Improving Plan Quality through Heuristics for Guiding and Pruning the Search: A Study Using LAMA

Francesco Percassi,* **Alfonso Emilio Gerevini,*** **Hector Geffner†**

*Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Brescia, Italy
{f.percassi,alfonso.gerevini}@unibs.it
† Departament de Tecnologies de la Informacio i les Comunicacions, Universitat Pompeu Fabra, Barcelona, Spain
hector.geffner@upf.edu

Abstract

Admissible heuristics are essential for optimal planning in the context of search algorithms like A*, and they can also be used in the context of suboptimal planning in order to find quality-bounded solutions. In satisfacing planning, on the other hand, admissible heuristics are not exploited by the best-first search algorithms of existing planners even when a time window is available for improving the first solution found. For example, in the well-know planner LAMA, better solutions within such a time window are sought by restarting a Weighted-A* search guided by inadmissible heuristics, each time a better solution is found. In this paper, we investigate the use of admissible heuristics in the context of LAMA for pruning nodes that cannot lead to better solutions. The revised search of LAMA is experimentally evaluated using two alternative admissible heuristics for pruning and three types of problems: planning with soft goals, planning with action costs, and planning with both action costs and soft goals. Soft goals are compiled into hard goals following the approach of Keyder and Geffner. The empirical results show that the use of admissible heuristics in LAMA can be of great help to improve the planner performance.

Introduction

While both satisficing and optimal planners rely on heuristics for guiding or pruning the search, the nature of these heuristics are different. Optimal planners rely on heuristics that are admissible, ensuring that the solutions found are provably best. Satisficing planners, on the other hand, rely on non-admissible heuristics, that while not ensuring optimality, provide better guidance for reaching the goal fast (Geffner and Bonet 2013; Ghallab, Nau, and Traverso 2016). In the well-known planner LAMA, for example, increasing quality solutions are sought within a time window by restarting a Weighted-A* (WA*) search guided by two inadmissible heuristics, with reduced weights, each time a better solution is found (Richter and Westphal 2010).

In this paper, we aim to show that the search strategy used in LAMA can significantly benefit from the use of a second, admissible heuristic for pruning nodes that cannot lead to better solutions. This is indeed a common idea in general best-first search branch and bound algorithms (Applegate et al. 2011) but less common in planning where admissible heuristics tend to be either expensive or uninformed. We show nonetheless that these heuristics can be cost-effective.

The revised search of LAMA is experimentally evaluated using two alternative admissible heuristics for pruning and three types of problems: planning with soft goals, planning with action costs, and planning with both action costs and soft-goals. Each of these types is of high interest in the planning community, and it was addressed in planning competitions. The empirical results show that the use of admissible heuristics in LAMA can play a crucial role in the anytime setting of this planner to improve its performance. Moreover, the use of admissible heuristics in LAMA makes the compilation approach to planning with soft goals (Keyder and Geffner 2009) competitive with the state-of-the-art planners that natively support them.

The focus of our work is studying the use of admissible heuristics for pruning in the context of satisficing and anytime planning with LAMA. The work that is mostly related to ours is AEES (Thayer, Benton, and Helmert 2012), a variant of EES (Thayer and Ruml 2011) adopting the "bounded suboptimal search" framework. AEES is a general anytime best-first search algorithm, not specifically targeted to planning. Both our versions of LAMA and AEES exploit admissible heuristics, but there are also significant differences: standard LAMA performs search restarting, while AEES continues the same search after finding a solution; the techniques for selecting the nodes from the open lists are different and use different weight parameters; the combination of the used heuristics is different. Moreover, our experimental analysis with LAMA is much more developed than the evaluation of AEES for domain-independent planning in (Thayer, Benton, and Helmert 2012). Finally, LAMA exploits some additional techniques that are specific for planning while AEES is a general search algorithm.

Search Pruning in Planning with LAMA

In this section, after the necessary background on compiling soft goals and planning with LAMA, we introduce our variants of LAMA exploiting admissible heuristics for pruning search nodes.

