Efficient Determination of Measurement Points for Sequential Diagnosis (Extended Abstract)*

Kostyantyn Shchekotykhin[1], Thomas Schmitz[2], and Dietmar Jannach[2]

[1] Alpen-Adria University Klagenfurt, Austria
kostyantyn.shchekotykhin@aau.at
[2] TU Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. Model-Based Diagnosis is a principled AI approach to determine the possible explanations why a system under observation behaves unexpectedly. For complex systems the number of such explanations can be too large to be inspected manually by a user. In these cases *sequential diagnosis* approaches can be applied. In order to find the true cause of the problem, these approaches iteratively take additional measurements to narrow down the set of possible explanations.
One computationally demanding challenge in such sequential diagnosis settings can be to determine the "best" next measurement point. This paper summarizes the key ideas of our recently proposed sequential diagnosis approach, which uses the newly introduced concept of "partial" diagnoses to significantly speed up the process of determining the next measurement point. The resulting overall reductions of the required computation times to find the true cause of the problem are quantified using different benchmark problems and were achieved without the need for any information about the structure of the system.[3]

Keywords: Model-Based Diagnosis, Sequential Diagnosis, Conflicts

1 Introduction

Model-Based Diagnosis (MBD) techniques are used to determine the possible causes when an observed system behaves unexpectedly. To find the possible causes, these approaches use knowledge about the system's expected behavior when all of its components work correctly. Since their development in the 1980s [1, 9, 7], MBD-based approaches were applied to many different problem settings like electronic circuits and software artifacts, e.g., java programs, knowledge bases, logic programs, ontologies, and spreadsheets.

When applying MBD for complex systems, the number of possible diagnoses can be too large to be checked individually by a user. For example, even when we

* 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).

[3] This paper summarizes the work from [14], presented at IJCAI'16.

only search for diagnoses up to a size of five, there are already 6,944 diagnoses for the system *c432 (scenario 0)* of the DX 2011 diagnosis competition benchmark.

Different approaches to deal with the problem exist. One option is to rank the diagnoses based on weights or fault probabilities and return only the highest ranked ones. However, these methods might be incomplete in cases when existing information is not sufficient to give a high rank to the correct explanation of a fault. Another possibility is to take additional measurements to discriminate between fault causes [7], thereby ensuring completeness. A recent work of Shchekotykhin et al. compared two different strategies for taking the next measurement, applied to the problem of ontology debugging [11]. The result indicates that the computation of a query, i.e., a suggestion for a next measurement to be made by a user, can be computationally expensive. Therefore, the approach of [11] first searches for a few *leading* diagnoses and then determines the optimal query to the user. However, for some real-world cases the computation of even a few leading diagnoses remains challenging [12].

In our work we address this problem setting and aim to reduce the diagnosis computation time. Specifically, the technical contribution of our work is the new notion of "partial" diagnoses, which can be efficiently computed using only a subset of the minimal conflicts. The partial diagnoses are then used to determine the best next query, i.e., we determine the best possible partitioning of the partial diagnoses, which typically form a smaller search space than in the original problem setting. In [14] we proved that our 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 all kinds of systems without exploiting problem-specific structural characteristics.

2 Sequential diagnosis

Our technique is based on the approach to sequential (interactive) diagnosis developed in [7, 9]. A diagnosable system in this approach is represented by a *model*, which describes the normal behavior of a system in terms of components and relations between them. In many scenarios models are encoded using constants representing the components and first-order sentences representing the relations. A diagnosis problem arises when the observed behavior of the system – represented as a finite set of consistent first-order sentences – differs from the expected one, represented by the model. In this case, a diagnosis Δ corresponds to a set of components that, if assumed to be faulty, explains the observed misbehavior.

In our approach – as in many others – the computation of diagnoses is based on the concept of conflicts. Informally speaking a minimal conflict is an irreducible set of components that cannot all work correctly at the same time given the observations. To resolve a minimal conflict every diagnosis therefore needs to comprise at least one of its components. Given a method for computing minimal conflicts such as QuickXplain [5] or Progression [8], algorithms like HS-Tree [9] find *all diagnoses* $\mathbf{D}$ by enumerating all subset-minimal hitting sets of the set of *all minimal conflicts* $\mathbf{CS}$.

If an MBD system returns more diagnoses than can be manually inspected, additional information is required in order to find the so-called *preferred* diagnosis Δ^*, which corresponds to the set of actually faulty components. This information is usually specified by means of measurements expressed as first-order sentences [2, 7, 9]. However, it is often unclear which measurements must be taken to uniquely determine Δ^*. In order to find Δ^*, sequential methods ask a user or some oracle a number of queries. The answers to these queries provide measurements required to rule out irrelevant diagnoses [7, 11].[4] The problem in this context is to determine "good" measurement points and correspondingly construct a set of first-order sentences Q, called *query*. Given a set of diagnoses $D \subseteq \mathbf{D}$, queries are designed such that at least one element of D can be ruled out regardless of the answer. If more than one query is possible, the best one can be selected using strategies like split-in-half, entropy, or risk-optimization [7, 10].

