Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?

Malte Helmert
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Georges-Köhler-Allee 52
79110 Freiburg, Germany
helmert@informatik.uni-freiburg.de

Carmel Domshlak
Technion
Faculty of Industrial Engineering and Management
Haifa 32000, Israel
dcarmel@ie.technion.ac.il

Abstract

Current heuristic estimators for classical domain-independent planning are usually based on one of four ideas: *delete relaxations*, *critical paths*, *abstractions*, and, most recently, *landmarks*. Previously, these different ideas for deriving heuristic functions were largely unconnected.

We prove that admissible heuristics based on these ideas are in fact very closely related. Exploiting this relationship, we introduce a new admissible heuristic called the *landmark cut heuristic*, which compares favourably with the state of the art in terms of heuristic accuracy and overall performance.

Introduction

Heuristic search, either in the space of world states reached through progression or in the space of subgoals reached through regression, is a common and successful approach to classical planning. For example, at the recent 6th International Planning Competition (IPC-2008), the three best-performing satisficing planners and two of the three best-performing optimal planners in the sequential planning tracks followed this paradigm.[1] Apart from the choice of search algorithm, the main feature that distinguishes heuristic planners is their heuristic estimator. Most current heuristic functions are based on one of the following four ideas:

1. *delete relaxations*: e. g., h^+, $h^{\max}$, h^{add}, h^{FF}, h^{pmax}, h^{sa}
2. *critical paths*: the h^m heuristic family
3. *abstractions*: pattern databases, merge-and-shrink abstractions, and structural patterns
4. *landmarks*: LAMA's h^{LM}, and the admissible landmark heuristics h^{L} and h^{LA}

We discuss these heuristics in detail and provide literature references later. For now, we remark that these four ideas have been developed in relative isolation. Indeed, apart from Haslum and Geffner's (2000) result that $h^{\max}$ is a special case of the h^m family ($h^{\max} = h^1$), we are not aware of any published formal connections.

In this paper, we prove further results that relate the quality of heuristics from the above four families. We limit our attention to *admissible* heuristics because it is hard to define a notion of "heuristic quality" for inadmissible heuristics that is independent of the search algorithm being used. Admissible heuristics, in contrast, have a clear notion of *dominance*: if $h_1(s) \geq h_2(s)$ for all states s, then h_1 is superior or equal to h_2 in terms of heuristic quality, with provable consequences for the performance of optimal search algorithms. In the theoretical part of this paper, we establish several such dominance results:

- Landmark heuristics dominate additive $h^{\max}$ heuristics.
- Additive $h^{\max}$ heuristics dominate landmark heuristics.
- Additive critical path heuristics with $m \geq 2$ strictly dominate landmark heuristics and additive $h^{\max}$ heuristics.
- Merge-and-shrink abstractions strictly dominate landmark heuristics and additive $h^{\max}$ heuristics.
- Pattern database abstractions are incomparable with landmark heuristics and additive $h^{\max}$ heuristics.

These statements are informal summaries, and some restrictions apply. In particular, the results for landmark heuristics only apply to *relaxation-based* landmarks which are verifiable by a relaxed planning graph criterion. (All landmark heuristics considered in the literature fall into this class.) On the positive side, all results are constructive, showing how to compute a dominating heuristic in polynomial time.

As a result of our dominance proofs, we obtain a new admissible heuristic called the *landmark cut heuristic* $h^{\text{LM-cut}}$, which can alternatively be viewed as a landmark heuristic, a cost partitioning scheme for additive $h^{\max}$, or an approximation to the (intractable) optimal relaxation heuristic h^+. We experimentally demonstrate that $h^{\text{LM-cut}}$ gives excellent approximations to h^+ and compares favourably to other admissible heuristics in terms of accuracy. Moreover, we show an optimal planner based on the landmark cut heuristic to be highly competitive with the state of the art.

Preliminaries

STRIPS planning. For the theoretical results of this paper, we use the propositional STRIPS formalism augmented with non-negative actions costs (e. g., Keyder and Geffner, 2008). Some of the heuristics we discuss were originally introduced for the more general SAS$^+$ formalism (Bäckström and Nebel 1995), to which our results apply equally.

[1] See `http://ipc.informatik.uni-freiburg.de`.

Definition 1 (planning task)
A planning task *is a 4-tuple* $\Pi = \langle V, O, I, G \rangle$*, where*

- V *is a finite set of* propositional state variables*,*
- O *is a finite set of* operators*, each with associated* preconditions $pre(o) \subseteq V$*,* add effects $add(o) \subseteq V$*,* delete effects $del(o) \subseteq V$ *and* cost $cost(o) \in \mathbb{R}_0^+$*,*
- $I \subseteq V$ *is the* initial state*, and*
- $G \subseteq V$ *is the set of* goals*.*

State variables of planning tasks are also called *propositions* or *facts*. A *state* in our formalism is a subset of facts, representing the propositions which are currently true. States can alternatively be defined as assignments to state variables, but set notation is more convenient for the purposes of this paper. *Applying* an operator o in s results in state $(s \backslash del(o)) \cup add(o)$, which we denote as $s[o]$. The notation is only defined if o is *applicable* in s, i. e., if $pre(o) \subseteq s$.

Applying a sequence $o_1, \ldots, o_n$ of operators to a state is defined inductively as $s[\epsilon] := s$ and $s[o_1, \ldots, o_{n+1}] := (s[o_1, \ldots, o_n])[o_{n+1}]$. A plan for a state s (*s-plan*, or *plan* when s is clear from context) is an operator sequence π such that $s[\pi]$ is defined and satisfies all goals (i. e., $G \subseteq s[\pi]$). The *cost* of plan π is $cost(\pi) := \sum_{i=1}^{n} cost(o_i)$. The objective of *optimal planning* is to find an I-plan of minimal cost (called an *optimal I-plan*) or prove that no plan exists.

Heuristics. Heuristic functions or *heuristics* are a key ingredient of heuristic search planners. A heuristic is a function $h : 2^V \to \mathbb{R}_0^+ \cup \{\infty\}$ with the intuition that $h(s)$ estimates the cost of an s-plan. The *perfect heuristic* h^* maps each state to the cost of an optimal s-plan (infinite if no s-plan exists). A heuristic h is *admissible* if $h(s) \leq h^*(s)$ for all states s. All common heuristic search algorithms for optimal planning require admissible heuristics. If $h(s) \geq h'(s)$ for all states s, we say that h *dominates* h'.

