Towards Clause-Learning State Space Search: Learning to Recognize Dead-Ends (Extended Abstract)

Marcel Steinmetz and Jörg Hoffmann

Saarland University,
Saarbrücken, Germany
{steinmetz,hoffmann}@cs.uni-saarland.de

1 Clause-Learning State Space Search

The ability to learn from conflicts is a key algorithm ingredient in constraint satisfaction (e. g. [6, 24, 20, 22, 8, 2]). For state space search, like goal reachability in classical planning which we consider here, progress in this direction has been elusive, and almost entirely limited to *length-bounded reachability*, where reachability testing reduces to a constraint satisfaction problem, yet requires iterating over different length bounds until some termination criterion applies [5, 19, 16, 28]. But *do we actually need a length bound to be able to do conflict analysis and nogood learning in state space search?*

Arguably, the canonical form of a "conflict" in state space search is a *dead-end* state, from which no solution (of any length) exists. Such conflicts are not as ubiquitous as in constraint satisfaction (including length-bounded reachability), yet they do occur, e. g., in oversubscription planning [26], in planning with limited resources [11], in single-agent puzzles [15, 4], and in explicit-state model checking of safety properties [7] where a dead-end is any state from which the error property cannot be reached.

We introduce a method that learns sound and generalizable knowledge from dead-end states during state space search classical planning. To our knowledge, this is the first of its kind. Prodigy [21] comes closest with its learning of sound action-pruning rules in backward search. Inspired by Prodigy, Bhatnagar and Mostow [3] considered forward-search conflict-based learning, yet their techniques are not sound (do not guarantee that pruned states actually are dead-ends). Kolobov et al's SixthSense technique [17] is sound, yet is placed in probabilistic planning and incorporates classical planning as a sub-procedure. Value function refinement using Bellman updates [18, 25, 1] will eventually learn that a state is a dead-end, yet does not generalize that knowledge.

The key to our technique are *critical-path heuristics* h^C [10, 9], relative to a set C of *atomic conjunctions*. These heuristics incorporate an approximation allowing to break up conjunctive subgoals into the elements of C. We don't give a full definition here, but the central equation should be suitable to get an idea:

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

This equation is easiest understood as a recursive estimation of goal distance. The bottom case in the equation splits up the current subgoal G into its atomic subgoals. The

middle case in the equation minimizes over all actions the subgoal can be regressed through. The top case terminates the recursion on subgoals that are true in our state s. The overall distance estimate is obtained through a top-level call (not shown here) on s and the global planning goal specified in the input planning task.

Critical-path heuristics were originally designed for admissible goal distance estimation. Here we are interested only in their ability to *recognize* dead-end states s, returning $h^C(s) = \infty$. This happens if every recursion path in the equation eventually hits an unsupported subgoal. Intuitively, $h^C(s) = \infty$ if s has no solution even when allowing to break up conjunctive subgoals into atomic conjunctions.

It is easy to see that, for sufficiently large C, all dead-ends will be recognized (just let C be the set of *all* conjunctions). But how to find a small yet informative set C useful for search? Our key idea is to *learn* C during search, through *conflict analysis*.

We start from the simple set C that contains only the singleton conjunctions. We augment forward state space search to identify *unrecognized* dead-ends s, where $h^C(s) < \infty$ yet search has already explored all descendants of s and thus proved s to be a dead end. We design h^C*-refinement* methods analyzing the situation at such s, adding new conjunctions into C to obtain $h^C(s) = \infty$, thus learning to recognize s as well as similar dead-ends search may encounter in the future. The refinement step is the most technical part of our work, and we refer to our AAAI'16 paper [27] for details. In a nutshell, the technique assumes as input a component of states s, where all direct successors t of any state s are already pruned (recognized to be dead-ends) by h^C. It then tackles open h^C-paths on the states s, canceling each such path by combining conjunctions canceling corresponding paths on states t. Suitable combined conjunctions necessarily exist, so that the method is constructive, guaranteeing to find the desired new conjunctions to learn, without having to do any search or exploration.

