Efficient Sequential Model-Based Fault-Localization with Partial Diagnoses

Kostyantyn Shchekotykhin[1] and **Thomas Schmitz**[2] and **Dietmar Jannach**[2]
[1]Alpen-Adria University Klagenfurt, Austria
e-mail: kostyantyn.shchekotykhin@aau.at
[2]TU Dortmund, Germany
e-mail: {firstname.lastname}@tu-dortmund.de

Abstract

Model-Based Diagnosis is a principled approach to identify the possible causes when a system under observation behaves unexpectedly. In case the number of possible explanations for the unexpected behavior is large, *sequential diagnosis* approaches can be applied. The strategy of such approaches is to iteratively take additional measurements to narrow down the set of alternatives in order to find the true cause of the problem.

In this paper we propose a sound and complete sequential diagnosis approach which does not require any information about the structure of the diagnosed system. The method is based on the new concept of "partial" diagnoses, which can be efficiently computed given a small number of minimal conflicts. As a result, the overall time needed for determining the best next measurement point can be significantly reduced. An experimental evaluation on different benchmark problems shows that our sequential diagnosis approach needs considerably less computation time when compared with an existing domain-independent approach.

1 Introduction

Model-Based Diagnosis (MBD) techniques aim at determining the possible causes of an unexpected behavior of an observed system based on knowledge about the system's expected behavior when all of its components work correctly. The basic MBD principles were developed in the late 1980s [Davis, 1984; Reiter, 1987; de Kleer and Williams, 1987] and have since then been applied to various problem settings including electronic circuits and various sorts of software artifacts like knowledge bases, logic programs, ontologies, or spreadsheets.

One challenge when applying MBD is that the number of possible diagnoses can sometimes be large, making it infeasible for the user to check each diagnosis individually. There are, e.g., 6,944 diagnoses for the system *c432 (scenario 0)* of the DX 2011 diagnosis competition benchmark even when we limit the maximum cardinality of the diagnoses to five.

Different approaches to deal with the problem exist. One option is to rank the diagnoses based on fault probabilities so that the chance increases that the user finds the true diagnosis earlier. Or, we can focus the diagnostic process itself on the most *probable* diagnoses [de Kleer, 1992] and compute only a subset of all diagnoses, which comes at the price of incompleteness. Finally, we can take additional measurements to discriminate between fault causes, e.g., based on information-theoretic considerations [de Kleer and Williams, 1987].

[Shchekotykhin *et al.*, 2012] more recently compared two different strategies for taking the next measurement, applied to the problem of ontology debugging. These strategies are part of a *sequential diagnosis* process in which the ontology engineer is interactively queried about the correctness of certain axioms inferred by the faulty ontology. The answers are then used to reduce the space of the remaining diagnoses. Compared to heuristic or approximate approaches, the advantage of their method is that it is *complete*, i.e., that the true error will be identified at the end, which can be a requirement in different MBD-application domains like software debugging.

One limitation of the work of [Shchekotykhin *et al.*, 2012] is that for hard problem instances the computation of a query can be computationally expensive. In their approach, they therefore first search for a few *leading* diagnoses given the current state of the sequential debugging process and then determine the optimal query to the user, i.e., the one that partitions this set of diagnoses in the best possible way. However, for some real-world cases the computation of even a few leading diagnoses is challenging [Shchekotykhin *et al.*, 2014].

In our work we address the same problem setting and aim to reduce the time of the sequential diagnosis sessions. Specifically, the technical contribution of our work is the notion of "partial" diagnoses, which can be efficiently computed using a subset of the minimal conflicts. As usual, we then determine the best possible partitioning of the partial diagnoses, which however typically form a smaller search space than in the original problem setting. Moreover, we prove that our sequential method remains complete, i.e., it is guaranteed that the true problem cause – called *preferred* diagnosis – will be found. An experimental evaluation on different benchmarks shows significant reductions of the diagnosis time compared to previous works. Our method furthermore is not dependent on the availability of application-specific problem decomposition methods and can therefore be applied to efficiently diagnose complex ontologies or electronic circuits without exploiting problem-specific structural characteristics.

2 Sequential diagnosis

Before we describe our technical approach we will briefly summarize the ideas of sequential (interactive) diagnosis based on the definitions by [Reiter, 1987].

Definition 1 (Diagnosable system)**.** *Let* COMPS *be a set of components represented as a finite set of constants, and* SD *be a system description represented by a finite set of first-order sentences, then* (SD, COMPS) *defines a* diagnosable system*.*

The pair (SD, COMPS) captures the normal behavior of the system when we assume that all components work properly. The latter can be expressed by a set $\{\neg \text{AB}(c) \mid c \in \text{COMPS}\}$, where the "abnormal" AB/1 predicate is used in SD to model the expected behavior of the components. Consequently, the set of sentences $\text{SD} \cup \{\neg \text{AB}(c) \mid c \in \text{COMPS}\}$ describing the normal behavior of the diagnosable system must be consistent. A diagnosis problem arises when the observed behavior of the system – represented as a finite set of consistent first-order sentences OBS – differs from the expected one.

Furthermore, information about a fault can be obtained by means of measurements. Following the proposals of [Reiter, 1987; de Kleer and Williams, 1987; Felfernig *et al.*, 2004] we allow a user to provide *positive* and *negative* measurements.

