Critical-Path Dead-End Detection vs. NoGoods: Offline Equivalence and Online Learning

Marcel Steinmetz and **Jörg Hoffmann**
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
{steinmetz,hoffmann}@cs.uni-saarland.de

Abstract

One traditional use of critical-path heuristic functions is as effective sufficient criteria for unsolvability. To employ this for dead-end detection, the heuristic function must be evaluated on every new state to be tested, incurring a substantial runtime overhead. We show herein that the exact same dead-end detector can be captured through a *nogood*, a formula ϕ^{OFF} computed once prior to search. This is mostly of theoretical interest, as ϕ^{OFF} is large. We obtain practical variants by instead incrementally generating a stronger nogood ψ, that implies ϕ^{OFF}, online during search, generalizing from already tested states to avoid future heuristic-function evaluations.

Introduction

The family of *critical-path* heuristic functions h^m, introduced in its general form by Haslum and Geffner (2000), has a long tradition of use as effective sufficient criteria for unsolvability. If $h^m(s) = \infty$, then the goal cannot be reached from s even when allowing to break up conjunctive subgoals into their *atomic subgoals*, namely the size-m subsets. This idea has its roots in the use of h^2 as encoded by the planning graph, Graphplan's mechanism for early termination on unsolvable tasks (Blum and Furst 1997). The idea persisted in delete relaxation heuristics (Bonet and Geffner 2001; Hoffmann and Nebel 2001), as well as some partial delete relaxation heuristics (Domshlak, Hoffmann, and Katz 2015), which identify a state to be a dead-end if and only if h^1 does. The idea recently culminated in the use of the generalized critical-path heuristic h^C (Hoffmann and Fickert 2015) – which reasons over arbitrary sets C of atomic subgoals, rather than all size-m ones – for conflict-based learning during forward search (Steinmetz and Hoffmann 2016).

For dead-end detection in forward search, $h^C(s)$ must be evaluated for every new state s, incurring a substantial runtime overhead, potentially a prohibitive one if C is large. Kolobov et al. (2012) introduced the idea to alleviate this through *nogoods*, formulas ψ^{SM} where $s \models \psi^{SM} \Rightarrow h^2(s) = \infty$. Steinmetz and Hoffmann (2016) adopted the same idea. Here, we make a more radical observation: as far as dead-end detection goes, h^C can be equivalently replaced by a nogood formula ϕ^{OFF} computed offline, prior to search, avoiding the computation of h^C completely.

This is mostly of theoretical interest though, as ϕ^{OFF} has size worst-case exponential in the size of the input planning task, a blow-up which does tend to occur in practice (although evaluating ϕ^{OFF} does tend to be faster than evaluating h^C when ϕ^{OFF} *can* be constructed). We obtain practical variants by instead incrementally learning a nogood during search. We maintain and incrementally weaken a stronger formula ψ^{CL} that implies ϕ^{OFF}. Future states s where $s \models \psi^{\text{CL}}$ need not be evaluated. This online nogood ψ^{CL} is an alternative to the nogoods ψ^{SM} previously introduced by Kolobov et al. As our experiments show, ψ^{CL} often generalizes better, avoiding more future evaluations of h^C. This tends to pay off, in particular for the large sets C of atomic subgoals underlying h^2.

Background

We consider STRIPS planning. A planning *task* is a tuple $\Pi = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ of *facts* $\mathcal{F}$, *actions* $\mathcal{A}$, *initial state* $\mathcal{I}$, and *goal* $\mathcal{G}$. An action $a \in \mathcal{A}$ is a tuple $(pre(a), add(a), del(a))$ where $add(a) \cap del(a) = \emptyset$. (Action costs are irrelevant to dead-end detection, so we assume unit action costs.)

Critical-path heuristics approximate goal distance through the relaxing assumption that, for achieving a conjunction G of facts (represented as a fact set), it suffices to achieve the most costly *atomic conjunction* contained in G. The set C of atomic conjunctions is a parameter.