Cost partitioning. If $h_1, \ldots, h_k$ are admissible heuristics, then their pointwise *maximum* is an admissible heuristic dominating each individual heuristic. Under certain conditions, their pointwise *sum*, which dominates the maximum, is also admissible. Many recent advances in the accuracy of admissible planning heuristics are due to better, more fine-grained methods for finding admissible additive heuristics.

Katz and Domshlak (2008a) introduced a very general criterion for admissible additive combinations. Let $\Pi, \Pi_1, \ldots, \Pi_k$ be planning tasks which are identical except for the operator costs. Let $cost : O \to \mathbb{R}_0^+$ denote the operator cost function for Π and $cost_i : O \to \mathbb{R}_0^+$ denote the operator cost functions for Π_i. If $\sum_{i=1}^{k} cost_i(o) \leq cost(o)$ for all operators $o \in O$, then the sum of arbitrary admissible heuristics for Π_i is an admissible heuristic for Π. We call such a separation of Π into planning tasks $\Pi_1, \ldots, \Pi_k$ with smaller operator costs a *cost partitioning* of Π.

Cost partitioning offers a very flexible way of additively combining different heuristic estimates in an admissible way. In particular, it subsumes earlier additivity criteria for pattern database heuristics by Edelkamp (2001) and for general admissible heuristics by Haslum et al. (2005).

Of course, different cost partitionings for the same component heuristics lead to overall heuristics of different quality, and the question of how to automatically derive a *good* cost partitioning has attracted considerable interest. At least theoretically, this question has recently been fully resolved for *abstraction heuristics* (Katz and Domshlak 2008a) and *landmark heuristics* (Karpas and Domshlak 2009) with the development of algorithms that compute an *optimal* cost partitioning for a given state and component heuristic set in polynomial time. Practically, it is still an interesting question how to reduce the time requirements of these optimal partitioning algorithms or come up with approximate results quickly. For *relaxation heuristics* and *critical path heuristics*, the question of optimal cost partitioning remains open. Suboptimal algorithms have been presented by Haslum et al. (2005) and by Coles et al. (2008).

Planning Heuristics

We now formally introduce the heuristic functions we consider in our analysis.

Relaxation heuristics. Relaxation heuristics estimate the cost of reaching a goal state by considering a *relaxed task* Π^+ derived from the actual planning task Π by ignoring all delete effects of operators, i. e., replacing each operator o by a new operator o^+ with the same preconditions, add effects and cost as o and $del(o^+) = \emptyset$. (*Notation:* if S is an operator set, S^+ denotes the set $\{ o^+ \mid o \in S \}$.)

The idealized h^+ heuristic (Hoffmann and Nebel 2001) uses the cost of an optimal s-plan in Π^+ as the heuristic estimate for state s. This is an admissible heuristic that is often very informative (Hoffmann 2005; Helmert and Mattmüller 2008; Betz 2009), but NP-hard to compute (Bylander 1994). Due to its computational complexity, the h^+ heuristic has not been used in a domain-independent planning system. Instead, *inadmissible* estimates of h^+ such as the additive heuristic (Bonet and Geffner 2001), FF heuristic (Hoffmann and Nebel 2001), pairwise max heuristic (Mirkis and Domshlak 2007) or set-additive heuristic (Keyder and Geffner 2008) are commonly used. Inadmissible estimates of h^+ cannot be admissible heuristics (consider the case $\Pi = \Pi^+$), and hence we do not discuss them further.

Instead, we focus on the max heuristic $h^{\max}$ (Bonet and Geffner 2001), which provides an admissible estimate for h^+. The $h^{\max}$ value of a state s of planning task $\langle V, O, I, G \rangle$ is defined in terms of *proposition costs* $c_s(v)$, with $h^{\max}(s) = \max_{v \in G} c_s(v)$. The proposition costs are defined as the maximal solution to the recursive equations $cost_s(v) = 0$ for $v \in s$ and $c_s(v) = \min_{o \in O \text{ with } v \in add(o)} (cost(o) + \max_{p \in pre(o)} c_s(p))$ for $v \notin s$. (Empty minima are defined as ∞ here, empty maxima as 0.)

The max heuristic is often not very informative, but this weakness can be overcome to a large extent by using suitable cost partitioning (e. g., Coles et al., 2008). In this case, all component heuristics are copies of the $h^{\max}$ heuristic with different cost functions. We will call such heuristics *additive $h^{\max}$ heuristics*. Note that all additive $h^{\max}$ heuristics are admissible heuristics for Π^+ and hence dominated by h^+.

Critical path heuristics. The h^m heuristics (Haslum and Geffner 2000) estimate goal distances by computing lower-bound estimates on the cost of achieving sets of facts of car-

dinality m, where $m \in \mathbb{N}_1$ is a parameter. Roughly speaking, the underlying simplifying assumption is that a set of facts is reachable with cost K iff all its m-subsets are reachable with cost K. Computing $h^m(s)$ requires polynomial time for fixed m, but exponential time in m.

We call the h^m family *critical path heuristics* because their heuristic estimate is based on the length of the most expensive branch in a tree-shaped plan (not unlike a partial-order plan) for the simplified problem. All h^m heuristics are admissible. In typical planning benchmarks, the h^m heuristics have unbounded relative error (Helmert and Mattmüller 2008), but as with $h^{\max} = h^1$ this weakness can be overcome by suitable cost partitioning. For all $m \geq 2$, h^m dominates h^{m-1} and is incomparable with h^+.

Abstraction heuristics. Abstraction heuristics map each state s of Π to an *abstract state* $\alpha(s)$ through a homomorphism function α. The heuristic value $h^\alpha(s)$ is then the distance from $\alpha(s)$ to the nearest abstract goal state in the transition system induced by α on the transition system of Π. This always leads to an admissible heuristic because each plan for Π has a corresponding abstract plan with the same cost. The real cost can be underestimated because α is generally not injective and hence not every abstract plan corresponds to a plan for Π.

