Generalization of the Landmark Graph as a Planning Problem

Vidal Alcázar
Universidad Carlos III de Madrid
Avenida de la Universidad, 30
28911 Leganés, Spain
valcazar@inf.uc3m.es

Abstract

Landmarks are logical formulæ over sets of propositions or actions that must be satisfied at some point in a planning task. The landmark graph, a proposed representation of the set of landmarks and their interactions, is built using the landmarks of the task and ordering relations between them. A formalization of the landmark graph in terms of a planning task has not been proposed yet, which makes it difficult to determine the significance of the landmark graph with respect to the original planning problem. In this work we propose a generalization of the landmark graph as an abstraction of the original problem and analyze its characteristics.

Introduction

Landmarks are currently one of the most important lines of research in automated planning, exemplified for instance by the success of landmark-based planners like LAMA (Richter and Westphal 2010). They were initially defined as disjunctive sets of propositions that had to be made true at some time in every solution plan to a problem (Hoffmann, Porteous, and Sebastia 2004), and later on its definition was extended to include both action landmarks (Richter and Westphal 2010) and conjunctive sets of propositions (Keyder, Richter, and Helmert 2010). Landmarks were also formalized in a framework that relates them to abstractions and critical paths (Helmert and Domshlak 2009), giving them a stronger theoretical background.

Although finding the complete set of landmarks is PSPACE-complete (Hoffmann, Porteous, and Sebastia 2004) and thus not tractable in most cases, current methods can efficiently compute a subset of the landmarks using a delete-relaxation representation of the problem. Partial orders between proposition landmarks can be obtained with these techniques too, which can be used to build the landmark graph. The landmark graph is an important part of many of the techniques that exploit landmarks, as the interactions and orders between landmarks are often as important as the landmarks themselves.

Despite the landmark graph providing important information, it still has some limitations in its current form. First, an informative total order may not be straightforwardly deduced from the set of partial orders, which is critical in applications like factored planning; second, landmarks may have to be achieved several times in every solution plan, a fact that is not taken into account due to the lack of negative interactions between landmarks other than the causal orderings (Hoffmann, Porteous, and Sebastia 2004); third, the exploitation of the cycles in the graph is still unclear and so current techniques usually just remove them.

To address these shortcomings, in this work a generalization of the landmark graph as a planning problem is proposed. The main motivation is creating an automatic landmark-based abstraction whose solution can capture the causal structure of the problem in a more accurate way than the landmark graph. The idea presented in this work is using the landmarks to build the set of propositions of the problem and transforming the achievers of those landmarks into the actions of the abstraction, adding information from landmark orderings and mutexes. This abstraction is a planning task itself, so it can be solved just like any other planning problem. Besides, it has several properties of its own that can be exploited when solving the original task.

Background

Automated planning is the task of finding a solution plan, composed by an ordered sequence of actions, so a set of goals can be achieved by applying the actions in the plan from the initial state. In this work only propositional planning is considered. A planning problem is a tuple $P=(S,A,I,G)$ where S is a set of propositions, A is the set of grounded actions instantiated from the operators of the domain, $I \subseteq S$ is the initial state and $G \subseteq S$ the set of goal propositions. Actions are applicable when their preconditions are satisfied. The effects of an action make propositions true or false, which is commonly known as *adding* and *deleting* the propositions respectively. Thus, an action is defined as a triple $\{pre(a), add(a), del(a)\}$ in which $a \in A$ and $pre(a), add(a), del(a) \subseteq S$. Actions can have an associated cost; in this work only non-negative cost actions are used.

A landmark is a logical formula over either S (proposition landmark) or A (action landmark). Every solution plan must satisfy every action landmark. For each proposition landmark, at least one state in every sequence of states generated by a solution plan must satisfy it. L is the set of discovered landmarks of the problem.

The achiever of a proposition landmark $l \in S$ is an action $a \in A$ such that $l \in add(a)$. Informally, an achiever is an action that makes a given landmark true, or *adds* it. A late

achiever $a \in A$ is a regular achiever of a landmark $l \in S$ with the special condition that, in every plan executed from I, it is not possible for a to appear before l has been achieved at least once by some other action $a' \in A$.

Orderings between landmarks are relations between two proposition landmarks that represent the partial order in which they must be achieved. The landmark graph is a directed graph composed by the proposition landmarks of a problem and the orderings between them (Hoffmann, Porteous, and Sebastia 2004). There are the following orderings:

- Natural ordering: A proposition *a is naturally ordered before b* ($a <_{nat} b$) if *a* must be true at some time before *b* is achieved
- Necessary ordering: A proposition *a is necessarily ordered before b* ($a <_{n} b$) if *a* must be true one step before *b* is achieved
- Greedy-necessary ordering: A proposition *a is greedy-necessarily ordered before b* ($a <_{gn} b$) if *a* must be true one step before *b* when *b* is first achieved

There are also reasonable orderings between landmarks, but these orderings are unsound, that is, they do not have to be respected in every solution plan. Therefore, reasonable orderings will not be considered in this work.

