A Novel Lookahead Strategy for Delete Relaxation Heuristics in Greedy Best-First Search

Maximilian Fickert
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
fickert@cs.uni-saarland.de

Abstract

Best-first width search (BFWS) is a recent approach to satisficing planning that combines traditional heuristics with novelty measures to achieve a balance between exploration and effective search guidance (exploitation). One such novelty measure is based on counting the number of subgoals achieved on the path from a state in which a relaxed plan was computed. We introduce a new lookahead strategy for greedy best-first search based on this idea, where after each expansion, a bounded lookahead search is guided by relaxed subgoal counting. Furthermore, we combine this technique with partial delete relaxation heuristics to improve the subgoals. Using the $h^{C\text{FF}}$ heuristic with online-refinement of conjunctions, we obtain a planner that significantly outperforms the state of the art in satisficing planning on the IPC benchmarks.

Introduction

A successful approach to AI Planning is heuristic search. In satisficing planning, recent advancements to more informed heuristics and effective search techniques have significantly improved the state of the art.

Delete relaxation heuristics (Bonet and Geffner 2001; Hoffmann 2001) have been successfully used in state-of-the-art planners since their inception. While these heuristics are very efficient to compute, they may ignore important features of the task (e.g. resource consumption). Partial delete relaxation methods aim to mitigate these drawbacks by considering *some* delete information. Two such techniques are *red-black planning* (Domshlak, Hoffmann, and Katz 2015; Fickert, Gnad, and Hoffmann 2018), where some state variables are un-relaxed, and partial delete relaxation with *explicit conjunctions* (Haslum 2012; Keyder, Hoffmann, and Haslum 2012; Fickert, Hoffmann, and Steinmetz 2016), where some combinations of facts are treated atomically.

Recently, one of the most influential concepts for search enhancement in satisficing planning has been *novelty* (Lipovetzky and Geffner 2012). The simplest form prunes states that do not contain a fact (or a tuple of facts) that has not been contained in previously explored states. The novelty measure can also be tied to a heuristic (Lipovetzky and Geffner 2017; Katz et al. 2017), preferring states that contain novel facts among states with the same heuristic value.

Best-first width search (BFWS) (Lipovetzky and Geffner 2017) is a best-first search that combines traditional heuristics with novelty measures as lexicographic preferences. One of the measures employed in its best performing variants, BFWS(f_5) and Dual-BFWS, is based on the delete relaxation: counting the number of relaxed subgoals achieved along the path from the last state in which a relaxed plan was computed (Lipovetzky and Geffner 2014).

We introduce a search algorithm called GBFS-RSL, that is tailored around this technique. GBFS-RSL is an extension to greedy best-first search (GBFS) that performs a bounded lookahead after each expansion, using relaxed subgoal counting as heuristic guidance. Local exploration methods like random walks or local search have been used before to escape local minima and plateaus in GBFS (e.g. Nakhost and Müller 2009; Xie, Müller, and Holte 2014; Lipovetzky and Geffner 2017). In contrast, the main purpose of the lookahead of GBFS-RSL is to accelerate progress towards the goal. It is closely related to YAHSP's lookahead (Vidal 2004), which inserts additional states into the open list by applying actions of the relaxed plan. GBFS-RSL also exploits the structure of the relaxed plan to quickly try to find a state closer to the goal, but follows the relaxed plan more loosely by considering its subgoals instead of its actions.

Furthermore, we explore using heuristics based on red-black planning (Domshlak, Hoffmann, and Katz 2015), gray planning (an extension of red-black to limited-memory state variables, Speicher et al. 2017), and explicit conjunctions (Fickert, Hoffmann, and Steinmetz 2016) to provide the relaxed subgoals. Their partially relaxed plans are more accurate, and should therefore yield better subgoals. An additional motivation is that these heuristics are typically more expensive to compute than a standard delete relaxation heuristic, making the cheap approximation through relaxed subgoal counting particularly desirable. On the IPC benchmarks, a natural extension of GBFS-RSL with online-refinement of explicit conjunctions considerably outperforms state-of-the-art planners, beating e.g. Dual-BFWS (Lipovetzky and Geffner 2017) by +42 and MERWIN (Katz et al. 2018) by +31 instances in total coverage.

