Strengthening Landmark Heuristics via Hitting Sets

Blai Bonet[1] and **Malte Helmert**[2]

Abstract. The landmark cut heuristic is perhaps the strongest known polytime admissible approximation of the optimal delete relaxation heuristic h^+. Equipped with this heuristic, a best-first search was able to optimally solve 40% more benchmark problems than the winners of the sequential optimization track of IPC 2008. We show that this heuristic can be understood as a simple relaxation of a hitting set problem, and that stronger heuristics can be obtained by considering stronger relaxations. Based on these findings, we propose a simple polytime method for obtaining heuristics stronger than landmark cut, and evaluate them over benchmark problems. We also show that hitting sets can be used to characterize h^+ and thus provide a fresh and novel insight for better comprehension of the delete relaxation.

1 INTRODUCTION

Many admissible heuristics for classical planning are closely linked to the idea of *delete relaxation*, including several heuristics that are ostensibly based on very different ideas. For example, Helmert and Domshlak [10] showed that Karpas and Domshlak's [13] admissible landmark heuristics can be seen as a variation of the additive $h^{\max}$ heuristic [8], which is a delete relaxation heuristic, and Haslum [7] proved that the h^m heuristic family [9] can be polynomially reduced to Bonet and Geffner's [2] $h^{\max}$ heuristic on a modified planning task.

Delete relaxation heuristics strive to get as closely as possible to the *optimal delete relaxation heuristic* h^+ [11], which dominates all heuristics based on delete relaxation, but is NP-hard to compute [4] or approximate within a constant factor [1]. The landmark cut heuristic $h^{\text{LM-cut}}$, LM-cut for short, is a recent admissible delete relaxation heuristic that provides one of the best-known polytime approximations of h^+ [10]. In an experiment designed to measure the accuracy of different heuristics with respect to h^+, heuristic values were computed for the initial states of 505 tasks from 22 domains of the International Planning Competition (IPC). The heuristics considered were $h^{\max}$ [2], the original additive $h^{\max}$ [8], the additive $h^{\max}$ of Coles et al. [5], the admissible landmark heuristic h^{LA} [13], and LM-cut, which are all admissible delete relaxation heuristics and hence bounded from above by h^+. The results of the experiment were astonishing. The average additive errors of the heuristics compared to h^+ were 27.99, 17.37, 8.05, 1.94 and 0.28 respectively, and the relative errors were 68.5%, 40.9%, 25.2%, 9.5% and 2.5%, respectively. This shows that LM-cut is on average far more accurate than the other heuristics, improving on the second most accurate heuristic by a factor of 6.9 in terms of additive error and 3.8 in terms of relative error. For more than 70% of the instances, LM-cut computed the exact h^+ value.

In another experiment, Helmert and Domshlak measured the overall performance of an optimal planner equipped with LM-cut against other optimal state-of-the-art planners and showed that despite the high computational cost of the heuristic, the LM-cut-based planner was faster and solved many more tasks than the others.

These results raise the question of whether one could define a heuristic better than LM-cut that is based on delete relaxation and is polynomial-time computable. In this paper, we answer this question in the affirmative and in a very strong way: first, we not only provide a new heuristic but a general method for obtaining strong landmark heuristics that are admissible and computable in polynomial time; and second, we show that a simple instantiation of this method produces heuristics that *provably dominate* LM-cut.

These are strong theoretical guarantees, yet a thorough empirical study is still needed in order to measure the accuracy and performance of the new heuristics. As a first step, we present preliminary empirical results that show that the novel heuristics expand considerably fewer nodes on the instances where LM-cut is less accurate and that in a few cases the overall running time is also smaller.

The novel heuristics are obtained by optimally solving a relaxation of a hitting set problem induced by a collection of landmarks. Furthermore, we show that the hitting set problem characterizes the heuristic h^+, and that the optimal cost partitioning for a collection of landmarks [14] is closely related to a well-known relaxation of the hitting set problem induced by the landmarks.

In the next section, we perform a study of LM-cut that identifies its main limitations. Section 3 presents the hitting set problem, some of its properties, and notions of decomposition and width, and Sect. 4 contains a general treatment of landmarks, the hitting set problems induced by landmarks, and fundamental results characterizing h^+ and optimal cost partitionings. Then, in Sect. 5, we show that LM-cut is a relaxation of a hitting set problem and give a general method to improve it by considering stronger relaxations. Finally, we present the results of some preliminary experiments and conclude.

2 ASSESSMENT OF THE LM-CUT HEURISTIC

In order to design improved heuristics, it is imperative to understand the "deficiencies" or "flaws" of LM-cut. So far, we have observed two deficiencies. The first one is intrinsic to the definition of LM-cut, while the second is related to its implementation as LM-cut is not fully specified. We begin by illustrating the first flaw with a simple example, a planning task with fluents $F = \{s, t, q_1, q_2, q_3\}$ and operators $O = \{a_1, a_2, a_3, \mathit{fin}\}$ given by: $a_1 : s \Rightarrow q_1, q_2$, $a_2 : s \Rightarrow q_1, q_3$, $a_3 : s \Rightarrow q_2, q_3$ and $\mathit{fin} : q_1, q_2, q_3 \Rightarrow t$, where $a_1 : s \Rightarrow q_1, q_2$ means that a_1 has precondition s and adds q_1 and q_2. The initial state and goal are $I = \{s\}$ and $G = \{t\}$, and the costs are $c(a_1) = c(a_2) = c(a_3) = 1$ and $c(\mathit{fin}) = 0$.

