Memory-Limited Model-Based Diagnosis  
(Extended Abstract)*

Patrick Rodler  
University of Klagenfurt, Austria  
patrick.rodler@aau.at

Abstract

Model-based diagnosis is a principled and broadly applicable AI-based approach to tackle debugging problems in a wide range of areas including software, knowledge bases, circuits, cars, and robots. Whenever the sound and complete computation of fault explanations in a given preference order (e.g., wrt. cardinality or probability) is required, all existing diagnosis algorithms suffer from an exponential space complexity. This can prevent their application on memory-restricted devices and for memory-intensive problem cases. As a remedy, we propose RBF-HS, a diagnostic search based on Korf’s seminal RBFS algorithm which can enumerate an arbitrary fixed number of fault explanations in best-first order within linear space bounds, without sacrificing other desirable properties. Evaluations on real-world diagnosis cases show that RBF-HS, when used to compute minimum-cardinality fault explanations, in most cases saves substantial space while requiring only reasonably more or even less time than Reiter’s HS-Tree, one of the most influential diagnostic algorithms with the same properties.

1 Introduction and Preliminaries

1.1 Model-Based Diagnosis

Model-Based Diagnosis [Reiter, 1987] is a well-founded, principled and broadly applicable approach to detecting, finding and fixing faults in numerous types of systems, such as software [Hunt, 1998], recommender systems [Felfernig et al., 2007], spreadsheets [Jannach and Schmitz, 2016], ontologies [Shchekotykhin et al., 2012], knowledge bases [Rodler, 2015], hardware [Friedrich et al., 1999], circuits [de Kleer and Williams, 1987], robots [Zaman et al., 2013], scheduling problems [Rodler et al., 2021], cars [Sachenbacher et al., 1998], and aircrafts [Gorinevsky et al., 2002].

Technically, model-based diagnosis assumes a system (e.g., software, circuit, knowledge base, physical device) consisting of a set of components \( \text{COMPS} = \{ c_1, \ldots, c_n \} \) (e.g., lines of code, gates, axioms, physical constituents) which is formally described in some monotonic logical language. Besides any relevant general knowledge about the system, this system description \( \text{SD} \) includes a specification of the normal behavior (logical sentence \( \text{BEH}(c_i) \)) of all components \( c_i \in \text{COMPS} \) of the form \( \text{OK}(c_i) \rightarrow \text{BEH}(c_i) \). As a result, when assuming all components to be fault-free, i.e., \( \text{OK}(|\text{COMPS}|) := \{\text{OK}(c_1), \ldots, \text{OK}(c_n)\} \), conclusions about the normal system behavior can be drawn by means of theorem provers. When the real system behavior, ascertained through system observations and/or system measurements (stated as logical sentences \( \text{OBS} \) and \( \text{MEAS} \)), is inconsistent with the system behavior predicted by \( \text{SD} \), the normality-assumption for some components has to be retracted. We call \( (\text{SD}, \text{COMPS}, \text{OBS}, \text{MEAS}) \) a diagnosis problem instance (DPI).

1.2 Diagnoses

Given a DPI, one of the most central goals in model-based diagnosis is the localization of the actually faulty system components, e.g., the code lines that have to be modified in order for the software to produce the right output, or the parts of a car that have to be repaired or replaced in order for the car to start. One step to this end is the enumeration of sets of potentially faulty components, called diagnoses. Formally, a (minimal / minimum-cardinality) diagnosis is an (irreducible / minimal-cardinality) set of components \( \text{D} \subseteq \text{COMPS} \) such that \( \text{SD} \cup \text{OBS} \cup \text{MEAS} \cup \text{OK}(|\text{COMPS}| \setminus \text{D}) \cup \text{NOK}(|\text{D}|) \) is consistent where \( \text{NOK}(|X|) := \{\neg\text{OK}(c_i) | c_i \in X\} \). So, a diagnosis is a set of components whose abnormality would explain the observed incorrect system behavior. To deal with the potentially numerous diagnoses for a given system (cf., e.g., [Shchekotykhin et al., 2012]), it can often be pivotal to 1. ascertain diagnoses in order from most to least plausible, e.g., minimal cardinality or maximal probability first [de Kleer, 1991; Reiter, 1987] (best-first diagnosis computation), and/or 2. acquire additional knowledge about the system, e.g., in terms of measurements, to eliminate spurious diagnoses [de Kleer and Williams, 1987] (sequential diagnosis).

