BJOLP: The Big Joint Optimal Landmarks Planner*

Carmel Domshlak
Technion

Malte Helmert
Albert-Ludwigs-Universität Freiburg

Erez Karpas
Technion

Emil Keyder
Universitat Pompeu Fabra

Silvia Richter
NICTA

Gabriele Röger
Albert-Ludwigs-Universität Freiburg

Jendrik Seipp
Albert-Ludwigs-Universität Freiburg

Matthias Westphal
Albert-Ludwigs-Universität Freiburg

Abstract

BJOLP, The Big Joint Optimal Landmarks Planner uses landmarks to derive an admissible heuristic, which is then used to guide a search for a cost-optimal plan. In this paper we review landmarks and describe how they can be used to derive an admissible heuristic. We conclude with presenting the BJOLP planner.

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, Sebastia, and Hoffmann 2001; Porteous and Cresswell 2002; Hoffmann, Porteous, and Sebastia 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, Sebastia, and Hoffmann 2001). Still, many landmarks can often be efficiently discovered (Hoffmann, Porteous, and Sebastia 2004; Richter and Westphal 2010; Keyder, Richter, and Helmert 2010).

Once discovered, landmarks can be extremely helpful in guiding the search for a plan, as evidenced by the performance of the LAMA planner (Richter and Westphal 2010) in IPC-2008. LAMA uses landmarks to derive a (inadmissible) pseudo-heuristic, used to guide a satisficing heuristic search.

In this paper,we describe 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 also describe a simple best-first search that exploits such heuristics, and finds optimal solutions more efficiently than standard A^*.

*This paper is strongly based upon Karpas and Domshlak (2009) and Keyder, Richter and Helmert (2010).

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 4-tuple $\Pi = \langle V, A, s_0, G\rangle$. $V = \{v_1, \ldots, v_n\}$ 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 iff $\mathsf{pre}(a) \subseteq s$. Applying 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 $G \subseteq s_0[\![\langle a_1, \ldots, a_k\rangle]\!]$, and the cost of the plan is $\Sigma_{i=1}^{k}\mathcal{C}(a_i)$. In cost-optimal planning, we are interested in finding a plan with a minimal cost.

Let $\Pi = \langle V, A, s_0, G\rangle$ 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 also 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 $i-1$.

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. There-

fore, practical methods for finding landmarks are either incomplete or unsound. In what follows we assume access to a sound such procedure; in particular, we combine the landmarks from the RHW landmark generation method (Richter and Westphal 2010) and the h^m landmark generation method (Keyder, Richter, and Helmert 2010).

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.

Admissible Landmark Heuristics

Deriving heuristic estimates from landmarks has been proposed by Richter et al. (2010) 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

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

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 made 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, or is a goal.

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.

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

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

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.

In our planner, 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) to estimate the achievers of simple or disjunctive landmarks (the achievers of a disjunctive landmark ϕ can be simply estimated by the set of all actions achieving some element of ϕ). The possible achievers of a conjunctive landmark ϕ are estimated as the actions which achieve (at least) one of the conjuncts, without deleting any of the other conjuncts, and the first-time possible achievers of a conjunctive landmark ϕ are estimated as the subset of the possible achievers that do not have ϕ as a landmark for achieving their preconditions.

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 $h^*(s)$.*

Proof sketch: Let $P = \langle a_1, a_2, \ldots, a_r\rangle$ be any plan from s to the goal. Let $lm(a) = \{\phi | \phi \in \mathsf{eff}(a) \cap L(s,\pi)\}$ be the set of landmarks that are achieved by action a. Then $\bigcup_{i=1}^r lm(a_i)$ is the set of landmarks that are achieved by P. By the definition of landmarks, P must achieve all the landmarks in $L(s,\pi)$, and therefore $L(s,\pi) \subseteq \bigcup_{i=1}^r lm(a_i)$, and $cost(L(s,\pi)) \le cost(\bigcup_{i=1}^r lm(a_i))$. It is easy to see that $cost(\bigcup_{i=1}^r lm(a_i)) \le \sum_{i=1}^r cost(lm(a_i))$, because some landmarks could potentially be counted twice in the right-hand side expression (i.e. achieved by two or more actions). From the requirements on landmark costs we have that $cost(lm(a_i)) \le \mathcal{C}(a_i)$, and therefore $\sum_{i=1}^r cost(lm(a_i)) \le \sum_{i=1}^r \mathcal{C}(a_i)$. If we combine all of this we get $h_L(s,\pi) = cost(L(s,\pi)) \le cost(\bigcup_{i=1}^r lm(a_i)) \le \sum_{i=1}^r cost(lm(a_i)) \le \sum_{i=1}^r \mathcal{C}(a_i) = \mathcal{C}(P)$ ∎

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 $1 \le i \le k$, 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 compiling Eq. 2 into strictly linear constraints, and solving the 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 $h_L(s, \pi) = k/2 + 0.5$.

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.

Although it is possible to discover action landmarks in a pre-processing phase (a sufficient and efficiently testable condition for a being an action landmark is that a relaxed planning task without a is not solvable), The BJOLP planner discovers action landmarks dynamically during uniform cost-partitioning as follows: whenever $|\mathsf{ach}(\phi|s, \pi)| = 1$ (that is, there is only one achiever of ϕ), then that single achiever, which we denote a, is an action landmark. Since a must be used to achieve ϕ, it makes no sense to divide its cost between other landmarks it might possibly achieve. Therefore we assign the full cost of a to ϕ (that is, $cost(a, \phi) = \mathcal{C}(a)$), and assign 0 cost from a to its other effects (that is, $cost(a, \phi') = 0$ for $\phi \neq \phi' \in L(a|s, \pi)$). This allows us to improve upon "naive" uniform cost-partitioning. We call the heuristic resulting from the use of action landmarks h_{LA}. Clearly h_{LA} is still admissible.