LM-cut is defined in terms of justification graphs that are associated with what we call *precondition-choice functions*. A precondition-choice function (pcf) is a function D that maps each ac-

[1] Universidad Simón Bolívar, Venezuela, bonet@ldc.usb.ve
[2] Albert-Ludwigs-Universität Freiburg, Germany, helmert@informatik.uni-freiburg.de

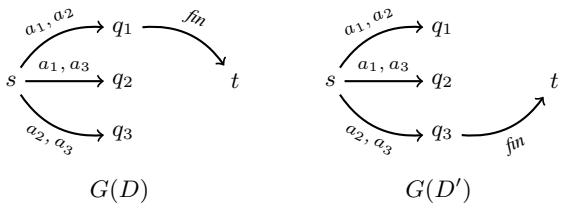

Figure 1. Justification graphs for the pcfs $D = \{a_1 \mapsto s, a_2 \mapsto s, a_3 \mapsto s, \mathit{fin} \mapsto q_1\}$ and $D' = D[\mathit{fin} \mapsto q_3]$.

tion into one of its preconditions. (We assume that each action has at least one precondition.) For example, $D = \{a_1 \mapsto s, a_2 \mapsto s, a_3 \mapsto s, \mathit{fin} \mapsto q_1\}$ is a pcf for the example. The justification graph for D, denoted by $G(D)$, is a directed graph whose vertices are fluents and which has an edge (p, q) labeled with a iff the action a adds q and $D(a) = p$. Figure 1 shows two justification graphs for the example. In the left graph, the pair of subsets $(\{s, q_2, q_3\}, \{q_1, t\})$ is an s-t-cut (a partition of the vertices into two sets that separates s from t) whose cut-set (edges that *cross* from the set containing s to the set containing t) is $\{s \xrightarrow{a_1} q_1, s \xrightarrow{a_2} q_1\}$. The set $\{a_1, a_2\}$ of labels for the cut-set forms a *landmark* for the planning task, meaning that every plan must contain at least one of its actions. Therefore, every plan for the example must incur a cost of at least 1.

In this example, there are 3 different pcfs depending on the precondition chosen for the action *fin*. The landmarks inferred by LM-cut in each case are $\{a_1, a_2\}$, $\{a_1, a_3\}$ and $\{a_2, a_3\}$, implying that a relaxed plan must contain a_1 or a_2, and a_1 or a_3, and a_2 or a_3; i.e., a relaxed plan must contain at least two actions from $\{a_1, a_2, a_3\}$ and thus its cost is at least 2. Unfortunately, LM-cut is not capable of making this inference since as soon as it resolves one landmark, say $\{a_1, a_2\}$, it assigns zero cost to the actions in it, which "resolves" the other two landmarks as a side effect. Thus, in this case, LM-cut computes a value of 1 instead of the exact h^+ value of 2; e.g., $\langle a_1, a_2, \mathit{fin}\rangle$ is an optimal plan with cost 2. The example can be enlarged in order to bound the value of LM-cut as far away from h^+ as one would like.

This example reveals the first flaw of LM-cut of only considering one landmark at a time. We can improve over LM-cut by taking a more global perspective, considering the collection of landmarks $\{\{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}\}$ as an instance of a *hitting set problem*, for which we can compute a minimum-cost solution.

Another way to combine information about multiple landmarks is optimal cost partitioning, introduced by Katz and Domshlak [14] and applied to landmark heuristics by Karpas and Domshlak [13]. For the given example, optimal cost partitioning over the set of all landmarks increases the heuristic value to $\frac{3}{2}$, which improves on LM-cut but does not close the gap to h^+ entirely. We will get back to the relationship between cost partitioning and hitting sets later.

The second flaw of LM-cut refers to the way that justification graphs are computed. The justification graphs of LM-cut are defined by pcfs that assign to each action a precondition with maximum $h^{\max}$ value, breaking ties in an unspecified manner. In our experiments, we have observed that the accuracy of LM-cut varies with this choice, sometimes significantly.

In the rest of the paper, we address these flaws in a principled way to obtain more informative heuristics and formal results on their quality. Throughout the paper, we consider delete relaxation heuristics, which are based on the notion of *relaxed plans* (plans for delete-free STRIPS problems). Such relaxed plans never need to repeat actions and thus can be represented by the set of actions they use (e.g. $\{a_1, a_2, \mathit{fin}\}$). From such a set, the plan can be recovered in polynomial time. We will show that this set notation for relaxed plans is closely related to the notion of *hitting sets*, which we introduce next.

3 HITTING SETS

Let $A = \{a_1, \ldots, a_n\}$ be a set and $\mathcal{F} = \{F_1, \ldots, F_m\}$ a family of subsets of A. A subset $H \subseteq A$ has the hitting set property, or is a hitting set, iff $H \cap F_i \neq \emptyset$ for all $1 \leq i \leq m$ (i.e., H "hits" each set F_i). If we are given a cost function $c : A \to \mathbb{N}$, the cost of H is $\sum_{a \in H} c(a)$. A hitting set is of minimum cost if its cost is minimal among all hitting sets.

The problem of finding a minimum-cost hitting set for family $\mathcal{F}$ and cost function c is denoted by $\langle \mathcal{F}, c\rangle$, and the cost of its solution by $\min(\mathcal{F}, c)$. A relaxation for $\langle \mathcal{F}, c\rangle$ is a problem $\langle \mathcal{F}', c'\rangle$ such that $c' \leq c$, and for all $F' \in \mathcal{F}'$ there is $F \in \mathcal{F}$ with $F \subseteq F'$. In words, $\langle \mathcal{F}, c\rangle$ can be relaxed by reducing costs, dropping sets from $\mathcal{F}$, or enlarging elements of $\mathcal{F}$. Determining the existence of a hitting set for a given cost bound is a classic problem in computer science, one of the first problems to be shown NP-complete [12].

