Landmark Heuristics for Lifted Classical Planning

Julia Wichlacz, **Daniel Höller** and **Jörg Hoffmann**
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
{wichlacz, hoeller, hoffmann}@cs.uni-saarland.de

Abstract

While state-of-the-art planning systems need a grounded (propositional) task representation, the input model is provided "lifted", specifying predicates and action schemas with variables over a finite object universe. The size of the grounded model is exponential in predicate/action-schema arity, limiting applicability to cases where it is small enough. Recent work has taken up this challenge, devising an effective lifted forward search planner as basis for lifted heuristic search, as well as a variety of lifted heuristic functions based on the delete relaxation. Here we add a novel family of lifted heuristic functions, based on *landmarks*. We design two methods for landmark extraction in the lifted setting. The resulting heuristics exhibit performance advantages over previous heuristics in several benchmark domains. Especially the combination with lifted delete relaxation heuristics to a LAMA-style planner yields good results, beating the previous state of the art in lifted planning.

1 Introduction

Heuristic search is a dominant paradigm in planning (e. g. [Hoffmann and Nebel, 2001; Helmert and Domshlak, 2009; Richter and Westphal, 2010; Seipp, 2019]). Yet, this success is based on grounded task representations, in contrast to the actual PDDL input which is *lifted*, specifying predicates and action schemas parameterized with variables ranging over a finite universe of objects. The grounded representation has size exponential in the arity (number of arguments) of the PDDL predicates and action schemas. This effectively limits its current heuristic search planning to planning tasks whose grounded representation is small enough to be feasible. It has been frequently observed that, for various applications, this is not naturally the case (e. g. [Hoffmann *et al.*, 2006; Koller and Hoffmann, 2010; Koller and Petrick, 2011; Haslum, 2011; Matloob and Soutchanski, 2016]).

Lifted planning methods do not require to ground the entire task representation as a pre-process. This has received attention throughout the history of AI planning (e. g. [Penberthy and Weld, 1992; Russell and Norvig, 1995; Younes and Simmons, 2003]. Heuristic search planning, however, has begun to address lifted planning only recently.

Corrêa *et al.* [2020] have devised an effective lifted forward search mechanism, grounding actions on demand in state expansion, in the *Power Lifted Planner (PWL)* through a connection to database queries. But how to transfer the wealth of known planning heuristics, which are based on reachability analyses over the grounded task representation? A simple option is goal counting, $h^{\mathrm{GC}}(s) := |\{g \mid g \text{ goal fact and not true in s}\}|$ which does not require such an analysis. All other approaches explored thus far are delete relaxation heuristics [Bonet and Geffner, 2001; Hoffmann and Nebel, 2001], which ignore negative action effects (the "delete lists") and have been addressed in the lifted setting through object symmetries [Ridder and Fox, 2014], a link to database technology [Corrêa *et al.*, 2021] and through a further relaxation which splits the predicates into unary predicates [Lauer *et al.*, 2021].

Here we contribute a third family of lifted heuristics, which has been very successful in grounded planning, but is – so far – not available in the lifted setting: heuristics based on *landmarks* [Hoffmann *et al.*, 2004]. Landmarks (LMs) are facts that have to be true at some point along every plan, or actions that need to be contained in every plan. Given a set L of LMs, goal distance is estimated by the number of unseen LMs. Landmark heuristics have been widely explored in planning [Karpas and Domshlak, 2009; Helmert and Domshlak, 2009; Richter and Westphal, 2010], in various forms and extensions, including ones incorporating disjunction and leveraging cost partitioning for admissibility. They are suited also for lifted planning, as the set L can be computed once before search, limiting the computational overhead incurred by the required reachability analysis. Especially a search guided by a combination of delete relaxation heuristics and LMs as used by the (grounded) LAMA system [Richter *et al.*, 2008] has proven to be very effective in practice.

We introduce two methods to extract lifted LMs by adapting methods known from grounded planning [Hoffmann *et al.*, 2004; Richter *et al.*, 2008]. We combine a heuristic based on these LMs with the lifted delete relaxation heuristic by Corrêa *et al.* [2021] to a system similar to LAMA. Our experiments show that (while only using LMs is less effective) this combination outperforms the state of the art in lifted planning, making landmark heuristics a valuable tool in this area.

2 Preliminaries

We first give some background on lifted planning, and introduce lifted landmarks afterwards.

2.1 Lifted Planning

A lifted planning task is a tuple $\Pi = (\mathcal{P}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ where $\mathcal{P}$ is a set of (first-order) *predicates*, $\mathcal{A}$ is a set of *action schemas*, $\mathcal{O}$ is a set of *objects*, $\mathcal{I}$ is the *initial state*, and $\mathcal{G}$ is the *goal*. Predicates $P \in \mathcal{P}$ have an *arity* k and every occurrence in Π has k *parameters*, written $P(x_1, \ldots, x_k)$ where the x_i are variables; we write $\mathcal{X}$ for the set of all variables used in Π. Parameters can be *grounded* (instantiated) with objects $\mathcal{O}$. We write $P(u_1, \ldots, u_k)$ to denote a partially grounded predicate along with its arguments $u_i \in \mathcal{X} \cup \mathcal{O}$. If $u_i \in \mathcal{O}$ for all i then $P(u_1, \ldots, u_k)$ is a *ground predicate* or *atom*, which we write as p. By $\mathcal{P}^{\mathcal{O}}$ we denote the set of all ground atoms in Π. States (in particular the initial state) are sets of ground atoms. The goal also is a set of ground atoms.