Definition 2 (Diagnosis)**.** *Let* (SD, COMPS) *be a diagnosable system and* OBS *be a set of observations such that* $\text{SD} \cup \{\neg \text{AB}(c) \mid c \in \text{COMPS}\}$ *is consistent and* $\text{SD} \cup \{\neg \text{AB}(c) \mid c \in \text{COMPS}\} \cup \text{OBS}$ *is inconsistent. In addition, let P and N be consistent sets of first-order sentences, called positive and negative measurements resp., such that* $\forall n \in N : P \not\models n$.

Then, a diagnosis *for* $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$ *is a subset-minimal set* $\Delta \subseteq \text{COMPS}$ *for which a knowledge base*

- $KB[\Delta] := \text{SD} \cup \text{OBS} \cup \{\text{AB}(c) \mid c \in \Delta\} \cup \{\neg \text{AB}(c) \mid c \in \text{COMPS} \setminus \Delta\}$ *is* consistent*,*
- $KB[\Delta] \cup P$ *is* consistent*, and*
- $\forall n \in N : KB[\Delta] \cup P \not\models n$*.*

Any diagnosis Δ corresponds to a set of components that, if assumed to be faulty, explain the observed misbehavior. If there are more diagnoses than can be manually inspected, additional measurements are taken in sequential MBD approaches to find the so-called *preferred* diagnosis Δ^*, which corresponds to the set of actually faulty components.

Definition 3 (Preferred diagnosis)**.** *Let* Δ^* *be a diagnosis for* $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$*.* Δ^* *is the* preferred diagnosis *iff* $\Delta^* = \{c \mid c \in \text{COMPS}, c \textit{ is faulty}\}$*.*

In our approach – as in many others – the computation of diagnoses is based on the concept of conflicts.

Definition 4 (Conflict)**.** *A set of components* $CS \subseteq \text{COMPS}$ *is a* conflict *for* $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$ *iff (a)* $\text{SD} \cup \text{OBS} \cup P \cup \{\neg \text{AB}(c) \mid c \in CS\}$ *is inconsistent or (b)* $\exists n \in N : \text{SD} \cup \text{OBS} \cup P \cup \{\neg \text{AB}(c) \mid c \in CS\} \models n$*. A conflict CS is* minimal *iff there is no* $CS' \subset CS$ *such that* CS' *is a conflict.*

Informally speaking a conflict is a set of components that cannot all work correctly at the same time given the observations and measurements. To resolve a minimal conflict every diagnosis therefore needs to comprise at least one of its components. Given a method for computing minimal conflicts for $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$ such as QUICKXPLAIN [Junker, 2004] or PROGRESSION [Marques-Silva *et al.*, 2013], algorithms like HS-Tree [Reiter, 1987] find *all diagnoses* $\mathbf{D}$ by enumerating all subset-minimal hitting sets of the set of *all minimal conflicts* $\mathbf{CS}$.

In sequential diagnosis settings, we are interested in the true cause of the error. This *preferred diagnosis* is found through additional information about the correctness of components which is obtained through measurements.

Property 1. Δ^* *is the* preferred diagnosis *for* $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$ *iff* Δ^* *is a diagnosis for* $(\text{SD}, \text{COMPS}, \text{OBS}, P^*, N^*)$*, where* $P^* := \{\neg \text{AB}(c) \mid c \in \text{COMPS}, c \textit{ is correct}\}$ *and* $N^* := \{\neg \text{AB}(c) \mid c \in \text{COMPS}, c \textit{ is faulty}\}$*.*

According to Definition 2 only one diagnosis exists for $(\text{SD}, \text{COMPS}, \text{OBS}, P^*, N^*)$ and, consequently, by Property 1 only one preferred diagnosis Δ^* for any $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$. However, in many cases the sets P and N do not comprise sufficient measurements to uniquely determine Δ^*. In order to find Δ^*, sequential methods extend the sets P and N by asking a user or some oracle to perform additional measurements allowing the algorithm to rule out irrelevant diagnoses [de Kleer and Williams, 1987; Shchekotykhin *et al.*, 2012]. The problem in this context is to determine "good" measurement points and correspondingly construct a set of first-order sentences Q, called query. An oracle must evaluate the correctness of the sentences in Q, thereby providing the required additional measurements.

Given a set of diagnoses D for $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$, queries are designed such that they induce two non-empty disjoint sets of diagnoses $D_1, D_2 \subseteq D$ for which: (i) If the elements of Q are stated to be correct by some oracle – such as an MBD user or some automated system[1] – then all elements of D_2 are not diagnoses for $(\text{SD}, \text{COMPS}, \text{OBS}, P \cup Q, N)$. (ii) Otherwise, if the elements of Q are considered to be incorrect, all elements of D_1 are not diagnoses for $(\text{SD}, \text{COMPS}, \text{OBS}, P, N \cup Q)$.

