Cost-Optimal Planning with Landmarks

Erez Karpas and **Carmel Domshlak**
Faculty of Industrial Engineering & Management
Technion, Israel

Abstract

Planning landmarks are facts that must be true at some point in every solution plan. Previous work has very successfully exploited planning landmarks in satisficing (non-optimal) planning. We propose a methodology for deriving admissible heuristic estimates for cost-optimal planning from a set of planning landmarks. The resulting heuristics fall into a novel class of multi-path dependent heuristics, and we present a simple best-first search procedure exploiting such heuristics. Our empirical evaluation shows that this framework favorably competes with the state-of-the-art of cost-optimal heuristic search.

1 Introduction

Landmarks for deterministic planning are (possibly logically compound) facts that must take place at some point in every plan for a given planning task [Porteous *et al.*, 2001; Porteous and Cresswell, 2002; Hoffmann *et al.*, 2004]. For example, if a goal in a Blocksworld task is to have block *A* stacked on block *B*, and initially this does not hold, then *clear(B)* must hold at some point for the goal to be achieved, and thus it is a landmark for that task. Goals are trivially landmarks, and thus *on(A,B)* is a landmark as well. We can also infer that *clear(B)* must be achieved before stacking *A* on *B*, establishing an ordering between these two landmarks.

The two issues with planning landmarks are how to discover them, and how to exploit them. Even for propositional landmarks only, sound and complete discovery of all such landmarks is known to be PSPACE-complete [Porteous *et al.*, 2001]. Still, many landmarks can often be efficiently discovered, e.g., by a polynomial time reasoning over relaxed planning graphs [Hoffmann *et al.*, 2004], and/or domain transition graphs of multi-valued state variables [Richter *et al.*, 2008].

Once discovered, landmarks can be extremely helpful in guiding the search for a plan. In particular, Hoffmann *et al.* [2004] suggest a local-search procedure that iteratively searches for sub-plans to landmarks "needed earliest", and this instead of searching directly for a plan to the problem's goal. An empirical evaluation by the same authors has shown a substantial speed-up compared to the otherwise identical planner that does not use landmarks. However, the higher greediness of this approach typically results in longer plans, and the iterative search for subgoals may sometimes lead to incompleteness of otherwise complete planners [Hoffmann *et al.*, 2004]. Recently, Richter *et al.* [2008] proposed a novel way of using a set of landmarks as a pseudo-heuristic within a satisficing heuristic search. This technique allowed both reducing the length of the generated plans, as well as improving success rate both with respect to the iterative approach of Hoffmann *et al.*, and with respect to other state-of-the-art satisficing planners. In particular, the LAMA planner by Richter and Westphal utilizing such a landmarks-based heuristic search was the clear winner of the Sequential Satisficing Track at the 2008 International Planning Competition.

In this work, we depart from satisficing planning, and consider planning landmarks through the lens of cost-optimal planning needs. We suggest a method for deriving *admissible estimates from a set of planning landmarks*, with its instances varying from easy to compute, to, in some sense, optimally accurate. The resulting heuristics are what we call *multi-path dependent*. We present a simple best-first search that exploits such heuristics, and finds optimal solutions more efficiently than standard A^*. Experimentally, we show that our heuristics paired with the suggested best-first search procedure are at the state-of-the-art of cost-optimal heuristic search planning, and also offer an unprecedented scalability of the estimate's accuracy in the size of the planning tasks.

2 Notation and Background

We consider planning in the SAS$^+$ formalism [Bäckström and Nebel, 1995]; a SAS$^+$ description of a planning task can be automatically generated from its PDDL description [Helmert, 2009]. A SAS$^+$ task is given by a 5-tuple $\Pi = \langle V, A, s_0, G, \mathcal{C} \rangle$. $V = \{v_1, \ldots, v_n\}$ is a set of *state variables*, each associated with a finite domain $dom(v_i)$, where (assuming name uniqueness) the union of the variable domains $F = \bigcup_i dom(v_i)$ is the set of *facts* . Each complete assignment s to V is called a *state*; s_0 is an *initial state*, and the *goal* G is a partial assignment to V. A is a finite set of *actions*, where each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a) \rangle$ of partial assignments to V called *preconditions* and *effects*, respectively. Each action $a \in A$ has a non-negative cost $\mathcal{C}(a)$.

An action a is applicable in a state $s \in dom(V)$ iff $\mathsf{pre}(a) \subseteq s$, and applying such a changes the value of each state variable v to $\mathsf{eff}(a)[v]$ if $\mathsf{eff}(a)[v]$ is specified. 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 a plan if $G \subseteq s[\![\langle a_1, \ldots, a_k \rangle]\!]$.

