Cost Optimal Planning with LP-Based Multi-valued Landmark Heuristic

Lei Zhang, Chong-Jun Wang*, Jun-Yuan Xie
State Key Laboratory for Novel Software Technology
Department of Computer Science and Technology
Nanjing University, Nanjing 210046, China
zhanglei.com@gmail.com,{chjwang,jyxie}@nju.edu.cn

ABSTRACT

Landmark based heuristics are among the most accurate current known admissible heuristics for cost optimal planning. Disjunctive action landmarks can be considered as at-least-one constraints on the actions they contains. In many planning domains, there are many critical propositions which have to be established for a number of times. Previous landmarks fail to express this kind of general cardinality constraints. In this paper, we propose to generalize landmarks to multi-valued landmarks to model general cardinality constraints in cost optimal planning. We show existence of complete multi-valued landmark sets by explicitly constructing complete multi-valued action landmark sets for general planning tasks. However, it's computationally intractable to extract and exploit exact lower bounds of general multi-valued action landmarks. We devise a linear programming based multi-valued landmark heuristic h^{lpml} which extracts and exploits multi-valued landmarks using a linear programming solver. The heuristic h^{lpml} is guaranteed to be admissible and can be computed in polynomial time. Experimental evaluation on benchmark domains shows h^{lpml} beats state-of-the-art admissible heuristic in terms of heuristic accuracy and achieves better overall coverage performance at the cost of using more CPU time.

Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Heuristic methods

General Terms

Algorithms, Experimentation

Keywords

Heuristic search planning; Landmark heuristics; Cost optimal planning

1. INTRODUCTION

The task of cost optimal planning is to find a solution plan with minimum cost. It is a challenging problem that involves not only finding a satisficing plan, but also proving there is no plan with lower cost. Currently, the dominating approach to cost optimal planning is heuristic search based planners [2] which are guided with admissible heuristics [8, 15]. Admissible heuristics are usually obtained from various relaxations of original problems. Currently, there are four main classes of relaxations commonly used in classical planning: delete relaxations, critical paths, landmarks and abstractions. Landmark based heuristics are among the most accurate admissible heuristics.

Landmarks are sub-goals which must be achieved in all valid solution plans. They were first used to decompose planning tasks into smaller sub-tasks and guide search [9]. Landmarks are later on exploited for heuristic functions. The first successful landmark heuristic, landmark-count, counts the number of fact landmarks that still need to be achieved [15]. The planner LAMA based on this inadmissible heuristic showed the best performance in the sequential satisficing track of the sixth International Planning Competition (IPC-6). The first admissible heuristic based on landmarks exploits disjunctive action landmarks which are induced by propositional fact landmarks [10]. Helmert and Domshlak [8] proposed an efficient disjunction action landmark extraction procedure that iteratively cuts between reached and unreached parts of planning tasks, the resulting heuristic LM-cut is currently one of the most successful admissible heuristics in practice. In an experiment measuring the accuracy of different heuristics, LM-cut computes exact h^+(the optimal delete relaxation heuristic) for more than 70% of all problem instances [3]. Meanwhile, previous work try to encode delete free planning tasks(Π^+) as MaxSAT problems and extract action landmarks as unsatisfiable cores using MaxSAT solvers [16, 4].

However, previous disjunctive action landmarks can all be seen as at-least-one constraints: at least one action of a disjunctive action landmark must be included in all valid solution plans. This kind of at-least-one constraints is too weak for domains where propositions need to be established for many times. In these domains, we need more general constraints like cardinality constraints to express non-binary lower bounds. In this paper, we introduce multi-valued landmarks to model cardinality constraints in cost optimal planning. Since we can reduce multi-valued landmarks to landmarks, it's easy to see that general multi-valued landmarks are at least as intractable to extract as general landmarks. However, we observe that for a certain class of propositions there is a strong relationship between the number of times they are added and the number of times they are deleted. We refer to this class of propositions as regular transition propositions. Based on this property of regular transition propositions, we devise a multi-valued landmark heuristic h^{lpml} which use linear programming solvers to extract and exploit multi-valued landmarks, the heuristic is proved to be admissible and can be computed in polynomial time. Experiment results on benchmark domains from recent IPC show that h^{lpml} is more

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

*Corresponding author email: chjwang@nju.edu.cn

accurate than state-of-the-art landmark based admissible heuristic and achieves better overall coverage performance.

The reminder of this paper is structured as follows: we first provide a brief background on classical planning and heuristic search planners in Section 1. In Section 2, we present main notions of landmark based planning and current admissible heuristics that are based on landmarks. In Section 3 and Section 4, we introduce multi-valued actions landmarks and multi-valued proposition landmarks, we discuss their properties and how multi-valued actions landmarks can be induced from multi-valued proposition landmarks. We then present the linear programming based multi-valued landmark heuristic h^{lpml} in Section 5. Section 6 provides experiment evaluation of h^{lpml} on benchmark domains. Conclusions are given in Section 7.

2. NOTATION AND BACKGROUND

Planning and Delete Relaxation. A planning problem $\Pi = \langle P, I, G, A, cost \rangle$ consists of a set of facts P, an initial state $I \subseteq P$, a goal state $G \subseteq P$, and a set of actions A. Each action $a \in A$ is specified by the sets $\langle pre(a), add(a), del(a) \rangle$, where $pre(a)$ are the preconditions of a, $add(a)$ are the add (positive) effects of a, and $del(a)$ are the delete (negative) effects of a. Each of these sets is a subset of P. *cost* is a cost function that assigns each action in A a positive real valued cost bounded away from zero. The delete relaxation Π^+ of Π is the same problem but for every $a \in A$ we set $del(a) = \emptyset$. States are subsets of P.