Definition 5 (Query)**.** *Let D be a set of diagnoses for* $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$ *and Q be a set of first-order sentences. Then Q is a* query *iff the sets* $D^P := \{\Delta_i \in D \mid KB[\Delta_i] \cup P \models Q\}$ *and* $D^N := \{\Delta_j \in D \mid KB[\Delta_j] \cup P \cup Q \textit{ is inconsistent}\}$ *are not empty.*

A query Q induces a triple $(D^P, D^N, D^\emptyset)$ *of pairwise disjoint subsets of the set D, where* $D^\emptyset = D \setminus (D^P \cup D^N)$*.*

The overall goal is to use a series of queries to narrow down the set of diagnoses D and to finally find the preferred diagnosis Δ^*. To select the best query we can use different strategies such as split-in-half, entropy, or risk-optimization [de Kleer and Williams, 1987; Rodler *et al.*, 2013].

3 Query Computation with Partial Diagnoses

3.1 Algorithm Details

Algorithm 1 summarizes our approach, which in contrast to previous works can operate on the basis of "partial" diagnoses. In its main loop the algorithm repeatedly searches

[1] As in previous works we assume the oracle to answer correctly.

Algorithm 1: FINDDIAGNOSIS

Input: A tuple $I := (\text{SD}, \text{COMPS}, \text{OBS}, P, N)$, k: number of minimal conflicts, n: number of diagnoses
Output: A preferred diagnosis Δ^*

$\Delta^* \leftarrow \emptyset$;
while $true$ **do**
  $C \leftarrow \text{FINDCONFLICTS}(I, \Delta^*, k)$;
  if $C = \emptyset$ **then return** Δ^*;
  $\Delta^* \leftarrow \Delta^* \cup \text{GETPREFERREDPD}(I, C, \emptyset, n)$;

function GETPREFERREDPD (I, C, PD, n)
  $PD \leftarrow PD \cup \text{FINDPDS}(C, n - |PD|)$;
  if $|PD| = 1$ **then return** $\delta^* : \delta^* \in PD$;
  $Q \leftarrow \text{GETQUERY}(I, PD)$;
  $(P', N') \leftarrow \text{ASKQUERY}(Q)$;
  $I \leftarrow \text{UPDATEMEASUREMENTS}(I, P', N')$;
  $C \leftarrow \text{UPDATECONFLICTS}(I, C)$;
  $PD \leftarrow \text{UPDATEPD}(I, PD)$;
  return $\text{GETPREFERREDPD}(I, C, PD, n)$;

for such preferred partial diagnoses and thereby incrementally identifies the preferred diagnosis Δ^*. The idea of partial diagnoses is that we do not compute *all* conflicts and diagnoses for a given problem in each iteration, but only determine a subset of the minimal conflicts. Finding such a subset of the existing minimal conflicts can be done e.g. with the recently proposed MERGEXPLAIN method [Shchekotykhin *et al.*, 2015]. Then, we find a set of minimal hitting sets *for this subset of the conflicts*, which correspond to partial diagnoses.

Definition 6 (Partial Diagnosis)**.** *$\delta \subseteq \text{COMPS}$ is a* partial diagnosis *for a set of minimal conflicts $C \subseteq \mathbf{CS}$ iff $\forall CS \in C : \delta \cap CS \neq \emptyset$ and there is no $\delta' \subset \delta$ such that δ' is a partial diagnosis.*[2]

Algorithm 1 starts with the computation of at most k minimal conflicts C (FINDCONFLICTS) such that $\forall CS \in C : CS \cap \Delta^* = \emptyset$. In case the returned set is empty, i.e., the provided system description is consistent with all observations and measurements, the algorithm returns $\Delta^* = \emptyset$ as a diagnosis. Otherwise, it calls GETPREFERREDPD to interactively find a preferred partial diagnosis for the minimal conflicts C.

GETPREFERREDPD calls FINDPDS which returns at most n leading partial diagnoses of C called PD. Depending on C, these partial diagnoses might have different properties. We consider two cases:

1. FINDCONFLICTS returned *all* minimal conflicts of the original problem ($C = \mathbf{CS}$). In this case all partial diagnoses computed by FINDPDS are diagnoses. Existing methods, e.g. [de Kleer and Williams, 1987], guarantee that GETPREFERREDPD finds the preferred diagnosis.

2. Only *some* of the minimal conflicts are returned in the set C. Therefore, the partial diagnoses returned by FINDPDS are not necessarily diagnoses.

[2]Note that our definition of a partial diagnosis is different from the one in [de Kleer *et al.*, 1992].

If PD comprises only one partial diagnosis, then its only element δ^* is returned as the preferred partial diagnosis. Otherwise, Algorithm 1 calls GETQUERY which computes a query Q to discriminate between the elements of PD. Inside GETQUERY, existing methods, e.g., entropy- and probability-based ones, can be used to determine the "best" query. These methods internally use the underlying problem-specific reasoning engine to derive the consequences of the different answers to possible queries. This engine can for example be a Description Logic reasoner in case of ontology debugging problems [Horridge *et al.*, 2008; Shchekotykhin *et al.*, 2012] or a constraint solver when the problem is to diagnose digital circuits [de Kleer and Williams, 1987].

Next, Algorithm 1 asks an external oracle (ASKQUERY) for a classification of the query sentences into positive and negative ones (P' and N'). If the oracle for example answers that a queried component c works correctly, $\neg\text{AB}(c)$ or any other set of logically equivalent first-order sentences is added to P', and to N', otherwise. In general, queries are not limited to atoms over the AB predicate. An MBD system can for example use problem-specific knowledge to convert Q into a logically equivalent set of first-order sentences that are easier to answer for users. We can, e.g., ask users about the specific observed outcomes of a set of gates in a faulty circuit based on knowledge about the expected behavior of the gates [Reiter, 1987; de Kleer and Williams, 1987].

