# A Comparison of Cost Partitioning Algorithms for Optimal Classical Planning

**Jendrik Seipp** and **Thomas Keller** and **Malte Helmert**
University of Basel
Basel, Switzerland
{jendrik.seipp,tho.keller,malte.helmert}@unibas.ch

## Abstract

Cost partitioning is a general and principled approach for constructing additive admissible heuristics for state-space search. Cost partitioning approaches for optimal classical planning include optimal cost partitioning, uniform cost partitioning, zero-one cost partitioning, saturated cost partitioning, post-hoc optimization and the canonical heuristic for pattern databases. We compare these algorithms theoretically, showing that saturated cost partitioning dominates greedy zero-one cost partitioning. As a side effect of our analysis, we obtain a new cost partitioning algorithm dominating uniform cost partitioning. We also evaluate these algorithms experimentally on pattern databases, Cartesian abstractions and landmark heuristics, showing that saturated cost partitioning is usually the method of choice on the IPC benchmark suite.

## Introduction

Optimal planning as heuristic search requires a heuristic function that is admissible, i.e., never overestimates the cost of the cheapest plan from any state. A simple approach that allows to use multiple admissible heuristics is to use the highest estimate in each state. *Cost partitioning* (Katz and Domshlak 2008; Yang et al. 2008) is an alternative that actually *combines* information from individual estimates instead of just selecting the most accurate one. It distributes operator costs among the heuristics, allowing to add up the heuristic estimates admissibly.

Before the emergence of cost partitioning methods, additive admissible heuristics such as *disjoint pattern databases* exploited the natural independence between multiple heuristics that arises when no operator contributes to the estimate of more than one heuristic (Korf and Felner 2002; Felner, Korf, and Hanan 2004; Edelkamp 2006). The further development of this idea led to the *canonical heuristic* for pattern databases (Haslum et al. 2007), which computes all maximal subsets of pairwise independent abstractions and then uses the maximum of all sums over independent abstractions as the heuristic value. In a formal sense, this is the most accurate heuristic that can be derived from a given set of heuristics if the only available information apart from the heuristic values is which pairs of heuristics are independent.



*Zero-one cost partitioning* was introduced as a generalization of such independence-based additive heuristics. This approach, first formally described by Haslum, Bonet, and Geffner (2005), artificially enforces independence between heuristics by treating operators as free of cost if they have already been "used up" by another heuristic.

The development of cost partitioning as a general concept is due to Katz and Domshlak (2008), who also introduced *uniform cost partitioning* as a practical cost partitioning method: instead of assigning the whole cost of an operator to a single heuristic, the cost is equally distributed among all heuristics for which the operator is relevant.

A more recent addition to the set of cost partitioning algorithms is *saturated cost partitioning* (Seipp and Helmert 2014). To compute it, the heuristics are arranged in an ordered sequence and in turn offered the (initially full) cost of each operator. Each heuristic "consumes" as much or as little of the operator cost as is needed to preserve all heuristic estimates, and the remaining costs are then offered to the following heuristics in the sequence until all heuristics have been served in this way.

All these approaches are theoretically dominated by *optimal cost partitioning*, which has been shown to be computable in polynomial time for abstraction (Katz and Domshlak 2008; 2010) and landmark (Karpas and Domshlak 2009) heuristics and has recently been extended to permit negative costs (Pommerening et al. 2015).

However, these promising theoretical results do (so far) not translate well into practice: Pommerening, Röger, and Helmert (2013) show that even for comparatively small pattern database heuristics, optimal cost partitioning tends to be prohibitively expensive, as it requires to solve linear programs with "up to millions of variables and billions of constraints for realistic problem sizes". Their *post-hoc optimization heuristic* is an approximation to optimal cost partitioning based on a linear program where a single weight is computed for each heuristic, which dramatically reduces the solution space, making the approach much cheaper but less accurate than optimal cost partitioning.

As this short literature overview shows, cost partitioning is an active (and, due to the high quality of cost-partitioned heuristics demonstrated in many experiments, important) research area within planning as heuristic search. However, apart from a few isolated results, no thorough theoretical or

experimental analysis of cost partitioning approaches exists.

This paper provides such an analysis. Theoretically, we prove a number of dominance and non-dominance results, including the previously unreported fact that saturated cost partitioning dominates greedy zero-one cost partitioning. It turns out that the key idea that distinguishes these two cost partitioning approaches can also be applied to uniform cost partitioning, leading to a new *opportunistic* version of uniform cost partitioning that dominates the original.

Experimentally, we report results for pattern databases that are derived systematically (Pommerening, Röger, and Helmert 2013) and by hill climbing (Haslum et al. 2007), for Cartesian abstractions (Seipp and Helmert 2013), and for landmark heuristics (Karpas and Domshlak 2009), showing that earlier evidence for the strength of saturated cost partitioning for Cartesian abstractions (Seipp, Keller, and Helmert 2017) generalizes to all considered kinds of heuristics.

## Background

Cost partitioning is a method that can be applied to derive admissible heuristics for state-space search problems in general, and hence our problem formalization is not specific to classical planning. A *state space* is a directed, labeled graph $\mathcal{T} = \langle S, \mathcal{L}, c, T, s_{\mathrm{I}}, S_\star \rangle$, where $S$ is a finite set of *states*; $\mathcal{L}$ is a finite set of *labels*; $c : \mathcal{L} \mapsto \mathbb{R}$ is a (possibly negative) *cost function*; $T$ is a set of labeled and weighted *transitions* $s \xrightarrow{l,c(l)} s'$ for $s, s' \in S$ and $l \in \mathcal{L}$; $s_{\mathrm{I}} \in S$ is the *initial state*; and $S_\star \subseteq S$ is the set of *goal states*. A state space is *regular* if $c(l) \geq 0$ for all labels $l$. We mainly consider regular state spaces, but permit negative costs within cost partitionings.

The *goal distance* $h^*(s) \in \mathbb{R} \cup \{-\infty, \infty\}$ of a state $s \in S$ is the cost of a cheapest path from $s$ to a goal state in $S_\star$. It is $\infty$ if no such path exists and $-\infty$ if paths of arbitrarily low negative cost exist, which cannot happen if $\mathcal{T}$ is regular.