We furthermore learn *clauses* ϕ, in a manner inspired by, and similar to, a nogood certification technique in SixthSense: we minimize the commitments made in a dead-end state s while still preserving that $h^C(s) = \infty$. The clauses are sound in that $s' \not\models \phi$ implies $h^C(s') = \infty$, i. e., we learn sufficient conditions for h^C dead-end detection. While these sufficient conditions constitute a weaker pruning method than h^C itself, they are much faster to evaluate. Doing so prior to computing h^C strongly reduces the runtime overhead, which can otherwise be prohibitive.

Arranging these techniques in a depth-first search, we obtain an algorithm approaching the elegance of clause learning in SAT: When a subtree is fully explored, the h^C-refinement and clause learning (1) learns to refute that subtree, (2) enables backjumping to the shallowest non-refuted ancestor, and (3) generalizes to other similar search branches in the future. Our experiments show that this can be quite powerful. On problems where dead-ends abound, relative to the same search but without learning, our technique often reduces the search space by several orders of magnitude.

2 Empirical Results

Our implementation is in FD [12]. Our current experiments focus on *resource-constrained* planning, where the goal must be achieved subject to a fixed resource budget. We use the benchmarks by Nakhost et al. [23], which are *controlled* in that the minimum re-

	NoMystery (30 base instances)						Rovers (30 base instances)						TPP (5 base instances)					
		FD-h^{FF}			DFS-CL			FD-h^{FF}			DFS-CL			FD-h^{FF}			DFS-CL	
			M&S						M&S						M&S			
W	Blind	FD-h^{FF}	OA	NM	W/O L	W/ L	Blind	FD-h^{FF}	OA	NM	W/O L	W/ L	Blind	FD-h^{FF}	OA	NM	W/O L	W/ L
0.5	19	25	**30**	**30**	25	**30**	2	5	**30**	29	5	**30**	4	4	**5**	**5**	**5**	**5**
0.6	10	16	**30**	**30**	16	**30**	1	2	29	25	2	**30**	1	1	**5**	**5**	2	4
0.7	0	11	**30**	29	11	29	0	0	29	23	0	**30**	0	0	**5**	3	0	3
0.8	0	0	**30**	26	0	24	0	0	**24**	21	0	**24**	0	0	**1**	**1**	0	0
0.9	0	0	**29**	24	0	16	0	0	16	13	0	**22**	0	0	0	0	0	0
1.0	0	6	**26**	20	0	12	0	1	10	6	0	**21**	0	1	0	**2**	0	0
1.1	0	10	**24**	21	0	11	0	0	5	3	6	**14**	0	3	0	**4**	0	2
1.2	0	16	19	**22**	0	13	0	1	3	1	1	**14**	0	**3**	0	**3**	**3**	**3**
1.3	0	20	18	**24**	0	8	0	2	1	2	1	**12**	0	**4**	0	**4**	3	3
1.4	0	25	15	**27**	0	11	0	2	0	3	3	**12**	0	4	0	4	4	**5**
$\sum$	29	129	251	**253**	52	184	3	13	147	126	12	**208**	5	20	16	**31**	17	25

Fig. 1. Coverage results. Best per-domain results highlighted in **boldface**. DFS-CL is our approach, "W/O L" without learning, "W/ L" with learning. Other abbreviations see text. For each base instance and value of W, the resource budget is set according to W.

quired budget $b_{\min}$ is known, and the actual budget is set to $W * b_{\min}$. The parameter W allows to control the frequency of dead-ends; values of W close to 1.0 are notoriously difficult. In difference to Nakhost et al., we also consider values $W < 1$ where the tasks are unsolvable.[1]