3 Query Computation with Partial Diagnoses

Our algorithm operates on the basis of "partial" diagnoses, which can informally be defined as follows: Given a set of minimal conflicts $C \subseteq \mathbf{CS}$, a set of components δ is a *partial diagnosis* iff it is a subset-minimal hitting set of C. Our algorithm repeatedly searches for preferred partial diagnoses and thereby incrementally identifies the preferred diagnosis Δ^*. In contrast to existing sequential approaches, we do not compute *all* conflicts required to find a set of diagnoses D 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 MergeXplain method [13]. Then, we find a set of minimal hitting sets *for this subset of the conflicts*, which correspond to partial diagnoses.

In the first step of our overall diagnosis algorithm, we compute at most k minimal conflicts C. In case there are no conflicts, i.e., the provided system description is consistent with all observations and measurements, the algorithm returns $\Delta^* = \emptyset$ as a diagnosis. Next, for the minimal conflicts C it finds a set of partial diagnoses PD. If PD comprises only one partial diagnosis, then its only element δ^* is returned as the *preferred partial diagnosis*. Otherwise, the algorithm determines a query Q to discriminate between the elements of PD and provides it to the oracle. The query computation method internally uses 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 [4, 11] or a constraint solver when the problem is to diagnose digital circuits [7].

Given an answer of the oracle the algorithm adds the corresponding first-order sentences to the set of measurements. This update requires the set C to be

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

reviewed because some of its elements might not be minimal conflicts given the new measurements. Then, the set of partial diagnoses is updated by removing all elements of *PD* that are not partial diagnoses with respect to the updated set of minimal conflicts C. Finally, the algorithm recursively calls itself and continues the search until the preferred partial diagnosis is found.

Given a preferred partial diagnosis, we declare all its components as faulty and check whether the model is consistent with the observations. If this is the case, the set of all faulty components, corresponding to the preferred diagnosis, is returned. Otherwise, the algorithm starts searching for the next preferred partial diagnosis. The suggested algorithm is shown to be sound and complete given correct answers of an oracle to its queries.

Example 1. Let us consider the system 74L85, Scenario 10, from the DX Competition 2011 Synthetic Track (DXC 2011). There are three minimal conflicts: **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 **CS** is 14, i.e., |**D**|=14. The preferred diagnosis Δ^* as specified in the benchmark is {o1, z22}.

The interactive diagnosis process starts with the computation of a subset C of all minimal conflicts using MergeXplain, e.g., C={{o1}, {o2, z2, z22}}. We then compute the minimal hitting sets of C and partial diagnoses *PD*={{o1, o2}, {o1, z2}, {o1, z22}}. Next, we use *PD* to find the query o2 asking the user if component {o2} is faulty. Since o2 is correct, the user answers "no". Given the answer we update the conflicts in C, i.e., C={{o1}, {z2, z22}} as well as *PD*. From the latter we remove all elements that are no partial diagnoses for the updated set of C resulting in *PD*={{o1, z2}, {o1, z22}}.

In the second iteration we first search for new partial diagnoses. Since we have already found all partial diagnoses for the conflicts in C, *PD* remains unchanged and non-empty. Therefore, we compute a new query asking if {z22} is faulty and the user answers "yes". 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}}. The third iteration returns {o1, z22} as the preferred partial diagnosis δ^*, since no additional partial diagnosis can be found.

Next, the algorithm declares the components {o1, z22} as faulty and finds that this assumption explains the observed misbehavior. Therefore, it returns Δ^*={o1, z22} as the preferred diagnosis and ignores the third conflict in **CS**. As a result, only two queries were required to find the true diagnosis.

4 Experimental Evaluation

We evaluated our method on two sets of benchmark problems: (a) the ontologies of the OAEI Conference benchmark as used in [12], (b) the systems of the DXC 2011. 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 [12]. In addition, we evaluated how many queries were required to find the preferred diagnosis and how many statements were queried. We compared the following strategies:

1. Inv-HS-DFS: The Inverse-HS-Tree method proposed in [12] which computes diagnoses using Inverse QuickXplain and builds a search tree in depth-first manner to find additional diagnoses.
2. Inv-HS-BFS: A breadth-first variant of Inv-HS-DFS, similar to [3].
3. QXP-HS-DFS: A depth-first variant of Reiter's Hitting-Set-Tree algorithm [9] using QuickXplain to find all conflicts required for complete diagnoses.
4. MXP-HS-DFS: Our proposed method which uses MergeXplain to find a set of conflicts and a depth-first variant of Reiter's Hitting-Set-Tree algorithm [9] to find partial diagnoses based on the found conflicts.

