Artificial Intelligence 248 (2017) 26–45

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

How many diagnoses do we need?

Roni Stern [a,*], Meir Kalech [a], Shelly Rogov [a], Alexander Feldman [b]

[a] *Ben Gurion University of the Negev, Israel*
[b] *Palo Alto Research Center (PARC), United States*

ARTICLE INFO

Article history:
Received 24 December 2015
Received in revised form 2 March 2017
Accepted 5 March 2017
Available online 8 March 2017

Keywords:
Artificial intelligence
Model-based diagnosis

ABSTRACT

A known limitation of many diagnosis algorithms is that the number of diagnoses they return can be very large. This is both time consuming and not very helpful from the perspective of a human operator: presenting hundreds of diagnoses to a human operator (charged with repairing the system) is meaningless. In various settings, including decision support for a human operator and automated troubleshooting processes, it is sufficient to be able to answer a basic diagnostic question: is a given component faulty? We propose a way to aggregate an arbitrarily large set of diagnoses to return an estimate of the likelihood of a given component to be faulty. The resulting mapping of components to their likelihood of being faulty is called the system's *health state*. We propose two metrics for evaluating the accuracy of a health state and show that an accurate health state can be found without finding all diagnoses. An empirical study explores the question of how many diagnoses are needed to obtain an accurate enough health state, and an online stopping criteria is proposed.

1. Introduction

A diagnosis problem arises when a system does not behave as expected. A solution to a diagnosis problem is a *diagnosis*, which is a set of components assumed to have caused the system's abnormal behavior. One of the fundamental approaches to automated diagnosis is Model-Based Diagnosis (MBD) [1,2]. MBD has been studied in a vast spectrum of domains, including: automotive industry [3,4], space [5,6], robotics [7,8] and computer software [9,10].

MBD requires an explicit model of the normal system behavior. Most MBD algorithms consider as a *diagnosis* every assumptions about which components are faulty that is *logically consistent* with the system model and the observed system behavior. This consistency-based approach to solve the diagnosis problem is principled and elegant, and many consistency-based diagnosis algorithms (DAs) have been proposed throughout the years (see examples referenced above).

However, consistency-based DAs have a known limitations: the number of diagnoses that can be deduced from the system model and observations is often very large in non-trivial systems. This may be due to having an insufficient number of observations, an inaccurate model of the diagnosed system, and computational limitations.

Some approaches apply minimality criteria to limit the number of diagnoses returned. One approach displays only diagnoses with the smallest number of components. Such diagnoses are known as *minimal cardinality diagnoses*. Considering only minimal cardinality diagnoses may miss potentially important diagnoses [11]. Another common approach is to return only diagnoses that are not a superset of another diagnosis. Such diagnoses are called *minimal-subset diagnoses*. Unfortunately,

* Corresponding author.
E-mail addresses: roni.stern@gmail.com (R. Stern), kalech@bgu.ac.il (M. Kalech), shellyr4u@gmail.com (S. Rogov), afeldman@parc.com (A. Feldman).

http://dx.doi.org/10.1016/j.artint.2017.03.002
0004-3702/© 2017 Elsevier B.V. All rights reserved.

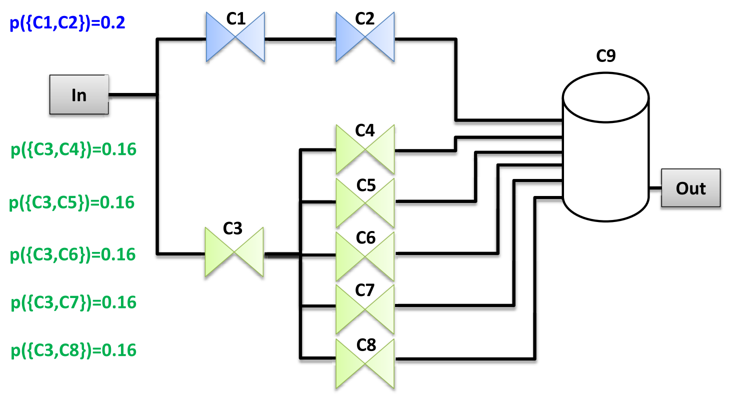


Fig. 1. An example where viewing the most probable diagnosis can be more misleading than viewing the system's health state. The system illustrates a hydraulic system, where components $C1, \ldots, C8$ are closed valves and component $C9$ is a tank. The observed system behavior is that oil flows into the system (depicted in the figure by the rectangle named "In") and unexpectedly some oil leaks in to the tank ($C9$), whose oil level is monitored by a sensor (depicted in the figure by the rectangle named "Out'). The most probable diagnosis is $\{C1, C2\}$, having a probability of 0.2. The health state, however, would show that $C3$ has a 0.8 probability of being faulty, and is therefore more likely to be faulty than $C1$ and $C2$.

for systems with a large number of components, even the number of minimal cardinality diagnoses is so large that even enumerating them is time consuming [12]. For instance, in the known ISCAS-85 benchmark [13] the average number of diagnoses for the observations of Siddiqi [12] for system c880 with 383 components is 963,342 and for the system c1908 with 880 components is 1,894,733 [14].

An alternative approach to reduce the number of diagnoses is to perform additional diagnostic actions, e.g., probing internal components and performing additional system testing [1,15,16]. Such actions can provide invaluable information for reducing the set of possible diagnosis. However, these actions often are costly that make it prohibitively expensive to reduce the set of diagnoses to a manageable size. Moreover, in some cases the set of possible diagnoses may stay too large even after performing any number of additional tests, due to inherent ambiguity of the system [17,18].

Some DAs assign a score to every returned diagnosis, which estimates the likelihood of each diagnosis to be correct. Given the likelihood of each diagnosis to be correct, one may consider displaying only the most probable diagnoses. Fig. 1, which is explained in greater detail later in this paper, shows an example where returning the most likely diagnosis is very misleading. The most likely diagnosis is that C_1 and C_2 are faulty, but it is actually more likely that C_3 is faulty since C_3 appears as faulty in most diagnoses. Thus, while showing a long list of diagnoses to a human operator is not helpful, focusing on a single diagnosis may miss important diagnostic information.

The **first contribution** of this work is a method to aggregate the information stored in an arbitrarily large set of diagnoses in a useful and compact form. This aggregated form, which we call the *health state* of the diagnosed system, maps every component in the system to the likelihood of it being faulty. Having a system's health state allows answering a basic diagnostic question: is a given component faulty? This question is key in various diagnostic settings, including decision support for a human operator and automated troubleshooting processes.

The **second contribution** of this work is an experimental investigation of the tradeoff between the quality of a health state and the number of diagnoses used to generate it. For this purpose, we propose three metrics for measuring the quality of a health state. The first metric is based on Euclidean distance from the optimal health state (where the true faults are known). The second metric views a health state as a classifier's model, and measures its quality according to the area under the curve (AUC) of a receiver operating characteristic (ROC) curve [19], which is a standard evaluation metric in the Machine Learning literature. The third metric is an adaptation of the wasted effort metric previously proposed for evaluating diagnoses [20,21].

The experimental investigation was performed over all these quality metrics. The first domain we experimented on is Boolean circuits, using the well established benchmarks ISCAS-85 [13] and 74XXX [22]. The second domain we used is software diagnosis, using the source files and reported bugs of Apache Ant, a popular open-source Java build tool [23]. For all quality metrics, we observed that the health state converges to a stable quality value without finding all diagnoses. This stable value is shown to be close to the best quality achievable, and in most cases such a value can be obtained by considering only minimal cardinality diagnoses (diagnoses with a minimal number of components). This supports the common MBD approach of focusing first on diagnoses with lower cardinality.

Moreover, in most cases only a fraction of the set of minimal cardinality diagnoses is needed in order to reach a health state with a stable value. This result opens a great opportunity to develop DAs that determine online when to stop searching for more diagnoses [24]. A **third contribution** of this paper is by proposing such an online stopping condition for diagnosis algorithms. An empirical evaluation of this stopping condition shows that the quality of the system's health state is very close to the quality of the health state generated from all diagnoses.

A shorter version of this work was published in the proceedings of AAAI 2015 [25]. Beyond providing more detailed explanations of the previously proposed health state quality metrics, we also consider another health state quality metric

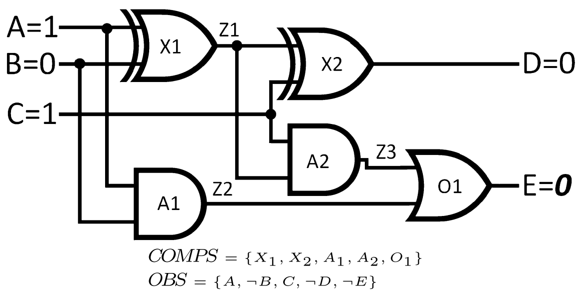

A=1
B=0
C=1
X1
Z1
X2
D=0
A2
Z3
O1
E=0
A1
Z2
$COMPS = \{X_1, X_2, A_1, A_2, O_1\}$
$OBS = \{A, \neg B, C, \neg D, \neg E\}$

Fig. 2. MBD: a full adder.

based on the wasted effort cost incurred when using this health state. An important contribution of this manuscript over the previous version is that we investigate how the quality of the health state is affected by different DAs. This investigation reveals interesting insights into the effectiveness of different DAs and the importance of finding minimal cardinality diagnoses. The final contribution of this paper, beyond the previous version, is by performing our experimental analysis over an additional domain of software diagnosis.

The paper is structured as follows. Section 2 provides a formal definition of model-based diagnosis, and lists the basic assumptions we make about the output of a DA. Section 3 defines the notion of a *health state*, describes how we compute it, and proves the correctness of this computation under certain conditions. Section 4 presents several measures that one can use to evaluate the accuracy of a health state. Section 5 provides an empirical evaluation, where Section 5.2 evaluates the number of diagnoses required to achieve an accurate-enough health state and Section 5.3 studies the impact of different DAs on the number of diagnoses required to achieve an accurate health state. Section 6 shows how a simple online stopping condition, based on the health state values, can be used to decide online when to stop search for more diagnoses. Section 7 discusses how health states can be computed for DAs that consider a strong fault model. Finally, Section 8 lists related work and Section 9 concludes and lists possible directions for future work.

2. Model based diagnosis

An MBD problem is specified as a triplet $\langle SD, COMPS, OBS\rangle$ where: SD is a model of the diagnosed system, $COMPS$ is a set of components, and OBS is an observation. SD takes into account that some components might be abnormal (faulty). This is specified by an unary predicate $h(\cdot)$, such that $h(c)$ and $\neg h(c)$ denote that component c is healthy or faulty, respectively. Denoting the correct behavior of c as a propositional formula, φ_c, SD is given formally as

$$SD = \bigwedge_{c \in COMPS} h(c) \Rightarrow \varphi_c$$

Namely, a healthy component follows its correct behavior. A diagnosis problem (DP) arises when the assumption that all components are healthy is inconsistent with the system description and observation [1,2]. System models can be classified into weak fault models (WFMs) and strong fault models (SFMs). In WFMs only the nominal behavior of every component is specified, while in SFMs additional information is available about how faulty components behave. For ease of presentation, we assume in this paper a WFM. In Section 7 we describe how the theoretical results in this paper generalize to SFM.

Fig. 2 shows an example of an MBD problem, where the normal behavior would give the output $E = 1$ but the observation had $E = 0$. See Appendix A for the formal system description and observations of this example. MBD algorithms try to find the actual faults – the faulty components that have caused the observed abnormal behavior of the diagnosed system. To this end, MBD algorithms often search for *diagnoses*, which are subsets of $COMPS$ that explain the observation if assumed faulty.