The problems involving soft goals are precompiled into problems with action costs and no soft goals (Keyder and Geffner 2009). The net benefit of a plan π is the difference

between the utility $u(\pi)$ of the plan and its cost $c(\pi)$. The utility is the sum of the given utilities $u(p)$ associated with the soft goals p such that p holds in the state that results from the plan. The cost of a plan $\pi = a_0, \ldots, a_n$ is the sum of the action costs $c(a_i)$. A problem P with soft goals is compiled into a problem P' with no soft goals by adding a new atom p' for each soft goal p that is made into a hard goal of the problem P'. Two new actions are introduced in P' for achieving the hard goal p': a *collect*(p) action with precondition p, effect p', and zero cost, and a *forgo*(p) action with empty preconditions, effect p', and cost $u(p)$. Keyder and Geffner show that the plans that maximize the net-benefit in P correspond to the plans that minimize the sum of action costs in the compiled problem P' with no soft goals. Satisficing and optimal solutions to the net-benefit problem P can then be obtained from satisficing and optimal solutions to P'. In the compilation additional atoms are used to force the *collect*/*forgo* actions in the plan to appear at the end of it, and to order them for removing redundant permutations.

Once problems with soft goals have been compiled, one has to deal with the standard setting of satisficing planning with action costs only. However, for testing we will distinguish between pure soft goal problems, soft goal problems with actions costs, and problems with action costs only.

LAMA is a satisficing planner built on top of the more general architecture of Fast Downward (Helmert 2006). In LAMA, solutions are improved by restarting a WA* search with the weight reduced, each time a better solution is found. The WA* in LAMA reopens nodes when cheaper paths to the state represented by a node are found. The first solution is computed by a slightly different search algorithm, namely GBFS, which is basically WA* with a very large weight for the heuristic component. For problems with non-unit action costs, a second solution is then computed, also by GBFS, but using a cost-sensitive heuristic. In both cases, better solutions are sought then using restarting WA*.

LAMA uses inadmissible heuristics during these searches. In the first search, always a GBFS, LAMA uses the FF-heuristic h_{FF} (Hoffmann and Nebel 2001) in the form described in (Keyder and Geffner 2008), using min $h_{\max}$ supports under the assumption that action costs are all 1. In the following searches, however, if the problem involves non-unit action costs, LAMA uses a cost-sensitive version of the FF-heuristic, obtained from the min h_{add} supports and action costs set to the true action costs plus 1. The value of the heuristic in both cases is given by the sum of the action costs used in the computation of the supports, over the actions that occur in the resulting relaxed plan. In a sense, the heuristic used in the first search estimates distance to the goal, while the heuristic in the successive searches estimates a cost to the goal that combines distance estimates, given by the +1 term, and cost estimates given by the true action costs.

To test the idea of using pruning heuristics in LAMA's search strategy, we developed two variations of this planner:

- $\text{LAMA}_{\text{P}}(h)$, where "P" stands for pruning, that follows exactly the same search strategy as LAMA, but pruning in restarting WA* search nodes using the admissible heuristic function h;
- $\text{LAMA}_{\text{B}}(h)$, where "B" stands for branch and bound, that that mimics LAMA during the GBFS search (two GBFS searches if there actions with non-unit costs), but then WA* is run with *no restarts*, pruning nodes using the admissible heuristic h.

The two admissible heuristics h that we use in $\text{LAMA}_{\text{P}}(h)$ and $\text{LAMA}_{\text{B}}(h)$ are an admissible action landmark heuristic h_{R} and the well-known LM-cut heuristic $h_{\text{LM-Cut}}$ (Helmert and Domshlak 2009). The first heuristic is developed ad-hoc for problems where soft goals have been compiled away and it is defined as follows: $h_{\text{R}}(s)$ *is the sum over the costs of all forgo*(p_i) *such that* p_i *(the precondition of collect*(p_i)*) is unReachable from* s *in the delete-relaxed problem*. Note that $h_{\text{R}}(s)$ can be computed as a variant of the original cost-sensitive FF-heuristic where the costs of all non-*forgo* actions are set to 0.