Different abstraction mappings lead to heuristics of different quality. Examples include pattern databases (Edelkamp 2001; Haslum, Bonet, and Geffner 2005; Haslum et al. 2007), merge-and-shrink abstractions (Helmert, Haslum, and Hoffmann 2007) and structural patterns (Katz and Domshlak 2008b). Katz and Domshlak (2008a) showed that optimal cost partitions for an ensemble of abstraction heuristics can be computed in polynomial time. However, finding an abstraction mapping that is compactly representable and leads to an informative heuristic remains difficult.

Landmark heuristics. A *fact landmark* for a state s is a fact that is true at some point in every s-plan. The first planning algorithm exploiting fact landmarks was presented by Porteous et al. (2001). The first (inadmissible) landmark-based heuristic is due to Richter et al. (2008), and several admissible landmark heuristics have recently been defined by Karpas and Domshlak (2009). The latter papers use extended notions of landmarks which are subsumed by *disjunctive action landmarks*: sets of actions of which at least one is part of every s-plan. Since this is the most general and (for this paper) useful notion of landmarks, we simply refer to disjunctive action landmarks for a state s as *s-landmarks*.

Deciding whether an operator set $L \subseteq O$ is an s-landmark in a planning task Π is PSPACE-hard (Porteous, Sebastia, and Hoffmann 2001) and therefore existing landmark heuristics employ a sufficient criterion based on relaxed planning graphs. This criterion is equivalent to testing whether L^+ is an s-landmark in the *delete relaxation* Π^+. For this reason, the existing landmark heuristics assign the same heuristic value to states of Π and Π^+, and therefore they are *relaxation heuristics* in the sense that they are dominated by h^+ (if they are admissible). We only consider relaxation-based landmarks in this paper, but more general landmarks are certainly conceivable, for example based on reachability criteria for higher-order critical path heuristics.

The *elementary landmark heuristic* for planning task Π and operator subset L assigns the estimate $\min_{o \in L} cost(o)$ to a state s if L^+ is an s-landmark of Π^+ and 0 otherwise. This is clearly admissible: if L^+ is an s-landmark, one of its elements must be contained in every plan for Π^+ and hence at least the cost of the cheapest operator in L must be paid. The admissible heuristics of Karpas and Domshlak can be understood as cost partitionings over sets of elementary landmark heuristics. This is a slightly idealized view because their search algorithm does not actually test for each search state whether an operator set is a landmark but rather uses a more efficiently computable sufficient criterion that loses some heuristic accuracy.

Landmarks vs. $h^{\max}$ vs. Abstractions

We now present the main theorems relating admissible landmark heuristics to additive $h^{\max}$ heuristics and abstraction heuristics. All our results take on the form of *per-state compilations*: given a state s, a planning task Π and an additive ensemble of heuristics $h_1, \ldots, h_k$ from a given class (e. g., elementary landmark heuristics), we show how to compute an additive ensemble of heuristics from another class (e. g., merge-and-shrink abstraction heuristics) $h'_1, \ldots, h'_m$ with $\sum_{i=1}^m h'_i(s) \geq \sum_{i=1}^k h_i(s)$. We only discuss the case $k = 1$, which suffices: general results follow by compiling the component heuristics individually. Our algorithms are polynomial in $\|\Pi\|$, which prevents some trivial compilations. (For example, exponential-size abstractions can always represent the perfect heuristic h^*, which dominates everything.)

Throughout this section, $\Pi = \langle V, O, I, G \rangle$ is the given planning task and we assume that the heuristic value is to be computed for the state I.

From Landmarks to Abstractions

We first consider compilations from elementary landmark heuristics to abstraction heuristics, namely to pattern databases and merge-and-shrink abstractions. The converse compilations are clearly not possible in general because, unlike elementary landmark heuristics, neither of these abstraction heuristics are bounded by h^+. For pattern database heuristics, we prove a negative result.

Theorem 1 (Landmarks to pattern databases)
There is no polynomial-time compilation of elementary landmark heuristics into pattern database heuristics.

Proof sketch: Consider the planning task family $(\Pi_n)_{n \in \mathbb{N}_1}$ where Π_n has state variables $\{v_1, \ldots, v_n, g\}$, initial state $\emptyset$, goal $\{g\}$ and an operator set containing, for each $i \in \{1, \ldots, n\}$, one "setup operator" of cost 1 that achieves $\{v_i\}$ with no preconditions and one "goal operator" of cost 0 that achieves the goal $\{g\}$ with precondition $\{v_i\}$.

The set of all setup operators forms an I-landmark in Π_n, and we get $h(I) = 1$ for the associated elementary landmark heuristic. However, all pattern database heuristics h' for Π_n that project away at least one state variable yield $h'(I) = 0$, and the polynomial-time requirement demands that some variables need to be projected away for large n. ∎

The weakness of pattern databases exploited in our proof is that their capabilities for representing "disjunctive resources" like the facts v_i are very limited. We remark that this does not apply to *symbolic* pattern databases (Edelkamp 2002), which have exponentially more compact representations than traditional pattern databases on some planning tasks. Merge-and-shrink abstractions (Helmert, Haslum, and Hoffmann 2007) are another class of abstraction heuristics that do not share this weakness. Indeed, they are powerful enough to completely capture landmark heuristics.

Theorem 2 (Landmarks to merge-and-shrink abstractions)
Elementary landmark heuristics can be compiled into merge-and-shrink heuristics in polynomial time.

Proof sketch: We are given the elementary landmark heuristic h for the operator subset L. Let V' be the set of facts that can be reached from I in Π^+ without using the operators L^+. The set V' can be computed in polynomial time by a standard relaxed exploration. Now consider the abstraction heuristic h' induced by

$$\alpha(s) = \begin{cases} \tilde{s}_1 & \text{if } s \subseteq V' \\ \tilde{s}_2 & \text{if } s \not\subseteq V' \end{cases}$$

(where the abstract state space consists of only two states).

We must show that α can be computed as a merge-and-shrink abstraction in polynomial time. But this is easy to see: we can use a linear merge strategy with an arbitrary variable order and shrink all intermediate abstract transition graphs to two abstract states, one corresponding to all states where the state variables *not* in V' that were considered so far in the heuristic construction all have value 0, and the other corresponding to all other states.