An action a is applicable in state s if $pre(a) \subseteq s$. The result of applying action a to state s is $result(s,a) = (s \cup add(a)) \setminus del(a)$. The result of applying an action sequence to state s is defined recursively as: $result(s, \langle a_1, \cdots, a_n \rangle) = result(result(s, \langle a_1 . \cdots, a_{n-1} \rangle), a_n)$. The sequence is said to be executable if each a_i is applicable, i.e., $pre(a_i) \subseteq result(s, \langle a_1, \ldots, a_{i-1} \rangle)$. A fact ϕ is true at time i iff $\phi \in result(I, \langle a_1, \cdots, a_i \rangle)$. A valid plan π for a state $s \subseteq P$ is an executable sequence of actions such that $G \subset result(s, \pi)$. The cost of π, $cost(\pi)$, is the sum of the cost of its actions. The cost of a minimum cost of plan for the initial state I in the relaxed planning problem Π^+ is denoted by $h^+(\Pi^+)$. The cost of an optimal plan, $cost(\pi^*)$, is denoted by h^*.

Heuristic-Search Planning. Heuristic search planners solve planning tasks using exact search algorithms like A*. The exact search algorithms are usually guided with heuristics which are derived automatically from problem definitions. The success of heuristic search planners depends mainly on how informative the heuristics are. Heuristics are functions $h : S \mapsto \mathbb{R}_0^+$ that estimate cost of path to goal states. In order to find optimal plans, the heuristic function h needs to be admissible: it does not overestimate the cost of reaching the goal states, that is, for all state s, we have $h(s) \leq h^*(s)$, where h^* stands for the optimal heuristic which is defined as the cost of the lowest-cost path from initial state to goal states. With optimal heuristics, we can then use best first search to compute optimal plans in linear time. However, the optimal heuristics h^* are intractable to compute in general, in fact, even computing h^+ is NP-hard. Our objective is to get as close to optimal heuristics as possible with efficient algorithms.

3. PREVIOUS ADMISSIBLE LANDMARK HEURISTICS

Landmarks are subgoals that must be achieved at some time point in every solution plan. There are two main kinds of landmarks, fact landmarks and action landmarks, which are formally defined as follows:

DEFINITION 1. *A fact ϕ is a **fact landmark** of planning task Π iff ϕ is true at some time in every solution plan of Π.*

DEFINITION 2. *An **action landmark** of planning task Π is a single action that must be executed at some time in every solution plan of Π.*

Moreover, a disjunction f of facts is a **disjunction fact landmark** if f must be true at some time in every solution plan of Π. A set O of actions is a **disjunction action landmark** if O has non-empty intersection with every solution plan of Π. It is intractable (PSPACE-complete in general) to decide whether a disjunction of facts or a disjunction of actions is a landmark for Π or not [14].

Landmarks were first used as subgoals to decompose planning tasks, but most current planners exploit landmarks for heuristics. A disjunction action landmark $l = \{a_1, \cdots, a_k\}$ for Π can be seen as a heuristic function $h_l = min_{a \in l} cost(a)$, and h_l is referred to as l's cost. Given a set of disjunction action landmarks $L = \{l_1, \cdots, l_m\}$, we can exploit L by taking the maximum cost: $h_L = max_{l \in L} h_l$. This exploitation approach is very weak and loses information contained in landmarks. If we take the sum of landmark costs, the resulting heuristic would not be admissible in general because some actions might be counted more than once. Current admissible landmark based heuristics use more sophisticated landmark exploitation approaches like cost partitioning and hitting set.

Cost partitioning. A cost partitioning for landmark set L is a set $C = \{c_1, \cdots, c_m\}$ of cost functions that satisfy the constraint $\sum_{i=1}^{m} c_i(a) \leq cost(a)$ for each action a. The cost partition C defines a new heuristic $\sum_{i=1}^{m} min_{a \in l_i} c_i(a)$. Through a cost partitioning, each action's cost is distributed across m new planning tasks defined by $c_1, \cdots, c_m$. Each cost function c_i defines a new planning task. The constraints ensure that the sum of costs is admissible. Given the same set of landmarks, different cost partitions will lead to different heuristics. Optimal cost partitionings can be obtained by solving linear programs, the resulting heuristics are usually more informative [10]. LM-cut can also be seen as a cost partitioning based heuristic [8], it obtains disjunction action landmarks by iteratively cutting justification graphs, and the cost partitioning used in LM-cut can be computed very efficiently.

Hitting set. Hitting set is another way of exploiting disjunctive action landmarks, it is computationally more expensive than LM-cut and optimal cost partitioning, however it can produce more informative heuristics . Assume $A = \{a_1, \cdots, a_n\}$ is an action set and $L = \{l_1, \cdots, l_m\}$ is a family of subset of A. A subset $S \subseteq A$ is called as the hitting set of L if and only if S has non-empty intersection('hits') with every subset in L. A hitting set is called minimum cost hitting set if its cost is less or equal to all hitting sets. A landmark set $L = \{l_1, \cdots, l_m\}$ of planning task Π can be seen as a instance of hitting set problems. The cost of L's minimum-cost hitting set is an admissible heuristic. Hitting set based heuristics dominate both LM-cut and optimal cost partitioning heuristics if given the same set of landmarks [3], but finding a minimum-cost hitting set is a NP-complete problem.

Note that most previous admissible landmark heuristics are based on landmarks of delete relaxation Π^+. They are admissible approximations of h^+ and ignore delete information. Some recent landmark techniques take into account delete effects by compiling general planning tasks into delete free tasks [12, 5, 11, 6]. However, very few work has been done on the topic of repeated or multi-valued landmarks except [13]. In this paper, we try to further explore this notion of multi-valued landmarks in the context of cost optimal planning.

4. MULTI-VALUED ACTION LANDMARKS

In many planning domains, some propositions have to be established more than once. For example, in the blocks world domain, if the length of an optimal plan π^* for Π is n, proposition HANDEMPTY will have to be established for at least $\lfloor n/2 \rfloor$ times. Since only PUTDOWN actions and STACK actions have HANDEMPTY as an add effect, the action landmark that contains all PUTDOWN actions and STACK actions has a lower bound of $\lfloor n/2 \rfloor$. Previous disjunctive action landmarks can be seen as a form of at-least-one constraints on the actions it contains, which makes them unsuitable for expressing these non-binary lower bounds. In these cases, it will be convenient to have multi-valued action landmarks to model these cardinality constraints.