Lemma 1. *If $\langle \mathcal{F}', c'\rangle$ is a relaxation of $\langle \mathcal{F}, c\rangle$, then $\min(\mathcal{F}', c') \leq \min(\mathcal{F}, c)$. Furthermore, if $\{\langle \mathcal{F}_i, c_i\rangle\}$ is a collection of relaxations of $\mathcal{F}$ such that $\sum_i c_i \leq c$, then $\sum_i \min(\mathcal{F}_i, c_i) \leq \min(\mathcal{F}, c)$.*

Proof. For lack of space, we refer to a technical report [3]. □

A collection of relaxations as defined in Lemma 1 is called an *additive relaxation* of the hitting set problem.

It is well known that a minimum-cost hitting set for $\langle \mathcal{F}, c\rangle$ can be found by solving the Integer Linear Program (ILP) defined by the $\{0, 1\}$-variables $\{x_a : a \in A\}$, the objective $\min \sum_{a \in A} x_a c(a)$, and the constraints $\sum_{a \in F} x_a \geq 1$ for each $F \in \mathcal{F}$. The solution of this ILP has cost equal to $\min(\mathcal{F}, c)$, and its LP relaxation has solution cost less than or equal to $\min(\mathcal{F}, c)$.

Decomposition and Width

We now define the width of hitting set problems and show that the time complexity of solving problems of bounded (or constant) width is polynomial. This result will play a fundamental role later.

Let $\mathcal{F}$ be a family of subsets of A that can be partitioned into $\Pi = \{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ satisfying $(\bigcup \mathcal{F}_i) \cap (\bigcup \mathcal{F}_j) = \emptyset$ for all $i \neq j$; i.e., the blocks in the partition are pairwise *independent*. Then, for any cost function c, $\min(\mathcal{F}, c) = \sum_{i=1}^{m} \min(\mathcal{F}_i, c)$ and the problem of finding a minimum-cost hitting set for $\mathcal{F}$ can be decomposed into smaller subproblems. We call the maximum size of a block in Π the *width* of Π, and define the *width* of $\mathcal{F}$, denoted by *width*$(\mathcal{F})$, as the minimum width of Π over all partitions Π of $\mathcal{F}$ into independent blocks. Finding a partition that minimizes the width is an easy problem similar to computing connected components of a graph.

Let $\mathcal{F} = \{F_1, \ldots, F_k\}$ be a family over A with k subsets, but with no assumptions on the sizes of each F_i or A. We show that $\min(\mathcal{F}, c)$ and a hitting set achieving this cost can be computed in time bounded by $O(\|\mathcal{F}\| + k4^k)$, where $\|\mathcal{F}\|$ refers to the input size for $\mathcal{F}$. To see this, consider the hypergraph $H_{\mathcal{F}} = (X, E)$ where $X = \{1, \ldots, k\}$ and there is a hyperedge $e(a) = \{i : a \in F_i\}$ with cost $c(a)$ for each $a \in A$. The hitting sets for $\mathcal{F}$ are in one-to-one correspondence with the covers of $H_{\mathcal{F}}$ (a cover is a set of hyperedges that "touch" every vertex). Hence, finding $\min(\mathcal{F}, c)$ is equivalent to finding a minimum-cost cover for $H_{\mathcal{F}}$. For the latter, observe that all hyperedges $e(a)$ for which there is a hyperedge $e(a')$ with $e(a) =$

$e(a')$ and $c(a') < c(a)$ may be removed (and if $c(a') = c(a)$, only one of the hyperedges needs to be kept).

Since a hyperedge is a subset of X, this implies that we only need to consider hypergraphs with at most 2^k edges. Using dynamic programming, a minimum cost cover for such a hypergraph can be found in time $O(k4^k)$. Combining this with the time required for constructing the hypergraph yields the overall $O(\|\mathcal{F}\| + k4^k)$ bound.

Theorem 2. *The problem of computing* $\min(\mathcal{F}, c)$ *is fixed-parameter tractable when considering the width of* $\mathcal{F}$ *as the parameter. In particular, for any* fixed *bound* k, $\min(\mathcal{F}, c)$ *for families of width at most* k *can be computed in linear time.*

The term $k4^k$ is a theoretical worst-case limit. In practice, one can use a branch-and-bound or best-first search to find optimal covers. In our experiments, we have solved problems of width up to 15.

4 LANDMARKS

Having provided the necessary background on hitting sets, we now define the planning framework and the concept of delete relaxation, in order to present the hitting set problem that defines h^+.

A STRIPS problem with action costs is a tuple $P = \langle F, O, I, G, c\rangle$ where F is the set of fluents, O is the set of actions or operators, I and G are the initial state and goal description, and $c : O \to \mathbb{N}$ is the cost function. We are interested in delete relaxations, so we assume that the operators have empty delete lists, and thus "plan" and "relaxed plan" shall denote the same. For a definition of the basic concepts underlying delete relaxations, such as the $h^{\max}$ and h^+ functions, we refer to the literature [10]. We also assume from now on that all fluents have finite $h^{\max}$ values, which implies that the problem has finite h^+ value. As additional simplifying assumptions, we require that all operators have nonempty preconditions, that there are two fluents $s, t \in F$ such that $I = \{s\}$ and $G = \{t\}$, and that there is a unique operator *fin* that adds t. When these simplifying assumptions are not met, they can be achieved through simple linear-time transformations. We denote the precondition and effects of $a \in O$ by $\mathrm{pre}(a)$ and $\mathrm{post}(a)$. The h^+ value for state I is denoted by $h^+(P)$.