For cost partitioning, it is important that heuristic estimates can be computed for varying cost functions, so we formally define a *heuristic* as a function from cost functions and states to cost estimates and write $h^{c'}(s) \in \mathbb{R} \cup \{-\infty, \infty\}$ for the heuristic estimate of state $s$ under cost function $c'$. A heuristic $h$ is *cost-monotonic* if $h^c(s) \geq h^{c'}(s)$ for all states $s$ whenever $c \geq c'$ (i.e., making transitions more expensive cannot reduce heuristic estimates). A heuristic $h$ is *admissible* if $h^{c'}(s) \leq h^*_{c'}(s)$ for all cost functions $c'$ and states $s$, where $h^*_{c'}$ is the goal distance in the transition system $\mathcal{T} = \langle S, \mathcal{L}, c', T, s_{\mathrm{I}}, S_\star \rangle$.

Label $l$ *affects* heuristic $h$ if heuristic estimates of $h$ may depend on $c(l)$, i.e., there exist states $s$ and non-negative cost functions $c$ and $c'$ which differ only on $l$ with $h^c(s) \neq h^{c'}(s)$. We define $\mathcal{A}(h) = \{l \in \mathcal{L} \mid l \text{ affects } h\}$. Heuristics $h$ and $h'$ with $\mathcal{A}(h) \cap \mathcal{A}(h') = \emptyset$ are called *independent*.

Some cost partitioning algorithms require the information how much of the cost of a label actually contributes to the heuristic estimates of $h$. This is described by the *saturated cost function* for heuristic $h$ and cost $c$, written *saturate*$(h, c)$, which is the minimal cost function $c' \leq c$ with $h^{c'}(s) = h^c(s)$ for all states $s$ (Seipp and Helmert 2014). A unique minimum does not exist for all classes of heuristics; a sufficient condition is that $h$ is an *abstraction heuristic* and $h^c(s)$ is finite for all states. (The latter restriction is not limiting, as infinities can be handled separately.)

## Cost Partitioning

For challenging state-space search problems, using a single admissible heuristic is often not very informative. It can therefore be beneficial to generate multiple admissible heuristics that focus on different aspects of the problem (e.g., Holte et al. 2006). Cost partitioning is a general way to make the sum of such heuristics admissible (Katz and Domshlak 2008). Following a recent generalization by Pommerening et al. (2015), we consider state spaces with non-negative costs, but allow negative costs for the component heuristics.

**Definition 1.** ***Cost partitioning.***
*Let $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$ be a tuple of admissible heuristics for a regular state space $\mathcal{T} = \langle S, \mathcal{L}, c, T, s_{\mathrm{I}}, S_\star \rangle$. A* cost partitioning *over $\mathcal{H}$ is a tuple $\mathcal{C} = \langle c_1, \ldots, c_n \rangle$ of (general) cost functions whose sum is bounded by $c$: $\sum_{i=1}^n c_i(l) \leq c(l)$ for all $l \in \mathcal{L}$. The* cost-partitioned heuristic *$h^{\mathcal{C}}$ is defined as $h^{\mathcal{C}}(s) := \sum_{i=1}^n h_i^{c_i}(s)$.*

Cost-partitioned heuristics are always admissible. Intuitively, since each heuristic yields an admissible estimate for the search problem under consideration of the original cost function, their sum remains admissible due to the way $\mathcal{C}$ distributes the individual costs among the components.

### Optimal Cost Partitioning

An optimal cost partitioning for a given state is a cost partitioning $\mathcal{C}^*$ where $h^{\mathcal{C}^*}(s)$ is maximal among all possible cost partitionings.

**Definition 2.** ***Optimal cost partitioning.***
*Given a regular state space $\mathcal{T}$ with states $S$ and a tuple of admissible heuristics $\mathcal{H}$, a cost partitioning $\mathcal{C}^*$ is an* optimal cost partitioning *for state $s \in S$ if $h^{\mathcal{C}^*}(s) \geq h^{\mathcal{C}}(s)$ for all cost partitionings $\mathcal{C}$. We write $h^{OCP}$ for the heuristic that is cost-partitioned with an optimal cost partitioning in each state.*

Katz and Domshlak (2008; 2010) showed that optimal cost partitionings can be found in polynomial time by linear programming for a wide range of abstraction heuristics, a result that has strongly influenced the further development of the theory and practice of cost-partitioned heuristics. Computing optimal cost partitionings for some or even all states encountered during search has been shown to be a practically viable approach for landmark heuristics and certain classes of implicit abstraction heuristics (Karpas and Domshlak 2009; Katz and Domshlak 2010; Karpas, Katz, and Markovitch 2011).

However, computing optimal cost partitionings can already be prohibitively expensive for abstractions of modest size: Pommerening, Röger, and Helmert (2013), for example, performed experiments with systematic pattern databases of up to size 2, showing that for 249 tasks in their benchmark set, computing an optimal cost partitioning for a single state is infeasible even with a 24-hour time limit

and 2 GiB memory limit. This includes 206 cases where the LP computation runs out of memory and 43 timeouts after 24 hours. Similarly, Seipp, Keller, and Helmert (2017) noted that there are 211 tasks which could be optimally solved when using their best suboptimal cost partitioning algorithm, while computing an optimal cost partitioning for only a single state takes longer than 30 minutes.

In the following, we discuss several alternatives to optimal cost partitioning that aim to balance the trade-off between computational effort and accuracy of the resulting cost-partitioned heuristic.

## Post-hoc Optimization

Like optimal cost partitioning, *post-hoc optimization* (Pommerening, Röger, and Helmert 2013) is a cost partitioning method based on linear programming. Each component heuristic is assigned a single real-valued weight in the range $[0, 1]$, and the overall heuristic value is the weighted sum of component heuristics. For each label, there is a constraint that ensures that the total weight of all heuristics affected by the label sum up to at most 1. This corresponds to a cost partitioning where operators that do not affect a given heuristic are assigned a cost of 0, and operators affecting a heuristic which receives the weight $w_i$ are assigned the fraction $w_i$ of their full cost.