DEFINITION 3. *A **multi-valued action landmark** [1] $l = \langle A_l, lb_l \rangle$ of planning task Π contains a set of actions $A_l = \{a_1, \cdots, a_k\}$ and a lower bound lb_l such that the number of occurrences of actions in A_l is greater than or equal to lb_l in every solution plan π,*

$$\forall \pi, \quad \sum_{a_i \in A_l, a_i \in \pi} a_i \geq lb_l.$$

Landmarks in previous work can be seen as a special case of multi-valued landmarks with lower bounds set to 1. Note that every set of action can be seen as a trivial multi-valued landmark with lower bound equals 0. Another example of multi-valued landmarks is the global multi-valued landmark set L whose action set consists of all actions in A: $L = \langle \{a | a \in A\}, 0 \rangle$.

From known multi-valued action landmarks, new multi-valued landmarks can be derived. For instance, let $l_1 = \langle A_1, lb_1 \rangle$ and $l_2 = \langle A_2, lb_2 \rangle$ be two multi-valued action landmarks, then we have: 1) $l_3 = \langle A_1 \cup A_2, max(lb_1, lb_2) \rangle$ is a valid landmark; 2) If $A_1 \cap A_2 = \emptyset$, $l_4 = \langle A_1 \cup A_2, lb_1 + lb_2 \rangle$ is a valid landmark.

A collection C of landmarks is complete for planning task Π if the cost of a minimum-cost hitting set of C equals $h^*(\Pi^+)$. The minimal cost hitting set is an optimal plan for Π^+ [1]. The minimal cost hitting set model , however, can not be used to exploit multi-valued landmarks, because general multi-valued landmarks have non-binary lower bounds and need to be "hit" multiple times. So a model more general than the minimal cost hitting set is needed. We use covering integer programs here since multi-valued landmarks can be naturally modeled as **at-least-k constraints**. For a multi-valued landmark set L, we define $C(L)$ as the cost of multi-valued landmark set L.

$$C(L) = \min\{\sum_{a_i} a_i * cost(a_i) | \forall l \in L, \sum_{a_i \in A_l} a_i \geq lb_l\} \qquad (1)$$

Now we can define a similar notion of completeness for multi-valued action landmark set.

DEFINITION 4. *A multi-valued landmark set L is **complete** for planning task Π if the cost of L equals $h^*(\Pi)$: $C(L) = h^*(\Pi)$.*

In [3], it's proved that there always exists a collection C of delete relaxation landmarks that is complete for delete relaxed tasks Π^+. We can obtain similar results for multi-valued landmarks. In the following, we show the existence of complete multi-valued landmark set by explicitly constructing complete multi-valued landmark sets for delete relaxed tasks as well as planning tasks with delete effects.

[1]Here we don't differentiate between action landmarks and disjunction action landmarks in the context of multi-valued landmarks, as we can consider action landmarks as special disjunction action landmarks whose action sets have only one action. Both are referred to as multi-valued action landmarks in the following.

PROPOSITION 1. *Existence of complete multi-valued landmark set:*

- *The multi-valued landmark set $\{L = \langle A, h^+ \rangle\}$ is **complete** for delete relaxed task Π^+.*
- *The multi-valued landmark set $\{L' = \langle A', h^*(\Pi) \rangle\}$ is **complete** for Π, where $A' = \{a^{i,t} | t \in [1, |\pi|], i \in [1, |A|]\}$ and Π has a satisficing plan π with bounded plan length.*

PROOF. In delete relaxed task Π^+, actions don't have any delete effect, so every action is required at most once. Therefore, the optimal delete relaxed plan π^+ contains no duplicate actions. For any optimal relaxed plan π^+, we have $\{a | a \in \pi^+\} \subseteq A$. The cost of multi-valued landmark set $\{L\}$ equals h^+. Therefore, $\{L\}$ is complete for Π^+.

In planning tasks with delete effects, actions might be required for more than once, so action sets of complete multi-valued landmark set should contain enough action duplicates. Since for planning task Π there is a satisficing plan π with bounded plan length $|\pi|$, then the number of occurrences of each action will also have a upper bound $|\pi|$. In the action set of multi-valued landmark L', each action a_i has $|\pi|$ duplicates. Therefore, for any optimal solution plan π^*, we have $\{a | a \in \pi^*\} \subseteq A'$. Since the lower bound of L' is equal to h^*, $\{L'\}$ is complete for Π. □

Given a complete set L of multi-valued landmark set, there is a solution π of the covering integer program $C(L)$ that is an "optimal plan". Note that the solution π is a multi-set of actions that are not sequenced. By appropriate sequencing, we can get an executable optimal plan out of π. However, since we only use multi-valued landmarks for generating heuristic functions, we don't consider the sequencing issue in this paper.

As the exact lower bounds of L is equal to h^+, it's NP-hard to extract L's exact lower bound. Similarly, it's PSPACE-complete in general to extract L''s exact lower bound. In fact, even for multi-valued landmarks whose action sets consist of only one action, it can be shown that it's PSPACE-complete to decide whether their exact lower bounds are greater than 0. In the following section, we introduce multi-valued proposition landmarks from which multi-valued action landmarks can be extracted more efficiently.

5. MULTI-VALUED PROPOSITION LANDMARKS

Although general multi-valued action landmarks are hard to extract, there are sound extraction methods for fact landmarks. For example, every goal fact that is not present in the initial state is a fact landmark; if all actions achieving a subgoal have a common precondition fact which is not in the initial state, the common precondition fact is also a fact landmark. These methods are incomplete in that they can only extract a certain class of fact landmarks. Previous admissible landmark heuristics rely on these methods to generate fact landmarks and use the induced disjunction action landmarks [10]. In this section, we introduce multi-valued proposition landmarks and discuss how can we extract multi-valued action landmarks by using multi-valued proposition landmarks.

DEFINITION 5. *A **multi-valued proposition landmark** $l = \langle p, lb \rangle$ of Π consists of a proposition p and a lower bound lb such that proposition p will be added at least lb times in any solution plan of Π.*

For proposition p, we define $add(p)$ as the set of actions that have p as add effect and $del(p)$ as the set actions that have p as delete effect. Given a multi-valued proposition landmark $l =$

$\langle p, lb \rangle$, by the definition of multi-valued proposition landmarks, we can get an **induced** multi-valued action landmark $\langle add(p), lb \rangle$.