Let $\Pi = \langle V, A, s_0, G, \mathcal{C} \rangle$ be a planning task, ϕ be a propositional logic formula over facts F, $\pi = \langle a_1, \ldots, a_k \rangle$ be an action sequence applicable in s_0, and $0 \leq i \leq k$. Following the terminology of Hoffmann *et al.*, we say that ϕ is *true at time* i in π iff $s_0[\![\langle a_1, \ldots, a_i \rangle]\!] \models \phi$, ϕ is *first added at time* i in π iff ϕ is true in π at time i, but not at any time $j < i$, and ϕ is a *landmark* of Π iff in each plan for Π, it is true at some time. In addition to knowing landmarks, it is sometimes useful to know in which order they should be achieved on the way to the goal. Hoffmann *et al.* define different types of potentially useful orderings. In particular, landmark ϕ is said to be *greedy-necessarily ordered* before landmark ψ iff, for each action sequence applicable in s_0, if ψ is first added in π at time i, then ϕ is true in π at time $i - 1$.

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. Porteous *et al.* show that deciding if just a single fact is a landmark, as well as deciding an ordering between two fact landmarks, are PSPACE-complete problems. Therefore, 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 evaluation we use LAMA's sound landmark discovery procedure, introduced by Richter *et al.* [2008]. The actual way of discovering landmarks and orderings is tangential to our contribution. Hence, in what follows we assume that a planning task Π is simply given to us with a landmark structure $\langle L, Ord \rangle$, where L is a set of Π's landmarks, and Ord is a set of typed orderings over L, containing, in particular, the greedy-necessary ordering over L.

3 Admissible Landmark Heuristics

Deriving heuristic estimates from landmarks has been proposed by Richter et al. [2008] who estimate the goal distance of a state s, reached via a sequence of actions π from the initial state, by the *number of landmarks* $L(s, \pi)$ *yet to be achieved from* s *onwards*. Specifically, if the search starts with the landmark structure $\langle L, Ord \rangle$, then

$$L(s, \pi) = L \setminus (\mathsf{Accepted}(s, \pi) \setminus \mathsf{ReqAgain}(s, \pi)) \qquad (1)$$

where $\mathsf{Accepted}(s, \pi) \subseteq L$ and $\mathsf{ReqAgain}(s, \pi) \subseteq \mathsf{Accepted}(s, \pi)$ are the sets of *accepted* and *required again* landmarks, respectively. A landmark is accepted if it is true at some time along π. An accepted landmark is required again if (i) it does not hold in s, and (ii) it is ordered greedy-necessarily before some landmark which is not accepted.

The estimate $|L(s, \pi)|$ is not a proper heuristic in the usual sense, but rather *path-dependent*; it is a function of both an evaluated state s, and a path from s_0 to s. However, $|L(s, \pi)|$ can still be used like a state-dependent heuristic in best-first search. In particular, combined with some other helpful techniques, it has been successfully used by the LAMA planner at the Sequential Satisficing Track of the IPC-2008 competition.

3.1 Action Cost Sharing by Landmarks

It is not hard to verify that the estimate $|L(s, \pi)|$ is not admissible. For instance, in a Blocksworld task, let $L(s, \pi) = \{$*crane-empty*, *on(A,B)*$\}$. While $|L(s, \pi)| = 2$, it is possible a single action *stack(A,B)* reaches the goal from s. However, below we show that the gap between the estimate $|L(s, \pi)|$ and admissibility is not that hard to close.

Considering the landmarks through the actions that can possibly achieve them, let $cost(\phi)$ be a cost assigned to each landmark ϕ, and $cost(a, \phi)$ be a cost "assigned" by the action a to ϕ. Suppose also that these (all non-negative) costs satisfy

$$\begin{aligned} \forall a \in A : \sum_{\phi \in L(a|s,\pi)} cost(a, \phi) \leq \mathcal{C}(a) \\ \forall \phi \in L(s, \pi) : \; cost(\phi) \leq \min_{a \in \mathsf{ach}(\phi|s,\pi)} cost(a, \phi) \end{aligned} \qquad (2)$$

where each action subset $\mathsf{ach}(\phi|s, \pi) \subseteq A$ (in particular) contains all the actions that can possibly be used to directly achieve landmark ϕ along a goal-achieving suffix of π, and, reversely, $L(a|s, \pi) = \{\phi \mid \phi \in L(s, \pi), a \in \mathsf{ach}(\phi|s, \pi)\}$. Informally, Eq. 2 enforces partitioning of each action cost among the landmarks this action can possibly establish, and verifies that the cost of each landmark ϕ is no greater than the minimum cost assigned to ϕ by its possible achievers. The set $\mathsf{ach}(\phi|s, \pi) \subseteq A$ for a disjunctive landmark ϕ can be simply estimated by the set of all actions achieving some element of ϕ. In our evaluation we use the (initial-state dependent and efficiently computable) set of "possible", and its subset of "first-time possible", achievers of ϕ [Porteous and Cresswell, 2002]. If $\phi \notin \mathsf{Accepted}(s, \pi)$, then we set $\mathsf{ach}(\phi|s, \pi)$ to the first-time possible achievers of ϕ, and otherwise to the possible achievers of ϕ. In any event, action cost sharing is all we need to derive from $L(s, \pi)$ an admissible estimate of the goal distance.