Landmark Counting Heuristics

The admissible landmark counting heuristic h_{LA} formulated by Karpas and Domshlak (2009) is an admissible estimation of the distance to the goal computed by adding the cost of achieving the propositional (disjunctive) landmarks that are still needed. These landmarks may be landmarks that have not been achieved yet or that are required again. There are two versions of h_{LA}. The simplest version splits uniformly the cost of every achiever amongst the landmarks that they achieve (h^{uni}_{LA}), so the cost of each landmark is the minimum of these "split costs". The more complex version solves a Linear Programing problem per state that yields the maximum cost of achieving the required landmarks by selecting which achiever contributes to the cost of which landmark and by how much while keeping the estimate admissible (h^{opt}_{LA}).

h_{LA} depends on which landmarks are true in a given state $s \in S$ and on which landmarks have already been achieved. The latter depends on the path or paths that led from I to s, so this information must also be encoded. This makes h_{LA} a *path-dependent* heuristic, which means that h_{LA} is inconsistent and may yield different values for the same state s.

Generalization of the Landmark Graph as a Planning Problem

In this section the process of generating a new problem from the original problem using the information of the landmark graph is described.

Definition of the New Problem P_{abs}

In this section we will take into account only single proposition landmarks; other cases will be analyzed in a subsequent subsection. The generalization of the landmark graph as a planning problem is a tuple $P_{abs} = (S_{abs}, A_{abs}, I_{abs}, G_{abs})$ where S_{abs} is a set of propositions derived from the discovered proposition landmarks, A_{abs} is a set of grounded actions derived from the achievers of the propositions contained in S_{abs}, $I_{abs} \subseteq S_{abs}$ is the value of S_{abs} in the current state and $G_{abs} \subseteq S_{abs}$ the set of goal propositions.

The set of propositions S_{abs} is formed by two propositions per landmark $l \in L$, one proposition l_{abs}indicating whether the landmark is true or not and another proposition $achieved(l_{abs})$ indicating whether the landmark l_{abs} has been previously achieved.[1] All the propositions of the original problem $S \setminus L$ that are not landmarks are discarded. The set of actions A_{abs} are the actions in A that are landmark achievers, that is, at least one landmark proposition appears in its *add* effects. Formally, if $a \in A$ and $add(a) \cup L \neq \emptyset$, then a is used to derive new actions in A_{abs}. A priori, a single new action per achiever in A is created. However, due to the possibility of having disjunctive preconditions in the new actions, some actions in A_{abs} may have to be split into several ones. All the actions in A that do not add at least one landmark proposition are ignored. The goal propositions G_{abs} are the goal propositions G of the original problem, since a goal proposition is a landmark by definition. Additionally, the propositions that encode whether the original goal propositions have been achieved can also be added to G_{abs}, since a proposition being true necessarily implies that it has been achieved. Every goal state s must ensure that $\forall g \in G_{abs} : s(g) = \{true\}$.

This tuple is analogous to that of a regular planning task, which means that the new problem P_{abs} has all the properties of a regular planning problem. Besides, the resulting P_{abs} is an abstraction of the original problem P. The function α that maps a state $s \subseteq S$ to $\alpha(s) \subseteq S_{abs}$ consists of determining which landmarks are true and which landmarks have been previously achieved in s. Note that this depends not only on s but also on the path that led from I to s, so the information about which landmarks have already been achieved must be stored in a similar fashion as when computing h_{LA}. The transitions are defined by the relationship between A_{abs} and A, described in the following subsections.

Preconditions of the New Actions

Actions belonging to A_{abs} are applicable when the landmarks appearing in their preconditions have the required value. Three different cases define the preconditions of the actions:

- Preconditions derived from the orderings between the landmarks.

[1] A more concise representation using multi-valued state variables could be obtained in two ways: first, the variables could take the values of *true*, *false* or *false but previously achieved* instead of using two different variables to avoid redundancy, although this can lead to actions with disjunctive preconditions; second, landmarks belonging to the same variable in the original problem or that are otherwise mutually exclusive could be grouped in a single variable to indicate whether they are true or not.

- Preconditions of the achievers in the original problem that are landmarks.
- Preconditions obtained from actions labeled as *late achievers*.

Causal relationships between the landmarks are encoded by the orderings of the landmark graph. To represent this fact in the new task P_{abs}, new preconditions are added to the actions in A_{abs}. These preconditions enforce the partial orders between landmarks without hindering the computation of a valid total order. Each type of ordering implies a different set of new preconditions. In particular, the correspondence is as follows:

- Natural ordering: Every achiever of a given landmark l has a precondition for every natural ordering which represents that the landmarks naturally ordered before l must have been previously achieved. Formally, if $l \in add(a)$, $\forall l' <_{nat} l : achieved(l') \in pre(a)$.
- Necessary ordering: Every achiever of a given landmark l has a precondition for every necessary ordering which represents that the landmarks necessarily ordered before l must be true. Formally, if $l \in add(a)$, $\forall l' <_{n} l : l' \in pre(a)$.
- Greedy-necessary ordering: Every achiever of a given landmark l has a precondition for every necessary ordering which represents that the landmarks greedy necessarily ordered before l must be true or that l has been previously achieved. Formally, if $l \in add(a)$, $\forall l' <_{nat} l : (achieved(l) \vee l') \in pre(a)$.