Formally, for a conjunction of facts $G \subseteq \mathcal{F}$ and an action $a \in \mathcal{A}$, the *regression* of G over a is defined by $R(G, a) = (G \cup pre(a)) \setminus add(a)$ if $add(a) \cap G \neq \emptyset$ and $del(a) \cap G = \emptyset$. Otherwise, the regression is undefined, and we write $R(G, a) = \bot$. By $\mathcal{A}[G]$ we denote the set of actions a where $R(G, a) \neq \bot$. Let C be any set of conjunctions. The generalized critical-path heuristic $h^C(s)$ is defined through $h^C(s) = h^C(s, \mathcal{G})$ where

$$h^C(s, G) = \begin{cases} 0 & G \subseteq s \\ 1 + \min_{a \in \mathcal{A}[G]} h^C(s, R(G, a)) & G \in C \\ \max_{G' \subseteq G, G' \in C} h^C(s, G') & \text{else} \end{cases} \quad (1)$$

The computation of $h^C(s)$ is polynomial in $|C|$ and $|\Pi|$, yet incurs substantial runtime overhead in practice.

Offline NoGood h^C Representation

The basic idea to avoid h^C evaluations, already applied in the aforementioned prior works, is to identify *nogoods*, formulas ϕ over the facts $\mathcal{F}$ such that $s \models \phi \Rightarrow h^C(s) = \infty$. The nogoods are tested prior to the computation of h^C, and that computation is skipped if at least one nogood fires (is satisfied by the state at hand). In short, the nogoods serve as an easy-to-test sufficient criterion. We next observe that one can actually construct, before the search even begins, an *exact* nogood ϕ^{OFF}, where $s \models \phi^{\mathrm{OFF}} \Leftrightarrow h^C(s) = \infty$.

We have identified two alternative ways of constructing such a nogood ϕ^{OFF}, which differ in terms of formula structure and size. We detail in what follows the simpler formula, which turns out to be more useful in practice; we discuss the alternative at the end of this section.

Our construction is based on what we call *atomic regression traces*. These are sets of atomic conjunctions, gleaned from the recursion in Equation 1 by selecting a single G in the maximization (bottom case) and following all a in the minimization (middle case). This yields a disjunction of atomic conjunctions. We observe that every state along a path to the goal must satisfy every such disjunction, and that the states not satisfying at least one such a disjunction are exactly the dead-end states recognized by h^C.

Definition 1. *Let C be a set of conjunctions. A C-**atomic regression trace**, short C**ART**, is a subset σ of C so that*

(i) there exists $G \in \sigma$ with $G \subseteq \mathcal{G}$, and

(ii) for every $G \in \sigma$ and for every $a \in \mathcal{A}[G]$, there exists $G' \in \sigma$ with $G' \subseteq R(G, a)$.

We identify a CART σ with the disjunction $\bigvee_{G \in \sigma} G$ of its element conjunctions. Note that each CART is, hence, a DNF formula.

To see how conditions (i) and (ii) capture why h^C evaluates to ∞, consider a state s and an atomic conjunction $G \in C$. In the simplest case, $h^C(s, G)$ is ∞ because G is not true in s and there is no action that achieves G, i. e., if $G \not\subseteq s$ and $\mathcal{A}[G] = \emptyset$. If $G \subseteq \mathcal{G}$, then $\{G\}$ is a CART on its own: if a state does not satisfy G, there is no way in which G (and hence the goal) can be made true from that state. In contrast, for an atomic conjunction G with non-empty set of achievers, $h^C(s, G) = \infty$ depends on the reachability of other atomic conjunctions. For $h^C(s, G) = \infty$ to be true, the regression of G over each of its achievers must contain an atomic conjunction G' with $h^C(s, G') = \infty$. Condition (ii) ensures that a CART σ covers all achievers. But then, as $h^C(s, G) < \infty$ only if s supports at least one achiever, $h^C(s, G) < \infty$ entails that s satisfies σ:

Lemma 1. *For any state s, if $s \models \neg\sigma$, then $h^C(s, G) = \infty$ for every $G \in \sigma$.*