Definition 1 *(Diagnosis).* A set of components $\omega \subseteq COMPS$ is a *diagnosis* if the following term is consistent (i.e., it does not lead to a contradiction)

$$SD \wedge \bigwedge_{c \in \omega} \neg h(c) \;\wedge \bigwedge_{c \notin \omega} h(c) \;\wedge\; OBS$$

Ideally, a DA will return a single diagnosis that contains exactly the actual faults. We refer to this diagnosis as the *correct diagnosis*. Clearly, the correct diagnosis is a diagnosis according to Definition 1, as long as the system model is correct and observations are accurate. Thus, a DA that returns all diagnoses is complete, in the sense that it will return the correct diagnosis.

The key problem we address in this work is that for non-trivial systems, the number of diagnoses (as defined in Definition 1) is often much larger than one. Indeed, it is common that DAs return a set of diagnoses, denoted here as Ω, and in benchmark systems $|\Omega|$ can be several hundreds of thousands [14,12]. Note that such large number of diagnoses are not due to errors in the system description, but due to partial observability of the diagnosed system: if the input and output of every system component was observed, then finding the correct diagnosis would have been easy.

Finding all of the diagnoses can be very time consuming and it is not clear how to reason about large sets of diagnoses in a meaningful and effective way. A common approach to limit the set of diagnoses to find and reason about is to only consider diagnoses that answer some minimality criteria.

Definition 2 *(Minimal Diagnoses).* We say that a diagnosis ω is a *minimal diagnosis* if no proper subset $\omega' \subset \omega$ is a diagnosis, and that ω is a *minimal cardinality diagnosis* if no other diagnosis $\omega' \subseteq \textit{COMPS}$ exists such that $|\omega'| < |\omega|$.

For the MBD of Fig. 2, $\omega_1 = \{X_1, X_2\}$, $\omega_2 = \{O_1\}$, $\omega_3 = \{A_2\}$ are all the minimal diagnoses, and ω_2, ω_3 are all the minimal cardinality diagnoses. Some MBD algorithms search for subset minimal diagnoses [26–28] and other MBD algorithms search only for minimal cardinality diagnoses [14,29]. Finding a single minimal cardinality diagnosis is an NP-hard problem [30] and finding more than one minimal subset diagnosis is also NP-hard [31].

Even if limiting the output of an MBD algorithm to only return minimal cardinality diagnoses, the number of returned diagnoses can still be exponential in the size of the diagnosis cardinality (the number of components assumed to be faulty). This occurs in practice in standard MBD benchmarks [12,14]. The question of how to manage such a large set of output diagnoses still holds, even if limiting the set of diagnoses to only include minimal cardinality diagnoses.

2.1. Returning only the most probable diagnosis

Another principled way to limit the set of diagnoses returned is by only considering the diagnoses that are most likely to be the correct diagnosis. Indeed, some DAs return, in addition to a set of diagnoses Ω, a probability distribution p over Ω [32–35,21]. This $p(\omega \in \Omega)$ represents the probability that ω is the correct diagnosis given the observations (i.e., it is a posterior probability). Given such a probability function, one can focus the search for diagnoses towards more likely diagnosis [35,28] or even return only the most probable diagnosis.

While appealing, returning only the most probable diagnosis can be very misleading, as it ignores the information stored in all other diagnoses. Fig. 1 illustrates this. It depicts a hydraulic system in which components $C1, \ldots, C8$ are closed valves and component $C9$ is a tank. The observed system behavior is that oil flows into the system (depicted by the rectangle named “In”) and unexpectedly some oil leaks into the tank ($C9$), which is monitored by a sensor (depicted by the rectangle named “Out”). The set of subset minimal diagnoses[1] is

$$\Omega = \{\{C_1, C_2\}, \{C_3, C_4\}, \{C_3, C_5\}, \{C_3, C_6\}, \{C_3, C_7\}, \{C_3, C_8\}\}$$

Now, assume that the DA that we used returned also that the most probable diagnosis is $\{C1, C2\}$, having a probability of 0.2, and all other diagnoses have a probability of 0.16. However, all diagnoses except the most probable diagnosis assume that C_3 is faulty. The probability mass of all these diagnoses is 0.8 – much more than the probability of most likely diagnosis (0.2). Thus the most likely explanation is actually that C_3 has failed, along with one of the components C_4, C_5, C_6, C_7 and C_8. More generally, considering only the most probable diagnosis ignores all the information that exists in the other diagnoses, and that information can lead to hypotheses about the system components' health that together are more likely than the most probable diagnoses being correct. The *system health state* defined next provides a principled way to consider the entire set of diagnoses and their probabilities.

3. The health state of a system

Given a large set of diagnoses, a reasonable question that a human operator might ask is “what is the probability that a component C is faulty?” This is helpful, for example, to highlight to an operator which component should be manually checked first. In addition, being able to estimate the likelihood of each component being faulty is helpful in troubleshooting algorithms [10,36]. The health state defined below is exactly such a mapping of component to the probability that it is faulty.

Definition 3 *(Health State).* A health state is a function $H : COMPS \rightarrow [0, 1]$ such that for every component $C \in COMPS$:

$$H(C) = Pr(\neg h(C)|OBS)$$

Given a set of diagnoses Ω and a probability function over them p, we can derive a health state as follows

$$H(C) = \sum_{\omega \in \Omega} p(\omega) \cdot \mathbb{1}_{C \in \omega} \tag{1}$$

[1] The observant reader will notice that in this example we actually consider a strong fault-model, as we do not consider as a diagnosis the assumption that a closed valve was leaking if it was not given fluid as input. While an example with a weak fault-model is possible we find the example in Fig. 1 clear and easy to understand, and keep it for didactic purposes.

Table 1
Diagnoses and health state for Fig. 1.

	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	$H(C_i)$	$H^*(C_i)$
C_1	1	0	0	0	0	0	**0.20**	0.0
C_2	1	0	0	0	0	0	**0.20**	0.0
C_3	0	1	1	1	1	0	**0.80**	1.0
C_4	0	1	0	0	0	0	**0.16**	0.0
C_5	0	0	1	0	0	0	**0.16**	1.0
C_6	0	0	0	1	0	0	**0.16**	0.0
C_7	0	0	0	0	1	0	**0.16**	0.0
C_8	0	0	0	0	0	1	**0.16**	0.0
C_9	0	0	0	0	0	0	**0.00**	0.0
p	0.2	0.16	0.16	0.16	0.16	0.16		

where $\mathbb{1}_{C\in\omega}$ is the indicator function defined as:

$$\mathbb{1}_{C\in\omega} = \begin{cases} 1 & C \in \omega \\ 0 & \text{otherwise} \end{cases}$$

Theorem 1 states exactly the assumptions required for Equation (1) to hold.

Theorem 1. *Given a diagnosis problem $\langle SD, COMPS, OBS\rangle$ a set of diagnoses Ω and a diagnosis likelihood function $p : \Omega \rightarrow [0, 1]$, we can compute the corresponding health state using Equation (1) if all the following conditions hold:*

1. *Ω is the set of all diagnoses.*
2. *$p(\omega)$ is the probability that ω is correct given the observations, i.e.,*

$$p(\omega) = Pr(\bigwedge_{C'\in\omega} \neg h(C') \wedge \bigwedge_{C''\notin\omega} h(C'')|OBS)$$

Proof. Let $\Omega[C]$ be the set of diagnoses in Ω that contain C. Since we assume that Ω contains all the diagnoses, then if C is faulty it means that one of the diagnoses in Ω that assume it is faulty ($= \Omega[C]$) must be the correct diagnoses.

$$\begin{aligned} H(C) &= Pr(\neg h(C)|OBS) \\ &= Pr(\bigvee_{\omega\in\Omega[C]} \omega \text{ is correct}|OBS) \end{aligned}$$

Every diagnosis is an assumption over all the components in COMP (see Definition 1). Therefore, there is exactly one correct diagnosis in Ω. Thus, two diagnoses cannot both be true, and more generally the probability that a given set of diagnoses contain the correct diagnosis is the sum of their probabilities. In our context:

$$\begin{aligned} &Pr(\bigvee_{\omega\in\Omega[C]} \omega \text{ is correct}|OBS) \\ &= \sum_{\omega\in\Omega[C]} Pr(\omega \text{ is correct}|OBS) \\ &= \sum_{\omega\in\Omega[C]} p(\omega) = \sum_{\omega\in\Omega} p(\omega)\cdot \mathbb{1}_{C\in\omega} \end{aligned}$$

as required. □

Informally, Theorem 1 says that if Ω contains all diagnoses and the uncertainty over the true state of the world is accurately represented by p, then the health state generated by Equation (1) is accurate. Note that since only a single diagnosis is correct in a given state of the world, there is no notion of probability dependence between diagnoses. Generating a health state from a set of diagnoses Ω and p using Equation (1) is referred to in this paper as *generating a health state from Ω and p*.

We argue that presenting a human operator with a health state is more meaningful and helpful than a long list of diagnoses, even if the diagnoses are prioritized by their probabilities. For example, consider Table 1, which shows the diagnoses found for the system depicted in Fig. 1. Every column represents a diagnosis and every row represents a components, where a value of "1" in column i and row j means that component j is part of diagnosis i. The values in the row labeled "p" show the probabilities of each diagnosis, and the values in the column labeled "$H(C_i)$" show the value in the health state for the corresponding row. So, Table 1 shows that the DA returned five diagnoses $\{C_3, C_4\}, \{C_3, C_5\}, \{C_3, C_6\}, \{C_3, C_7\}, \{C_3, C_8\}$ with a probability of 0.16 each, and another diagnosis $\{C_1, C_2\}$ with a probability of 0.2. The most probable diagnosis is $\{C_1, C_2\}$, having a probability of 0.2. The health state, however, would point at C_3 as the component that is most likely to be faulty, having the highest value in the health state: $H(C_3) = 0.8$. This example demonstrates how a health state provides a more useful output than the most probable diagnosis.

While a health state is an informative aggregation of a given set of diagnoses, some information contained in the set of diagnoses it was generated from, is lost. This lost information is the dependencies between the different components. For example, consider again the diagnoses in Table 1. The component C_1 is only a member of a single diagnosis $\{C_1, C_2\}$. Thus, repairing only C_1 without repairing C_2 is not likely to fix the system. This relation between C_1 and C_2 is lost in the aggregated perspective of the health state. Note that this does not mean that the values in the health state are incorrect – component C_1 is still faulty with 20%. Automated troubleshooting algorithms [16,1,15] might make use of such additional relation information and might thus prefer as input a list of diagnoses over a health state.

Finding a compact representation of a large set of diagnoses that does not lose some information is a hard problem that may not be solvable. The health state is proposed as a good compromise – it is compact ($O(|COMPS|)$) and enables answering the basic diagnostic question – what is the probability of a given component to be faulty. Due to complexity reasons, DAs often do not return all consistent diagnoses. We expect that the quality of a health state depends on the diagnoses used to generate it. To study this relation, we require exact methods to evaluate the quality of a given health state. Next, we propose several such health state evaluation metrics.

4. Evaluating a health state

An ideal health state indicates exactly which components are faulty and which components are healthy – setting $H(C_i) = 1$ for every C_i that is faulty and setting $H(C_j) = 0$ for every C_j that is not. We refer to such a health state as the *offline optimal health state*, denoted it by H^*. Formally:

$$H^*(C_i) = \begin{cases} 1 & C_i \text{ is faulty} \\ 0 & \text{otherwise} \end{cases}$$

Having such an ideal health state is not practical, since DAs commonly return more than a single diagnoses. In such cases, the resulting health state will be different from the offline optimal health state, reflecting the uncertainty over which diagnosis is correct. The question we raise is how accurate is a given health state. This question becomes even more important when considering that computational limitations often prevent MBD algorithms from returning all the diagnoses for a given diagnosis problem. In addition, the diagnosis probability function p is often a (usually rough) approximation. Thus, the actual question is how accurate is a health state generated from a given Ω and p. A complementing question is which and how many diagnoses should a DA return in order to generate a high quality health state.