However, we can not reach similar conclusions about the lower bound of $del(p)$: given a multi-valued proposition landmark $\langle p, lb \rangle$, it's not always the case that exact lower bounds of $add(p)$ and $del(p)$ are both equal to lb. It's possible that the lower bounds of $add(p)$ and $del(p)$ are independent. For instance, p can be added several times before it is deleted for the first time, or it can be deleted many times and never be added. In these cases, not all actions that have p in their add list or delete list will induce a truth transition of p. For propositions that do satisfy this constraint, we will refer to them as regular transition propositions, and we will show that the exact lower bounds of $add(p)$ and $del(p)$ for regular transition proposition p are very strongly correlated.

DEFINITION 6. *A proposition p is a **regular transition proposition** for Π if the following conditions hold:*
1) $\forall a \in add(p), \forall s \in S_I, pre(a) \subseteq s \rightarrow \neg p \in s$
2) $\forall a \in del(p), \forall s \in S_I, pre(a) \subseteq s \rightarrow p \in s$
where S_I is the set of states reachable from initial state I.

For regular transition proposition p, all actions having p as add effect will induce a false to true transition of p when applied, and all actions having p as delete effect will induce a true to false transition of p when applied. By the definition, to determine whether a proposition is a regular transition one or not, we have to know the set of reachable states from I, which is not feasible in practice. However, we can conclude that p is not a regular transition proposition if it can be added in a state where p is already true. For example, in the visitall domain, location x can be visited many times, which means that actions adding $visited(x)$, for example $move(y,x)$, are applicable in states where $visited(x)$ is already true, therefore, $move(y,x)$ does not necessarily change the truth value of $visited(x)$, and $visited(x)$ is not a regular transition proposition.

On the other hand, we can conclude that p is a regular transition proposition if we are certain that p can not be added in states where p is already true, and at the same time p can not be removed in states where p is already false. This is a sufficient condition for p to be a regular transition proposition and it can be checked as follows,

PROPOSITION 2. *Proposition p is a regular transition proposition if both the following conditions hold:*

- $\forall a \in add(p)$, $\neg p \in pre(a)$ *or* $\exists q \in pre(a)$ *such that q and p are inconsistent.*
- $\forall a \in del(p), p \in pre(a)$

The proof is omitted. Using this proposition, we can determine whether p is a regular transition proposition in $O(|add(p)|+|del(p)|)$ time. For example, consider the *put-down*(x) action in the blocksworld domain, its preconditions and effects are as follows,

- precondition: holding(x)
- add effect: clear(x), handempty, ontable(x)
- delete effect: ¬holding(x)

We can see from above that *handempty* is an add effect of *put-down*, but ¬*handempty* is not present in *put-down*'s precondition. Since *holding*(x) and *handempty* is mutex, *handempty* must be false in all states where *put-down*(x) is applicable. Therefore *put-down*(x) will induce truth transitions of *handempty* whenever they are applied. By this kind of analysis, all four classes of actions *pick-up*, *put-down*, *stac*k, *unstack* in the blocksworld domain will induce truth transitions of *handempty* when applied, therefore *handempty* is a regular transition proposition. This kind of analysis can be done for planning domains statically requiring no search.

If p is a regular transition proposition, then there is a strong correlation between lower bounds of $add(p)$ and $del(p)$. For example, if p is present in initial state and all goal states, $p \in I$ and $p \in G$, then the lower bound of $add(p)$ should be equal to the lower bound of $del(p)$. The result is stated formally below.

PROPOSITION 3. *If p is a regular transition proposition, then the following equation holds for all state $S \in S_G$:*

$$I(p) + \sum_{a_i \in add(p)} a_i = S(p) + \sum_{a_j \in del(p)} a_j$$

where S_G is the set of all goal states that are reachable from initial state I.

PROOF. Note that $S(p)$ is a well defined goal state(i.e., not a set of goal states). Assume $I(p)$ and $S(p)$ have the same truth value. Because p is a regular transition proposition, all actions in $add(p)$ and $del(p)$ will change the truth value of p when they are applied, so the number of times p is added must be equal to the number of times it is deleted. The exact lower bounds of $add(p)$ and $del(p)$ are equal and the above equation holds. When p is in i but not in S, p must be deleted one more time than it is deleted. The proof is similar when $I(p)$ and $S(p)$ have different truth values. □

Proposition 3 can be used to extract multi-valued landmarks. For instance, if $p \in G$ and $add(p)$ has a lower bound of n, then $del(p)$ will also be a multi-valued action landmark with a lower bound of $n + I(p) - 1$. If p is not a regular transition proposition, then the above equation does not hold in general. However, in many domains, it's often the case that goal facts are not regular transition propositions For example, in the visitall domain, goal facts like $visited(x)$ are not regular transition propositions. In this case, the above equation will not give any non-trivial lower bounds for these goal facts. We can eliminate the gap by using the following constraints for non-regular transition propositions.

PROPOSITION 4. *If p is a non-regular transition proposition, then the following inequality holds for all state $S \in S_G$, $I(p) - S(p) + \sum_{a_i \in add(p)} a_i \geq 0$.*

PROOF. If $p \in S$ and $I(p) = 1$, then the constraint become trivial $\sum_{a_i \in add(p)} a_i \geq 0$. If $p \in S$ and $I(p) = 0$, then p has to be added at some time, the constraint become $\sum_{a_i \in add(p)} a_i \geq 1$. On the other hand, if $p \notin S$, then $I(p) - S(p) = I(p)$, the constraint become again trivial $I(p) + \sum_{a_i \in add(p)} a_i \geq 0$. □

Example. As an example, consider the planning task Π shown in figure 1. Initially, we are at position 0 and only position 0 is visited. The goal is to visit all positions from 0 to 8. By iteratively cutting the justification graph, LM-cut will give the following landmark set L:

$$L = \{\{Move(0,i)\} | i \in [1,8]\}$$

The heuristic value given by LM-cut is 8, which is equal to $h^+(\Pi)$. In fact, $\pi = \cup_{l \in L} l$ is an optimal relaxed plan for Π. Consider proposition $At(0)$, for each action a in $add(At(0)) = \{Move(i,0) | i \in [1,8]\}$, we have $\neg At(0) \in pre(a)$; for each action a in $del(At(0)) = \{Move(0,i) | i \in [1,8]\}$, we have $At(i) \in pre(a)$, which means $\neg At(0) \in pre(a)$. Therefore $At(0)$ is a regular transition proposition. Since