An *action schema* A is a tuple $(X_A, pre(A), add(A), del(A))$ with a set of parameter variables X_A as well as *precondition*, *add list*, and *delete list*, all of which are sets of predicates parameterized with variables from X_A. The arity of A is $|X_A|$. We can instantiate action schemas by replacing each $x \in X_A$ by some $o \in \mathcal{O}$ to obtain *ground actions* a. The set of ground actions, or *actions* for short, is $\mathcal{A}^{\mathcal{O}}$.

Action a is applicable in state s if $pre(a) \subseteq s$. Applying a to s results in the state $(s \setminus del(a)) \cup add(a)$. A *plan* for Π is a sequence π of ground actions that is iteratively applicable in $\mathcal{I}$ and results in a state s' such that $\mathcal{G} \subseteq s'$.

2.2 Landmarks

We next give limited background on landmarks as needed in this paper. We then introduce a simple generalization to the lifted (partially grounded) setting.

The Grounded Case. We first consider "fact landmarks" and "action landmarks". A ground atom p is a *landmark* of Π if, for every plan π for Π, there exists a state s traversed by π where $p \in s$. A ground action a is a *landmark* of Π if, for every plan $\pi = (a_1, \ldots, a_n)$ for Π, there exists an index i with $a_i = a$.

We also require notions of orderings between landmarks. For the following ordering definition, consider the execution of a plan as a sequence $(s_0, a_1, s_2, a_3, \ldots a_{n-1}, s_n)$ of actions a_i and states s_i resulting from executing the plan with $s_0 = \mathcal{I}$ and s_{i+1} the state resulting from the application of a_i in s_{i-1}. Let a be an action LM, f be a fact landmark, and p and q be state *or* action landmarks of Π. Then:

- p is *ordered before* q (written $p \to q$), when in every plan for Π, p is fulfilled (contained in the state, or executed, respectively) strictly before q is fulfilled;
- f is *ordered directly before* q (written $f \to_D q$) if in every plan, when q is fulfilled (added to the state, or executed, respectively) f holds.
- a is *ordered directly before* q (written $a \to_D q$) if in every plan, when q is fulfilled (added or executed) a has been executed at the preceding action step.

2.3 Lifted Landmarks

As we ground only partially in our landmark extraction methods, we naturally end up with partially grounded landmarks:

Definition 1 *A partially grounded predicate $P(u_1, \ldots, u_k)$ is a* lifted landmark *of Π if there exists a ground instance p of $P(u_1, \ldots, u_k)$ that is a landmark of Π.*

We remark that, apart from being natural algorithmically in lifted landmark extraction, Definition 1 also carries some novelty for grounded planning. It captures a simple special case of disjunctive landmarks [Hoffmann *et al.*, 2004; Helmert and Domshlak, 2009], which is extremely compact and easy to extract. Consider e. g. an object o that needs to be transported with one of a fleet of trucks: the lifted landmark *in(o, x)* compactly represents the fact that o will have to be inside one of the trucks at some point.

3 Landmark Extraction

Deciding whether a ground atom is a landmark is **PSPACE**-complete in grounded planning, because the decision is closely linked to solvability: intuitively, something is a landmark iff the task cannot be solved without it. There is however a wealth of *landmark extraction* methods based on sufficient criteria, finding some but not necessarily all LMs. A key trick is to consider delete relaxation, where solvability – and therewith the landmark-decision question – can be decided in time polynomial in the size of the grounded encoding. We adapted two such methods to the lifted setting: *necessary subgoals* and landmarks based on *Domain Transition Graphs*.

3.1 Necessary Subgoals

Our first method was introduced by Hoffmann *et al.* [2004] for the grounded case. It is based on a simple form of backchaining: starting at the goal atoms g, consider all actions a adding g, and intersect their preconditions $\bigcap_a pre(a)$; clearly, any $p \in \bigcap_a pre(a)$ is a landmark ordered directly before g, $p \to_D g$; iterate the process on p until no more new landmarks can be derived.

For example, in Blocksworld, *holding(A)* and *clear(B)* are necessary subgoals for *on(A,B)*, and in turn *clear(A)* and *arm-free()* are necessary subgoals for *holding(A)*.

In the lifted setting, we also start at the ground goal atoms g, but in general the subgoals considered take the form of partially grounded predicates $P(u_1, \ldots, u_k)$. Given such a $P(u_1, \ldots, u_k)$, we consider all action schemas $A = (X_A, pre(A), add(A), del(A))$ where P occurs in $add(A)$. We partially ground each A as needed to match the objects $u_i \in \mathcal{O}$ instantiating the subgoal, obtaining a partially grounded action of the form $A(v_1, \ldots, v_m)$ with $v_i \in X_A \cup \mathcal{O}$. We then intersect the preconditions of these $A(v_1, \ldots, v_m)$, but only on the *predicates*: we produce a new subgoal from every predicate Q that occurs in all preconditions. Let these preconditions be $Q(v_{11}, \ldots, v_{m1}), \ldots, Q(v_{1n}, \ldots, v_{mn})$. Then, the new subgoal is $Q(w_1, \ldots, w_n)$ where

$$w_i := \begin{cases} o & w_{ij} = o \text{ for all } j \\ x_i & \text{otherwise} \end{cases}$$

where the x_i are arbitrary variables. We then iterate the process as before.