Background

We first introduce the basic planning definitions, and briefly summarize the relevant heuristics and search techniques.

Planning Framework

An FDR (Bäckström and Nebel 1995; Helmert 2009) planning task is defined as a 4-tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, where $v \in \mathcal{V}$ are the *state variables*, each with a finite domain $\mathcal{D}_v$, $\mathcal{A}$ is a set of *actions*, each with *preconditions* and *effects* (partial variable assignments), $\mathcal{I}$ is the *initial state* (complete assignment), and $\mathcal{G}$ is the *goal* (partial assignment). We refer to variable/value pairs as *facts*, written (variable = value). For a partial assignment p, $\mathcal{V}(p)$ denotes the set of variables on which p is defined. For a variable subset $V' \subseteq \mathcal{V}(p)$, the assignment made by p on V' is denoted by $p[V']$.

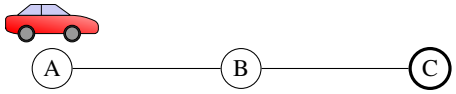
A *state* is a complete variable assignment. An action $a \in \mathcal{A}$ is *applicable* in a state s if it satisfies all of a's preconditions, i.e. $s[\mathcal{V}(\mathsf{pre}_a)] = \mathsf{pre}_a$. Applying a in s yields the same state as s, except that $s[\![a]\!](v) = \mathsf{eff}_a(v)$ for each $v \in \mathcal{V}(\mathsf{eff}_a)$. A *plan* π for a state s is a sequence of actions leading from s to a state compliant with the goal, and is called *optimal* if π is the shortest among all plans for s. A plan for $\mathcal{I}$ is a plan for the task Π.

Delete Relaxation

The delete relaxation is a simplification of the planning task, where variables accumulate their values instead of switching between them. The heuristic estimates of the delete relaxation heuristic h^{FF} (Hoffmann and Nebel 2001) are given by the length of a (not necessarily optimal) relaxed plan, and can be computed in polynomial time.

Partial Delete Relaxation

While delete relaxation heuristics have had great success in satisficing planning, there are domains where important aspects are ignored. Consider the following example:



The car needs to move from A to C. Each drive action consumes fuel. The car initially holds one unit of fuel, so it must refuel at location B. The task has two variables: the location of the car (*loc*), and the fuel status (*fuel*). A delete relaxed plan does not need to include the refueling action, since the fact that the car has fuel is never deleted.

Partial delete relaxation methods can fix this issue. With the red-black heuristic h^{RB} (Domshlak, Hoffmann, and Katz 2015), *some* variables are treated with the standard, non-relaxed, semantics. In the example above, if the fuel variable is un-relaxed, the partially relaxed plan must refuel in B, making it a real plan for this task.

A more fine-grained approach to partial delete relaxation is based on *explicit conjunctions*, where specific combinations of facts must be achieved simultaneously if required in preconditions of actions. In the example above, the key conjunction is being at location B while having fuel, $\{loc = B, fuel = true\}$, which must be achieved as a precondition for the drive action from B to C. Both individual facts are reached after driving from A to B (under delete-relaxed semantics), but not their combination because the drive action deletes the $fuel = true$ fact. The conjunction can only be achieved by the refuel action, and the resulting partially relaxed plan will be a real plan.

The $h^{C\text{FF}}$ heuristic (Fickert, Hoffmann, and Steinmetz 2016) is based on this concept. The conjunctions for $h^{C\text{FF}}$ are generated by a variant of counter-example guided abstraction refinement, adding conjunctions that prevent conflicts in previous relaxed plans (Haslum 2012; Keyder, Hoffmann, and Haslum 2014). The heuristic is most effective if the set of conjunctions is refined online (Fickert and Hoffmann 2017a; 2017b; Fickert 2018), as it allows the heuristic to adapt to the search space as it is being explored.

Novelty Pruning

A recent search enhancement technique is based on measuring the "novelty" of states: given a state s and a set of states seen so far S, the *novelty* of s is the size of the smallest tuple t of facts that are true in s but not in any other state $s' \in S$.

