The Thirteenth International Symposium
on Combinatorial Search (SoCS 2020)

Too Good to Throw Away: A Powerful Reuse Strategy for Reiter's Hitting Set Tree*

Patrick Rodler[1]
[1]University of Klagenfurt, Austria,
patrick.rodler@aau.at

Abstract

We propose DynamicHS, a variant of Reiter's influential HS-Tree algorithm for computing fault explanations for defective systems in a sound, complete and best-first way. DynamicHS is geared to sequential diagnosis, where additional information about the defective system is iteratively collected until the true fault explanation is isolated. The key idea is to reuse the search tree from one iteration and to appropriately adapt it for the next iteration based on the acquired information. DynamicHS preserves both the generality and all desirable properties of HS-Tree and clearly outperforms the latter in extensive experiments on real-world diagnosis problems.

Model-Based Diagnosis (Reiter 1987) assumes a system (e.g., software, circuit, knowledge base, physical device) consisting of a set *components* $\text{COMPS} = \{c_1, \ldots, c_n\}$ (e.g., lines of code, gates, axioms, physical constituents) which is formally described by means of some monotonic logical language. Beside any relevant general knowledge about the system, this *system description* SD includes a specification of the normal behavior (logical sentence $\text{BEH}(c_i)$) of all involved components c_i 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 behavior of the system can be drawn by means of logical theorem provers. When the real system behavior, ascertained through *system observations* and/or *system measurements* (stated as logical sentences OBS and MEAS), is inconsistent with the system behavior predicted by SD, the normality-assumption for some of the components has to be retracted. We call $\langle \text{SD}, \text{COMPS}, \text{OBS}, \text{MEAS} \rangle$ a *diagnosis problem instance (DPI)*. A *(minimal) diagnosis* is an (irreducible) set of components $\mathcal{D} \subseteq \text{COMPS}$ such that $\text{SD} \cup \text{OBS} \cup \text{MEAS} \cup \text{OK}(\text{COMPS} \setminus \mathcal{D}) \cup \text{NOK}(\mathcal{D})$ is consistent where $\text{NOK}(X) := \{\neg \text{OK}(c_i) \mid c_i \in X\}$. So, a diagnosis is a set of components whose abnormality would explain the observed incorrect system behavior.

Diagnosis Computation is often accomplished with the aid of conflicts. A *(minimal) conflict* is an (irreducible) set comp-

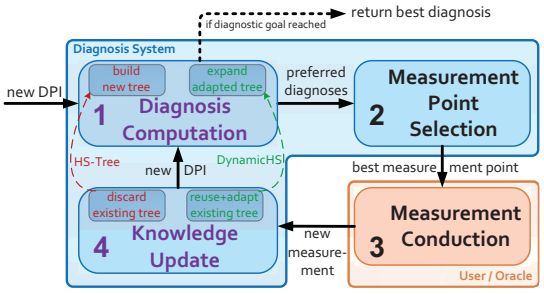

Figure 1: Sequential diagnosis: Phases impacted by DynamicHS (violet), idea of HS-Tree/DynamicHS (red/green).

onents $\mathcal{C} \subseteq \text{COMPS}$ such that assuming all of them fault-free, i.e., $\text{OK}(\mathcal{C})$, is inconsistent with the current knowledge about the system, i.e., $\text{SD} \cup \text{OBS} \cup \text{MEAS} \cup \text{OK}(\mathcal{C}) \models \bot$. 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 $\mathbf{S}$ iff $X \subseteq \bigcup_{S_i \in \mathbf{S}} S_i$ and $X \cap S_i \neq \emptyset$ for all $S_i \in S$.) For complexity and efficiency reasons, diagnosis computation is usually focused on minimal diagnoses only.

Sequential Diagnosis. In many cases, there is a substantial number of competing diagnoses. To isolate the *actual diagnosis* (which pinpoints the *actually* faulty components), *sequential diagnosis* (de Kleer and Williams 1987) methods gather additional system measurements (MEAS) to gradually refine the set of diagnoses. They can be characterized by a recurring execution of four phases (cf. Fig. 1): *(1)* computation of a set of preferred (e.g., most probable) minimal diagnoses, *(2)* selection of the best measurement point based on these, *(3)* conduction of measurement actions (by some user or oracle, e.g., an electrical engineer if a circuit is diagnosed), and *(4)* exploitation of the measurement outcome to update the system knowledge. That is, the DPI is modified (in terms of MEAS) between each two sequential diagnosis iterations. The goal in sequential diagnosis is to achieve sufficient diagnostic certainty (e.g., a single or highly-probable remaining diagnosis) with highest efficiency.

Reiter's HS-Tree (Reiter 1987) is one of the most popular and widely used algorithms for diagnosis computation. It has been employed in various domains such as for the debugging

*See (Rodler 2020) for the full paper, (Rodler 2015) for details and proofs, http://isbi.aau.at/ontodebug/evaluation for evaluation data, and (Schekotihin, Rodler, and Schmid 2018) for an interactive knowledge-base debugging tool incorporating DynamicHS.

of software, hardware and circuits, and is still state-of-the-art in diverse diagnosis areas. The reasons for its widespread adoption are that *(i) it is broadly applicable*, since all it needs is any sound and complete inference method for the logic used in SD, *(ii) it is sound and complete*, as it computes *only* and *all* minimal diagnoses, and *(iii) it computes diagnoses in best-first order* according to a given preference criterion.