To answer these questions, a metric is needed to measure the quality of a generated health state. The term *quality* in general is often ambiguous. In our context, we use the term quality of a health state H to refer to how much different is H from the offline optimal health state H^*. This follows the reasoning that the offline optimal health state is perfect, and any deviation from it constitutes lower quality. Next, we present three such metrics for quantifying such a loss of quality, i.e., for quantifying the differences between H and H^*. Each metric has pros and cons, and we do not claim that any quality metric by itself is superior to the others.

4.1. The Euclidean distance metric

The first metric we propose for evaluating the quality of a health state is to consider every health state as a vector, and measure the accuracy of a health state H by its distance from H^*. There are several ways to compute distances between vectors, and in this work we use a simple Euclidean distance for this purpose, i.e.,

$$\sqrt{\sum_{C_i \in COMPS} (H(C_i) - H^*(C_i))^2}$$

To demonstrate the Euclidean distance metric, consider our running example from Table 1 and assume that the faulty components (which are unknown to the diagnoser) are $\{C_3, C_5\}$. The corresponding offline optimal health state is given in Table 1 in the column headed by $H^*(C_i)$. The Euclidean metric for this case is $\sqrt{2 \cdot (0.2-0)^2 + (0.8-1)^2 + (0.16-1)^2 + 4 \cdot (0.16-0)^2} = \mathbf{0.93}$. In general, a better health state according to this quality metric is one that has lower Euclidean distance, and zero Euclidean distance indicates that Ω contains a single diagnosis consisting of exactly the faulty components (i.e., the correct diagnosis).

4.2. The area under the curve (AUC) metric

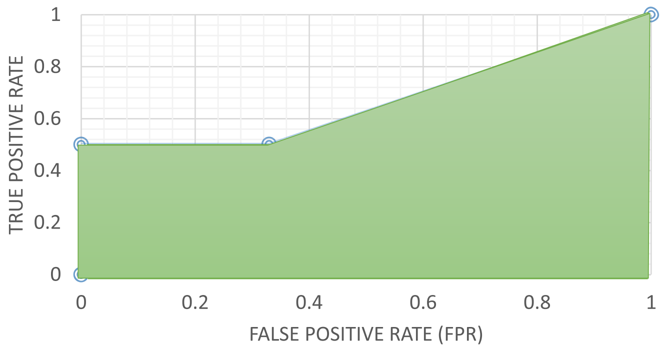
The second health state metric we propose is based on viewing the diagnosis problem as a classification problem and the health state as a parametric model of a classifier. Prior work have already formulated the diagnosis problem as a multi-label classification problem [37]. An optimal classifier labels correctly components as faulty or normal. They used a

Table 2
Definitions.

True Positive (TP)	faulty components correctly diagnosed as faulty
True Negative (TN)	healthy components correctly diagnosed as healthy
False Positive (FP)	healthy components incorrectly diagnosed as faulty
False Negative (FN)	faulty components incorrectly diagnosed as healthy

Threshold	Faulty components (above threshold)	TPR	FPR
$T > 0.8$	$\{\}$	0	0
$0.8 \geq T > 0.2$	$\{C_3\}$	0.5	0
$0.2 \geq T > 0.16$	$\{C_1, C_2, C_3\}$	0.5	0.33
$T \leq 0.16$	$\{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8\}$	1	1

(a) TPR and FPR for ranges of thresholds values. Recall that the correct diagnosis is $\{C_3, C_5\}$. Thus, assuming that only C_3 if faulty yields a TPR of only 0.5, since half of the faulty components are detected (missing C_5). Similarly, assuming that C_1, C_2, and C_3 are faulty result in a FPR of $\frac{1}{3}$, since only one of the components assumed faulty are actually faulty.



(b) The ROC curve. The area under the curve (AUC) is marked in green.

Fig. 3. AUC computation the scenario in Fig. 1 and Table 1.

Machine Learning algorithm to learn a classifier and then model-based reasoning to refine the classifications suggested by that classifier (see Section 8 for a discussion on that work). We borrow from that prior work the idea of a diagnosis problem as a classification problem, and consider the health state as a parametric classifier, as follows. A component C is classified as faulty if $H(C) \geq T$, where T is a threshold parameter that can be tuned to control the sensitivity and specificity of the resulting classifier. Indeed, many classifiers accept as input a similar threshold parameter exactly for this reason. In our context, Setting $T = 0$ results in the classifier classifying all components as faulty, while $T = 1$ results in a classifier that only classifies components as faulty if they are faulty for certainty according to the health state (these are the components that are members of all the diagnoses in Ω). This classifier-oriented perspective of a health state corresponds to a natural repair policy: repair every component whose health state value is above the given threshold.

A common metric for evaluating parametric classifiers is the AUC metric, and this is the second health state metric we propose. For completeness, we provide next a brief explanation of how the AUC metric is computed. For a given T, H, and H^*, it is possible to count the true/false positives/negatives as defined in Table 2. These definitions are used to compute the true positive rate (TPR) $TPR = TP/(TP + FN)$ and the false positive rate $FPR = FP/(FP + TN)$. Intuitively, TPR represents proportion of positives (in our case – faulty components) that are correctly identified as such, and FPR represents the probability that an instance is classified incorrectly as a positive (in our case – the probability that a component assumed faulty is actually healthy). TPR is also known as sensitivity or recall, and the complement of FPR – the true negative rate – is also known as specificity. By varying T the Receiver Operating Characteristic (ROC) curve can be drawn, which is the curve resulting from plotting the TPR against the FPR for a range of T values. A common measure for evaluating classifiers with such a threshold parameter is by measuring the area under the ROC curve. The AUC metric is exactly this area under the ROC curve. AUC values range from zero to one, where the best AUC value is one. An AUC value can be calculated by using a trapezoidal integration (approximation) or by an integral. We chose to use the trapezoidal integration: $AUC = \sum_{i\in\{0,0.1,\ldots,0.9\}}((TPR_i + TPR_{i+0.1})/2) \cdot (FPR_{i+0.1} - FPR_i)$, where i represents the varied T (in our case it varies in interval of 0.1).

Back to the running example, depicted in Table 1. Table (a) in Fig. 3 presents the TPR and the FPR for some thresholds, and Fig. 3b presents the appropriate ROC curve. The AUC in this case is 0.6675. In general, according to this quality metric higher AUC values represents a more accurate health state and the optimal health state will have an AUC of one.

4.3. The wasted effort metric

Another possible health state quality metric is based on the *wasted effort* metric proposed by Abreu et al. [20,21] for evaluating the output of a DA. The wasted effort metric **estimates the effort wasted in repairing healthy components**, given the diagnoses and probabilities returned by the DA. This metric is computed by assuming that components are repaired in the following order. First, the diagnoses returned by the DA are sorted according to their likelihoods. Then, we assume that components in the first diagnosis are repaired. Afterward, we assume that all the components in the second diagnosis are repaired. This continues until all faulty components are repaired. The wasted effort metric is the number of healthy components that are repaired in this process divided by the total number of healthy component. Thus, a wasted effort of zero is optimal, while a wasted effort of one is the worst outcome, meaning that all healthy components were repaired before repairing all the faulty ones.

Consider our running example (from Fig. 2 and Table 1), where C_3 and C_5 are faulty. The first diagnoses is $\{C_1, C_2\}$. Since both components are healthy and there are six healthy components, repairing C_1 and C_2 adds $\frac{2}{6}$ to the wasted cost metric. The subsequent diagnoses all have equal probabilities, so assume that they are ordered lexicographically, i.e., the first diagnosis to be repaired is $\{C_3, C_4\}$, then $\{C_3, C_5\}$ and so on. Repairing $\{C_3, C_4\}$ adds $\frac{1}{6}$ to the wasted effort, as C_3 is indeed faulty. Finally, after $\{C_3, C_5\}$ is repaired all faulty components are repaired. Thus, the total wasted effort is $\frac{3}{6}$.

It is possible to adapt the wasted effort metric, which is used to evaluate diagnoses and probabilities, to evaluate health states. Instead of repairing the components in the diagnoses ordered by their probabilities, the health state wasted effort metric considers repairing components order by their probabilities in the health state. Thus, in our running example and assuming the same lexicographical tie-breaking, the first component repaired is C_3 (having $H(C_3) = 0.8$), having the highest probability in the health state. Then C_1 and C_2 would be repaired (having $H(C_1) = H(C_2) = 0.2$). Finally, C_4 and C_5 will be repaired (having $H(C_4) = H(C_5) = 0.16$). The resulting total wasted effort is $\frac{3}{6}$, since C_1, C_2, and C_4 are all healthy.

Formally, we define the wasted effort metric as follows. Let F be the set of faulty components and assume that the components are sorted according to their health state values, i.e., C_1 has the highest $H(\cdot)$ value, C_2 has the second highest $H(\cdot)$ value, etc. Now, let $\mathcal{C}_i$ be the set of components $\{C_1, \ldots C_i\}$, i.e., the i components that have the highest $H(\cdot)$ values. We define the *wasted effort metric* of a health state H as follows:

$$\min_{1 \le i \le |COMPS|} \frac{|F \setminus \mathcal{C}_i|}{|COMPS \setminus F|} \tag{2}$$

The wasted effort metric is presented since it is a natural adaptation of the existing wasted effort metric for a set of diagnoses and their probabilities. However, note that the wasted effort does not necessarily provide an accurate estimate of the actual effort wasted when repairing the system. First, troubleshooting algorithms for system repair often employ various costly diagnostic actions, such as setting probes and performing addition tests to get more observations. Second, intelligent system repair algorithms do not necessarily repair components in the order assumed by the wasted effort metric (in order of the probabilities in the health state). Other considerations also affect the choices made, such as cost of repair and uncertainty of repair outcomes [36,38,39]. For example, if two components have almost the same probability of being faulty according to the health state, but one is cheaper to repair than the other, then an intelligent system may choose to repair that component first, even if its probability is slightly smaller than the other component. Thus, the wasted effort metric is mostly given here for completeness and compatibility with previous work. As we show empirically in the next section, there is a strong correlation between this metric and the Euclidean and AUC metrics.

All the proposed metrics can be used to evaluate health states. Using these metrics, we can study empirically the relation between the diagnoses returned by a DA and the quality of the resulting health state. As we show next, the resulting analysis provides key insights into how to design DAs to find high quality health states.

5. Empirical analysis

Next, we study experimentally the relation between the diagnoses returned by a DA and the quality of the health state generated from them. In particular, we (1) examine the amount of diagnoses required to produce a high-quality health state (Section 5.2), (2) compare the health states generated by using different DAs (Section 5.3), and (3) propose a stopping criterion that can be used online by a DA to halt the search for more diagnoses (Section 6).

5.1. Experimental settings

The experiments were performed on two domains: diagnosis of boolean circuits, and diagnosis of software bugs. Below we describe the details of our experiments in these domains, including the benchmark problems used, the DAs evaluated, and how diagnosis likelihood was computed.

5.1.1. Diagnosis of Boolean circuits

In this domain, the diagnosed system is a Boolean circuit, such as those used to design electrical circuits. The components in this domain are logic gates and the goal is to diagnose which gates are faulty. Evaluating DAs on Boolean circuits

Table 3
A description of the used systems and observations.

System	|COMPS|	In	Out	|Observations|	|Ω|	Max minc
74181	65	14	8	49	299	5
74182	19	9	5	50	145	7
74283	36	9	5	30	347	4
c1355	546	41	32	27	304	3
c432	160	36	7	35	266	4
c499	202	41	32	24	146	3
c880	383	60	26	21	140	3

is standard in the literature, and in our evaluation we used two well-established standard Boolean circuit benchmarks: ISCAS-85 [13] and 74XXX [22]. Table 3 presents some basic information about these benchmarks: the systems' names (the column labeled "System"), number of components and number of inputs and outputs (columns labeled "$|COMPS|$", "In", and "Out"). Due to computational complexity we did not run experiments on larger systems in ISCAS-85. The observations for the systems were drawn from Feldman's set of observations [40]. Note that observations for which the minimal cardinality diagnoses have a high cardinality correspond to scenarios that are more difficult computationally for most DAs, and, importantly, correspond to having a larger number of possible diagnoses. Since we study the question of how many diagnoses are needed, we experimented on a representative subset of Feldman's set of observations, choosing a random subset of observations per minimal cardinality. For each observation, we chose randomly one of the diagnoses to serve as the correct diagnosis. The rest of the columns in Table 3 include information about the observations we used: the number of observations (the column labeled "$|Observations|$"), the average number of found diagnoses per observation (the column labeled "$|\Omega|$"), and the highest minimal cardinality value found among the different observations for each system (the column labeled "Max minc").