Now let us compare $h'(I)$ to $h(I)$. If L^+ is not an I-landmark of Π^+, then $h(I) = 0$ and hence clearly $h'(I) \geq h(I)$. So consider the case where L^+ is an I-landmark of Π^+. This implies that $G \not\subseteq V'$, and hence $s^* \not\subseteq V'$ for all goal states s^*, which shows that $\alpha(s^*) = \tilde{s}_2$ for all goal state. Therefore, $\tilde{s}_2$ is the only abstract goal state. Moreover, $\alpha(I) = \tilde{s}_1$, as clearly $I \subseteq V'$. All abstract plans must therefore perform a transition from $\tilde{s}_1$ to $\tilde{s}_2$, and $h'(I)$ is the minimal cost of all such transitions.

Assume that there is a state transition in Π from a state $s_1 \subseteq V'$ (i. e., a state with $\alpha(s_1) = \tilde{s}_1$) to a state $s_2 \not\subseteq V'$ (i. e., a state with $\alpha(s_2) = \tilde{s}_2$) by an operator $o \notin L$. Then the relaxation of o is applicable in state s_1 in Π^+ and leads to a state $s_2' \not\subseteq V'$, which contradicts the definition of V'. Hence, all abstract transitions from $\tilde{s}_1$ to $\tilde{s}_2$ are induced by some operator from L and thus have cost at least $\min_{o \in L} \mathit{cost}(o)$. Because all abstract solutions must contain such a transition, we get $h'(I) \geq \min_{o \in L} \mathit{cost}(o) = h(I)$. ■

This concludes our comparison of landmark heuristics and abstraction heuristics: landmark heuristics are strictly less powerful than merge-and-shrink abstractions and incomparable to pattern database heuristics.

From Landmarks to $h^{\max}$ and Back

We now move away from abstraction heuristics and consider the relationship between landmark heuristics and the max heuristic. Our first result has a simple proof.

Theorem 3 (Landmarks to $h^{\max}$)
Elementary landmark heuristics can be compiled into additive $h^{\max}$ heuristics in polynomial time.

Proof sketch: We are given the elementary landmark heuristic h for the operator subset L. We directly compile h to the $h^{\max}$ heuristic for the same planning task (i. e., we do not perform additional cost partitioning). Again, let L^+ be the relaxed operators induced by L.

As in the proof of Theorem 2, if L^+ is not an I-landmark of Π^+, then clearly $h^{\max}(I) \geq 0 = h(I)$. If L^+ *is* an I-landmark of Π^+, then Π^+ is not solvable without using some operator in L^+, which means that $h^{\max}(I)$ would be infinite if the operators in L did not exist. This implies that if $h^{\max}(I) \neq \infty$, then the cost computation for $h^{\max}(I)$ *must* make use of at least one of the operators in L, and hence $h^{\max}(I) \geq \min_{o \in L} \mathit{cost}(o) = h(I)$. ■

Extending this result to arbitrary cost partitionings, we observe that additive $h^{\max}$ heuristics are at least as powerful as admissible landmark heuristics. We now show (maybe somewhat more surprisingly) that the converse is also true under a small additional condition.

Theorem 4 ($h^{\max}$ to landmarks)
For states with finite $h^{\max}$ value, the $h^{\max}$ heuristic can be compiled into additive elementary landmark heuristics in polynomial time.

Proof sketch: If $h^{\max}(I) = 0$, there is nothing to prove. Otherwise, we show how to find a cost partitioning $\mathit{cost} = \mathit{cost}_1 + \mathit{cost}_2$ for Π with two heuristics h_1 and h_2 such that

- $h^{\max}(I) \leq h_1(I) + h_2(I)$,
- h_1 is an elementary landmark heuristic with cost function cost_1, and
- h_2 is the $h^{\max}$ heuristic with cost function cost_2.

The reduction is then applied recursively to h_2 if $h_2(I) > 0$. As we show later, the set of zero cost operators is strictly larger for cost_2 than for cost, so that the process terminates in polynomially many (at most $|O|$) steps. We now describe how to find the landmark that defines heuristic h_1 and how to perform the cost partitioning of cost into cost_1 and cost_2.

Without loss of generality, we assume that there exists at least one fact in I, at least one fact in G, and that each operator has at least one precondition. (If necessary, add a "dummy fact" d to I, G and all preconditions.) We first determine the $h^{\max}$ cost values of all facts V and then compute a modified planning task Π' obtained from Π^+ by two transformations. First, we turn the goal set and all operator preconditions into singleton sets by keeping only one fact with *maximal* $h^{\max}$ *cost* among the set elements. This is a further relaxation of Π^+ because we drop goals and operator preconditions while leaving all else identical. Because we keep the $h^{\max}$ cost maximizers in each condition, $h^{\max}(I)$ is *not changed* by this transformation. In the second transformation step, we replace each operator o with $\mathit{pre}(o) = \{p\}$ and $\mathit{add}(o) = \{a_1, \ldots, a_k\}$ with k operators $o_1, \ldots, o_k$ with $\mathit{pre}(o_i) = \{p\}$ and $\mathit{add}(o_i) = \{a_i\}$ for $i \in \{1, \ldots, k\}$. That is, operators are split up according to their add effects. Again, it is easy to verify that this transformation does not affect the $h^{\max}$ value.

The resulting planning task, which we call Π', has a singleton goal set and all operators have exactly one precondition, exactly one add effect, and no delete effects. Based on Π', we define a directed weighted graph $G_{\Pi'}$ whose vertices are the facts of Π and which has an arc from u to v with weight w iff Π' has an operator with precondition u, add effect v and cost w. (Parallel arcs are possible for multiple operators with identical precondition and effect.) We call this graph a *justification graph* for Π because, even though it describes a planning task much simpler than Π, it retains enough information to *justify* the $h^{\max}$ costs of Π. The $h^{\max}$ cost of a fact v is the length of a shortest path in the justification graph from some fact in I to v.

A *cut* in the justification graph is a set of arcs of $G_{\Pi'}$ such that all paths from some fact in I to the goal fact g traverse at least one such arc. Cuts in the justification graph are guaranteed to exist: for example, the set of *all* arcs clearly forms a cut. (We cannot have $g \in I$ because $h^{\max}(I) > 0$.) Cuts in the justification graph correspond to I-landmarks of Π': without the operators of Π' that define the cut, Π' cannot be solved. Due to the construction of Π' as a further relaxation of Π^+, this implies that the operators L^+ that induce the cut form a landmark of Π^+.