Greedy-necessary orderings add a disjunctive precondition. Most planners do not support disjunctive preconditions, so actions with preconditions derived from greedy-necessary orderings must be split into several actions. In theory this can lead to having 2^n new actions per action that achieves landmarks with greedy-necessary orders, where n is the number of greedy-necessary orderings. However, due to the dominance that often appears between actions (described in a subsequent subsection), the actual number of new actions is at most $2^{n'}$ actions, where n' is the number of landmarks with greedy-necessary orderings achieved by the action. Although still exponential, in practice it seldom happens that an action achieves more than 2 such landmarks, which explains why there are no exponential blow-ups of the size of P_{abs} in the current benchmarks.

Preconditions can also be derived from the landmarks that appeared in the preconditions of the actions in A. Landmarks in the original problem P are kept as part of the new problem P_{abs}. Because of this, occurrences of landmarks in the actions of A must be taken into account when generating the actions in A_{abs}. Regarding the preconditions of every action in A_{abs}, this means that the preconditions of the original actions in A must also appear in the actions in A_{abs} if they are landmarks: if $a \in A$ was used to create $a' \in A_{abs}$, then $pre(a) \setminus L \subseteq pre(a')$. This overlaps with the preconditions obtained from greedy-necessary orderings and strictly dominates those obtained from necessary orderings, hence making the computation of the latter not necessary. This is so is because both types of orders are computed from common preconditions of the achievers of the landmark, already taken into account this way.

On the other hand, not all achievers are considered equal. There are two kinds of achievers: *first achievers*, which can achieve some landmark when it has never been achieved before, and *late achievers*, which can achieve a landmark only if it had been already achieved at some time before. Hence, a late achiever $a' \in A_{abs}$ derived from action $a \in A$ has an additional precondition $achieved(l) \in pre(a')$ per landmark $l \in add(a) \setminus L$ if a is a late achiever of l.

Effects of the New Actions

Effects of the original achievers and relations of mutual exclusivity between propositions (Haslum and Geffner 2000), typically called mutexes, determine the effects of the new actions. The positive effects of the actions, also known as *add* effects, make the propositions true. Similar to the propositions that are landmark in the preconditions of the actions in A, the *add* effects of the actions in A can be straightforwardly encoded in the new actions belonging to A_{abs}. Thus, every landmark proposition added by an action in A appears also as an *add* effect in the actions created from it in A_{abs}, whereas non-landmark propositions added by the action are ignored. Similarly, for every regular *add* effect of a landmark l, an *add* of $achieved(l)$ must be included. In summary, if $a \in A$ was used to derive $a' \in A_{abs}$ and landmark $l \in add(a) \setminus L$ then $l \wedge achieved(l) \in pre(a')$.

Before describing how the delete effects of the new actions are computed, the definition of mutex is given:

Definition 1 *A relation of mutual exclusivity between propositions (mutex) is a set of propositions $M = \{p_1, \ldots, p_m\}$ such that there is no reachable state $s \subseteq S$ such that all the propositions $p \in M$ are true in s.*

Negative effects, also known as *delete* effects, are linked to the notion of *e-delete* (Vidal and Geffner 2005).

Definition 2 *An action e-deletes a proposition if the proposition p must be false in every state after the action is executed.*

This means that if an action *e-deletes* a given proposition *p*, either it explicitly deletes *p* or it is only applicable in states in which *p* is false. There are three cases in which an action *e-deletes* a given proposition *p*: it deletes *p*; it has a set of preconditions mutex with *p* and does not add *p*; and it adds a set of propositions mutex with *p*.

Computing sets of mutexes of size $m > 2$ is usually impractical (Haslum and Geffner 2000), so in this work only sets of size $m = 2$ will be considered. Hence, to fulfill the second and third condition a single proposition mutex with *p* is enough. This way, every landmark that is *e-deleted* by an action $a \in A$ is added to the new action in A_{abs} as a *delete* effect. There is an exception to this rule, though: if an action $a \in A$ *e-deletes p* because it has a precondition mutex with *p* and does not add it (second condition) and that precondition is a landmark, it means that the new action in A_{abs} is only applicable *iff p is false*. In this case *p* will be always false after the execution of the new action in A_{abs}, which means there is no need to explicitly delete it.

Dominance Between the New Actions

Due to the abstraction of non-landmark propositions and the splitting of disjunctive preconditions, it is possible to have actions in A_{abs} that are equivalent to or dominated by other actions. In particular, an action with the same effects and with a cost greater or equal than another action needs not to be included in the definition of the problem as long as its preconditions are a superset of the preconditions of the dominating action. In terms of the formal definition of the planning task P_{abs}, $a \in A_{abs}$ dominates $a' \in A_{abs}$ if $a \neq a' \wedge add(a) = add(a') \wedge del(a) = del(a') \wedge pre(a) \subseteq pre(a') \wedge cost(a) \leq cost(a')$. This can be done after generating all the actions to avoid redundancy and obtain a smaller instance of P_{abs}.