While h_{R} is a simple heuristic for problems with soft goals, the heuristic is also related to $h_{\text{LM-Cut}}$ and action landmarks (Karpas and Domshlak 2009). If we say that a STRIPS action is a *landmark action* iff it belongs to all the plans, then the heuristic $h_{\text{LA}}(s)$ defined as the size of a set LA of landmark actions computed from the problem initial state that have not been done in the way to s is an admissible heuristic. The heuristic h_{R} is equivalent to the cost-sensitive landmark heuristic $h_{\text{LA}}(s)$ when LA is defined as the set of *forgo* action landmarks in the delete-relaxed of the compiled problem. The LM-cut heuristic is also an admissible action landmark heuristic, and it dominates h_{R} as it considers disjunctions of action landmarks as well (Helmert and Domshlak 2009).

As usual, a search state s is pruned in $\text{LAMA}_{\text{P}}(h)$ or $\text{LAMA}_{\text{B}}(h)$ when the pruning condition $g(s) + h(s) \geq B$ is true, where $g(s)$ stands for the accumulated cost up to s and B stands for the cost of the best plan found so far. Since $h_{\text{LM-Cut}}$ is expensive to compute, *the evaluation of the pruning condition with* $h_{\text{LM-Cut}}$ *is done only if the condition is true for the (inadmissible)* h_{FF} *heuristic used by LAMA for guiding the search*. That is, if this pruning condition fails for h_{FF}, the state s is not pruned and $h_{\text{LM-Cut}}$ is not evaluated. Thus a state is actually pruned when suggested by the inadmissible heuristic and then confirmed by the admissible heuristic. Note that, since $h_{\text{LM-Cut}} \leq h_{\text{FF}}$, the pruning is both sound and complete; i.e., all and only nodes that can be pruned according to the admissible heuristic $h_{\text{LM-Cut}}$ are pruned. It follows that $\text{LAMA}_{\text{P}}(h)$ and $\text{LAMA}_{\text{B}}(h)$ are both *anytime optimal* when the heuristic h is admissible. That is, both will find an optimal solution if given sufficient time and memory, and otherwise will just report a satisfying solution.

Experimental Results

We experimentally evaluated the performance of LAMA_{P} and LAMA_{B} considering three types of problems:

Soft Goals (SG) Planning with zero action costs and (compiled) soft goals.

Net Benefit (NB) Planning with non-zero action costs and (compiled) soft goals, called net-benefit planning (Van den Briel et al. 2004)

$\text{LAMA}_\text{B}(h_\text{R})$						
Pure Soft Goals Domains (#problem)	%pruned	better	worse	equal	search reduction	optimal
Trucks (40)	65.87	24	3	13 (0)	0.03	19 (12)
TPP (40)	24.09	12	2	26 (1)	0.22	8 (7)
Storage (40)	41.94	5	1	34 (14)	0.14	7 (7)
Openstacks (40)	63.31	31	2	7 (4)	0.04	7 (0)
Pegsol (30)	54.28	0	0	30 (0)	0.16	30 (28)
Pathways (30)	93.71	0	0	30 (25)	0.04	4 (4)
Total (220)	**64.48**	**72**	**8**	**140 (44)**	**0.12**	**75 (58)**
$\text{LAMA}_\text{B}(h_\text{LMCut})$						
Pure Soft Goals Domains (#problem)	%pruned	better	worse	equal	search reduction	optimal
Trucks (40)	62.02	23	3	14 (0)	0.03	19 (12)
TPP (40)	24.09	12	2	26 (1)	0.22	8 (7)
Storage (40)	32.14	6	1	33 (14)	0.14	7 (7)
Openstacks (40)	55.86	31	2	7 (4)	0.04	6 (0)
Pegsol (30)	54.29	0	0	30 (0)	0.03	28 (28)
Pathways (30)	93.71	0	0	30 (25)	0.04	4 (4)
Total (220)	**61.17**	**72**	**8**	**140 (44)**	**0.05**	**72 (58)**

Table 1: LAMA_B versus LAMA without restarts for SG problems. "%pruned": % of nodes pruned over all visited nodes for all instances of the domain; "better"/"worse"/"equal": instances where LAMA_B finds a solution better/worse/equal relative to LAMA's solution (in brackets instances equally solved by the first GBFS); "search reduction": ratio between the overall numbers of nodes expanded by LAMA_B in relation to LAMA, from time of first solution to a solution that agrees in quality; "optimal": instances where LAMA_B (and in brackets LAMA) finds provably optimal solutions .