Given a cut, we can compile $h^{\max}$ into an elementary landmark heuristic h_1 that represents the cut and a remaining $h^{\max}$ heuristic h_2 that represents everything else. In detail, we choose as h_1 the elementary landmark heuristic for the landmark L that corresponds to the cut, and associate with it the cost function $cost_1$ that assigns cost $c_{\min} := \min_{o \in L} cost(o)$ to all operators $o \in L$ and cost 0 to all other operators. The heuristic h_2 is the $h^{\max}$ heuristic with cost function $cost_2 := cost - cost_1$. To complete the proof, we need to find a cut that guarantees

(a) $c_{\min} > 0$ and

(b) $h_1(I) + h_2(I) \geq h^{\max}(I)$.

Condition (a) ensures that $cost_2$ has more zero-cost actions than $cost$ and hence the partitioning process eventually terminates. Condition (b) ensures that we obtain a heuristic value at least as large as the initial $h^{\max}(I)$ estimate.

To find a suitable cut, we partition the facts of Π into three sets:

- the *goal zone* V^*, consisting of all facts from which the goal fact g can be reached in $G_{\Pi'}$ through a path of cost 0. The goal zone is disjoint from I because otherwise we would have $h^{\max}(I) = 0$.
- the *before-goal zone* V^0, consisting of all facts that can be reached from I in $G_{\Pi'}$ without ever entering V^*, and
- the *beyond-goal zone*, consisting of everything else.

As our cut, we choose the set of all arcs from facts in V^0 to facts in V^*. Let L be the set of operators inducing the cut. We now show that this satisfies conditions (a) and (b) above.

For (a), assume that $c_{\min} = 0$, i. e., $cost(o) = 0$ for some $o \in L$. Then there is an arc of weight 0 from some $v^0 \in V^0$ to some $v^* \in V^*$. But this violates the definition of the goal zone V^*, because then there would be a 0-cost path from v^0 to the goal fact g and v^0 should be included in V^*. This is a contradiction, and we can conclude that $c_{\min} > 0$.

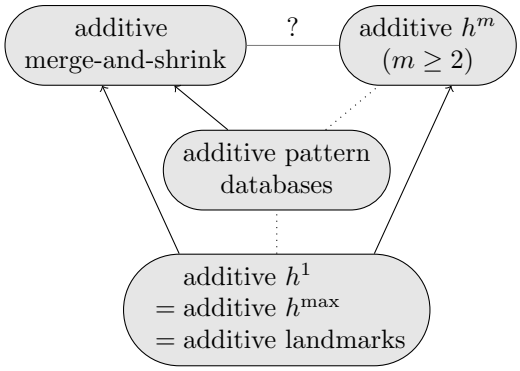

Figure 1: Summary of compilation results. Arrows indicate that heuristics in one class can be compiled into dominating heuristics in the other class in polynomial time. Dotted lines indicate that such compilations are not possible in either direction. Additive merge-and-shrink heuristics cannot be compiled into additive h^m heuristics in polynomial time; the converse question is open.

For (b), we know that L^+ is an I-landmark of Π^+ and hence $h_1(I) = \min_{o \in L} cost_1(o) = c_{\min}$. It remains to be shown that $h_2(I) \geq h^{\max}(I) - c_{\min}$. In words, we must show that reducing the cost of the operators in L by $c_{\min}$ does not decrease the $h^{\max}$ value of the task by more than $c_{\min}$. To see this, observe that we can require that every "reasonable" path in the justification graph from I to the goal fact only passes from V^0 to V^* (and hence uses an operator from L) *once*: as soon as the goal zone is reached, the goal fact can be reached free of cost, and thus, once the goal zone is reached, it should not be left again. Hence, reducing the costs of the operators in L by $c_{\min}$ cannot reduce the goal distances in the justification graph by more than $c_{\min}$. Since the goal distances correspond to the $h^{\max}$ costs of Π' and Π' is a relaxation of Π^+ in terms of $h^{\max}$ value, we can conclude that the cost reduction only reduces the $h^{\max}$ value of Π by at most $c_{\min}$. This concludes the proof. ∎

We remark that the restriction of the theorem to states with finite $h^{\max}$ values is necessary because elementary landmark heuristic values – and hence there admissible additive combinations – are always finite. However, in practical applications of the compilation algorithm this restriction is not problematic because infinite $h^{\max}$ estimates can be detected in the first compilation step.

Discussion of the Compilation Results

Our compilation results are summarized in Fig. 1.[2] There are two results we consider quite surprising, namely the equivalence of additive $h^{\max}$ and additive landmarks, and the fact that these heuristics are strictly dominated by merge-and-shrink heuristics. Besides being an academic curiosity, do

[2]For space reasons, we do not provide a proof for one of the results shown in the figure, namely that pattern database heuristics cannot be compiled to general additive h^m heuristics. The proof is quite simple and exploits the fact that h^m is bounded by $O(N^m)$ in uniform-cost domains, where N is the set of atoms of the task.

these results have any further implications for the study of admissible planning heuristics?

For the compilation of additive $h^{\max}$/landmarks to merge-and-shrink heuristics, we do not see any immediate application. The reduction in Theorem 2 appears too costly to use for every state of a search, and also it typically leads to a heuristic estimate that is no larger than for the original landmark heuristic. Still, it is worth noting that our results imply that the scaling behaviour of abstraction heuristics can be at least as good as that of landmark heuristics – opposite to recent experimental evidence (cf. the scaling experiments by Karpas and Domshlak, 2009). This suggests that there is scope for improving the current methods for generating merge-and-shrink abstractions. However, we have no immediate insights on how such improvements could be obtained.

