Hitting Set Heuristics for Overlapping Landmarks in Satisficing Planning

Clemens Büchner, Remo Christen, Salomé Eriksson, Thomas Keller

University of Basel, Switzerland
{clemens.buechner, remo.christen, salome.eriksson, tho.keller}@unibas.ch

Abstract

Landmarks are a core component of LAMA, a state-of-the-art satisficing planning system based on heuristic search. It uses landmarks to estimate the goal distance by summing up the costs of their cheapest achievers. This procedure ignores synergies between different landmarks: The cost of an action is counted multiple times if it is the cheapest achiever of several landmarks. Common admissible landmark heuristics tackle this problem by underapproximating the cost of a minimum hitting set of the landmark achievers. We suggest to overapproximate it by computing suboptimal hitting sets instead if admissibility is not a requirement. As our heuristics consider synergies between landmarks, we further propose to relax certain restrictions LAMA imposes on the number of landmarks and synergies between them. Our experimental evaluation shows a reasonable increase in the number of landmarks that leads to better guidance when used with our new heuristics.

Introduction

Classical planning aims to find a sequence of actions leading from an initial state to a goal state in a deterministic transition system (Ghallab, Nau, and Traverso 2004). To this day, LAMA (Richter and Westphal 2010) is a competitive approach to finding suboptimal solutions for classical planning problems. This was recently demonstrated at the International Planning Competition (IPC) 2023, where LAMA was used as a baseline; it beat all competitors in the *agile* track (finding any solution quickly) and almost made the podium in the *satisficing* track (finding cheap solutions).

LAMA approaches planning as a series of explicit *heuristic searches*, guided by the h^{FF} and h^{sum} heuristics. The former sums up the costs of best achievers of relevant atoms in the delete-relaxation to estimate the goal distance (Hoffmann and Nebel 2001), the latter sums up the costs of the cheapest achiever of each *landmark* (Richter and Westphal 2010). The landmarks considered by LAMA are sets of atoms such that one atom from each set must hold in some state along all plans. Since an action may achieve multiple landmarks, adding up the costs of each individual cheapest achiever may overestimate the true cost to satisfy all landmarks. Common admissible landmark heuristics for optimal planning avoid this overcounting by underapproximating the cost of a *minimum hitting set* (Bonet and Helmert 2010; Pommerening et al. 2014; Büchner, Keller, and Helmert 2021), since an exact computation is NP-complete (Karp 1972). We instead study two approaches that *overapproximate* this cost by finding suboptimal hitting sets computable in polynomial time in the number of landmarks, and use their cost as an inadmissible heuristic for satisficing planning. Our first approach fixes one cheapest achiever for each landmark, making the set of these fixed achievers a hitting set. The second uses a known greedy approximation.

Beyond proposing new landmark heuristics, we also revisit the algorithm LAMA employs to *find* landmarks. LAMA's landmark generation restricts landmarks to be disjoint sets of atoms with cardinality 4 or less (Richter, Helmert, and Westphal 2008). These restrictions ensure that the landmark generation algorithm is polynomial in the task representation. However, limiting the number of landmarks also limits the informedness of the heuristic. We thus relax both restrictions. Firstly, we remove the requirement for disjointedness, since our heuristics are designed to consider interactions between landmarks. Secondly, we increase the maximal cardinality. These changes still guarantee a polynomial bound, albeit with a larger factor.

Finally, we compare our heuristics and generation changes against LAMA in two contexts: a single search focusing on the landmark side of LAMA, and a fully fledged LAMA-style planning system. While our first approach achieves higher coverage in the single search setting, it can also lead to higher plan costs. The second approach on the other hand takes longer to compute but finds better plans, resulting in the best IPC score in the full planner comparison.