Proof. Assume that $h^C(s, G) < \infty$ for some $G \in \sigma$. Let $G_0 \in \sigma$ be an element of σ with minimal $h^C(s, G_0)$ value. If $G_0 \subseteq s$, then $s \models \sigma$ and there is nothing to prove. Assume for contradiction that $G_0 \not\subseteq s$. Consider the action a from the second case of Equation 1, in the computation of $h^C(s, G_0)$. By the definition of σ, there exists $G' \in \sigma$ with $G' \subseteq R(G_0, a)$. But then, $h^C(s, G_0) > h^C(s, R(G_0, a)) \geq h^C(s, G')$, which contradicts the selection of G_0. □

By Definition 1 (i), each CART contains at least one goal conjunction. Hence, by Lemma 1, the existence of an unsatisfied CART implies that $h^C(s) = \infty$. We next show that the opposite direction also holds:

Lemma 2. *If $h^C(s) = \infty$, then there exists a C-atomic regression trace σ so that $s \models \neg\sigma$.*

Proof. We construct σ through a simple recursive procedure. Initially, let $\sigma = \emptyset$. Starting with $\mathcal{G}$, there must exist $G \in C$ so that $h^C(s, G) = \infty$ and $G \subseteq \mathcal{G}$. Since $h^C(s, G) = \infty$, we must have $G \not\subseteq s$, and for every $a \in \mathcal{A}[G]$, we must have $h^C(s, R(G, a)) = \infty$. We add G to σ, and we recurse on $R(G, a)$ for all $a \in \mathcal{A}[G]$. We terminate the recursion when the selected G already is an element of σ (which happens at the latest when $\sigma = C$). The resulting σ satisfies Definition 1, and $s \models \neg\sigma$ by construction. □

Lemma 1 and Lemma 2 together give the desired exact characterization of h^C dead-end detection:

Theorem 1. *Let Σ be the set of all C-atomic regression traces. Define $\phi^{\mathrm{OFF}} := \bigvee_{\sigma \in \Sigma} \neg\sigma$. Then, for every state s, $s \models \phi^{\mathrm{OFF}}$ if and only if $h^C(s) = \infty$.*

The construction of ϕ^{OFF} requires the enumeration of all CARTs. Obviously, the number of CARTs is worst-case exponential in the size of the planning task. We designed two optimizations to alleviate this blow-up: (1) avoiding redundancies, and (2) exploiting mutex information.

Regarding (1), if σ' subsumes σ, i. e., $\sigma' \subseteq \sigma$, then $\neg\sigma$ can be equivalently removed from ϕ^{OFF} as all states satisfying $\neg\sigma$ necessarily also satisfy $\neg\sigma'$. Furthermore, instead of naïvely removing subsumed CARTs after ϕ^{OFF} was already fully generated, one can identify, during the generation process, CARTs that *will* be subsumed. Organizing the generation process as a tree search where tree leaves are CARTs, we do so via checking, prior to expanding an atomic subgoal G, whether G was explored on a previous search path already. If so, and if the prefix before G on that previous search path subsumes our current prefix, then the re-expansion of G is pruned. This disconsiders many CARTs long before they are generated.

Regarding (2), for the purpose of dead-end detection in forward search it is enough if $s \models \phi^{\mathrm{OFF}} \Leftrightarrow h^C(s) = \infty$ for those s that are actually reachable from the initial state. In particular, we may disregard states that violate mutex constraints. So let $\neg\sigma$ and $\neg\sigma'$ be two elements of ϕ^{OFF}. Then we can remove from σ any atomic conjunction $G \notin \sigma'$ that is mutual exclusive with all atomic conjunctions in $\sigma' \setminus \sigma$: if s satisfies the mutex constraints, s does not satisfy $\sigma \setminus \{G\}$, but $s \models G$, then $s \models \neg\sigma'$.

As advertised, we have also explored an alternative way to build an exact offline nogood formula. So far, when gleaning our regression traces from the recursion in Equation 1, we chose to select only a single G in the bottom-case maximization. The alternative is to select all these G, recursively constructing an AND/OR tree, and therewith a corresponding formula, over the atomic conjunctions. That formula is equivalent to ϕ^{OFF} as defined above, and it may be exponentially smaller. However, this alternative suffers from its complex formula structure, with arbitrarily deep nesting of conjunction and disjunction, in contrast to ϕ^{OFF} which is a disjunction of CNFs. This has two major practical implications. First, the AND/OR tree construction is not amenable to our subsumption pruning (1), and in our experiments was much less feasible than constructing ϕ^{OFF}, despite its potential size advantage. Second, the simple structure of ϕ^{OFF} – a disjunctive list of CNF nogoods – lends itself to constructing that formula incrementally, enabling online nogood learning as described in the next section, which tends to be much more practical than the offline construction anyway. We therefore consider, from here on, only ϕ^{OFF} as above.