The DA we used was a forward search algorithm, first checking for single fault diagnoses, then all double faults, etc. Thus, diagnoses were returned in order of their cardinality, finding first all minimal cardinality diagnoses, then all subset-minimal diagnoses with cardinality higher than the minimal cardinality by one, and so on. To avoid memory limitations, we implemented this DA using an iterative deepening scheme. This DA was halted when all subset-minimal diagnoses were found or after 15 minutes runtime limit. While more sophisticated DAs exists, the focus of this work is not a new DA but to examine the outputted diagnoses. Moreover, what affects the number of diagnoses needed to produce a high quality health state is only the order in which the diagnoses are returned, not the runtime required to get it. Many DAs search diagnoses in an order similar to a forward search algorithm, starting by the smallest cardinality to largest. In Section 5.3 we discuss the impact of different diagnoses orderings on the number of diagnoses required.

To compute the likelihood of a diagnosis ω, we followed a standard Bayesian approach:

$$p(\omega) = Pr(\omega|Obs) = \frac{Pr(Obs|\omega) \cdot Pr(\omega)}{Pr(Obs)} \tag{3}$$

Since all the diagnoses our DA returned are consistent and subset-minimal, we have that $Pr(Obs|\omega) = 1$. The value $Pr(\omega)$ represents the prior probability that the components in ω are faulty, without any observations. To compute $Pr(\omega)$, we make two assumptions: (1) that we know for every component C the prior probability $Pr(C)$ that it will fail, and (2) that components fail independently of each other. Under these assumptions, $Pr(\omega) = \prod_{C\in\omega} Pr(C)$.[2] In reality, the component priors $Pr(C)$ can be given by the component manufacturer or approximated by applying Machine Learning methods [41]. In our experiments we set all $Pr(C)$ values to a constant 0.01. The assumption that faults occur independently, while not realistic in many cases, is a common assumption in MBD [1,33,28]. Finally, $Pr(Obs)$ is a normalizing factor, and thus, if Ω is the set of all diagnoses then we have

$$p(\omega) = \frac{\prod\limits_{C_i\in\omega} Pr(C_i)}{\sum\limits_{\omega'\in\Omega}\prod\limits_{C'_i\in\omega'} Pr(C'_i)} \tag{4}$$

The focus of this work is not in proposing a novel way to compute the posterior probability that a diagnosis is correct. While commonly assumed in the academic literature, the assumptions we make in the computation of Equation (4) – known priors and independent failures – can be incorrect in practice, and there has been several prior work on taking into consideration more complex settings where these assumptions do not hold [34,42]. Implementing and evaluating different methods for computing diagnosis probabilities is beyond the scope of this paper. Importantly, all the theoretical discussions

[2] Note that in a WFM there is no assumption about the behavior of faulty components. Thus, any component that is not in the diagnosis can be either faulty or healthy. Thus, we computed the prior $Pr(\omega)$ as $\prod_{C\in\omega} Pr(C)$ and not $\prod_{C\in\omega} Pr(C) \prod_{C'\notin\omega}(1 - Pr(C'))$.

Table 4
A comparison of the two experimental settings described in Sections 5.1.1 and 5.1.2.

Setup #1 (Sec. 5.1.1)	Setup #2 (Sec. 5.1.2)
Boolean Circuits	Software Diagnosis
Forward search	DA-Barinel
A model and a single observation	Multiple observations (tests traces)
Non-intermittent faults	Intermittent faults
Uniform priors	Priors generated using Machine Learning
Complete set of diagnoses	Incomplete set of diagnoses

in this paper are agnostic to how the diagnosis probabilities are obtained, and directly apply also to DAs that consider dependencies between component failures when computing $p(\omega)$.

5.1.2. *Software diagnosis*

The second domain we used is open-source software. In this domain, the software classes are the system components and the task is to find the classes that contain faults – software bugs – that caused some observed tests to fail.

As a benchmark we used the open-source project Apache Ant [23], which is a popular build framework for Java. Apache Ant has 1,195 files and 10,830 methods. Over 1,176 bugs were found in Apache Ant and reported in the Bugzilla issue tracking system (www.bugzilla.org). Following prior work on software diagnosis [10,41], we ran experiments on this dataset as follows. In every experiment we chose a subset of the reported bugs, created observations by running 40 tests selected randomly from the set of relevant JUnit tests. The passed and failed tests are given as input to a DA to find diagnoses (see detailed about the DA we used below). This type of experiment was performed 136 times.

The DA we used is a data-augmented version of Barinel [41], referred to hereonafter as DA-Barinel. Barinel is a recent successful DA specifically designed for software diagnosis [20,43]. It is significantly different from classical MBD algorithms in that it does not require a logical model of the correct functionality of every software component in the system. Instead, the traces of the observed tests are considered: Barinel treats every trace of a failed test as a *conflict* – a set of components that contain at least one faulty component, and returns hitting sets of theses conflicts as diagnoses. Barinel uses a fast but incomplete hitting set algorithm called STACATTO [44] that allows Barinel to scale well to large systems. An integral part of Barinel is how it computes diagnoses likelihoods. Briefly, they also use a Bayesian approach to compute diagnoses likelihoods, but since they handle multiple observations their computation also considers fault intermittency by incorporating a "goodness" function that estimates the probability that a faulty component will cause a test to fail. See [21] for details.

DA-Barinel [41], which is the DA we used in our experiments, improves Barinel by using a data-driven approach to find more informed priors, which are then used by Barinel's Bayesian computation of diagnosis likelihood. In more details, these priors were generated using Machine Learning techniques on historical data of code versions and prior bugs. The learning process is done offline, before observing any failed test. In general, DA-Barinel is part of the LDP (Learn, Diagnose, and Plan) paradigm for software testing and debugging [41].

As mentioned above, Barinel and DA-Barinel use a fast but incomplete hitting set algorithm called STACATTO [44] to scale to large systems. Thus it is possible that the correct diagnosis will not be included in the list of diagnoses they return. In practice, DA-Barinel found the correct diagnoses in 83 experiments out of the 136 experiments we performed. In the results below, we show experimental results averaged over *all* the 136 experiments (denoted as "All" in the tables below), including cases where the correct diagnoses was not found. To provide additional insights, we also show experimental results that are averaged only over the 83 experiments in which DA-Barinel found the correct diagnoses (denoted as "Complete" in the tables below).

This experimental setting described above – software diagnosis using DA-Barinel – is significantly different from the standard Boolean circuits domain that was described in Section 5.1.1. Table 4 summarizes the differences between the experimental setups. First, the problem domains are different. Boolean circuits are the de-facto standard in evaluating MBD algorithms, and software diagnosis is an important application of automated diagnosis. Second, the DA used for Boolean circuit was a standard forward search DA while for the second setting we used a specialized DA for software diagnosis. Third, in the first setting the input to the DA is a system description (a model) and a single observation, while in the second setting the input are a set of observations – software tests with their traces and outcome (pass or fail). Forth, in the first setting the computation of diagnosis likelihoods relied on common simplifying assumption such as that all component have the same prior probability to fail. In contrast, the diagnosis likelihood computation in the second setting was evaluated in practice by prior work, and includes learning a fault prediction model to generate specific priors for every component. Lastly, the DA used for the first setting is complete, in the sense that the set of diagnoses it returns contains the correct diagnosis, while the DA used for the second setting is not.

5.2. *Amount of diagnoses required*

The main question we ask in this analysis is **how many diagnoses are needed to produce a high quality health state**. As a preliminary, we observed that after a certain number of diagnoses the quality of the health state converges to some stable value, after which adding more diagnoses results in negligible or non-existent change in health state quality. To demonstrate

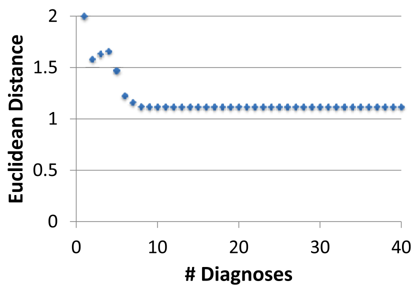


Fig. 4. An observation instance of system 74182, where adding more diagnoses improves the health state quality.

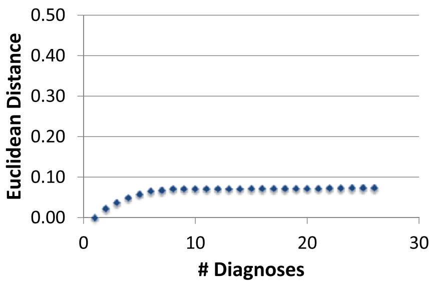


Fig. 5. An observation instance of system 74182, where adding more diagnoses degrades the health state quality.

this, consider Figs. 4 and 5 that show the health state quality for two observations of the 74182 system as a function of the number of diagnoses found. Quality (the y-axis) was measured here using the Euclidean distance metric, and thus lower values correspond to health state of a higher quality.

Indeed, in both Figures after a certain number of diagnoses the quality of the health state converges to a stable value. For example, in Fig. 4 the health state quality converges to approximately 1.16 after 7 diagnoses returned by the DA. These two specific observations are representative examples, as we observed similar convergence in most of the problem instances we experimented with. This raises the question of *how many diagnoses are needed for the health state to converge*.

A complementing question is *what is the quality of the converged health state*. As shown in Fig. 5, the health state after convergence can be worse than the health state generated from a smaller subset of all diagnoses. There (Fig. 5), the health state value starts at the optimal value (zero distance from H^*) and degrades to 0.07 where it remains stable. This is caused because the correct diagnosis happened to be the first diagnosis that was found by the DA.

5.2.1. Measuring convergence

To answer the first question – how many diagnoses are needed for the health state to converge – we first define when a health state is considered to have converged. Let Ω_i be the set of the first i diagnoses returned by the DA, and let H_i to be the health state generated from the diagnoses in Ω_i. For instance, H_2 is computed based on a set containing the first two diagnoses found by the DA. We say that the health state has *converged* after i diagnoses if H_i is *stable*, where a stable health state is defined as follows.

Definition 4 *(ϵ-Stable Health State).* A health state H_i is said to be ϵ-stable if for every $j > i$ it holds that $|eval(H_j) - eval(H_i)| \leq \epsilon$, where $\epsilon \geq 0$ is a parameter and $eval(H_i)$ is the quality of a health state H_i according to the given quality metric.

Note that for the AUC metric ϵ cannot be greater than one, while for the Euclidean distance it should not be larger than $\sqrt{|COMPS|}$.

Now, we can ask the question of how many diagnoses are required to obtain an ϵ-stable health state. This is called the **convergence rate** of an observation. More formally, the convergence rate is the minimal $i \in [1, |\Omega|]$ for which H_i is ϵ-stable. In the results below we normalized the convergence rate by dividing them by $|\Omega|$. As an example, Table 5 lists the health state obtained for every Ω_i for our running example from Fig. 1 and Table 1. The right-most column shows the quality of every H_i as measured by the Euclidean distance. For $\epsilon = 0.1$ the convergence rate is $\frac{3}{6}$, since $|eval(H_3) - eval(H_i)| < 0.1$ for every $i > 3$, and there are 6 diagnoses. For $\epsilon = 0.2$, two diagnoses sufficed to reach convergence, as H_2 is already 0.2-stable (as defined in Definition 4), and thus the convergence rate for $\epsilon = 0.2$ is $\frac{2}{6}$.