Background

We consider classical planning in the SAS$^+$ formalism (Bäckström and Nebel 1995). A *planning task* is a 4-tuple $\Pi = \langle V, A, I, G \rangle$ where V is a finite set of *finite-domain state variables*; A is a finite set of *actions* (or *operators*); I is a *state*; and G is a *partial state*. Each variable $v \in V$ has an associated finite domain $\mathit{dom}(v)$. An *atom* (or *fact*) $v \mapsto d$ is an assignment mapping a variable $v \in V$ to a value in its domain $\mathit{dom}(v)$. A *partial state* s is a set of atoms, each over a different variable. With $\mathit{vars}(s) = \{v \mid v \mapsto d \in s \text{ for some } d\}$ we denote the variables defined by

s, and $s(v)$ denotes the value d of v in s. A partial state is called a *state* if $\mathit{vars}(s) = V$. Each *action* $a \in A$ is a triple $\langle \mathit{pre}(a), \mathit{eff}(a), \mathit{cost}(a) \rangle$ where *precondition* $\mathit{pre}(a)$ and *effect* $\mathit{eff}(a)$ are partial states, and $\mathit{cost}(a) \in \mathbb{R}_0^+$.

An action $a \in A$ is *applicable* in state s if $\mathit{pre}(a) \subseteq s$. Applying an applicable action a in s leads to the *successor state* $s' = s[\![a]\!]$ where $s'(v) = d$ if $v \mapsto d \in \mathit{eff}(a)$ and $s'(v) = s(v)$ otherwise. An *action sequence* $\pi = \langle a_1, \ldots, a_n \rangle$ is applicable in s if a_1 is applicable in s, a_2 is applicable in $s[\![a_1]\!]$ and so forth, and we denote the resulting state by $s[\![\pi]\!]$. If $s[\![\pi]\!] \supseteq G$, then π is called an *s-plan*. The *cost* of π is defined as $\mathit{cost}(\pi) = \sum_{i=1}^{n} a_i$. The aim of classical planning is to find an I-plan (also just called plan).

A *disjunctive action landmarks* (or *landmark*) for a state s of planning task $\Pi = \langle V, A, I, G \rangle$ is a set of actions $l^A \subseteq A$ such that for all s-plans $\pi = \langle a_1, \ldots, a_n \rangle$ it holds that $\{a_1, \ldots, a_n\} \cap l^A \neq \emptyset$. Some of the literature we discuss instead considers *disjunctive fact landmarks*, which are sets of atoms l^F such that all s-plans visit a state containing some atom in l^F. More formally, there exists an $s' \in \{s[\![\langle a_1, \ldots, a_i \rangle]\!] \mid 0 \leq i \leq n\}$ such that $s' \cap l^F \neq \emptyset$. These landmarks can be translated into disjunctive action landmarks, for example with the set of actions that contain an atom in l^F as an effect, i.e., $l^A = \{a \in A \mid \mathit{eff}(a) \cap l^F \neq \emptyset\}$. We call the actions in l^A the *achievers* of l^F and l^A.

Hitting Set Heuristics

Given state s and an associated set of landmarks $\mathcal{L}$, $h^{\max}$ sums up the cheapest landmark achiever costs: $h^{\max}(s) = \sum_{l \in \mathcal{L}} \min_{a \in l} \mathit{cost}(a)$. While fast to compute, it ignores the possibility that landmarks overlap, meaning that one action may achieve several landmarks. For example, given $\mathcal{L} = \{\{a_1, a_2\}, \{a_1, a_3\}\}$ with $\mathit{cost}(a_1) = 1$ and $\mathit{cost}(a_2) = \mathit{cost}(a_3) = 2$, we get $h^{\max}(s) = \mathit{cost}(a_1) + \mathit{cost}(a_1) = 2$ even though a single application of a_1 with a cost of 1 satisfies both landmarks. Considering an alternative cost function $\mathit{cost}' = \mathit{cost}$ except $\mathit{cost}'(a_1) = 3$, $h^{\max}(s) = \mathit{cost}'(a_2) + \mathit{cost}'(a_3) = 4$, even though applying only a_1 again achieves both landmarks with only a cost of 3.

To account for overlapping landmarks, we aim to find a *set of actions* that achieves all landmarks. This can be modeled as a *weighted hitting set problem*: Given a universe of elements U and a set $\mathcal{S}$ consisting of sets of elements from U, along with a cost function $\mathit{cost} : U \to \mathbb{R}$, a hitting set is a set $H \subseteq U$ that "hits" each element of $\mathcal{S}$, that is $H \cap S \neq \emptyset$ for all $S \in \mathcal{S}$. The cost of H is the sum of all its elements, i.e., $\mathit{cost}(H) = \sum_{u \in H} \mathit{cost}(u)$. By setting $U = A$, $\mathcal{S} = \mathcal{L}$, and cost as the action cost function, finding a set of actions that achieves all landmarks means finding a hitting set for $\mathcal{L}$. The cost of the hitting set can then be used in the heuristic.

