When Abstractions Met Landmarks

Carmel Domshlak and **Michael Katz** and **Sagi Lefler***
Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology
Haifa, Israel

Abstract

Abstractions and landmarks are two powerful mechanisms for devising admissible heuristics for classical planning. Here we aim at putting them together by integrating landmark information into abstractions, and propose a concrete realization of this direction suitable for structural-pattern abstractions, as well as for other abstraction heuristics. Our empirical evaluation shows that landmark information can substantially improve the quality of abstraction heuristic estimates.

Introduction

Heuristic state-space search is a common and successful approach to classical planning, and in particular, to cost-optimal classical planning. Apart from the choice of the search algorithm, heuristic-search solvers for cost-optimal planning differ mainly in their admissible heuristic estimators. Recent years have seen a growing body of work on expanding the palette of heuristic estimators, with most (if not all) current admissible heuristics being based on one of the following three ideas:

1. *critical paths*: the h^m heuristic family (Haslum and Geffner 2000), with the $h^1 \equiv h^{\max}$ member being closely related to the *delete relaxation* idea,
2. *abstractions*: pattern databases (Edelkamp 2001), merge-and-shrink abstractions (Helmert, Haslum, and Hoffmann 2007), and structural patterns (Katz and Domshlak 2008b),
3. *landmarks*: the admissible landmark heuristics h^{L} and h^{LA} (Karpas and Domshlak 2009), and $h^{\text{LM-cut}}$ (Helmert and Domshlak 2009), with all three being also closely related to delete relaxation.

*The work of the authors was partly supported by Israel Science Foundation grant 670/07.

Until very recently, these three ideas have been developed in relative isolation, and thus there has been no cross-fertilization between them. In a recent work aiming at connecting between the different approaches, Helmert and Domshlak (2009) in particular show that additive $h^{\max}$ and admissible landmark heuristics are in fact very much related. This realization allowed the authors to develop a novel admissible landmark heuristic, $h^{\text{LM-cut}}$, that has dramatically changed the state of the art in performance for cost-optimal planning.

In this work we consider another edge of the above triangle of ideas, namely abstractions and landmarks, and try to exploit the best of both worlds by fertilizing the former with the latter. In general, abstraction heuristics have been shown by Helmert and Domshlak (2009) to be more expressive (in a proper sense of this notion) than landmark heuristics. However, all the currently used mechanisms for devising abstraction heuristics appear to be quite dependent on the richness of the goal description that comes with the problem specification. Informally, the fewer the sub-goals explicitly mentioned by the problem, the less guided (and thus potentially less effective) are the procedures for selecting concrete sets of abstractions. Our empirical evaluation of this issue described at the beginning of the paper clearly exemplifies this dependence.

In this work we tackle precisely this Achilles heel of automatically devised abstractions, and show how problem's *landmarks* can be used to substantially cure it.

- We show how landmarks, constituting *implicit sub-goals* of the problem, can be exploited in enhancing abstraction heuristics by *compiling the landmarks into the problem specification*. The proposed problem compilation is extremely simple, yet it preserves all the essential reachability properties of the original problem, and results in boosting substantially the quality of the

induced heuristic estimates. Focusing on fork-decomposition structural patterns (Katz and Domshlak 2008b) we also show that some investment in action-cost partitioning improves the informativeness of the landmark-enhanced abstractions even further.

- We both investigate the straightforward approach of directly solving the landmark enhanced problem, as well as propose a novel approach of *searching for plans in the state space of the original problem while estimating the search nodes via their mappings to the landmark enhanced problem*. The latter demands maintaining information about the achievement of the landmarks during the search, and thus the *LM-A** algorithm (Karpas and Domshlak 2009) is used. Our empirical evaluation clearly testifies for the higher effectiveness of the latter approach.

Preliminaries

We consider classical planning tasks corresponding to state models with single initial state and only deterministic actions; here we consider state models captured by the SAS$^+$ formalism (Bäckström and Nebel 1995) with non-negative action costs. Such a planning task is given by a quintuple $\Pi = \langle V, A, I, G, cost\rangle$, where:

- V is a set of *state variables*, each $v \in V$ is associated with a finite domain $dom(v)$; each complete assignment to V is called a *state*, and $S = dom(V)$ is the *state space* of Π. I is an *initial state*. The *goal* G is a partial assignment to V; a state s is a *goal state* iff $G \subseteq s$.
- A is a finite set of *actions*. Each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V called *preconditions* and *effects*, respectively. $cost : A \to \mathbb{R}^{0+}$ is a real-valued, non-negative *action cost* function.

The value of a variable v in a partial assignment p is denoted by $p[v]$. By $V(p) \subseteq V$ we denote the set of variables instantiated by p. An action a is applicable in a state s iff $s[v] = \mathsf{pre}(a)[v]$ for all $v \in V(\mathsf{pre}(a))$. Applying a changes the value of each $v \in V(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$. The resulting state is denoted by $s[\![a]\!]$; by $s[\![\langle a_1, \ldots, a_k\rangle]\!]$ we denote the state obtained from sequential application of the (respectively applicable) actions $a_1, \ldots, a_k$ starting at state s. Such an action sequence is an *s-plan* if $G \subseteq s[\![\langle a_1, \ldots, a_k\rangle]\!]$, and it is a *cost-optimal* (or, in what follows, *optimal*) s-plan if the sum of its action costs is minimal among all s-plans. The purpose of (optimal) planning is finding an (optimal) I-plan.

