Alternation-Based Novelty Search

Augusto B. Corrêa[1,2], **Jendrik Seipp**[3]

[1]University of Oxford, United Kingdom
[2]University of Basel, Switzerland
[3]Linköping University, Sweden
augusto.blaascorrea@chch.ox.ac.uk, jendrik.seipp@liu.se

Abstract

One key decision for heuristic search algorithms is how to balance exploration and exploitation. In classical planning, the two strongest approaches for this problem are to *alternate* between different heuristics and to enhance heuristics with *novelty* measures. The most well-known planner using alternation is LAMA, which cycles between different open-lists that are ordered using different heuristics. The strongest novelty-based algorithms use *best-first width search* (BFWS), which prefers states that contain previously unseen combinations of atoms. Considerable effort has been put into trying to combine these two approaches, but so far, no combination has been able to significantly improve over the individual planners. In this paper, we explore the simple idea of using BFWS as just another open-list for LAMA. Our results show that adding even the strongest BFWS version to LAMA is detrimental. However, combining only parts of each approach yields a new state-of-the-art agile planner.

Introduction

Agile planning involves solving planning tasks as fast as possible, with little or no consideration for plan quality. While this might not be suitable for some applications, it is still an important setting in general, and it is closely related to the problem of deciding plan existence (e.g., Bylander 1994). Since 2014, there have been dedicated tracks for agile planning in the classical part of the International Planning Competition (IPC). Usually, planners are given five minutes per task, and they are scored based on how quickly they solve the task.

In the recent IPC 2023, LAMA (Richter and Westphal 2010),[1] a 15 year-old planner that was included as a baseline, obtained a higher total agile score than all actual competitors, including the winner DecStar (Gnad, Torralba, and Shleyfman 2023). A similar situation occurred in the IPC 2018, where the LAMA baseline scored second place.

LAMA uses *preferred operators* and *deferred evaluation* (Richter and Helmert 2009), and multiple open-lists (Röger and Helmert 2010). *Alternating* between the open-lists allows LAMA to balance between multiple heuristic estimates and let it win the Satisficing Track of the IPC in 2008. Even though the LAMA code base is continuously improved, the core behavior remains unchanged. This raises the question of how we can advance the state of the art and obtain a planner that finds plans faster than LAMA.

In the IPC 2018 Agile Track, the only planner to outperform LAMA was based on best-first width search (BFWS) (Lipovetzky and Geffner 2017; Francès et al. 2018). The idea of BFWS is to favor states that are *novel* (Lipovetzky and Geffner 2012). The novelty of a state is the size of the smallest set of atoms that has not been part of any previously evaluated state. When combined with heuristic estimates, BFWS achieves a good exploration-exploitation trade-off which helps find plans quickly.

In this paper, we aim to combine the advantages of LAMA and BFWS. It turns out though, that this combination actually yields a *worse* planner than LAMA and BFWS(f_6) alone. However, a detailed ablation study reveals that *removing features* from BFWS drastically speeds up the planner. In fact, discarding partition functions and tie-breakers from BFWS(f_6), improves the coverage and IPC agile score compared to the original systems. Additionally, we show a method that identifies when to use lower width based solely on the structure of the task. This greatly improves the performance of BFWS and related algorithms for tasks with many variables. Our planner also outperforms all agile competitors of the last IPCs by a large margin, with an IPC agile score 17% higher than the winner of IPC 2023.

Background

A *state space* is a tuple $\mathcal{S} = \langle S, s_I, G, succ \rangle$, where S is a set of *states*, $s_I \in S$ is the *initial state*, G is the *goal description*, *succ* is a *successor function* mapping each state to a finite (possible empty) set of successor states. A state is a set of (ground) *atoms*. It is sufficient to consider S as the minimal set where $s_I \in S$, and $succ(s) \subseteq S$ for each $s \in S$. The goal G is also a set of (ground) atoms. A state s_* is a *goal state* if $G \subseteq s_*$.