5.2.2. Convergence rate results

We present first the result for the Boolean circuits experiments. Tables 6 and 7 show the average convergence rate and the wasted effort cost (in brackets) for AUC and Euclidean distance metrics, respectively, for a range of ϵ values. Each

Table 5
Health state for each Ω_i and its quality, measured by the Euclidean distance metric.

	H(C_1)	H(C_2)	H(C_3)	H(C_4)	H(C_5)	H(C_6)	H(C_7)	H(C_8)	eval(H_i)
Ω_1	0.20	0.20	0.00	0.00	0.00	0.00	0.00	0.00	1.44
Ω_2	0.20	0.20	0.16	0.16	0.00	0.00	0.00	0.00	1.35
Ω_3	0.20	0.20	0.16	0.16	0.16	0.00	0.00	0.00	1.23
Ω_4	0.20	0.20	0.16	0.16	0.16	0.16	0.00	0.00	1.24
Ω_5	0.20	0.20	0.16	0.16	0.16	0.16	0.16	0.00	1.25
Ω_6	0.20	0.20	0.16	0.16	0.16	0.16	0.16	0.16	1.26

Table 6
Boolean circuits. Convergence rate and wasted effort cost for the AUC metric.

ϵ	74181	74182	74283	c1355	c432	c499	c880
0 (ALL)	1.00(0.26)	1.00(0.20)	1.00(0.18)	1.00(0.24)	1.00(0.15)	1 .00(0.01)	1.00(0.15)
0.1	0.17(0.26)	0.48(0.21)	0.15(0.19)	0.30(0.26)	0.21(0.18)	0.17(0.01)	0.29(0.22)
0.2	0.06(0.44)	0.10(0.31)	0.04(0.26)	0.27(0.29)	0.14(0.24)	0.13(0.22)	0.24(0.28)
0.3	0.02(0.55)	0.04(0.40)	0.02(0.36)	0.14(0.44)	0.05(0.31)	0.05(0.40)	0.03(0.49)
0.4	0.01(0.58)	0.03(0.44)	0.01(0.36)	0.14(0.44)	0.04(0.34)	0.05(0.43)	0.02(0.49)
0.5	0.01(0.65)	0.02(0.49)	0.00(0.44)	0.01(0.49)	0.01(0.38)	0.02(0.47)	0.01(0.64)

Table 7
Boolean circuits. Convergence rate and wasted effort cost for the Euclidean metric.

ϵ	74181	74182	74283	c1355	c432	c499	c880
0 (ALL)	1.00(0.26)	1.00(0.20)	1.00(0.18)	1.00(0.24)	1.00(0.15)	1.00(0.01)	1.00(0.15)
0.1	0.01(0.63)	0.13(0.38)	0.01(0.44)	0.01(0.55)	0.01(0.38)	0.01(0.52)	0.01(0.64)
0.2	0.01(0.65)	0.03(0.49)	0.00(0.45)	0.01(0.55)	0.01(0.38)	0.01(0.52)	0.01(0.64)
0.3	0.01(0.65)	0.02(0.50)	0.00(0.44)	0.01(0.55)	0.01(0.38)	0.01(0.52)	0.01(0.64)

Table 8
Software diagnosis. Convergence and wasted effort results for the AUC metric.

ϵ	0 (All)	0.1	0.2	0.3	0.4	0.5
All	1.00(0.05)	0.14(0.06)	0.14(0.06)	0.09(0.06)	0.09(0.06)	0.08(0.06)
Complete	1.00(0.00)	0.06(0.00)	0.06(0.00)	0.06(0.00)	0.06(0.00)	0.05(0.00)

column represents the results for a set of observation of a single system (see Section 5.1 for details on each system). The row $\epsilon = 0$ serves as a baseline, as convergence for $\epsilon = 0$ is only reached once all diagnoses are found.

The first clear trend we observe in these results is that it is possible to converge to a ϵ-stable health state without finding all diagnoses even for $\epsilon = 0.1$. For example, consider the convergence rate results when using the AUC metric for system 74181. Only 17% of the diagnoses are needed to generate a health state that is at most 0.1 far from the health state generated from all diagnoses. This suggests that **intelligent DAs can halt early without significant loss in health state quality**.

A second trivial trend is that increasing ϵ values decreases the convergence rate for both evaluation metrics. This flows directly from the definition of convergence rate. Comparing the convergence rate of the two quality metrics (AUC and Euclidean) suggests that the AUC metric provides more diverse results. For example, the convergence rate of the Euclidean distance metric quickly drops to 0.01 with virtually no effect of the ϵ value beyond $\epsilon = 0.3$. By contrast, the convergence rate of the AUC metric ranges from 0.15 to 0.48, suggesting that AUC metric provides an evaluation metric that is easier to tune. This can be explained by the fact that AUC is normalized between 0 and 1, while Euclidean distance can range between 0 and $|COMPS|$. Normalizing the Euclidean distance metric to the range $[0, 1]$ is possible, but that would bias health states with more components to seem more accurate.

The same trends observed in Tables 6 and 7 were also observed in the software diagnosis experimental results, which are shown in Table 8. The evaluation metric used is AUC. The columns are the ϵ values. The "All" row shows the results for all instances and the "Complete" row shows the results of only the instances where the DA found the correct diagnosis (the column marked "Complete"). The main trend – reaching convergence with a small fraction of all diagnoses – is clearly observed. For example, only 14% of the diagnoses were needed to reach a health state that is 0.1-stable. As could be expected, fewer diagnoses are needed to reach convergence in the cases where the DA actually returned the correct diagnoses. For example, only 6% of the diagnoses were needed to generate a 0.1-stable health state in the "Complete" experiment set (compared to 14% for the "All" experiment set). This provides a partial support that fewer diagnoses are needed to reach convergence for stronger DA. See more on this in Section 5.3.

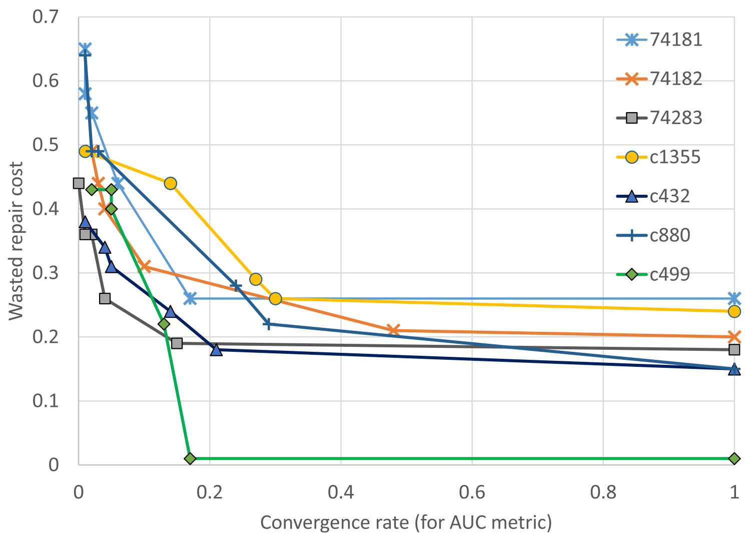


Fig. 6. Plotting the wasted effort costs for the convergence rate values obtained by using AUC and different values of ϵ. The clear trend is that increased convergence rate results in lower wasted efforts.

5.2.3. *The quality of ϵ-stable health states*

So far, we have examined the question of how many diagnoses are needed to reach a ϵ-stable health state. Here, we ask **what is the quality of the converged health state**, i.e., we inspect the actual quality of ϵ-stable health states. To this end, we considered the three quality metrics described in Section 4: wasted effort, Euclidean, and AUC.

First, we analyzed the wasted effort metric. Fig. 6 shows the relation between the average wasted effort (y-axis) and the convergence rate (x-axis), for the Boolean circuits domain. The convergence rate was computed using the AUC metric for different values of ϵ. The relation is clear: the wasted effort decreases when the convergence rate increases. This was also verified by computing Pearson correlation coefficient, showing a strong negative correlation between the wasted effort and convergence rate, with $r = 0.63$, $n = 42$, and $p < 0.001$. This correlation indicates that health states generated with more diagnoses tend to have a lower wasted effort value.

Now, consider the wasted effort values of the converged health state for different values of ϵ. These results are given in brackets in Tables 6, 7, and 8. It is clear from the results in all the tables that the wasted effort cost grows with ϵ. This follows from the correlation of wasted effort and convergence rate (shown in Fig. 6): converging to an ϵ-stable health state for smaller ϵ requires a more diagnoses, and thus the health state is more reliable and, on average, the wasted effort cost decreases.

Considering the wasted effort values provides an additional reason to prefer AUC over Euclidean distance. For most systems, the gap between the wasted effort cost of $\epsilon = 0$ and $\epsilon = 0.1$ is high. By contrast, the gap between similar ϵ values for AUC is more modest. For example, consider the 74283 system. The wasted effort cost for the health state generated from all diagnoses (the $\epsilon = 0$ line) is 0.18. Now consider the health state for $\epsilon = 0.1$. The health state generated from the diagnoses found until convergence according to the Euclidean distance has a wasted effort cost of 0.44 – much worse than the 0.18 for $\epsilon = 0$. By contrast health state generated for the AUC metric has a wasted effort cost of 0.19 – almost the same as the wasted effort cost for $\epsilon = 0$. This further supports AUC as a more flexible metric. Moreover, note that this health state (that resulted in 0.19 wasted effort cost) was generated from only 15% of the diagnoses. This result strengthens the conclusion that intelligent DAs can halt early and obtain a useful health state.

This conclusion is also supported by the software diagnosis results (shown in Table 8). When considering the “All” results, the wasted effort having all diagnoses is 0.05, while it is only increased by 0.01 even for $\epsilon = 0.5$. This is especially surprising, since the convergence rate for $\epsilon = 0.5$ is 0.08, i.e., only 8% of the diagnoses are needed to create a health state that has only 0.01 more wasted effort cost than the health state generated by all diagnoses. Moreover, for the “Complete” version of the experiments, the wasted cost was 0 even for $\epsilon = 0.5$.[3] This means that even by finding only 5% of the diagnoses we do not loss any information related to the diagnosis, in this setting.

In Section 4.3 we argued against the wasted effort cost metric (as adapted for evaluating health states) because its computation assumes a very rigid repair policy. Nonetheless, we provided the wasted efforts results above since this metric is based on a known metric for evaluating a set of diagnoses [20,21]. Next, we analyzed health state quality using the other health state quality metrics we proposed: AUC and Euclidean distance.

[3] Actually the values of the wasted cost where a little bit above 0, but since we rounded the values in a precision of two decimal places, all the values were rounded to 0.

Table 9
AUC values at convergence (higher is better).

ϵ	Boolean circuits (ISCAS-85)							Software diagnosis	
	74181	74182	74283	c1355	c432	c499	c880	Complete	All
Oracle	0.92	0.92	0.91	0.86	0.93	0.99	0.87	0.99	0.71
0 (ALL)	0.82	0.90	0.84	0.80	0.82	0.95	0.84	0.95	0.70
0.1	0.82	0.88	0.86	0.80	0.82	0.95	0.84	0.95	0.70
0.2	0.78	0.83	0.83	0.76	0.80	0.89	0.81	0.95	0.70
0.3	0.76	0.76	0.76	0.70	0.78	0.84	0.73	0.95	0.67
0.4	0.73	0.72	0.74	0.69	0.76	0.81	0.72	0.95	0.67

Table 10
Euclidean distance values at convergence (lower is better).