**Definition 3.** ***Post-hoc Optimization.***
*Let $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$ be a tuple of admissible heuristics for regular state space $\mathcal{T}$ with cost function $c$, and let $\langle w_1, \ldots, w_n \rangle$ be a solution to the linear program that maximizes $\sum_{i=1}^{n}(w_i \cdot h_i(s))$ subject to*

$$\sum_{i \in \{1,\ldots,n\}: l \in \mathcal{A}(h_i)} w_i \leq 1 \text{ for all } l \in \mathcal{L}$$

$$w_i \geq 0.$$

*Then, the* post-hoc optimization cost partitioning *is the tuple $\mathcal{C} = \langle w_1 \cdot c_1, \ldots, w_n \cdot c_n \rangle$, where $c_i(l) = c(l)$ if $l \in \mathcal{A}(h_i)$ and $c_i(l) = 0$ otherwise. We write $h^{PHO}$ for the heuristic that is cost-partitioned with the post-hoc optimization cost partitioning.*

Post-hoc optimization always generates a non-negative cost partitioning, i.e., one where all component costs are non-negative. Following the results of Pommerening et al. (2015) on general cost partitioning, one might wonder if $h^{\text{PHO}}$ could be strengthened by dropping the non-negativity constraint $w_i \geq 0$. However, this does not work as expected, as $\sum_{i=1}^{n}(w_i \cdot h_i(s))$ is no longer an admissible estimate without this constraint. The reason for this is that negative weighting changes which paths are optimal in a state space.

## Zero-One Cost Partitioning

While post-hoc optimization considers a much simpler linear program than optimal cost partitioning, it still requires an optimization to be performed in every state. This is not the case in a zero-one cost partitioning (Haslum, Bonet, and Geffner 2005; Edelkamp 2006), which is precomputed and applied to all states. In a zero-one cost partitioning, the whole cost of each label is assigned to (at most) a single component.

**Definition 4.** ***Zero-one cost partitioning.***
*Given a regular state space $\mathcal{T}$ and a tuple of admissible heuristics $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$, a tuple $\mathcal{C} = \langle c_1, \ldots, c_n \rangle$ is a* zero-one cost partitioning *if for each $l \in \mathcal{L}$ we have $c_i(l) = c(l)$ for at most one $c_i \in \mathcal{C}$ and $c_j(l) = 0$ for all other $c_j \in \mathcal{C}$.*

For a state space with $l$ labels and $n$ admissible heuristics, Definition 4 allows for $(n+1)^l$ different zero-one cost partitionings. The question is therefore how to obtain an informative zero-one cost partitioning. The only method considered in the literature is a greedy algorithm that iterates over the heuristics in a specified order and assigns the cost of each label to the first heuristic that is affected by this label. Due to this greedy assignment, the algorithm is susceptible to the order in which the heuristics are considered.

**Definition 5.** ***Greedy zero-one cost partitioning.***
*Given a regular state space $\mathcal{T}$ and a set of admissible heuristics $\mathcal{H} = \{h_1, \ldots, h_n\}$ for $\mathcal{T}$, the set of orders of $\mathcal{H}$, denoted by $\Omega(\mathcal{H})$, consists of all permutations of $\mathcal{H}$, i.e., all tuples of heuristics obtained by ordering $\mathcal{H}$ in any way.*

*For a given order $\omega = \langle h_1, \ldots, h_n \rangle \in \Omega(\mathcal{H})$, the* greedy zero-one cost partitioning *is the tuple $\mathcal{C} = \langle c_1, \ldots, c_n \rangle$, where*

$$c_i(l) = \begin{cases} c(l) & \text{if } l \in \mathcal{A}(h_i) \text{ and } l \notin \bigcup_{j=1}^{i-1} \mathcal{A}(h_j) \\ 0 & \text{otherwise} \end{cases}$$

*for all $l \in \mathcal{L}$. We write $h^{GZOCP}_{\omega}$ for the heuristic that is cost-partitioned by greedy zero-one cost partitioning for order $\omega$.*

Each greedy zero-one cost partitioning is a zero-one cost partitioning as the cost of each label is assigned to at most one heuristic (the first one in the order affected by the label).

## Saturated Cost Partitioning

Greedy zero-one cost partitioning always assigns the full cost of a label $l$ to the first heuristic $h$ affected by $l$, even if $h$ can only benefit from a small fraction of the cost. To avoid "wasting" costs, Seipp and Helmert (2014) proposed saturated cost partitioning. This algorithm also distributes costs greedily by considering the component heuristics in sequence, but unlike greedy zero-one cost partitioning it only assigns as much cost to each heuristic as that heuristic can usefully exploit. The remaining costs are therefore saved for subsequent heuristics. The generalization to possibly negative component costs considered here has first been described in the context of state-dependent cost partitioning (Keller et al. 2016).

**Definition 6.** ***Saturated cost partitioning.***
*Let $\mathcal{T}$ be a regular state space and $\mathcal{H}$ be a set of admissible heuristics. Given an order $\omega = \langle h_1, \ldots, h_n \rangle \in \Omega(\mathcal{H})$, the* saturated cost partitioning *$\mathcal{C} = \langle c_1, \ldots, c_n \rangle$ and the* remaining cost functions *$\langle \bar{c}_0, \ldots, \bar{c}_n \rangle$ are defined by*

$$\bar{c}_0 = c$$
$$c_i = \mathit{saturate}(h_i, \bar{c}_{i-1})$$
$$\bar{c}_i = \bar{c}_{i-1} - c_i$$

*We write $h^{SCP}_{\omega}$ for the heuristic that is cost-partitioned by saturated cost partitioning for order $\omega$.*

Whether and how the saturated cost function required for saturated cost partitioning can be computed efficiently depends on the type of heuristic. If $h$ is an abstraction heuristic, the saturated cost of operator $o$ is the maximum over $h(s) - h(s')$ for all abstract state transitions $s \to s'$ induced by $o$. For explicit-state abstraction heuristics based on pattern databases or *Cartesian abstraction* (Ball, Podelski, and Rajamani 2001; Seipp and Helmert 2013), this can be computed at negligible overhead during the construction of the heuristic. The same is true for merge-and-shrink heuristics not using *label reduction* (Sievers, Wehrle, and Helmert 2014). For merge-and-shrink heuristics using label reduction, computing the saturated cost function is more expensive, but still polynomial.