For a pair of states $s_1, s_2 \in S$, by $cost(s_1, s_2)$ we refer to the cost of a cheapest action sequence taking us from s_1 to s_2 in the state model of Π; $h^*(s) = \min_{s' \supseteq G} cost(s, s')$ is the custom notation for the cost of optimal s-plans for Π. Finally, we refer later on to the causal graphs induced by the planning tasks. The causal graph of a task $\Pi = \langle V, A, I, G, cost\rangle$ is a digraph over the nodes V. An arc (v, v') belongs to the causal graph iff $v \neq v'$ and there exists an action $a \in A$ such that $v' \in V(\mathsf{eff}(a))$ and $v \in V(\mathsf{eff}(a)) \cup V(\mathsf{pre}(a))$.

Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task, $F = \bigcup_{v \in V} dom(v)$ be the set of *facts* (assuming name uniqueness), ϕ be a propositional logic formula over facts F, $\pi = \langle a_1, \ldots, a_k\rangle$ be an action sequence applicable in I, and $0 \leq i \leq k$. Following the terminology of Hoffmann *et al.* 2004, we say that ϕ is *true at time i* in π iff $I[\![\langle a_1, \ldots, a_i\rangle]\!] \models \phi$, and ϕ is a *landmark* of Π iff in each I-plan for Π, it is true at some time.

While landmarks can be any formulas over facts, we restrict our attention to disjunctions of facts, and use notation $\phi \subseteq F$ to denote "disjunction over the fact subset ϕ of F". This restriction covers all the landmark discovery procedures suggested in the literature. Due to hardness of deciding even that a single fact is a landmark (Porteous, Sebastia, and Hoffmann 2001), practical methods for finding landmarks are either incomplete or unsound. In what follows we assume access to a sound such procedure; in particular, in our empirical evaluation reported here we use LAMA's sound landmark discovery procedure by Richter *et al.* (2008). In general, however, the actual way of discovering landmarks is tangential to our work.

Landmarks are exploited these days in both satisficing and optimal planning as heuristic search, either for devising an incremental, landmark-by-landmark search strategy (Hoffmann, Porteous, and Sebastia 2004) or for deriving heuristic estimates (Richter, Helmert, and Westphal 2008; Karpas and Domshlak 2009; Helmert and Domshlak 2009). In parallel, other sources of information for heuristic guidance have been proven extremely valuable, and this in particular so with various problem abstractions.

An *abstraction* heuristic is based on mapping Π's transition system over states S to an *abstract transition system* over states S^α. The mapping is defined by an abstraction function $\alpha : S \to S^\alpha$ that guarantees $cost_\alpha(\alpha(s), \alpha(s')) \leq cost(s, s')$ for all states $s, s' \in S$. The abstraction heuristic $h^\alpha(s)$ is then the distance from $\alpha(s)$ to the closest abstract goal state. Abstraction heuristics are always admissible by their very definition. Two families of abstractions are used these days for deriving admissible heuristics: abstractions such as in pattern databases (Edelkamp 2001; Haslum et al. 2007) and merge-and-shrink (Helmert, Haslum,

domain ($\mathcal{D}$)	$S(\mathcal{D})$	MS-10^4					$h^{\mathcal{F}}$				
		Π^1	Π	SB	E(Π^1)	E(Π)	Π^1	Π	SB	E(Π^1)	E(Π)
airport-ipc4	20	16	17	16	14.60	15.96	17	20	17	10.68	17.00
blocks-ipc2	21	18	18	18	7.29	18.00	18	21	18	3.83	18.00
depots-ipc3	7	5	7	5	3.22	4.88	4	7	4	0.88	4.00
driverlog-ipc3	12	11	12	11	2.59	11.00	10	12	10	1.84	10.00
freecell-ipc3	5	5	4	4	2.47	3.93	5	5	5	2.53	5.00
grid-ipc1	2	2	2	2	1.53	2.00	1	2	1	0.09	1.00
gripper-ipc1	7	7	7	7	7.00	6.45	7	7	7	6.91	7.00
logistics-ipc1	6	4	4	4	3.43	3.03	2	6	2	0.01	2.00
logistics-ipc2	22	16	16	16	6.18	16.00	10	22	10	0.07	10.00
miconic-strips-ipc2	55	50	55	50	29.78	50.00	50	51	50	30.51	50.00
mprime-ipc1	23	20	21	20	19.12	19.79	20	23	20	1.56	20.00
mystery-ipc1	21	17	17	17	16.57	16.22	18	21	18	4.84	18.00
openstacks-ipc5	7	7	7	7	6.91	7.00	7	7	7	4.24	7.00
pathways-ipc5	4	4	4	4	3.13	3.42	4	4	4	4.00	4.00
pipesworld-notankage-ipc4	21	17	21	17	9.93	14.32	13	16	13	7.77	13.00
pipesworld-tankage-ipc4	14	11	13	11	4.91	10.11	8	10	8	5.15	8.00
psr-small-ipc4	50	50	50	50	37.92	50.00	49	49	49	41.17	49.00
rovers-ipc5	6	6	6	6	5.59	5.79	6	6	6	2.46	6.00
satellite-ipc4	6	6	6	6	5.37	5.04	6	6	6	2.93	6.00
schedule-strips	46	15	19	14	14.00	14.00	46	46	46	34.95	39.03
tpp-ipc5	6	6	6	6	5.99	6.00	6	6	6	2.33	6.00
trucks-ipc5	6	6	5	5	5.00	2.01	6	6	6	2.95	6.00
zenotravel-ipc3	11	9	11	9	7.05	7.86	8	11	8	2.06	8.00
	378	308	328	305	219.57	292.81	321	364	321	173.74	314.03