A sequence of states $\tau = \langle s_0, \ldots, s_n \rangle$ is a *path* from s_0 to s_n in S if $s_i \in succ(s_{i-1})$ for $i \in \{1, \ldots, n\}$. Path τ is an *s-plan* if $s_0 = s$ and $G \subseteq s_n$ and a *plan* for $\mathcal{S}$ if $s_0 = s_I$. We consider *agile planning*, the problem of computing plans as fast as possible, without caring about their length.

A common method to find plans is via *heuristic search*. A *heuristic* is a function $h : S \to \mathbb{R}_0^+ \cup \{\infty\}$. It estimates

[1]Throughout the paper, "LAMA" refers to its *first iteration*.

the length of an s-plan for states $s \in S$. Heuristic search algorithms start from s_I and expand states guided by some heuristic h, preferring states s with low $h(s)$ values. Examples of strong heuristics for agile planning are h^{add} (Bonet and Geffner 2001), h^{FF} (Hoffmann and Nebel 2001), and h^{LM} (Richter, Helmert, and Westphal 2008). We assume familiarity with common search algorithms such as *greedy best-first search* (Doran and Michie 1966).

Instead of being guided by a heuristic, BFS(w) (Lipovetzky and Geffner 2012, 2017) selects states based on their *novelty*, preferring states s with low $w(s)$ values. The novelty $w(s)$ of a state s is the size of the smallest set of atoms A such that s is the first evaluated state that subsumes A.

This simple scheme can be turned into state-of-the-art *best-first width search* (BFWS) algorithms by extending it with *partition functions* (Lipovetzky and Geffner 2017; Francès et al. 2017, 2018). For BFWS, the novelty $w_{\langle h_1,\ldots,h_n\rangle}(s)$ of a state s given the *partition functions* $\langle h_1,\ldots,h_n\rangle$ is the size of the smallest set of atoms A such that s is the first evaluated state that subsumes A, among all states s' visited before s for which $h_i(s) = h_i(s')$ for $1 \leq i \leq n$. In practice, BFWS planners only evaluate novelty up to a bound k, where usually $k = 2$. If a state s has no novel tuple of size k or less, then $w(s) = k + 1$.

Balancing Exploration and Exploitation

One important design choice of a planner is how it balances *exploration* and *exploitation*. Exploration techniques search parts of the state space that have not yet been visited, while exploitation techniques prefer going into parts that are considered more promising by some metric.

Modern planners usually mix both of them. A common approach is to keep several open-lists during search, each one guided by a different heuristic (Röger and Helmert 2010). The simplest yet most successful method for combining multiple open-lists is *alternation* (Helmert 2006). An alternation-based search algorithm maintains n open-lists, where the i-th open-list is ordered by some heuristic h_i. We denote it as $[h_1,\ldots,h_n]$. The search alternates between the open-lists in a round-robin fashion: first it expands the best state according to h_1, adds all successors to all (or some of the) open-lists, then it expands the best state according to h_2, and so on. In iteration $n+1$ it expands from h_1 again.

Alternation is one of the main building blocks used by the LAMA planner (Richter and Westphal 2008, 2010; Richter, Westphal, and Helmert 2011). LAMA uses four open-lists: $[h^{\text{FF}}, h^{\text{FF}}_+, h^{\text{LM}}, h^{\text{LM}}_+]$, where h_+ denotes an open-list ordered by h but only containing states reached via preferred operators (Hoffmann and Nebel 2001).

An orthogonal way of combining exploration and exploitation is by using *tiebreakers* (Röger and Helmert 2010). A tiebreaking open-list $\langle h_1,\ldots,h_n\rangle$ uses a ranking over n heuristics. It selects states based on h_i and, if there is a tie, breaks this tie using h_{i+1}. It keeps only a single open-list, but the order of this list is defined by multiple heuristics. Throughout the paper, we assume that if all $h_1,\ldots,h_n$ are tied, then remaining ties are broken by g-value. If ties persist, then we assume a FIFO ordering.