$Q(w_1, \ldots, w_n)$ represents a least commitment where we instantiate only those parameters agreed upon by all preconditions.[1] Hence, the extraction process is sound:

Proposition 1 *The subgoals $Q(w_1, \ldots, w_n)$ computed as above are lifted landmarks of* Π.

Proof: The goal atoms trivially are lifted landmarks. In each subsequent step of the algorithm, assume as induction hypothesis that the subgoal $P(u_1, \ldots, u_k)$ addressed is a lifted landmark. Then every plan must make a ground instance p of $P(u_1, \ldots, u_k)$ true. The ground action a doing so must be a ground instance of a partially grounded action $A(v_1, \ldots, v_m)$ considered by the algorithm. But then, by construction, a has a precondition instantiating $Q(w_1, \ldots, w_n)$, showing the claim. ∎

3.2 FAM-Cut Landmarks

Our second method is inspired by a method from grounded planning based on *Domain Transition Graphs (DTGs)*, e.g. used by the LAMA system [Richter *et al.*, 2008, p. 3].

In grounded planning, the basic propositional representation is (usually) compiled to a finite domain representation before planning, i.e., one where variables have a finite range of values (see, e.g. the Fast Downward system [Helmert, 2006]). Consider e.g. a simple transport domain. Here, a package might either be *at* a location (e.g. a city), or *in* a transporter and in a single state it will be at exactly one such place: the values are mutually exclusive, they form a *mutex group*. Such structures are detected in a preprocessing step.

DTGs. A DTG is a directed graph capturing all transitions between values of such a variable x. Values of x form its nodes, while edges between the nodes capture transitions between the values, i.e., actions. [Richter *et al.*, 2008, Def. 2].

DTG-based LMs. Given a landmark f_g consisting of a single fact (initially a fact from the goal definition) not contained in the initial state, one can obtain landmarks by determining which parts (nodes or edges) of the DTG need to be passed on *every* way from the initial value f_0 of this variable to the goal value (or, the other way around, which elements need to be cut to make the goal unreachable). Richter *et al.* iteratively remove nodes v from the DTG and test reachability of f_g starting from f_0: when f_0 and f_g have been connected before removing v but are not without v, all paths from f_0 to f_g pass v, and it forms a LM [Richter *et al.*, 2008, p. 3].

FAM-Cut-LMs. We introduce a similar method for lifted planning. It is based on Lifted Fact-Alternating Mutex (FAM groups) as introduced by Fišer [2020]: a lifted FAM group is a tuple $\nu = (\mathcal{V}^{fix}[\nu], \mathcal{V}^{cnt}[\nu], atoms(\nu))$ with $\mathcal{V}^{fix}[\nu] \cap \mathcal{V}^{cnt}[\nu] = \emptyset$. For the set of atoms $atoms(\nu) = \{P_1(v^1_1, \ldots, v^1_{k_1}), \ldots, P_l(v^l_1, \ldots, v^l_{k_l})\}$, all variables are from $\mathcal{V}^{fix}[\nu] \cup \mathcal{V}^{cnt}[\nu]$.

[1] Note that, if Q appears $n > 1$ times within a single precondition, then we could actually construct n lifted landmarks, and indeed combinations across similar choices in other preconditions. This possibility did not seem relevant in the benchmarks we consider here, but it may be an interesting avenue for future work.

The semantics are the following: For a given assignment of fixed variables, the full grounding of the counted variables forms a mutex group [Fišer, 2020].

Analogously to DTG landmarks, we generate FAM-cut LMs in the following way: Given a known fact LM g (initially a fact from the goal definition), we identify the respective FAM group and its value in the initial state $f_{\mathcal{I}} = P_{\mathcal{I}}(v^{\mathcal{I}}_1, \ldots, v^{\mathcal{I}}_{k_{\mathcal{I}}})$. Now we build a graph that – similar to the ground DTG – captures *all* transitions from $f_{\mathcal{I}}$ to g. The transitions are induced by the actions of the task. There are three ways atoms of a FAM group can be contained in an action definition:

1. It contains the atom in its preconditions (but not in the effects)
2. It contains the atom in its delete effects (and maybe but not necessarily in its preconditions)
3. It contains the atom in its preconditions, delete, and add effects

Case 1 does not change the value, we can ignore it. Case 2 results in a dead end in our transition graph: it deletes the value, but from the properties of FAM groups, we know that there cannot be an action adding it without having it also in its preconditions (otherwise we could use it to add a second value). Thus, case 2 will not contribute to the transitions from the initial state to the landmark either. What we are interested in is case 3: actions where atoms from the FAM group are contained in the precondition, the delete effects, and in the add effects, since they form value transitions in our graph. It should be noted that there might be more that one combination of precondition, add, and delete effects from a single action contributing transitions in the graph.

Now we build our graph capturing all value transitions in the FAM group. We start in $f_{\mathcal{I}} = P_{\mathcal{I}}(v^{\mathcal{I}}_1, \ldots, v^{\mathcal{I}}_{k_{\mathcal{I}}})$ and build it in a forward manner until it converges. We identify all actions from case 3 and for each add a transition to the graph. For each action, we only bind the parameter variables contained in $P_{\mathcal{I}}(v^{\mathcal{I}}_1, \ldots, v^{\mathcal{I}}_{k_{\mathcal{I}}})$. So the graph can contain partially grounded nodes.