Table 1: Evaluation of the impact of goal reformulation on merge-and-shrink and fork-decomposition heuristics. Per heuristic, the first three columns capture the number of solved tasks under original formulation (Π), single-goal reformulation (Π^1), and under *both* formulations (SB). The last row in these columns captures the total number of solved planning tasks. The last two columns per heuristic depict the measure of success in terms of expanded nodes, with each entry being the sum of our measure over all the tasks in the domain solved under both formulations. The last row in those columns provides the overall measures.

and Hoffmann 2007) are represented explicitly by their induced transition systems, while structural patterns (Katz and Domshlak 2008b; 2009) correspond to implicitly represented abstractions.

Abstractions and Goal Sensitivity

One key feature of abstraction heuristics is that typically there is a great degree of flexibility in abstraction selection. This flexibility is a mixed blessing because the choice of abstraction may dramatically affect the quality of the heuristic estimate, while homing in on a better/best choice is not easy. A closer look at some successful approaches to both (explicit) pattern database abstractions and (implicit) structural pattern abstractions reveals some commonality in their strategies to resolve that choice dilemma. When a set of pattern databases is selected, the farther state variables are from the goal-mentioned variables $V(G)$ in the causal graph of the problem, the more they are likely to be abstracted away (ignored) altogether. The picture with the fork-decomposition structural patterns is very much similar. While there is not much room for flexibility in selecting such structural patterns, the size of the patterns' set, and thus the quality of the resulting heuristic, depend crucially on the number of goal-mentioned variables. The situation with the merge-and-shrink abstractions is a bit more complicated to frame, but the concrete merge-and-shrink procedure suggested and evaluated by Helmert *et al.* (2007) also puts more focus on the goal-mentioned variables and their close ancestors in the causal graph.

This dependence of some abstraction heuristics on the size of $V(G)$ is quite problematic as any SAS$^+$ planning task can easily be reformulated to contain just a single goal-mentioned variable. To evaluate this dependence empirically, we have conducted a targeted evaluation on a wide sample of planning domains from the International Planning Competitions (IPC) 1998-2006. We have focused on two abstraction heuristics, namely merge-and-shrink (Helmert, Haslum, and Hoffmann 2007) with 10^4 abstract states for explicit abstractions and the structural-pattern database version of the $h^{\mathcal{F}}$ heuristic (Katz and Domshlak 2009) for implicit abstractions. Both these heuristics have been implemented within a standard heuristic forward search framework of the Fast Downward planner (Helmert 2006), and A^* algorithm with full duplicate elimination was used. All the experiments were run on a 3GHz Intel E8400 CPU; the time and memory limits were set to 30 minutes and 1.5 GB, respectively.

In terms of expanded nodes, Table 1 shows that the accuracy of the abstraction heuristics on the original tasks is substantially higher than this on the single-goal reformulations. The specific measure for comparison is as follows. For each of the two problem formulations, each task contributes a value equal the minimal number of expanded nodes among the two formulations divided by the number

of expanded nodes under the respective formulation. If the denominator is 0 (if, e.g., the initial state is a goal state, or the heuristic estimate of the initial state is ∞), then this value is defined to be 1. As we are interested in comparing expanded nodes, we account only for tasks solved under both formulations, and thus the nominator is always well-defined. Each task grants the winning formulation the value of 1, and the other formulation a value in $[0, 1]$. For example, if A^* on Π^1 opens 1000 nodes and on Π it opens 3000 nodes, then Π contributes 1 to the measure $\mathrm{E}(\Pi^1)$ and 1/3 to $\mathrm{E}(\Pi)$. The last row in the table sums up these values over all tasks solved under both problem formulations. Note that, while both merge-and-shrink and fork-decomposition heuristics got hurt by "hiding" the goals of the tasks, the degradation in accuracy of the fork-decomposition heuristic was much higher due to its explicit reliance on the richness of goal specification in the task.

Bringing Landmarks into Abstractions

We now proceed with, first, arguing that landmarks have a natural potential to enhance abstraction heuristics by targeting one of the major sources of their vulnerability. We then describe a simple technique for enhancing a planning task with landmark information, and evaluate and discuss two ways of exploiting this enhancement in abstraction based heuristic search optimal planning. Here as well, our empirical evaluation of the framework is focused on two representatives of abstraction heuristics: merge-and-shrinks (Helmert, Haslum, and Hoffmann 2007) for explicit abstractions and fork-decomposition (Katz and Domshlak 2009) for implicit abstractions.