## Uniform Cost Partitioning

Katz and Domshlak (2008) proposed uniform cost partitioning, where the cost of each label is distributed uniformly among all heuristics affected by this label.

**Definition 7.** ***Uniform cost partitioning.***
*Given a regular state space $\mathcal{T}$ and a tuple of admissible heuristics $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$, the* uniform cost partitioning *is the tuple $\mathcal{C} = \langle c_1, \ldots, c_n \rangle$, where for all $l \in \mathcal{L}$*

$$c_i(l) = \begin{cases} \frac{c(l)}{|\{h \in \mathcal{H} | l \in \mathcal{A}(h)\}|} & \text{if } l \in \mathcal{A}(h_i) \\ 0 & \text{otherwise.} \end{cases}$$

*We write $h^{UCP}$ for the heuristic that is cost-partitioned by uniform cost partitioning.*

Unlike (greedy) zero-one and saturated cost partitioning, uniform cost partitioning is not affected by the order in which the heuristics are considered.

## Opportunistic Uniform Cost Partitioning

Uniform cost partitioning suffers from the same problem as greedy zero-one cost partitioning: even if costs are not fully consumed by a heuristic, they are not offered to other heuristics where the increased cost function might lead to increased (yet still admissible) estimates.

We propose a variant that remedies this shortcoming. Like uniform cost partitioning, we propose to split the label costs evenly among the heuristics affected by a label, but like saturated cost partitioning, heuristics are considered in sequence and any unneeded costs are saved and redistributed to the heuristics encountered later in the sequence.

**Definition 8.** ***Opportunistic uniform cost partitioning.***
*Let $\mathcal{T}$ be a state-space and $\mathcal{H}$ be a set of admissible heuristics. Given an order $\omega = \langle h_1, \ldots, h_n \rangle \in \Omega(\mathcal{H})$, the* opportunistic uniform cost partitioning *$\mathcal{C} = \langle c_1, \ldots, c_n \rangle$, the* remaining cost functions *$\langle \overline{c}_0, \ldots, \overline{c}_n \rangle$ and the* offered cost functions *$\langle \tilde{c}_1, \ldots, \tilde{c}_n \rangle$ are defined by*

$$\overline{c}_0 = c$$

$$\tilde{c}_i(l) = \begin{cases} \frac{\overline{c}(l)_{i-1}}{|\{h \in \{h_i, \ldots, h_n\} | l \in \mathcal{A}(h)\}|} & \text{if } l \in \mathcal{A}(h_i) \\ 0 & \text{otherwise} \end{cases}$$

$$c_i = \mathit{saturate}(h_i, \tilde{c}_i)$$

$$\overline{c}_i = \overline{c}_{i-1} - c_i$$

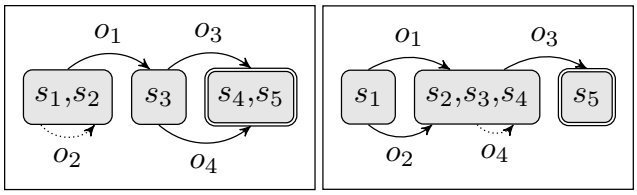


Figure 1: Abstractions used in the proofs of Theorems 1, 2 and 3. Operators $o_1$ and $o_3$ cost 4, whereas $o_2$ and $o_4$ cost 1.

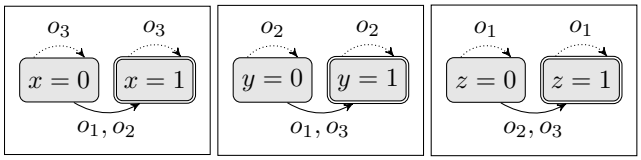


Figure 2: (Figure 1 in Seipp, Keller, and Helmert 2017) Abstractions used in the proof of Theorem 3. The initial state $s_\mathrm{I}$ is $\{x \mapsto 0, y \mapsto 0, z \mapsto 0\}$. All operators cost 1.

*We write $h^{OUCP}_\omega$ for the heuristic that is cost-partitioned by opportunistic uniform cost partitioning for order $\omega$.*

As with saturated cost partitioning, the component costs for opportunistic uniform cost partitioning may be negative, which can lead to higher remaining costs and hence potentially higher overall heuristic estimates.

## Canonical Heuristic

Haslum et al. (2007) introduced the canonical heuristic as a heuristic that allows the combination of information from multiple pattern database heuristics. We give a definition for general admissible heuristics.

**Definition 9.** ***Canonical Heuristic.***
*Let $\mathcal{H}$ be a tuple of admissible heuristics for regular state space $\mathcal{T}$, and let MIS be the set of all maximal (w.r.t. set inclusion) subsets of $\mathcal{H}$ such that all heuristics in each subset are independent. The* canonical heuristic *in state $s \in S$ is*

$$h^{CAN}(s) = \max_{\sigma \in MIS} \sum_{h \in \sigma} h(s).$$

# Theoretical Evaluation

We now study the relationships between the introduced cost partitioning algorithms. Apart from a result by Pommerening, Röger, and Helmert (2013), who show that post-hoc optimization dominates the canonical heuristic, we are not aware of formal comparisons of these algorithms in the literature. We begin with two theorems that show that saving unused costs is beneficial with cost-monotonic heuristics.