These sentences are then added to the corresponding sets of positive P and negative N measurements of the updated problem description I (UPDATEMEASUREMENTS). The update requires the set C to be reviewed because some of its elements might not be minimal conflicts given the new measurements. UPDATECONFLICTS therefore internally implements a minimization method to ensure the minimality of the conflicts in C. A trivial method would be to test for every $c_i \in CS$ whether $CS' = CS \setminus \{c_i\}$ is inconsistent. If this is the case, CS is replaced by CS'. Then, UPDATEPD removes all elements of PD that are not partial diagnoses for this updated set of minimal conflicts C. We do this because these removed partial diagnoses comprise components that are not elements of any updated minimal conflict anymore. Finally, we recursively call GETPREFERREDPD to continue to search.

When GETPREFERREDPD returns, its result is added to Δ^*. Algorithm 1 then continues with the outermost while loop to check if additional conflicts exist given the updated measurements in I and the partial preferred diagnosis Δ^*.

3.2 Illustrating Example

Let us consider the system 74L85, Scenario 10, from the DX Competition 2011 Synthetic Track. There are three minimal conflicts: $\mathbf{CS}$ ={{o1}, {o2, z2, z22}, {o2, o3, z7, z9, z10, z11, z12, z13, z14, z17, z18, z19, z22, z27}}. These conflicts are not known in advance. The number of minimal hitting sets (diagnoses) for $\mathbf{CS}$ is 14, i.e., |$\mathbf{D}$|=14. The preferred diagnosis Δ^* as specified in the benchmark is {o1, z22}.

The proposed interactive diagnosis process starts with the computation of a subset C of the existing minimal conflicts using MERGEXPLAIN, e.g., C={{o1}, {o2, z2, z22}} for any $k > 1$. We then compute the minimal hitting sets of C, leading to the partial diagnoses PD={{o1, o2}, {o1, z2},

$\{$o1, z22$\}\}$, which are all subsets of diagnoses of the original problem. Based on this outcome, we compute the query Q that partitions the elements in PD in the best possible way using, e.g., an entropy-based strategy. The goal is to remove as many non-preferred diagnoses as possible through the additional measurement.

Let us assume that Q=$\{\neg\text{AB}(\text{o2})\}$, i.e., we ask the user if component o2 is working correctly. Since o2 is not actually faulty, the user answers that o2 is correct, which means that we can add o2 to P and remove it from the conflicts in C, i.e., C=$\{\{$o1$\}$, $\{$z2, z22$\}\}$. Next, we update PD and remove all elements that are no partial diagnoses for the updated set of C resulting in PD=$\{\{$o1, z2$\}$, $\{$o1, z22$\}\}$. Within the next recursive call of GETPREFERREDPD we first search for new partial diagnoses, but as we have already found all partial diagnoses for the conflicts in C, PD remains unchanged. As PD still contains more than one element, the function continues to search for the preferred partial diagnosis.

In a next step we compute $\{$z22$\}$ as the optimal query Q. Because the user correctly answers that z22 is not working normally, we again update the measurements with the new knowledge by adding z22 to N. This means that the preferred diagnosis must be a superset of $\{$z22$\}$ and we can remove all elements of PD that do not contain z22, resulting in PD=$\{\{$o1, z22$\}\}$. Furthermore, we can ignore all conflicts that contain z22 in the next steps. The next recursive call of GETPREFERREDPD will directly return $\{$o1, z22$\}$ as the preferred partial diagnosis δ^*, because again no additional partial diagnosis can be found.

Back in the main algorithm, within the while loop we try to find new conflicts with the updated measurements in I and the partial preferred diagnosis stored in Δ^*. As Δ^* already resolves all conflicts of the original diagnosis problem, we do not have to search for the third conflict in $\mathbf{CS}$ and can return Δ^*=$\{$o1, z22$\}$ as the preferred diagnosis. As a result, in the example only two user interactions were required to narrow down the set of diagnoses to the true diagnosis.

3.3 Algorithm Properties

In this section we show that Algorithm 1 always terminates and returns the preferred diagnosis Δ^*. First, we show that on every iteration GETPREFERREDPD finds a query that discriminates between the partial diagnoses in the set PD.

Proposition 1. *Let C be an arbitrary set of minimal conflicts for* $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$ *and PD be a set of partial diagnoses for C, such that $|PD| > 1$. Then, a set of first-order sentences Q exists which is a query (Definition 5) for the set of diagnoses* $D = \{\Delta \in \mathbf{D} \mid \exists \delta \in PD : \delta \subseteq \Delta\}$.

Proof. Consider two arbitrary partial diagnoses $\delta', \delta'' \in PD$. By Definition 6, at least one minimal conflict set $CS \in C$ exists for δ' and δ'' which is hit in different ways. I.e., there exists at least one constant $c \in CS$ such that $c \in \delta'$ and $c \notin \delta''$. The component c can be used to generate a query Q discriminating between the hitting sets.