Figure 1 shows the graph for a transport domain, where *package_0* that is currently at a *city_loc_1* must be delivered to a target position. Each unassigned variable (starting with "?") might be any object of the particular type, i.e., we are (as before) generating a special type of disjunctive landmarks.

We now start LM generation by first building the set of nodes unifiable with our known landmark g, which we call G. If G includes $f_{\mathcal{I}}$, we return the empty set of landmarks. Else, we generate the set C of incoming edges of G, which forms our candidate for a disjunctive action landmark. However, it might contain edges not connecting $\mathcal{I}$ with G. Therefore we test which edges from C have to be cut to make G unreachable from $\mathcal{I}$. The resulting set forms our first disjunctive action landmark L_i. We add the starting nodes from edges in L_i to G and continue the process until we have reached $f_{\mathcal{I}}$.

Theorem 1 *The generated cuts $\{L_1, \ldots, L_n\}$ form disjunctive action landmarks for* Π.

Proof: We first need to show that our graph represents all transitions of that particular FAM group. Consider it does not, then there must be an action adding one of its values

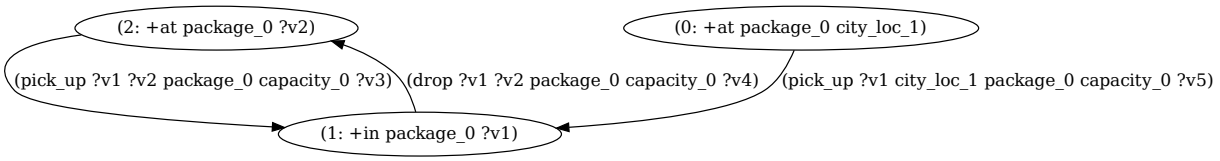

Figure 1: Lifted DTG generated based on a FAM group. When building the graph, we check whether static preconditions of actions fully determine the value of a variable and set such values (here: the capacity in the actions).

not represented in the graph. However, due to the properties of FAM groups, every such action must also delete a value, as given above in case 3. Since we constructed the graph using these actions up to convergence, we know that every transition is represented and our graph contains all transitions in that FAM group. Thus, every cut between $\mathcal{I}$ and G must be passed to reach a goal value when starting from the initial state, including our cuts. As a result, one of the actions in our cut must be in every plan, and the cuts form landmarks. ∎

We intersect the preconditions of the resulting disjunctive action LMs as done for necessarily subgoal extraction. This results in new fact LMs, these are used for FAM-based extraction. This process is continued until no new LMs are found.

3.3 Ordering Relations

The above extraction methods deliver ordering relations as a side effect: $\rightarrow$ relations for FAM-Cut landmarks, $\rightarrow_D$ relations for necessary subgoals. For FAM-Cut landmarks, the extracted ordering conditions are weaker $\rightarrow$ conditions since actions belonging to different FAM groups may be interleaved. For necessary sub-goals, if $Q(w_1, \ldots, w_n)$ was extracted as a joint precondition for $P(u_1, \ldots, u_k)$, then we know that some ground instance q of $Q(w_1, \ldots, w_n)$ must be true directly before the ground instance p of $P(u_1, \ldots, u_k)$ that will be used in the plan. Therefore, if no such q was made true yet, then we know that p has not been made true yet either and so must still be made true (even if some other ground instance of $P(u_1, \ldots, u_k)$ was already true). In this sense, the $\rightarrow_D$ relations carry over to the lifted setting, and (as discussed in Section 4) we use them as before.

4 Landmark-based Heuristics

The basic idea behind landmark heuristics is simple as outlined in the introduction: $h^{\mathrm{LM}}(s) := |\{p \in L \mid p \text{ must still be fulfilled}\}|$.

The notion of "must still be fulfilled" however is a bit subtle to spell out: 1. this is path-dependent as we need to know which landmarks have already been fulfilled in the past; 2. we must take into account that something may already have been fulfilled but must be made true again. These issues have been amply addressed in the literature on grounded planning. Here we adapt the LM_COUNT heuristic [Richter and Westphal, 2010], which is simple, and canonical in terms of being part of a state-of-the-art system. The heuristic is defined as

$$h^{\mathrm{LM}}(\pi_s) := |(L \setminus Acc(\pi_s)) \cup Req(\pi_s)|$$

where $\pi_s = \langle s_0, \ldots, s_n \rangle$ is the search path (the sequence of states) up to $s = s_n$, and the ingredients are:

- Accepted landmarks: $Acc(\pi_s)$ is the set of all $p \in L$ such that either $p \in Acc(\langle s_0, \ldots, s_{n-1} \rangle)$, or $p \in s$ and for all q that are transitively ordered before p we have $q \in Acc(\langle s_0, \ldots, s_{n-1} \rangle)$.
- Required-again landmarks: $Req(\pi_s)$ is the set of all $p \in L \setminus s$ where either $p \in \mathcal{G} \cup (L \setminus Acc(\pi_s))$ or p is ordered $p \rightarrow_D q$ directly before $q \in Req(\pi_s)$.

We adapt this definition to lifted landmarks $P(u_1, \ldots, u_k)$ simply by replacing state-membership tests "$p \in s$?" with tests checking whether there exists a ground atom p of $P(u_1, \ldots, u_k)$ so that $p \in s$.