Revisiting our experiment with "hiding" the goals of the planning tasks, it is clear that things that were goals before the reformulation do not really cease to be goals, but only become *implicit goals*. In fact, this is just one type of possible implicit goals; in particular, *any landmark is such an implicit goal by its very definition.* Given that many such *de facto* goals are not explicitly given in the description of the planning task, it is only natural to explore the possibility of converting some (discoverable) implicit goals to explicit goals. Probably the most direct way to achieve that is via a specific notion of one-sided equivalence between planning tasks which we call *surrogate*. For a planning task Π, let Φ_Π denote the set of all optimal plans for Π. Given two planning tasks Π and Π', we say that Π' is a *surrogate* of Π if

(i) $\Phi_{\Pi'} = \emptyset$ iff $\Phi_\Pi = \emptyset$,

(ii) there exists a mapping $f : \Phi_{\Pi'} \to \Phi_\Pi$ such that, for any $\rho' \in \Phi_{\Pi'}$, $f(\rho')$ can be computed in time polynomial in $||\Pi||$, $||\rho'||$, and $||f(\rho')||$.

Note that, if Π' is a surrogate of Π, then instead of optimally solving Π, one can optimally solve Π', and then reconstruct an optimal plan for Π from the obtained plan for Π'. This is precisely what we suggest to exploit using what we call *direct landmark-based surrogates*. Given a planning task $\Pi = \langle V, A, I, G, cost\rangle$, and a set of initially unachieved (that is, not true in I) disjunctive landmarks $L \subseteq 2^F$ of Π, the direct landmark-based surrogate $\Pi^L = \langle V^L, A^L, I^L, G^L, cost^L\rangle$ of Π is constructively defined as follows.[1]

- For each landmark $\phi \in L$, we introduce a new variable v_ϕ with $dom(v_\phi) = \{0, 1\}$, and set $V^L = V \cup \{v_\phi \mid \phi \in L\}$.
- The initial state and goals are set to $I^L = I \cup \{v_\phi{=}0 \mid \phi \in L\}$, and $G^L = G \cup \{v_\phi{=}1 \mid \phi \in L\}$.
- For each action $a \in A$, we introduce an action

  $a^L = \langle \mathsf{pre}(a),\ \mathsf{eff}(a) \cup \{v_\phi{=}1 \mid \mathsf{eff}(a)[v] \in \phi \in L\}\rangle,$

  that is, for each landmark $\phi \in L$ that a can achieve, a^L will assign the corresponding auxiliary variable v_ϕ to its goal value. Given that, we set $A^L = \{a^L \mid a \in A\}$.

Note that, for any plan ρ for Π, there is a plan ρ' for Π^L of the same cost, obtained by replacing every action a along ρ with a^L. Thus, there is bijective, cost-preserving correspondence between the plans of Π and Π^L.

Proposition 1 *Given a planning task Π and a set of initially unachieved disjunctive landmarks L of Π, the direct landmark-based surrogate Π^L of Π is a surrogate of the latter, and can be constructed from Π and L in polynomial time.*

To illustrate the process of creating a direct landmark-based surrogate, as well as the potential of using direct landmark-based surrogates with abstraction heuristics, consider a planning task Π with three binary variables $V = \{x, y, z\}$, action set A of three actions

$$a_1 = \langle\{x = 0\}, \{x = 1\}\rangle,$$
$$a_2 = \langle\{x = 1, y = 0\}, \{y = 1\}\rangle,$$
$$a_3 = \langle\{y = 1, z = 0\}, \{z = 1\}\rangle$$

initial state $I = \{x{=}0, y{=}0, z{=}0\}$, and the goal $G = \{z{=}1\}$. All the actions have the same cost 1.

[1] Landmarks corresponding to the explicit goals of the task can be safely ignored in the construction of Π^L, compiling in only landmarks $\phi \in L$ such that $I \not\models \phi$ and $G \not\models \phi$.

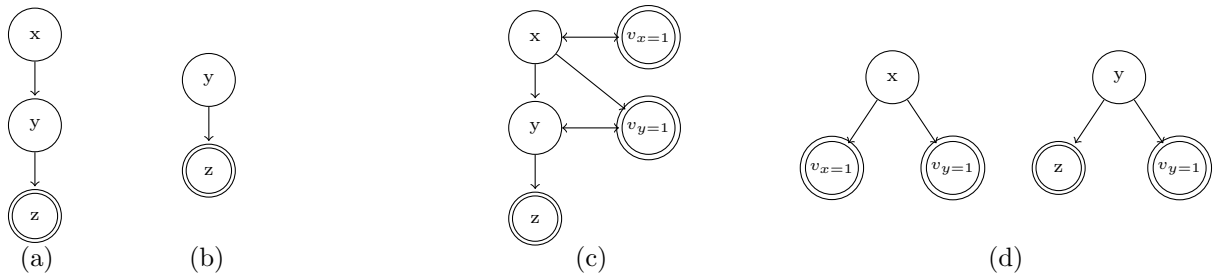

Figure 1: Causal graphs (a,c) and fork decomposition (b,d) for the example task Π and its direct landmark-based surrogate Π^L, respectively. Nodes with double-line contour correspond to the goal-mentioned variables $V(G)$ and $V(G^L)$.

