And Yet Another Variant of Reiter's Complete On-the-fly Hitting Set Algorithm

Ingo Pill and Thomas Quaritsch
Institute for Software Technology
Graz University of Technology
Inffeldgasse 16b/II, 8010 Graz, Austria,
{ipill, quaritsch}@ist.tugraz.at

Abstract

Complementing recent research regarding the direct computation of diagnoses in theorem provers, complete hitting set algorithms are still an essential technique in the context of model-based diagnosis. Besides deriving diagnoses from available conflicts, they may even drive the search for those unsatisfiable cores in the problem description. Thus a series of algorithms have been emerging, consecutively aiming to lower required resources. In this paper, we show an extension to Greiner et al.'s variant of Reiter's hitting set algorithm, that like Wotawa's variant aims at minimizing the nodes and corresponding subset-checks needed for maintaining the DAG/tree encoding the search space exploration. First experimental results show the attractiveness of our algorithmic extension.

1 Introduction

With today's complex systems, the identification of root causes for encountered issues is an essential aspect of any development process, regardless of the system domain. Aimed at tackling this issue, the diagnosis community has been proposing a large variety of competing solutions based on different approaches and in a multitude of variants. Minimal hitting set (MHS) algorithms play a central role in several of these approaches, either as a last step "combining" intermediate results in the form of conflicts [de Kleer and Williams, 1987; de Kleer, 2011; Reiter, 1987], or, even driving the search space exploration and the computation of conflicts [Reiter, 1987]. Catching an essential connection between diagnoses and conflicts, they are of interest also when interleaving conflict and diagnosis computation [Stern *et al.*, 2012].

This resulted in a series of corresponding MHS algorithms, where, for instance, Greiner et al.'s HS-DAG [Greiner *et al.*, 1989] is an optimization (and correction) of Reiter's search space exploration idea that he defined in his formalizations regarding consistency-based diagnosis. However, there is still a lot of redundant work in HS-DAG's search, which is reflected, for instance, in the "reuse" of nodes when varying decision sequences lead to the same assumptions. Wotawa aimed with his variant HST [Wotawa, 2001] of Reiter's algorithm at minimizing the subset-checks required for building the tree by implementing a specific structure in the search. In this paper, we show how to implement a somehow related strategy by mimicking to some extent the divide-and-conquer-based strategy present in the Boolean algorithm [Lin and Jiang, 2003] (that repeatedly splits the search space into two parts; those solutions containing some chosen split element e and those not containing e) with the HS-DAG algorithm.

That is, our reasoning about the search space explored in some sub-DAG allows us to actually exclude specific elements from consideration in other sub-DAGs. Exploiting this reasoning we are able to achieve occasionally a significant reduction in the number of nodes derived (and the related subset- and consistency checks). In our first tests, we achieved a runtime speed-up, node reduction, and memory reduction of up to factors 3.77 / 4.15 / 2.72 respectively.

We present our work as follows. In Section 2 we cover the preliminaries, with Section 3 focusing on our extensions to HS-DAG. Following the experimental results reported in Section 4, we draw our conclusions and depict future work in Section 5. Related work is discussed throughout the paper where appropriate.

2 Preliminaries

In the context of model-based diagnosis, Reiter defined in [Reiter, 1987] a minimal hitting set (MHS) algorithm aiming at the computation of diagnoses as the minimal hitting sets of a diagnosis problem's conflicts.

Definition 1. *A* hitting set *for a set CS of sets C_i is a set $h \subseteq \bigcup_{C_i \in CS} C_i$ s.t. $\forall C_i \in CS : h \cap C_i \neq \emptyset$. h is minimal, iff there is no $h' \subset h$ that is a hitting set as well.*

In Reiter's formulations focusing on a system of connected components, he proposed an algorithm for deriving minimal hitting sets for some set *CS* containing the conflicts responsible for encountered inconsistencies between experienced and expected system behavior. His complete algorithm is based on a breadth-first exploration of the search space (exponential in the size of $\bigcup_{C_i \in CS}$) that is organized by maintaining a specific tree. While his algorithm was basically correct, it could miss diagnoses when non-minimal conflicts were used. Greiner et al. presented [Greiner *et al.*, 1989] a corrected version that furthermore uses a DAG instead of a tree. Their algorithm, HS-DAG, is defined for some ordered (i.e., consistently traversable) *CS*, whose C_is may be in arbitrary order or—more favorably—sorted by their cardinality. For our paper to be self-contained, we recap their algorithm in the following, adopting its description in [Greiner *et al.*, 1989]:

1. Let D represent a growing node- and edge-labeled DAG with some initial and unlabeled root node n_0. Proceed with Step 2.
2. Process the unlabeled nodes in D in a breadth-first order. To process a node n:
   (a) Define $h(n)$ to be the set of edge labels on the path in D from root node n_0 to node n ($h(n_0) = \emptyset$).
   (b) Iff for all $C_i \in CS$: $C_i \cap h(n) \neq \emptyset$, then label n with "✓". Otherwise label n with some C_j: C_j is the first set in *CS* s.t. $C_j \cap h(n) = \emptyset$.
   (c) If n is labeled with some $C_i \in CS$, generate for each $c_i \in C_i$ a new edge labeled with c_i. This edge leads to a new node n' with $h(n') = h(n) \cup \{c_i\}$. The new node n' will be processed (labeled and expanded) after all new nodes n_i in the same generation as n (s.t. $|h(n_i)| = |h(n)|$) have been processed.
3. Return the resulting DAG D.

To (correctly) compute only the *minimal* hitting sets of *CS*, Greiner et al. proposed the following pruning enhancements to the algorithm:

1. Reusing nodes: The algorithm will not always generate a new node m as a descendant of node n. There are two cases to consider:
   (a) If there is a node n' in *D* such that $h(n') = h(n) \cup \{c_i\}$, then let the edge labeled c_i originating from n point to n'. Hence, n' will have more than one parent.
   (b) Otherwise, generate a new node m as destination for the edge labeled c_i as described in the basic algorithm.
2. Closing: If there is a node n' such that $h(n') \subset h(n)$, and which is labeled with "✓", then close node n. Neither a label is computed for n, nor are any successor nodes generated.
3. Pruning: If a priorly unused set C_i is used to label a node, attempt to prune D as described in the following.
   (a) For nodes n' labeled with some $C_j \in CS$ such that $C_i \subset C_j$, relabel n' with C_i. Then, for any c_i in $C_j \setminus C_i$, the edge labeled c_i originating from n' is no longer allowed. The node connected by this edge and all of its descendants are removed, except for those nodes with another ancestor that is not being removed. Note that this step may eliminate the node which is currently being processed.
   (b) Interchange the sets C_j and C_i in *CS*. (Note that this has the same effect as eliminating C_j from *CS*.)

Note that the last rule (3) is relevant only if *CS* contains some sets C_i and C_j s.t. $C_i \subset C_j$ and the sets in *CS* are not sorted in respect of their growing cardinality. This is of specific interest when computing *CS* on-the-fly and the theorem prover returns not necessarily minimal conflicts.

In the context of our diagnosis examples in our evaluation, we follow Reiter's definitions [Reiter, 1987], and assume a diagnosis problem to be defined by a set of system components *COMP*, a system behavior description *SD*, and some actual system behavior observations *OBS*. For our examples, *SD* defines the correct behavior of the system in the form of logic sentences $\neg AB(c_i) \Rightarrow NominalBehavior(c_i)$, where assumption predicates $AB(c_i)$ for all $c_i \in COMP$ encode whether c_i behaves *abnormally* or not.

Definition 2. *A* conflict C *for (SD, COMP, OBS) is a set* $C \subseteq COMP$ *such that* $SD \cup OBS \cup \{\neg AB(c_i) | c_i \in C\}$ *is inconsistent. Iff there is no* $C' \subset C$*, such that* C' *is a conflict,* C *is a* minimal *conflict.*

Definition 3. *A* diagnosis *for (SD, COMP, OBS) is a subset-minimal set* $\Delta \subseteq COMP$ *such that* $SD \cup OBS \cup \{\neg AB(c_i) | c_i \in COMP \setminus \Delta\}$ *is consistent.*

Proposition 1. *(Theorem 4.4 in [Reiter, 1987])* $\Delta \subseteq COMP$ *is a* diagnosis *for (SD, COMP, OBS) iff* Δ *is a minimal hitting set for the collection CS of conflicts C for (SD, COMP, OBS).*

Consequently, a new label (conflict) for a node n can be computed by a theorem prover on-the-fly as an (ideally subset-minimal) unsatisfiable core in the assumption predicates for a consistency check of $SD \cup OBS \cup \{\neg AB(c_i) | c_i \in COMP \setminus h(n)\}$.

When HS-DAG's search for solutions is limited to some bound k regarding their cardinality, nodes with $|h(n)| = k$ need not be expanded. As a consequence, not all possible conflicts have to be computed to solve such problems.