While LAMA opted for an alternation open-list, the more sophisticated versions of BFWS use tiebreaking (Lipovetzky and Geffner 2017). BFWS(f) orders its open-list by $f = \langle f_1,\ldots,f_n\rangle$. The strongest version of BFWS is BFWS(f_6), where the open-list is ordered by $f_6 = \langle w_{\langle h^{\text{LM}},h^{\text{FF}}\rangle}, \mathit{pref}, h^{\text{LM}}, w_{\langle h^{\text{FF}}\rangle}, h^{\text{FF}}\rangle$, where pref is an indicator function yielding 1 for states reached via a preferred operator. Simpler versions include BFWS(f_4), where $f_4 = \langle w_{\langle h^{\text{LM}},h^{\text{FF}}\rangle}, h^{\text{LM}}, h^{\text{FF}}\rangle$, and BFWS($f_2$), where $f_2 = \langle w_{\langle h^{\text{FF}}\rangle}, h^{\text{FF}}\rangle$.

Experimental Setup

Before we analyze BFWS, LAMA and their combination, we present the experimental setup used throughout the paper. We implemented all algorithms within the Scorpion planning system (Seipp, Keller, and Helmert 2020), which is an extension of Fast Downward (Helmert 2006). Our implementation uses finite-domain representation (Helmert 2009).

For running our experiments, we use Downward Lab (Seipp et al. 2017) on AMD EPYC 7742 processors running at 2.25 GHz. We use the same limits as the IPC 2023 Agile Track: 5 minutes and 8 GiB of memory per task.

We use two benchmark sets: the set of *old IPC tasks* consists of 2502 tasks from 51 domains from IPCs 1998–2014, while the set of *new IPC tasks* consists of 360 tasks from 17 domains from IPCs 2018 and 2023. We omit the Slitherlink domain from the new IPC tasks, as some planners do not support it. For fairness, we use only the old IPC tasks to select the best configuration of our planner, because the new IPC tasks were not available during the development of the IPC 2018 and 2023 planners we compare to. We use the new IPC tasks for comparing our strongest algorithm to state-of-the-art agile planners on unseen tasks.

We use the h^2-preprocessor for all planner runs (Alcázar and Torralba 2015), including the state-of-the-art planners, and employ three main metrics to evaluate planner performance: coverage, expansion score and agile score.[2] For each metric, the total score is the sum over all tasks. All benchmarks, code and experiment data are available online (Corrêa and Seipp 2025).

Analyzing BFWS

We start by analyzing the most crucial parameter of BFWS, the width value k, by running BFWS(f_6) with $k = 1$ and $k = 2$ on the old IPC tasks. The results in the left part of Table 1 show that BFWS(f_6) with $k = 1$ solves 2228 tasks, while $k = 2$ solves 2159. While $k = 1$ has strictly higher coverage in 14 domains, $k = 2$ is superior in 4. Ideally, we would like to set k dynamically based on the task at hand. Looking closer at the results, we see that $k = 1$ is

[2]The expansion score is based on the number of expanded states, while the agile score is based on the runtime (Richter and Helmert 2009). Performance better than a lower bound (100 states for expansions and 1 second for runtime) counts as 1. Performance worse than an upper bound U (10^6 states for expansions and 300s for runtime) counts as 0. We interpolate intermediate values with a logarithmic function: $1 - \log(x)/\log(U)$ where x is the number of expansions/runtime in seconds.

	Fixed		Dynamic		
	k=1	k=2	V=10	V=100	V=1000
Coverage	2228	2159	2227	**2233**	2182
Exp. Score	1909.5	1877.6	1908.3	**1909.7**	1886.3
Agile Score	1786.1	1707.3	1785.6	**1786.4**	1711.5

Table 1: Results for BFWS(f_6) using different width values.

preferable for tasks with many state variables, which makes sense because evaluating the novelty of a state runs in time $O(n^k)$ for tasks with n variables. So we evaluate setting k based on a threshold V on the number of variables n. If $n \leq V$, we use $k = 2$, otherwise we use $k = 1$. We test $V \in \{10, 100, 1000\}$ and can observe in Table 1 that all dynamic choices outperform BFWS with k fixed to 2. In particular, the best performance in all metrics tested is obtained with $V = 100$, which we use for all further experiments.