**Theorem 1.** ***$h^{SCP}$ dominates $h^{GZOCP}$***
*Let $\mathcal{T}$ be a regular state space and $\mathcal{H}$ be a set of cost-monotonic admissible heuristics for $\mathcal{T}$. Then $h^{SCP}_\omega(s) \geq h^{GZOCP}_\omega(s)$ for all orders $\omega \in \Omega(\mathcal{H})$ and all $s \in S$. Moreover, there are cases where the inequality is strict for some $s \in S$ and all orders $\omega \in \Omega(\mathcal{H})$.*

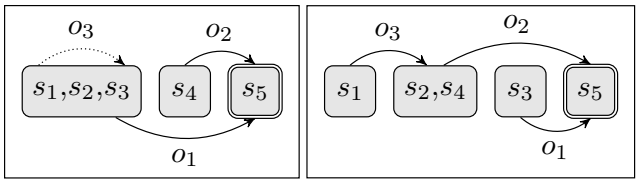


Figure 3: Abstractions used in the proofs of Theorems 3 and 4. Operator $o_1$ costs 4, and $o_2$ and $o_3$ cost 1.

*Proof.* For the second part, Figure 1 is an example with $h^{\text{SCP}}_{\omega}(s_1) = 8$ and $h^{\text{GZOCP}}_{\omega}(s_1) = 5$ for all $\omega \in \Omega(\mathcal{H})$.

For the first part, we show the stronger result $h^{\text{SCP}}_{\omega',c'}(s) \geq h^{\text{GZOCP}}_{\omega',c''}(s)$ for all orders of admissible cost-monotonic heuristics $\omega'$ and non-negative cost functions $c' \geq c''$, where the subscripts $c'$ and $c''$ indicate that the heuristics are evaluated in a modified state space with the given cost function. The theorem follows from the case $\omega = \omega'$ and $c = c' = c''$.

Strengthening the claim allows proving the result by induction over the length of $\omega'$. For the empty sequence $\omega'$, both heuristics are 0, so the inequality holds trivially.

Otherwise decompose $\omega'$ into the first component $h_1$ and remaining sequence $\omega''$. The value contributed by $h_1$ to $h^{\text{SCP}}_{\omega',c'}(s)$ is $h^{c'}_1(s)$ by definition of saturated cost. The value contributed by $h_1$ to $h^{\text{GZOCP}}_{\omega',c''}(s)$ is $h^{c''}_1(s)$ because $h_1$ receives the full operator costs from $c''$ for all labels affecting $h_1$. We get $h^{c'}_1(s) \geq h^{c''}_1(s)$ because $h_1$ is cost-monotonic.

For labels that do not affect $h_1$, SCP and GZOCP assign cost 0 to $h_1$, and hence the remaining costs for $\omega''$ are at least as large under SCP as under GZOCP. For labels that affect $h_1$, GZOCP uses up the whole cost for $h_1$, so the remaining costs for $\omega''$ are again at least as large under SCP as under GZOCP because the latter are 0. By the induction hypothesis, the heuristic value contributed by $\omega'$ is then at least as large for SCP as for GZOCP, concluding the proof. □

**Theorem 2.** ***$h^{OUCP}$ dominates $h^{UCP}$***
*Let $\mathcal{T}$ be a regular state space and $\mathcal{H}$ be a set of cost-monotonic admissible heuristics for $\mathcal{T}$. Then $h^{OUCP}_{\omega}(s) \geq h^{UCP}(s)$ for all orders $\omega \in \Omega(\mathcal{H})$ and all $s \in S$. Moreover, there are cases where the inequality is strict for some $s \in S$ and all orders $\omega \in \Omega(\mathcal{H})$.*

*Proof.* For the second part, Figure 1 is an example with $h^{\text{OUCP}}_{\omega}(s_1) = 7$ for all $\omega \in \Omega(\mathcal{H})$ and $h^{\text{UCP}}(s_1) = 6$.

The proof of the first part is analogous to the proof of Theorem 1. The only difference is that only a fraction of the label cost of a label $l$ affecting $h_1$ may be used for $h_1$, but because this fraction is the same for OUCP and UCP ($1/k$, where $k$ is the number of heuristics in $\omega'$ affected by $l$), this does not make a difference to the proof argument. □

We now know that there are three cost partitioning algorithms – saturated cost partitioning, opportunistic uniform cost partitioning and post-hoc optimization – that dominate one of the other three discussed algorithms. Next, we show that none of these three algorithms dominates any of the other two.

**Theorem 3.** ***Comparison of $h^{SCP}$, $h^{OUCP}$, and $h^{PHO}$***
*For each of the following cost partitioning algorithms, there exists a regular state space $\mathcal{T}$ and a set of cost-monotonic admissible heuristics $\mathcal{H}$ such that*

$$h^{OUCP}_{\omega}(s) > h^{SCP}_{\omega}(s) \quad (1)$$
$$h^{SCP}_{\omega}(s) > h^{OUCP}_{\omega}(s) \quad (2)$$
$$h^{PHO}(s) > h^{SCP}_{\omega}(s) \quad (3)$$
$$h^{SCP}_{\omega}(s) > h^{PHO}(s) \quad (4)$$
$$h^{PHO}(s) > h^{OUCP}_{\omega}(s) \quad (5)$$
$$h^{OUCP}_{\omega}(s) > h^{PHO}(s) \quad (6)$$

*for a state $s \in S$ and all orders $\omega \in \Omega(\mathcal{H})$.*

*Proof.* Consider the two abstractions in Figure 1. For all orders $\omega \in \Omega(\mathcal{H})$, we have $h^{\text{SCP}}_{\omega}(s_1) = 8$, $h^{\text{OUCP}}_{\omega}(s_1) = 7$ and $h^{\text{PHO}}(s_1) = 5$, showing (2), (4) and (6).

Consider the three abstractions in Figure 2. For all orders $\omega \in \Omega(\mathcal{H})$, we have $h^{\text{OUCP}}_{\omega}(s_{\text{I}}) = h^{\text{PHO}}(s_{\text{I}}) = 0.5 + 0.5 + 0.5 = 1.5$ and $h^{\text{SCP}}_{\omega}(s_{\text{I}}) = 1$, showing (1) and (3).