As we will refer in the discussion of our HS-DAG variant to the Boolean algorithm [Lin and Jiang, 2003], we also rehearse it briefly in order for the paper to be self-contained:

The Boolean approach encodes *CS* as a Boolean formula in disjunctive normal form (DNF), with the individual $C_i \in CS$ as conjunction of the corresponding negated bits (propositions) for the components $c_i \in C_i$. To derive all minimal hitting sets for *CS*, a recursive function H containing five rules (considered in ascending order) derives from this *CS* formula another formula encoding the MHSs. That is, the result still needs some subset-checks (or the use of Boolean laws) in order to derive a canonical DNF where the conjuncts represent the individual *MHS*s. Assuming $\mathcal{C}$ a Boolean formula, e an atomic proposition with $\bar{e}$ denoting its negation, and $\bot/\top$ referring to $e \wedge \bar{e}/e \vee \bar{e}$, the function $H(\mathcal{C})$ is defined as:

R1: $H(\bot) = \top, H(\top) = \bot$;

R2: $H(\bar{e}) = e$;

R3: $H(\bar{e} \wedge \mathcal{C}) = e \vee H(\mathcal{C})$;

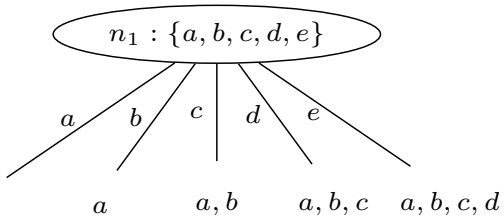
R4: $H(\bar{e} \vee \mathcal{C}) = e \wedge H(\mathcal{C})$;

R5: $H(\mathcal{C}) = e \wedge H(\mathcal{C}_1) \vee H(\mathcal{C}_2)$ for some *arbitrary* atomic proposition e present in $\mathcal{C}$, with
$\mathcal{C}_1 = \{c_i \mid c_i \in \mathcal{C} \wedge \bar{e} \notin c_i\}$ and
$\mathcal{C}_2 = \{c_i \mid \bar{e} \notin c_i \wedge (c_i \in \mathcal{C} \vee c_i \cup \{\bar{e}\} \in \mathcal{C})\}$.

Rule R5 encodes the algorithm's general strategy of how to conquer the search space by splitting the solution space into those solutions not containing split element e (and consequently pruning e from all conjuncts in $\mathcal{C}$) and those solutions including e (removing those conjuncts hit by e from further search). In [Pill and Quaritsch, 2012] we presented a variant where R5 is optimized for a bounded search (and also R4 is slightly improved), which we will use for our performance comparison later on.

3 HS-DAG with Reduced Conflicts

In this section, we show how to reduce redundancies in HS-DAG's search by narrowing the focus for a sub-DAG when it is clear that some solutions would be considered also by



Θ_C for the various sub-DAGs

Figure 1: Conflict reduction concept.

other sub-DAGs. Our basic reasoning is as follows: when expanding a node n (Step 2b), that is creating new edges originating in n, we create for each destination node n' a set $\Theta(n')$ of components for which we will not create new edges in the corresponding sub-DAG. $\Theta(n') = \Theta_C(n') \cup \Theta_N(n')$ for a node $n' \neq n_0$ consists of two subsets. The set $\Theta_C(n')$ of components c_i for which we created outgoing edges from node n already, and $\Theta_N(n') = \Theta(n)$ as the exclusion set inherited from n''s parent n, which we use to propagate our reasoning within a sub-DAG. For the root node n_0 we have $\Theta(n_0) = \Theta_C(n_0) = \Theta_N(n_0) = \emptyset$. In Figure 1 we illustrate the various $\Theta_C(n'')$s when assuming $\Theta(n_1) = \emptyset$ and creating the edges according to their label's appearance in n_1's label C.

Assume now that *CS* contains only subset-minimal conflicts, so that the pruning rule (3) (which we will consider later on) has no effect at all and is thus also not applied. Then we can define the following variant of HS-DAG:

Definition 4. *For our algorithmic variant HS-DAG-RC', besides HS-DAG's standard node-label, assume a further label $\Theta(n) \subseteq$ COMP. Let the algorithm be as the original HS-DAG, but without pruning rule 3 and the following redefinition of Step 2(c):*

If n is labeled with some $C \in$ CS, generate for each $c_i \in C \setminus \Theta(n)$ a new edge labeled with c_i. This edge leads to a new node n' with $h(n') = h(n) \cup \{c_i\}$, where $\Theta(n') = \Theta_C(n') \cup \Theta_N(n')$ with $\Theta_C(n') = \{c_j | c_j \in C$ and we already created an edge labeled c_j from $n\}$ and $\Theta_N(n') = \Theta(n)$. The new node n' will be processed (labeled and expanded) after all new nodes n_i in the same generation as n (s.t. $|h(n_i)| = |h(n)|$) have been processed.