An (action) *landmark* for P is a set $\{a_1, \ldots, a_n\}$ of actions such that every plan for P must contain at least one such action.

Recall that a pcf D assigns a precondition $D(a) \in \mathrm{pre}(a)$ to each action $a \in O$. Our first result relates cuts of the justification graph $G(D)$ with landmarks for P.

Lemma 3. *Let* D *be a pcf and* C *an* s-t*-cut of* $G(D)$*. Then, the labels of the edges in the cut-set of* C *form a landmark.*

Proof. A relaxed plan defines an s-t-path on $G(D)$ that must cross every s-t-cut. □

Given a pcf D, we denote the set of landmarks associated with the cut-sets of $G(D)$ by $\mathrm{Landmarks}(D)$. By considering all pcfs and all cuts in the justification graphs, we obtain the hitting set problem $\mathcal{F}_L \doteq \bigcup\{\mathrm{Landmarks}(D) : D \text{ is a precondition-choice function}\}$.

Theorem 4. *If* H *is a plan for* P*, then* H *is a hitting set for* $\mathcal{F}_L$*. Conversely, if* H *is a hitting set for* $\mathcal{F}_L$*, then* H *contains a plan for* P*. Therefore,* $\min(\mathcal{F}_L, c) = h^+(P)$.

Proof. The first claim is direct, since by Lemma 3, every element of $\mathcal{F}_L$ is hit by every plan. The last claim follows from the first two.

For the second claim, let H be a hitting set for $\mathcal{F}_L$ and let R be the set of fluents that can be reached by only using operators in H. If R contains the goal t, then H contains a plan and there is nothing to prove. So, assume $t \notin R$. We construct a pcf D such that $G(D)$ contains an s-t-cut whose cut-set is not hit by H, thus reaching a contradiction. We classify operators into three types and define D:

T1. If $\mathrm{pre}(a) \subseteq R$ and $\mathrm{post}(a) \subseteq R$, then set $D(a)$ arbitrarily to some $p \in \mathrm{pre}(a)$.
T2. If $\mathrm{pre}(a) \subseteq R$ and $\mathrm{post}(a) \not\subseteq R$, then set $D(a)$ arbitrarily to some $p \in \mathrm{pre}(a)$.
T3. If $\mathrm{pre}(a) \not\subseteq R$, then set $D(a)$ to some $p \in \mathrm{pre}(a) \setminus R$.

Now consider the cut (R, R^{c}) of $G(D)$, where R^{c} is the set of all fluents not in R. It is a cut since $s \in R$ and $t \notin R$. We show that H does not hit the cut-set, i.e., there exists no operator $a \in H$ that labels an edge going from some fluent in R to some fluent not in R.

Assume that $a \in H$ were such an operator. It cannot be of type T1, because edges labeled by type T1 operators go from R into R. It cannot be of type T2, because $\mathrm{pre}(a) \subseteq R$ and $a \in H$ implies $\mathrm{post}(a) \subseteq R$ (by definition of R). Finally, it cannot be of type T3, as edges labeled by type T3 operators do not start in R. Hence, no such operator exists. □

In practice, computing $\mathcal{F}_L$ according to the definition above is infeasible because there are usually exponentially many pcfs. However, if we can compute and solve, in polynomial time, a relaxation of $\mathcal{F}_L$, then this provides a polytime admissible approximation of h^+.

Corollary 5. *Let* $\langle \mathcal{F}, c'\rangle$ *be a polynomial-time computable relaxation of* $\langle \mathcal{F}_L, c\rangle$ *(possibly additive*[3]*) whose solution is polynomial-time computable. Then the heuristic* $h = \min(\mathcal{F}, c')$ *is a polytime admissible approximation of* h^+.

An important special case covered by the corollary are landmark heuristics based on cost partitioning, including LM-cut (see below) and the heuristics of Karpas and Domshlak [13]. In general, given a set $\mathcal{L} = \{L_1, \ldots, L_n\}$ of landmarks, a cost partitioning for $\mathcal{L}$ is a collection $\mathcal{C} = \{c_1, \ldots, c_n\}$ of cost functions such that $\sum_{i=1}^n c_i(a) \leq c(a)$ for each action a. The partitioning defines the heuristic $h_{\mathcal{C}} \doteq \sum_{i=1}^n \min_{a \in L_i} c_i(a)$, which is an additive relaxation of $\mathcal{F}_L$ when $\mathcal{L} \subseteq \mathcal{F}_L$.

Karpas and Domshlak studied *uniform* cost partitioning, defined as $c_i(a) \doteq 0$ if $a \notin L_i$ and $c_i(a) \doteq c(a)/|\{i : a \in L_i\}|$ if $a \in L_i$, and *optimal* cost partitioning, which maximizes $h_{\mathcal{C}}$ through linear programming (LP). Interestingly, there is a close connection between the optimal cost partitioning LP and the hitting set ILP for $\mathcal{L}$.

Theorem 6. *Let* $\mathcal{L}$ *be a collection of landmarks, and let* c *be the cost function for the actions. Then, the LP that defines the optimal cost partitioning is the* dual *of the LP relaxation of the ILP for* $\langle \mathcal{L}, c\rangle$.

Proof. See the technical report [3]. □

5 THE LM-CUT HEURISTIC

LM-cut is a cost-partitioning-based landmark heuristic obtained from a sequence $\{(L_i, c_i)\}_{i=1}^n$ of landmarks and cost functions such that

$$h^{\text{LM-cut}}(P) = h(L_1, c_1) + h(L_2, c_2) + \cdots + h(L_n, c_n)\,.$$

In this expression, $h(L, c)$ is the landmark heuristic for L and cost function c that satisfies $h(L, c) = \min_{a \in L} c(a)$.