Consider the two abstractions in Figure 3. We have $h^{\text{PHO}}(s_1) = 4$ and for all orders $\omega \in \Omega(\mathcal{H})$, $h^{\text{OUCP}}_{\omega}(s_1) = 3.5$, showing (5). □

Our last dominance result compares the canonical heuristic to greedy zero-one cost partitioning. While no dominance result exists between $h^{\text{CAN}}$ and a *single* heuristic $h^{\text{GZOCP}}_{\omega}$, we can maximize over multiple zero-one cost-partitioned heuristics with different orders in the same way that $h^{\text{CAN}}$ maximizes over multiple independent sets of heuristics, and for such a maximum heuristic, a dominance result can be established.

Formally, if $\Omega$ is a set of orders for heuristics $\mathcal{H}$, we define $h^{\text{GZOCP}}_{\Omega}(s) := \max_{\omega \in \Omega} h^{\text{GZOCP}}_{\omega}(s)$ and $h^{\text{SCP}}_{\Omega} := \max_{\omega \in \Omega} h^{\text{SCP}}_{\omega}(s)$. We show that both of these heuristics dominate the canonical heuristic if the orders $\Omega$ are suitably chosen.

**Theorem 4.** ***$h^{GZOCP}_{\Omega}$ dominates $h^{CAN}$***
*Let $\mathcal{T}$ be a regular state space and $\mathcal{H}$ be a set of admissible heuristics for $\mathcal{T}$. Then there is a set of orders $\Omega \subseteq \Omega(\mathcal{H})$ with $h^{GZOCP}_{\Omega}(s) \geq h^{CAN}(s)$ for all $s \in S$. Moreover, there are cases where the inequality is strict for some $s \in S$.*

*Proof.* Given a $\sigma = \{h_1, \ldots, h_n\} \in \text{MIS}$, we construct an order $\omega$ by appending all $h \in \mathcal{H} \setminus \sigma$ in arbitrary order to the tuple $\langle h_1, \ldots, h_n \rangle$. Due to the pairwise independence of all heuristics in $\sigma$, all $l \in \mathcal{L}$ affect at most one heuristic in $\sigma$ and hence $h^{\text{GZOCP}}_{\omega}$ assigns their full cost to any $h_i \in \sigma$ which they affect. Therefore, we get $h^{\text{GZOCP}}_{\omega}(s) \geq \sum_{h \in \sigma} h(s)$ for all $s \in S$. The dominance claim follows by setting $\Omega$ to the set of all orders that can be constructed in this way from any $\sigma \in \text{MIS}$.

For a case where the inequality is strict, consider the example from Figure 3. The two heuristics are not independent, and therefore the set of all maximal independent subsets of $\mathcal{H}$ contains each heuristic individually, yielding $h^{\text{CAN}}(s_1) = 4$. However, $h^{\text{GZOCP}}_{\omega}(s_1) = 5$ for the order $\omega$ which considers the left abstraction first. □

| | $h^{\text{UCP}}$ | $h^{\text{OUCP}}_{\text{one}}$ | $h^{\text{OUCP}}_{\text{div}}$ | $h^{\text{GZOCP}}_{\text{one}}$ | $h^{\text{GZOCP}}_{\text{div}}$ | $h^{\text{SCP}}_{\text{one}}$ | $h^{\text{SCP}}_{\text{div}}$ | $h^{\text{CAN}}$ | $h^{\text{PHO}}$ | $h^{\text{OCP}}$ | coverage | std. dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $h^{\text{UCP}}$ | – | 2 | 1 | 16 | 1 | 6 | 1 | 1 | 7 | 33 | 788.0 | – |
| $h^{\text{OUCP}}_{\text{one}}$ | 11 | – | 2 | 20 | 2 | 8 | 2 | 5 | 13 | 34 | 798.1 | 2.77 |
| $h^{\text{OUCP}}_{\text{div}}$ | 11 | 11 | – | 19 | 2 | 12 | 1 | 5 | 14 | 34 | 807.9 | 0.32 |
| $h^{\text{GZOCP}}_{\text{one}}$ | 1 | 1 | 1 | – | 1 | 1 | 1 | 2 | 5 | 33 | 771.7 | 4.37 |
| $h^{\text{GZOCP}}_{\text{div}}$ | 14 | 15 | 6 | 19 | – | 12 | 1 | 4 | 15 | 35 | 816.7 | 0.48 |
| $h^{\text{SCP}}_{\text{one}}$ | 13 | 12 | 4 | 19 | 4 | – | 1 | 8 | 17 | 34 | 799.2 | 3.33 |
| $h^{\text{SCP}}_{\text{div}}$ | 18 | 18 | 9 | 22 | 7 | 17 | – | 10 | 19 | 35 | **830.6** | 0.52 |
| $h^{\text{CAN}}$ | 13 | 13 | 5 | 19 | 2 | 11 | 0 | – | 12 | 35 | 813.0 | – |
| $h^{\text{PHO}}$ | 6 | 6 | 4 | 16 | 1 | 7 | 0 | 0 | – | 35 | 795.0 | – |
| $h^{\text{OCP}}$ | 3 | 2 | 2 | 3 | 1 | 2 | 1 | 1 | 1 | – | 469.0 | – |

Table 1: Left: Pairwise comparison of cost partitioning algorithms using PDBs generated by hill climbing. The entry in row $x$ and column $y$ holds the number of domains in which algorithm $x$ solved more tasks than algorithm $y$. Right: Total number of solved tasks by each algorithm. Results for randomized algorithms are averaged over 10 runs.

**Corollary 1.** ***$h^{SCP}_{\Omega}$ dominates $h^{CAN}$***
*Let $\mathcal{T}$ be a regular state space and $\mathcal{H}$ be a set of cost-monotonic heuristics for $\mathcal{T}$. Then there is a set of orders $\Omega \subseteq \Omega(\mathcal{H})$ where $h^{SCP}_{\Omega}(s) \geq h^{CAN}(s)$ for all $s \in S$. Moreover, there are cases where the inequality is strict for some $s \in S$.*

*Proof.* Follows directly from Theorem 1 and 4. □

As a final comment, we remark that the canonical heuristic still has an important advantage over the proposed $h^{\text{GZOCP}}_{\Omega}$ and $h^{\text{SCP}}_{\Omega}$, namely that it suffices to compute the component heuristics w.r.t. a single cost function. This is different for cost partitioning algorithms that maximize over a (possibly large) number of orders, with each order requiring a different cost function. This can be a concern especially for memory-based heuristics like PDB heuristics, where each cost function requires a separate PDB.