The soundness and completeness of our variant HS-DAG-RC' can be easily seen.

Theorem 1. *For CS containing only subset-minimal sets, the algorithmic variant as of Definition 4 is complete. For any minimal hitting set Δ for CS, D contains some node n labeled ✓ such that $h(n) = \Delta$.*

Proof. (Sketch) Let us assume that there is some minimal hitting set Δ such that there is no node n with $h(n) = \Delta$. Now we know that HS-DAG is complete. As our optimizations so far only concern avoiding the construction of some edges in Step 2(c), these optimizations would have prevented the construction of the necessary node. It is however easily seen, that while we restrict some sequences of element choices that result in a certain combination $h(n) = \Delta$, we do not prevent all possible sequences. That is, for some given Δ and DAG D and starting with the root node n_0, we can always choose the "left-most" branch possible (removing the edge-label c_i from Δ), which has to be an allowed sequence due to the exhaustive nature of step 2(b) and minimizing Θ (i.e., there is no restriction regarding those c_i still left in Δ) along this path. □

Theorem 2. *For CS containing only subset-minimal sets, the algorithmic variant as of Definition 4 is sound. That is, $h(n)$ of any node n in D such that n is labeled with ✓ is indeed a minimal hitting set of CS.*

Proof. (Sketch) Step 2(b) ensures that $h(n)$ for any node labeled ✓ is indeed a hitting set, so that it remains to show that such a $h(n)$ is indeed subset-minimal. Let us now assume that there is some node n labeled ✓ such that $h(n)$ is not minimal. As $h(n)$ is non-minimal, due to Theorem 1, there would have to be some node n' labeled ✓ such that $h(n') \subset h(n)$. For such an n', the closing rule would however have closed n, so that there could not have been such a node n'. Thus for any node n labeled ✓, we indeed have that $h(n)$ is a subset-minimal hitting set for *CS*. □

Now that we have a complete and sound algorithm for subset-minimal conflicts, let us consider the case of non-minimal conflicts. While the respective ideas behind the pruning rule (3) and our reasoning do not interfere, the actual implementation of our reasoning in Step 2(c) does. This comes from the fact that the actually expanded edges interfere with the construction of the sub-DAGs via Θ_C. If an edge then gets pruned, some Θ_C and the Θs in its sub-DAG might need an update. Obviously this also means that some edges which some Θ prevented from being constructed, need to be created with the updated Θs. Our following variant of Definition 4 that includes an adapted *pruning* rule takes care of these issues.

Definition 5. *Let HS-DAG-RC be an algorithm as of Def. 4, but with an updated* pruning *rule 3 from HS-DAG, such that it extends HS-DAG's rule 3(a) as follows:*

Now, for all children n'' of n' update $\Theta_C(n'')$ to $\Theta_C(n'') \setminus (C_j \setminus C_i)$ and for all descendants n''' of some n'' propagate the update (most likely reflected in $\Theta_N(n''')$) accordingly. Then create for all n'' and n''' all the edges that are not avoided anymore (due to the updates to their Θs), and process the new nodes in a breadth first order (always choosing a node with the smallest $h(n)$) with Step 2 as usual.

Theorem 3. *The algorithm as of Def. 5 is complete and sound.*

Proof. It is easy to see that our extension to the pruning rule ensures that Θ and the corresponding node expansion is consistent with what would have been derived when using C_i instead of C_j in the first place. That is, the corresponding missing edge(s) and the potential sub-DAG(s) get constructed accordingly. Considering rule 3(b) we could even remove C_j from *CS*, which in the end would result in a final *CS* that contains only subset-minimal conflicts. Thus, completeness and soundness of HS-DAG-RC follow directly from completeness and soundness of HS-DAG-RC'. □

From an abstract point of view the way our reasoning affects the construction is similar to the reasoning behind Wotawa's idea in HST [Wotawa, 2001]. That is, we aim to avoid the construction of assumption combinations ($h(n)$) in a sub-DAG iff this $h(n)$ would be considered in other reasoning branches anyway. However, our underlying reasoning and the implementation differ significantly from HST.