Action Costs (AC) Planning with non-zero action costs and no soft goals.

Plan quality is defined as the weighted sum of violated soft goals for SG, the difference between the utility of the achieved soft goals and the total costs of the plan actions for NB, and the sum of plan action costs for AC. For each of the three problem types, we evaluate LAMA_P and LAMA_B, and compare them with LAMA. For the SG and NB problems, we also give results showing that LAMA with pruning is competitive with two state-of-the-art planners that natively support soft goals: OPTIC (Benton, Coles, and Coles 2012) and GBL15-NB-B15 (Coles and Coles 2013).[1]

The benchmark domains that we consider are all those of the last four IPCs that are supported by the planners. These are: for SG, five domains from IPC-2006 (Trucks, TPP, Storage, Pathways and Openstacks) and one domain from the Net Benefit track of IPC-2008 where all action costs are zero (Pegsol); for NB, two domains from IPC-2008 (Elevators and Openstacks-08) and a domain from IPC-2006 that has both soft goals and non-zero action costs (Rovers); for AC, all the domains from IPC-2011 and IPC-2014 except those with conditional effects (because h_LMCut does not support them) and a domain for which LAMA finds no solution.

[1]These planners were run using the original (uncompiled) problems. We thank the authors for making their code available.

$\text{LAMA}_\text{P}(h_\text{R})$						
Net Benefit Domains (#problems)	%pruned	better	worse	equal	search reduction	optimal
Rovers (20)	47.26	1	0	19 (0)	0.22	8 (8)
Openstacks-08 (30)	32.52	27	0	3 (0)	< 0.01	6 (3)
Elevators (30)	47.33	0	0	30 (0)	0.5	11 (12)
Total (80)	**40.53**	**28**	**0**	**52 (0)**	**0.15**	**25 (23)**
$\text{LAMA}_\text{P}(h_\text{LMCut})$						
Net Benefit Domains (#problems)	%pruned	better	worse	equal	search reduction	optimal
Rovers (20)	58.9	0	1	19 (0)	0.03	13 (8)
Openstacks-08 (30)	30.7	27	0	3 (0)	< 0.01	6 (3)
Elevators (30)	67.19	0	2	28 (0)	0.02	18 (12)
Total (80)	**45.64**	**27**	**3**	**50 (0)**	**0.01**	**37 (23)**

Table 2: LAMA_P versus LAMA for NB problems. "%pruned": % of nodes pruned over all visited nodes for all instances of the domain; "better"/"worse"/"equal": instances where LAMA_P finds a solution that is better/worse/equal relative to LAMA's solution (in brackets the instances equally solved by the initial greedy search); "search reduction": ratio between the overall numbers of nodes expanded by LAMA_P in relation to LAMA, from time of first solution to a solution that agrees in quality; "optimal": instances where LAMA_P (and in brackets LAMA) finds provably optimal solutions.

Planner	TOTAL SG	TOTAL NB	TOTAL SG+NB
$\text{LAMA}_\text{B}(h_\text{R})$	215.17	77.77	292.94
$\text{LAMA}_\text{P}(h_\text{R})$	210.04	79.76	289.8
$\text{LAMA}_{\text{B}(h_\text{LMCut})}$	208.39	77.32	285.71
$\text{LAMA}_\text{P}(h_\text{LMCut})$	203.76	79.36	283.12
LAMA2011	184.77	73.67	258.44

Table 3: Total IPC scores for the SG benchmarks and NB benchmarks of $\text{LAMA}_\text{P/B}$ compared with standard LAMA.

Overall, we used 32 domains and 760 instances.

All the experiments were conducted on a 2.00GHz Core Intel(R) Xeon(R) CPU E5-2620 machine with CPU-time and memory limits of 30 minutes and 8GiB, respectively, for each run of every tested planner. The time for the soft goal compilation was included in the 30 minutes. All versions of LAMA were run with its default weights of WA*.