Starting from $h^{\max}$, a straightforward way to calculate a hitting set while maintaining computational efficiency is to start with the collection of actions that $h^{\max}$ would sum over, and then remove all duplicates. The resulting collection is a set because there are no duplicates, and it hits all landmarks because only duplicate actions are removed. We denote the resulting heuristic "hitting sum", or h^{hs}. Since $h^{\max}$ may have multiple cheapest achievers, we pick the first minimal cost action according to a fixed ordering.

Algorithm 1: Greedy Hitting Set Heuristic h^{ghs}

Input: set of landmarks $\mathcal{L}$
Output: heuristic estimate h^{ghs}
1: $h^{\text{ghs}} \leftarrow 0$
2: **while** $\mathcal{L} \neq \emptyset$ **do**
3: select $a \in A$ with minimal $\frac{\mathit{cost}(a)}{|\{l \in \mathcal{L} \mid a \in l\}|}$
4: $h^{\text{ghs}} \leftarrow h^{\text{ghs}} + \mathit{cost}(a)$
5: $\mathcal{L} \leftarrow \{l \in \mathcal{L} \mid a \notin l\}$
6: **return** h^{ghs}

While this approach avoids the problem of counting the same action multiple times (illustrated by the example with cost function *cost*) the problem of summing multiple cheaper actions instead of a single expensive action (illustrated by the example with cost function *cost*′) still persists. To also address this second issue, we instead use a well-known greedy approximation of small hitting sets, originally described for the equivalent *set cover* problem (Chvátal 1979; Ausiello, D'Atri, and Protasi 1980). Algorithm 1 is an adaption written in the hitting set perspective and computes our greedy hitting set heuristic h^{ghs}. It iteratively selects actions such that the ratio between the cost of the action and the number of newly covered landmarks is minimal, until all landmarks are covered. In the example with cost function *cost*′, this approach selects a_1 in line 3 during the first iteration already because it appears in two landmarks, rendering its ratio lower than for a_2 and a_3.

Preferred Operators LAMA performs heuristic search with multiple queues, where some queues only contain states reached by *preferred operators* (Helmert 2006). A heuristic can flag certain actions as preferred, indicating that it deems them important for advancing towards the goal. While h^{FF} provides preferred operators as a side product of the heuristic computation, $h^{\max}$ performs an extra step on top of its heuristic computation to flag applicable actions that achieve landmarks. In contrast, the hitting sets computed by our new heuristics are exactly the set of actions the heuristic deems important, and we instead flag those actions as preferred operators without imposing any additional effort.

Landmark Generation

Planners using landmark heuristics also need a way to *generate* landmarks. While this topic is orthogonal to how the heuristic uses landmarks, the properties of the generated landmark set can influence the performance of the heuristic, as the example with overlapping landmarks and $h^{\max}$ shows. We investigate the generator used in LAMA to identify possible changes that might improve our hitting set heuristics.

The LAMA landmark generator finds disjunctive fact landmarks in a backwards fashion. It first creates a fact landmark for each goal atom, and then iteratively finds new landmarks: Given landmark l it tries to find a (small) set of atoms that contains at least one precondition of all actions achieving l. The number of landmarks is limited to be polynomial in the task representation through several mechanisms. Most importantly for us, the generator disallows overlap between

the disjunctive fact landmarks, that is, every atom appears in at most one landmark. Note that the implied disjunctive action landmarks can still overlap, since one action may achieve several atoms. Furthermore, LAMA limits the disjunction size of landmarks to 4, with the intuitive motivation that smaller disjunctions are harder to satisfy. Smaller landmarks also block fewer atoms from being used in other landmarks, potentially leading to more landmarks overall.

Since our new heuristics account for overlap between landmarks, we propose to allow overlapping disjunctive fact landmarks. While this change primarily targets our new heuristics, we remark that overlapping landmarks can contain valuable information in general. Consider for example a problem with atoms x, y, z and the landmarks $\{x, y\}$, $\{x, z\}$, and $\{y, z\}$. Since every landmark overlaps with the other two, LAMA considers at most one of them, assuming it is sufficient to achieve one of the atoms when we need to achieve at least two.