Combining LAMA and BFWS

LAMA and BFWS present distinct ways of combining exploration and exploitation but they also share similarities. If we consider BFWS(f_6), then both planners use exactly the same information (with the exception of the novelty measures): h^{FF}, h^{LM}, and preferred operators. The difference is in the way the planners process the information and the question is how we can combine the advantages of both in a single planner. Arguably the most direct approach to combine both planners is to use the tiebreaking open-list of BFWS(f_6) as an additional open-list in LAMA. We call this modification LAMA-W(f_6), and we use the same dynamic selection of the width as just explained.

The first two columns of Table 2 compare LAMA with LAMA-W(f_6). Despite both achieving the same coverage, LAMA-W(f_6) is much worse than LAMA in the other metrics. The low expansion score indicates that the addition of the new open-list makes the search less informed. This is somewhat counter-intuitive because in many cases combining different evaluators within the same search algorithm decreases the number of expansions (c.f., Corrêa et al. 2023).

To investigate this, we decompose the features of LAMA-W(f_6). Our first step is to iteratively simplify the new open-list by removing subsets of its features (partition functions, tie-breakers). We define the following variations: 1. LAMA-W(f_4), where $f_4 = \langle w_{\langle h^{\text{LM}}, h^{\text{FF}}\rangle}, h^{\text{LM}}, h^{\text{FF}}\rangle$; 2. LAMA-W($f_2^h$). where $f_2^h = \langle w_{\langle h\rangle}, h\rangle$. We test two versions: $h = h^{\text{FF}}$ and $h = h^{\text{LM}}$; 3. LAMA-W($w_{\langle h\rangle}$) uses the open-list of BFWS($w_{\langle h\rangle}$) (Lipovetzky and Geffner 2017), a best-first width search with a single open-list ordered by $w_{\langle h\rangle}$ and breaking ties by accumulated cost g. Here, we evaluate three versions: $w_{\langle h^{\text{FF}}\rangle}$, $w_{\langle h^{\text{LM}}\rangle}$ and $w_{\langle\rangle}$. The latter does not use any partition function.

Table 2 and Figure 1 show the results for all methods. From left to right, the novelty-based open-list becomes increasingly simple. There are two immediate observations to be made: first, simpler approaches, such as LAMA-W($f_2^{h^{\text{LM}}}$), perform better than LAMA-W(f_6); second, using

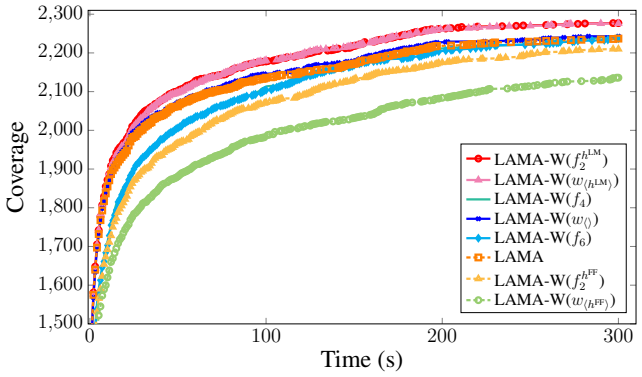

Figure 1: Coverage over time for our new algorithms on the old IPC tasks.

h^{LM} in the novelty open-list is consistently superior to h^{FF}. LAMA-W($f_2^{h^{\text{LM}}}$) improves over LAMA in coverage, and over LAMA-W(f_6) in all three criteria. It is also superior to BFWS(f_6) in coverage and agile scores (see Table 1).