Regarding the equivalence of additive $h^{\max}$ and admissible landmark heuristics, however, there are some immediate consequences of our result which we consider to be of interest. Firstly, we get some insights into the problem of computing *optimal additive* $h^{\max}$ *heuristics*, about which very little was known previously (cf. the discussion of Katz and Domshlak, 2008a). We now know that this problem is equivalent to finding an optimal admissible *landmark* heuristic, for which we can at least provide a brute force algorithm: generate all possible disjunctive fact landmark heuristics, then compute an optimal cost decomposition. Since there are exponentially many possible disjunctive fact landmarks, this is an exponential algorithm and hence not practical, but this is still an improvement over the previous situation where *no algorithm* for optimal additive $h^{\max}$ was known.

Secondly, the reduction used in Theorem 4 can actually be used in practice to improve the heuristic estimates of a given additive $h^{\max}$ heuristic. The computation of a single cut can be performed in $O(\|\Pi\|)$, the same asymptotic time it takes to compute a single $h^{\max}$ estimate. The overall runtime of the procedure is then bounded by $O(|O| \cdot \|\Pi\|)$, no worse than the time required for computing h^2.

Landmark Cut Heuristic

One way to make use of the reduction from additive $h^{\max}$ to landmarks is to apply it to the simplest possible additive $h^{\max}$ heuristic, namely to *standard* $h^{\max}$ without any cost decomposition. This results in a heuristic that estimates goal distances by repeatedly computing landmarks that constitute cuts in justification graphs, until the generated landmarks have eroded so much cost from the original $h^{\max}$ heuristic that no further cuts of nonzero cost can be found. We have implemented the resulting heuristic, which we call the *landmark cut heuristic* $h^{\text{LM-cut}}$, and performed two experiments with it. In the first experiment, we focus on heuristic accuracy, without regard for computation time. In the second experiment, we consider the use of the heuristic within an A*-based optimal planning algorithm.

Our planner implementation is built on the implementation of merge-and-shrink abstractions by Helmert et al. (2007). We evaluate on all IPC benchmark domains supported by that system.

All experiments were conducted on Linux computers with 3 GHz Intel E8400 CPUs using a 30 minute timeout. We set a 1.5 GB memory limit except for runs with the Gamer planner, which for technical reasons required 2 GB of memory.

Domain	h^+	$h^{\max}$	**HBG**	**CFLS**	h^{LA}	$h^{\text{LM-cut}}$
Airport (37)	114.38	36.68	n/a	110.49	108.97	**114.38**
Blocks (35)	17.37	7.54	16.86	12.00	**17.37**	**17.37**
Gripper (20)	47.00	2.00	n/a	**47.00**	**47.00**	**47.00**
Logistics-2000 (26)	35.12	5.85	31.42	33.81	35.00	**35.12**
Miconic-STRIPS (150)	50.47	2.99	5.11	32.00	**50.47**	**50.47**
Pathways (5)	15.60	5.80	5.80	9.00	7.60	**15.60**
PSR-Small (50)	3.14	1.46	2.78	2.46	**3.14**	**3.14**
TPP (18)	32.17	6.39	n/a	n/a	17.61	**32.17**
Depot (10)	20.90	4.70	14.80	17.40	17.50	**20.50**
Driverlog (14)	15.50	4.71	10.71	12.00	13.43	**15.00**
Grid (2)	15.00	10.50	10.50	11.50	11.50	**14.00**
Logistics-1998 (10)	27.90	5.30	n/a	22.10	23.50	**27.70**
MPrime (24)	5.42	3.54	n/a	4.38	3.42	**4.92**
Rovers (14)	18.21	3.71	12.21	11.43	11.64	**18.00**
Satellite (9)	17.11	3.00	4.22	9.33	15.89	**16.89**
Zenotravel (13)	11.62	2.85	9.08	9.46	11.00	**11.54**
FreeCell (6)	8.33	3.00	7.00	3.33	**7.67**	7.17
Mystery (18)	6.44	3.56	n/a	4.72	3.67	**5.39**
Openstacks (5)	21.00	4.00	12.00	17.00	**21.00**	17.20
Pipesworld-NoTankage (18)	10.28	4.33	n/a	4.50	7.17	**8.28**
Pipesworld-Tankage (11)	8.36	3.91	n/a	3.91	6.27	**6.82**
Trucks (10)	21.70	4.00	**20.50**	9.90	14.60	**20.50**
avg. additive error compared to h^+		27.99	17.37	8.05	1.94	**0.28**
avg. relative error compared to h^+		68.5%	40.9%	25.2%	9.5%	**2.5%**

Table 1: Comparison of heuristic accuracy of relaxation heuristics. Domains are annotated with the number of instances considered, and the following columns show the average h values for the initial states of these instances. Entries *n/a* indicate running out of memory or time on some instances. Best approximations to h^+ are highlighted in bold.

Accuracy. To evaluate its accuracy, we compare the landmark cut heuristic to other heuristics from the same class, i. e., based on $h^{\max}$ or landmarks. In detail, we compare to

- plain $h^{\max}$ (Bonet and Geffner 2001),
- the original additive $h^{\max}$ heuristic (Haslum, Bonet, and Geffner 2005), abbreviated as HBG,
- the recent ADHG decomposition for additive $h^{\max}$ (Coles et al. 2008), abbreviated as CFLS, and
- the admissible landmark heuristic h^{LA} (Karpas and Domshlak 2009).

None of these heuristics take delete effects into account, so they are all bounded by h^+. The objective of our experiment is to determine *how closely* each of them approximates h^+. For this purpose, we compute the initial state heuristic values for all of our benchmark tasks where we were able to compute the corresponding value for h^+.[3]

Table 1 shows the results of the experiment. We see that $h^{\text{LM-cut}}$ is a significantly more accurate approximation to h^+ than the other approaches. The average additive error compared to h^+ across all instances is 0.28, implying that for

[3]It is likely that in cases where we cannot determine h^+, the heuristic errors are larger for all five heuristics, so the absolute errors reported are not necessarily indicative of large instances. The h^+ values were computed through domain-specific techniques (Betz 2009) and heuristic search in the delete relaxation.