Novelty-based techniques aim to improve the balance of exploration and exploitation. The simplest such algorithm is IW(k), which performs a breadth-first search with *k-novelty pruning*, denoted $\mathcal{N}_k$, pruning all states with novelty greater than k (Lipovetzky and Geffner 2012). An extension to this is C-novelty pruning ($\mathcal{N}_C$), where, for a given set of conjunctions C, all states that do not make a conjunction $c \in C$ true for the first time are pruned (Fickert 2018).

More complex novelty measures have been devised that tie the novelty score to a heuristic value, and consider the novelty of a state only among other states with the same (Lipovetzky and Geffner 2017) or lower (Katz et al. 2017) heuristic value. In best-first width search (BFWS), the function ordering the open list is a tie-breaking sequence of multiple evaluation functions, where the primary function is a novelty measure (Lipovetzky and Geffner 2017).

Relaxed Subgoal Counting

The empirically best-performing evaluation function for BFWS is called f_5, which is given by the tie-breaking sequence $\langle w_{\#g,\#r}, \#g \rangle$. The first evaluation function is a novelty measure based on $\#g$ and $\#r$, where, for a state s,

- $\#g(s)$ is the number of unsatisfied goal facts in s, and
- $\#r(s)$ is the number of achieved subgoals of the last relaxed plan π^+ on the path to s from the state where π^+ was computed (Lipovetzky and Geffner 2014).

In BFWS(f_5), relaxed plans are computed only in states where the $\#g$ counter changes compared to its parent (and in the initial state), making f_5 very cheap to compute overall.

We denote the *relaxed subgoal counting* heuristic by h^{rsc}, which corresponds to $\#r$, but counting the number of *un*achieved subgoals. Furthermore, we make one additional change: $\#r$ considers all facts made true by the effects of all actions in the relaxed plan as subgoals, whereas we only consider the *necessary* subgoals, i.e. those that are required as preconditions for other actions in the relaxed plan and goals, ignoring facts that are only side effects. We made this change to more accurately capture the intention of the relaxed plan, and empirically found that it improves performance for our search algorithm introduced in the following as well as for BFWS(f_5) (albeit to a lesser degree).

Relaxed Subgoal Counting for Lookahead in GBFS

The key challenges for using relaxed subgoal counting are (a) to have a good strategy to select states for the computation of relaxed plans, and (b) to avoid comparison of subgoal counts with different underlying relaxed plans.

In BFWS (Lipovetzky and Geffner 2017), (a) is answered by computing a relaxed plan only in states where the number of achieved top-level goals increases over its parent. Point (b) is less important in BFWS, where the relaxed subgoal counter is only used as a novelty measure, not as a heuristic value directly. However, the novelty measure does not distinguish between states with different underlying relaxed plans, and these comparisons can potentially be misleading.

We introduce a new search algorithm called GBFS-RSL (for *Relaxed Subgoals Lookahead*) that is designed around these properties of the subgoal counting heuristic (Algorithm 1). The algorithm is an extension of GBFS, with a bounded lookahead search after each expansion of a state s. The lookahead search is guided with a relaxed subgoal counting heuristic h^{rsc}, which makes the lookahead very fast as heuristic evaluations are cheap. The challenges (a) and (b) are effectively solved by initializing the subgoal counting heuristic with a relaxed plan computed in the root state of the lookahead (line 10), which is already availably assuming that the search uses a (partial) delete relaxation heuristic. When the lookahead finishes, it returns the state s' with the lowest h^{rsc}-value seen in the local search space (breaking ties arbitrarily, line 11). The heuristic value of that state is then compared to that of s, and if s' has a lower heuristic value, it is inserted at the front of the open list (line 13).

We bound the lookahead using incomplete novelty pruning (only considering the states seen within the current lookahead search space for the novelty pruning). Using $\mathcal{N}_1$, the lookahead search will expand at most as many states as there are different facts in the task. GBFS-RSL could in principle be instantiated with an arbitrary method to bound the lookahead, e.g. a simple bound on the search depth or number of expansions. However, novelty pruning improves the exploration of the lookahead by taking the structure of the local search space into account, and novelty-bounded lookaheads have had great results in a related hill-climbing search algorithm (Fickert 2018). The search algorithm for the lookahead can also be chosen freely; in our evaluation we consider best-first search with the ordering functions g, $g + h$, and h, i.e. BrFS, A*, and GBFS.