For lack of space we present a selection of the results. In particular, for each considered benchmark domain, we compare the performance of LAMA with: $\text{LAMA}_\text{B}(h_\text{R})$ and $\text{LAMA}_\text{B}(h_\text{LMCut})$ for solving SG problems; $\text{LAMA}_\text{P}(h_\text{R})$ and $\text{LAMA}_\text{P}(h_\text{LMCut})$ for solving NB problems; and $\text{LAMA}_\text{P}(h_\text{LMCut})$ for solving AC problems (h_R is undefined without soft goals). Moreover, for SG and NB problems, we compare every considered version of LAMA in terms of total IPC score (introduced at IPC-2008) using all domains together (Table 3). As reference plan in the IPC score definition of a problem, we used the best plan generated by the compared planners solving the problem.

Regarding the use of LAMA_P for SG problems and LAMA_B for NB and AC problems, we observed that they give results that in general are less good (see Table 3). In particular, LAMA_B for NB and AC appears to be less per-

Planner	TOTAL SG	TOTAL NB	TOTAL SG+NB
GBL15-NB-B15	164.74	75.18	239.92
$\text{LAMA}_{\text{P}}(h_{\text{R}})$	161.99	77.72	239.71
$\text{LAMA}_{\text{B}}(h_{\text{R}})$	154.84	75.74	230.58
OPTIC	157.68	62.41	220.09

Table 4: Total IPC scores for the SG benchmarks and NB benchmarks of $\text{LAMA}_{\text{P/B}}(h_{\text{R}})$, OPTIC and GBL15-NB-B15. Domain Openstacks was not used for NB because it contains conditional effects supported only by LAMA_{P} and LAMA_{B}.

formant than LAMA_{P} because the lack of restarts decreases the number of solutions (Richter, Thayer, and Ruml 2010) and so also the bounds to use for the pruning.

Tables 1-2 and 5 show aggregated results for LAMA versus LAMA_{B} and LAMA_{P}, while Table 4 compares two of the considered versions of LAMA with pruning with two state-of-the-art planners supporting soft goals. In Tables 1-2 and 5, every planner solved the same subset of the problems.

Column "%pruned" is the percentage of search nodes that are pruned during search over all visited nodes for all instances of the domain (the first greedy search of LAMA_{P} and LAMA_{B} is not considered because in this search pruning is not applied). Columns "better", "worse" and "equal" give the number of instances in which LAMA with pruning finds a solution that is better, worse and equal, respectively, with respect to the solution of LAMA without pruning; the values in brackets in the "equal" column indicate the number of instances where both compared planners found just one solution (through their initial greedy search). Column "search reduction" gives, for each domain, the ratio between the total numbers of nodes expanded by LAMA with pruning in relation to LAMA without pruning, after their first greedy search, considering for each instance the nodes expanded in all search episodes until a solution that agrees in quality is found by both planners.[2] This ratio is a measure of the overall search space reduction obtained with the pruning. Column "optimal" gives the number of instances where an optimal plan was found by fully exploring the search space; the values in brackets are for LAMA without pruning, the others for LAMA with pruning.

Results for pure soft goals problems (SG). For this class of problems, we compare $\text{LAMA}_{\text{P/B}}$ with a version of standard LAMA where the restarts of WA* are disabled. This is because we observed that, for the SG benchmarks, LAMA without restarts performs generally better than the standard LAMA.[3]

The results in Table 1 show that $\text{LAMA}_{\text{B}}(h_{\text{R}})$ does considerable pruning, obtaining in Trucks, TPP and Openstacks

[2] All instances for which only one of the two planners finds a solution after the first GBFS are ignored. If both planners find more solutions of the same increasing qualities, the last of them is considered.

[3] In LAMA without restarts as in LAMA_{B} the search continues after finding a solution, with the search heuristic weighted according to the same default sequence of decreasing weights in LAMA's WA* (weights are changed when an improved solution is found).