The sequence $\{(L_i, c_i)\}_{i=1}^n$ is computed iteratively, in stages, as follows. Initially, $c_1 = c$, and at stage i, the landmark L_i and cost function c_i are computed through the following steps:

[3] In the additive case, we slightly abuse notation since $\langle \mathcal{F}, c'\rangle$ should be replaced by a collection $\{\langle \mathcal{F}_i, c_i\rangle\}_i$.

Step 1. Compute $h^{\max}_{c_i}(p)$ values for every fluent p. Terminate if $h^{\max}_{c_i}(t) = 0$.
Step 2. Modify the operators by keeping just one fluent in the precondition of each operator: a fluent that maximizes $h^{\max}_{c_i}$, breaking ties arbitrarily. After this step, each action has exactly one precondition.
Step 3. Split each action of the form $a : p \Rightarrow q_1, \ldots, q_k$ into k actions of the form $a : p \Rightarrow q_i$.
Step 4. Construct the justification graph G_i, whose vertices are the fluents and which contains, for every action $a : p \Rightarrow q$, an edge from p to q with cost $c_i(a)$ and label a.
Step 5. Construct an s-t-cut $C_i = (V_i^0, V_i^* \cup V_i^{\mathrm{b}})$ as follows: V_i^* contains all fluents from which t can be reached through a zero-cost path, V_i^0 contains all fluents reachable from s without passing through some fluent in V_i^*, and V_i^{b} contains all remaining fluents. Clearly, $t \in V_i^*$ and $s \in V_i^0$.
Step 6. The landmark L_i is the set of labels of the edges that cross the cut C_i (i.e., lead from V_i^0 to V_i^*).
Step 7. Let $m_i \doteq \min_{a \in L_i} c_i(a)$, and define $c_{i+1}(a) \doteq c_i(a)$ if $a \notin L_i$, and $c_{i+1}(a) \doteq c_i(a) - m_i$ if $a \in L_i$.

Step 2 defines a pcf D that is used in Step 4 to construct the justification graph $G(D)$. Each landmark L_i is the set of labels of a s-t-cut-set of $G(D)$. Hence, if the cost functions correspond to a cost partitioning for $\mathcal{L} = \{L_1, \ldots, L_n\}$, then LM-cut is an additive relaxation of $\mathcal{F}_L$. This is indeed the case:

Theorem 7. *LM-cut is an additive relaxation of F_L.*

Proof. It is sufficient to show $\sum_i c_i(a) \leq c(a)$ for all $a \in \bigcup \mathcal{L}$. The (not difficult) proof is given in the technical report [3]. □

We now focus on problems with action costs that are either 0 or 1. This is an important class that contains all STRIPS problems.

Improving LM-cut for Problems with 0/1 Costs

The main limitation of LM-cut is that it only considers very simple subproblems at each stage, namely $\mathcal{F}_i = \{L_i\}$. For 0/1 cost functions, the landmarks $\{L_i\}_{i=1}^n$ computed by LM-cut satisfy the decomposition $L_i \cap L_j = \emptyset$ for all $i \neq j$ (i.e., all computed landmarks are disjoint), and hence the family $\mathcal{F}_{\text{LM-cut}} \doteq \{L_i\}_{i=1}^n$ has width 1.

Our goal is to define a method for obtaining polytime heuristics stronger than LM-cut. The general idea is to iteratively construct a family $\mathcal{F}$ that contains $\mathcal{F}_{\text{LM-cut}}$ and is contained in $\mathcal{F}_L$ while assuring its polytime solvability.

We start from $\mathcal{F} := \mathcal{F}_{\text{LM-cut}}$ and grow $\mathcal{F}$ by adding one landmark at a time. We assume that there is a "stream of landmarks" from which one can iteratively obtain a next landmark to consider for inclusion in $\mathcal{F}$. The complexity of solving $\mathcal{F}$ is controlled by controlling its width. Thus, for each landmark L from the stream, L is added to $\mathcal{F}$ if $\mathit{width}(\mathcal{F} \cup \{L\}) \leq k$, where k is a parameter of the algorithm. Additionally, we implement two simple dominance tests to ensure that $\mathcal{F}$ never contains two landmarks L_1 and L_2 with $L_1 \subset L_2$. (In such a situation, L_2 would carry no information since every set that hits L_1 also hits L_2, and removing L_2 from $\mathcal{F}$ might reduce its width.) Figure 2 shows the algorithm for growing $\mathcal{F}$. This is a general algorithm that will produce a heuristic that dominates LM-cut given any method for generating a stream of landmarks.

In this paper, we implement a simple stream. Recall that LM-cut does not specify how to break ties when selecting preconditions (Step 2), and often there is high variability on the quality of LM-cut

```
Construct-Relaxation(k)
1. F := F_LM-cut;
2. foreach L from stream do
3.    if there is L' ∈ F such that L' ⊆ L then
4.       continue
5.    elsif there is L' ∈ F such that L ⊆ L' then
6.       F := (F \ {L'}) ∪ {L};
7.    elsif width(F ∪ {L}) ≤ k then
8.       F := F ∪ {L};
9. return F;
```

Figure 2. General algorithm for growing $\mathcal{F}_{\text{LM-cut}}$.

with respect to this choice. Thus, we generate the stream of landmarks using the same LM-cut loop but breaking ties randomly among preconditions with maximum $h^{\max}$ value. The total number of new landmarks is controlled with another parameter $p \geq 1$ that specifies how many "passes" of the LM-cut loop are executed. In order to initialize $\mathcal{F}$ to $\mathcal{F}_{\text{LM-cut}}$, the first pass uses the tie-breaking criterion used in the original LM-cut implementation.