Online Refinement for $h^{C\text{FF}}$

The $h^{C\text{FF}}$ heuristic is most effective when its set of conjunctions C is refined online, and a suitable condition for when to trigger the refinement is to identify a state where the heuristic is inaccurate, e.g. local minima (Fickert and Hoffmann 2017a). In GBFS-RSL, such a situation occurs when the lookahead does not return a state with lower heuristic value. Thus, we can naturally extend GBFS-RSL by invoking its refinement procedure in that case (line 15). The refinement adds a single conjunction to C based on the conflicts identified in the current relaxed plan, preventing at least one of them from occuring again in future partially relaxed plans.

Algorithm 1: GBFS-RSL

```
1  Open := [I], Closed := ∅
2  while Open ≠ [] do
3      s := Open.pop()
4      if s ∈ Closed then continue
5      if s ⊇ G then return path to s
6      Closed := Closed ∪ {s}
7      if h(s) ≠ ∞ then
8          Insert the successors of s into Open
9          Let π+ be the relaxed plan extracted by h in s
10         Initialize h^rsc with π+
11         s' := lookahead(s, h^rsc)
12         if h(s') < h(s) then
13             Insert s' at the front of Open
14         else // only with h^CFF
15             Refine h in s
16 return UNSOLVABLE
```

Experiments

We implemented GBFS-RSL in Fast Downward (Helmert 2006). The experiments were run using the lab framework (Seipp et al. 2017) on machines with Intel Xenon E5-2660 processors with a clock rate of 2.2 GHz, with time and memory limits of 30 minutes and 4 GB respectively. The benchmark set consists of all STRIPS domains from the satisficing tracks of all International Planning Competitions up to 2018, for a total of 1825 instances of 49 domains.

GBFS-RSL with Different Heuristics

First we compare the usage of different (partial) delete relaxation heuristics in GBFS-RSL: h^{FF} (Hoffmann and Nebel 2001), the red-black heuristic h^{RB} (Domshlak, Hoffmann, and Katz 2015), its extension to limited-memory state variables h^{Gray} (Speicher et al. 2017), and $h^{C\text{FF}}$ (Fickert, Hoffmann, and Steinmetz 2016) with both offline and online refinement. Table 1 shows an overview for these heuristics with different lookahead search algorithms, and a comparison to standard GBFS and YAHSP's lookahead (Vidal 2004; 2011). All configurations use a dual queue with preferred operators and lazy evaluation. The results of $h^{C\text{FF}}$ configurations are averaged over 5 random seeds as the heuristic uses random tie breaking (the table shows rounded values); standard deviation ranges from 2.6 to 9.4 (5.8 on average).

GBFS works best as the lookahead search algorithm for all considered heuristics except $h^{C\text{FF}}$, where A* is better.

Compared to standard GBFS, GBFS-RSL generally works well if the lookahead does not add much overhead, i.e. the computation of the base heuristic still accounts for most of the overall time (this is the case in e.g. Agricola, Snake, and Spider for h^{FF}), or the lookaheads consistently find states with lower heuristic value (e.g. Elevators and VisitAll). In VisitAll, the advantage for GBFS-RSL is most obvious: all 40 instances (compared to 3 with standard GBFS) are solved with at most 8 (non-lookahead) expansions. This is due to the additive nature of the goals: the lookahead

h	GBFS-RSL Nov.	BrFS	A*	GBFS	GBFS	YAHSP
h^{FF}	$\mathcal{N}_1$	1359	1505	1518	1494	1529
h^{RB}		1432	1490	1513	1508	1541
h^{Gray}		1486	1538	1543	1555	1579
$h^{C\text{FF}}_{off}$		1454	1570	1555	1498	1603
$h^{C\text{FF}}_{on}$		1531	1639	1613	–	1573
$h^{C\text{FF}}_{off}$	$\mathcal{N}_C$	1476	1577	1546	–	–
$h^{C\text{FF}}_{on}$		1584	**1665**	1627	–	–

Table 1: Coverage of GBFS-RSL with different configurations. The $h^{C\text{FF}}$ heuristic is included with offline ($h^{C\text{FF}}_{off}$) and online ($h^{C\text{FF}}_{on}$) refinement variants, and additional GBFS-RSL configurations where the conjunctions are also used for novelty pruning ($\mathcal{N}_C$). The columns on the right show a comparison to standard GBFS and YAHSP's lookahead.