Proposition 1 *Given a set of action-to-landmark and landmark costs satisfying Eq. 2,* $h_L(s, \pi) = cost(L(s, \pi)) = \sum_{\phi \in L(s,\pi)} cost(\phi)$ *is an admissible estimate of the goal distance* $h^*(s)$.

The (simple) proof is omitted due to space constraints. Proposition 1 leaves the choice of the actual action-cost partitioning open. The most straightforward choice here is probably *uniform cost sharing* in which each action partitions its costs equally among all the landmarks it can possibly achieve, that is, $cost(a, \phi) = \mathcal{C}(a)/|L(a|s, \pi)|$. The advantage of such a uniform cost sharing is the efficiency of its computation. However, the induced action-cost partition can be sub-optimal. For instance, consider a planning task with a landmark set $\{p_1, \ldots, p_k, q\}$ such that the only possible achiever of each p_i is a unit-cost action a_i with $\mathsf{eff}(a_i) = \{p_i, q\}$. For $1 \leq i \leq k$, the uniform cost sharing assigns here $cost(a_i, p_i) = cost(a_i, q) = 0.5$, which gives $cost(p_i) = cost(q) = 0.5$, and thus $h_L(s, \pi) = k/2 + 0.5$. In contrast, the optimal cost sharing would assign, for all $1 \leq i \leq k$, $cost(a_i, p_i) = 1$ and $cost(a_i, q) = 0$, implying $cost(p_i) = 1$, $cost(q) = 0$, and thus $h_L(s, \pi) = k$.

The good news, however, is that such an optimal cost sharing can be computed in poly-time by (i) compiling Eq. 2 into

strictly linear constraints by a trivial elimination of the minimization, and (ii) solving a linear program induced by these constraints and the objective $\max \sum_{\phi \in L(s,\pi)} cost(\phi)$. In addition, this cost sharing scheme alleviates an annoying shortcoming of ad hoc (e.g., uniform) cost sharing schemes, and satisfies *monotonicity along the inclusion relation of the landmark sets* $L(s,\pi)$. It is not hard to verify that, for any two sets of landmarks L and L' such that $L \subseteq L'$, the LP-based cost sharing ensures $cost(L') \geq cost(L)$ by the very virtue of being optimal, and thus yields at least as informative heuristic estimate with L' as with L. This property is appealing as it allows separating landmark discovery and landmark exploitation without any loss of accuracy, leaving the phase of discovery with the simple principle of "more landmarks can never hurt". In contrast, the simple yet ad hoc uniform cost sharing cannot guarantee such monotonicity. For instance, the uniform cost sharing in the example above but *without* landmark q yields $h_L(s,\pi) = k$, while with q it results in $h_L(s,\pi) = k/2 + 0.5$.

3.2 Action Landmarks

The LP-based "admissibilization" of the landmark sets $L(s,\pi)$ is optimal, but this, of course, only when the landmark costs are estimated with respect to solely Eq. 2. Any additional information about landmarks may help improving the accuracy of the estimate. One type of such information corresponds to *action landmarks* [Zhu and Givan, 2004; Vidal and Geffner, 2006]. Similarly to landmarks over facts, an action a is an action landmark of a planning task Π iff it is taken along every plan for Π. A sufficient and efficiently testable condition for a being an action landmark is that a relaxed planning task without a is not solvable.

Now, let $U(s,\pi)$ be a set of pre-discovered action landmark that were *not* used along the path π to the evaluated state s. By the definition of landmarks, we know that all these actions must occur at some time after s, and thus we should account for their cost in the heuristic estimate of (s,π). A simple way of incorporating this information into the cost sharing mechanism as above is what we call *action landmark covering*: First, sum up the cost of all the unused action landmarks $U(s,\pi)$, then remove from the landmark set $L(s,\pi)$ all the landmarks achievable by the actions in $U(s,\pi)$, and only then perform the regular action cost sharing over the remaining landmarks. The resulting heuristic estimate is $h_{LA}(s,\pi) = cost(L_U(s,\pi)) + \sum_{a \in U(s,\pi)} \mathcal{C}(a)$, where $L_U(s,\pi) = L(s,\pi) \setminus \bigcup_{a \in U(s,\pi)} L(a|s,\pi)$, and $cost(L_U(s,\pi))$ is computed using a cost sharing as in Eq. 2.