Computing the width of a family amounts to finding a suitable partition of it. Since this operation is performed multiple times, an efficient representation is required.

Efficient Representation

Let $\mathcal{F}$ be a family over A decomposed into $\Pi = \{\mathcal{F}_i\}_{i=1}^n$ where each $\mathcal{F}_i$ is a family over $A_i = \bigcup \mathcal{F}_i$. By definition, $A_i \cap A_j = \emptyset$ for $i \neq j$. In order to add landmarks while controlling the width, we shall need four basic operations:

- `Find(`a`)`: find A_i that contains a.
- `Union(`a_i, a_j`)`: compute $A_i \cup A_j$ where $a_i \in A_i$ and $a_j \in A_j$.
- `Add(`L`)`: add landmark L to $\mathcal{F}$.
- `Width(`L`)`: return the width that would result from adding L.

The partition $\{A_i\}_{i=1}^n$ of A is stored using a disjoint-set data structure that permits efficient union/find operations [6]. In detail, the partition is stored as a forest of trees in which each A_i corresponds to a tree whose root is the element of A_i that *represents* it. In this data structure, each $a \in A$ has a reference $a.\mathit{parent}$ that points to its parent in the tree or to itself if a is a root. `Find(`a`)` returns the root of the tree that contains a by moving along the parent pointers up to the root of the tree, and `Union(`a_i, a_j`)` sets the parent of `Find(`a_j`)` to `Find(`a_i`)`. The `Find(`a`)` and `Union(`a_i, a_j`)` procedures can be enhanced by performing *path compression* and *union by rank* [6]. With these enhancements, all `Find` and `Union` operations take constant amortized time for all practical values of $|A|$.

The representatives of the sets A_i are used to store the landmarks by using an array F of linked lists: $F[a]$ is the list of landmarks in $\mathcal{F}_i$ when a is the representative of A_i. Thus, all landmarks "related" to an element a can be found by listing $F[$`Find(`a`)`$]$. The operations `Add(`L`)` and `Width(`L`)` can be implemented in $O(|L|)$ time using the first two operations. Figure 3 depicts the pseudo-code for an initialization procedure, for `Add(`L`)` and for `Width(`L`)`.

We denote the family of landmarks obtained using the above method for parameters p and k by $\mathcal{F}_{p,k}$. Similarly to $h^{\text{LM-cut}}$, this family is not fully specified since it depends on the exact criteria for selecting the precondition of each action during each pass of LM-cut;

```
Initialize(F, A)
1. foreach a ∈ A do a.parent := a; F[a] := ∅;
2. foreach L ∈ F do Add(L);

Add(L = {a_1, ..., a_m})
1. for i = 2 to m do Union(a_1, a_i);
2. a := Find(a_1);
3. F[a] := F[a] ∪ {L};

Width(L = {a_1, ..., a_m})
1. for i = 1 to m do mark Find(a_i);
2. width := 1;
3. for i = 1 to m do
4.    if Find(a_i) is marked then
5.       width := width + |F[Find(a_i)]|;
6.       unmark Find(a_i);
7. return width;
```

Figure 3. Basic algorithms for manipulating families $\mathcal{F}$.

we assume that some criterion is given. We then define the (p, k)-LM-cut heuristic $h^{\text{LM-cut}}_{p,k}$ as $\min(\mathcal{F}_{p,k}, c)$. With this definition, the $(1, 1)$-LM-cut heuristic equals the LM-cut heuristic, while the other heuristics provide estimates which are at least as large.

Theorem 8. *For any fixed $p \geq 1$ and $k \geq 1$, $h^{\text{LM-cut}}_{p,k}$ is computable in polynomial time and dominates $h^{\text{LM-cut}}$.*

6 PRELIMINARY EXPERIMENTS

We implemented the (p, k)-LM-cut heuristic within the framework of the optimal Fast Downward planner and ran preliminary experiments over 86 problems: 67 from domains considered challenging for LM-cut, and 19 from other domains. A domain was deemed challenging when the relative error of LM-cut with respect to h^+ for the initial states of the problems in the domain was more than 10% (cf. [10]). The challenging domains are Pipesworld-notankage (17), Pipesworld-tankage (11), Openstacks (7), Mystery (17) and Freecell (15). The other domains are Satellite (9) and Trucks (10).

The evaluation consisted in optimally solving each problem with $h^{\text{LM-cut}}_{p,k}$ (performing p passes of the LM-cut loop and constructing a hitting set problem $\mathcal{F}_{p,k}$ of width at most k). In our implementation, each pass of LM-cut not only generates a stream of landmarks, but also the LM-cut estimate for the pcfs used during that pass. The maximum of the LM-cut values of all p passes defines another heuristic that we call the Max LM-cut heuristic, denoted by $\max h^{\text{LM-cut}}_p$. The value for $\max h^{\text{LM-cut}}_p$ may differ from the value of LM-cut only when there are actions with more than one precondition with maximum $h^{\max}$ value; in such case, we say that the action has more than one $h^{\max}$ supporter. In our experiments, all actions in the most difficult problems for all domains, except Mystery, have multiple $h^{\max}$ supporters, and their number ranges from a typical value of 2 to 10 in some cases. Since the computation of $\max h^{\text{LM-cut}}_p$ causes negligible overhead, we *enhance* $h^{\text{LM-cut}}_{p,k}$ by taking the maximum with $\max h^{\text{LM-cut}}_p$.