ϵ	Boolean circuits (ISCAS-85)							Software diagnosis	
	74181	74182	74283	c1355	c432	c499	c880	Complete	All
Oracle	0.14	0.22	0.18	0.05	0.08	0.05	0.06	0	0.03
0 (ALL)	0.17	0.23	0.21	0.06	0.10	0.08	0.06	0.01	0.03
0.1	0.21	0.28	0.27	0.07	0.11	0.09	0.09	0.00	0.03
0.2	0.21	0.34	0.29	0.07	0.11	0.09	0.09	0.00	0.03
0.3	0.21	0.36	0.29	0.07	0.11	0.09	0.09	0.00	0.03
0.4	0.21	0.36	0.29	0.07	0.11	0.09	0.09	0.00	0.03

Tables 9 and 10 show the average AUC and Euclidean distance values, respectively, for a range of ϵ values. For example, the value 0.82 for system 74181 and $\epsilon = 0.1$ in Table 9 is the AUC value of the health state obtained by halting the diagnosis process when reaching a ϵ-stable health state for $\epsilon = 0.1$. Recall that higher quality health states corresponds to higher values of AUC and to lower values of Euclidean distance.

The results show that increasing ϵ results in a lower quality health state. This demonstrates the expected tradeoff between the number of diagnoses used to generate a health state and its quality. Indeed, we already observed this tradeoff for the wasted effort metric in Fig. 6. This is basically the classical runtime vs. solution quality tradeoff, as finding more diagnoses requires more runtime.

Nonetheless, we observe that the actual difference between the AUC/Euclidean values of obtained with all diagnoses ($\epsilon = 0$) are often very close to those obtained for $\epsilon = 0.1$. For example, the AUC obtained with all diagnoses for system c880 is 0.87 while for $\epsilon = 0.1$ it was 0.84. This is especially remarkable because it means that one can obtain a 0.1-stable health state with only 29% of the diagnoses (see Table 6 for these results). Similar trends can be observed in most systems, including the software diagnosis domain.

Note that while the results showed that the health state obtained using all the diagnoses is better than the one obtained with fewer diagnoses *on average*, there are individual observations where adding more diagnoses do not improve the health state quality. Such an example was shown in Table 5, where H_3 was better than health states generated with more diagnoses. Thus, to provide further context for the presented quality values, Tables 9 and 10 also contain a special row, marked “Oracle”. The Oracle values represent the quality of the health state generated by a DA that has an optimal stopping condition: it stops when adding more diagnoses will not improve the health state. More formally, for AUC the Oracle value is $\max_{i\in\{1,\dots,|\Omega|\}}(eval(H_i))$ and for the Euclidean distance the Oracle value is $\min_{i\in\{1,\dots,|\Omega|\}}(eval(H_i))$. For example, in our running scenario the Oracle value is 1.23, which is obtained after finding the first three diagnoses (see Table 5 for the $eval(H_i)$ values). Note that the Oracle is computed for a specific order of the returned diagnoses, and for that order the quality of a ϵ-stable health state cannot be better than the Oracle value.

Considering the results in Tables 9 and 10, we observe that in practice the difference in quality between Oracle value and the health state obtained using all the diagnoses ($\epsilon = 0$) is relatively small. For the Boolean circuits experiments, the largest difference is 0.11 and 0.03 for the AUC and the Euclidean distance metric, respectively. The differences are even smaller for the software diagnosis experiments. Note that the AUC value of the “All” software diagnosis experiments is not as high as in the Boolean circuit experiments (0.71 compared to approximately 0.90) even for the Oracle. This is because there are 53 out of 136 observations in which the correct diagnosis (the one that consists the faulty components) does not exist in the list of diagnoses found by the DA (Ω) and thus the true positives are low.

The fact that using all diagnoses is relatively close to the Oracle value is especially interesting because the results also show that with $\epsilon = 0.1$ we get a health state that is almost the same as for $\epsilon = 0$, and consequently to the Oracle value. This is important because as shown in Tables 6, 7, and 8, the ϵ-stable health state for $\epsilon = 0.1$ is generated from a small portion of the entire set of diagnoses. This demonstrates the potential benefit of a smart DA that will halt early the search for diagnoses but still end up with a high quality health state. More generally, our conclusions from the analysis so far are:

- For all quality metrics, only a small subset of all diagnoses are needed to reach a ϵ-stable health state.
- Increasing ϵ results in fewer diagnoses required to find a ϵ-stable health state.

Table 11
Convergence rate and wasted effort cost for the orders: Forward, Backward and Random and SAFARI using the AUC metric.

Order	74181	74182	74283	c1355	c432	c499	c880
Forward	**0.17**(0.26)	0.48(0.21)	**0.15**(0.19)	0.30(0.26)	**0.21**(0.18)	0.17(0.01)	0.29(0.22)
SAFARI	0.40(0.18)	0.65(0.17)	0.31(0.2)	**0.23**(0.32)	0.23(0.21)	**0.08(0.17)**	**0.26(0.16)**
Random	0.54(0.27)	0.59(0.18)	0.40(0.19)	0.46(0.28)	0.42(0.19)	0.47(0.02)	0.27(0.24)
Backward	0.84(0.28)	**0.46**(0.17)	0.85(0.19)	0.72(0.27)	0.63(0.2)	0.76(0.03)	0.28(0.29)

Table 12
Convergence rate and wasted effort cost for the orders: Forward, Backward and Random and SAFARI using the Euclidean distance metric.

Order	74181	74182	74283	c1355	c432	c499	c880
Forward	**0.01**(0.63)	0.13(0.38)	**0.01**(0.44)	**0.01**(0.55)	**0.01**(0.38)	**0.01**(0.52)	**0.01**(0.64)
SAFARI	**0.01**(0.60)	**0.05**(0.30)	**0.01**(0.44)	0.01(0.61)	**0.01**(0.41)	0.02(0.53)	**0.01**(0.55)
Random	0.03(0.66)	0.07(0.30)	0.03(0.54)	**0.01**(0.66)	**0.01**(0.51)	0.02(0.54)	**0.01**(0.59)
Backward	0.19(0.69)	0.25(0.26)	0.27(0.64)	**0.01**(0.66)	**0.01**(0.53)	**0.01**(0.53)	**0.01**(0.60)

- AUC is easier to tune effectively than Euclidean distance, since its values are normalized between 0 and 1.
- The wasted effort cost is correlative with the convergence rate.
- A small fraction of the diagnoses is needed to generate a health state that has almost the same quality as the health state generated by all diagnoses, for all the proposed quality metrics (wasted effort, AUC, and Euclidean distance).

5.3. *Diagnoses from different DAs*

In the above analysis we used a domain-specific solver for the software diagnosis domain, and a standard *forward-search DA* for the Boolean circuits domain that returns diagnoses in increasing order of their cardinality. Denote this order of increasing cardinality as a **"Forward"** order. Many DAs search for diagnoses in a forward order [14,45,27, inter alia], but finding diagnoses in different orders may be faster [46]. In this section we **investigate the impact of different ordering of diagnoses on the quality of the generated health state**.

"Backward" order stands for a decreasing order of the diagnoses, from biggest to smallest cardinality. In this order, the first health state is calculated based on a subset-minimal diagnosis that had the maximal number of faulty components. The last health state is calculated based on having all previous diagnoses and a new diagnosis that had the minimal number of faulty components. **"Random"** order stands for a random order of the subset-minimal diagnoses. To minimize variance, the results for Random order were obtained by repeatedly (20 times) calculating the health state of a randomly ordered set of subset-minimal diagnoses found by our DA. The last order we examined is the order of diagnoses as returned by the SAFARI algorithm [46], a stochastic state-of-the-art DA. We denote this order by **"SAFARI"**. SAFARI uses a SAT solver to initiate a candidate diagnosis and then employs an iterative elimination procedure in order to minimize its cardinality. SAFARI requires two parameters M and N as inputs. N represents the number of randomly independent searches for the first consistent candidate diagnosis and M represents the number of attempts the DA makes to improve the cardinality of the initial diagnoses (while preserving their consistency). It is important to state that this procedure does not guarantee minimal cardinality diagnoses. In our experiment we set $N = 440$ which is approximately the average number of overall diagnoses found per system and M was not set to a fixed number but rather to the cardinality of the initial candidate diagnosis (the number of faulty components in the initial consistent candidate diagnosis).

Tables 11 and 12 present the convergence rate of the different orderings – Forward, Backward, Random and SAFARI. All results are for $\epsilon = 0.1$, where Table 11 presents the results achieved using the AUC metric and Table 12 presents the results achieved using the Euclidean distance metric. For every system, the smallest convergence rate is marked in bold. It is clear that the order significantly affects the convergence rate, while the wasted effort cost is not significantly affected. In most cases the Forward order convergence rate is lower than the convergence rate in other orders. The difference is especially prominent when comparing it to the Backward order. Note that the quality of the health state obtained after convergence was very similar between the ordering. The wasted effort values shown in brackets show this, and similar trends were observed for the other quality metrics.

Thus, a forward search, in general, is able to find health states of similar quality but with fewer diagnoses, which can translate to lower runtime. To illustrate the benefit of the Forward order, consider the convergence rate and wasted effort cost achieved using the AUC metric for 74181 system. Returning diagnoses in Forward order needs only 17% of the diagnoses in order to obtain a ϵ-stable health state, whereas the Backward order needs the computation of 84% of the diagnoses to obtain a ϵ-stable health state. Moreover, the wasted effort cost obtained by the Backward order is greater by 0.02 than that of the Forward order.

The convergence rate of the Random order is somewhere between the Forward and Backward order. The convergence rate results for the different orders can be explained by the ratio of minimal cardinality diagnoses found (diagnoses that have minimal number of faulty components). The Forward order health state calculation is based on fewer diagnoses (lower convergence rate), however, these diagnoses include all minimal cardinality diagnoses. The Random order health state calcu-

Table 13
The fraction of the minimal cardinality diagnoses required to reach convergence using AUC metric.

ϵ	74181	74182	74283	c1355	c432	c499	c880
0.1	0.58(0.26)	0.71(0.21)	0.56(0.19)	0.53(0.26)	0.71(0.18)	0.69(0.01)	0.55(0.22)
0.2	0.38(0.44)	0.51(0.31)	0.40(0.26)	0.37(0.29)	0.59(0.24)	0.52(0.22)	0.47(0.28)
0.3	0.26(0.55)	0.39(0.40)	0.27(0.36)	0.20(0.44)	0.42(0.31)	0.39(0.4)	0.30(0.49)
0.4	0.25(0.58)	0.36(0.44)	0.26(0.36)	0.19(0.44)	0.41(0.34)	0.36(0.43)	0.28(0.49)

Table 14
The fraction of the minimal cardinality diagnoses required to reach convergence using Euclidean metric.

ϵ	74181	74182	74283	c1355	c432	c499	c880
0.1	0.17(0.63)	0.50(0.38)	0.10(0.44)	0.08(0.55)	0.33(0.38)	0.26(0.52)	0.13(0.64)
0.2	0.15(0.65)	0.32(0.49)	0.07(0.45)	0.08(0.55)	0.32(0.38)	0.26(0.52)	0.13(0.64)
0.3	0.15(0.65)	0.29(0.50)	0.07(0.44)	0.08(0.55)	0.32(0.38)	0.26(0.52)	0.13(0.64)

lation is based on more diagnoses but fewer minimal cardinality diagnoses and the Backward order health state calculation is based on the least amount of minimal cardinality diagnoses. This indicates that finding minimal cardinality diagnoses, as commonly believed in MBD, is better.

Let us observe the SAFARI results. SAFARI was shown to return diagnoses of good quality [46], however, theses diagnoses are not guaranteed to be of minimal cardinality. Surprisingly, SAFARI results in our experiments sometimes outperform even the Forwards order. This can be explained as follows: The Forward search order is generated by performing a breadth-first search over the space of possible diagnoses. Thus, diagnoses with common components are found together. As a result, subsequent diagnoses in Forward order are relatively similar to each other. This may affect the health state much by increasing the value of the common components. By contrast, SAFARI searches the space of diagnoses semi-randomly, and thus the diagnoses it returns are more diverse and build the health state more evenly.