Drawing on the hitting set property that relates conflicts with diagnoses (see above), HS-Tree builds a hitting set tree in breadth-first (minimum-cardinality solutions first) or uniform-cost (most-probable solutions first) way. Starting from a queue including only $\mathsf{n} = \emptyset$ (root), the first node n from the queue is successively labeled: *(A)* If n is a non-minimal diagnosis or a (set-equal) duplicate of some node, then it is labeled with $\times$ (leaf node; irrelevant node; discard). *(B)* Else, if n is not a hitting set of all minimal conflicts, then it is labeled by some minimal conflict $\mathcal{C}$ where $\mathsf{n} \cap \mathcal{C} = \emptyset$ (internal node). This results in $|\mathcal{C}|$ successor nodes of n, each constructed as $\mathsf{n} \cup \{c_i\}$ and connected to n by an edge labeled with c_i, for all $c_i \in \mathcal{C}$. Note that the computation of each conflict requires $O(|\text{COMPS}|)$ expensive theorem prover calls. *(C)* Else, n is labeled with ✓ (leaf node; minimal diagnosis; store). After the tree is completed, each minimal diagnosis corresponds to one tree node n labeled with ✓.

However, HS-Tree per se does not encompass any specific provisions for being used in a sequential way, where a new DPI is considered in each iteration. Already Raymond Reiter, in his seminal paper (Reiter 1987), asked:

> When new diagnoses do arise as a result of system measurements, can we determine these new diagnoses in a reasonable way from the (...) HS-Tree already computed in determining the old diagnoses?

As the first work, we shed light on this very question.

New Approach: DynamicHS. Due to the DPI update (addition of a measurement) per iteration throughout a sequential diagnosis session, HS-Tree can only be used in a *"discard+rebuild"* fashion, where a fresh search tree is generated from scratch in, and discarded after, each iteration. As the hitting set tree for a new DPI usually resembles the existing tree for the current DPI, this approach generally requires substantial redundant computations (costly theorem proving). Thus, we propose DynamicHS, which is based on a *"reuse+adapt"* principle and can be seen as "HS-Tree optimized for sequential diagnosis." It targets the improvement of HS-Tree by addressing the diagnosis computation and knowledge update phases in sequential diagnosis (cf. Fig. 1, green, red and violet colors). That is, (the knowledge inherent in) the search tree built for one DPI is reused and adapted for the new, updated DPI. As a result, one and the same search tree is used and gradually refined throughout the entire sequential diagnosis session. The key changes to HS-Tree are: storage of duplicate nodes and non-minimal diagnoses; pruning and relabeling (some) nodes at each iteration; replacement of (some) pruned nodes by adequate stored duplicates; regular freshness-checks of node-labeling conflicts.

Related Works. Literature covers a wide variety of diagnosis computation algorithms with different properties (e.g., complete vs. incomplete, best-first vs. any-first, etc.), motivated by different diagnosis problems, domains and requirements. As theoretical and empirical analyses attest, the best choice of algorithm depends on the particular problem (domain and requirements). Moreover, when new algorithms achieve performance improvements compared to existing ones, this often comes at the cost of losing desirable properties (e.g., completeness or the best-first property). Hence, it is particularly noteworthy that DynamicHS aims to improve HS-Tree *while preserving all its favorable properties*.

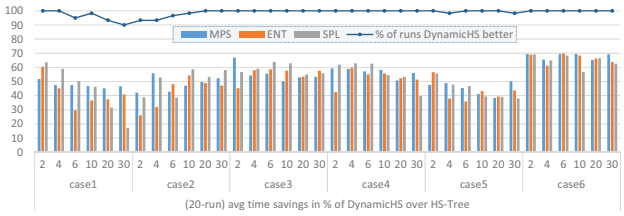

Figure 2: Experiment results.

Evaluation. We conducted extensive experiments on real-world cases from the knowledge-base (KB) debugging domain, where HS-Tree is the state-of-the-art diagnosis computation method. DynamicHS was compared to HS-Tree by running 20 sequential diagnosis sessions (stop criterion: single diagnosis remaining), each with a different random target solution, for each of the two algorithms, for each diagnosis case (inconsistent Description Logic KB), for each measurement point selection strategy (cf. Fig. 1, phase 2) among the three superior ones (MPS,ENT,SPL) in the field (Rodler and Schmid 2018), and for each number of preferred diagnoses in $\{2, 4, 6, 10, 20, 30\}$ that had to be computed in each sequential iteration (cf. Fig. 1, phase 1). The findings are: *(i)* In *all* scenarios, DynamicHS led to significant average time savings (of up to 70 %) compared to HS-Tree (bars in Fig. 2). *(ii)* DynamicHS outperformed HS-Tree in 98.75 % of all single sessions (blue line in Fig. 2). *(iii)* DynamicHS has exactly the same desirable properties—soundness, completeness, best-first property, general applicability—as HS-Tree (and exhibited similar space requirements in our tests).

Acknowledgments. This work was supported by the Austrian Science Fund (FWF), contract P-32445-N38.

References

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artif. Intell.* 32(1):97–130.

Reiter, R. 1987. A Theory of Diagnosis from First Principles. *Artif. Intell.* 32(1):57–95.

Rodler, P., and Schmid, W. 2018. On the impact and proper use of heuristics in test-driven ontology debugging. In *RuleML+RR 2018*.

Rodler, P. 2015. *Interactive Debugging of Knowledge Bases*. Ph.D. Dissertation, Univ. of Klagenfurt.

Rodler, P. 2020. Reuse, Reduce and Recycle: Optimizing Reiter's HS-Tree For Sequential Diagnosis. In *ECAI 2020*. (to appear).

Schekotihin, K.; Rodler, P.; and Schmid, W. 2018. OntoDebug: Interactive ontology debugging plug-in for Protégé. In *FoIKS 2018*.