Since c is an element of some minimal conflict, there exists at least one diagnosis Δ_i such that $c \in \Delta_i$ and, by Definition 2, $KB[\Delta_i]$ comprises a sentence $\text{AB}(c)$. Similarly, there exists at least one diagnosis Δ_j such that $c \notin \Delta_j$ and $KB[\Delta_j]$ comprises $\neg\text{AB}(c)$. Consequently, $KB[\Delta_i] \models \text{AB}(c)$ and $KB[\Delta_j] \models \neg\text{AB}(c)$. The set $Q = \{\text{AB}(c)\}$ is a query, since $D^P = \{\Delta_i \in \mathbf{D} \mid \delta' \subseteq \Delta_i\}$ and $D^N = \{\Delta_j \in \mathbf{D} \mid \delta'' \subseteq \Delta_j\}$ are not empty. Any other partial diagnosis $\delta \in PD \setminus \{\delta', \delta''\}$ can then be classified w.r.t. c into one of the sets D^P (if $c \subseteq \delta$) and D^N (if $\delta \cap (CS \setminus c) \neq \emptyset$). □

Corollary 1. GETPREFERREDPD *always terminates and returns a preferred partial diagnosis δ^*.*

Proof. By Proposition 1, a query exists for an arbitrary set of partial diagnoses PD. Consider a minimal conflict $CS \in C$ and a query $Q = \{\text{AB}(c)\}$ where $c \in CS$. If GETQUERY returns P' such that $\neg\text{AB}(c) \in P$ after UPDATEMEASUREMENTS, then UPDATECONFLICTS must replace CS with CS' such that $c \notin CS'$ by Definition 4 (a). Otherwise, $\neg\text{AB}(c) \in N$ and UPDATECONFLICTS replaces CS with $CS' = \{c\}$, since by Definition 4 (b) CS' is a minimal conflict, i.e., $\text{SD} \cup \text{OBS} \cup P \cup \{\neg\text{AB}(c)\} \models \neg\text{AB}(c)$. Therefore, given any answer of an oracle at least one element of PD must contain a component, which is not in any of the updated minimal conflicts. Such partial diagnoses are removed in line 12 and cannot be re-computed in further iterations.

Consequently, GETPREFERREDPD terminates and returns the only remaining partial diagnosis δ^*, which is consistent with all positive and negative measurements. □

Theorem 1. FINDDIAGNOSIS *always terminates and returns a preferred diagnosis Δ^* given correct answers of an oracle.*

Proof. First we show that a set of components Δ^* hits every conflict in $\mathbf{CS}$ and then that Δ^* is subset-minimal.

For any set of minimal conflicts C returned by FINDCONFLICTS the function GETPREFERREDPD always returns the preferred partial diagnosis δ^* for an updated $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$. The addition of δ^* to Δ^* (line 5) ensures that none of the minimal conflicts C will be returned by FINDCONFLICTS in the next iteration. Consequently, Algorithm 1 terminates as soon as Δ^* hits every $CS \in \mathbf{CS}$.

Furthermore, Δ^* is subset-minimal since (a) Δ^* comprises only components of some minimal conflict $CS \in \mathbf{CS}$ (by definition of GETPREFERREDPD) and (b) every $CS \in \mathbf{CS}$ is hit by Δ^* only once. The latter is due to fact that GETPREFERREDPD returns only if δ^* is the only diagnosis for the updated set of minimal conflicts C and $(\text{SD}, \text{COMPS}, \text{OBS}, P, N)$. Consequently, $|CS| = 1$ for every $CS \in C$. Otherwise, there would be another partial diagnosis in PD and GETPREFERREDPD would continue. □

4 Experimental Evaluation

We evaluated our method on two sets of benchmark problems: (a) the ontologies of the OAEI Conference benchmark as used in [Shchekotykhin *et al.*, 2014], (b) the systems of the DX Competition (DXC) 2011 Synthetic Track. As the main performance measure we use the wall clock time to find the preferred diagnosis. The oracle's deliberation time to answer a query was assumed to be independent of the query as done in [Shchekotykhin *et al.*, 2014]. In addition, we report how many queries (#Q) were required to find the preferred diagnosis and how many statements (#S) were queried.

We compared the following strategies:

1. INV-HS-DFS: The Inverse-HS-Tree method proposed in [Shchekotykhin *et al.*, 2014] which computes diagnoses using Inverse QuickXplain and builds a search tree in depth-first manner to find additional diagnoses.
2. MXP-HS-DFS: Our proposed method which uses MERGEXPLAIN to find a set of conflicts (FINDCONFLICTS) and a depth-first variant of Reiter's Hitting-Set-Tree algorithm [Reiter, 1987] to find partial diagnoses based on the found conflicts (FINDPDS).

For both strategies, we set the number of diagnoses n that are used to determine the optimal query to 9 as done in [Shchekotykhin *et al.*, 2014], and used the best-performing Entropy strategy for query selection (GETQUERY). We did not set a limit k on the number of conflicts to search for during a single call of MERGEXPLAIN. For the ontology benchmark, the failure probabilities used by the Entropy strategy are predefined. For the DXC problems, we used random probabilities and added a small bias for the actually faulty components to simulate partial user knowledge about the faulty components. The components were ordered according to the probabilities, which is advantageous for the conflict detection process for both tested algorithms.[3] To simulate the oracle, we implemented a software agent that knew the preferred diagnosis in advance and answered all queries accordingly. All tests were performed on a modern laptop computer. The algorithms were implemented in Java. Choco was used as a constraint solver and HermiT as Description Logic reasoner.