Besides working on a DAG, our expansion is still based on a pruned version of a node's label (conflict), rather than a range within bounds propagated (and altered) when constructing HST's tree.

In some sense, our reasoning is more similar to a specific scenario in our Boolean algorithm variant that we presented in [Pill and Quaritsch, 2012]. This variant revised Rule 5 (see our brief description in the preliminaries) such that we consecutively choose as e the elements in a single conflict (for specific details please refer to Lemma 2 in [Pill and Quaritsch, 2012]). When iterating over a single $C \in \mathit{CS}$, these consecutive decisions the Boolean algorithm makes regarding the split elements reflect the structure of Θ for the various branches when HS-DAG-RC expands the same C. For the example in Figure 1, that is for $\mathit{CS}' = \{a, b, c, d, e\} \cup \mathit{CS}$, our Boolean algorithm variant would construct the formula $a \wedge H(\mathit{CS}^a) \vee b \wedge H(\mathit{CS}^b_a) \vee c \wedge H(\mathit{CS}^c_{a,b}) \vee d \wedge H(\mathit{CS}^d_{a,b,c}) \vee e \wedge H(\mathit{CS}^e_{a,b,c,d})$, where $\mathit{CS}^e_{a,b,c,d}$ means that we pruned from the conflicts in *CS* elements a, b, c, d and take the subset of pruned conflicts that is not hit by e. This perfectly resembles the effects of Θ_C as illustrated in Figure 1 and the edge label's inclusion in $h(n)$. While this is an interesting analogy specifically regarding the evaluation, please let us remind you at this point that the Boolean algorithm requires *CS* to be computed a priori, while HS-DAG (including our variant) can operate also on-the-fly, besides other important differences (a prominent one is the pruning).

4 Experimental Results

As reference implementation for HS-DAG, we used our Python implementation (CPython 2.7.1) that we used also for the diagnosis algorithm performance comparison in [Nica *et al.*, 2013]. For our HS-DAG-RC variant we adopted the implementation accordingly. A small change we made for both algorithms is the encoding of the work-list, that is the list of nodes to be processed. As evident in its definition, HS-DAG-RC might construct nodes with an $|h(n)|$ smaller than that of the currently expanded node n, due to the updates regarding Θ in the pruning rule. For an easily manageable node-processing list, we thus use a work-list where nodes are grouped by their $|h(n)|$. Hence, a node with the smallest $h(n)$ can be retrieved easily without the need to sort the list when adding a new node (as we would have to do for a monolithic list). As we experienced no penalty for the HS-DAG variant when using the same grouped list (a list of lists) instead of the original single monolithic list, the reported results for HS-DAG also use this type of worklist.