Online NoGood Learning

Online nogood learning was first suggested by Kolobov et al. (2012) in the context of probabilistic planning. It constitutes a middle ground between evaluating h^C on all states encountered, and computing an offline nogood ϕ^{OFF} that replaces the evaluation of h^C completely (yet incurs prohibitive overhead in its own right). The idea is to incrementally learn a formula ψ where $s \models \psi$ *only if* $h^C(s) = \infty$; in contrast to ϕ^{OFF}, we forsake the "if" direction. Consequently, instead of replacing h^C completely, we only avoid some of its evaluations, namely those on states s that satisfy ψ.

The general framework for online nogood learning can be phrased as the incremental construction of an under-approximation ψ of ϕ^{OFF}, i. e., $\psi \Rightarrow \phi^{\text{OFF}}$. Start with $\psi := \bot$, the trivial under-approximation of ϕ^{OFF}. Then, during search, whenever $h^C(t) = \infty$ is computed, find a formula ψ' that (1) captures t, $t \models \psi'$; that (2) is weaker than ψ, $\psi \Rightarrow \psi'$ but $\psi \not\Leftarrow \psi'$; and that (3) still is an under-approximation of ϕ^{OFF}, $\psi' \Rightarrow \phi^{\text{OFF}}$. Replace ψ by ψ' and continue the search. Thanks to (3), ψ always remains an admissible pruning condition, thanks to (2) it becomes incrementally weaker (pruning more states). Thanks to (1), provided that ψ *generalizes* over t, i. e., that t is not the only state satisfying ψ, ψ may prune future evaluations of h^C.

The only previously known instantiation of this framework is *state minimization* (Kolobov, Mausam, and Weld 2012; Muise, McIlraith, and Beck 2012).[1] Given a new state t with $h^C(t) = \infty$, state minimization views t as a constraint $\bigwedge_{p \in \mathcal{F} \setminus t} \neg p$, and iteratively weakens that constraint by removing one fact from $\mathcal{F} \setminus t$, so long as $h^C(t) = \infty$ is preserved.[2] Once this minimization step is completed, ψ' is obtained as $\psi' := \psi \vee (\bigwedge_{p \in \mathcal{F} \setminus t} \neg p)$. We denote the nogood formula built in this manner by ψ^{SM}. Note that the minimization step is crucial as, without it, there wouldn't be any generalization. The smaller $\mathcal{F} \setminus t$ becomes, the more states will satisfy the new conjunction $\bigwedge_{p \in \mathcal{F} \setminus t} \neg p$.

The alternative online nogood learning method we propose here, *CART learning*, is obtained directly from Lemma 2. Given a new state t with $h^C(t) = \infty$, there must exist a C-atomic regression trace σ not touched by t. The proof of Lemma 2 is constructive, showing how to find such a σ. We then obtain ψ' simply as $\psi' := \psi \vee \neg\sigma$. We denote the nogood formula built in this manner by ψ^{CL}.

[1]Kolobov et al. and Muise et al. call this *state generalization*. We use the term state minimization instead as this method is closely related to techniques used under this name in property-directed reachability (Bradley 2011; Suda 2014).