The hitting set problems of width at most k are solved with a breadth-first search that prunes duplicates but performs no node ordering. This simple search is able to solve all the hitting set problems for values of k up to 15 that were generated when solving the 86 problems; these are millions of hitting set problems.

The experiments were run on machines with 2.3 GHz Opteron CPUs under a 2 GB memory limit. For each problem in the benchmark set, we evaluated LM-cut, $\max h^{\text{LM-cut}}_p$ and $h^{\text{LM-cut}}_{p,k}$ for $p = 3, 4, 5$ and $k = 5, 10, 15$, measuring the number of node expansions up to the last f layer. (We prefer this measure to the overall number of node expansions because it is not affected by tie-breaking behavior of the A* implementation.) Using this measure, we report the absolute expansion numbers for LM-cut and the *percentage of reduction* of the expansion numbers compared to LM-cut for the other heuristics.

Table 1 presents results for problems in which the reduction was *at least 50%* for at least one of our improved heuristics; e.g., the entry #2-5 of Freecell shows that $h^{\text{LM-cut}}_{5,15}$ expanded 74% fewer nodes than LM-cut; i.e., 72 nodes instead of 277 since $(277 - 72)/277 = 0.74$.

We can draw several conclusions from the data. First, the values of LM-cut vary considerably when different tie-breaking rules are used to select $h^{\max}$ supporters. This is reflected in the number of expanded nodes for the Max LM-cut heuristic, which is sometimes much lower than the number for LM-cut. For example, in Pipesworld-tankage #08, $\max h^{\text{LM-cut}}_3$ expanded 76.3% fewer nodes than LM-cut. Second, the improved heuristics are sometimes stronger than Max LM-cut and hence the idea of collecting the landmarks into independent hitting set problems of bounded width and optimally solving them really does make a difference. For example, in Openstacks #06, $h^{\text{LM-cut}}_{5,15}$ expanded 72% fewer nodes than LM-cut while $\max h^{\text{LM-cut}}_5$ expanded 61.5% fewer nodes. Third, the improvement is sometimes considerable. Indeed, although the improved heuristic is more costly than LM-cut, there are some problems in which the overall running time for the improved heuristic is lower than for LM-cut. For example, on Satellite #10, the planner with LM-cut required 1,430 seconds to find a solution while the planner with $h^{\text{LM-cut}}_{5,5}$ required 313 seconds.

On the other hand, in the vast majority of cases, the planner equipped with LM-cut was able to solve the problems in less time than the planner equipped with the stronger heuristics. This is mainly because in most cases LM-cut already does a superb job of approximating h^+, which also bounds the novel heuristics from above. However, our ideas are general and can be exploited in many different ways. The purpose of the presented experiment is just to show how a simple implementation can outperform LM-cut in terms of heuristic accuracy.

7 CONCLUSIONS

There are several contributions in this work. We performed an analysis of LM-cut, the current best heuristic for optimal planning, which revealed two major deficiencies. This analysis led us to consider the problem of computing a minimum-cost hitting set as a way to improve LM-cut. However, we observed that hitting sets are not only related to LM-cut, but more generally to the underlying delete relaxation heuristic h^+ and to optimal cost partitioning for landmarks: we showed that h^+ can be understood as the minimum-cost solution of a hitting set problem, that LM-cut is the optimal solution for a relaxation of this problem, and that the LP that defines the optimal cost partitioning for a landmark set is the dual of the LP relaxation of the ILP associated with the hitting set problem for that landmark set.

On the practical side, we used the above findings to define a general method for obtaining heuristics based on landmarks and gave details about its implementation. We instantiated the method and obtained a family of polynomial heuristics $\{h^{\text{LM-cut}}_{p,k} : p, k \geq 1\}$ that provably dominate LM-cut. Some of the heuristics were evaluated over benchmark problems and shown to be superior to LM-cut with respect to the number of expanded nodes.

In the future, we would like to consider other instantiations of the method, e.g. by adapting the landmarks generated by LAMA [15].

Table 1. Percentage of reduction of expanded nodes up to last f layer compared to LM-cut. Only instances with at least one entry $> 50\%$ are reported.

