

Heuristic Planning via Roadmap Deduction

Lin Zhu and Robert Givan*

Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907 USA
{l Zhu, givan}@purdue.edu

Abstract

Porteous et al. (2001) introduced the concept of “planning landmarks”—propositions that must be true at some point during the execution of every successful plan. We define “relaxed landmarks,” a subset of the planning landmarks, and give a sound and complete algorithm for computing relaxed landmarks. All the landmarks computed by the previous method are relaxed landmarks, but that method was significantly incomplete for finding relaxed landmarks. We additionally discriminate between useful “causal” landmarks and misleading “non-causal” landmarks, and our method easily omits the latter. We then present a novel method for partially ordering landmarks into “landmark roadmaps”, where two ordered landmarks are present in the given order in every successful plan execution. Finally, we give an efficient means of extending FF’s heuristic to leverage a landmark roadmap by weighting the components of the relaxed plan. The SCHEME variant of FF using this heuristic, ROADMAPPER, works on the non-temporal ADL versions of the IPC4.

Our ROADMAPPER planner is a variant of FF where the heuristic is significantly more complex and derived from a partially ordered set of landmarks. In what follows, we formalize, motivate, and define the heuristic used in ROADMAPPER.

Background

We refer to (McAllester & Rosenblitt 1991) as SNLP and generally follow and adapt it for notation regarding STRIPS planning and partial order planning.

Strips Planning. Let X be a finite set of propositions. A state S is a finite subset of X . An action o is a triple $o = \langle \text{PRE}(o), \text{ADD}(o), \text{DEL}(o) \rangle$ where $\text{PRE}(o)$ are the *preconditions*, $\text{ADD}(o)$ is the *add list* and $\text{DEL}(o)$ is the *delete list*, each being a set of propositions. The result $\text{RESULT}(S, (o_1, \dots, o_n))$ of applying an action sequence (o_1, \dots, o_n) to a state S is given by $\text{RESULT}(\text{RESULT}(S, (o_1, \dots, o_{n-1})), (o_n))$, where for n equals 1 the result is undefined unless $\text{PRE}(o_1) \subseteq S$, and $(S \cup \text{ADD}(o_1)) - \text{DEL}(o_1)$, otherwise.

*We are grateful to Alan Fern and Matthew Greig for useful discussions.

Copyright © 2004, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

A *planning task* P is a set of actions containing actions START and FINISH, where $\text{PRE}(\text{START})$, $\text{DEL}(\text{FINISH})$, and $\text{ADD}(\text{FINISH})$ are all empty. We refer to $\text{PRE}(\text{FINISH})$ as the *goal region* and $\text{ADD}(\text{START})$ as the *initial state*. We also consider the *relaxed planning task* P^R (which ignores delete effects) given by $\{(\text{PRE}(o), \text{ADD}(o), \emptyset) \mid (\text{PRE}(o), \text{ADD}(o), \text{DEL}(o)) \in P\}$.

A *linear solution* for a task P is an ordered action sequence \vec{o} , beginning with START, ending with FINISH such that $\text{RESULT}(\emptyset, \vec{o})$ is defined.

Partial Order Planning To allow multiple occurrences of the same action or the same proposition within our nonlinear plans, we introduce finite sets of *step names* and *fact names*, respectively. Each plan includes a *symbol table* mapping step names to actions and fact names to propositions¹. We use step names and fact names as actions or propositions, respectively, assuming an implicit look-up of the corresponding action or proposition in the appropriate symbol table. Note that naming for facts is needed so that we can later allow a fact to be a landmark more than one time, indicating that that fact must be added multiple times in any successful plan.

A *nonlinear plan*, or *plan* for short, is a pair $\langle \Sigma, \leq \rangle$ of a symbol table Σ , and a partial order² \leq on names³ (step names and fact names) in Σ . We write $x < y$ to abbreviate $x \leq y \wedge x \neq y$. The *length* of the plan is the number of step symbols in Σ .

SNLP introduced the concept of *causal links* to help developing a systematic, sound and complete search algorithm. Causal links can be inferred from our representation. A *causal link* is a triple $\langle s, p, w \rangle$, written as $s \xrightarrow{p} w$, where s and w are step names, and p is a proposition⁴ in $\text{ADD}(s) \cap \text{PRE}(w)$, such that $s < x < w$ for some x mapped to p , and that either $v < s$ or $w < v$ for every step name v in the set $\{y \in \Sigma - X \mid p \in \text{DEL}(y)\} - \{s, w\}$. Note, there can be two different causal links $s_1 \xrightarrow{p} w$ and $s_2 \xrightarrow{p} w$ for

¹This is different from the original SNLP paper, where the symbol table contained only step names.

²For our purpose, a partial order is a reflexive, transitive, anti-symmetric relation, viewed as a set of orders $x < y$.

³Again, ordering on fact names is necessary to allow proposition landmarks.

⁴Note, not a fact name.

the same step name w and the same proposition p . This is not the case for SNLP.