To ensure landmarks do not overlap, LAMA discards a newly generated landmark l^{new} if it overlaps with a previously found landmark l^{old}. The one exception to this rule is if $|l^{\text{new}}| = 1$, in which case it discards l^{old} instead. Consider the example above with the additional landmark $\{x\}$ and initially $\mathcal{L} = \{\{x, y\}\}$. Assume the algorithm now finds the new landmark $\{y, z\}$ and discards it since it overlaps with $\{x, y\}$. Next it finds $\{x\}$, which replaces $\{x, y\}$ in $\mathcal{L}$. At this point, $\{y, z\}$ does not overlap with any landmark in $\mathcal{L}$, but the generator no longer knows about it and returns fewer landmarks than it could.

When allowing overlap, we add all landmarks in both examples. To keep the overall number of landmarks tractable, we still enforce a maximum size limit but increase the constant since the disadvantage of large landmarks blocking atoms no longer applies. In order to avoid redundant information (e.g., $\{x, y\}$ is always satisfied when $\{x\}$ is), we add a post-processing step that removes all landmarks $l \in \mathcal{L}$ for which a landmark $l' \subset l$ exists in $\mathcal{L}$.

Experimental Evaluation

We implemented our approach on top of Fast Downward version 23.06 (Helmert 2006), which also contains a modernized implementation of LAMA. LAMA is an anytime planner aiming to find some plan fast and repeatedly restarts the search with different parameters to find cheaper plans while time permits. In order to analyze our hitting set heuristics and changes to the landmark generation in isolation, we first run a simplified version of LAMA's first iteration, removing the h^{FF} heuristic and the use of preferred operators. In a second step we test them as fully fledged planning systems. Our experiments were conducted on AMD EPYC 7742 2.25GHz processors, using a time limit of 30 minutes and memory limit of 3.5 GiB. The results were evaluated with Downward Lab (Seipp et al. 2017). Our benchmark suite consists of 2882 planning tasks from the satisficing track of the IPCs 1998–2023. All code, benchmarks and experimental data are publicly available (Büchner et al. 2024).

Fast Downward utilizes a formalism where computing the achievers of a disjunctive fact landmark is not always straightforward. In particular, if a disjunctive fact landmark contains *derived variables* (Edelkamp and Hoffmann 2004), it uses the set of all actions as a trivial overapproximation. For h^{hs} and h^{ghs} this results in all such landmarks being covered by one single action, rendering them significantly less informed compared to h^{sum}. We thus also report results restricted to domains with no derived variables.

		all domains			no derived variables		
		h^{sum}	h^{hs}	h^{ghs}	h^{sum}	h^{hs}	h^{ghs}
4	○○	1984	1982	1959	1647	1705	1681
	◎◎	1956	1969	1954	1619	1692	1676
10	○○	**1995**	**2004**	1975	**1666**	**1728**	1697
	◎◎	1976	**2004**	**1986**	1647	**1728**	**1708**

Table 1: Coverage results between h^{sum}, h^{hs}, and h^{ghs}. The left-most column indicates the maximal landmark size and the following column whether overlapping is allowed (◎◎) or not (○○). The right side restricts from all 90 domains to those 81 without derived variables.

Simplified First Iteration

To analyze the impact of our proposed landmark generation changes, we separately controlled disjunction size and overlap. For the disjunction size we tried values from 4 to 14 in increments of 2. When not allowing overlap, the number of landmarks across all tasks only increased slightly (from $\approx 484\,000$ for 4 to $\approx 511\,000$ for 14), but with overlap we observed two larger jumps from 6 to 8 ($\approx +90\,000$) and from 10 to 12 ($\approx +85\,000$). Table 1 reports the coverage results for 4 (baseline) and 10 (best overall performance).

Disjunction size The configurations using the larger disjunction size dominate across all used heuristics. We find that h^{sum} solves the fewest additional tasks and h^{hs} the most, and that configurations with overlap generally benefit more. The increase in the number of landmarks stays reasonable; using both increased landmark size and overlap yields 42% more landmarks compared to using neither. The time required compared to the original landmark generation is usually less than doubled, staying below 10 seconds for the majority of problems.