		$\max h_p^{\text{LM-cut}}$			$h_{p,k}^{\text{LM-cut}}$ with $k=5$			$h_{p,k}^{\text{LM-cut}}$ with $k=10$			$h_{p,k}^{\text{LM-cut}}$ with $k=15$		
#	LM-cut	$p=3$	$p=4$	$p=5$	$p=3$	$p=4$	$p=5$	$p=3$	$p=4$	$p=5$	$p=3$	$p=4$	$p=5$
Pipesworld-notankage (relative error of LM-cut w.r.t. h^+ = 19.45%)													
06	107	44.9	54.2	58.9	45.8	54.2	67.3	49.5	54.2	68.2	49.5	54.2	68.2
07	3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
08	84	57.1	64.3	75.0	47.6	57.1	81.0	58.3	75.0	76.2	58.3	75.0	76.2
10	137,092	27.0	36.8	43.0	30.2	40.1	46.9	32.9	43.9	50.0	33.7	47.0	55.1
Pipesworld-tankage (relative error of LM-cut w.r.t. h^+ = 18.42%)													
03	106	61.3	73.6	77.4	67.9	74.5	81.1	67.9	74.5	81.1	67.9	74.5	81.1
05	74	58.1	70.3	70.3	58.1	70.3	70.3	58.1	67.6	70.3	58.1	67.6	70.3
06	223	37.2	51.6	60.5	41.7	52.0	60.5	43.0	55.6	70.0	43.0	55.6	70.0
07	323	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
08	36,203	76.3	83.6	86.7	77.3	84.9	87.6	77.5	85.0	88.2	77.9	85.8	89.2
Openstacks (relative error of LM-cut w.r.t. h^+ = 18.09%)													
01	1,195	44.5	45.9	46.5	53.6	57.3	60.1	59.9	64.8	67.0	60.8	68.3	70.4
03	1,195	43.6	44.5	49.2	54.0	55.6	60.5	59.6	63.6	65.8	61.0	64.8	70.5
04	1,195	42.8	45.9	47.2	53.4	57.8	59.0	58.5	63.9	66.7	63.7	66.8	71.5
05	1,195	43.6	46.2	47.7	52.6	57.4	59.7	58.8	65.0	66.6	61.5	65.6	69.8
06	211,175	61.3	61.4	61.5	64.6	64.9	65.2	69.0	70.7	71.7	69.8	71.2	72.0
07	266,865	55.9	56.2	56.4	60.7	61.3	61.8	65.1	66.4	67.2	65.4	66.8	67.3
Mystery (relative error of LM-cut w.r.t. h^+ = 16.30%)													
06	2,619	40.4	46.2	52.5	40.4	46.2	52.5	40.4	46.2	52.5	40.4	46.5	52.8
09	13	23.0	53.8	61.5	23.0	53.8	61.5	23.0	53.8	61.5	23.0	53.8	61.5
10	32	71.9	84.4	84.4	71.9	84.4	84.4	71.9	84.4	84.4	71.9	84.4	84.4
27	3	33.3	33.3	66.7	33.3	33.3	66.7	33.3	33.3	66.7	33.3	33.3	66.7
28	1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Freecell (relative error of LM-cut w.r.t. h^+ = 13.92%)													
pf1	54	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
pf2	10,802	78.2	83.7	86.4	79.3	83.9	87.2	80.3	83.7	87.4	80.6	84.5	87.6
pf3	17,641	73.6	78.2	81.1	74.5	78.9	82.0	75.7	81.1	82.5	74.1	80.0	82.4
pf4	36,603	68.6	74.1	76.9	70.7	75.2	78.4	70.3	76.3	79.6	72.3	77.3	79.8
pf5	53,670	73.2	75.4	77.1	73.6	76.0	77.9	74.4	77.1	78.8	75.0	77.6	79.3
2-5	277	72.9	73.3	74.0	72.9	73.3	74.0	72.9	73.3	74.0	72.9	73.3	74.0
3-4	17,763	44.6	62.0	72.0	44.6	62.8	73.1	44.7	62.8	72.1	44.7	62.6	72.1
Satellite (relative error of LM-cut w.r.t. h^+ = 1.28%)													
03	6	33.3	50.0	50.0	33.3	50.0	50.0	66.7	50.0	66.7	66.7	50.0	66.7
07	3,616	37.2	53.0	60.9	37.4	52.9	61.7	38.3	56.5	63.1	40.5	60.3	66.8
09	3,666	48.7	61.0	65.9	49.2	61.4	67.7	51.3	64.5	69.6	51.6	66.9	71.5
10	6,671	91.4	93.1	93.8	90.5	93.4	94.0	91.2	93.7	94.1	91.7	93.4	94.1

ACKNOWLEDGEMENTS

We thank the reviewers for their helpful comments. This work was partly supported by the German Research Foundation (DFG) as part of SFB/TR 14 "Automatic Verification and Analysis of Complex Systems" (AVACS).

REFERENCES

[1] Christoph Betz and Malte Helmert, 'Planning with h^+ in theory and practice', in *Proc. KI 2009*, pp. 9–16, (2009).
[2] Blai Bonet and Héctor Geffner, 'Planning as heuristic search', *AIJ*, **129**(1), 5–33, (2001).
[3] Blai Bonet and Malte Helmert, 'Strengthening landmark heuristics via hitting sets: Proofs', Technical Report 259, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, (2010).
[4] Tom Bylander, 'The computational complexity of propositional STRIPS planning', *AIJ*, **69**(1–2), 165–204, (1994).
[5] Andrew Coles, Maria Fox, Derek Long, and Amanda Smith, 'Additive-disjunctive heuristics for optimal planning', in *Proc. ICAPS 2008*, pp. 44–51, (2008).
[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.
[7] Patrik Haslum, '$h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m', in *Proc. ICAPS 2009*, pp. 354–357, (2009).
[8] Patrik Haslum, Blai Bonet, and Héctor Geffner, 'New admissible heuristics for domain-independent planning', in *Proc. AAAI 2005*, pp. 1163–1168, (2005).
[9] Patrik Haslum and Héctor Geffner, 'Admissible heuristics for optimal planning', in *Proc. AIPS 2000*, pp. 140–149, (2000).
[10] Malte Helmert and Carmel Domshlak, 'Landmarks, critical paths and abstractions: What's the difference anyway?', in *Proc. ICAPS 2009*, pp. 162–169, (2009).
[11] Jörg Hoffmann, 'Where 'ignoring delete lists' works: Local search topology in planning benchmarks', *JAIR*, **24**, 685–758, (2005).
[12] Richard M. Karp, 'Reducibility among combinatorial problems', in *Complexity of Computer Computations*, eds., Raymond E. Miller and James W. Thatcher, 85–103, (1972).
[13] Erez Karpas and Carmel Domshlak, 'Cost-optimal planning with landmarks', in *Proc. IJCAI 2009*, pp. 1728–1733, (2009).
[14] Michael Katz and Carmel Domshlak, 'Optimal additive composition of abstraction-based admissible heuristics', in *Proc. ICAPS 2008*, pp. 174–181, (2008).
[15] Silvia Richter and Matthias Westphal, 'The LAMA planner: Guiding cost-based anytime planning with landmarks', *JAIR*, (2010). To appear.