Figure 1(a) depicts the causal graph of this planning task. While the cost of the optimal plan for this task is 3, we have $h^{\mathcal{F}}(I) = 2$. The reason for that is shown in Figure 1(b). Structural patterns will not account for the cost of achieving $x = 1$, despite the fact that it has to hold on any plan for this task. Note that there are two initially unachieved landmarks in this task that go beyond G, namely $x{=}1$ and $y{=}1$. Thus, the direct landmark-based surrogate Π^L of Π will consist of the variables $V^L = \{x, y, z, v_{x=1}, v_{y=1}\}$, action set A^L

$$a_1 = \langle\{x = 0\}, \{x = 1, v_{x=1} = 1\}\rangle,$$
$$a_2 = \langle\{x = 1, y = 0\}, \{y = 1, v_{y=1} = 1\}\rangle,$$
$$a_3 = \langle\{y = 1, z = 0\}, \{z = 1\}\rangle$$

initial state $I^L{=}\{x{=}0, y{=}0, z{=}0, v_{x=1}{=}0, v_{y=1}{=}0\}$, and goal $G^L = \{z = 1, v_{x=1} = 1, v_{y=1} = 1\}$. The causal graph of Π^L is shown in Figure 1(c), and Figure 1(d) shows the fork-decomposition of Π^L. The left pattern will have the action representatives $a_1^1 = \langle\{x = 0\}, \{x = 1\}\rangle$, $a_1^2 = \langle\{x = 1\}, \{v_{x=1} = 1\}\rangle$, and $a_2^1 = \langle\{x = 1\}, \{v_{y=1} = 1\}\rangle$. The right pattern will have the action representatives $a_2^2 = \langle\{y = 0\}, \{y = 1\}\rangle$, $a_2^3 = \langle\{y = 1\}, \{v_{y=1} = 1\}\rangle$, and $a_3^1 = \langle\{y = 1, z = 0\}, \{z = 1\}\rangle$. Uniform action cost partitioning will assign 1 to a_3^1, 1/2 to a_1^1 and a_1^2, and 1/3 to a_2^1, a_2^2, and a_2^3. The optimal plan costs for the left and right patterns are 4/3 and 5/3, respectively, resulting in $h^{\mathcal{F}}(I^L) = h^*(I^L) = 3$.

In general, while the resemblance between the respective components of Π and Π^L is high, fork decomposition of Π^L both enriches the patterns already present in the fork decomposition of Π (by, e.g., adding more children to forks' roots), and induces patterns that would not be present in the fork decomposition of Π at all. In turn, the richer is the fork-decomposition in terms of the number of patterns and the comprehensiveness of each pattern, the higher the estimate we can possibly obtain from that decomposition. However, "possibly obtain" and "obtain" are not necessarily the same thing, and a lot depends on the specific action cost partitioning between the patterns of the additive ensemble comprising $h^{\mathcal{F}}$ (Katz and Domshlak 2008a). The choice of action cost partitioning can vary from optimal (in terms of maximizing the estimate) to almost arbitrarily bad. The good news is that, under *optimal action cost partitioning* (achievable in polynomial time; see Katz and Domshlak 2008a), the dominance relation $h^{\mathcal{F}}(I^L) \geq h^{\mathcal{F}}(I)$ always holds. In fact, a stronger claim holds.

Proposition 2 *Given a planning task* $\Pi = \langle V, A, I, G, cost\rangle$*, and a direct landmark-based surrogate* Π^L *of* Π*, for any state* s *of* Π *and any state* s' *of* Π^L *such that* $s'[V] = s$*, under the optimal action cost partitioning we have* $h^{\mathcal{F}}(s') \geq h^{\mathcal{F}}(s)$*.*

While the statement of Proposition 2 is very positive, in practice the picture is more complicated. The procedure of Katz and Domshlak (2008a) for devising an optimal action cost partition is polynomial-time, yet it is based on solving large linear programs, and thus takes too much time to be computed in practice at every search node. As the first sanity check for the practical usefulness of switching from Π to Π^L, we compared the initial-state estimates of a sub-optimal (yet cheep to compute) fork-decomposition heuristic for planning tasks from a wide sample of IPC domains, as well as for their respective direct landmark surrogates. The landmarks were discovered using LAMA's landmark discovery procedure (Richter, Helmert, and Westphal 2008). The construction of the surrogate task Π^L from the original task Π was done as a part of the preprocessing.

Table 2 summarizes the implication of switching to the surrogate problems in terms of the quality of the initial state estimation with the above setting of $h^{\mathcal{F}}$. These results appear to be very promising. Except for the Blocksworld domain where estimates on Π^L on average got slightly worse, in all other domains the estimates improved (or re-

domain	$\frac{h^{\mathcal{F}}(I)}{c^*}$	$\frac{h^{\mathcal{F}}(I^L)}{c^*}$
airport-ipc4 (20)	0.627	0.969
blocks-ipc2 (30)	0.477	0.432
depots-ipc3 (7)	0.419	0.508
driverlog-ipc3 (14)	0.632	0.708
freecell-ipc3 (7)	0.295	0.636
grid-ipc1 (2)	0.258	0.571
gripper-ipc1 (11)	0.375	0.546
logistics-ipc1 (6)	0.854	0.931
logistics-ipc2 (22)	0.998	0.994
miconic-strips-ipc2 (140)	0.434	0.964
mprime-ipc1 (25)	0.489	0.625
mystery-ipc1 (22)	0.621	0.713
openstacks-ipc5 (7)	0.611	0.829
pathways-ipc5 (5)	0.187	0.187
pipesworld-notankage-ipc4 (22)	0.302	0.359
pipesworld-tankage-ipc4 (14)	0.218	0.293
psr-small-ipc4 (50)	0.169	0.178
rovers-ipc5 (7)	0.529	0.653
satellite-ipc4 (9)	0.541	0.844
tpp-ipc5 (6)	0.838	0.836
trucks-ipc5 (9)	0.385	0.608
zenotravel-ipc3 (12)	0.694	0.809