Problem Characteristics: Table 1 shows the characteristics of the ontology benchmarks. This evaluation scenario is designed to verify whether our method is applicable to problems for which the consistency checking is beyond NP. Therefore, we selected a set of hard cases for which the problem of consistency checking is at least EXPTIME-complete.

Since no pre-defined preferred diagnoses exist for this benchmark, we randomly selected one of the diagnoses as the preferred one and repeated the process 100 times – each time with a randomly chosen preferred diagnosis – to factor out random effects. In Table 1 we report the description logic (DL) used to formulate the ontology, the number of axioms (#A) in the knowledge base that were used as the possibly faulty components in the diagnosis process, and the average size of the preferred diagnoses ($|\Delta^*|$).

The characteristics of the DX Competition problems are given in Table 2. For each system 20 scenarios are given, each with a pre-specified injected fault consisting of several components. These faults correspond to our preferred diagnoses. Each of the 20 diagnosis scenarios was run 5 times to factor out possible effects resulting from the randomized fault probabilities. Overall, we therefore performed 100 runs for each tested system.[4] We encoded the scenarios as CSP problems and report the number of constraints (#C) and variables (#V) in Table 2. Furthermore, we list the range of the sizes of the injected faults (#F) per system and the corresponding average size of the found preferred diagnoses ($|\Delta^*|$). For some systems $|\Delta^*|$ can be smaller than the size of #F because some predefined injected faults were non-minimal.

Ontology	DL	#A	$\lvert\Delta^*\rvert$
ldoa-sof-ctool	$\mathcal{SHIN}^{(D)}$	402	16.8
ldoa-cmt-ekaw	$\mathcal{SHIN}^{(D)}$	338	22.4
mpso-ctool-ekaw	$\mathcal{SHIN}^{(D)}$	458	17.3
opt-sof-ekaw	$\mathcal{SHIN}^{(D)}$	467	22.9
opt-ctool-ekaw	$\mathcal{SHIN}^{(D)}$	340	16.9
ldoa-sof-ekaw	$\mathcal{SHIN}^{(D)}$	487	15.3
csa-sof-ekaw	$\mathcal{SHIN}^{(D)}$	491	16.1
mpso-sof-ekaw	$\mathcal{SHIN}^{(D)}$	491	22.3
ldoa-cmt-edas	$\mathcal{ALCOIN}^{(D)}$	434	1.5
csa-sof-edas	$\mathcal{ALCHOIN}^{(D)}$	860	1.0
csa-edas-iasted	$\mathcal{ALCOIN}^{(D)}$	885	8.3
ldoa-ekaw-iasted	$\mathcal{SHIN}^{(D)}$	629	9.7
mpso-edas-iasted	$\mathcal{ALCOIN}^{(D)}$	1,152	16.4

Table 1: Characteristics of the ontology benchmarks.

System	#C	#V	#F	$\lvert\Delta^*\rvert$
74182	19	28	4 - 5	4 - 5
74L85	33	44	1 - 3	1 - 3
74283	36	45	2 - 4	2 - 4
74181	65	79	3 - 6	3 - 6
c432	160	196	2 - 5	2 - 5
c499	202	243	10 - 15	10 - 15
c880	383	443	20 - 25	20 - 25
c1355	546	587	12 - 17	12 - 17
c1908	880	913	22 - 63	9 - 34
c2670	1,193	1,502	79 - 107	4 - 23
c3540	1,669	1,719	9 - 14	9 - 14
c5315	2,307	2,485	79 - 155	19 - 64
c7552	3,515	3,720	57 - 113	13 - 40

Table 2: Characteristics of the DXC benchmarks.

Results – Ontologies: The results for the ontologies are shown in Table 3. In terms of the computation times MXP-HS-DFS leads to a substantial speedup in all tests. The runtime improvements range from 51% for the two simplest ontologies to 98% for the most complex one, for which the calculation time could be reduced from 16 minutes to 23 seconds. On average the improvements are as high as 83%.

Looking at the number of required interactions and queried statements, we can observe that in particular for the most complex problems our method is advantageous as well, i.e., we ask fewer queries which involve fewer statements. For some ontologies, however, using partial diagnoses requires the user to answer more questions. The computation time to determine these questions is significantly lower though.

Results – DXC Benchmarks: Table 4 shows the results for the DXC problems. The results corroborate the observations made for the ontologies. Except for the tiny problems which can be solved in fractions of a second in either case,

[3] Without these slightly higher probabilities for the actually faulty components the absolute running times are higher for all algorithms. The relative improvements however remain very similar.

[4] The system c6288 could not be tested because the used Choco solver did not return a result for any single instance of this system.