Note, the novel technique suggested in this work is applicable for both tasks 1 and 2 (see Sections 2 and 3).

1.3 Diagnosis Computation

Diagnoses are often computed with the aid of conflicts. A (minimal) conflict is an (irreducible) set of components \( \text{C} \subseteq \text{COMPS} \)
COMPS such that assuming all of them fault-free, i.e., OK(C), is inconsistent with the current knowledge about the system, i.e., SD∪OBS∪MEAS∪OK(C) ⊨ ⊥. Diagnoses and conflicts are related in terms of a hitting set property: A (minimal) diagnosis is a (minimal) hitting set of all minimal conflicts. (X is a hitting set of a collection of sets S iff X ⊆ \bigcup_{S \in S} S and X ∩ S ≠ ∅ for all S ∈ S.) For complexity and efficiency reasons, diagnosis computation is usually focused on minimal diagnoses only.

Given a DPI (SD, COMPS, OBS, MEAS), a generic (hitting-set-based) diagnosis search algorithm works as follows:

• Start with a queue including only the root node ∅.
• While the queue is non-empty and not enough minimal diagnoses have been found, poll the first node n from the queue and process it. That is, compute a label L for n, and, based on L, assign n (or potentially its successors) to an appropriate node class (e.g., solutions, non-solutions).
• Different specific diagnosis computation algorithms are obtained by (re)defining (i) the sorting of the queue and (ii) the node processing procedure (node labeling and assignment).

### 1.4 A Prominent Example: HS-Tree

An instance of an influential and important diagnostic search is HS-Tree [Reiter, 1987]. It uses a FIFO-queue (breadth-first search) and an initially empty list D to store the found minimal diagnoses, and defines node labeling and assignment as follows:

1. If n is a non-minimal diagnosis (superset of some already found minimal diagnosis in D) or a duplicate (set-equal to some other node in the queue), then it is labeled with × (leaf node; irrelevant node; discard n).
2. Else, if there is a minimal conflict C such that n ∩ C = ∅, then n is labeled by C (internal node). This results in |C| successor nodes of n that are added to the queue, each constructed as n ∪ {c_i} for all c_i ∈ C. Note that the computation of each conflict requires O(|COMPS|) theorem prover calls, and can be accomplished, e.g., by the QuickXplain algorithm [Junker, 2004; Rodler, 2022a].
3. Else, n is labeled with ✓ (leaf node; minimal diagnosis; add n to the list of solutions D).

After the tree is completed (queue is empty), D includes exactly all minimal diagnoses for the given DPI, sorted by cardinality. Other sortings of D (e.g., based on diagnosis probability) can be obtained by sorting the queue using suitable cost functions (uniform-cost HS-Tree [Rodler, 2015]).

### 1.5 Existing Diagnostic Algorithms

Desirable and often also necessary properties of diagnostic algorithms are that only and all minimal diagnoses are found (soundness and completeness), that diagnoses are enumerated in order as per some preference criterion, e.g., maximal probability or minimal cardinality (best-first property), and that the algorithm is applicable to any DPI regardless of the problem domain, used logic and chosen theorem prover (generality). Unfortunately, however, all existing diagnosis algorithms featuring these four properties require a worst-case exponential amount of memory. This can prevent their successful adoption for memory-intensive problem cases [Schchekotykhin et al., 2014] or for memory-limited (e.g., IoT) devices.