This concludes our theoretical investigation of the different cost partitioning algorithms, and we now turn to the experimental analysis.

## Experimental Evaluation

We implemented all cost partitioning algorithms in the Fast Downward planning system (Helmert 2006). For all conducted experiments, we limit time and memory to 30 minutes and 2 GiB. We use the 1667 benchmark tasks from 40 different domains from all optimization tracks of the 1998–2014 International Planning Competitions (IPC).

The accuracy of cost-partitioned heuristics generated by saturated cost partitioning greatly depends on the order in which the component heuristics are considered (Seipp, Keller, and Helmert 2017). By using multiple orders and maximizing over the produced cost partitionings, it is possible to obtain heuristics that are significantly more accurate than heuristics for a single order. In light of this result, we also consider versions of $h^{\text{OUCP}}$, $h^{\text{GZOCP}}$ and $h^{\text{SCP}}$ that maximize over multiple cost partitionings. We use the diversification procedure by Seipp, Keller, and Helmert (2017) to obtain a diverse set of cost-partitioned heuristics, denoted by $h^{\text{OUCP}}_{\text{div}}$, $h^{\text{GZOCP}}_{\text{div}}$ and $h^{\text{SCP}}_{\text{div}}$ below.

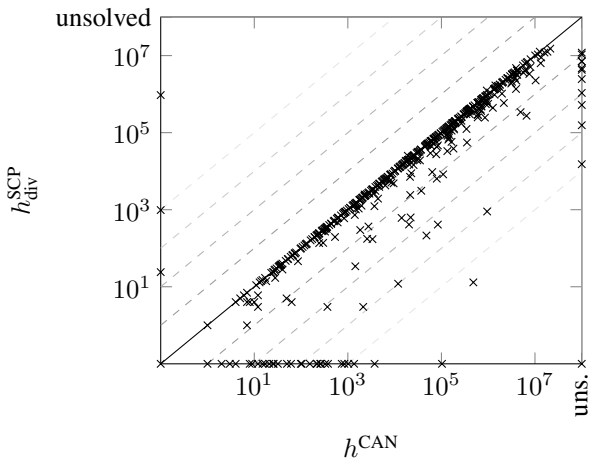


Figure 4: Number of expanded states before the last $f$ layer for the canonical heuristic $h^{\text{CAN}}$ and saturated cost partitioning $h^{\text{SCP}}_{\text{div}}$ using PDBs found by hill climbing.

We start our experimental analysis by computing cost partitionings for two kinds of pattern database (PDB) heuristics: PDBs found by hill climbing (Haslum et al. 2007) and PDBs for systematically generated patterns (Pommerening, Röger, and Helmert 2013).

### Hill Climbing PDBs

The algorithm by Haslum et al. (2007) uses hill climbing in the space of pattern collections and evaluates candidate patterns with the canonical heuristic.

Table 1 shows a domain-wise comparison of the different cost partitioning algorithms. We can see that reusing costs is beneficial when using a single order: $h^{\text{OUCP}}_{\text{one}}$ has an edge over $h^{\text{UCP}}$ in 11 domains, while the opposite is true in only 2 domains and $h^{\text{SCP}}_{\text{one}}$ outperforms $h^{\text{GZOCP}}_{\text{one}}$ 19 to 1. Uniformly distributing costs beats greedy cost assignment when we ignore unused costs: $h^{\text{UCP}}$ beats $h^{\text{GZOCP}}$ 16 to 1. The picture is less clear when we reuse costs: $h^{\text{OUCP}}_{\text{one}}$ solves more tasks than $h^{\text{SCP}}_{\text{one}}$ in 8 domains, while the opposite is true in 12 domains. Contrasting the theoretical dominance, $h^{\text{CAN}}$ solves more tasks than $h^{\text{PHO}}$ in 12 domains, while the opposite case never happens. $h^{\text{OCP}}$ is outperformed by all other cost partitioning algorithms in almost all domains.

All order-dependent cost partitioning algorithms greatly benefit from using more than one order. When using multiple orders, uniformly distributing costs hurts performance in almost all domains: $h^{\text{GZOCP}}_{\text{div}}$ beats $h^{\text{UCP}}$ 14 to 1 and $h^{\text{SCP}}_{\text{div}}$ beats $h^{\text{OUCP}}_{\text{div}}$ 9 to 1. As for single orders, reusing costs boosts

| | $h^{\text{UCP}}$ | $h^{\text{OUCP}}_{\text{one}}$ | $h^{\text{OUCP}}_{\text{div}}$ | $h^{\text{GZOCP}}_{\text{one}}$ | $h^{\text{GZOCP}}_{\text{div}}$ | $h^{\text{SCP}}_{\text{one}}$ | $h^{\text{SCP}}_{\text{div}}$ | $h^{\text{CAN}}$ | $h^{\text{PHO}}$ | $h^{\text{OCP}}$ | coverage | std. dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $h^{\text{UCP}}$ | – | 0 | 3 | 15 | 3 | 4 | 0 | 11 | 10 | 30 | 709.0 | – |
| $h^{\text{OUCP}}_{\text{one}}$ | 14 | – | 9 | 22 | 8 | 6 | 0 | 14 | 13 | 31 | 744.9 | 3.07 |
| $h^{\text{OUCP}}_{\text{div}}$ | 13 | 8 | – | 22 | 7 | 6 | 0 | 14 | 14 | 31 | 734.6 | 2.01 |
| $h^{\text{GZOCP}}_{\text{one}}$ | 3 | 1 | 4 | – | 3 | 0 | 0 | 9 | 11 | 29 | 694.0 | 2.58 |
| $h^{\text{GZOCP}}_{\text{div}}$ | 15 | 12 | 14 | 20 | – | 9 | 0 | 13 | 13 | 30 | 749.9 | 1.66 |
| $h^{\text{SCP}}_{\text{one}}$ | 20 | 19 | 17 | 23 | 16 | – | 0 | 18 | 21 | 32 | 775.7 | 4.47 |
| $h^{\text{SCP}}_{\text{div}}$ | 27 | 26 | 24 | 28 | 22 | 22 | – | 23 | 26 | 33 | **854.9** | 2.33 |
| $h^{\text{CAN}}$ | 8 | 7 | 7 | 17 | 5 | 8 | 2 | – | 13 | 28 | 656.0 | – |
| $h^{\text{PHO}}$ | 9 | 7 | 7 | 15 | 7 | 6 | 3 | 10 | – | 31 | 737.0 | – |
| $h^{\text{OCP}}$ | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 5 | 3 | – | 471.0 | – |