We use a cluster of Intel E5-2660 machines running at 2.20 GHz, with runtime (memory) limits of 30 minutes (4 GB). Our technique runs a forward depth-first search; in selecting the next children node to expand, it prefers children with smaller h^{FF} [14] value. We compare to blind search, and to FD's greedy best-first dual-queue search with h^{FF} and preferred operators (denoted "FD-h^{FF}"), as baselines. We compare to Hoffmann et al.'s [13] two most competitive configurations of merge-and-shrink (M&S) heuristics for proving unsolvability (denoted here "OA" and "NM"). We run the latter as dead-end detectors in FD-h^{FF} to obtain variants competitive also for satisficing planning.

Figure 1 gives coverage data. Our approach easily outperforms the standard planner FD-h^{FF}. It is vastly superior in Rovers, and generally for budgets close to, or below, the minimum needed. The stronger planners using FD-h^{FF} with M&S dead-end detection are better than DFS-CL in NoMystery, worse in Rovers, and about on par in TPP.

For the exciting news, consider the comparison between with vs. without learning. The former outperforms the latter dramatically. The *only* reason for this is generalization, i. e., refinements of h^C on states s leading to pruning on states other than s. Without generalization, the search spaces would be *identical*, including tie breaking. So generalization occurs at *massive* scale. It lifts a hopeless planner (DFS with singleton-conjunction, aka h^1, dead-end detection) to a planner competitive with the state of the art in resource-constrained planning.

Figure 2 compares the search space sizes directly. On instances solved by both, the reduction factor min/geometric mean/maximum is: NoMystery 7.5 / 412.0 / 18117.9; Rovers 58.9 / 681.3 / 70401.5; TPP 1 / 34.4 / 1584.3.

[1] Not all actions consume resources, so that the fixed budget does not per se entail an upper bound on plan length. The more important role of the fixed budget for our technique, to our current understanding, is that search paths will tend to be short, i. e., a depth-first forward search will quickly run into dead-ends from which we are then able to learn.

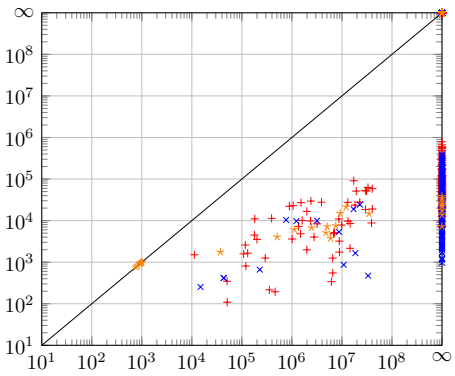

Fig. 2. Search space size for DFS-CL with learning (y-axis) vs. without learning (x-axis). "+" (red) NoMystery, "×" (blue) Rovers, "⋆" (orange) TPP, "∞": out of time or memory.

3 Conclusion

Our work pioneers conflict-directed learning, of sound generalizable knowledge, from dead-end states in forward state-space search. This is made possible by the progress in modern classical-planning heuristic functions, specifically h^C, and our key technical contribution in that context is a method for refining h^C's dead-end detection capabilities during search. The resulting technique is, in our humble opinion, quite elegant, and suggests that the learning from "true" conflicts in state space search, not necessitating a solution length bound, is worth the community's attention.

Beauty contests aside, from a pragmatical point of view the technique certainly does, as it stands, not deliver an empirical breakthrough. It vastly improves over using the same technique without learning, and it appears to have strengths in (certain) resource-constrained situations. As ours is merely a first foray into this kind of technique, and lots more remains to be explored – combinations with alternate search techniques, refinement of different dead-end detection machineries, reasoning over knowledge learned from different sources, etc. – we expect this to be the beginning of the story, not its end.

One thing we would particularly like to see is the export of this (kind of) technique, from classical planning where it is presently placed, to game-playing and model checking. For h^C refinement, this works "out of the box" modulo the applicability of Equation 1, i. e., the definition of critical-path heuristics. As is, this requires conjunctive subgoaling behavior. But more general logics (e. g. minimization to handle disjunctions) should be manageable.