We compared our approach MXP-HS-DFS to these other three, because the performance of each of them highly depends on the problem characteristics. Overall, we expect the Inverse-HS-Tree methods to be faster than QXP-HS-DFS for most of the tested problems. For all strategies, we set the number of diagnoses n that are used to determine the optimal query to 9 as done in [12], and used the best-performing Entropy strategy for query selection. We did not set a limit 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 all tested algorithms. 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.

For the ontologies the runtime improvements of our approach compared to the fastest of the other ones range from 28% for one of the simplest ontologies to 93% for the most complex one, for which the calculation time could be reduced from 6 minutes to 23 seconds. On average the improvements are as high as 80%. Looking at the number of required interactions and queried statements, our method is also advantageous for the most complex problems. For some ontologies, however, using partial diagnoses requires the user to answer more questions.

The results for the DXC problems corroborate these observations. Except for the tiny problems, which can be solved in fractions of a second with all approaches, significant improvements in terms of the running times could be achieved with our method compared to all other approaches. For the DXC problems, Inv-HS-DFS was the fastest of the other approaches. The strongest relative improvement of our approach compared with this method is at 86%; on average, the performance improvement is at 58%. For those systems where the computation times of Inv-HS-DFS were more than one second, the average improvement is as high as 77%. Some of the benchmark problems could not be solved by some of the other approaches at all in 24 hours. QXP-HS-DFS, which was the fastest of the other methods for the ontologies, could, for example, not find the preferred diagnosis for the three most complex systems. The most complex system could not be diagnosed in 24 hours by any of the other approaches, while our new approach MXP-HS-DFS finished in about 40 minutes.

5 Conclusion

In this work we presented a new approach to speed up the sequential diagnosis process. Our approach uses the new concept of partial diagnoses to reduce the computation time needed to determine the next best question to ask to the user. This can be particularly useful in cases when many conflicts exist.

In our future work we plan to additionally speed up the process of determining the leading diagnoses by incorporating additional information, e.g., the system's structure or prior fault probabilities of the components, and will explore if such information can help us to generate more informative queries.

References

1. Davis, R.: Diagnostic reasoning based on structure and behavior. Artif. Intell. 24(1–3), 347–410 (1984)
2. Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M.: Consistency-based diagnosis of configuration knowledge bases. Artif. Intell. 152(2), 213–234 (2004)
3. Felfernig, A., Schubert, M., Zehentner, C.: An efficient diagnosis algorithm for inconsistent constraint sets. AI EDAM 26(1), 53–62 (2 2012)
4. Horridge, M., Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. In: ISWC '08. pp. 323–338 (2008)
5. Junker, U.: QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In: AAAI '04. pp. 167–172 (2004)
6. de Kleer, J.: Readings in model-based diagnosis. chap. Focusing on Probable Diagnosis, pp. 131–137 (1992)
7. de Kleer, J., Williams, B.C.: Diagnosing multiple faults. Artif. Intell. 32(1), 97–130 (1987)
8. Marques-Silva, J., Janota, M., Belov, A.: Minimal Sets over Monotone Predicates in Boolean Formulae. In: CAV '13. pp. 592–607 (2013)
9. Reiter, R.: A Theory of Diagnosis from First Principles. Artif. Intell. 32(1), 57–95 (1987)
10. Rodler, P., Shchekotykhin, K., Fleiss, P., Friedrich, G.: RIO: minimizing user interaction in ontology debugging. In: RR '13. pp. 153–167 (2013)
11. Shchekotykhin, K., Friedrich, G., Fleiss, P., Rodler, P.: Interactive ontology debugging: Two query strategies for efficient fault localization. J. Web Semant. 12-13, 88–103 (2012)
12. Shchekotykhin, K., Friedrich, G., Rodler, P., Fleiss, P.: Sequential diagnosis of high cardinality faults in knowledge-bases by direct diagnosis generation. In: ECAI '14. pp. 813–818 (2014)
13. Shchekotykhin, K., Jannach, D., Schmitz, T.: MergeXplain: Fast Computation of Multiple Conflicts for Diagnosis. In: IJCAI '15. pp. 3221–3228 (2015)
14. Shchekotykhin, K., Schmitz, T., Jannach, D.: Efficient Sequential Model-Based Fault-Localization with Partial Diagnoses. In: IJCAI '16. (2016)