A bijection σ on names is called a *renaming*. We extend such renamings naturally to bijections on complex objects containing names (such as plans), in each case renaming the names appearing within. We say a nonlinear plan $\langle \Sigma', \leq' \rangle$ *refines* $\langle \Sigma, \leq \rangle$ whenever, for some renaming σ , $\sigma(\Sigma) \subseteq \Sigma'$ and $\sigma(\leq) \subseteq \leq'$. If either of the containment is proper, the refinement is called *strict*.

A nonlinear plan is called *complete* if FINISH is named by Σ , and for every step name $v \in \Sigma$ and every proposition p in $\text{PRE}(v)$, there is *at least one* causal link $s \xrightarrow{p} v$. Later in this paper, we generally restrict our attention to nonlinear plans that are complete.

A *relaxed (nonlinear) plan* for P is a nonlinear plan for the corresponding relaxed task P^R . Obviously every plan for P is a relaxed plan for P . A relaxed plan is called *non-redundant* if any proposition or action is named at most once. Any relaxed plan refines some non-redundant relaxed plan.

Landmarks and Roadmaps

Definition 1 A nonlinear plan $\langle \Sigma, \leq \rangle$ is a roadmap for planning task P if every complete nonlinear plan for P refines $\langle \Sigma, \leq \rangle$.

We call actions or propositions appearing in Σ for a roadmap *causal landmarks*. Causal landmarks that are propositions are *landmarks* in the sense of Porteous et al. 2001: the planning problem cannot be solved if the actions adding such a proposition are removed. However, not every landmark is a causal landmark: some landmarks are just “incidental” effects of the action that adds them. Consider a problem where the agent must travel in the rain to solve the problem. “Getting wet” will be a non-causal landmark, as it is a necessary effect of an essential action. Setting “getting wet” as a subgoal would be misleading. Thus we consider non-causal landmarks to be misleading and inappropriate as subgoals for the planning task.

Porteous et al. showed the problem of finding landmarks for a planning task to be PSPACE-hard. The proof can be easily extended here.

Theorem 1 *The problem of deciding whether a proposition or an action is a causal landmark is PSPACE-hard.*

Therefore deducing any nontrivial roadmap is difficult as well. Here we will concentrate on a tractable subset of roadmaps.

Definition 2 A relaxed roadmap for planning task P is a roadmap for the corresponding relaxed planning task P^R .

We call actions or propositions appearing in Σ for a relaxed roadmap *relaxed causal landmarks*. Every relaxed roadmap is a roadmap, and therefore every relaxed causal landmark is a causal landmark.

To compute relaxed roadmaps, we first assume a base algorithm $\text{A_RELAXED_PLAN}(P)$ that finds some *non-redundant* plan for the relaxed planning task P^R , if there exists one, and returns FALSE otherwise. The heuristic

computation in FF contains an efficient implementation of A_RELAXED_PLAN , which empirically often returns a good approximation of the shortest relaxed plan.

The relaxed roadmap is computed in a generate-and-test way. We first call A_RELAXED_PLAN to generate a relaxed plan $\langle \Sigma, \leq \rangle$. Since by definition any relaxed roadmap is refined by $\langle \Sigma, \leq \rangle$, we select a subset of Σ and a subset of \leq to get a relaxed roadmap, by the test phase described below.

Again, the function A_RELAXED_PLAN is used to test whether a proposition or an action is a relaxed causal landmark. To do so, we first define the reduced planning problem $P_{\bar{x}}$, intended to be solvable exactly when x is *not* a causal landmark for P . If the landmark x which we want to test is a proposition, $P_{\bar{x}}$ is $\{o_{\bar{x}} = \langle \text{PRE}(o), \text{ADD}(o) - \{x\}, \text{DEL}(o) \rangle \mid o \in P\}$; otherwise $P_{\bar{x}} = P - \{x\}$. We know x is a relaxed causal landmark for P if and only if $\text{A_RELAXED_PLAN}(P_{\bar{x}})$ returns FALSE.⁵

Further, we can use the above method to verify $x < y$ for a (relaxed) roadmap. To do so, we define $P_{\rightarrow y}$, the subproblem of P with goal of reaching y . $P_{\rightarrow y}$ is the same as P except that FINISH is replaced with $\langle \text{PRE}(y), \emptyset, \emptyset \rangle$ if y is a step name, and $\langle \{y\}, \emptyset, \emptyset \rangle$ otherwise. For every pair of causal landmarks x and y , we know that $x < y$ appears in a relaxed roadmap if and only if x is a relaxed causal landmark of $P_{\rightarrow y}$, i.e., $\text{A_RELAXED_PLAN}(P_{\bar{x}, \rightarrow y})$ returns FALSE.

In the algorithm below, we use R^* to denote the reflexive transitive closure of a relation R .

Algorithm 1 $\text{RELAXED_ROADMAP}(P)$

Input: A planning task

```

 $\langle \Sigma_c, \leq_c \rangle \leftarrow \text{A\_RELAXED\_PLAN}(P)$ 
 $\Sigma_r \leftarrow \{x \in \Sigma_c \mid$ 
  not  $\text{A\_RELAXED\_PLAN}(P_{\bar{x}})\}$ 
 $\leq_r \leftarrow \{(x, y) \in \leq_c \cap \Sigma_r \times \Sigma_r \mid$ 
  not  $\text{A\_RELAXED\_PLAN}(P_{\bar{x}, \rightarrow y})\}$ 
return  $\langle \Sigma_r, \leq_r^* \rangle$ 

```

The bound on the times of calling $\text{A_RELAXED_PLAN}(P)$ is $O(n + m^2)$, where n is the total number of actions and propositions, and m is the total number of relaxed landmarks. In practice, m is typically much smaller than n . There are several ways to make this computation more efficient which are omitted here.