LAMA uses four open-lists, while our new best method, LAMA-W($f_2^{h^{\text{LM}}}$) uses five. To streamline notation in the rest of the paper, we refer to this version as *NOLAN*, as it uses (among other features) NOvelty, Landmarks and AlternatioN. In exploratory experiments, we tried to simplify NOLAN using a similar strategy as the one presented in Table 2: we started with all five open-lists and tested the effects of removing different subsets of them. While some subsets get close to NOLAN, none of them outperforms it.

Table 3 shows a per-domain coverage comparison between BFWS(f_6), LAMA and NOLAN. While BFWS(f_6) and LAMA have their strengths in different domains, NOLAN is usually preferable to both of them. This shows that NOLAN's higher total coverage does not stem from only one or two domains, but that alternation-based novelty search increases coverage for a large number of domains.

Related Work

Before we compare NOLAN to state-of-the-art agile planners, we discuss related work, some of which forms the basis for some of the planners we compare to.

Balancing exploration and exploitation is a longstanding challenge in classical planning (Hoffmann and Nebel 2001; Richter and Westphal 2008; Nakhost and Müller 2009; Röger and Helmert 2010; Vidal 2011; Katz et al. 2017; Asai and Fukunaga 2017; Fickert 2018). In recent years, BFWS has emerged as the most successful approach for this problem (Lipovetzky and Geffner 2012, 2017; Francès et al. 2017). We evaluate two representatives, BFWS-Preference and Dual-BFWS (Francès et al. 2018), in the next section.

Follow-up work combined BFWS with other search techniques. For example, Katz et al. (2017) combine the concept of novelty with heuristic estimates, extending the definition of novelty by Shleyfman, Tuisov, and Domshlak (2016) to take into account the heuristic value of the states. This allows them to quantify how novel a state is, so the search can be guided directly by this value.

	LAMA	LAMA-W(f_6)	LAMA-W(f_4)	LAMA-W($f_2^{h^{FF}}$)	LAMA-W($f_2^{h^{LM}}$)	LAMA-W($w_{(h^{FF})}$)	LAMA-W($w_{(h^{LM})}$)	LAMA-W($w_{()}$)
Coverage	2237	2237	2243	2210	**2277**	2136	2276	2243
Expansion Score	1794.2	1551.6	1551.9	1516.9	**1817.0**	1417.0	1806.5	1797.1
Agile Score	1863.9	1769.1	1775.3	1747.2	**1888.4**	1681.0	1886.3	1868.4

Table 2: Scores for the baselines, LAMA and LAMA-W(f_6), and for simplifications of LAMA-W(f_6) on the old IPC tasks.

	BFWS(f_6)	LAMA	NOLAN
BFWS(f_6)	—	13	10
LAMA	**13**	—	2
NOLAN	**17**	**14**	—

Table 3: Per-domain coverage comparison for old IPC tasks.

Another successful approach is due to Fickert (2020), who uses an orthogonal approach to the one by Katz et al.: instead of using novelty as the main guidance for the search, Fickert uses traditional heuristics to guide a greedy best-first search, and uses a lookahead strategy to find states with lower heuristic values quickly. This lookahead strategy is designed to reach states satisfying relaxed subgoals (Lipovetzky and Geffner 2014). To make the procedure efficient, he uses novelty pruning (Lipovetzky and Geffner 2012; Fickert 2018) to reduce the number of evaluated states. Fickert shows that there is a synergy between the novelty-based lookahead and the h^{CFF} heuristic (Fickert and Hoffmann 2017), as the result from the lookahead can be used to trigger the refinement procedure of h^{CFF}. This idea was also used in the OLCFF planner (Fickert and Hoffmann 2018) from the IPC 2018. We include OLCFF and an improved version of it, GBFS-RSL, in our empirical comparison below.