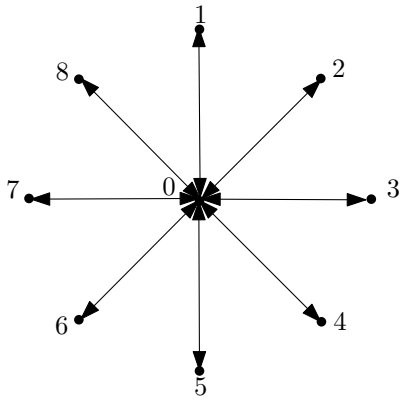

Figure 1: A visitall planning task Π: $P = \{At(i), Visited(i) | i \in [0,8]\}$, $I = \{At(0), Visited(0)\}$, $G = \{Visited(i) | i \in [0,8]\}$, $A = \{Move(i,j) | i,j \in [0,8]\}$. For every position i and position j that are adjacent, action $Move(i,j)$ will delete $At(i)$ and add $At(j)$ and $Visited(j)$.

all eight actions $Move(0,1), \cdots, Move(0,8)$ are all action landmarks, we then get the following multi-valued action landmark l_{del}:

$$l_{del} = \langle \{Move(0,i) | i \in [1,8]\}, 8 \rangle = \langle del(At(0)), 8 \rangle$$

Since At(0) is a regular transition proposition, by proposition 3, we have $add(At(0))$'s lower bound $lb_{add} = 8 + G(At(0)) - I(At(0)) \geq 7$. Then we get the following multi-valued action landmark l_{add}:

$$l_{add} = \langle \{Move(i,0) | i \in [1,8]\}, 7 \rangle = \langle add(At(0)), 7 \rangle$$

Since the action sets of l_{add} and l_{del} are disjoint, we can safely combine them into a new landmark,

$$l_c = \langle \{Move(0,i), Move(i,0) | i \in [1,8]\}, 15 \rangle$$

The lower bound of l_c is tight: $h^* = lb_c = 15$. There are C_8^7 different optimal plans, we get the optimal heuristic without finding out one optimal plan. Meanwhile, note that $Visited(0)$ is a non-regular transition proposition, we could visit a position many times even in optimal plans. The exact lower bounds for the following landmarks are not the same: $\langle Visited(0), 0 \rangle$, $\langle add(Visited(0)), 7 \rangle$, $\langle del(Visited(0)), 0 \rangle$,

Note all multi-valued action landmarks can not be induced from multi-valued proposition landmarks. There are fact landmarks that consist of conjunctions of propositions. For example, a fact landmark of proposition p and q encodes information about at least how many times $p \wedge q$ is established in any solution plan. How to extend multi-valued proposition landmark to handle this kind of fact landmarks is a future research direction.

6. A LP-BASED MULTI-VALUED LANDMARK HEURISTIC

As we have seen, multi-valued landmark heuristics can be very accurate: given a complete multi-valued action landmark set L, we can extract optimal heuristics by solving the covering integer program $C(L)$. With optimal heuristics, we only need linear time to find an optimal plan. However, it's not the case that more accurate heuristics would lead to better overall performance. To compute optimal heuristics, we need to solve a NP-hard problem at every search node, which would need exponential time in the worst case. Therefore, the accuracy of multi-valued heuristics does not always pay off.

Algorithm 1 LP-based multi-valued landmarks heuristic

Input: Planning task Π , an initial action landmark set L'
Output: LP-based multi-valued landmarks heuristic h^{lpml}

```
1: function LPML(Π, L')
2:     ▷ cons is the constraint set
3:     cons=cons ∪{a ≥ 0}
4:     for all proposition p do
5:         if p ∈ G then
6:             cons=cons ∪{G(p) = 1}
7:         else if ¬p ∈ G then
8:             cons=cons ∪{G(p) = 0}
9:         else
10:            cons=cons ∪{0 ≤ G(p) ≤ 1}
11:        end if
12:        ▷ add regular proposition constraints
13:        if p is a regular proposition then
14:            c= {I(p) + Σ_{p∈add(a_i)} a_i = G(p) + Σ_{p∈del(a_j)} a_j}
15:            cons=cons ∪ c
16:        else
17:            ▷ add constraints for non-regular propositions
18:            cons=cons ∪{I(p) + Σ_{p∈add(a_i)} a_i − G(p) ≥ 0}
19:        end if
20:    end for
21:    for l ∈ L' do
22:        ▷ constraints implied by initial landmark set L'
23:        cons=cons ∪{Σ_{a∈l} a ≥ 1}
24:    end for
25:    ▷ resulting linear program lp subject to cons
26:    lp = min{Σ a_i * cost(a_i) | cons}
27:    return lp_solve(lp)
28: end function
```

In this section, we present an admissible heuristic h^{lpml} which uses linear programming solvers to extract and exploit multi-valued landmarks. Since only heuristic functions are needed, we do not have to maintain a explicit set L of multi-valued landmarks and then exploit L by solving the resulting covering integer program $C(L)$. We only need to minimize total plan costs subject to regular transition propositions constraints in proposition 3 and non-regular transition propositions constraints in proposition 4. These constraints will induce an implicit set of multi-valued landmark. So we can extract and exploit multi-valued landmarks by using the following integer program C',

$$\begin{array}{ll} \text{Minimize} & \sum_i a_i * cost(a_i) \\ \text{subject to:} & a_i \in \mathbb{N}_0, \forall a_i \in A \\ & I(p) + \sum_{p \in add(a_i)} a_i = G(p) + \sum_{p \in del(a_j)} a_j, \forall p \in P^R \\ & I(p) + \sum_{p \in add(a_i)} a_i - G(p) \geq 0, \forall p \in P \setminus P^R \\ & G(p) \in [0,1], \forall p \in P \end{array} \quad (2)$$