5.3.1. Finding minimal cardinality diagnoses

Many previously proposed DAs focus on finding only minimal cardinality diagnoses, which are diagnoses that have minimal number of faulty components [14,45]. The rational of this focus lies in the belief that malfunction containing smaller number of faulty components is more likely to occur. As a side note, we explored the relation between the number of diagnoses required until the health state converges to a stable value and the number of minimal cardinality diagnoses.

Our results provide some support for this rational: in more than 70% of the observations in all the examined Boolean circuits systems, the health state has converged to a ϵ-stable value (using the AUC metric) without requiring any diagnosis that is not of minimal cardinality. Using the Euclidean distance metric, almost all the observations (more than 90%) needed only to compute minimal cardinality diagnoses to obtain a ϵ-stable health state. Thus, in most cases, limiting the DA to find only minimal cardinality diagnoses seems to be a reasonable choice.

Moreover, a high-quality health state can be found with only a subset of all minimal cardinality diagnoses. Tables 13 and 14 show the ratio of minimal cardinality diagnoses (where 1 corresponds to all minimal cardinality diagnoses) required on average, to reach a ϵ-stable health state using AUC and Euclidean metrics respectively. As can be seen, even for $\epsilon = 0.1$, only a fraction of the minimal cardinality diagnoses are needed to reach a ϵ-stable health state. For example, only 17% of the minimal cardinality diagnoses are needed to generate a health state that is ϵ-stable for $\epsilon = 0.1$ for the 74181 system under the Euclidean distance metric (Table 14). Note that the values are normalized by the total number of the minimal cardinality diagnoses and therefore they are higher than the values presented in Tables 6 and 7 which are normalized by all diagnoses.

6. Online stopping criteria

The results described above suggest that finding high quality health states is possible without finding all the diagnoses. This is a very important observation because it means that we can halt the DA earlier without losing the quality of the resulting health state. Now, we wish to propose a stopping condition that could be used online to determine when to stop the search for more diagnoses. Such a stopping criteria is especially needed in stochastic DAs such as SAFARI [46], which are known to work well in practice but require parameter tuning. We propose a simple stopping condition: halt when the Euclidean distance between two subsequent health states (H_i and H_{i+1}) is below a predefined threshold δ.[4]

The "$\delta = \mathbf{X}$" rows in Table 15 and 16 show the average AUC and Euclidean distances, respectively, obtained by using this online stopping criteria for a range of δ values. Since no subsequent diagnoses are exactly the same the Euclidean distance between H_i and H_{i+1} is never zero. Thus, the $\delta = 0$ row actually means the health state generated from all diagnoses. Marked in bold are the best values per system.

[4] Note that we cannot use AUC values for an online stopping condition, because computing the AUC requires knowing the correct diagnosis.

Table 15
Average AUC values obtained with the online stopping criterion (the higher the better).

δ	Boolean circuits (ISCAS-85)							Software diagnosis	
	74181	74182	74283	c1355	c432	c499	c880	Complete	All
$\delta = 0$ (ALL)	0.82	**0.90**	**0.84**	0.80	0.82	0.95	**0.84**	**0.95**	**0.70**
$\delta = 0.1$	**0.85**	0.87	0.80	**0.82**	**0.88**	**0.96**	0.82	**0.95**	**0.70**
$\delta = 0.2$	0.82	0.86	0.76	0.82	0.87	0.96	0.80	0.96	0.70
$\delta = 0.3$	0.77	0.82	0.65	0.72	0.83	0.93	0.78	0.96	0.70
$\delta = 0.4$	0.76	0.81	0.65	0.72	0.83	0.89	0.74	0.95	0.70

Table 16
Average Euclidean distance values obtained with the online stopping criterion (the lower the better).

δ	Boolean circuits (ISCAS-85)							Software diagnosis	
	74181	74182	74283	c1355	c432	c499	c880	Complete	All
$\delta = 0$ (ALL)	**0.17**	**0.23**	**0.21**	**0.06**	**0.10**	0.08	**0.06**	**0.01**	**0.03**
$\delta = 0.1$	**0.17**	0.25	0.22	**0.06**	**0.10**	**0.07**	0.07	**0.01**	0.04
$\delta = 0.2$	0.18	0.26	0.23	0.07	0.09	0.08	0.07	0.01	0.04
$\delta = 0.3$	0.19	0.28	0.25	0.07	0.10	0.08	0.07	0.01	0.04
$\delta = 0.4$	0.19	0.29	0.25	0.07	0.10	0.09	0.08	0.01	0.04

Although the proposed online stopping condition is very simple, the results in Tables 15 and 16 show that the resulting values for $\delta = 0.1$ are almost exactly the same as those obtained for health states generated from all diagnoses ($\delta = 0$). In some cases, the health state obtained by the online stopping condition is even better than the health state of all diagnoses. The health state found when using the online stopping is also comparable to the quality obtained by the ϵ-stable health state for $\epsilon = 0.1$ (presented in Tables 9 and 10).

An online stopping condition such as the one proposed above can be only one part of a more holistic diagnosis system, in which one is also capable of performing diagnostic actions that provide additional information and prune possible diagnoses. Such a system poses interesting challenges such as when obtaining additional observations is better than searching for additional diagnoses for the current set of observations. To this end one needs to weigh the potential benefit to the obtained health state of finding more diagnoses compared to obtaining new observations. Proposing such a system is beyond the scope of this work.

7. Discussion

Our experiments were performed on two domains: Boolean circuits and a software diagnosis. The Boolean circuit domain were chosen since they are a standard benchmark in the diagnosis literature, used in numerous prior works [47,46,48,12,26,49] and even used as a standard for automatic benchmark generation [50]. The software diagnosis domain is a real world domain and have been recently used for software diagnosis experiments [41]. However, both domains have a weak-fault model, i.e., the abnormal behavior of components is not modeled.

However, the theoretical results in this paper can be generalized to a model where the faulty behavior of the components is expressed (a strong fault-model). The key difference between a health state for a weak fault-model (WFM) and health state for a strong fault model (SFM). In a WFM, the health state maps a component to the probability that it is faulty. In a SFM, the health state maps a component and behavior mode to the probability that this behavior mode it is the correct behavior mode for that component. Note that since it is harder for DAs to produce diagnoses for SFM, and usually DAs outputs more diagnoses, then the expected benefit from considering the health state and ending the search before finding all diagnoses can be great. This is a topic for future work.

Another interesting topic is how to obtain and evaluate health states in real hybrid systems, which deal with continuous values rather than Boolean values. For components that have discrete behavior modes, e.g., health or faulty, the health state can be defined exactly as in this paper. The problem becomes more complex for components that do not have such discrete behavior modes, but have a continuous "health" value that specifies its deviation from the nominal behavior. The challenge there is how to aggregate such values from a set of diagnoses in a meaningful way. One approach can be to average for every component the health value assumed to them in every diagnoses. This too is an exciting topic for future work.

8. Related work

Many approaches and algorithms for MBD have been proposed. A classic approach to MBD, implemented in DAs like GDE [1], CDA* [28], SDE [26] and HS-DAG [51], find diagnoses in a two stage process. First, conflict sets are identified, each of which includes at least one fault. Then, a hitting set algorithm is used to extract diagnoses from these set of conflicts [2]. Others have proposed a compilation-based approach. For instance, Torasso and Torta [52] apply Binary Decision Diagrams (BDDs) and Darwiche [53] compiles an MBD problem into Decomposable Negation Normal Form (DNNF). Feldman et al. [54] propose MERIDIAN which compiles the MBD problem as a MAX-SAT problem. Recently, Metodi et al. [14] proposed a novel SAT-based approach to compute minimal cardinality diagnoses. A diagnosis algorithm based on a greedy stochastic approach was also proposed [24]. All these DAs may potentially return a large set of diagnoses. Thus, they are orthogonal to our work, as a health state can be generated from the set of diagnoses, regardless of the DA used.

Keren et al. [37] present an alternative approach to diagnosis that combines MBD with multi-label classification. They propose to build a classifier that maps symptoms (observations) of the system to possible faults. The major advantage of this approach is in reducing significantly the online computational complexity. The learning process of the relations between observations and the diagnoses is performed in advance offline. Afterward (online), a diagnosis can be computed immediately by using the classifier that was learned offline. Unlike other DAs mentioned above, this machine learning approach to diagnosis returns a single diagnosis. Similar to the AUC metric we used to evaluate health states, the output of this machine learning DA is measured using standard classification metrics such as false positives and false negatives. A key difference is that we do not propose a new DA that returns a single diagnosis, but propose a general approach for aggregating a set of diagnoses that can be applied for a wide range of DAs.

Maier et al. [55] remark that solutions to AI problems in engineering domains need to be compactly represented for the needs of engineers. However, we approach this challenge from opposite sides: we aim to compactly represent information generated from a low-level representation (a large number of diagnoses), while they assume a high-level representation is available and aim to auto-generate low-level representation such as Bayesian networks to serve as input to off-the-shelf diagnostic tools. Thus, the two approaches are complementary.

Some methods try to deal with reducing the number of diagnoses after they are found. There are two known methods to discriminate diagnoses: *testing* and *probing* [1]. In testing, the diagnosis process is run through additional input vectors. Probing is similar, but instead of running the diagnosis on a new input vector, probes are observations of the output of internal components. Both methods can prune diagnoses, removing diagnoses that are not consistent with the new observations. Both methods can be run iteratively until hopefully focusing on a single diagnosis. Since assigning probes and performing tests are costly tasks, the main challenge in both methods is to reduce the number of probes or tests. A common greedy approach to address this challenge is to choose a probe or a test that maximizes the information gain [16]. Others analyze the model and diagnoses to find crucial literals, whose values can differentiate the possible diagnoses [56], or model testing as abductive reasoning [57]. The fundamental difference between any form of interactive diagnosis – probing and testing – and ours is that we do not consider how to obtain new information, but only how to aggregate the information already given by the DA. Moreover, in probing and testing the aim is to minimize the “physical” cost incurred by performing probes and tests, while we aim to avoid the computational cost of computing all diagnoses.

A new approach to testing has been proposed by Feldman et al. [6] as part of the LYDIA-NG framework for Model-Based Diagnosis. This framework uses active testing on a real satellite Electrical Power System to decrease the diagnostic uncertainty in order to determine likely explanation faults. Similar to our work, they also use several known metrics in order to compute the residual difference, meaning the distance to the truly diagnosis (the faults that have actually occurred). The computation of metrics allows users to compare the performance of various diagnostic and simulation approaches. The LYDIA-NG framework presents a single diagnostic output containing the normalized probability of failure for each component. Our approach uses the residual difference to calculate the added value of finding more diagnoses, and presents as an output the health state – the likelihood of each component to be faulty.

9. Conclusions and future work

In this work we proposed an alternative form of output for DAs called the health state. A health state maps every component to a probability that it is faulty. We argue that a health state is for many cases a more reasonable output than a single or k most probable diagnoses since it contains aggregated information over all the found diagnoses. The health state provides an answer to a fundamental diagnostic question: which components is likely to be faulty. It provides an overview of the state of the system that is readable by a human operator and can be useful for an automated troubleshooter. We presented a simple and efficient method to generate a health state from an arbitrarily large set of diagnoses. Two metrics were proposed to measure the quality of generated health states – Euclidean distance and the Area Under the Curve (AUC).

Next, we observed empirically that:

1. According to all quality metrics, a health state generated using more diagnoses is better (higher AUC, lower Euclidean distance and wasted efforts). Thus, introducing an expected trade-off between the number of diagnoses used to generate a health state and its quality.
2. Empirically, the AUC metric provided a finer grained control over this trade-off, allowing easier parameter tuning.
3. For all quality metrics, the health state quality tends to converge quickly to a relatively high (i.e., good) value, where in most cases a fraction of the diagnoses are needed to obtain a health state that is almost as accurate as the one generated using all diagnoses.
4. A DAs that return diagnoses in increasing cardinality order have a tendency to converge faster while preserving high health state quality, but SAFARI [46], a stochastic DA, sometimes reaches convergence faster but resulting in a lower quality health state.
5. In many cases only a small subset of all minimal cardinality diagnoses was needed in order to reach this convergence.