[2]In Kolobov et al.'s original nogood learning method, state minimization is preceded by a non-trivial nogood *candidate* construction, involving reasoning over previously encountered goal trajectories. We skip this part, using the entire state t as candidate.

		h^1				h^2				ϕ^{OFF} built	
Domain	#	–	ψ^{SM}	ψ^{CL}	ϕ^{OFF}	–	ψ^{SM}	ψ^{CL}	ϕ^{OFF}	h^1	h^2
Unsolvable Benchmarks (Hoffmann, Kissmann, and Torralba 2014)											
3unsat	30	**15**	**15**	**15**	**15**	10	**15**	**15**	**15**	30	15
Mystery	9	2	2	2	1	**8**	**8**	**8**	0	1	0
UIPC'16 Benchmarks											
BagBarman	20	**8**	**8**	**8**	0	0	0	0	0	0	0
BagGripper	25	**3**	**3**	**3**	0	0	0	0	0	0	0
BagTransport	29	6	6	6	1	**16**	**16**	**16**	0	1	0
Bottleneck	25	20	20	20	10	19	**21**	**21**	0	10	0
CaveDiving	25	**7**	**7**	**7**	2	6	6	6	0	2	0
ChessBoard	23	**5**	**5**	**5**	4	4	4	4	0	4	0
Diagnosis	11	**5**	**5**	**5**	1	4	**5**	**5**	0	6	0
DocTransfer	20	7	6	7	1	**8**	7	**8**	0	2	0
NoMystery	24	**2**	**2**	**2**	**2**	**2**	**2**	**2**	0	17	0
Rovers	20	**7**	**7**	**7**	**7**	**7**	**7**	**7**	0	20	0
TPP	30	**16**	**16**	**16**	**16**	14	15	15	0	25	0
PegSol	24	**24**	**24**	**24**	0	**24**	22	22	0	0	0
PegSolRow5	15	**5**	**5**	**5**	3	4	4	4	1	3	1
SlidingTiles	20	**10**	**10**	**10**	**10**	**10**	**10**	**10**	0	10	0
Tetris	20	**5**	**5**	**5**	0	**5**	**5**	**5**	0	0	0
Unsolvable Resource-Constrained Benchmarks (Nakhost, Hoffmann, and Müller 2012)											
NoMystery	150	45	45	45	45	74	**81**	**81**	0	150	0
Rovers	150	5	5	5	5	66	**67**	**67**	0	150	0
TPP	25	6	6	6	0	4	**7**	**7**	0	0	0
$\sum$	695	203	202	203	123	285	302	**303**	16	431	16

Table 1: Coverage. Best results highlighted in **boldface**. "ϕ^{OFF} built" shows the number of benchmark instances where ϕ^{OFF} was constructed successfully.

Observe that ψ^{CL} is a sub-disjunction of ϕ^{OFF}, consisting of CARTs for the h^C dead-ends encountered so far. Observe furthermore that CART learning is inspective, based on an analysis of the *reasons* for $h^C(t) = \infty$, in contrast to state minimization which treats h^C like a blackbox. One advantage of this is that, in difference to state minimization, CART learning does not involve any intermediate reevaluations of h^C. On the other hand, ψ^{CL} (a disjunction of CNFs) has a more complex structure than ψ^{SM} (a DNF), making the formula itself more costly to evaluate. The more practically important advantage of ψ^{CL} over ψ^{SM} turns out to be better generalization, which makes sense given the inspective vs. blackbox nature of the two methods.

Experiments

Our implementation is in FD (Helmert 2006). As our techniques apply to dead-end detection, we focus on proving unsolvability. We run the UIPC'16 benchmarks;[3] those of Hoffmann et al.'s (2014) unsolvable benchmarks not used in UIPC'16; and *resource-constrained* benchmarks by Nakhost et al. (2012), with constrainedness set to be $\in \{0.5, 0.6, \ldots, 0.9\}$ so that the tasks are unsolvable. All experiments were run on a cluster of Intel Xeon E5-2650v3 machines, with runtime (memory) limits of 30 minutes (4 GB).

In what follows, we inspect the influence of ϕ^{OFF}, ψ^{CL}, and ψ^{SM} respectively on breadth-first search using either h^1 or h^2 for dead-end detection. We used the aforementioned optimizations only for the construction of ϕ^{OFF}, and not for ψ^{CL}. The mutex constraints were derived from FD's state-variable representation, i. e., that every state variable can have only one value at any time.

Consider Table 1. On the right, we see that constructing ϕ^{OFF} is feasible about 62% of the time for h^1, and is hardly ever feasible for h^2, exhibiting the mentioned blow-up. On

[3]We removed 9 instances of the Diagnosis domain because conditional effects were introduced during FD's grounding procedure, and our implementation currently does not support these.