	GBFS-RSL	BFWS(f_5)	Dual-BFWS	LAMA	Mercury	MERWIN	Coverage
GBFS-RSL	–	20	16	20	15	13	**1665**
BFWS(f_5)	8	–	5	15	11	9	1530
Dual-BFWS	9	22	–	18	12	10	1623
LAMA	6	19	10	–	5	4	1574
Mercury	9	19	13	14	–	2	1605
MERWIN	10	20	14	17	12	–	1634

Table 2: Pairwise comparison of GBFS-RSL and state-of-the-art satisficing planners. The number in row r and column c shows the number of domains where the planner in row r has higher coverage than the one in column c (considering domain-wise rounded coverage for GBFS-RSL).

greedily moves to as many unvisited locations as possible, and the resulting state is almost guaranteed to be closer to a goal state than the state where the lookahead was initiated.

Against intuition, the results with h^{RB} are worse than those with h^{FF}, and GBFS-RSL with either h^{RB} and h^{Gray} does not beat standard GBFS. While the partially relaxed plans obtained by h^{RB} and h^{Gray} are closer to real plans, their subgoals do not provide significantly better guidance than those of h^{FF}. We believe this is due to the structure of these partially relaxed plans, which contain sequences of actions that repair conflicts on black (non-relaxed) preconditions. In our context however, these repair sequences may create misleading subgoals, as the lookahead will likely not follow these sequences exactly. Furthermore, red-black planning was designed to address cases where the agent is required to move back and forth, which may not be possible in our lookahead due to novelty pruning.

In contrast, GBFS-RSL with $h^{C\text{FF}}$ has dramatically higher coverage than its baseline. The extension with online refinement adds another big leap in coverage, which confirms our expectation that unsuccessful lookaheads are a suitable condition to trigger the refinement process for $h^{C\text{FF}}$. Consistent with previous work (Fickert 2018), using the conjunctions of $h^{C\text{FF}}$ for novelty yields yet another boost in performance, reaching a coverage of 1664.6 ($\pm$8.96).

For comparison, we adapted the lookahead technique of YAHSP (Vidal 2004; 2011) to lazy GBFS, inserting a single lookahead state at the beginning of the open list after each expansion similar to GBFS-RSL. This outperforms GBFS-RSL with all considered heuristics, but, surprisingly, coverage drops when adding online refinement. In some domains (e.g. Barman, Childsnack), the YAHSP lookahead often fails to find a better state, and the frequent refinement incurs too much computational overhead in $h^{C\text{FF}}$.

State-of-the-Art Comparison

We next compare our best performing configuration (GBFS-RSL with $h^{C\text{FF}}$ and online refinement) to various state-of-the-art planners: the BFWS-based planners BFWS(f_5) and Dual-BFWS (Lipovetzky and Geffner 2017), LAMA (Richter and Westphal 2010), Mercury (Domshlak, Hoffmann, and Katz 2015; Katz and Hoffmann 2014), and its successor MERWIN (Katz et al. 2017; 2018).

Table 2 shows a pairwise comparison. GBFS-RSL has higher coverage in more domains than the other way around for each considered planner. The closest competitors are MERWIN (GBFS-RSL is better in 13 and worse in 10 domains) and Dual-BFWS (better in 16, worse in 9). GBFS-RSL has strictly higher coverage than the other planners in Childsnack, Data Networks, Floortile, Pipesworld-Tankage, and Spider. Furthermore, in 10 other domains, GBFS-RSL has similar coverage but strictly lower search time than any other planner, e.g. Hiking (on average 89% faster than the next best planner), Thoughtful (75% faster), and Depots (58% faster). On the other hand, it performs poorly in Parking, Sokoban, and Termes. In Parking, each lookahead generates a large number of states and GBFS-RSL quickly hits the memory limit on larger instances[1]; in Sokoban and Termes, the lookahead often returns a state that was already closed (or, in Sokoban, identified as a dead end). Overall, GBFS-RSL has the highest coverage out of the considered planners by a significant margin of +31.