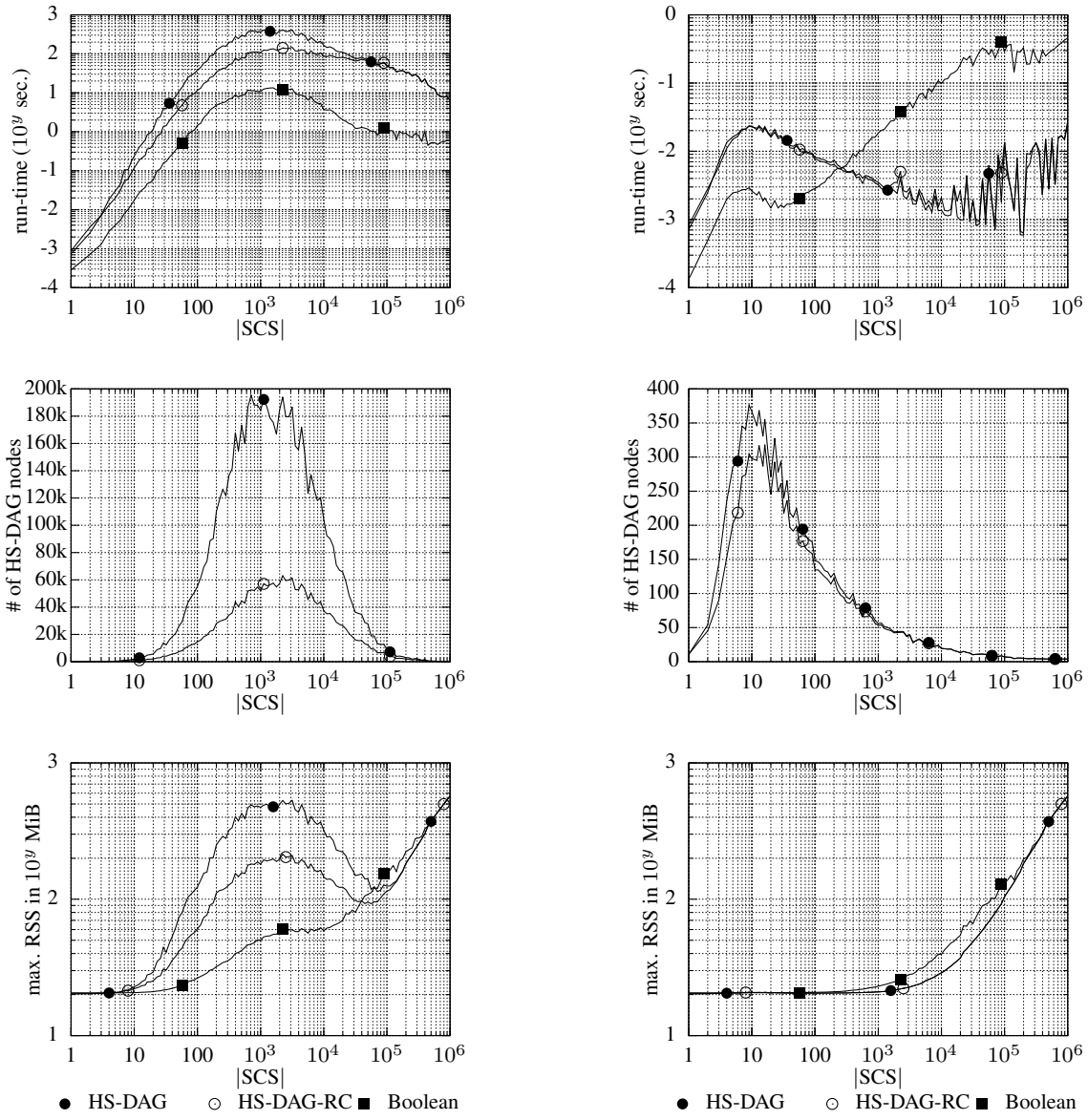
In the following, we report on results for two different scenarios. The first, artificial one we used also for evaluating our optimizations to the Boolean algorithm in [Pill and Quaritsch, 2012] (named TSA1 there). In this scenario, a $C \in \mathit{CS}$ contains elements drawn randomly from *COMP* such that every component $c_i \in \mathit{COMP}$ is included in some $C_i \in \mathit{CS}$ with a probability of 50 percent (no duplicate Cs in *CS* allowed).

Figure 2 reports on the run-time, node-amount and memory consumption (maximum resident size) regarding our experiments with $|\mathit{COMP}| = 20$, and a growing *CS*. We aimed at approximately 120 points equally distributed on the x-axis, which due to rounding resulted in 110 different values for $|\mathit{CS}| \approx 10^{6i/120}$ and $0 \leq i \leq 120$. For each such *CS*, we derived 10 samples and report the average values over the results for the individual samples. Furthermore, we ran both a bounded ($|\mathit{MHS}| \leq 3$) and an unbounded search for each sample. The results of the unbounded search are reported in Figure 2a, those for the bounded one in Figure 2b. Let us consider the unbounded search first. Regarding a comparison between HS-DAG and our variant HS-DAG-RC, we see a run-time advantage for the majority of our $|\mathit{CS}|$ range, with performance on par otherwise. For a $|\mathit{CS}| = 1000$ we experienced a run-time reduction of 70.6 percent, a node reduction of 71.7 percent and a reduction regarding max. RSS of approximately 61.6 percent. The maximum average reductions for any $|\mathit{CS}|$ were 73.5 percent, 75.9 percent and 63.3 percent respectively. For the bounded case we see virtually no difference between HS-DAG and HS-DAG-RC in respect of run-time and memory-consumption. Here the additional computations and variables for Θ seem to counterbalance the slight reduction in the number of nodes (18.9 percent for $|\mathit{CS}| = 10$). As the "pruning"-effect of Θ should increase with the DAG-depth, this is not entirely unexpected for this low cardinality limit of 3, that is, however, often a reasonably low bound in practice. Evidently, in those cases where the Boolean algorithm outperforms HS-DAG, HS-DAG-RC reduces the gap, but is closer to HS-DAG than to the Boolean algorithm.