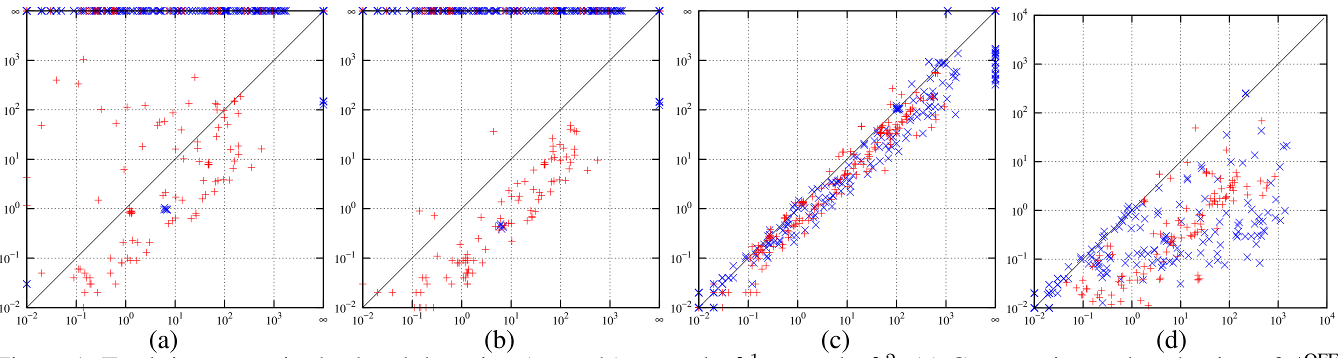
Figure 1: Total time spent in dead-end detection (seconds), + marks h^1, x marks h^2. (a) Construction and evaluation of ϕ^{OFF} (y-axis), vs. evaluation of h^1/h^2 (x-axis); (b) like (a) but without the construction. (c) Evaluation of h^1/h^2 plus refinement and evaluation of ψ^{CL} (y-axis), vs. always evaluating h^1/h^2 (x-axis); (d) like (c) but restricted to dead-ends recognized by h^1/h^2.

the left, we see that none of the nogood methods improves coverage for h^1. This is expected as computing h^1 is cheap so there is not much to gain by avoiding h^1 computations.

Matters are different for h^2, computing which is much more costly (on average in our experiments, 3 orders of magnitude slower). Observe first that, somewhat surprisingly perhaps – a common perception being that h^2 is way too costly to be recomputed on every state in a forward search – h^2 outperforms h^1 clearly here, mostly but not exclusively due to the resource-constrained benchmarks. More importantly for our work here, in contrast to h^1, online nogood learning *does* have a considerable beneficial effect on coverage for h^2. That said, there is little difference here between the previous technique ψ^{SM} and our new technique ψ^{CL}.

We now analyze the data in more detail, beyond the coarse measurement afforded by coverage. Figure 1 shows the effect of nogood learning on the time spent in dead-end detection. In (a) and (b), we see that, despite the size blow-up in offline nogood learning, on those instances where ϕ^{OFF} can be constructed, dead-end detection using ϕ^{OFF} is typically much more effective than with the equivalent critical-path heuristic. (a) shows that this is often so even when taking into account the time spent constructing ϕ^{OFF}; (b) shows that this is almost consistently so when disregarding that time, i. e., when considering the effort spent during search only.

Figure 1 (c) and (d) examine the runtime impact of online nogood learning with our new method ψ^{CL}. We see in (c) that ψ^{CL} improves performance almost consistently, and especially for h^2. We see in (d) that the improvement is quite strong – up to 4 orders of magnitude – when considering only those states recognized as dead-ends by h^1/h^2, i. e., those states where nogood learning may actually help (on all other states, it merely incurs an additional overhead).

Figure 2 (a) examines the power of generalization, in terms of the impact of online nogood learning on the number of h^1/h^2 evaluations. We consider only those states where nogood learning actually makes a difference – those recognized as dead-ends by h^1/h^2 – and we show the improvement factor, i. e., the ratio between the number of h^1/h^2 evaluations with vs. without nogood learning. We see that both ψ^{CL} and ψ^{SM} generalize quite well, yielding large improvement factors. Our new method ψ^{CL} is clearly superior, with improvement factors up to 3 orders of magnitude larger than those of ψ^{SM}. It reduces the number of evaluations by factors in the thousands for h^2, and up to millions for h^1.