Dominance between actions also explains why splitting actions when using greedy-necessary orderings leads to having fewer additional actions than expected. For example, let's assume that an action $a \in A$ adds a landmark $l \in S$ which is greedy-necessarily ordered after n landmark propositions $p_i \in S$. This means that the new action $a' \in A_{abs}$ will have n disjunctive preconditions of the form $\{achieved(l) \vee p_i\}$. If a' is split in two for every disjunctive precondition, a total of 2^n new actions will be created. However, every time an action is split, either $achieved(l)$ or p_i will be added to the preconditions; if $achieved(l)$ is added and it is already present, no new action is needed, and if p_i is added and $achieved(l)$ is already a precondition, it will be forcibly dominated by another action with $achieved(l)$ as precondition. In the end, only 2 actions are generated per added landmark with greedy-necessary orders, one with $achieved(l)$ as additional precondition and another one with $p_1 \wedge ... \wedge p_n$ as additional preconditions.

Conjunctive Landmarks, Disjunctive Landmarks and Action Landmarks

Landmarks are not restricted to the single proposition case. When generalizing the landmark graph this has to be taken into account. Particular cases are treated in the following way:

- Action landmarks: They can be safely ignored, as their preconditions and effects are landmarks themselves. A possible optimization is allowing only the action landmark as achiever if the landmarks derived from the effects of the action are not ordered after any landmark other than the preconditions of the action landmark.
- Conjunctive landmarks: All the propositions that form the set can be treated as independent landmarks. The achievers of landmarks ordered after a conjunctive landmark will have all the propositions in the set as preconditions.
- Disjunctive landmarks: All the propositions that form the set can be treated as independent landmarks. Only one of the propositions that form the set must have the required value for an action $a \in A_{abs}$ that has the disjunctive landmark as precondition to be applicable. This is done by splitting every such action into several actions, one per proposition in the set, and including that proposition as precondition. This can lead to an exponential number of actions, so a more efficient way of dealing with disjunctive landmarks could improve the compactness of the problem.

Properties of the New Problem

The new planning problem P_{abs} has all the properties of a regular planning problem in a propositional representation. Besides, due to its relationship with the original problem P, P_{abs} has several additional characteristics:

- All the propositions being true and having been achieved are landmarks, unless they were generated from a disjunctive landmark. If this is the case, they will be part of a disjunctive landmark if a landmark discovery method that can find disjunctive landmarks of size equal or greater than the original disjunctive landmarks is used.
- The optimal cost to the goal in the new problem h^*_{abs} is an admissible estimation of the cost to the goal in the original problem h^*, since P_{abs} is an abstraction of P. The proof is as follows: P_{abs} is a projection of P except for the added propositions of the type $achieved(l)$ for $l \in L$. If a problem p' is a projection of p, then $h^*_{p'} \leq h^*_p$. Hence, the only source of inadmissibility can be the propositions of the type $achieved(l)$. However, these propositions are added to enforce the orderings of the landmark graph. These orderings are respected by any optimal solution of P, so the presence of these propositions cannot make that $h^*_{p'} > h^*_p$.
- h^*_{abs}, when being used as a heuristic in a forward search algorithm, is *not* necessarily a consistent estimation of h^*, as I_{abs} depends on the path that led to the current state in P. That is, h^*_{abs} as an admissible estimation of the cost to the goal is a path-dependent heuristic. For the proof, we refer the reader to (Karpas and Domshlak 2009).
- h^*_{abs} dominates the admissible landmark counting heuristic h_{LA} (Karpas and Domshlak 2009) and is not bounded by h^+, the optimal cost of the delete-relaxation version of the original problem. The proof is straightforward: on one hand, if $S = L \subset S_{abs}$ and at least one landmark must be reachieved after being deleted in every optimal plan, then $h^*_{abs} = h^*$ and $h^*_{abs} > h^+ \geq h_{LA}$; on the other hand, if P_{abs} is relaxed by removing orderings and *deletes*, then $h^*_{abs} = h^{opt}_{LA}$. There is no further relaxation other than projecting landmarks away that may make h^*_{abs} lower, so h_{LA} is a lower bound of h^*_{abs} if all the landmarks are considered when building P_{abs}.
- The optimal cost of the delete-relaxation version of the new problem h^+_{abs} still dominates h_{LA}, even with optimal cost partitioning (h^{opt}_{LA}). This is because landmark preconditions of the achievers add additional constraints that may make the cost of the optimal plan go beyond h_{LA}'s value. An example is shown in the delete-free problem appearing in Figure 1: if $I = \{l\}$ and $G = \{l', l''\}$, h_{LA} would assume a cost of 1 for both l' and l'' for a total cost of 2; h^+_{abs} however takes into account that to achieve l' with a cost of 1 l'' must be achieved first, so h^+_{abs} would yield a value of 3.

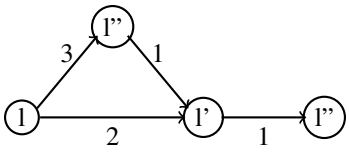

Figure 1: Example in which h^+_{abs} yields a higher value than h_{LA}.