6. The simple online stopping criteria we proposed for identifying when a health state with a high quality has been found and the search for more diagnoses can be halted was found to be very effective, successfully reaching a high quality health state by computing only a small subset of the diagnoses.

These findings highlight the usefulness of the health state as a way to consider and evaluate a set of diagnoses.

There are many exciting topics for future work. Some were outlined in Section 7, dealing with more complex system models. Another interesting future research is to use and test our health state in the context of probing and testing [15, 16]. Probing methods assess where the placement of probes will be most beneficial. It will be interesting to examine the performance of probing and testing approaches when using upon a subset of the diagnoses given by a stopping criteria based on the health state. In addition, an interesting topic for future work is to use the values in the health state as a heuristic for guiding a DA to find more diagnoses.

Appendix A. System description of a simple Boolean circuit

Here is a system description (SD) of the simple Boolean circuit given in Fig. 2.

$$h(X1) \rightarrow ((A XOR B) \leftrightarrow Z1) \tag{A.1}$$

$$h(X2) \rightarrow ((C XOR Z1) \leftrightarrow D) \tag{A.2}$$

$$h(A1) \rightarrow ((A \wedge B) \leftrightarrow Z2) \tag{A.3}$$

$$h(A2) \rightarrow ((C \wedge Z1) \leftrightarrow Z3) \tag{A.4}$$

$$h(O1) \rightarrow ((Z2 \vee Z3) \leftrightarrow E) \tag{A.5}$$

The observations in Fig. 2 are represented as the following conjunction.

$$A = 1 \wedge B = 0 \wedge C = 1 \wedge D = 0 \wedge E = 0$$

References

[1] J. de Kleer, B.C. Williams, Diagnosing multiple faults, Artif. Intell. 32 (1) (1987) 97–130.
[2] R. Reiter, A theory of diagnosis from first principles, Artif. Intell. 32 (1) (1987) 57–96.
[3] P. Struss, C. Price, Model-based systems in the automotive industry, AI Mag. 24 (4) (2003) 17–34.
[4] O. Niggemann, B. Stein, T. Spanuth, H. Balzer, Using models for dynamic system diagnosis: a case study in automotive engineering, in: Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme V, Schloss Dagstuhl, Tagungsband Modellbasierte Entwicklung eingebetteter Systeme, 2009, pp. 46–56.
[5] B.C. Williams, P.P. Nayak, A model-based approach to reactive self-configuring systems, in: AAAI, 1996, pp. 971–978.
[6] A. Feldman, H.V. de Castro, A. van Gemund, G. Provan, Model-based diagnostic decision-support system for satellites, in: 2013 IEEE Aerospace Conference, IEEE, 2013, pp. 1–14.
[7] G. Steinbauer, A. Kleiner, F. Wotawa, Using qualitative and model-based reasoning for sensor validation of autonomous mobile robots, in: The 20th International Workshop Principles of Diagnosis, DX-09, 2009, pp. 219–226.
[8] E. Khalastchi, M. Kalech, L. Rokach, Sensor fault detection and diagnosis for autonomous systems, in: M.L. Gini, O. Shehory, T. Ito, C.M. Jonker (Eds.), International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6–10, 2013, IFAAMAS, 2013, pp. 15–22, http://dl.acm.org/citation.cfm?id=2484927.
[9] R. Abreu, P. Zoeteweij, A.J.C. van Gemund, Simultaneous debugging of software faults, J. Syst. Softw. 84 (4) (2011) 573–586.
[10] T. Zamir, R. Stern, M. Kalech, Using model-based diagnosis to improve software testing, in: AAAI, 2014, pp. 1135–1141.
[11] T. Eiter, W. Faber, N. Leone, G. Pfeifer, The diagnosis frontend of the dlv system, AI Commun. 12 (1–2) (1999) 99–111.
[12] S.A. Siddiqi, Computing minimum-cardinality diagnoses by model relaxation, in: IJCAI, 2011, pp. 1087–1092.
[13] F. Brglez, D. Bryan, K. Koźmiński, Combinational profiles of sequential benchmark circuits, in: IEEE International Symposium on Circuits and Systems, 1989, IEEE, 1989, pp. 1929–1934.
[14] A. Metodi, R. Stern, M. Kalech, M. Codish, A novel sat-based approach to model based diagnosis, J. Artif. Intell. Res. (2014) 377–411.
[15] M. Shakeri, V. Raghavan, K.R. Pattipati, A. Patterson-Hine, Sequential testing algorithms for multiple fault diagnosis, IEEE Trans. Syst. Man Cybern., Part A, Syst. Hum. 30 (1) (2000) 1–14.
[16] A. Feldman, G. Provan, A. van Gemund, A model-based active testing approach to sequential diagnosis, J. Artif. Intell. Res. 39 (2010) 301–334.
[17] A. Feldman, J. de Kleer, G. Provan, Computing manifestations of max-size min-cardinality ambiguity groups, in: The Diagnostic Reasoning: Model Analysis and Performance ECAI Workshop, DREAMAP, 2012, pp. 26–33.
[18] R. Mirsky, K. Gal, R. Stern, M. Kalechy, Sequential plan recognition, in: International Joint Conference on Artificial Intelligence (IJCAI), 2016, pp. 401–407.
[19] T. Mitchel, Machine Learning, McGraw Hill, 1997.
[20] R. Abreu, P. Zoeteweij, A.J. Van Gemund, Spectrum-based multiple fault localization, in: 24th IEEE/ACM International Conference on Automated Software Engineering, 2009, ASE'09, IEEE, 2009, pp. 88–99.
[21] R. Abreu, A.J. Van Gemund, Diagnosing multiple intermittent failures using maximum likelihood estimation, Artif. Intell. 174 (18) (2010) 1481–1497.
[22] M.C. Hansen, H. Yalcin, J.P. Hayes, Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering, IEEE Des. Test Comput. 16 (1999) 72–80.
[23] Apache Ant, ant.apache.org.
[24] A. Feldman, T. Janssen, A. van Gemund, Modeling diagnostic stochastic search, in: International Workshop on Principles of Diagnosis, 2011, pp. 1–6.
[25] R.T. Stern, M. Kalech, S. Rogov, A. Feldman, How many diagnoses do we need?, in: AAAI, 2015, pp. 1618–1624.
[26] R.T. Stern, M. Kalech, A. Feldman, G.M. Provan, Exploring the duality in conflict-directed model-based diagnosis, in: AAAI, 2012, pp. 828–834.
[27] J. de Kleer, B.C. Williams, Diagnosing multiple faults, Artif. Intell. 32 (1) (1987) 97–130.
[28] B.C. Williams, R.J. Ragno, Conflict-directed A* and its role in model-based embedded systems, Discrete Appl. Math. 155 (12) (2007) 1562–1595.
[29] J.D. Kleer, Minimum cardinality candidate generation, in: The 20th International Workshop Principles of Diagnosis (DX-09), 2009, pp. 397–402.

[30] B. Selman, H.J. Levesque, Abductive and default reasoning: a computational core, in: AAAI, 1990, pp. 343–348.
[31] T. Bylander, D. Allemang, M.C. Tanner, J.R. Josephson, The computational complexity of abduction, Artif. Intell. 49 (1) (1991) 25–60.
[32] J. Kohlas, B. Anrig, R. Haenni, P.-A. Monney, Model-based diagnostics and probabilistic assumption-based reasoning, Artif. Intell. 104 (1) (1998) 71–106.
[33] J. de Kleer, B.C. Williams, Diagnosis with behavioral modes, in: IJCAI, vol. 1, 1989, pp. 1324–1330.
[34] P.J. Lucas, Bayesian model-based diagnosis, Int. J. Approx. Reason. 27 (2) (2001) 99–119.
[35] J. de Kleer, Focusing on probable diagnoses, in: National Conference on Artificial Intelligence, AAAI, 1991, pp. 842–848.
[36] D. Heckerman, J.S. Breese, K. Rommelse, Decision-theoretic troubleshooting, Commun. ACM 38 (3) (1995) 49–57.
[37] B. Keren, M. Kalech, L. Rokach, Model-based diagnosis with multi-label classification, in: The International Workshop on Principles of Diagnosis, DX, 2011.
[38] R. Stern, M. Kalech, H. Shinitzky, Implementing troubleshooting with batch repair, in: The International Workshop on Principles of Diagnosis, DX, 2015 (accepted).
[39] R. Stern, M. Kalech, Repair planning with batch repair, in: The International Workshop on Principles of Diagnosis, DX, 2014.
[40] A. Feldman, T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, J. de Kleer, L. Kuhn, A. van Gemund, Empirical evaluation of diagnostic algorithm performance using a generic framework, Int. J. Progn. Health Manag. 1 (2) (2010).
[41] A. Elmishali, R. Stern, M. Kalech, Data-augmented software diagnosis, in: The AAAI Conference on Artificial Intelligence, 2016, pp. 4003–4009.
[42] A. Darwiche, Modeling and Reasoning with Bayesian Networks, Cambridge University Press, 2009.
[43] R. Abreu, P. Zoeteweij, A.J.C. van Gemund, Simultaneous debugging of software faults, J. Syst. Softw. 84 (4) (2011) 573–586.
[44] R. Abreu, A.J. van Gemund, A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis, in: SARA, vol. 9, 2009, pp. 2–9.
[45] S.A. Siddiqi, J. Huang, Sequential diagnosis by abstraction, J. Artif. Intell. Res. 41 (2011) 329–365.
[46] A. Feldman, G. Provan, A. van Gemund, Approximate model-based diagnosis using greedy stochastic search, J. Artif. Intell. Res. 38 (2010) 371–413.
[47] A. Feldman, A.J.C. van Gemund, A two-step hierarchical algorithm for model-based diagnosis, in: AAAI, 2006, pp. 827–833.
[48] S.A. Siddiqi, J. Huang, Hierarchical diagnosis of multiple faults, in: IJCAI, 2007, pp. 581–586.
[49] I. Nica, I. Pill, T. Quaritsch, F. Wotawa, The route to success: a performance comparison of diagnosis algorithms, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2013, pp. 1039–1045.
[50] J. Wang, G. Provan, A benchmark diagnostic model generation system, part A: systems and humans, IEEE Trans. Syst. Man Cybern. Syst. 40 (5) (2010) 959–981.
[51] B. Peischl, F. Wotawa, Computing diagnoses efficiently, a fast theorem prover for propositional horn clause theories, in: International Workshop on Principles of Diagnosis, DX, 2003.
[52] P. Torasso, G. Torta, Model-based diagnosis through OBDD compilation: a complexity analysis, in: Reasoning, Action and Interaction in AI Theories and Systems, 2006, pp. 287–305.
[53] A. Darwiche, Decomposable negation normal form, J. ACM 48 (4) (2001) 608–647.
[54] A. Feldman, G. Provan, J. de Kleer, S. Robert, A. van Gemund, Solving model-based diagnosis problems with max-sat solvers and vice versa, in: International Workshop on Principles of Diagnosis, 2010.
[55] P. Maier, D. Jain, M. Sachenbacher, Diagnostic hypothesis enumeration vs. probabilistic inference for hierarchical automata models, in: The International Workshop on Principles of Diagnosis, DX, Murnau, Germany, 2011.
[56] A. Sattar, R. Goebel, Using crucial literals to select better theories, Comput. Intell. 7 (1) (1991) 11–22.
[57] S.A. McIlraith, Generating tests using abduction, in: The International Conference on Principles of Knowledge Representation and Reasoning, KR, 1994, pp. 449–460.