Conclusion

We have introduced a new lookahead strategy in greedy best-first search with (partial) delete relaxation heuristics. When using the $h^{C\text{FF}}$ heuristic with online-refinement of conjunctions, we observe significant gains over current state-of-the-art satisficing planners. For future work, different weights (or entirely different search algorithms) could be tried in the lookaheads. Furthermore, different variants of h^{rsc} could be explored, and we had some initial success when considering h^{add} values of the achieved subgoals instead of just counting them. In principle, GBFS-RSL is not restricted to delete relaxation heuristics, but could be used with other heuristics that have a different means of providing subgoals as well.

[1]We remark that this issue can be effectively solved by adding the h^2 preprocessor (Alcázar and Torralba 2015).

Acknowledgments

This work was funded by DFG grant 389792660 as part of TRR 248 (see https://perspicuous-computing.science).

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 2–6. AAAI Press.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS^+ planning. *Computational Intelligence* 11(4):625–655.

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

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

Fickert, M., and Hoffmann, J. 2017a. Complete local search: Boosting hill-climbing through online heuristic-function refinement. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*, 107–115. AAAI Press.

Fickert, M., and Hoffmann, J. 2017b. Ranking conjunctions for partial delete relaxation heuristics in planning. In *Proceedings of the 10th Annual Symposium on Combinatorial Search (SOCS'17)*, 38–46. AAAI Press.

Fickert, M.; Gnad, D.; and Hoffmann, J. 2018. Unchaining the power of partial delete relaxation, part II: finding plans with red-black state space search. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, 4750–4756.

Fickert, M.; Hoffmann, J.; and Steinmetz, M. 2016. Combining the delete relaxation with critical-path heuristics: A direct characterization. *Journal of Artificial Intelligence Research* 56(1):269–327.

Fickert, M. 2018. Making hill-climbing great again through online relaxation refinement and novelty pruning. In *Proceedings of the 11th Annual Symposium on Combinatorial Search (SOCS'18)*, 158–162. AAAI Press.

Haslum, P. 2012. Incremental lower bounds for additive cost planning problems. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 74–82. AAAI Press.

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

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* 173:503–535.

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. 2001. FF: The fast-forward planning system. *The AI Magazine* 22(3):57–62.

Katz, M., and Hoffmann, J. 2014. Mercury planner: Pushing the limits of partial delete relaxation. In *IPC 2014 planner abstracts*, 43–47.

Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2017. Adapting novelty to classical planning as heuristic search. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*, 172–180. AAAI Press.

Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2018. MERWIN planner: Mercury enchanced with novelty heuristic. In *IPC 2018 planner abstracts*, 53–56.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2012. Semi-relaxed plan heuristics. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 128–136. AAAI Press.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2014. Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research* 50:487–533.

Lipovetzky, N., and Geffner, H. 2012. Width and serialization of classical planning problems. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*, 540–545. Montpellier, France: IOS Press.

Lipovetzky, N., and Geffner, H. 2014. Width-based algorithms for classical planning: New results. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, 1059–1060. Prague, Czech Republic: IOS Press.

Lipovetzky, N., and Geffner, H. 2017. Best-first width search: Exploration and exploitation in classical planning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, 3590–3596. AAAI Press.

Nakhost, H., and Müller, M. 2009. Monte-carlo exploration for deterministic planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, 1766–1771. Pasadena, California, USA: Morgan Kaufmann.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461.

Speicher, P.; Steinmetz, M.; Gnad, D.; Hoffmann, J.; and Gerevini, A. 2017. Beyond red-black planning: Limited-memory state variables. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*, 269–273. AAAI Press.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS'04)*, 150–160. Whistler, Canada: Morgan Kaufmann.

Vidal, V. 2011. YAHSP2: Keep it simple, stupid. In *IPC 2011 planner abstracts*, 83–90.

Xie, F.; Müller, M.; and Holte, R. 2014. Adding local exploration to greedy best-first search in satisficing planning. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, 2388–2394. Austin, Texas, USA: AAAI Press.