We remark that this dynamic action landmark discovery was not implemented originally in Karpas and Domshlak (2009), but was added later in Keyder, Richter, and Helmert (2010).

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 2 shows that such a set of paths can be much more informative than any of its individual components.

Proposition 2 *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. Then for each path $\pi \in \mathcal{P}$, π_g achieves all landmarks in $L \setminus \mathsf{Accepted}(s, \pi)$.*

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.

Proposition 2 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

$$L(s, \mathcal{P}) = (L \setminus (\mathsf{Accepted}(s, \mathcal{P})) \cup \mathsf{ReqAgain}(s, \mathcal{P}) \quad (3)$$

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, by replacing $L(s, \pi)$ with $L(s, \mathcal{P})$.

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 path and a newly discovered path from s_0 to s, respectively, then we may have $h(s, \pi') > h(s, \pi)$. That is, a newly 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 s's heuristic value is marked as "dirty". 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 set $L(s, \mathcal{P}(s))$ as in Eq. 3. When a new path π to s is discovered (and extends $\mathcal{P}(s)$), the landmarks are incrementally updated to $L(s, \mathcal{P}(s) \cup \{\pi\})$ by exploiting the monotonicity of the intersection set operator.

When a state s is removed from the open list for expansion, before actually performing the expansion, *LM-A** first checks whether the s's heuristic is marked as "dirty" (which happens when new paths to s have been discovered between the time s was inserted into the open list, and the time it is remove from the open list). If s's h-value is dirty, we reevaluate $h(s)$ (using the new information), and if the new heuristic value is higher than the previous heuristic value, we reinsert s into the open list with the new h-value. Note that both the old and new h-values are admissible, and so if the new h-value is lower (which could happen when using uniform cost-partitioning), admissibility is maintained. If the s is not "dirty", or if the newly computed h-value is not higher than the old value, then s is expanded as usual. *LM-A** is described in pseudo-code in Figure 1.

Implementation

We have implemented h_L, h_{LA} and *LM-A** on top of the Fast Downward planning system. The BJOLP planner uses *LM-A** with the h_{LA} heuristic (using uniform cost-partitioning and the new dynamic action-landmark discovery).

As mentioned before, the landmarks we use for BJOLP are obtained by combining the landmarks discovered by the RHW method (Richter and Westphal 2010) and the h^m landmarks (Keyder, Richter, and Helmert 2010) with $m = 1$. First the entire landmarks graph is generated by each of these discovery methods. Then, the landmarks and orderings are merged, and dominated (in the sense of logical implication) landmarks are discarded. For example, if one method discovers landmark ϕ and the other discovers landmark $\phi \vee \phi'$, then $\phi \vee \phi'$ will be discarded (along with all its orderings). Dominated orderings are also eliminated by logical implication (remember that every greedy-necessary ordering is a natural ordering, but not vice versa).

Finally, this version of BJOLP uses a much more efficient implementation of the way landmark information for each state is stored. While previous versions used a set (of the C++ standard template libraries) to store, for each state, the set of accepted landmarks, BJOLP uses a boolean vector, which uses one bit per landmark. This speeds BJOLP up considerably over previous versions and dramatically reduces its memory footprint.

LM-A*

1. Put the start node s on a list called *OPEN* of unexpanded nodes. Assign g(s) = 0.
2. If *OPEN* is empty, exit with failure; no solution exists.
3. Remove from *OPEN* a node n at which f=g+h is minimum. Break ties in favor of low h (although ties can be broken arbitrarily, as long as goal nodes are favored).
4. If n is a goal node, exit successfully with the solution obtained by tracing back the path along the pointers from n to s (pointers are assigned in steps 7 and 8).
5. If n is marked as dirty, calculate $h(n)$. Else, goto step 7.
6. Compare the newly computed $h(n)$ with that previously assigned to n. If the new value is greater, substitute it for the old, update $f(n)$, move n back to *OPEN*, and goto step 2.
7. Place n on a list called *CLOSED* to be used for expanded nodes, and expand node n, generating all its successors with pointers back to n.
8. For every successor n' of n:
   (a) Store the current path to n' (through n).
   (b) Calculate $g(n')$ = $g(n)$ + $\mathcal{C}(a)$, where a is the action leading from n to n'.
   (c) Calculate $h(n')$.
   (d) If n' is neither in *OPEN* nor in *CLOSED*, then add it to *OPEN*. Assign the newly computed $g(n')$ and $h(n')$ to node n'.
   (e) If n' already resides in *OPEN* or *CLOSED*:
      i. Store the new path to n' and mark n' as dirty.
      ii. Compare the newly computed $g(n')$ with that previously assigned to n'. If the new value is lower, substitute it for the old (n' now points back to n instead of to its predecessor), and update $f(n')$. Move the matching node n' back to *OPEN* if it resided in *CLOSED*.
9. Go to step 2.

Figure 1: Pseudo-code of *LM-A**

References

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

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

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. 173:503–535.

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

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*, 1728–1733.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for and/or graphs. In *ECAI*, 335–340.

Porteous, J., and Cresswell, S. 2002. Extending landmarks

analysis to reason about resources and repetition. In *PLAN-SIG*.

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

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

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

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