Analogously we extend the definitions to lifted disjunctive action landmarks L_i by replacing the state-membership test with a check whether the grounded operator used to reach s is an instantiation of one of the actions in L_i.

Observe that, thanks to the definition of $Req(\pi_s)$, the resulting heuristic function dominates goal counting.

We extend the search of PWL to a LAMA-like approach [Richter and Westphal, 2010], i.e., to a multi-queue heuristic search. It uses two heuristics to guide the search, in our case h^{LM} and the lifted h^{Ladd} [Corrêa *et al.*, 2021]. To exploit the strength of both heuristics, separate open lists are maintained. Whenever a new state is expanded, it is evaluated with *both* heuristics and added to *both* queues. Further, each heuristic has a preferred-operator queue into which states are added if they are reached applying a preferred operator.

Preferred operators are a concept used by certain heuristic functions (e.g. the FF system [Hoffmann and Nebel, 2001]): they return actions that are promising to lead towards a goal state, e.g. because they have been included in the relaxed plan (in case of FF). This concept is also used by the currently best-performing lifted heuristic [Corrêa *et al.*, 2021] and it is a main reason for its performance. For LM_COUNT, we consider an operator to be preferred if a new landmark is fulfilled in the resulting state. When choosing the next state to be expanded, the search decides between the queues based on a numeric priority. All queues initially have a priority of 0, after a state is removed from a queue, its priority is decreased by 1. Whenever a state drawn from a preferred operator queue has a better heuristic estimate than all previous states, the priority of all preferred operator queues is increased by 1000, increasing the use of preferred operators.

5 Experiments

We implemented our heuristics in the Power Lifted (PWL) planner [Corrêa *et al.*, 2020][2]. All configurations use greedy