AC domains of IPC-2011 (#problems)	%pruned	better	worse	equal	search reduction	optimal
Tidybot (20)	10.78	1	0	15 (0)	0.96	4 (0)
Visitall (20)	8.41	2	5	13 (5)	0.99	0 (0)
Nomystery (20)	19.44	4	0	9 (3)	0.03	7 (3)
Woodworking (20)	69.14	3	0	17 (12)	0.67	1 (1)
Transport (20)	0.07	0	0	17 (12)	1.0	0 (0)
Floortile (20)	36.4	1	0	5 (1)	0.17	3 (2)
Openstacks-11 (20)	74.03	0	9	11 (0)	0.24	0 (0)
Elevators (20)	0.14	0	0	20 (19)	0.99	0 (0)
Barman (20)	0.0	0	0	20 (19)	1.0	0 (0)
Pegsol (20)	37.13	2	0	18 (0)	0.44	17 (17)
Scanalyzer (20)	40.24	8	3	9 (0)	0.21	1 (0)
Parking (20)	7.34	1	4	15 (5)	0.81	0 (0)
Sokoban (20)	2.28	0	3	16 (5)	0.54	10 (7)
Parcprinter (20)	34.46	11	0	9 (4)	0.33	3 (0)
Total (280)	**27.36**	**33**	**24**	**194 (85)**	**0.33**	**46 (30)**
AC Domains of IPC-2014 (#problems)	%pruned	better	worse	equal	search reduction	optimal
Visitall (20)	0.0	0	0	20 (17)	1.0	0 (0)
Openstacks-14 (20)	0.01	3	1	16 (0)	0.98	0 (0)
GED (20)	1.43	0	4	16 (1)	0.96	0 (0)
thoughtful (20)	41.85	3	2	10 (1)	0.04	5 (1)
Barman (20)	—	0	0	19 (19)	—	0 (0)
Parking (20)	2.73	2	2	16 (7)	0.98	0 (0)
Tetris (20)	6.18	0	0	7 (6)	0.92	0 (0)
Transport (20)	0.01	0	0	12 (8)	1.0	0 (0)
Floortile (20)	20.86	0	0	2 (0)	0.35	2 (2)
Total (180)	**12.32**	**8**	**9**	**118 (59)**	**0.43**	**7 (3)**

Table 5: $\text{LAMA}_{\text{P}}(h_{\text{LMCut}})$ versus LAMA for AC problems. "%pruned": % of nodes that are pruned over all visited nodes for all instances of the domain; "better"/"worse"/"equal": instances in which LAMA_{P} finds a solution that is better/worse/equal relative to the solution of LAMA (in brackets the instances equally solved by the first GBFS); "search reduction": ratio between the overall numbers of nodes expanded by LAMA_{P} in relation to LAMA, from time of first solution to a solution that agrees in quality; "optimal": instances where LAMA_{P} (and in brackets LAMA) finds provably optimal solutions; "–": for every instance of the domain, no one of the planners improved the 1st solution computed by the first GBFS.

solutions that are often better, and finding optimal solutions more often too. In Pegsol, $\text{LAMA}_{\text{B}}(h_{\text{R}})$ and LAMA find the same solutions (almost always proved optimal) but $\text{LAMA}_{\text{B}}(h_{\text{R}})$ expands much less nodes. In Storage $\text{LAMA}_{\text{B}}(h_{\text{R}})$ performs slighty better and in Pathways the pruning is ineffective. In terms of optimally solved problems, overall $\text{LAMA}_{\text{B}}(h_{\text{R}})$ finds provably optimal solutions for 17 more instances. The results for $\text{LAMA}_{\text{B}}(h_{\text{LMCut}})$ versus LAMA are similar.

Table 3 shows that for SG $\text{LAMA}_{\text{P/B}}(h_{\text{R}})$ performs better than $\text{LAMA}_{\text{P/B}}(h_{\text{LMCut}})$ and significantly better than LAMA, in terms of total IPC score over the considered 8 domains. Table 4 shows that for SG $\text{LAMA}_{\text{P}}(h_{\text{R}})$ performs slightly better than OPTIC and slightly worse than GBL15-NB-B15. The problem coverage of the compared planners is the same.

Results for net benefit problems (NB). Table 2 shows that LAMA_{P} does considerable pruning, and performs dra-

matically better in Openstacks, where the pruning is very effective, as indicated by the search reduction value. In the other two domains, while the pruning significantly reduces the search search space, no better solution is produced, except for one instance in Rovers using h_{R}. While pruning with h_{LMCut} gives a stronger search reduction than h_{R}, it leads to the same number of better solutions and to 3 worse solutions, probably due to the higher cost of computing h_{LMCut}. On the other hand, pruning with h_{LMCut} gives a higher number of problems that are optimally solved (12 more instances).