Domain	$h^{\text{LM-cut}}$	h^{LA}	$h^{\text{m\&s}}$	$h^{\max}$	h^0	Gamer	HSP^*_{F}
Airport (50)	**38**	24	16	20	17	11	15
Blocks (35)	28	20	18	18	18	**30**	**30**
Depot (22)	**7**	**7**	**7**	4	4	4	4
Driverlog (20)	**14**	**14**	12	8	7	11	9
FreeCell (80)	15	**28**	15	15	14	11	20
Grid (5)	**2**	**2**	**2**	**2**	1	**2**	0
Gripper (20)	6	6	7	7	7	**20**	6
Logistics-2000 (28)	**20**	**20**	16	10	10	**20**	16
Logistics-1998 (35)	**6**	5	4	2	2	**6**	3
Miconic-STRIPS (150)	**140**	**140**	54	50	50	85	45
MPrime (35)	**25**	21	21	24	19	9	8
Mystery (19)	**17**	15	14	15	15	8	9
Openstacks (30)	**7**	**7**	**7**	**7**	**7**	**7**	**7**
Pathways (30)	**5**	4	3	4	4	4	4
Pipesworld-NoTankage (50)	17	17	**20**	17	14	11	13
Pipesworld-Tankage (50)	11	9	**13**	10	10	6	7
PSR-Small (50)	49	48	**50**	49	48	47	**50**
Rovers (40)	**7**	6	6	6	5	5	6
Satellite (36)	**8**	7	6	5	4	6	5
TPP (30)	**6**	**6**	**6**	**6**	5	5	5
Trucks (30)	**10**	7	6	7	5	3	9
Zenotravel (20)	**12**	9	11	8	7	10	8
Total	**450**	422	314	294	273	321	279

Table 2: Comparison of solved tasks. Number of tasks in each domain is shown in parentheses (11 unsolvable Mystery tasks omitted). Best results are highlighted in bold.

at least 72% of the tasks we obtain the perfect h^+ estimate. The next best heuristic, h^{LA}, has an error of 1.94, almost 7 times the value for $h^{\text{LM-cut}}$. Considering relative error, $h^{\text{LM-cut}}$ outperforms h^{LA} by a factor of 3.8. Looking at individual domains, there are eight cases where all $h^{\text{LM-cut}}$ estimates are perfect (upper part of the table), eight cases where the average additive error is at most 1 (middle part) and six cases where it is larger (bottom part). Compared to the other heuristics, the estimates of $h^{\text{LM-cut}}$ are best in all domains except FreeCell and Openstacks, where h^{LA} is more accurate.

In summary, the experiment shows that the $h^{\text{LM-cut}}$ heuristic provides excellent heuristic estimates. While we only compare heuristic estimates to h^+-based heuristics, we remark that such heuristics define the state of the art of admissible planning heuristic accuracy. In particular, Karpas and Domshlak (2009) report that h^{LA} is much more informative than current merge-and-shrink abstractions on large IPC benchmarks. Our results suggest that the landmark cut heuristic is even more informative.

Optimal Planning. To evaluate the usefulness of the landmark cut heuristic for optimal planning algorithms, we compare our A* implementation to four other heuristics: h^{LA}, $h^{\text{m\&s}}$ (merge-and-shrink abstractions of size 10,000), $h^{\max}$, and h^0 (i. e., blind search). All these heuristics were implemented within an otherwise identical planning system except that for h^{LA} we replaced A* with LM-A*, which performs better for this heuristic. (We also ran experiments with merge-and-shrink abstractions of size 100,000, which performed worse overall than the $h^{\text{m\&s}}$ results reported here.)

Additionally, we report results for the winner and runner-up of IPC-2008, Gamer and HSP^*_{F}. Note that the h^0 heuristic corresponds to the baseline planner used at IPC-2008, which narrowly outperformed Gamer at the competition.

		$h^{\text{LM-cut}}$			h^{LA}			$h^{\max}$		
Inst.	h^*	h	Exp.	Time	h	Exp.	Time	h	Exp.	Time
Blocks										
#9-0	30	**16**	**13162**	**13.65**	**16**	260857	132.87	9	3840589	85.00
#9-1	28	**16**	**347**	**0.37**	**16**	14852	7.93	10	1200345	32.06
#9-2	26	**17**	**598**	**0.67**	**17**	6581	3.40	9	1211463	32.15
#10-0	34	**18**	**239854**	**316.84**	**18**			9		
#10-1	32	**19**	**28989**	**43.75**	**19**	1393515	919.68	8		
#10-2	34	**19**	**82883**	**131.31**	**19**			10		
#11-0	32	**19**	**63891**	**110.45**	**19**			8		
#11-1	30	**21**	**59335**	**120.92**	**21**			4		
#11-2	34	**19**	**53638**	**91.88**	**19**			9		
#12-0	34	**22**	**58123**	**133.34**	**22**			10		
#12-1	34	**22**	**6284**	**14.20**	**22**	272199	250.78	11		
#14-0	38	**25**	**100500**	**388.30**	**25**			10		
#14-1	36	**27**	**160350**	**762.76**	**27**			6		
Satellite										
#1	9	**8**	**10**	**0.00**	**8**	**10**	0.01	3	59	**0.00**
#2	13	**12**	**14**	**0.00**	**12**	**14**	**0.00**	3	940	**0.00**
#3	11	**10**	**16**	**0.02**	9	494	0.09	3	6822	0.11
#4	17	**17**	**26**	**0.05**	**17**	993	0.47	3	180815	3.37
#5	15	**14**	**41**	**0.26**	11	12013	7.78	3		
#6	20	**17**	**2584**	**9.85**	**17**	24200	25.19	3	10751017	371.43
#7	21	**20**	**934**	**14.10**	**20**	157984	290.03	3		
#9	27	**25**	**19855**	**673.06**	21			3		
Openstacks										
#1	23	17	1224	0.35	**21**	**601**	0.11	4	4016	**0.03**
#2	23	18	1565	0.38	**21**	**688**	0.12	4	4594	**0.04**
#3	23	17	1224	0.34	**21**	**593**	0.10	4	4016	**0.03**
#4	23	17	1224	0.35	**21**	**601**	0.11	4	4016	**0.03**
#5	23	17	1224	0.34	**21**	**592**	0.10	4	4016	**0.03**
#6	45	36	210469	234.67	**41**	**93267**	51.57	4	822514	**18.71**
#7	46	35	266533	269.67	**41**	**107679**	59.41	4	787163	**17.81**

Table 3: Detailed results (15 smallest Blocks tasks omitted). For each solved task, we report the optimal plan length h^* and, for each heuristic, the initial state estimate, number of node expansions, and search time (in seconds). Best results are highlighted in bold.