Ontology	Inv-HS-DFS			MXP-HS-DFS		
	Time	#Q	#S	Time	#Q	#S
ldoa-sof-ctool	21.6	7.4	12.1	4.2	8.7	12.6
ldoa-cmt-ekaw	32.0	12.2	14.3	4.0	9.6	14.8
mpso-ctool-ekaw	32.5	10.6	13.2	2.9	7.1	10.9
opt-sof-ekaw	47.6	11.3	15.5	7.1	14.9	14.9
opt-ctool-ekaw	13.1	6.9	8.2	2.5	7.0	7.0
ldoa-sof-ekaw	39.0	10.3	13.8	4.2	8.5	14.2
csa-sof-ekaw	44.6	9.6	16.2	4.5	14.4	15.7
mpso-sof-ekaw	88.1	14.7	22.0	6.1	10.7	19.0
ldoa-cmt-edas	0.9	1.0	1.0	0.4	1.0	1.0
csa-sof-edas	1.6	1.5	2.5	0.8	1.5	2.5
csa-edas-iasted	138	6.7	10.4	20.5	5.5	10.2
ldoa-ekaw-iasted	96.7	9.5	16.0	11.0	8.6	13.5
mpso-edas-iasted	963	12.2	18.8	23.4	8.9	16.6

Table 3: Results for ontologies. Time is given in seconds. #Q: Avg. nb. of queries. #S: Avg. nb. of queried statements.

System	Inv-HS-DFS			MXP-HS-DFS		
	Time	#Q	#S	Time	#Q	#S
74182	0.3	4.0	6.8	0.3	3.9	7.8
74L85	0.1	2.2	4.6	0.1	2.2	5.0
74283	0.3	4.1	8.4	0.2	4.2	11.4
74181	0.8	7.0	13.1	0.4	5.9	16.7
c432	1.6	9.1	18.0	0.5	6.2	18.5
c499	9.3	25.8	49.6	1.8	16.9	50.3
c880	36.5	36.4	70.7	9.1	28.0	84.0
c1355	71.9	79.8	167	12.7	31.5	116
c1908	146	106	230	46.4	44.7	163
c2670	31.8	7.7	15.7	7.2	7.0	18.8
c3540	1,081	247	458	239	41.6	159
c5315	1,528	87.9	181	217	44.4	143
c7552	-	-	-	2,446	72.5	283

Table 4: DXC results. Time is given in seconds. #Q: Avg. number of queries. #S: Avg. number of queried statements.

significant improvements in terms of the running times could be achieved with our method. The strongest relative improvement is at 86%; on average, the performance improvement is at 57%. For the systems that took longer than a second with Inv-HS-DFS the average improvement is as high as 77%. For the most complex system c7552 Inv-HS-DFS could not find the preferred diagnosis in 24 hours, because the computation of diagnoses took too long. Again, in terms of the number of required queries and queried statements, our method becomes advantageous when the problems are more complex.

Results – Alternative Strategies: As the performance of diagnosis algorithms depends on the problem characteristics, we tested two alternative ways of computing the diagnoses in addition to Inv-HS-DFS and MXP-HS-DFS: (a) a breadth-first variant of Inv-HS-DFS, and (b) a depth-first variant of Reiter's Hitting-Set-Tree [Reiter, 1987], which always computes all required conflicts to determine the diagnoses. For some of the ontology problems these variants worked slightly better than the original Inv-HS-DFS method, but in all test cases our MXP-HS-DFS approach was substantially faster than all other strategies. For the DXC benchmarks the depth-first variant of Reiter's HS-Tree performed worse than all other tested algorithms. The most complex system that it could solve in 24 hours was c1908 for which it already needed 1,845 seconds. In contrast our method MXP-HS-DFS only needed 46 seconds.

5 Related Works

The idea of using measurements in MBD has its roots in the landmark works of [Reiter, 1987] and [de Kleer and Williams, 1987]. The latter additionally suggest a query selection and generation method that was used and improved in numerous subsequent works including [Feldman *et al.*, 2010; Pietersma *et al.*, 2005; Gonzalez-Sanchez *et al.*, 2011; Siddiqi and Huang, 2011; Shchekotykhin *et al.*, 2012].

To generate such queries, typical sequential algorithms determine a set of diagnoses as a first step. In practical situations, however, this often cannot be done efficiently without additional knowledge. A number of sequential approaches were therefore proposed in the literature that rely on additionally available information about the underlying system. One option, for example, is to find a hierarchical abstraction of the diagnosed system [Chittaro and Ranon, 2004; Feldman and Van Gemund, 2006; Siddiqi and Huang, 2011] and then use specific methods to locate the possibly faulty components [Stumptner and Wotawa, 2001; Darwiche, 2003; Marques-Silva *et al.*, 2015; Metodi *et al.*, 2014]. Alternatively, in cases where many test cases are available, spectrum-based techniques can be applied to assess whether a component is faulty or not [Gonzalez-Sanchez *et al.*, 2011].

In contrast to these approaches, our method is domain-independent, does not depend on the presence of multiple test cases, and uses a problem decomposition approach inside MergeXplain that is not dependent on the existence of structural information about the system. Of course, if the structure is known, the performance of MergeXplain can be further increased by adapting the splitting strategy.

In more recent works, several researchers approached the diagnosis task by solving the dual problem. Different domain-independent methods were proposed for example in [Felfernig *et al.*, 2012; Stern *et al.*, 2012; Shchekotykhin *et al.*, 2014], which calculate diagnoses "directly", i.e., without computing conflict sets. This property allows dual algorithms (like Inv-HS-DFS) to find a diagnosis in a polynomial number of calls to a theorem prover. However, our results show that our method can outperform dual methods in both domains despite the need of computing minimal conflicts.

6 Conclusion

Interactive diagnosis approaches can be particularly useful in cases when many diagnoses exist. In our work we presented a novel approach to significantly speed up the process of determining the next best question to ask to the user by introducing the concept of partial diagnoses.