Proposition 2 *Given a planning task, a set of its landmarks, and a set of its action landmarks, for any state s, and any path π to s, we have $h_{LA}(s,\pi)$ being an admissible estimate of the goal distance $h^*(s)$, dominating $h_L(s,\pi)$ under any admissible cost sharing scheme as in Eq. 2.*

The intuition behind the (only technically tedious) proof is that Eq. 2 implies $\mathcal{C}(a) \geq cost(L(a|s,\pi))$, and thus enforcing a to be the only achiever of $L(a|s,\pi)$ can only increase the overall estimate. The admissibility follows from a being an action landmark.

Action landmark covering of $h_{LA}(s,\pi)$ is especially helpful when practical considerations of reducing per-search-node computations force us to give up on the more accurate yet relatively costly LP-based cost sharing, and settle for a low cost, ad hoc cost sharing scheme such as the uniform one. In such cases, the main benefit of action landmark covering may actually happen to be indirect. For instance, consider our running example with a landmark set $\{p_1, \ldots, p_k, q\}$ such that the only achiever of each p_i is a unit-cost action a_i with $\mathsf{eff}(a_i) = \{p_i, q\}$. In Section 3.1 we show that the uniform cost sharing results in $h_L(s,\pi) = k/2 + 0.5$. However, if an action a_i is known to be an action landmark, then the cost sharing for $h_{LA}(s,\pi)$ is performed only over $L_U(s,\pi) = \{p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_k\}$, providing us with $cost(L_U(s,\pi)) = k - 1$, and thus $h_{LA}(s,\pi) = k$. In other words, the action landmark covering may contribute not only via full accounting for the action landmark costs, but also via indirect elimination of some "action cost eroders" such as the landmark q in our example. This example actually is not just schematic as, e.g., a very much similar picture occurs with the *crane-empty* landmark in Blocksworld tasks. Later we empirically illustrate that the impact of action landmark covering appears substantial in many domains.

4 From Path to Multi-Path Dependence

Let us now return to the definition of the path-dependent set $L(s,\pi)$ in Eq. 1. Both LAMA's heuristic $|L(s,\pi)|$, and the admissible heuristics h_L and h_{LA}, exploit information provided by the path π to better estimate the goal distance from s. Suppose now that we are given a set of paths from s_0 to s; such a set of paths can in particular be discovered anyway by any systematic, forward-search procedure. Proposition 3 shows that such a set of paths can be much more informative than any of its individual components.

Proposition 3 *Let Π be a planning task with a landmark set L, s be a state of Π, $\mathcal{P}$ be a set of paths from s_0 to s, and π_g be a goal achieving path from s. For each path $\pi \in \mathcal{P}$:*

1. *π_g achieves all landmarks in $L \setminus \mathsf{Accepted}(s,\pi)$.*
2. *π_g applies all actions in $U(\pi,s)$.*

The proof is straightforward: Assume a landmark ϕ is achieved by a path $\pi \in \mathcal{P}$ but not by a path $\pi' \in \mathcal{P}$. The latter implies that all the extensions of π' should still achieve ϕ, and the extensions of π' are exactly the extensions of π. The same argument applies with unused action landmarks.

Proposition 3 immediately leads to *multi-path dependent* versions of h_L and h_{LA}. Given a set of landmarks L, and a set of paths $\mathcal{P}$ from s_0 to s, let

$$\begin{aligned} U(s,\mathcal{P}) &= \bigcup_{\pi \in \mathcal{P}} U(s,\pi) \\ L(s,\mathcal{P}) &= L \setminus (\mathsf{Accepted}(s,\mathcal{P}) \setminus \mathsf{ReqAgain}(s,\mathcal{P})) \end{aligned} \tag{3}$$

where $\mathsf{Accepted}(s,\mathcal{P}) = \bigcap_{\pi \in \mathcal{P}} \mathsf{Accepted}(s,\pi)$, and $\mathsf{ReqAgain}(s,\mathcal{P}) \subseteq \mathsf{Accepted}(s,\mathcal{P})$ is specified as before by s and the greedy-necessary orderings over L. Given that,

the multi-path dependent versions of h_L and h_{LA} straightforwardly reflect their path-dependent counterparts as

$$h_L(s,\mathcal{P}) = cost(L(s,\mathcal{P}))$$
$$h_{LA}(s,\mathcal{P}) = cost(L_U(s,\mathcal{P})) + \sum_{a\in U(s,\mathcal{P})} \mathcal{C}(a) \quad (4)$$

The improvement in accuracy in switching to multi-path landmark heuristics can be substantial. For instance, if we have access to two paths to s, each suggesting that half of the landmarks have been achieved, yet they entirely disagree on the identity of the achieved landmarks, then the estimate of the (still admissible) multi-path dependent heuristic might be twice as high as this of the path-dependent heuristic.