Table 2 shows the number of planning tasks solved by each planner. The two dominating approaches are A* with $h^{\text{LM-cut}}$ and LM-A* with h^{LA}, both of which solve more than 400 tasks, while the next best planner, Gamer, solves 312. To put these numbers in perspective, the margin between the two best planners and Gamer is about twice as large as the margin between Gamer and blind search. For 16 of the 22 domains, $h^{\text{LM-cut}}$ is one of the top performers, and in the other cases it usually gets close. The only clear exceptions are Gripper, where A* with imperfect heuristics cannot be competitive with BDD-based planners like Gamer for reasons discussed by Helmert and Röger (2008), and FreeCell.

Comparing $h^{\text{LM-cut}}$ to h^{LA}, we see that $h^{\text{LM-cut}}$ solves significantly more tasks than h^{LA} (450 vs. 422). Moreover, it solves more tasks than h^{LA} in 12 domains, while being worse in only one (FreeCell).

To provide some insights about *why* $h^{\text{LM-cut}}$ outperforms h^{LA}, Tab. 3 presents detailed experimental data for three domains, chosen as representatives of the three domain classes distinguished in the first experiment: Blocks (perfect h^+ estimates), Satellite (average errors between 0 and 1) and Openstacks (average errors greater than 1). We report results for $h^{\text{LM-cut}}$, h^{LA} and, as a baseline, $h^{\max}$.

We first discuss the Openstacks results; note that this is the domain where $h^{\text{LM-cut}}$ approximates h^+ worst among all domains in the first experiment. Table 3 shows that this leads to more expansions than h^{LA} (by a factor of 2.0–2.4) and consequently longer planning time (by a factor of 3.2–4.6). Still, both heuristics solve the same set of tasks. The best performer in this domain is $h^{\max}$, due to its comparatively fast heuristic computations and despite requiring many more node evaluations than $h^{\text{LM-cut}}$ and h^{LA}.

In Blocks and Satellite, $h^{\text{LM-cut}}$ requires much more time per node, but ends up faster overall due to better heuristic guidance, in particular for large instances. The same behaviour can be observed in many of the other domains.

The gap in heuristic quality between $h^{\text{LM-cut}}$ and h^{LA} in Blocks is huge: $h^{\text{LM-cut}}$ solves eight additional tasks and requires a factor of 43–48 fewer expansions on the two largest commonly solved tasks. At first glance, this appears surprising: according to Tab. 1, *both* heuristics always obtain perfect h^+ estimates in this domain. The reason for the discrepancy is that $h^{\text{LM-cut}}$ *recomputes all relevant information for every search state*, while h^{LA} relies on expensive precomputations performed only for the initial state. Therefore, the quality of h^{LA} deteriorates as search moves away from the initial state. We remark that this strong reliance on the initial state is not unique to h^{LA}: it is shared by all current abstraction heuristics and additive $h^{\max}$ partitioning algorithms.

Concluding our discussion of the second experiment, we think it is fair to say that the landmark cut heuristic advances the state of the art in optimal sequential planning.

Conclusion

The main motivation for this work was the lack of formal results that connect different kinds of admissible planning heuristics. We have contributed towards changing this by providing several compilation results for delete relaxations, critical paths, abstractions and landmarks. It turns out that these concepts are much more closely related than was previously known. However, open questions remain. In particular, we still do not know how hard it is to find an optimal cost decomposition for $h^{\max}$, although we have made some progress towards answering this question. We also do not know if abstraction heuristics are powerful enough to capture the h^m heuristics for larger values of m.

Theoretical results of the kind presented in this paper are occasionally criticized for not being of practical relevance. We believe these results to be highly relevant because they open new avenues towards finding better planning heuristics, combining the strengths of previously disconnected concepts. We have explored one such avenue: the landmark cut heuristic, based on our compilation from additive $h^{\max}$ to landmark heuristics, is highly informative and leads to a very strong optimal planning algorithm. Other avenues remain to be explored. Our results show that abstraction heuristics are theoretically more powerful than relaxation heuristics, yet the best-performing heuristics in use today are relaxation heuristics. Trying to leverage the power of abstraction heuristics in a way that makes them consistently competitive with good relaxation heuristics remains an important open problem.

Acknowledgments

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). Carmel Domshlak was partly supported by ISF grant 670/70.

References

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

Betz, C. 2009. Komplexität und Berechnung der h^+-Heuristik. Diplomarbeit, Albert-Ludwigs-Universität Freiburg.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *AIJ* 129(1):5–33.

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

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. Additive-disjunctive heuristics for optimal planning. In *Proc. ICAPS 2008*, 44–51.

Edelkamp, S. 2001. Planning with pattern databases. In *Proc. ECP 2001*, 13–24.

Edelkamp, S. 2002. Symbolic pattern databases in heuristic search planning. In *Proc. AIPS 2002*, 274–283.

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

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proc. AAAI 2007*, 1007–1012.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *Proc. AAAI 2005*, 1163–1168.

Helmert, M., and Mattmüller, R. 2008. Accuracy of admissible heuristic functions in selected planning domains. In *Proc. AAAI 2008*, 938–943.

Helmert, M., and Röger, G. 2008. How good is almost perfect? In *Proc. AAAI 2008*, 944–949.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proc. ICAPS 2007*, 176–183.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Hoffmann, J. 2005. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *JAIR* 24:685–758.

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

Katz, M., and Domshlak, C. 2008a. Optimal additive composition of abstraction-based admissible heuristics. In *Proc. ICAPS 2008*, 174–181.

Katz, M., and Domshlak, C. 2008b. Structural patterns heuristics via fork decomposition. In *Proc. ICAPS 2008*, 182–189.

Keyder, E., and Geffner, H. 2008. Heuristics for planning with action costs revisited. In *Proc. ECAI 2008*, 588–592.

Mirkis, V., and Domshlak, C. 2007. Cost-sharing approximations for h^+. In *Proc. ICAPS 2007*, 240–247.

Porteous, J.; Sebastia, L.; and Hoffmann, J. 2001. On the extraction, ordering, and usage of landmarks in planning. In *Proc. ECP 2001*, 37–48.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proc. AAAI 2008*, 975–982.