[2]The code can be found at: https://github.com/minecraft-saar/powerlifted

	max. arity $\mathcal{P}$	max. arity $\mathcal{A}$	Grounded (FD) #ground	Grounded (FD) $\lvert\mathcal{A}^{\mathcal{O}}\rvert$	Grounded (FD) $\lvert\mathcal{P}^{\mathcal{O}}\rvert$	Grounded (FD) h^{GC}	Grounded (FD) h^{FF}	Lifted h^{Ladd}	Lifted $h^{\mathrm{Ladd}_{PO}}$	Lifted h^{UR}	Lifted $h^{\mathrm{UR\text{-}d}}$	Lifted h^{GC}	Lifted h^{LNS}	Lifted $h^{\mathrm{LNS}\rightarrow}$	Lifted $h^{\mathrm{LNS}}_{\mathrm{LAMA}}$	Lifted $h^{\mathrm{LNS}\rightarrow}_{\mathrm{LAMA}}$	Lifted h^{LFAM}	Lifted $h^{\mathrm{LFAM}\rightarrow}$	Lifted $h^{\mathrm{LFAM}}_{\mathrm{LAMA}}$	Lifted $h^{\mathrm{LFAM}\rightarrow}_{\mathrm{LAMA}}$
(a) Large action schema arity																				
ged (156)	2	3	156	32585	1206	**156**	62	92	137	**156**	**156**	**156**	**156**	146	150	125	155	155	142	141
ged-split (156)	2	2	156	4602	734	**156**	35	24	102	**156**	**156**	**156**	154	110	141	82	155	149	133	116
organic-synthesis-alk (18)	2	16	15	24475	74	15	15	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	12	12	13	13
organic-synthesis-mit (18)	2	31	2	2946	36	2	2	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	13	13	12	12
organic-synthesis-org (20)	2	31	1	137784	806	1	1	8	**11**	9	9	10	10	10	10	10	6	6	7	7
pipesworld (50)	3	12	16	119907	232	16	13	20	24	12	11	22	20	20	**32**	30	19	17	29	26
Sum (418)			346			346	128	180	310	369	368	**380**	376	375	369	283	360	352	336	315
(b) Large predicate arity: Visitall and Childsnack																				
3-dim-close-g1 (10)	3	4	7	140832	48386	7	7	**10**	**10**	**10**	**10**	8	8	8	**10**	**10**	8	8	**10**	**10**
3-dim-close-g2 (10)	3	4	7	140832	48388	7	7	8	**9**	**9**	**9**	6	5	5	**9**	**9**	5	5	**9**	**9**
3-dim-close-g3 (10)	3	4	7	140832	48390	7	7	8	**10**	9	9	7	7	7	9	9	5	5	9	9
3-dim-far-g1 (10)	3	4	7	140832	48386	7	7	9	**10**	1	**10**	0	0	0	**10**	**10**	0	0	**10**	**10**
3-dim-far-g2 (10)	3	4	7	140832	48388	7	7	7	**10**	2	7	0	0	0	8	9	0	0	8	9
3-dim-far-g3 (10)	3	4	7	140832	48390	7	7	6	**8**	2	6	0	0	0	**8**	**8**	0	0	**8**	**8**
4-dim-close-g1 (10)	4	5	3	122005	33143	3	3	**10**	**10**	**10**	**10**	6	6	6	**10**	**10**	6	6	**10**	**10**
4-dim-close-g2 (10)	4	5	3	122005	33145	3	3	9	**10**	**10**	**10**	8	8	8	**10**	**10**	8	8	**10**	**10**
4-dim-close-g3 (10)	4	5	3	122005	33147	3	3	7	**9**	7	6	5	5	5	**9**	**9**	5	5	**9**	**9**
4-dim-far-g1 (10)	4	5	3	122005	33143	3	3	4	5	1	**10**	0	0	0	6	6	0	0	6	6
4-dim-far-g2 (10)	4	5	3	122005	33145	3	3	3	5	2	**7**	0	0	0	3	3	0	0	3	3
4-dim-far-g3 (10)	4	5	3	122005	33147	3	3	3	**4**	2	**4**	0	0	0	3	3	0	0	**4**	3
5-dim-close-g1 (10)	5	6	2	175760	40546	2	2	**10**	**10**	**10**	**10**	9	9	9	**10**	**10**	9	9	**10**	**10**
5-dim-close-g2 (10)	5	6	2	175760	40548	2	2	7	**10**	9	8	7	6	6	**10**	**10**	5	5	**10**	**10**
5-dim-close-g3 (10)	5	6	2	175760	40550	2	2	9	**10**	**10**	9	8	8	8	**10**	**10**	8	8	**10**	**10**
5-dim-far-g1 (10)	5	6	2	175760	40546	2	2	3	4	2	**10**	0	0	0	4	4	0	0	4	4
5-dim-far-g2 (10)	5	6	2	175760	40548	2	2	2	4	2	**6**	0	0	0	3	3	0	0	3	3
5-dim-far-g3 (10)	5	6	2	175760	40550	2	2	2	4	2	**6**	0	0	0	3	3	0	0	3	3
Visitall Sum (180)	5	6	72	141947	44039	72	72	117	142	100	**147**	64	62	62	135	136	59	59	136	136
n1-g3 (12)	2	5	12	513	138	**12**	**12**	**12**	**12**	**12**	**12**	**12**	11	11	**12**	**12**	11	11	**12**	**12**
n1-g5 (12)	2	5	12	1930	218	5	**12**	4	**12**	**12**	**12**	3	2	2	**12**	**12**	2	2	**12**	**12**
n1-g7 (12)	2	5	12	5011	298	2	4	2	6	6	8	2	2	2	**12**	**12**	1	1	10	10
n2-g3 (12)	3	6	12	9758	159	5	7	4	**12**	11	**12**	3	3	3	**12**	**12**	2	2	**12**	**12**
n2-g5 (12)	3	6	12	74405	253	2	2	2	3	3	**12**	1	1	1	6	6	0	0	5	5
n2-g7 (12)	3	6	11	232344	332	0	**2**	1	**2**	1	1	0	0	0	**2**	**2**	0	0	**2**	**2**
n3-g3 (12)	4	8	12	65520	273	3	5	3	9	6	11	2	1	1	**12**	**12**	1	1	11	11
n3-g5 (12)	4	8	7	221398	364	0	1	1	2	1	**6**	0	0	0	3	3	0	0	2	2
n3-g7 (12)	4	8	3	259615	363	0	0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0
n4-g3 (12)	5	10	10	176161	819	0	4	2	6	6	**11**	0	0	0	10	10	0	0	9	8
n4-g5 (12)	5	10	1	376905	348	0	0	0	1	0	**2**	0	0	0	0	0	0	0	0	0
n4-g7 (12)	5	10	0	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Childsnack Sum (144)	5	10	104	85659	307	29	49	32	66	63	**87**	23	20	20	81	81	17	17	75	74
(c) Large object universe																				
blocksworld-g2 (10)	2	2	2	100000	50602	1	2	1	2	6	6	1	6	0	2	0	6	**8**	2	2
blocksworld-g3 (10)	2	2	2	100000	50602	0	0	0	1	0	0	0	0	0	1	0	0	0	**2**	**2**
blocksworld-g4 (10)	2	2	2	100000	50602	0	**1**	0	**1**	0	0	0	0	0	**1**	0	0	0	0	**1**
blocksworld-g5 (10)	2	2	2	100000	50602	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	**1**
logistics-g1 (10)	2	4	2	1282377	2252	2	2	1	**10**	0	0	5	5	5	**10**	**10**	5	5	**10**	**10**
logistics-g2 (10)	2	4	2	1284629	3379	2	2	1	**10**	0	0	5	5	5	**10**	**10**	5	5	**10**	**10**
logistics-g3 (10)	2	4	2	1286881	4506	2	2	0	**10**	0	0	4	4	4	**10**	**10**	5	5	**10**	**10**
logistics-g4 (10)	2	4	2	1289133	5633	2	2	0	**10**	0	0	4	4	4	**10**	**10**	5	5	**10**	**10**
rovers-g2 (10)	3	6	2	5316	2531	2	2	**10**	**10**	5	**10**	1	5	5	**10**	**10**	5	5	**10**	**10**
rovers-g4 (10)	3	6	1	4041	2006	1	1	1	**10**	2	3	0	2	2	**10**	**10**	2	2	**10**	**10**
rovers-g6 (10)	3	6	1	4983	2026	1	1	0	5	2	2	0	1	0	**10**	7	1	1	9	9
rovers-g8 (10)	3	6	1	3753	2965	1	1	0	4	1	0	0	1	0	**6**	4	1	1	**6**	**6**
Sum (120)			21			14	16	14	73	16	21	20	32	24	**81**	71	35	37	79	**81**
Total Sum (826)						461	265	343	591	548	624	487	490	481	**666**	571	471	465	626	606

Table 1: Coverage results (best results highlighted in **boldface**). h^{LNS} and $h^{\mathrm{LNS}\rightarrow}$ are the LM heuristics based on necessary subgoals, without vs. with ordering relations; h^{LFAM} and $h^{\mathrm{LFAM}\rightarrow}$ are based on FAM-Cuts, without vs. with ordering relations; Heuristics annotated with $LAMA$ use a LAMA style search. $\lvert\mathcal{A}^{\mathcal{O}}\rvert$and $\lvert\mathcal{P}^{\mathcal{O}}\rvert$show average grounding size for those instances that can be grounded (#ground).

