167.0
2-HP	61.5	77.8	94.1	110.4
3-HP	53.0	65.0	77.0	88.9
Opt.(1)	55.2	68.9	82.6	96.3
Opt.(2)	53.0	65.0	75.2	86.7
Opt.(3)	55.2	66.5	72.6	83.7
Pes.(1)	55.0	68.9	81.3	96.1
Pes.(2)	52.8	59.8	63.7	70.0
Pes.(3)	**49.6**	**50.4**	**55.9**	**64.6**

Table 1: Average repair costs for the 74182 system.

Algorithm	Overhead cost 10	15	20	25
BD	116.4	155.2	194.0	232.9
HP	109.3	145.7	182.1	218.6
2-HP	81.2	102.1	123.1	144.0
3-HP	70.5	85.7	101.0	116.2
Opt.(1)	76.0	95.7	115.2	134.8
Opt.(2)	72.9	89.8	102.4	111.7
Pes.(1)	75.2	95.7	114.0	134.8
Pes.(2)	**72.4**	**84.8**	**93.6**	**96.0**

Table 2: Average repair costs for the 74283 system.

results for the BD, HP, and k-HP repair algorithms (for k=2 and 3). The rows Opt.(1), Opt.(3), and Opt.(3) show the results for the union-based search repair algorithm using the wasted cost utility function with $\widehat{cost}_{FP}$ to estimate $cost_{FP}$ and $cost^o_{FN}$ to estimate $cost_{FN}$. The rows Pes.(1), Pes.(2), and Pes.(3) show results for the same configuration, except for using $cost^p_{FN}$ to estimate $cost_{FN}$ instead of $cost^o_{FN}$. The repair costs of a single component was arbitrary set to 5 and the cost of the overhead ($cost_{repair}$) was varied (10,15,20,25). Each column represents results for different values of $cost_{repair}$. In this domain, the results of HP and BD were virtually the same, and thus we grouped them to a single row.

The results clearly show the benefit of considering batch repair actions. The best performing repair algorithm is Pes.(3), which required more than half the repair costs needed for BD and HP, which do not consider batch repair. This supports the main hypothesis of this paper: batch repair actions can save significant amount of repair costs. As expected, the gain of batch repair actions increases as the repair overhead ($cost_{repair}$) increases. Also note that for Pes.(k) we observe the desired trend of increasing k resulting in lower repair costs. This is also observed for the k-HP repair algorithm (note that the HP algorithm is in fact 1-HP), but is not always the case for Opt.(k), where for lower overhead cost $k = 2$ yielded lower repair costs than $k = 3$. This suggests that the optimistic estimate of $cost_{FN}$ is not robust. Computationally, increasing k required much more runtime, and we could not run experiments with $k = 4$ on our current machines in reasonable time. Table 2 shows the results for the 74283 system. The trends observed are the same as those discussed above for the results of 74182 system.

6 Related Work

BRP is a troubleshooting problem, where the goal is to perform repair actions so as to fix a system. Algorithms for automated troubleshooting were proposed in previous works. Heckerman et al. [1] 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 [3; 15]. Torta et al. [4] 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 on BRP 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 [2] 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.

7 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. Future work will investigate when should batch repair be considered, and how to detect such cases upfront. Additionally, expanding beyond Boolean circuits is also needed, as well as addressing uncertainty on the outcome of repair actions.

References

[1] David Heckerman, John S Breese, and Koos Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57, 1995.

[2] Gerhard Friedrich and Wolfgang Nejdl. Choosing observations and actions in model-based diagnosis/repair systems. *KR*, 92:489–498, 1992.

[3] Anna Pernestål, Mattias Nyberg, and Håkan Warnquist. Modeling and inference for troubleshooting with interventions applied to a heavy truck auxiliary braking system. *Engineering Applications of Artificial Intelligence*, 25(4):705–719, June 2012.

[4] Gianluca Torta, Luca Anselma, and Daniele Theseider Dupré. Exploiting abstractions in cost-sensitive abductive problem solving with observations and actions. *AI Commun.*, 27(3):245–262, 2014.

[5] Marie-Odile Cordier, Yannick Pencolé, Louise Travé-Massuyès, and Thierry Vidal. Self-healablity = diag-

nosability + repairability. In *the International Workshop on Principles of Diagnosis (DX)*, pages 251–258, 2007.

[6] Roni Stern and Meir Kalech. Repair planning with batch repair. In *International Workshop on Principles of Diagnosis (DX)*, 2014.

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

[8] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. Simultaneous debugging of software faults. *Journal of Systems and Software*, 84(4):573–586, 2011.

[9] O.J. Mengshoel, M. Chavira, K. Cascio, S. Poll, A. Darwiche, and S. Uckun. 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, 2010.

[10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

[11] Stefan Edelkamp and Stefan Schroedl. *Heuristic search: theory and applications*. Elsevier, 2011.

[12] Roni Stern, Meir Kalech, Shelly Rogov, and Alexander Feldman. How many diagnoses do we need? In *AAAI*, 2015.

[13] Alexander Feldman, Gregory Provan, and Arjan van Gemund. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research (JAIR)*, 38:371, 2010.

[14] Tom Zamir, Roni Stern, and Meir Kalech. Using model-based diagnosis to improve software testing. In *AAAI (to appear)*, 2014.

[15] Håkan Warnquist, Jonas Kvarnström, and Patrick Doherty. 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)*, 2009.