Finally, utilizing multi-path dependent estimates for optimal search requires adapting the standard A^* search procedure. In fact, a slight adaptation of A^* is desirable even in case of such path-dependent heuristics. Designed for state-dependent estimates, A^* computes $h(s)$ for each state s only when s is first generated. This will still guarantee optimality with path-dependent estimates as well, yet, if π and π' are the current and a new discovered paths from s_0 to s, respectively, then we may have $h(s,\pi') > h(s,\pi)$. That is, a new discovered path may better inform us about the goal distance from s. We can slightly modify A^* to compute the heuristic value each time a new path to a state is discovered, and utilize the highest estimate discovered so far. This, of course, preserves search admissibility, and potentially reduces the number of expanded nodes. Note that this does not contradict "optimal efficiency" of the basic A^* as the latter holds only for monotonic, state-dependent heuristics [Dechter and Pearl, 1985].

The modification of A^* for multi-path dependent heuristics is very much similar in spirit. Each time a new path to state s is discovered, it is stored in the list of such paths $\mathcal{P}(s)$, and $h(s,\mathcal{P}(s))$ is evaluated. Of course, storing all paths to s is generally infeasible, and the algorithm is usable only in cases where the relevant information carried by $\mathcal{P}(s)$ can be captured and stored compactly. In fact, the adaptation of A^* to path-dependent heuristics as above constitutes such a special case of all the relevant information of a set of paths being the maximal value of the heuristic estimates induced by them individually. Nicely, the multi-path dependent landmark heuristics h_L and h_{LA} also constitute a usable special case as above. In our variant of A^*, referred later as *LM-A**, we associate each state s with the landmark sets $L(s,\mathcal{P}(s))$ and $U(s,\mathcal{P}(s))$ as in Eq. 3. When a new path π to s is discovered (and extends $\mathcal{P}(s)$), the respective sets are incrementally updated to $L(s,\mathcal{P}(s)\cup\{\pi\})$ and $U(s,\mathcal{P}(s)\cup\{\pi\})$ by exploiting the monotonicity of the union and intersection set operators.

5 Experimental Evaluation

Encouraging results obtained with landmark-based heuristics in satisficing planning do not necessarily imply that optimal search with landmark-based heuristics will also work well in practice. To evaluate the practical usefulness of the constructs presented above, we implemented both our admissible heuristics and the *LM-A** search procedure on the infrastructure of the Fast Downward planner [Helmert, 2006], and used the landmark discovery of LAMA [Richter *et al.*, 2008]. We conducted an empirical study on a wide sample of planning domains from the international planning competitions 1998-2006. All experiments were run on 3GHz Intel E8400 CPU; the time and memory limits were 30 minutes and 1.5 GB. The reported times do not include the PDDL to SAS$^+$ translation as it is common to all planners, and is tangential to the issues considered in our study.

There are three issues on the agenda of our study. First, we evaluate the marginal contributions of both action landmark covering and using *LM-A**, that is, h_{LA} vs. h_L, and *LM-A** vs. *A**. Here we focus on the h_{LA} heuristic based on the uniform cost sharing. While it is potentially less informative than h_{LA} under the LP-based cost sharing, our implementation of the latter was too costly to be applied at every search node under the time limit of 30 minutes. Moreover, our selective tests indicated that, while LP-based cost sharing substantially improves the accuracy of h_L, its contribution to action-landmark-covering h_{LA} is typically insignificant. (In Section 3.2 we discuss the key reason for that.)

The left part of Table 1 (up to and including the shaded column *LM-A**+h_{LA}) compares *LM-A** using h_{LA} with both *A** using the same heuristic, as well as *LM-A** using h_L. Table 1 depicts the results obtained over BLOCKSWORLD, LOGISTICS, DEPOTS, and SATELLITE, which represent well the picture obtained across all the considered domains. The table lists only tasks that have been solved by one of the planners. Empty entries in the table denote tasks that were not solved by the respective technique. First, note that *LM-A** with h_{LA} outperforms the standard *A** using the same heuristic both quantitatively and qualitatively, solving 15%, 43%, and 53% more tasks in BLOCKSWORLD, LOGISTICS, and DEPOTS, respectively. Only in SATELLITE *LM-A** does not depart far from *A**. The performance gain from switching within *LM-A** from h_L to the more accurate h_{LA} was also non-negligible: With h_{LA} we were able to solve two more tasks than with h_L, with savings in time and the number of expanded nodes per task reaching 70%. Note that there was no difference between h_L and h_{LA} in LOGISTICS, but that is not surprising because the SAS$^+$ representation of LOGISTICS is unary-effect, and action landmark covering cannot help in such domains with unit-cost actions.