Our second test scenario is based on conflicts created during specification diagnosis runs as described in [Pill and Quaritsch, 2013] (similar to those for Figure 1(b) in that paper). That is, briefly described, for some specification length in $\{50, 100, ..., 300\}$ we derived 10 random specifications φ in the Linear Temporal Logic (LTL) [Pnueli, 1977] as suggested in [Daniele *et al.*, 1999] with $N = \lfloor |\varphi|/3 \rfloor$ variables and a uniform distribution of LTL operators. We introduced triple faults as described in [Pill and Quaritsch, 2013] in order to derive φ_m from φ. Using the encoding from that paper we retrieved then an assignment for $\tau \wedge \varphi \wedge \neg\varphi_m$ that defines a variable trace τ of length $k = 100$ and loop-back time step $l = 50$. We then solved the diagnosis problem $E(\varphi_m, \tau)$ for a cardinality limit of 3 and recorded the conflicts derived.

Figure 3 reports on the run-time, node amount and memory consumption (maximum resident size) regarding our experiments with the *CS*s recorded for specification diagnosis runs. Again we ran both an unbounded and a bounded ($|\mathit{MHS}| \leq 3$) search for minimal hitting sets. While we observed a run-time reduction of 55.9 percent for $|\varphi| = 300$ in the unbounded case, the computation of Θ seems to entail a slight performance drawback (1.6 milliseconds instead of 1.3 milliseconds) for small samples with $|\varphi| = 50$. Nevertheless, we see also a significant reduction in the number of nodes, that is a reduction of 56.5 percent for $|\varphi| = 300$. The node reduction is however accompanied by an increase in the memory consumption from 40.4 MiB to 54.1 MiB, presumably related to managing Θ. An implementation optimized for low memory consumption could however drop the sets Θ_C and Θ_N after the construction of a subtree and recompute them if needed during a pruning/reconstruction step. The Boolean algorithm outperformed both, HS-DAG and our variant.

Like for the artificial scenario, in the bounded case we see virtually no difference in the run-time, and the reduction in the number of nodes (41.2 percent for $|\varphi| = 300$) is outweighed regarding memory consumption by the needs for Θ



(a) Unbounded

(b) Bounded, $|MHS| \leq 3$

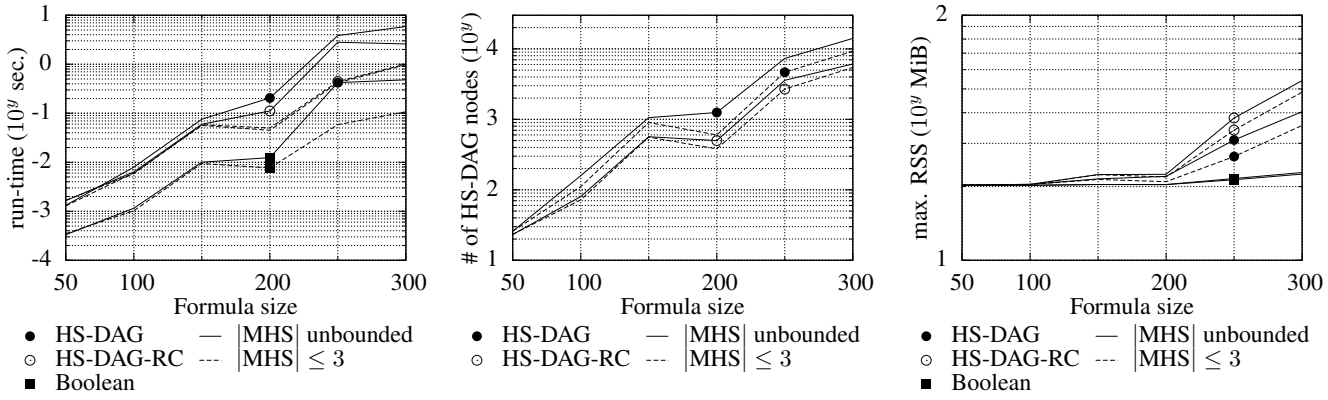
Figure 2: Performance results using random conflicts.

so that with 36.8 percent we have a similar memory penalty for $|\varphi| = 300$ as in the unbounded case (33.9 percent).

Summing up the reported results, we see an attractive performance advantage for our HS-DAG-RC against HS-DAG, specifically for the unbounded *MHS* search. For very small cardinality limits like 3 the still noticeable effects from the node reduction can be diminished by the needs for maintaining Θ, so that we end up with virtually no difference in the run-times but occasionally even experience a penalty in the memory consumption (for the LTL samples we had a penalty, while there was none for the random samples). Thus we see no reason why not to prefer our variant given the reductions in the run-time, node number, and memory consumption (by 73.5 / 75.9 / 63.3 percent, respectively by factors 3.77 / 4.15 / 2.72, for the random samples in the unbounded search) that we could achieve during our tests.