where P^R is the set of regular transition propositions. Since it's intractable to solve C', we relax C' into a linear program. All integer constraints for action variables a_i and fact variables $G(p)$ are replaced by linear constraints. We also accept an initial set L' of landmarks extracted by other methods (LM-cut is used in our implementation), and add those constraints induced by L' into the linear program CL'. In our experiments, an initial landmark set will often speed up the solving process of our LP solver. The resulting

linear programming relaxation CL' is shown below,

$$\begin{array}{ll} \text{Minimize} & \sum_i a_i * cost(a_i) \\ \text{subject to:} & a_i \geq 0, \forall a_i \in A \\ & I(p) + \sum_{p \in add(a_i)} a_i = G(p) + \sum_{p \in del(a_j)} a_j, \forall p \in P^R \\ & I(p) + \sum_{p \in add(a_i)} a_i - G(p) \geq 0, \forall p \in P \setminus P^R \\ & \sum_{a \in A_l} a \geq lb_l, \forall l = \langle A_l, lb_l \rangle \in L \\ & 0 \leq G(p) \leq 1, \forall p \in P \end{array} \quad (3)$$

The optimum of the above linear programming relaxation CL' is our LP-based multi-valued landmark heuristic, which is denoted by h^{lpml}. The details of the new multi-valued landmark heuristic are shown in algorithm 1. The new heuristic h^{lpml} is guaranteed to be admissible and easy to compute.

THEOREM 1. *$h^{lpml} = OPT(CL')$ is an admissible estimate of h^* and it can be computed in polynomial time.*

PROOF. By proposition 3 and proposition 4, we have all valid plans of Π are solutions to integer program C', so we have that the optimum of C', $OPT(C')$, lower bounds the optimal plan cost h^*. Meanwhile, since all solutions of C' are also solutions of CL', the optimum of CL' is a lower bound on $OPT(C')$: $OPT(CL') \leq OPT(C')$. Therefore both $OPT(CL')$ and $OPT(C')$ are admissible estimates of h^*.

Meanwhile, the number of variables of CL' is $O(|A| + |P|)$, and the number of constraints of CL' grows linearly with the size of L' and P, so the resulting linear programming relaxation CL' can be solved in polynomial time by linear programming solvers. □

7. EXPERIMENTS

In this section, we study the accuracy and performance of the new multi-valued heuristic on benchmark domains. The platform used for all experiments is Fast Downward [7]. The linear programming solver used in our planner is GNU Linear Programming Kit (GLPK).[2] Experiments were run on a Intel Core 2 Duo at 2.8Ghz with 2GB RAM. We use all benchmark domains from the optimal track of the seventh International Planning Competition (IPC-7) and another example domain blocksworld. For each problem instance, the timeout is set to 1800 seconds and the memory limit is 2GB.

In the first set of experiments, we evaluate the accuracy of multi-valued landmarks heuristic h^{lpml}. We use planning tasks from two particular benchmark domains, blocksworld and visitall, which are used as examples in previous sections. For each planning task, we compute the heuristics value $h(I)$ for its initial state I. The heuristics used for comparison are the most success landmark based heuristic LM-cut and our multi-valued landmark heuristic h^{lpml}. Both heuristics are implemented in the same planning system to allow an unbiased comparison. The result of heuristic value comparison is shown in figure 2. We can see from the figure that h^{lpml} outperforms LM-cut in term of heuristic accuracy on all tasks from both domains. Meanwhile, more detailed performance statistics of the two planners on these tasks, like node expansion and search time, are shown in table 1. Due to space limitation, we only show statistics for most difficult tasks in each domain. From table 1, we can see h^{lpml} strictly improves the number of node expansions in all but two tasks, which confirms the fact that h^{lpml} is more accurate than LM-cut on both domains. The search time performance of the two planners is more interesting, although h^{lpml} is more accurate than LM-cut on both domains, the planner based on h^{lpml} uses more search time on the blocksworld domain: out of all 13 planning tasks, the planner based on LM-cut achieves better search time performance in 11 tasks. This suggests that in the blocksworld domain the increased accuracy of h^{lpml} does not pay off, although h^{lpml} uses less node expansions, it spends more time on expanding each search node. In this case, LM-cut achieves a better trade-off. On the other hand, we can also see that in the visitall domain, h^{lpml} uses less node expansions and achieves better search time performance at the same time. For example, in task 07-half, h^{lpml} uses half a second to solve the task while LM-cut uses more than 104 seconds, which is due to the fact that h^{lpml} uses only 0.01% of LM-cut's node expansions. In fact, out of all 18 planning tasks in the visitall domain, the planner based on LM-cut failed to 7 tasks while h^{lpml} based planner manages to solves all tasks.

Table 1: Comparing node expansion and search time of LM-cut and h^{lpml} on two particular planning domains(visitall and blocksworld), bold results indicate better performance, and '-' indicates time or memory limit is reached.

	node expansion		search time(sec)	
Instance	LM-cut	lpml	LM-cut	lpml
blocksworld				
9-0	13430	**2627**	**2.72**	4.5
9-1	376	**165**	**0.1**	0.3
9-2	610	**239**	**0.12**	0.48
10-0	248336	**56091**	**66.32**	144.56
10-1	30086	**6476**	**9**	18.63
10-2	86009	**18620**	**26.2**	52.73
11-0	65767	**14243**	**24.66**	46.5
11-1	66742	**8591**	**29.1**	35.72
11-2	58261	**16606**	**20.88**	58.47
12-0	61800	**5977**	31.86	**27.18**
12-1	6560	**2718**	**3.36**	11.82
14-0	113539	**9475**	104.74	**70.13**
14-1	191948	**29505**	**203.26**	262.32
visitall				
02-full	4	4	0.1	0.1
02-half	2	2	0.1	0.1
03-full	23	**9**	0.1	0.1
03-half	15	**7**	0.1	0.1
04-full	727	**16**	0.1	0.1
04-half	16	**12**	0.1	0.1
05-full	69729	**25**	5.08	**0.1**
05-half	1029	**39**	0.1	0.1
06-full	-	**36**	-	**0.17**
06-half	1045	**31**	0.1	0.1
07-full	-	**49**	-	**0.4**
07-half	519064	**72**	104.72	**0.54**
08-full	-	**64**	-	**0.8**
08-half	6080268	**8210**	1663.94	**69.05**
09-full	-	**81**	-	**1.46**
10-full	-	**100**	-	**2.76**
10-half	-	**8504**	-	**168.74**
11-full	-	**121**	-	**4.46**