Overlap While allowing larger disjunctions is generally beneficial, not all configurations benefit from overlapping landmarks. The h^{sum} heuristic consistently performs worse with overlap, confirming that it is undesirable to consider overlapping landmarks independently. Our new heuristics perform worse with overlap and the default disjunction size, while for disjunction size 10 overlap has no influence for h^{hs} and a positive influence for h^{ghs}. We assume that overlap introduces some degree of inaccuracy for *all* considered heuristics. This results in a trade-off between the usefulness of the additional landmark information and the inaccuracies from the overlap. As we progress from h^{sum} over h^{hs} to h^{ghs} the heuristics become better equipped to handle overlap. Similarly, as we progress to larger disjunction sizes, the overlap between landmarks becomes more likely to be small relative to the landmark size, harming the heuristic less.

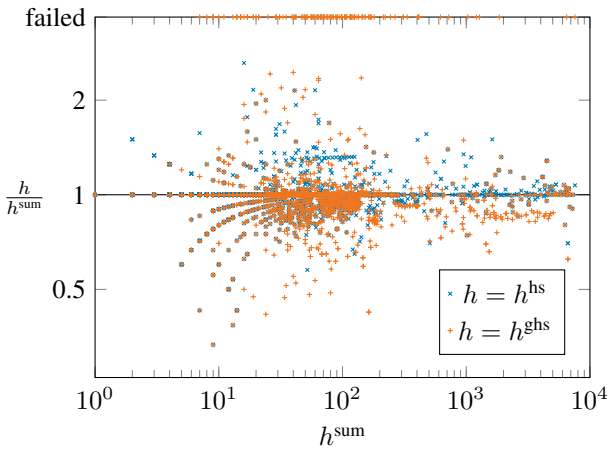

Figure 1: Relative cost of plans found by h^{sum} compared against those found by h^{hs} and h^{ghs}. Parcprinter has plan costs in the order 10^6 and is excluded to improve clarity.

Heuristics Finally, we compare the heuristics against each other by considering their respective best performing versions, namely size 10 without overlap for h^{sum}, and size 10 with overlap for both h^{hs} and h^{ghs}. Taking h^{sum} as a baseline, h^{hs} improves coverage, solving more tasks in 17 domains and fewer in 17. Coverage drops for h^{ghs} on the other hand, which solves more tasks in 16 domains compared to h^{sum}, and fewer in 20. On domains without derived variables, both h^{hs} and h^{ghs} achieve significantly higher coverage than h^{sum}.

We also studied the costs of plans found. Figure 1 compares the cost of h^{sum} to that of h^{hs} and h^{ghs}, respectively. While h^{hs} often finds worse plans than h^{sum}, especially as costs for h^{sum} get higher, h^{ghs} generally finds better plans, suggesting that the heuristic offers better guidance. However, this comes at the expense of higher computation time, leading to fewer solved problems. This can also be seen in the reasons for failure: While h^{sum} and h^{hs} time out in 5 tasks, h^{ghs} does so in up to 194 tasks, depending on the configuration.

Comparison against LAMA

For a full planner comparison we introduce LAMA-h^{hs} and LAMA-h^{ghs}. They are identical to LAMA except for the heuristic used and changes to disjunction size and overlap. Since h^{sum} improved with disjunction size 10, we further include LAMA-10 that differs only in the disjunction size.

The top part of Table 2 shows the IPC quality score and coverage of each planner using the standard LAMA settings. The IPC quality score is the best plan cost[1] divided by the found plan cost. We see that the coverage improvement for h^{sum} with larger disjunction size and h^{hs} do not carry over, most likely because h^{FF} compensates in domains where h^{sum} is weaker. We also again see the negative impact of h^{hs} in terms of cost, as its IPC quality score is significantly worse. However, LAMA-h^{ghs} is much closer to LAMA in terms of IPC quality score, compensating its comparatively low coverage with better plans.