State-of-the-Art Agile Planners

We now compare NOLAN to state-of-the-art agile planners which scored highly in previous IPCs: BFWS-Preference (Francès et al. 2018), the winner of the Agile Track of IPC 2018; Dual-BFWS (Francès et al. 2018), another BFWS-based participant of IPC 2018; OLCFF (Fickert and Hoffmann 2018), which combines novelty pruning with the h^{CFF} heuristic (Hoffmann and Fickert 2015); DecStar (Gnad, Torralba, and Shleyfman 2023), the winner of the IPC 2023 Agile Track; and Fast Downward Stone Soup 2023 (Büchner et al. 2023), the runner-up in the IPC 2023 Agile Track. Furthermore, we include LAMA, BFWS(f_6) with $V = 100$, and GBFS-RSL (Fickert 2020), which is an improved version of the OLCFF planner. As explained above, we use the new IPC set of instances in this experiment for fairness.

Table 4 compares NOLAN with the state-of-the-art planners.[3] NOLAN has the highest coverage and agile score and

[3]We do not compare expansion scores for this experiment because some planners perform multiple searches and some use lookaheads, which skews the total number of expansions.

	Coverage	Agile Score
NOLAN	**204**	**89.30**
LAMA	199	87.25
BFWS(f_6)	174	75.57
FDSS	171	76.22
DecStar	167	76.05
GBFS-RSL	142	57.87
BFWS-Pref.	133	63.59
Dual-BFWS	131	57.48
OLCFF	124	52.44

Table 4: Coverage and agile scores on the new IPC tasks.

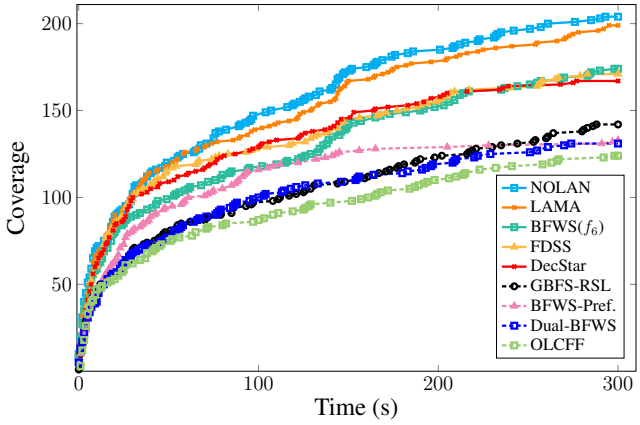

Figure 2: Coverage over time for state-of-the-art agile planners on the new IPC tasks.

the relative increase in agile score to DecStar, the last winner of the IPC Agile Track, is about 17%. Figure 2 shows that NOLAN solves more tasks than all other planners already after 75 seconds.

Conclusions

We showed how to combine two successful agile planners, LAMA and BFWS(f_6), yielding a planner that is stronger than its ingredients. We also introduced a new mechanism to choose the width k of a BFWS based on the structure of the task. Putting these ideas together, we obtained NOLAN, which has a higher coverage and agile score than all other evaluated planners.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proc. ICAPS 2015*, 2–6.

Asai, M.; and Fukunaga, A. 2017. Exploration Among and Within Plateaus in Greedy Best-First Search. In *Proc. ICAPS 2017*, 11–19.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *AIJ*, 129(1): 5–33.

Büchner, C.; Christen, R.; Corrêa, A. B.; Eriksson, S.; Ferber, P.; Seipp, J.; and Sievers, S. 2023. Fast Downward Stone Soup 2023. In *IPC-10 Planner Abstracts*.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *AIJ*, 69(1–2): 165–204.

Corrêa, A. B.; Francès, G.; Hecher, M.; Longo, D. M.; and Seipp, J. 2023. Scorpion Maidu: Width Search in the Scorpion Planning System. In *IPC-10 Planner Abstracts*.

Corrêa, A. B.; and Seipp, J. 2025. Code and experiment data from the ICAPS 2025 paper "Alternation-Based Novelty Search". https://doi.org/10.5281/zenodo.15313799.