Next, we provide a more detailed and extensive performance comparison of LM-cut and h^{lpml} on all benchmark domains from IPC-7 and the example domain blocksworld. There are in total 315 planning tasks. Table 2 shows the number of planning tasks solved by each heuristic, which is referred to as *coverage* in the table. Meanwhile, we also show the number of planning tasks solved exclusively by each heuristic, which is referred to as *exclusive coverage*. If a planner has a exclusive coverage of n, it means that the planner solved n planning tasks in this domain while the other planner did not solve. From the table, we can see that there are 3 domains where h^{lpml} based planner achieves strictly better coverage performance (parcprinter, visitall and woodworking), and there are 7 domains where LM-cut has better coverage performance. h^{lpml}

[2] http://www.gnu.org/software/glpk/

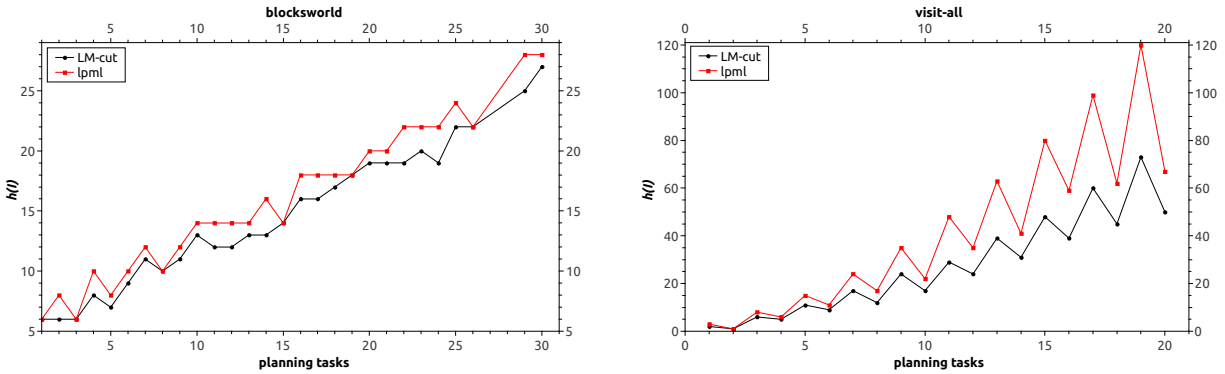

Figure 2: Comparison of heuristic values of initial states by each heuristic on two domains(visitall and blocksworld).

	coverage		exclusive coverage	
Domain	LM-cut	lpml	LM-cut	lpml
barman-opt11 (20)	4	4	0	0
blocks (35)	28	28	0	0
elevators-opt11 (20)	**16**	15	**1**	0
floortile-opt11 (20)	**7**	6	**1**	0
nomystery-opt11 (20)	**14**	12	**2**	0
openstacks-opt11 (20)	**11**	10	**1**	0
parcprinter-opt11 (20)	13	**20**	0	**7**
parking-opt11 (20)	1	1	0	0
pegsol-opt11 (20)	**17**	16	**1**	0
scanalyzer-opt11 (20)	10	10	0	0
sokoban-opt11 (20)	**20**	17	**3**	0
tidybot-opt11 (20)	**13**	6	**7**	0
transport-opt11 (20)	6	6	0	0
visitall-opt11 (20)	10	**18**	0	**8**
woodworking-opt11 (20)	11	**15**	1	**5**
sum(315)	181	**184**	17	**20**

Table 2: Comparing the number of tasks solved by LM-cut and h^{lpml} (coverage), and the number of tasks solved exclusively by LM-cut and h^{lpml}(exclusive coverage).

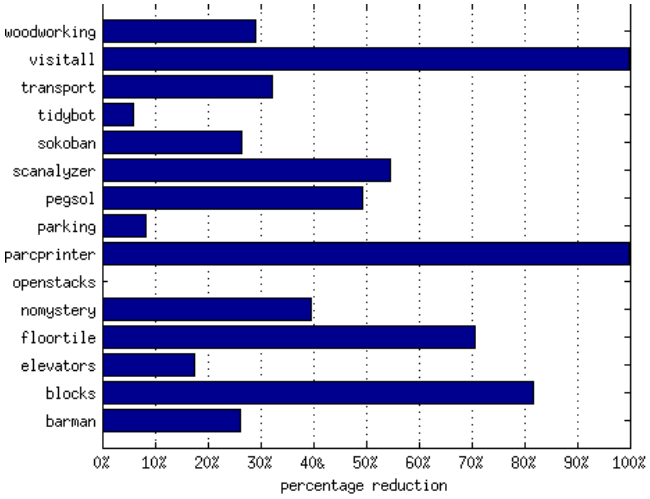

Figure 3: Node expansion reduction of $h^l pml$ compared with LM-cut, a 20% reduction means that h^{lpml} expands only 80% of nodes expanded by LM-cut. To allow fair comparison, we only uses tasks that can be solved by both heuristics.

has a particular bad performance in the tidybot domain (solve 7 less tasks), this is due to the fact that planning tasks in the tidybot domain usually have a large number of actions(19200 in average), which makes the LP encoding very large and slows down our LP solver. However, in total, the planner based on h^{lpml} solves 3 more tasks than the LM-cut based planner.