Table 2: Average ratios between the initial state estimates of $h^{\mathcal{F}}$ on the original and landmarks-enhanced problems, and the optimal solution length $c^* = h^*(I) = h^*(I^L)$. The number of tasks taken into consideration per domain appears in parentheses in the first column.

mained unchanged), with the most substantial average improvement of $\approx 120\%$ in Freecell, Grid, and Miconic. Given that LAMA's landmark discovery procedure typically takes very low time, these results suggest that the basic idea of incorporating landmark information into the process of problem abstraction is valuable. Next, however, we show that there is still some work to be done to make the basic idea trully effective.

Exploiting Π^L

The left half of Table 3 (A^*/h on Π^L vs. Π) describes the effeciveness of the straightforward approach of solving the surrogate task Π^L comparatively to solving the original task Π.

The first observation is that for both abstractions, the total number of solved planning tasks *decreases* when we switch from Π to Π^L. There are only four domains in which this number increased, namely BLOCKSWORLD with MS-10^4 heuristic and MICONIC, SATELLITE, and TRUCKS with $h^{\mathcal{F}}$ heuristic. On the rest of the domains, with both heuristics the number of solved tasks either decreases or remains unchanged. A deeper look into the expanded nodes information revealed that in the case of merge-and-shrink abstraction, the number of expanded nodes decreased substantially on three domains only, namely BLOCKSWORLD, FREECELL, and MPRIME, remaining almost unchanged or increasing substantially on the rest of the domains. In the case of structural-patterns the picture is different. Most of the domains are divided almost evenly to sets of substantially increased and substantially decreased numbers of expanded nodes, with only small number of domains having the number of expanded nodes almost unchanged. On BLOCKSWORLD, DEPOTS, DRIVERLOG, GRIPPER, LOGISTICS-00, PIPESWORLD-NOTANKAGE, and PIPESWORLD-TANKAGE the number of expanded nodes substantially increased, causing less tasks to be solved on almost all of these domains. On the other hand, on MICONIC, SATELLITE, TRUCKS, AIRPORT, FREECELL, GRID, LOGISTICS-98, MPRIME, OPENSTACKS, and ROVERS the number of expanded nodes substantially decreased, on each of the first three causing one more task to be solved. Interestingly, on AIRPORT, despite the decrease in expanded nodes on average, less tasks are solved overall.

The explanation for this "inconsistency" between Tables 2 and 3 is actually very simple. In general, the state space of the surrogate task Π^L is $2^{|L|}$ times larger the state space of the original Π. Thus the number of reachable states in Π^L can be up to $2^{|L|}$ times larger than in Π because every state of Π is now considered in different contexts of achievement of different subsets of landmarks. Hence, even if the heuristics functions on Π^L are getting more accurate, the effort required from A^* to "prove" the optimality of the optimal solution increases.

Exploiting Π^L Indirectly

To eliminate this pitfall to a large extent we have evaluated a scheme in which the heuristic estimates come from the landmark enhanced tasks while the (forward state-space) search is performed on the original task using the recent *LM-A** search algorithm of Karpas and Domshlak (2009). In general, this scheme works as follows.

1. Starting with the initial state I, each evaluated state s of Π is first associated with a subset of landmarks $L_s \subseteq L$ that must be achieved on the way to the goal *from* s. Determining the landmark set L_s is done using the techniques developed by Richter et al. (2008) and Karpas and Domshlak (2009).
2. Given L_s, the state s is mapped into the state s' of Π^L where $s'[V] = s$, and, for each $\phi \in L$, if $\phi \in L_s$, then $s'[v_\phi] = 0$, otherwise, $s'[v_\phi] = 1$.
3. The heuristic estimate for s is set to $h^{\mathcal{F}}(s')$. The latter is no longer a state-dependent, but path (or, in fact, multi-path) dependent estimate, and thus the machinery of *LM-A** is required. Importantly, however, this estimate is still admissible, and thus *LM-A** guarantees to find optimal solutions.

The right half of Table 3 (*LM-A** on Π; h on Π^L vs. A^*/h on Π) describes the effeciveness of such