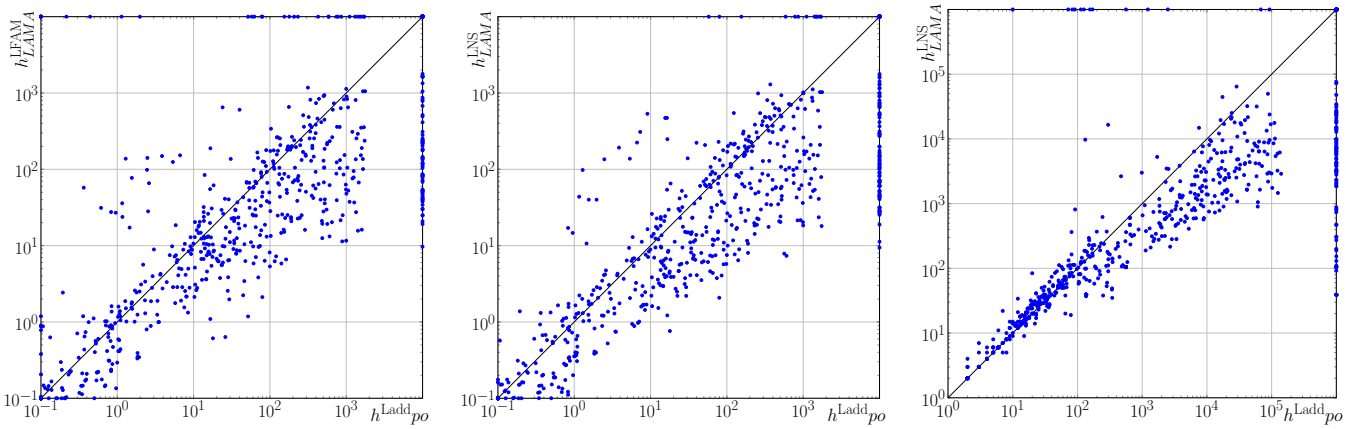

Figure 2: Scatter plots comparing two algorithms. The respective algorithms are given on the axis. The left and middle figures show the runtime and the right figure shows expansions.

best first search (GBFS). We compare against goal counting (h^{GC}), the lifted add heuristic with ($h^{\text{Ladd}}po$) and without (h^{Ladd}) preferred operators [Corrêa *et al.*, 2021] and the best-performing configurations of Lauer *et al.* [2021], goal count using unary relaxation heuristic as tiebreaker, with ($h^{\text{GC,ur-d}}$) and without ($h^{\text{GC,ur}}$) disambiguation of static predicates. h^{Ladd} computes the add heuristic based on the lifted model and might need runtime exponential in the size of the lifted input model. $h^{\text{GC,ur}}$ splits the original predicates from the domain in 1-ary predicates before heuristic calculation, resulting in a computation in polynomial time. Like ours, these heuristics are implemented in PWL.

For our system we included 8 configurations: We combine necessary subgoal landmarks with GBFS to the configurations $h^{\text{LNS}\rightarrow}$ and h^{LNS}; where the former uses landmark ordering information and the latter does not. The same is done for the FAM landmarks, denoted $h^{\text{LFAM}\rightarrow}$ and h^{LFAM}. Then, we also use the LAMA-style search instead of GBFS, resulting in the configurations $h^{\text{LNS}}_{\text{LAMA}}$, $h^{\text{LNS}\rightarrow}_{\text{LAMA}}$, $h^{\text{LFAM}}_{\text{LAMA}}$, and $h^{\text{LFAM}\rightarrow}_{\text{LAMA}}$.

We further report results for the grounded Fast Downward (FD) system [Helmert, 2006] with the goal counting and h^{FF} [Hoffmann and Nebel, 2001] heuristics. The experiments were run on a cluster of machines with Intel Xeon E5-2650 CPUs with a clock speed of 2.30GHz.Timeout and memory limits were set to 30 min and 4GB respectively for all runs.

We use the benchmark set used by Lauer *et al.* [2021] that was introduced to evaluate lifted planners. It explores different reasons for being hard to ground: large action-schema arity, large predicate arity, and large object universe.

Table 1 gives the coverage results. Of our new lifted landmark heuristics, configurations using the FAM landmarks perform slightly worse than necessary subgoals. The LAMA-like search ($h^{\text{LNS}}_{\text{LAMA}}$ and $h^{\text{LNS}\rightarrow}_{\text{LAMA}}$) drastically improves performance compared to simple goal counting, especially for the necessary subgoal landmarks. The resulting systems reach the highest overall coverage in our experiments.