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".

References

1. Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138, January 1995.

2. Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.
3. Neeraj Bhatnagar and Jack Mostow. On-line learning from search failures. *Machine Learning*, 15(1):69–117, 1994.
4. Ronald Bjarnason, Prasad Tadepalli, and Alan Fern. Searching solitaire in real time. *Journal of the International Computer Games Association*, 30(3):131–142, 2007.
5. Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):279–298, 1997.
6. Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, 1990.
7. Stefan Edelkamp, Alberto Lluch-Lafuente, and Stefan Leue. Directed explicit-state model checking in the validation of communication protocols. *International Journal on Software Tools for Technology Transfer*, 5(2-3):247–267, 2004.
8. Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *Proceedings of the 6th International Conference Theory and Applications of Satisfiability Testing (SAT'03)*, pages 502–518, 2003.
9. Maximilian Fickert, Jörg Hoffmann, and Marcel Steinmetz. Combining the delete relaxation with critical-path heuristics: A direct characterization. *Journal of Artificial Intelligence Research*, 56(1):269–327, 2016.
10. Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In S. Chien, R. Kambhampati, and C. Knoblock, editors, *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, pages 140–149, Breckenridge, CO, 2000. AAAI Press, Menlo Park.
11. Patrik Haslum and Hector Geffner. Heuristic planning with time and resources. In A. Cesta and D. Borrajo, editors, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 121–132. Springer-Verlag, 2001.
12. Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.
13. Jörg Hoffmann, Peter Kissmann, and Álvaro Torralba. "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In Thorsten Schaub, editor, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, Prague, Czech Republic, August 2014. IOS Press.
14. Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.
15. Andreas Junghanns and Jonathan Schaeffer. Sokoban: Evaluating standard single-agent search techniques in the presence of deadlock. In *Proceedings of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 1–15, 1998.
16. Subbarao Kambhampati. Planning graph as a (dynamic) CSP: Exploiting EBL, DDB and other CSP search techniques in graphplan. *Journal of Artificial Intelligence Research*, 12:1–34, 2000.
17. Andrey Kolobov, Mausam, and Daniel S. Weld. Discovering hidden structure in factored MDPs. *Artificial Intelligence*, 189:19–47, 2012.
18. Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 1990.
19. Derek Long and Maria Fox. Efficient implementation of the plan graph in stan. *Journal of Artificial Intelligence Research*, 10:87–115, 1999.
20. Joao Marques-Silva and Karem Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999.
21. Steven Minton, Jaime G. Carbonell, Craig A. Knoblock, Daniel Kuokka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40(1-3):63–118, 1989.

22. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Conference on Design Automation (DAC-01)*, Las Vegas, Nevada, USA, 2001. IEEE Computer Society.
23. Hootan Nakhost, Jörg Hoffmann, and Martin Müller. Resource-constrained planning: A monte carlo random walk approach. In Blai Bonet, Lee McCluskey, José Reinaldo Silva, and Brian Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 181–189. AAAI Press, 2012.
24. Patrick Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9:268–299, 1993.
25. Alexander Reinefeld and T. Anthony Marsland. Enhanced iterative-deepening search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):701–710, 1994.
26. David E. Smith. Choosing objectives in over-subscription planning. In Sven Koenig, Shlomo Zilberstein, and Jana Koehler, editors, *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, pages 393–401, Whistler, Canada, 2004. Morgan Kaufmann.
27. Marcel Steinmetz and Jörg Hoffmann. Towards clause-learning state space search: Learning to recognize dead-ends. In Dale Schuurmans and Michael Wellman, editors, *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, February 2016.
28. Martin Suda. Property directed reachability for automated planning. *Journal of Artificial Intelligence Research*, 50:265–319, 2014.