[1]The minimum of all found plan's costs and the upper bound from planning.domains (Muise 2016, accessed on April 5, 2024).

pref. op.		all domains		no derived var.	
		score	cov.	score	cov.
FF	LAMA	2359.58	**2457**	1959.78	2055
	LAMA-10	2352.82	2455	1952.54	2052
	LAMA-h^{hs}	2325.73	2425	1954.07	2052
	LAMA-h^{ghs}	2348.65	2428	1976.03	2054
FF + LM	LAMA	2331.20	2438	1930.84	2036
	LAMA-10	2323.94	2432	1925.88	2032
	LAMA-h^{hs}	2304.81	2403	1934.98	2032
	LAMA-h^{ghs}	**2364.48**	2444	**1989.87**	**2068**

Table 2: IPC quality score and coverage of LAMA-like planners.

The default configuration of LAMA only considers preferred operators from the h^{FF} heuristic, because the computation of h^{sum} preferred operators is too expensive. Since the hitting set heuristics compute preferred operators as a side product, we tested the impact of taking the preferred operators from landmark heuristics into account. The results are shown in the bottom part of Table 2. They confirm that preferred operators from h^{sum} are harmful for standard LAMA. Similarly they negatively affect LAMA-h^{hs}, presumably because the found hitting set is not a good approximation. For LAMA-h^{ghs} we however see a significant improvement, resulting in the highest IPC quality score despite lower coverage. This indicates that the hitting sets found by h^{ghs} are a good representation of which actions should be applied. When only considering domains without derived variables, LAMA-h^{ghs} with additional preferred operators from h^{ghs} achieves both the highest coverage and highest IPC quality score among all configurations.

Conclusion

The landmark heuristic used in LAMA can be inaccurate due to ignoring synergies between landmarks. We propose two hitting-set-based heuristics that take synergies into account: h^{hs} creates a hitting set by fixing one action as the achiever for each landmark, while h^{ghs} uses a greedy hitting set algorithm that is more expensive to compute but yields higher quality hitting sets. Together with allowing overlaps in the disjunctive fact landmarks, increasing the maximal disjunction size, and utilizing the hitting set as a source for preferred operators, trading computation cost for better guidance pays off for h^{ghs}: Its LAMA-like planner is on par with LAMA, and surpasses it on domains without derived variables.

An important open question is how to improve our heuristics on domains with derived variables. Furthermore, investigating sophisticated tie-breaking criteria for h^{hs} may improve its guidance. Finally, we believe LAMA can be further improved by using different heuristics at different stages, e.g. using the evaluation speed of h^{hs} or h^{sum} in the first iteration to find a solution fast and using the improved guidance of h^{ghs} afterwards to find cheaper plans.

Acknowledgments

This research was supported by TAILOR, a project funded by the EU Horizon 2020 research and innovation programme (grant agreement no. 952215) and by the Swiss National Science Foundation (SNSF) as part of the project "Lifted and Generalized Representations for Classical Planning" (LGR). A special thanks goes to Malte Helmert who supported this work with helpful recommendations and discussions.

References

Ausiello, G.; D'Atri, A.; and Protasi, M. 1980. Structure Preserving Reductions among Convex Optimization Problems. *Journal of Computer and System Sciences*, 21(1): 136–153.

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

Bonet, B.; and Helmert, M. 2010. Strengthening Landmark Heuristics via Hitting Sets. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 329–334. IOS Press.

Büchner, C.; Christen, R.; Eriksson, S.; and Keller, T. 2024. Code, Benchmarks and Experiment Data for the SoCS 2024 Paper "Hitting Set Heuristics for Overlapping Landmarks". https://doi.org/10.5281/zenodo.10948623.

Büchner, C.; Keller, T.; and Helmert, M. 2021. Exploiting Cyclic Dependencies in Landmark Heuristics. In Goldman, R. P.; Biundo, S.; and Katz, M., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 65–73. AAAI Press.

Chvátal, V. 1979. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3): 233–235.

Edelkamp, S.; and Hoffmann, J. 2004. PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition. Technical Report 195, University of Freiburg, Department of Computer Science.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

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

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

Karp, R. M. 1972. Reducibility among combinatorial problems. In Miller, R. E.; and Thatcher, J. W., eds., *Complexity of Computer Computations*, 85–103. Plenum Press.

Muise, C. 2016. Planning.Domains. In *ICAPS 2016 System Demonstrations and Exhibits*.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-based Heuristics for Cost-optimal Planning. In Chien, S.; Fern, A.; Ruml, W.; and Do, M., eds., *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, 226–234. AAAI Press.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks Revisited. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, 975–982. AAAI Press.

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.