As a part of our future work we will investigate the value of incorporating additional information, e.g., the system's structure or prior fault probabilities of the components, when determining the set of leading diagnoses and will explore if such information can help us to generate more informative queries.

Acknowledgements

This work was supported by the Carinthian Science Fund (contract KWF-3520/26767/38701), the Austrian Science Fund (contract I 2144 N-15) and the German Research Foundation (contract JA 2095/4-1).

References

[Chittaro and Ranon, 2004] Luca Chittaro and Roberto Ranon. Hierarchical model-based diagnosis based on structural abstraction. *Artificial Intelligence*, 155(1):147–182, 2004.

[Darwiche, 2003] Adnan Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

[Davis, 1984] Randall Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(1–3):347–410, 1984.

[de Kleer and Williams, 1987] Johan de Kleer and Brian C Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[de Kleer *et al.*, 1992] Johan de Kleer, Alan K Mackworth, and Raymond Reiter. Characterizing Diagnoses and Systems. *Artificial Intelligence*, 56(2-3):197–222, 1992.

[de Kleer, 1992] Johan de Kleer. Readings in model-based diagnosis. chapter Focusing on Probable Diagnosis, pages 131–137. 1992.

[Feldman and Van Gemund, 2006] Alexander Feldman and Arjan Van Gemund. A two-step hierarchical algorithm for model-based diagnosis. In *AAAI '06*, pages 827–833, 2006.

[Feldman *et al.*, 2010] Alexander Feldman, Gregory Provan, and Arjan Van Gemund. A model-based active testing approach to sequential diagnosis. *Journal of Artificial Intelligence Research*, 39:301, 2010.

[Felfernig *et al.*, 2004] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Stumptner. Consistency-based diagnosis of configuration knowledge bases. *Artificial Intelligence*, 152(2):213–234, 2004.

[Felfernig *et al.*, 2012] Alexander Felfernig, Monika Schubert, and Christoph Zehentner. An efficient diagnosis algorithm for inconsistent constraint sets. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 26(1):53–62, 2 2012.

[Gonzalez-Sanchez *et al.*, 2011] Alberto Gonzalez-Sanchez, Rui Abreu, Hans-Gerhard Gross, and Arjan J.C. van Gemund. Spectrum-based sequential diagnosis. In *AAAI '11*, 2011.

[Horridge *et al.*, 2008] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and Precise Justifications in OWL. In *ISWC '08*, pages 323–338, 2008.

[Junker, 2004] Ulrich Junker. QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In *AAAI '04*, pages 167–172, 2004.

[Marques-Silva *et al.*, 2013] Joao Marques-Silva, Mikoláš Janota, and Anton Belov. Minimal Sets over Monotone Predicates in Boolean Formulae. In *CAV '13*, pages 592–607, 2013.

[Marques-Silva *et al.*, 2015] João Marques-Silva, Mikoláš Janota, Alexey Ignatiev, and António Morgado. Efficient Model Based Diagnosis with Maximum Satisfiability. In *IJCAI '15*, pages 1966–1972, 2015.

[Metodi *et al.*, 2014] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. A novel sat-based approach to model based diagnosis. *Journal of Artificial Intelligence Research*, 51:377–411, 2014.

[Pietersma *et al.*, 2005] Jurryt Pietersma, Arjan J.C. van Gemund, and André Bos. A model-based approach to sequential fault diagnosis. In *AUTOTESTCON '05*, pages 621–627, 2005.

[Reiter, 1987] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[Rodler *et al.*, 2013] Patrick Rodler, Kostyantyn Shchekotykhin, Philipp Fleiss, and Gerhard Friedrich. RIO: minimizing user interaction in ontology debugging. In *RR '13*, pages 153–167, 2013.

[Shchekotykhin *et al.*, 2012] Kostyantyn Shchekotykhin, Gerhard Friedrich, Philipp Fleiss, and Patrick Rodler. Interactive ontology debugging: Two query strategies for efficient fault localization. *J. Web Semant.*, 12-13:88–103, 2012.

[Shchekotykhin *et al.*, 2014] Kostyantyn Shchekotykhin, Gerhard Friedrich, Patrick Rodler, and Philipp Fleiss. Sequential diagnosis of high cardinality faults in knowledge-bases by direct diagnosis generation. In *ECAI '14*, pages 813–818, 2014.

[Shchekotykhin *et al.*, 2015] Kostyantyn Shchekotykhin, Dietmar Jannach, and Thomas Schmitz. MergeXplain: Fast Computation of Multiple Conflicts for Diagnosis. In *IJCAI '15*, pages 3221–3228, 2015.

[Siddiqi and Huang, 2011] Sajjad Siddiqi and Jinbo Huang. Sequential diagnosis by abstraction. *Journal of Artificial Intelligence Research*, 2011.

[Stern *et al.*, 2012] Roni Stern, Meir Kalech, Alexander Feldman, and Gregory Provan. Exploring the Duality in Conflict-Directed Model-Based Diagnosis. In *AAAI '12*, pages 828–834, 2012.

[Stumptner and Wotawa, 2001] Markus Stumptner and Franz Wotawa. Diagnosing Tree-Structured Systems. *Artificial Intelligence*, 127(1):1–29, 2001.