The ordering constraints of landmarks do not improve coverage, we attribute this to the overhead generated by their tracking and the fact that the orderings that are produced as a byproduct of the landmark extraction methods are only very basic. When comparing against $h^{\text{Ladd}}po$, we can see that the landmarks heuristics on their own fall behind. But the LAMA-style search outperforms both the $h^{\text{Ladd}}po$ and the other landmark heuristics. The performance gain can especially be seen in the domains Childsnack and Rovers.

The unary relaxation heuristics h^{UR} and $h^{\text{UR-d}}$ perform best on problems with high predicate arity, but worse on the others, especially those with large object universe.

The scatter plots presented in Figure 2 (left and middle) compare the runtime of the LAMA search to $h^{\text{Ladd}}po$. It can be seen that the usage of both kinds of landmarks systematically decreases the runtime. The same holds when comparing expanded search nodes (Fig. 2 (right) compares $h^{\text{LNS}}_{\text{LAMA}}$ and $h^{\text{Ladd}}po$).

Compared to the runtime of the planner, the time spend for landmark generation is small. For the necessary subgoal landmarks, the mean generation time per domain is below 100ms (with a variance of less than 200ms) for all domains except for those with large object universes. Among these, generation in the Rovers domain takes the longest with a mean generation time of 550ms and a variance of 40ms. Among the FAM-Cut Landmarks the mean generation time is generally about 200ms with a variance of up to 40ms, with Rovers again being the exception. Here the mean generation time is 2.7 seconds with a variance of 3.5 seconds, with the longest generation taking 7.5 seconds.

6 Conclusion

Lifted heuristic search planning has been neglected but is currently taking up speed. Landmarks are a natural candidate for the design of heuristics in this setting, and our results clearly show their promise. We presented two methods to extract landmarks from the lifted planning model. While their usage in simple landmark count heuristics performs worse than the currently best-performing lifted heuristics, our evaluation shows that they are especially valuable when combined with the lifted add heuristics to a system similar to the grounded LAMA planning system. The resulting system outperforms all other configurations in our evaluation.

Acknowledgements

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – Projektnummer 232722074 – SFB 1102 / Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 232722074 – SFB 1102

References

[Bonet and Geffner, 2001] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33, 2001.

[Corrêa *et al.*, 2020] Augusto B. Corrêa, Florian Pommerening, Malte Helmert, and Guillem Francès. Lifted successor generation using query optimization techniques. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, pages 80–89. AAAI Press, 2020.

[Corrêa *et al.*, 2021] Augusto B. Corrêa, Florian Pommerening, Malte Helmert, and Guillem Francès. Delete-relaxation heuristics for lifted classical planning. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS'21)*, pages 94–102. AAAI Press, 2021.

[Fišer, 2020] Daniel Fišer. Lifted fact-alternating mutex groups and pruned grounding of classical planning problems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pages 9835–9842. AAAI Press, 2020.

[Haslum, 2011] Patrik Haslum. Computing genome edit distances using domain-independent planning. In *Proceedings of the SPARK Workshop*, 2011.

[Helmert and Domshlak, 2009] Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 162–169. AAAI Press, 2009.

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

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Hoffmann *et al.*, 2004] Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.

[Hoffmann *et al.*, 2006] Jörg Hoffmann, Stefan Edelkamp, Sylvie Thíebaux, Roman Englert, Frederico Liporace, and Sebastian Trüg. Engineering benchmarks for planning: the domains used in the deterministic part of IPC-4. *Journal of Artificial Intelligence Research*, 26:453–541, 2006.

[Karpas and Domshlak, 2009] Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 1728–1733, 2009.

[Koller and Hoffmann, 2010] Alexander Koller and Jörg Hoffmann. Waking up a sleeping rabbit: On natural-language sentence generation with FF. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 238–241. AAAI Press, 2010.

[Koller and Petrick, 2011] Alexander Koller and Ronald Petrick. Experiences with planning for natural language generation. *Computational Intelligence*, 27(1):23–40, 2011.

[Lauer *et al.*, 2021] Pascal Lauer, Álvaro Torralba, Daniel Fiser, Daniel Höller, Julia Wichlacz, and Jörg Hoffmann. Polynomial-time in PDDL input size: Making the delete relaxation feasible for lifted planning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*, pages 4119–4126. IJCAI org., 2021.

[Matloob and Soutchanski, 2016] Rami Matloob and Mikhail Soutchanski. Exploring organic synthesis with state-of-the-art planning techniques. In *Proceedings of the SPARK Workshop*, pages 52–61, 2016.

[Penberthy and Weld, 1992] J. Scott Penberthy and Daniel S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference (KR'92)*, pages 103–114. Morgan Kaufmann, 1992.

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

[Richter *et al.*, 2008] Silvia Richter, Malte Helmert, and Matthias Westphal. Landmarks revisited. In *Proceedings of the 23rd National Conference of the American Association for Artificial Intelligence (AAAI'08)*, pages 975–982. AAAI Press, 2008.

[Ridder and Fox, 2014] Bram Ridder and Maria Fox. Heuristic evaluation based on lifted relaxed planning graphs. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, pages 244–252. AAAI Press, 2014.

[Russell and Norvig, 1995] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[Seipp, 2019] Jendrik Seipp. Pattern selection for optimal classical planning with saturated cost partitioning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19*, pages 5621–5627, 2019.

[Younes and Simmons, 2003] Håkan L. S. Younes and Reid G. Simmons. VHPOP: versatile heuristic partial order planner. *Journal of Artificial Intelligence Research*, 20:405–430, 2003.