Next we check whether a heuristic planner based on *LM-A** with h_{LA} is competitive with the state of the art of forward-search cost-optimal planning. To answer this question, we perform a comparison to two baseline approaches, namely blind search (*A** with a heuristic function which is 0 for goal states and 1 otherwise) and *A** with the $h_{\max}$ heuristic [Bonet and Geffner, 2001]. We also compare to *A** with "flexible abstraction" (*FA*) heuristics of Helmert, Haslum, and Hoffmann [2007]. We use *FA* under the linear abstraction strategy suggested by the same authors, and under two fixed bounds on the size of the abstraction, 10^4 and 10^5. From our literature survey, *FA* appears to be one of the best-performing forward-search planners. To allow a fairly unbiased comparison, all the heuristics were implemented within the same planning system, and our *LM-A** was implemented on top of the same *A** implementation.

The right part of Table 1 (from and including the shaded column *LM-A** + h_{LA}) compares *LM-A** with h_{LA} with the

		$A^* + h_{LA}$		*LM-A** $+ h_L$		*LM-A** $+ h_{LA}$		*FA*-10^4		*FA*-10^5		h_{max}		blind	
task	C^*	*time*	*nodes*	*time*	*nodes*	*time*	*nodes*	*time*	*nodes*	*time*	*nodes*	*time*	*nodes*	*time*	*nodes*
	BLOCKSWORLD														
4-0	6	0	15	0	15	0	15	0.03	7	0.03	7	0	25	0.01	95
4-1	10	0.01	13	0	15	0	13	0.04	11	0.03	11	0	23	0.01	66
4-2	6	0.01	7	0.01	8	0	7	0.04	7	0.03	7	0.01	18	0	61
5-0	12	0	31	0.01	41	0	31	0.3	13	0.97	13	0.01	145	0	467
5-1	10	0.01	35	0	45	0	31	0.3	11	0.97	11	0.01	135	0.01	561
5-2	16	0.02	87	0.01	158	0.01	87	0.3	17	0.97	17	0.01	297	0	798
6-0	12	0.01	22	0.01	29	0	22	0.84	13	8.03	13	0.01	276	0.01	1826
6-1	10	0.02	39	0.01	44	0.01	39	0.86	11	8.12	11	0.01	755	0.02	4911
6-2	20	0.15	1146	0.12	943	0.21	933	0.97	733	8.66	68	0.04	2556	0.02	6409
7-0	20	0.06	249	0.06	292	0.08	240	1.92	579	23.04	144	0.11	5943	0.14	36333
7-1	22	1.68	10952	2.09	12224	2.44	9032	1.62	9977	19.47	1887	0.45	33194	0.21	63376
7-2	20	0.38	1941	0.45	2452	0.41	1453	1.56	1855	19.06	611	0.29	18293	0.19	55218
8-0	18	0.22	894	0.41	1762	0.29	827	3.93	5570	38.48	692	2	94671	2.18	519107
8-1	20	1.84	8283	2.63	10810	1.92	5647	4.17	45706	34.82	11880	3.83	199901	2.53	636138
8-2	16	0.03	95	0.12	504	0.08	219	3.55	293	33.56	63	1.28	52717	1.89	434664
9-0	30			175.36	469633	132.87	260857	14.38	1232639	66.73	971416	81.21	3840589	36.58	7983389
9-1	28	6.64	18609	8	22337	7.93	14852	9.41	94991	61.27	58867	30.89	1200345	29.49	5922420
9-2	26	1.63	4712	4.87	12572	3.4	6581	11.19	161653	103.8	20108	31.33	1211463	29.93	5984400
10-0	34							266.2	32869439						
10-1	32					919.68	1393515	201.66	23517042	275.71	12063661				
10-2	34							271.61	33331658	288.23	18457528				
11-0	32							195.71	16219694	138.77	7077045				
12-1	34			612.02	901023	250.78	272199								
	DEPOTS														
1	10	0.01	33	0.01	31	0.02	31	0.01	11	0.01	11	0.01	136	0	329
2	15	0.13	668	0.2	790	0.21	614	2.83	875	1.15	16	0.18	3771	0.11	15404
3	27	193.32	810127	121.36	169765	52.43	72979	24.74	348300	235.01	239255	96.47	1204646	27.16	2930398
4	30					1061.04	1041194	50.08	1284029	458.39	1219026				
7	21	213.66	890532	69.15	86052	29.72	39348	39.12	212544			163.48	1331701	71.79	6501100
10	24			400.99	447195	193.32	199412	156.91	3240433						
13	25			45.07	42808	41.66	27977	122.03	1427824	499.05	1183545				
	LOGISTICS														
4-0	20	0.09	258	0.01	76	0.04	76	0.04	21	0.05	21	0.04	4884	0.06	11246
4-1	19	0.18	1238	0.02	195	0.08	195	0.04	20	0.05	20	0.03	4185	0.05	9249
4-2	15	0.02	111	0.01	53	0.01	53	0.04	16	0.05	16	0.01	1205	0.03	4955
5-0	27	13.19	123081	0.13	936	0.64	936	0.1	28	0.37	28	0.58	74694	0.6	109525
5-1	17	0.16	1012	0.02	181	0.08	181	0.1	18	0.39	18	0.06	6199	0.12	22307
5-2	8	0	9	0.01	9	0.01	9	0.1	9	0.38	9	0.01	280	0	1031
6-0	25	35.49	244606	0.14	934	0.67	934	0.18	26	1.22	26	1.88	202229	3.27	490207
6-1	14	0.02	35	0.01	35	0.02	35	0.19	15	1.25	15	0.04	3604	0.16	24881
6-2	25	34.13	305202	0.07	516	0.37	516	0.19	26	1.24	26	1.89	200012	3.13	476661
6-9	24	33.28	337194	0.07	537	0.31	537	0.18	25	1.22	25	1.27	133521	2.79	422557
7-0	36			2.25	7751	9.84	7751	0.63	525	4.78	37				
7-1	44			57.09	155289	220.08	155289	8.94	666324	4.91	49				
8-0	31			0.86	3269	3.73	3269	0.92	1042	6.8	32				
8-1	44			17.29	43665	71.79	43665	1.14	16708	7.18	45				
9-0	36			5.41	14090	21.16	14090	1.53	20950	9.62	37				
9-1	30			0.24	707	1.05	707	1.23	31	9.33	31				
10-0	45			121.2	194038	416.06	194038			29.92	668834				
10-1	42			106.7	166190	356.38	166190			43.57	1456770				
12-0	42			87.06	117387	290.77	117387			43.44	775996				
12-1	68									88.68	2222340				
	SATELLITE														
1	9	0	10	0	10	0.01	10	0.01	10	0.01	10	0.01	59	0	89
2	13	0	14	0.01	14	0	14	0.02	14	0.01	14	0.01	940	0.01	1728
3	11	0.04	494	0.09	494	0.09	494	0.36	12	0.77	12	0.11	6822	0.16	15185
4	17	0.23	710	0.32	710	0.47	710	0.85	6780	4.35	18	3.34	180815	4.56	344380
5	15	3.91	12013	3.36	12013	7.78	12013	7.62	82393	74.63	134288	600.31	13973721		
6	20	10.06	24254	18.14	24200	25.19	24200	73.13	2534223	22.95	124001	368.23	10751017	641.65	25427494
7	21	173.91	157984	291.69	157984	290.03	157984								