Table 3 and figure 3 show detailed performance statistics for planning tasks that can be solved by *both heuristics*. Figure 3 shows the node expansion reduction of h^{lpml} compared with LM-cut on these tasks, we can see h^{lpml} greatly reduces node expansions in all domains except openstacks where the percentage reduction is 0. The reduction in visitall and parcprinter is nearly 100%. For each domain, table 3 shows additional information such as initial heuristic value $h(I)$, node expansion, search time and memory usage. We can see that h^{lpml} produces a stronger initial heuristic for all but three domains. h^{lpml} also has a better memory performance: in 10 out of 15 domains, h^{lpml} uses less memory than LM-cut. Meanwhile, h^{lpml} only achieves a better time performance in 3 domains(blocks, parcprinter and visitall), in other domains, LM-cut is faster at solving tasks which can be solved by both heuristics. This suggests that h^{lpml} is more suitable to solve difficult tasks, i.e., tasks can't be solved by LM-cut, However, note that the search time performance of our heuristic would be greatly improved if a faster linear programming solver is used. The reason is that we usually expand tens of thousands of search nodes for a typical planning task, and every node expansion will involve solving a linear program. A faster LP solver, will greatly improve the search time performance of h^{lpml}.

8. CONCLUSIONS

In this paper, we introduce multi-valued landmarks to model cardinality constraints in cost optimal planning. With multi-valued action landmarks, we are able to construct complete landmark sets for general planning tasks. As general multi-value action landmarks are hard to extract and exploit, we introduce multi-valued proposition landmarks from which multi-valued action landmarks can be efficiently extracted. For a certain class of propositions called regular transition propositions, we show there is a close relationship between the number of times they are added and the number of times they are deleted. Using this relation, we devise a linear programming based multi-valued landmarks heuristic h^{lpml} for cost optimal planning. h^{lpml} is guaranteed to be admissible and can be computed in polynomial time. Experiment evaluations show that our heuristic outperforms state-of-the-art admissible heuristic like LM-cut in terms of heuristic accuracy on benchmark domains from

	initial h(I) value		node expansion		total search time(sec)		memory(KB)	
domains	LM-cut	lpml	LM-cut	lpml	LM-cut	lpml	LM-cut	lpml
barman-opt11-strips (4)	141	**194**	5067552	**3742918**	**1835.6**	5915.03	1117012	**769504**
blocks (28)	410	**448**	946421	**172400**	737.28	**735.3**	477556	**223832**
elevators-opt11-strips (15)	501	**509**	430617	**355774**	**1650.55**	4732	619136	**560304**
floortile-opt11-strips (6)	207	**226**	861904	**253679**	**405.44**	886.16	463424	**172012**
nomystery-opt11-strips (12)	182	**188**	6300	**3813**	**76.43**	826.91	**88960**	114652
openstacks-opt11-strips (10)	10	10	5559961	5559961	**2291.42**	3039.16	**1576096**	1586948
parcprinter-opt11-strips (13)	11165236	**11547174**	299623	**464**	241.33	**2.72**	291844	**74800**
parking-opt11-strips (1)	9	9	3391	**3106**	**94.08**	214.31	**16344**	19984
pegsol-opt11-strips (16)	39	**53**	1325521	**671736**	**199.34**	1564.24	419388	**270304**
scanalyzer-opt11-strips (10)	308	**309**	20034	**9106**	**458.9**	1814.39	**209012**	409940
sokoban-opt11-strips (17)	174	**195**	1138432	**836751**	**252.47**	3956.15	276112	**250116**
tidybot-opt11-strips (6)	100	100	5720	**5378**	**85.13**	2682.27	**167412**	262900
transport-opt11-strips (6)	1734	**1919**	56246	**38174**	**223.39**	599.2	73532	**71328**
visitall-opt11-strips (10)	104	**142**	591654	**217**	184.79	**1.35**	2296180	**58252**
woodworking-opt11-strips (10)	2210	**2270**	208731	**148226**	**749.67**	1921.34	376160	**294916**
sum	11171365	**11553746**	16522107	**11801703**	**9485.82**	28890.53	8468168	**5139792**

Table 3: Detailed performance comparison of LM-cut and h^{lpml} on planning tasks that are solved by both heuristics, bold results indicate better performance

recent international planning competitions. Compared with LM-cut based planner, the planner based on h^{lpml} achieves better overall coverage performance at the cost of using more CPU time. This suggests cost optimal planning could benefit from more accurate and efficient multi-valued landmark heuristics.

Acknowledgements

The work is supported by national natural science foundation of China under grant no. 61021062, 61105069, 61321491, 973 program 2011CB505300 and Jiangsu key technology research and development program BE2011171, BE2012161.

9. REFERENCES

[1] B. Bonet and J. Castillo. A complete algorithm for generating landmarks. In *ICAPS*, 2011.

[2] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129:5–33, 2001.

[3] B. Bonet and M. Helmert. Strengthening Landmark Heuristics via Hitting Sets. In *European Conference on Artificial Intelligence*, pages 329–334, 2010.

[4] J. Davies and F. Bacchus. Solving maxsat by solving a sequence of simpler sat instances. In *Proceedings of the 17th international conference on Principles and practice of constraint programming*, CP'11, pages 225–239, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] P. Haslum. $h^m(p) = h^1(p^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*, 2009.

[6] P. Haslum, J. Slaney, and S. Thiébaux. Incremental lower bounds for additive cost planning problems. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 2012.

[7] M. Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[8] M. Helmert and C. Domshlak. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *International Conference on Automated Planning and Scheduling/Artificial Intelligence Planning Systems*, 2009.

[9] J. Hoffmann, J. Porteous, and L. Sebastia. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.

[10] E. Karpas and C. Domshlak. Cost-Optimal Planning with Landmarks. In *International Joint Conference on Artificial Intelligence*, pages 1728–1733, 2009.

[11] E. Keyder, J. Hoffmann, P. Haslum, et al. Semi-relaxed plan heuristics. *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, pages 128–136, 2012.

[12] E. Keyder, S. Richter, and M. Helmert. Sound and complete landmarks for and/or graphs. In *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 335–340, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[13] J. Porteous and S. Cresswell. Extending landmarks analysis to reason about resources and repetition. In *Proceedings of the 21st Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG-02)*, pages 45–54, 2002.

[14] J. Porteous, L. Sebastia, and J. Hoffmann. On the Extraction, Ordering, and Usage of Landmarks in Planning. In *European Conference an Planning*, 2001.

[15] S. Richter and M. Westphal. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research*, 39:127–177, 2010.

[16] L. Zhang and F. Bacchus. Maxsat heuristics for cost optimal planning. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI-2012)*, 2012.