Table 4 shows that for NB LAMA$_{\mathrm{P}}(h_{\mathrm{R}})$ is the best performing planner,[4] and for both SG and NB (SG+NB) it is competitive with the state-of-the-art planners supporting soft goals. The problem coverage of the compared planners for the considered domains is the same.

Results for action costs problems (AC). Table 5 gives results for the considered AC problems. The IPC-2011 and IPC-2014 benchmarks are shown separate as the results are qualitatively different in them. The 2014 benchmarks tend to be significantly harder for LAMA, and, with CPU time and memory limits used in our experiments, a large number of them are unsolved or solved only by the initial greedy search, providing an initial quality bound that is not improved within the given CPU time. In the IPC-2011 benchmarks, LAMA$_{\mathrm{P}}$ computes better solutions for 33 instances and worse solutions for 24 instances. The better solutions of LAMA$_{\mathrm{P}}$ are distributed over 9 domains and the worse solutions over 5 domains. In 9 domains the search reduction is significant and in 4 domains it is negligible or nonexistent. LAMA$_{\mathrm{P}}$ generated provably optimal solutions for 16 more problems distributed over 5 domains, which gives further evidence that the search space can be significantly reduced by the pruning also in the AC setting.

The picture is less clear for the IPC-2014 benchmarks. Under our experimental settings, the use of pruning for these benchmarks does not lead to significant improvements and the compared planners perform similarly.

Conclusions

While admissible heuristics are essential for optimal planning, they are seldom used in satisficing planners, even when there is a time window for improving the first solution found. In this work we have shown, however, that existing and new admissible heuristics can be effectively used for pruning in a statisficing planner like LAMA. Even heuristics such as LM-cut that are expensive, turn out to be often effective. In the case of planning with soft goals, cheaper but less informative admissible heuristics can be more cost-effective. Our experimental results confirm that the use of admissible heuristics can play a crucial role also in the practical settings of satisficing and anytime optimal planning.

[4] In the experiment of Table 3 LAMA$_{\mathrm{B}}(h_{\mathrm{R}})$ performs better than LAMA$_{\mathrm{P}}(h_{\mathrm{R}})$ because this experiment includes a domain for SG (Openstacks), where LAMA$_{\mathrm{P}}(h_{\mathrm{R}})$ performs best, that is not used in the experiment of Table 4 since it not supported by OPTIC and GBL15-NB-B15.

Acknowledgements

We thank the anonymous reviewers for their detailed and helpful comments.

References

Applegate, D.; Bixby, R.; Chvatal, V.; and Cook, W. 2011. *The traveling salesman problem: a computational study*. Princeton University Press.

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*.

Coles, A. J., and Coles, A. 2013. Searching for good solutions in goal-dense search spaces. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling*.

Geffner, H., and Bonet, B. 2013. *A concise introduction to models and methods for automated planning*. Morgan & Claypool Publishers.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What is the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*.

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.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 1728–1733.

Keyder, E., and Geffner, H. 2008. Heuristics for planning with action costs revisited. In *Proceedings of the 18th European Conference on Artificial Intelligence*.

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research* 36:547–556.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:122–177.

Richter, S.; Thayer, J. T.; and Ruml, W. 2010. The joy of forgetting: Faster anytime search via restarting. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*.

Thayer, J. T., and Ruml, W. 2011. Bounded suboptimal search: A direct approach using inadmissible estimates. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 674–679.

Thayer, J. T.; Benton, J.; and Helmert, M. 2012. Better parameter-free anytime search by minimizing time between solutions. In *Proceedings of the Fifth Annual Symposium on Combinatorial Search*, 120–128.

Van den Briel, M.; Sanchez, R.; Do, M. B.; and Kambhampati, S. 2004. Effective approaches for partial satisfaction (oversubscription) planning. In *Proceedings of the 19th National Conference on Artificial Intelligence*.