domain ($\mathcal{D}$)	A^*/h on Π^L vs. Π										LM-A^* on Π; h on Π^L vs. A^*/h on Π									
	MS-10^4					$h^{\mathcal{F}}$					MS-10^4					$h^{\mathcal{F}}$				
	Π^L	Π	SB	E(Π^L)	E(Π)	Π^L	Π	SB	E(Π^L)	E(Π)	Π^L	Π	SB	E(Π^L)	E(Π)	Π^L	Π	SB	E(Π^L)	E(Π)
airport-ipc4	15	17	15	14.71	15.00	18	20	18	17.95	13.86	16	17	16	16.00	13.90	17	20	17	17.00	10.86
blocks-ipc2	20	18	18	16.83	13.35	18	21	18	3.59	17.60	18	18	18	17.26	11.46	17	21	17	5.98	17.00
depots-ipc3	6	7	6	4.43	5.84	4	7	4	1.41	4.00	4	7	4	4.00	3.28	4	7	4	2.97	3.59
driverlog-ipc3	11	12	11	9.94	9.77	11	12	11	8.25	9.77	11	12	11	11.00	8.66	11	12	11	10.84	6.64
freecell-ipc3	4	4	4	3.99	2.67	5	5	5	5.00	1.38	4	4	4	3.87	3.59	5	5	5	5.00	1.45
grid-ipc1	2	2	2	1.40	1.34	2	2	2	2.00	1.03	1	2	1	1.00	0.12	2	2	2	2.00	0.20
gripper-ipc1	4	7	4	1.12	4.00	5	7	5	0.75	5.00	6	7	6	5.01	5.91	6	7	6	6.00	5.22
logistics-ipc1	3	4	3	0.61	3.00	6	6	6	4.31	2.94	3	4	3	2.12	1.67	5	6	5	4.38	1.40
logistics-ipc2	16	16	16	10.76	16.00	20	22	20	11.91	20.00	14	16	14	10.84	13.03	20	22	20	11.42	20.00
miconic-strips-ipc2	44	55	44	35.61	43.75	52	51	48	45.16	21.40	63	55	48	43.04	38.87	108	51	51	51.00	8.17
mprime-ipc1	17	21	17	14.03	10.93	22	23	22	17.39	9.58	20	21	20	18.22	13.64	23	23	23	21.07	11.43
mystery-ipc1	16	17	15	14.29	14.01	21	21	21	16.65	16.86	19	17	17	16.91	15.96	21	21	21	19.44	16.44
openstacks-ipc5	7	7	7	7.00	6.96	7	7	7	7.00	1.26	7	7	7	2.00	7.00	7	7	7	7.00	1.27
pathways-ipc5	4	4	4	3.08	4.00	4	4	4	4.00	3.99	4	4	4	4.00	4.00	4	4	4	4.00	4.00
pipes-notank-ipc4	16	21	16	13.21	14.51	15	16	15	11.57	14.16	15	21	15	13.50	13.57	15	16	15	15.00	9.61
pipes-tank-ipc4	12	13	12	8.38	11.54	10	10	10	7.21	8.99	13	13	12	10.14	11.20	10	10	10	10.00	6.75
psr-small-ipc4	50	50	50	49.97	49.66	49	49	49	48.41	48.88	49	50	49	43.68	48.43	48	49	48	48.00	47.58
rovers-ipc5	6	6	6	5.80	5.94	6	6	6	6.00	3.14	5	6	5	5.00	4.24	5	6	5	5.00	2.01
satellite-ipc4	5	6	5	4.93	4.97	7	6	6	6.00	2.13	7	6	6	6.00	3.25	7	6	6	6.00	1.01
schedule-strips	19	19	17	17.00	17.00	46	46	46	42.38	43.61	19	19	19	19.00	19.00	46	46	46	46.00	46.00
tpp-ipc5	6	6	6	6.00	5.48	6	6	6	5.48	6.00	6	6	6	6.00	5.37	5	6	5	4.49	5.00
trucks-ipc5	4	5	4	2.79	3.57	7	6	6	6.00	1.79	4	5	4	2.29	3.89	7	6	6	6.00	0.84
zenotravel-ipc3	9	11	9	9.00	8.23	9	11	9	7.66	8.04	9	11	9	9.00	7.36	9	11	9	9.00	5.31
	296	328	291	254.89	271.48	350	364	344	286.07	265.44	317	328	298	269.91	257.41	402	364	343	317.59	231.77

Table 3: A summary of the experimental results for A^* search comparing landmarks enriched tasks to the original tasks, and for searching the original tasks, comparing LM-A^* with estimation on the landmark enriched tasks to the plain A^*. The columns are as described previously in Table 1.

a scheme comparatively to the basic approach of solving the original task Π with A^*. In general, the reduction in the number of state expansions comparatively to running A^* directly on the surrogates Π^L has been consistently substantial. Note that, despite this improvement of pruning, some tasks solved with $A^*/h^{\mathcal{F}}$ on Π^L have not been solved using *LM-A*** (e.g., task 8 in DRIVERLOG), and this because of a generally higher search-node processing time of *LM-A***. However, the opposite has been observed as well (e.g., task 13 of DRIVERLOG, and task 6 of GRIPPER), with the major difference being observed in the MICONIC domain: while A^* with $h^{\mathcal{F}}$ solved 51 and 52 tasks from this domain on their original and surrogate representations, respectively, using the *LM-A*** scheme allowed for solving 108 tasks in MICONIC.

Cost Partition Revisited

Note that, as it is exemplified by the still not good results in the BLOCKSWORLD domain, the original growth of the search space was not the only source of troubles. Another critical pitfall is the interplay between our fixed, ad hoc action cost partitioning among the patterns and the typical nature of landmarks. A landmark ϕ should take place only at *some* time point along the plan. Very roughly, once ϕ is achieved, the portions of the action costs "assigned to the achievement of ϕ" are getting lost, resulting in erosion of the heuristic values. This is where better action cost partition can potentially improve the situation.