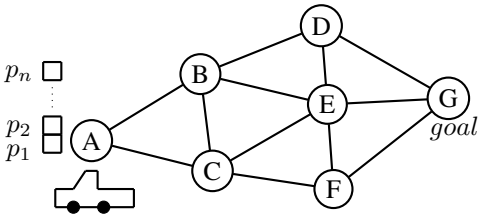

Figure 2: Initial state of the Logistics instance

Example

In this section we show an example of how P_{abs} is constructed. The example is taken from the Logistics domain, although in this case trucks can only carry one package at a time, encoded by the predicate *(empty ?t - truck)*. The initial state is shown in Figure 2. There is a single truck that can move through a graph of different locations. The truck is initially at location *A*, where there is an arbitrary number of packages. The packages can be loaded into the truck if the truck is empty and dropped at another location. The goal is carrying all the packages to the goal location *G*. The single proposition landmarks of this problem are:

- The truck at the initial location: *(at truck A)*
- The truck at the final location: *(at truck G)*
- The truck empty: *(empty truck)*
- Every package p_i at the initial location: *(at p_i A)*
- Every package p_i in the truck: *(in p_i truck)*
- Every package p_i at the final location: *(at p_i G)*

The actions of the planning problem would be the following (note that every time a landmark is made true, the proposition that encodes whether it has been achieved before or not is unconditionally made true as well):

- Move the truck to the initial location *A*: *pre*=$\{$*achieved(at truck A)*$\}$, *add*=$\{$*(at truck A)*$\}$, *del*=$\{$*(at truck G)*$\}$
- Move the truck to the final location *G*: *pre*=$\{$*achieved(at truck A)*$\}$, *add*=$\{$*(at truck G)*$\}$, *del*=$\{$*(at truck A)*$\}$
- Load a package p_i at the initial location *A*: *pre*=$\{$*(at truck A),(at p_i A),(empty truck)*$\}$, *add*=$\{$*(in p_i truck)*$\}$, *del*=$\{$*(at p_i A),(empty truck)*$\}$
- Load a package p_i at the final location *G*: *pre*=$\{$*(at truck G),(at p_i G),(empty truck)*$\}$, *add*=$\{$*(in p_i truck)*$\}$, *del*=$\{$*(at p_i G),(empty truck)*$\}$
- Drop a package p_i at the initial location *A*: *pre*=$\{$*(at truck A),(in p_i truck)*$\}$, *add*=$\{$*(at p_i A),(empty truck)*$\}$, *del*=$\{$*(in p_i truck)*$\}$
- Drop a package p_i at the final location *G*: *pre*=$\{$*(at truck G),(in p_i truck)*$\}$, *add*=$\{$*(at p_i G),(empty truck)*$\}$, *del*=$\{$*(in p_i truck)*$\}$

In this example all the preconditions are taken from the landmark preconditions of the original actions in A with the exception of the *move* actions. Similarly, apart from the *move* actions all the actions in A_{abs} can only be generated from a single action in A. The *move* action that achieves *(at truck A)* can be generated from two actions in A; however, both actions have the same preconditions and effects, so one arbitrarily dominates the other, which explains why there is only one action in A_{abs} that achieves *(at truck A)*. This action has *achieved(at truck A)* as precondition because the original actions in A are *late achievers* of *(at truck A)*. No other precondition appears as no precondition of the original actions is a landmark and *(at truck A)* is not ordered after any other landmark. The same dominance occurs with the action that achieves *(at truck G)*, although in this case the precondition is generated from the natural ordering existing between *(at truck A)* and *(at truck G)*. As a side note, every landmark of the form *achieved(s)* that is true in I_{abs} is a static fact, so it can be safely discarded, as with *achieved(at truck A)* in the example.

The key feature that regular delete-relaxation approaches do not capture in this case is achieving and deleting *(at truck A)* and *(at truck G)* alternatively. In this case, if *n* encodes the number of packages, the cost of the optimal solution in the original problem is $h^* = 8(n-1)+5$ and in the abstraction is $h^*_{abs} = 4(n-1)+3$. On the other hand, the value of both h^+ and h^{lmcut} (Helmert and Domshlak 2009) in the original problem is $h^+ = h^{lmcut} = 2n+3$ and the value of the admissible landmark heuristic with optimal cost partitioning h^{opt}_{LA} is in both the original and the delete-relaxation problem $h^{opt}_{LA} = 2n+1$.

Potential Applications of P_{abs}

Although beyond the scope of this work, it is interesting to analyze the possible applications of the resulting P_{abs} in regards to task solving. The main consideration when exploiting the information provided by P_{abs} is that P_{abs} is a planning problem itself and so it is PSPACE-complete (Bylander 1994). This means that in many cases solving it (optimally or otherwise) may not be tractable. However, the size of P_{abs} is in most cases smaller than the original task P, so if a planner is not able to solve P_{abs} it is highly unlikely that it would be able to solve P.

One of the first applications of landmarks, factored planning (Hoffmann, Porteous, and Sebastia 2004), seems to be a technique that would greatly benefit from this approach. As opposed to using the first layer of unachieved or required again landmarks as a disjunctive intermediate goal, suboptimally solving P_{abs} can be used to obtain a total order of single landmarks, which can be used as subgoals to partition P. Apart from being able to exploit cycles between landmarks

and offering a finer partitioning, this approach also has the advantage of not being subject to arbitrary decisions when dealing with disjunctive subgoals (for instance, when partitioning the Sussman's anomaly, a layered partitioning can lead to either finding the solution with the minimum number of expansions or to exploring the whole search space depending on how ties are broken).