Table 1: Runtimes of optimal planners across the test domains. Column *task* denotes problem instance, column C^* denotes optimal solution length. Other columns denote run *time* and number of expanded (*nodes*) of different cost-optimal heuristics and search procedures, including the (shaded) combination of *LM-A** with h_{LA} heuristic, the focus of our evaluation.

four referent planners over the same four domains as above. It is clear from the table that h_{LA} substantially outperforms both blind search and $h_{\max}$, and it is generally competitive with both *FA*-10^4 and *FA*-10^5. For a macro picture, Table 2 summarizes our full experimental results for (*LM-A** with) h_{LA} and (*A** with) *FA*-10^4 over tasks from 18 domains solved by at least one of these two planners. The summary is in terms of the number of solved tasks, as well as the averaged runtime and number of expanded nodes. On 7 domains h_{LA} solved more tasks than *FA*-10^4, on 5 domains it was the other way around, and on 6 domains they solved exactly the same instances. The aggregation of the results in the last line of the table also shows that h_{LA} solved slightly more tasks in total, and (on average) expanded fewer nodes and required less time on tasks solved by both planners. Of course, any empirical results on this or another concrete set of benchmarks should be taken very carefully, without over-interpretation. However, we can certainly observe that our landmarks-based approach favorably competes with the current state of the art for cost-optimal planning.

In the third part of our study we check the promise of h_{LA} beyond the currently solvable tasks. Many possible developments may in the future push the envelope of "what is feasible" in optimal planning, and some of these developments (in, e.g., reducing the amount of symmetry in optimal search) will be tangential to the accuracy of the heuristics in use. In

domain	N^+/N^1	*solved*		*nodes*		*time*	
		h_{LA}	*FA*	h_{LA}	*FA*	h_{LA}	*FA*
airport	18/24	24	18	1395	528152	8	123
blocks	19/23	20	22	89179	1319533	56	13
depots	7/7	7	7	197365	930573	196	56
driverlog	13/14	14	13	109611	765930	53	14
freecell	5/7	7	5	8487	1406793	10	232
grid	2/2	2	2	15313	1705511	12	29
gripper	6/8	6	8	403888	406411	71	3
logistics	16/19	19	16	14265	44111	20	0
mystery	13/13	13	13	95800	26684	17	25
pathways	4/4	4	4	43550	31051	5	8
psr	48/50	48	50	14541	3267	3	0
pw-no-tank	16/21	16	21	122455	295857	48	20
pw-tank	9/13	9	13	127383	85165	211	142
rovers	6/6	6	6	796370	652834	122	10
satellite	6/7	7	6	6287	437238	5	13
schedule-strips	23/50	49	24	3932	152	15	713
tpp	6/6	6	6	1008242	157840	108	2
trucks	6/7	7	6	249518	4586761	198	80
zeno-travel	9/11	9	11	6658	36030	9	1
total	232/292	273	251	107701	457336	44	101