### 2 Contribution

#### 2.1 New Approach: RBFS-HS

As a remedy to this issue, we propose a diagnostic search called Recursive Best-First Hitting Set Search (RBFS-HS) based on Korf’s seminal RBFS algorithm [Korf, 1993]. RBFS is a path-finding search that implements a scheme that can be synopsized as

• (complete and best-first): always expand current globally best node while storing current globally second-best node,
• (undo and forget to keep space linear): backtrack and explore second-best node if none of the child nodes of best node is better than second-best,
• (remember utility of forgotten subtrees to keep the search progressing): before deleting a subtree in the course of backtracking, store cost of subtree’s best node,
• (restore utility at regeneration to avoid redundancy): when re-exploring a subtree, use this stored cost value to update node costs in the subtree.

To devise RBFS-HS, we first analyzed which general aspects make diagnosis searches different from path-finding searches. In this regard, we identified, e.g., the necessity of defining a suitable node labeling and assignment strategy, that solutions are sets and not paths, that multiple solutions are generally sought, that different conditions on the cost functions have to apply, and that certain provisions are necessary to guarantee diagnostic soundness and completeness. We then modified RBFS accordingly to account for all these differences.

So, roughly, RBFS-HS integrates the search strategy of RBFS with the general principles of hitting-set-based diagnosis searches discussed in Section 1. As a result, RBFS-HS is sound, complete, best-first, general, and linear-space; a combination of features no existing diagnostic technique offers. More specifically, RBFS-HS allows to generate an arbitrary fixed number of minimal diagnoses in best-first order within linear space bounds, can be used out of the box for diagnosis problems expressed in any monotonic knowledge representation language, and can operate with any theorem prover.

#### 2.2 Application Scope

Notably, RBFS-HS is applicable to optimal (minimal) hitting-set computation (e.g., [Gainer-Dewar and Vera-Licona, 2017]) in general. Due to the relevance of hitting-set computation to many other important fields, the application scope of RBFS-HS spans far beyond the domain of model-based diagnosis. In particular, it may find fruitful application for, e.g.,...
(Max)SAT [Davies and Bacchus, 2011], constraint satisfaction [Bailey and Stuckey, 2005] and optimization problems [Saikko, 2019], set-theoretic duality [Slaney, 2014], explainable AI [Ignatiev, 2020], social and life sciences [Amburg et al., 2021], as well as for solving a wide range of other NP-complete problems [Moreno-Centeno and Karp, 2013].

3 Evaluation

3.1 Domain

We conducted comprehensive experiments on 12 real-world DPIs from a collection of widely used benchmark problems in the knowledge-based systems domain (cf., e.g., [del Vescovo et al., 2010; Kalyanpur, 2006; Qi and Hunter, 2007; Rodler, 2022d; Shchekotykhin et al., 2012; Stuckenschmidt, 2008]). In this field, soundness and completeness are required to guarantee the localization of the actually faulty knowledge in often critical (e.g., medical) applications; generality is pivotal to deal with a myriad of different logics and theorem provers that are used to optimally trade off expressivity against reasoning complexity [Baader et al., 2007]; and best-first computation is desired to monitor the most relevant fault explanations in order to terminate the debugging early if the actual fault is recognized, and, moreover, can boost the overall diagnostic efficiency [Rodler, 2022c]. For these reasons, HS-Tree, described in Section 1, which is a state-of-the-art method featuring these properties, is the commonly used method in this application area.

3.2 Experiments

In our experiments, we thus compared RBF-HS against HS-Tree. We considered two algorithm application scenarios: single-shot and sequential diagnosis. In the single-shot tests, each algorithm had to compute the \( k \) best diagnoses for each DPI. In the sequential diagnosis [de Kleer and Williams, 1987] tests, per DPI, each algorithm had to compute \( k \) best diagnoses multiple times in an iterative diagnosis session, each time for a different version (including one more measurement) of the given DPI. Each session was executed until all but one minimal diagnosis were ruled out by the measurements; these were selected based on the well-known information gain heuristic (cf., e.g., [de Kleer and Williams, 1987; Rodler, 2017; Rodler and Schmid, 2018; Siddiqi and Huang, 2011]). We used \( k \in \{2, 6, 10, 20\} \), defined the “best” diagnoses to be the ones of minimal cardinality, and adopted Pellet [Sirin et al., 2007] as a theorem prover.