Inspired by the potential of the serialization of the landmark graph, the recent planner PROBE(Lipovetzky and Geffner 2011) builds probes that try to achieve the goal with little to no search by estimating a total order of the landmarks and trying to achieve them in a greedy way. However, the way the probes are built is not based on any theoretical scheme and it has the additional problem of not allowing actions that delete a landmark to be used if that that landmark is a goal or is still needed (which is often the case when cycles between landmarks appear). Solving P_{abs} suboptimally yields a total order that may be more informative and that may capture stronger interactions between landmarks.

The use of P_{abs} to derive heuristics to be used in P has already been mentioned. Nevertheless, a naive approximation like solving P_{abs} at every state to obtain the heuristic estimate would be intractable in most cases, so more synergistic techniques are required. Since P_{abs} is an abstraction of P, hierarchical search algorithms like Hierarchical A* (Holte et al. 1996) can be an interesting alternative. When using HA*, P_{abs} is solved from I_{abs} to obtain a heuristic estimate and the expanded nodes are kept hoping that subsequent queries can be cached, which allows to obtain $h(s)$ with no search. If cache hits happen often enough, there will be a trade-off between the time spent solving P_{abs} and the time saved from computing $h(s)$, which may result in an increased efficiency. Furthermore, P_{abs} can be solved both optimally, superseding the admissible h_{LA}, or suboptimally, superseding LAMA's heuristic (Richter and Westphal 2010).

Lastly, the total order obtained from solving P_{abs} can also be used in combination with other search paradigms. For instance, local search planners like LPG (Gerevini and Serina 2002) are often ill-suited to solving highly sequential domains, particularly if they involve cycles. In this case, the solution to P_{abs} can be used as a seed so a skeleton of the plan is provided and the local search only has to fill the "gaps" between the landmarks.

Experimentation

As any other landmark-based technique, the proposed generalization of the landmark graph is highly dependent on the number and relevance of the landmarks found with current methods. For instance, in some domains only a few landmarks other than the propositions true in I and the goals in G are found, in which case landmark-based techniques perform poorly. Furthermore, one of the main advantages of P_{abs} over most ways of exploiting landmarks resides in the fact that it is able to account for negative interactions such as delete effects and mutexes. In particular, P_{abs} appears to capture the structure of the problem best when cycles are a key part of the domain. If such negative interactions are not representative of the planning task, using P_{abs} may not be more advantageous than using already existing landmark techniques. Because of this, the performance of any technique based on P_{abs} will depend on the number of landmarks and the features of the domain they represent.

Because of the aforementioned reasons, three different cases arise in problems with meaningful landmarks:

- Almost all the propositions are landmarks: the abstraction is very similar to the original problem and thus it is hard to obtain a balanced trade-off between the information it provides and the difficulty of solving P_{abs}. In this case directly solving P is the simplest alternative, though solving P_{abs} may not be necessarily worse, since the information obtained from P_{abs} probably would allow solving P with almost no search.
- There are few necessary orderings and/or landmark preconditions of the achievers: when solving the abstraction the landmarks do not need to be reachieved, so only a rather arbitrary total order of the landmark graph is obtained. Hence, in most cases the cost of the solution is equal or only slightly higher than h_{LA} with optimal cost partitioning. Besides, partitionings of P derived from the obtained total order will not offer much more information than the regular landmark partitioning using disjunctive subgoals.
- There are fairly numerous necessary orderings and/or landmark preconditions of the achievers: in these cases an informative plan is obtained with a cost higher than h_{LA} while still being solvable. As described before this is often the case in which resources or similarly "consumed" landmarks enforce cycles in the landmark graph, cases in which other techniques often do not fare that well.

Knowing this, it may be relatively simple to predict whether P_{abs} will be useful in the actual resolution of the planning task or not. This work is mainly theoretical, and hence P_{abs} was not used as a way of improving the efficiency of a given planner. Nevertheless, in order to assess its usefulness in practice, some experimentation has been done. First, P_{abs} was generated in a broad range of domains from the International Planning Competition so the relative size of P_{abs} compared to P could be estimated. The planner used to ground the instances in both cases was Fast Downward (Helmert 2006), and the chosen measure of size is the number of actions and fluents of each instance after preprocessing, that is, the cardinality of A, A_{abs}, S and S_{abs} in every instance. The reason why the number of actions was chosen is because it gives an idea of the size of the tasks and because generating P_{abs} often requires action splitting, whose impact can be measured by counting the final number of actions. The number of propositions $|S_{abs}|$ can be known a priori just by computing the landmarks of P, although a significant subset of the propositions derived from the landmarks of P may be static in P_{abs}, which means that $|S_{abs}|$ is often smaller than twice the number of landmarks. Table 1 shows the sum of the cardinality of A, A_{abs}, S and S_{abs} of every instance in each domain. All the orderings were used and dominance between actions was enabled. Only propositional landmarks were used, since disjunctive landmarks may easily lead to an exponential increase in the number of actions.