Table 4 compares the number of tasks solved with *LM-A*** under our time and memory limits using the basic uniform action cost partition with this using (on the left) per-state optimal action cost partition and (on the right) fixed action cost partitions being optimal for the initial states of the tasks. Note that, in the case of per-state optimal action cost partition, the evaluation is based on solving very large linear programs, and thus the evaluation time increases substantially. As a direct result, despite a dramatic reduction in node expansions, the total number of tasks solved within the time limit decreases.[2] Having said that, still some nine tasks of the MICONIC domain that were not solved with any other fork-decomposition approach are solved under the per-state optimal action cost partition. In the case of (fixed) optimal for the initial state action cost partition the results are much more promising. On most domains, the number of expanded nodes decreases, resulting in increase in the number of tasks solved on ROVERS, SCHEDULE-STRIPS, TRUCKS, and ZENOTRAVEL. Overall, the number of solved tasks either increased or remained the same on all the domains considered in the evaluation. In other words, even a fixed but not entirely ad hoc action cost partition can provide a nice balance between the effort invested in heuristic computation and the benefit it buys us in the time limited settings.

[2] On some problems solved under optimal action cost partitioning, there were still a few search nodes for which our LP solver failed solving the respective optimization problems. To avoid terminating the search, the heuristic value for these nodes was set to 1.

domain ($\mathcal{D}$)	optimal vs. uniform					optimal for I vs. uniform				
	O	U	SB	E(O)	E(U)	OI	U	SB	E(OI)	E(U)
airport-ipc4	7	17	7	6.77	6.52	17	17	17	17.00	17.00
blocks-ipc2	13	17	13	13.00	4.37	17	17	17	13.98	13.47
depots-ipc3	1	4	1	1.00	0.22	4	4	4	4.00	3.46
driverlog-ipc3	6	11	6	5.38	1.33	11	11	11	11.00	7.56
freecell-ipc3	2	5	2	2.00	0.07	5	5	5	5.00	3.22
grid-ipc1	0	2	0	0.00	0.00	2	2	2	2.00	2.00
gripper-ipc1	2	6	2	1.96	1.71	6	6	6	6.00	5.55
logistics-ipc2	17	20	17	17.00	11.20	20	20	20	19.01	18.62
logistics-ipc1	2	5	2	2.00	0.42	5	5	5	4.83	4.03
miconic-strips-ipc2	90	108	81	80.91	51.26	108	108	108	88.63	99.37
mprime-ipc1	9	23	9	9.00	2.79	23	23	23	21.98	18.08
mystery-ipc1	11	21	11	10.50	8.34	21	21	21	21.00	19.34
openstacks-ipc5	5	7	5	5.00	1.95	7	7	7	7.00	2.50
pathways-ipc5	4	4	4	4.00	2.36	4	4	4	4.00	2.37
pipesworld-notankage-ipc4	3	15	3	3.00	1.14	15	15	15	15.00	10.39
pipesworld-tankage-ipc4	2	10	2	2.00	0.91	10	10	10	10.00	6.76
psr-small-ipc4	40	48	40	40.00	26.98	48	48	48	35.86	47.94
rovers-ipc5	4	5	4	4.00	2.60	7	5	5	4.28	3.32
satellite-ipc4	5	7	5	5.00	2.30	7	7	7	5.61	5.02
schedule-strips	33	46	33	32.96	16.33	47	46	43	31.24	29.49
tpp-ipc5	5	5	5	5.00	4.02	5	5	5	5.00	4.02
trucks-ipc5	4	7	4	4.00	0.76	8	7	7	6.69	5.01
zenotravel-ipc3	7	9	7	7.00	2.48	11	9	9	8.84	5.24
	272	402	263	261.47	150.06	408	402	399	347.94	333.76

Table 4: A summary of the experimental results for $h^{\mathcal{F}}$ heuristic with optimal action cost partition per evaluated state (O) vs. fixed uniform cost partition (U) and fixed optimal for initial state partition (OI) vs. fixed uniform cost partition (U). The notation is similar to this in Table 1.

Summary

Motivated by an observation that the quality of abstraction heuristics heavily depends on the richness of the explicit goal specification in the planning tasks, we have investigated exploiting implicit goals in the form of discoverable landmarks for enhancing abstraction heuristics. We proposed a concrete scheme for such an enhancement that is based on (i) compiling the landmarks into the planning task specification, and (ii) using the compiled task within heuristic-search planning side by side with the original task. While this scheme is applicable to arbitrary heuristics, our empirical evaluation showed that it is especially attractive for structural-pattern heuristics, and less so for explicit abstractions. The major reason for that appears to be the fact that our approach to landmark compilation into the task increases the dimensionality of the latter up to a linear factor. While structural-pattern abstractions are not sensitive to that matter, this is substantially less so for explicit abstractions. Hence, probably the most valuable direction for future research is looking for alternative ways of incorporating landmark information into abstractions that would fit well the specifics of the explicit abstractions such as PDBs and merge-and-shrink.

References

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

Edelkamp, S. 2001. Planning with pattern databases. In *ECP*, 13–24.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *ICAPS*, 140–149.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI*, 1007–1012.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *ICAPS*, 162–169.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, 200–207.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR* 22:215–278.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*.

Katz, M., and Domshlak, C. 2008a. Optimal additive composition of abstraction-based admissible heuristics. In *ICAPS*, 174–181.

Katz, M., and Domshlak, C. 2008b. Structural patterns heuristics via fork decomposition. In *ICAPS*, 182–189.

Katz, M., and Domshlak, C. 2009. Structural-pattern databases. In *ICAPS*, 186–193.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *ECP*.

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