Theorem 2 *The output of $\text{RELAXED_ROADMAP}(P)$*

Soundness is a relaxed roadmap for P , and

Completeness refines every relaxed roadmap for P .

We note here that the above method is not the only to deduce roadmaps. Roadmaps generated in other ways can be incorporated.

Weighted Relaxed Plan Length as a Heuristic

A roadmap intuitively contains important ordered subgoals of a planning problem. Porteous et al. 2001 proposed to use it to sub-divide planning problems into smaller, easier

⁵In contrast, Porteous et al. define $P_{\bar{x}}$ as $P - \{o \mid x \in \text{ADD}(o)\}$ if x is a proposition. This can be used to test landmarks, but cannot distinguish causal landmarks.

pieces, and then use a base planner to solve them one by one. This methodology, however, ignores the interactions between solving subgoals. In particular, the base planner may solve a subgoal in a way so that later subgoals become hard or impossible to solve.

Another way to utilize landmarks is to simply use the number of landmarks as a heuristic guiding forward search. Empirical results show it is effective on some domains at a high level (Zhu & Givan 2003). However, this heuristic is not informative on how to solve the subgoals. It is only when a subgoal is solved, by blind search, that the heuristic decreases by one.

We introduce a novel usage of roadmaps below. We use roadmaps to weight the components of a successful heuristic, emphasizing solving one subgoal, while keeping an eye on the solution of other subgoals.

The success of FF (Hoffmann & Nebel 2001) mainly comes from its efficient and accurate heuristic, and its unique search strategy, *enforced hill-climbing*, that is incomplete but often very fast⁶. Unlike pure hill-climbing, which iteratively selects single actions with the best one-step-look-ahead heuristic value and often has difficulty with local minima and plateaus, enforced hill-climbing iteratively uses breadth-first search to find *action sequences* that lead to states with heuristic values that are strictly better than the current state.

Here, we discuss FF's heuristic and our way to improve its quality. We know that an ideal search heuristic would be the optimal length of a complete plan. Since this heuristic is not tractably computable, FF approximates it by two relaxations. In the following discussion, we denote the set of plans for task P by $PLANS(P)$, and the set of relaxed plans by $RELAXED_PLANS(P)$. Obviously $PLANS(P) \subseteq RELAXED_PLANS(P)$.

First, FF considers the relatively easier problem of computing $RELAXED_PLANS(P)$, and approximates (and lower bounds) the optimal length among $PLANS(P)$ by the optimal length among $RELAXED_PLANS(P)$. Empirical (Hoffmann 2001) and theoretical (Hoffmann 2002) results show that optimal relaxed plan length (applied with enforced hill-climbing) is a good heuristic for a large variety of planning domains, and often leads to polynomial search complexity.

Second, since it's still difficult to compute the optimal relaxed plan, it extracts one relaxed plan to get an approximation of the optimal relaxed plan length, utilizing various heuristic considerations to encourage near-optimality. Empirical results (Hoffmann 2001) show that the length of the relaxed plan extracted this way is often a good approximation of the optimal relaxed plan length. FF uses this length as its heuristic.

We extend the relaxed-plan-length heuristic by assigning weights to its components. Among all the landmarks that have no other landmark ordered before them in the roadmap, we choose one achievable by the shortest relaxed plan. The heuristic of the global problem is the weighted sum of relaxed plan lengths of all landmarks. The chosen landmark

gets weight f , and all the others get weight 1. We generally consider f that is greater than 1. The greater f is, the more aggressive the planner is on solving one subgoal, and the more oblivious it is to the difficulty of other subgoals.

In theory and in practice, the computation of this heuristic should add only trivial burden to that of FF, besides the one-time cost of computing roadmap.

We then utilize this heuristic in a similar way to FF, and apply the resulting planner, ROADMAPPER, to non-temporal ADL versions of the fourth international planning competition. Our implementation is fully written in SCHEME, a dialect of LISP.

References

- Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.
- Hoffmann, J. 2001. Local search topology in planning benchmarks: An empirical analysis. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, 453–458.
- Hoffmann, J. 2002. Local search topology in planning benchmarks: A theoretical analysis. In *Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-02)*. 379–387.
- McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, 634–639. Anaheim, California, USA: AAAI Press/MIT Press.
- Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *Recent Advances in AI Planning. 6th European Conference on Planning (ECP'01)*, 37–48.
- Zhu, L., and Givan, R. 2003. Landmark Extraction via Planning Graph Propagation. In *Printed Notes of ICAPS'03 Doctoral Consortium*. Trento, Italy.

⁶In the rare case the *enforced hill-climbing* fails, FF resorts to an expensive but complete search.