5 Discussion and Conclusions

Previous experiments [Pill *et al.*, 2011; Pill and Quaritsch, 2012] showed that the Boolean algorithm [Lin and Jiang, 2003] is a performant contender when striving for the computation of the complete set of minimal hitting sets. On the other hand it is missing an important feature present in HS-DAG: search space exploration guidance in respect of conflict set computation. As we found it intriguing that this algorithm can iteratively partition and prune the search space, we aimed at a similar concept for HS-DAG. That is retaining HS-DAG's guiding feature that enables an on-the-fly computation of *CS*, our variant HS-DAG-RC mimics the Boolean's divide-and-conquer strategy by removing already considered elements from the (sub-)problem at hand. As we saw, our updated node-expansion routine requires also a more complex pruning-/update function that ensures the tree's compliance with our conflict pruning rules regarding



(a) Run-time (b) Number of HS-DAG nodes (c) Maximum resident set size

Figure 3: Performance results using conflicts from LTL specification diagnosis.

the exclusion-set Θ.

Performance-wise, we could observe in our experiments benefits in the run-time and memory consumptions for our random and LTL specification diagnosis samples. Internally, the nodes constructed for the DAG could be reduced significantly.

While for bounded runs (i.e. limiting the maximum cardinality of solutions to some $|MHS|_{\max}$) our strategy occasionally induced some (minimal) performance and memory overhead compared to the simpler HS-DAG strategy, we expect the experienced benefits to grow with rising maximum cardinality. For unbounded searches computing *all* solutions to a given problem, the experienced savings could be as high as 50–70 percent regarding the run-time, 60–75 percent regarding the number of DAG nodes, and approximately 60 percent in respect of the memory consumption (maximum resident set size).

Future work will investigate the memory penalty occasionally experienced for low bounds in the search, as well as the impact of the more complicated pruning rule 3(a).

Acknowledgements

This work was supported by the Austrian Science Fund (FWF): P22959-N23 ("MoDiaForTeD"). The authors would like to thank Franz Wotawa for fruitful discussions and the anonymous reviewers for their valuable comments.

References

[Daniele *et al.*, 1999] M. Daniele, F. Giunchiglia, and M. Vardi. Improved automata generation for Linear Temporal Logic. In *Computer Aided Verification*, pages 249–260, 1999.

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

[de Kleer, 2011] J. de Kleer. Hitting set algorithms for model-based diagnosis. In *22nd Int. Workshop on the Principles of Diagnosis*, pages 100–105. 2011.

[Greiner *et al.*, 1989] R. Greiner, B. A. Smith, and R. W. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, 41(1):79–88, 1989.

[Lin and Jiang, 2003] L. Lin and Y. Jiang. The computation of hitting sets: review and new algorithms. *Information Processing Letters*, 86:177–184, 2003.

[Nica *et al.*, 2013] I. Nica, I. Pill, T. Quaritsch, and F. Wotawa. The route to success - a performance comparison of diagnosis algorithms. In *International Joint Conference on Artificial Intelligence*, pages 1039–1045, 2013.

[Pill and Quaritsch, 2012] I. Pill and T. Quaritsch. Optimizations for the Boolean approach to computing minimal hitting sets. In *20th European Conference on Artificial Intelligence*, pages 648–653, 2012.

[Pill and Quaritsch, 2013] I. Pill and T. Quaritsch. Behavioral diagnosis of LTL specifications at operator level. In *International Conference on Artificial Intelligence*, pages 1053–1059, 2013.

[Pill *et al.*, 2011] I. Pill, T. Quaritsch, and F. Wotawa. From conflicts to diagnoses: An empirical evaluation of minimal hitting set algorithms. In *22nd Int. Workshop on the Principles of Diagnosis*, pages 203–210, 2011.

[Pnueli, 1977] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[Reiter, 1987] R. Reiter. A theory of diagnosis from first principles. *Artif. Intelligence*, 32(1):57–95, 1987.

[Stern *et al.*, 2012] R. Stern, M. Kalech, A. Feldman, and G. Provan. Exploring the duality in conflict-directed model-based diagnosis. In *26th AAAI Conference on Artificial Intelligence*, 2012.

[Wotawa, 2001] F. Wotawa. A variant of Reiter's hitting-set algorithm. *Information Processing Letters*, 79:45–51, 2001.