Domain	$\|A\|$	$\|A_{abs}\|$	$\|S\|$	$\|S_{abs}\|$	h^{lmcut}	h^{opt}_{LA}	h^{uni}_{LA}
airport (50)	144963	125067	157592	46706	0.92	1.00	1.00
barman-opt11 (20)	13264	11004	4604	1200	0.64	1.97	2.50
blocks (35)	7490	7434	4826	2549	1.44	1.44	1.44
depot (22)	68894	51392	9423	1934	0.46	1.00	1.00
driverlog (20)	53494	5926	6007	542	0.46	1.00	1.00
elevators-opt08 (30)	18520	7574	3360	607	0.51	1.07	1.07
elevators-opt11 (20)	11450	4619	2097	571	0.50	1.09	1.09
floortile-opt11 (20)	9188	6036	3578	1008	0.76	1.06	1.06
freecell (80)	1071066	663857	23419	13286	2.42	1.03	1.03
grid (5)	38808	18010	3373	185	0.88	1.01	1.01
gripper (20)	3720	1880	2380	960	0.52	1.00	1.00
logistics00 (28)	6972	2380	3429	2409	1.09	1.09	1.09
logistics98 (35)	501186	13491	82687	3203	1.09	1.09	1.09
miconic (150)	189100	125128	13950	19964	1.03	1.03	1.03
mprime (35)	567960	2977	17796	139	0.32	1.00	1.00
mystery (30)	217800	3634	13066	164	0.36	1.01	1.01
nomystery-opt11 (20)	72522	59865	4434	1008	1.15	1.09	1.09
openstacks (30)	213470	212544	10634	11593	1.69	1.00	1.00
parcprinter-opt08 (30)	9066	2933	6139	4159	0.91	1.00	1.00
parcprinter-opt11 (20)	5096	1732	3993	2591	0.91	1.00	1.00
pathways-noneg (30)	40595	7126	13119	2596	0.40	1.00	1.00
pegsol-opt08 (30)	5346	5346	2920	2003	1.00	1.41	1.41
pipesworld-notankage (50)	187388	73935	44594	1649	1.03	1.21	1.21
pipesworld-tankage (50)	1135917	742542	28027	1649	1.06	1.20	1.20
psr-small (50)	14546	11751	2158	572	1.65	1.65	1.65
rovers (40)	231653	45064	29324	2705	0.51	1.04	1.04
satellite (36)	3709130	34163	30479	6192	0.55	1.01	1.01
scanalyzer-opt08 (30)	1145836	733474	4680	1691	1.05	1.07	1.23
scanalyzer-opt11 (20)	631288	420820	3088	1010	1.05	1.07	1.18
sokoban-opt08 (30)	12674	5657	8518	1129	0.59	1.05	1.09
sokoban-opt11 (20)	7166	3332	5306	706	0.56	1.08	1.11
tidybot-opt11 (20)	384018	114737	11476	662	0.59	1.09	1.09
tpp (30)	281351	59833	18807	1564	0.57	1.02	1.02
transport-opt08 (30)	105888	4200	6800	390	0.02	1.00	1.00
transport-opt11 (20)	35216	2360	2886	250	0.02	1.00	1.00
trucks (30)	442262	33896	6964	2065	0.76	1.04	1.04
visitall-opt11 (20)	3520	2688	2516	2259	1.25	1.00	1.00
woodworking-opt08 (30)	27835	22058	5677	2729	0.87	1.02	1.03
woodworking-opt11 (20)	18175	14267	3805	1837	0.90	1.03	1.04
zenotravel (20)	140433	9138	4518	428	0.45	1.01	1.01

Table 1: Comparison between P and P_{abs}: task size and heuristics.

Results show that the relative cardinality of A_{abs} varies greatly from domain to domain. At one extreme, in *pegsol* $|A_{abs}|$ has the same value as $|A|$; at the other, in some transportation domains like *transport* or *zenotravel* $|A_{abs}|$ is comparatively rather small because of the small number of single proposition landmarks found in those domains. This is confirmed by the low value of S_{abs} in those domains in which $|A_{abs}|$ is significantly lower than $|A|$. There are no domains in which $|A_{abs}| > |A|$, although in two domains (*miconic* or *openstacks*) $|S_{abs}| > |S|$ because of the additional propositions added to represent when a landmark has been achieved.

Additionally, the geometric mean of the ratio between h^*_{abs} and $h^{lmcut}/h^{opt}_{LA}/h^{uni}_{LA}$ is shown for every domain. Only instances in which P_{abs} could be solved optimally under a time limit of 300 seconds were included. A ratio between h^*_{abs} and h^{opt}_{LA} close to 1.00 means that using P_{abs} probably has little potential in those domains; a higher ratio, as in *barman*, *blocks*, *pegsol*, *pipesworld* (both versions) and *psr-small* means that using P_{abs} may be promising. Also, note that disjunctive landmarks were not used; if disjunctive landmarks had been used, the ratio could have been higher as well in domains with symmetric resources, like *Gripper* and *Logistics*. The comparison with h^{lmcut} shows that although h^{lmcut} is on average closer to h^*, it varies substantially from domain to domain.