Doran, J. E.; and Michie, D. 1966. Experiments with the Graph Traverser program. *Proceedings of the Royal Society A*, 294: 235–259.

Fickert, M. 2018. Making Hill-Climbing Great Again through Online Relaxation Refinement and Novelty Pruning. In *Proc. SoCS 2018*, 158–162.

Fickert, M. 2020. A Novel Lookahead Strategy for Delete Relaxation Heuristics in Greedy Best-First Search. In *Proc. ICAPS 2020*, 119–123.

Fickert, M.; and Hoffmann, J. 2017. Complete Local Search: Boosting Hill-Climbing through Online Relaxation Refinement. In *Proc. ICAPS 2017*, 107–115.

Fickert, M.; and Hoffmann, J. 2018. OLCFF: Online-Learning h^{CFF}. In *IPC-9 Planner Abstracts*, 17–19.

Francès, G.; Geffner, H.; Lipovetzky, N.; and Ramiréz, M. 2018. Best-First Width Search in the IPC 2018: Complete, Simulated, and Polynomial Variants. In *IPC-9 Planner Abstracts*, 23–27.

Francès, G.; Ramírez, M.; Lipovetzky, N.; and Geffner, H. 2017. Purely Declarative Action Representations are Overrated: Classical Planning with Simulators. In *Proc. IJCAI 2017*, 4294–4301.

Gnad, D.; Torralba, Á.; and Shleyfman, A. 2023. DecStar-2023. In *IPC-10 Planner Abstracts*.

Helmert, M. 2006. The Fast Downward Planning System. *JAIR*, 26: 191–246.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *AIJ*, 173: 503–535.

Hoffmann, J.; and Fickert, M. 2015. Explicit Conjunctions w/o Compilation: Computing $h^{\mathrm{FF}}(\Pi^C)$ in Polynomial Time. In *Proc. ICAPS 2015*, 115–119.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *JAIR*, 14: 253–302.

Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2017. Adapting Novelty to Classical Planning as Heuristic Search. In *Proc. ICAPS 2017*, 172–180.

Lipovetzky, N.; and Geffner, H. 2012. Width and Serialization of Classical Planning Problems. In *Proc. ECAI 2012*, 540–545.

Lipovetzky, N.; and Geffner, H. 2014. Width-based Algorithms for Classical Planning: New Results. In *Proc. ECAI 2014*, 1059–1060.

Lipovetzky, N.; and Geffner, H. 2017. Best-First Width Search: Exploration and Exploitation in Classical Planning. In *Proc. AAAI 2017*, 3590–3596.

Nakhost, H.; and Müller, M. 2009. Monte-Carlo Exploration for Deterministic Planning. In *Proc. IJCAI 2009*, 1766–1771.

Richter, S.; and Helmert, M. 2009. Preferred Operators and Deferred Evaluation in Satisficing Planning. In *Proc. ICAPS 2009*, 273–280.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *Proc. AAAI 2008*, 975–982.

Richter, S.; and Westphal, M. 2008. The LAMA Planner — Using Landmark Counting in Heuristic Search. IPC 2008 short papers, http://ipc.informatik.uni-freiburg.de/Planners.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *JAIR*, 39: 127–177.

Richter, S.; Westphal, M.; and Helmert, M. 2011. LAMA 2008 and 2011 (planner abstract). In *IPC 2011 Planner Abstracts*, 50–54.

Röger, G.; and Helmert, M. 2010. The More, the Merrier: Combining Heuristic Estimators for Satisficing Planning. In *Proc. ICAPS 2010*, 246–249.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *JAIR*, 67: 129–167.

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

Shleyfman, A.; Tuisov, A.; and Domshlak, C. 2016. Blind Search for Atari-Like Online Planning Revisited. In *Proc. IJCAI 2016*, 3251–3257.

Vidal, V. 2011. YAHSP2: Keep It Simple, Stupid. In *IPC 2011 Planner Abstracts*, 83–90.