3.3 Results

Fig. 1 shows the results for the sequential diagnosis tests (we got almost identical results for the single-shot tests). The main insights are:

- Whenever a DPI was non-trivial to solve, RBF-HS traded space favorably for time compared to HS-Tree (blue bars higher than orange ones).
- Space savings (blue bars) of RBF-HS were significant, amounting to an avg. \( \times \) max. of 93% / 98% of the memory consumed by HS-Tree. Time overheads (orange bars) of RBF-HS, in contrast, remained reasonable in all cases.

![Figure 1: Experiment Results](image)

**Figure 1: Experiment Results:** (y-axis) memory reduction and time overhead factors of RBF-HS vs. HS-Tree. (x-axis) 12 studied DPIs (sorted from low to high space savings) for each number of computed diagnoses \( k \in \{2, 6, 10, 20\} \). Note: The full paper [Rodler, 2022c] provides much more comprehensive evaluation results.

- In 38% of the cases, RBF-HS exhibited both a lower runtime and a lower space consumption than HS-Tree (upward blue bars and downward orange bars). We even observed 85% runtime along with 98% memory savings in one case.
- Additional findings (not shown in Fig. 1) are:
  - RBF-HS scales well to large numbers of computed diagnoses and to problems involving high-cardinality minimal diagnoses [Shchekotykhin et al., 2014].
  - RBF-HS can deal with highly expressive knowledge representation languages, which makes it well suited, e.g., for ontology quality assurance and debugging [Kalyanpur, 2006; Meilicke, 2011; Rodler, 2015; Shchekotykhin et al., 2012], important research fields in the context of the Semantic Web [Berners-Lee et al., 2001].
  - In several cases, HS-Tree ran out of memory (32 GB) while RBF-HS could successfully solve the diagnosis problem under negligible memory requirements.

4 Conclusion

We have proposed a novel diagnostic search based on Korf’s seminal RBFS algorithm which gives theoretical guarantees (soundness, completeness, best-first property, generality, linear space complexity) no other diagnostic method does. In experiments on real-world cases, our approach proved to be significantly more efficient wrt. memory consumption and almost on par wrt. runtime, compared to a highly influential and widely used diagnosis algorithm with the same properties. Importantly, RBF-HS is applicable to any hitting-set problem, and thus has the potential to positively impact a number of important domains beyond diagnosis, such as (Max)SAT, (Max)CSP, explainable AI, life sciences, or NP-complete problem solving.

5 Remarks and Additional Resources

**Restricted Scope:** For brevity, this extended abstract outlines the contributions of the associated full paper [Rodler, 2022c] only partially. For instance, the full paper presents a second diagnostic algorithm, called HBF-HS (Hybrid Best-First Hitting Set Search). In a nutshell, it constitutes a generalization of RBS-HS which combines the latter with HS-Tree in order to optimize the diagnosis computation time (of RBF-HS) while still preserving problem solvability by preventing a memory overflow. Besides all complexity analyses...
and correctness proofs, the full paper moreover contains a much more comprehensive evaluation, including results for HBF-HS, experiments using a second measurement selection heuristic as well as an alternative diagnosis preference criterion (maximal probability), scalability tests, and a discussion of the performance of the suggested algorithms on particularly hard diagnostic problem cases.


Presentations: A short video presentation on this work can be found at https://slideslive.com/38964052, and animated slides can be accessed via http://isbi.aau.at/ontodebug/presentation_rbfhs_dx2020_FINAL.pptx.

Debugging Tool: It is ongoing work to integrate the novel RBF-HS and HBF-HS algorithms into the ontology debugging tool OntoDebug [Schekotihin et al., 2018a; Schekotihin et al., 2018b], an official and free plug-in for Protégé [Mussen, 2015], the currently most popular open-source integrated development environment for ontologies. Please check http://isbi.aau.at/ontodebug/ for further information.

Acknowledgments
This work was funded by the Austrian Science Fund (FWF), contract P-32445-N38. I thank Richard Korf for his seminal work [Korf, 1993] which inspired me to do this research.

References


[Zhao and Ouyang, 2015] Xiangfu Zhao and Dantong Ouyang. De-