Related Work

Using landmarks in factored planning (Hoffmann, Porteous, and Sebastia 2004) was the first attempt to exploit the information contained in the landmark graph. However, it did not consider the fact that landmarks can be deleted nor the negative interactions between them. A subsequent work that employed mutexes to build layers of conjunctive landmarks addressed the latter (Hoffmann, Porteous, and Sebastia 2004). Although more informed than the original partitioning in disjunctive subgoals, computing the new layer was sometimes too inefficient, apart from offering no insight in terms of the formal properties of the problem.

Another work exploited inconsistencies between landmarks and local interactions between achievers and other landmarks to compute the minimum number of states required to satisfy given sets of landmarks (Porteous and Cresswell 2002), which in turn could be used to obtain a lower bound on the number of times a landmark must be achieved. This information captures the fact that in a sequential plan landmarks may have to be deleted even if they are needed again later and can be used in a cost-partition scheme to derive admissible heuristics with properties similar to the ones mentioned in this work. However, this approach is difficult to define formally due to its procedural nature and offers potentially less information than the generalization of the landmark graph.

Another closely related work proposed a SAT compilation of the landmark graph (Alcázar and Veloso 2011). The compilation generates a SAT problem which is solved using a regular SAT solver in order to obtain a more informative version of the landmark graph. Orderings of the landmark graph and londexes between landmarks (Chen, Xing, and Zhang 2007) are used to create the constraints of the problem. The new version of the landmark graph obtained from the solution of the SAT problem encodes the time steps in which landmarks may be needed, which implicitly captures the fact that some landmarks may have to be achieved several times and which gives a possible total order of the set of landmarks. However, this approach works with a parallel scheme and does not take into account the achievers of the landmarks, so it cannot consider costs nor offer reliable information about the possible total orders.

Finally, encoding achieved landmarks as new variables in P was also proposed (Domshlak, Katz, and Lefler 2012). This work focused on obtaining more informative abstractions to derive admissible heuristics, whereas here we make a preliminary study of the potential of the landmark abstraction in isolation. Both works have important points in common, so a deeper comparison between both approaches (like checking how close a projection of the enriched P is to P_{abs}) may be fundamental for future developments.

Conclusions and Future Work

In this work a generalization of the landmark graph as a planning problem has been presented. So far the main contribution of the discussed approach is a theoretical one, although several ways of exploiting the obtained abstraction have been proposed. This generalization bridges the gap towards the integration of landmarks in a common framework along with abstractions and heuristics in automated planning. Furthermore, the formal properties that relate it to the original problem have been analyzed.

It remains an open question whether there are additional ways of exploiting or improving the abstraction. On the one hand, it is unclear whether some characteristics of the abstraction, like symmetries or solvability features, can be extrapolated to the original problem. On the other hand, the costs encoded in the generalization could be improved by using cost-partitioning schemes such as the ones used in additive admissible heuristics. For example, the costs of the actions could be substituted by lower bounds on the cost of achieving a given landmark. This is actually already possible in a simple way: if, through the path to a landmark, there are no positive interactions, the cost of the achieved landmark can be substituted by the cost of the path from the closest landmark to the achieved landmark. This is trivial when using actions whose preconditions and effects affect a single invariant, like the *move* operator in the example presented in this work. In this case, the cost of the *move* actions can be the cost of getting to the achieved landmark from the closest landmark, which interestingly enough would make that $h^*_{abs} = h^*$. Other approaches could extend to more general cases, so this line of research may be an interesting follow up of this work.

Acknowledgments

The author would like to thank the whole research group for Foundations of Artificial Intelligence at Freiburg University and in particular Malte Helmert, without whom this

work would not have been possible. The author would like to thank Michael Katz too for his insightful comments about the related work. This work has been partially supported by the Spanish government through MICINN projects TIN2008-06701-C03-03 (as a FPI grant) and TIN2011-27652-C03-02.

References

Alcázar, V., and Veloso, M. 2011. A SAT compilation of the landmark graph. In *COPLAS*, 47–54.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2):165–204.

Chen, Y.; Xing, Z.; and Zhang, W. 2007. Long-distance mutual exclusion for propositional planning. In *IJCAI*, 1840–1845.

Domshlak, C.; Katz, M.; and Lefler, S. 2012. Landmark-enhanced abstraction heuristics. *Artificial Intelligence* 189(0):48 – 68.

Gerevini, A., and Serina, I. 2002. LPG: A planner based on local search for planning graphs with action costs. In *AIPS*, 13–22. AAAI.

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

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

Helmert, M. 2006. The Fast Downward planning system. *J. Artif. Intell. Res. (JAIR)* 26:191–246.

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

Holte, R. C.; Perez, M. B.; Zimmer, R. M.; and MacDonald, A. J. 1996. Hierarchical A*: Searching abstraction hierarchies efficiently. In *AAAI/IAAI, Vol. 1*, 530–535.

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

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

Lipovetzky, N., and Geffner, H. 2011. Searching for plans with carefully designed probes. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *ICAPS*, 154–161. AAAI.

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

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res. (JAIR)* 39:127–177.

Vidal, V., and Geffner, H. 2005. Solving simple planning problems with more inference and no search. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP'05)*, volume 3709 of *LNCS*, 682–696. Sitges, Spain: Springer.