Table 2: Comparison of h_{LA} and *FA*-10^4 across domains. N^+ and N^1 denote the number of tasks within each domain solved by both and any of the two planners, respectively. *solved* denotes the number of tasks solved by the planner, *nodes* and *time* denote average number of expanded nodes and time averaged over tasks solved by both planners. The last line summarizes the respective parameters over all the domains, with nodes and time averages being over all the tasks.

Figure 1 we compare the heuristic values provided by h_{LA} and *FA*-10^4 to the initial states of *all* tasks in BLOCKSWORLD, LOGISTICS, DEPOTS, and SATELLITE. (For many of these tasks, the cost of the optimal solution is currently not even known.) It is not hard to see from the plots that *FA*-10^4 is more accurate on the initial states with smaller goal distances; these typically correspond to smaller tasks, and for these, the abstraction of *FA*-10^4 usually induces perfect or close to perfect estimates. However, as the goal distance grows, h_{LA} quickly takes over, and substantially dominates *FA*-10^4 on the larger tasks. Hence, the scalability of h_{LA}'s accuracy appears to be excellent, at least in comparison to the (in itself, very accurate) flexible-abstraction heuristic of *FA*-10^4.

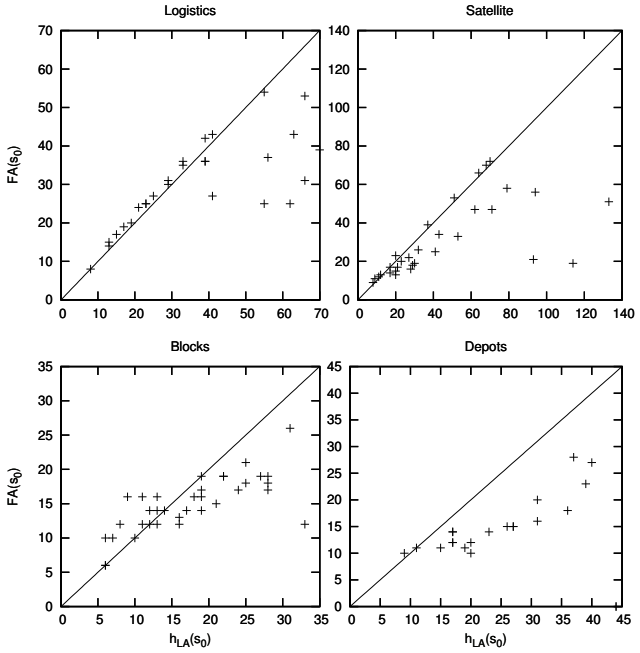

Figure 1: Estimates of h_{LA} and *FA*-10^4 on the initial states of all tasks from four domains. Each point represents the initial state s_0 of a single task, with its x and y coordinates denoting the estimates provided to s_0 by h_{LA} and *FA*-10^4, respectively.

Acknowledgments

The work of both authors is partly supported by the Israel Science Foundation (ISF) grant 890015 and C. Wellner Research Fund. The authors wish to thank Silvia Richter for her assistance with LAMA, and Shaul Markovitch and Malte Helmert for many useful discussions.

References

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

[Bonet and Geffner, 2001] B. Bonet and H. Geffner. Planning as heuristic search. *AIJ*, 129(1–2):5–33, 2001.

[Dechter and Pearl, 1985] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of A^*. *J. ACM*, 32(3):505–536, 1985.

[Helmert *et al.*, 2007] M. Helmert, P. Haslum, and J. Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, pages 176–183, 2007.

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

[Helmert, 2009] Malte Helmert. Concise finite-domain representations for pddl planning tasks. *AIJ*, 173(5-6):503–535, 2009.

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

[Porteous and Cresswell, 2002] J. Porteous and S. Cresswell. Extending landmarks analysis to reason about resources and repetition. In *PLANSIG*, 2002.

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

[Richter *et al.*, 2008] S. Richter, M. Helmert, and M. Westphal. Landmarks revisited. In *AAAI*, pages 975–982, 2008.

[Vidal and Geffner, 2006] V. Vidal and H. Geffner. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *AIJ*, 170(3):298–335, 2006.

[Zhu and Givan, 2004] L. Zhu and R. Givan. Heuristic planning via roadmap deduction. In *IPC-4*, pages 64–66, 2004.