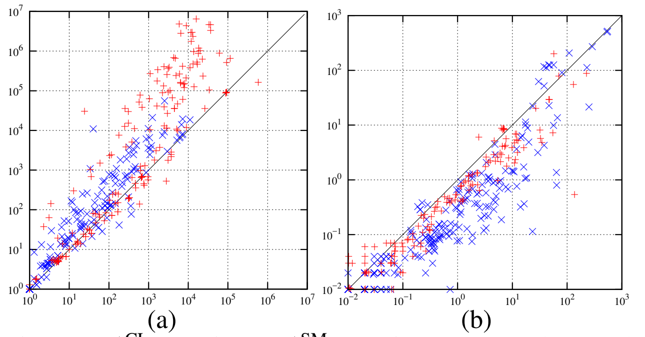
Figure 2: ψ^{CL} (y-axis) vs. ψ^{SM} (x-axis), (a) nogood generalization factor, (b) dead-end detection time (seconds).

Figure 2 (b) compares the runtime spent in ψ^{CL} vs. ψ^{SM} on those parts that actually differ, i. e., we disregard h^1/h^2 evaluation time on states not recognized by h^1/h^2, which is identical on both sides. We see that the generalization advantage of ψ^{CL} over ψ^{SM} carries over to runtime. We remark though that the identical runtime share on both sides is often so large as to overshadow this advantage.

Conclusion

We have shed new light on the issue of critical-path dead-end detection and its interplay with nogoods, leading in particular to the design of a new online nogood learning method that has advantages over the previously known one.

Addressing the runtime share on non-recognized states would require "goods" instead of "nogoods", formulas implying that $h^C(s) < \infty$, an interesting open direction.

It was surprising to us at first that one can represent critical-path dead-end detection exactly through a nogood formula computed once, offline before search. But perhaps this is not so surprising in hindsight – the formula basically enumerates all reasons why h^C could become ∞ during search. The more remarkable fact, then, is that this unwieldy formula, if feasible to build, is actually easier to evaluate than h^C. So there is hope yet for this kind of offline analysis, and the interesting question remains how to capture h^C, or relevant parts thereof, effectively. It may also be possible to combine knowledge gained this way with knowledge from other sources, e. g., abstractions.

Acknowledgments. This work was partially supported by the German Research Foundation (DFG), under grant HO 2169/5-1, "Critically Constrained Planning via Partial Delete Relaxation", as well as by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA, grant no. 16KIS0656).

References

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):279–298.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Bradley, A. R. 2011. Sat-based model checking without unrolling. In *Proceedings of the 12th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'11)*, 70–87.

Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence* 221:73–114.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In Chien, S.; Kambhampati, R.; and Knoblock, C., eds., *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, 140–149. Breckenridge, CO: AAAI Press, Menlo Park.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hoffmann, J., and Fickert, M. 2015. Explicit conjunctions w/o compilation: Computing $h^{\mathrm{FF}}(\Pi^C)$ in polynomial time. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*. AAAI Press.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J.; Kissmann, P.; and Torralba, Á. 2014. "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In Schaub, T., ed., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*. Prague, Czech Republic: IOS Press.

Kolobov, A.; Mausam; and Weld, D. S. 2012. Discovering hidden structure in factored MDPs. *Artificial Intelligence* 189:19–47.

Muise, C. J.; McIlraith, S. A.; and Beck, J. C. 2012. Improved non-deterministic planning by exploiting state relevance. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press.

Nakhost, H.; Hoffmann, J.; and Müller, M. 2012. Resource-constrained planning: A Monte Carlo random walk approach. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 181–189. AAAI Press.

Steinmetz, M., and Hoffmann, J. 2016. Towards clause-learning state space search: Learning to recognize dead-ends. In Schuurmans, D., and Wellman, M., eds., *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press.

Suda, M. 2014. Property directed reachability for automated planning. *Journal of Artificial Intelligence Research* 50:265–319.