Table 2: Results for systematic PDBs. For an explanation of the data, see the caption of Table 1.

performance: $h^{\text{OUCP}}_{\text{div}}$ scores 11 to 1 against $h^{\text{UCP}}$ and $h^{\text{SCP}}_{\text{div}}$ wins against $h^{\text{GZOCP}}_{\text{div}}$ by 7 to 1.

Saturated cost partitioning emerges as the winner of this comparison. No other cost partitioning has an edge over $h^{\text{SCP}}_{\text{div}}$ in more than two domains and $h^{\text{CAN}}$ and $h^{\text{PHO}}$ never dominate $h^{\text{SCP}}_{\text{div}}$. $h^{\text{SCP}}_{\text{div}}$ solves the highest total number of tasks on average (830.6). To understand why $h^{\text{SCP}}_{\text{div}}$ has an edge over its competitors in so many domains we compare it to $h^{\text{CAN}}$, which solves 813 tasks. Figure 4 shows the number of expansions made by the two heuristics. As we can see, $h^{\text{CAN}}$ needs fewer expansions than $h^{\text{SCP}}_{\text{div}}$ for only 3 tasks. For the majority of tasks, $h^{\text{SCP}}_{\text{div}}$ is more accurate, often reducing the number of expanded states by several orders of magnitude.

## Systematic PDBs

Our second analysis uses a procedure that generates all *interesting* patterns up to a given size (Pommerening, Röger, and Helmert 2013). Since generating the PDBs for all patterns of size 3 takes too long for many tasks, we generate all patterns of sizes 1 and 2. We compare the different cost partitioning algorithms for systematic PDBs in Table 2. Again, $h^{\text{OCP}}$ usually has the lowest coverage scores in all domains, though the numbers look a little bit better than for hill climbing PDBs. This is probably due to the systematic PDBs being smaller than the hill climbing PDBs.

The results for non-optimal cost partitioning algorithms show similar trends as in the setting evaluating hill climbing PDBs. Reusing costs boosts performance regardless of whether we assign costs uniformly or greedily and whether we use a single or multiple orders. Uniform cost partitioning is only preferable to greedily assigning costs if we use a single order and do not reuse costs. $h^{\text{CAN}}$ and $h^{\text{PHO}}$ outperform each other on 13 and 10 domains, respectively.

Going from one to multiple orders only has a small effect for $h^{\text{OUCP}}$ (9 vs. 8), but $h^{\text{GZOCP}}$ and $h^{\text{SCP}}$ greatly benefit from this change (20 vs. 3 and 22 vs. 0).

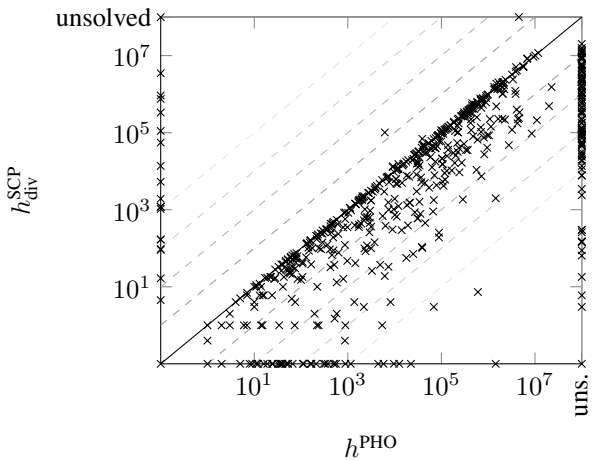


Figure 5: Number of expanded states before the last $f$ layer for post-hoc optimization $h^{\text{PHO}}$ and saturated cost partitioning $h^{\text{SCP}}_{\text{div}}$ using systematic PDBs.

| | $h^{\text{UCP}}$ | $h^{\text{OUCP}}_{\text{one}}$ | $h^{\text{OUCP}}_{\text{div}}$ | $h^{\text{GZOCP}}_{\text{one}}$ | $h^{\text{GZOCP}}_{\text{div}}$ | $h^{\text{SCP}}_{\text{one}}$ | $h^{\text{SCP}}_{\text{div}}$ | $h^{\text{OCP}}$ | coverage | std. dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| $h^{\text{UCP}}$ | – | 0 | 4 | 9 | 1 | 2 | 1 | 31 | 662.0 | – |
| $h^{\text{OUCP}}_{\text{one}}$ | 14 | – | 7 | 18 | 2 | 5 | 1 | 33 | 695.6 | 2.30 |
| $h^{\text{OUCP}}_{\text{div}}$ | 16 | 12 | – | 19 | 5 | 7 | 1 | 32 | 713.1 | 1.20 |
| $h^{\text{GZOCP}}_{\text{one}}$ | 3 | 0 | 4 | – | 0 | 0 | 0 | 31 | 648.2 | 3.49 |
| $h^{\text{GZOCP}}_{\text{div}}$ | 19 | 16 | 12 | 22 | – | 10 | 0 | 33 | 734.0 | 0.67 |
| $h^{\text{SCP}}_{\text{one}}$ | 22 | 18 | 18 | 25 | 13 | – | 0 | 34 | 763.2 | 9.60 |
| $h^{\text{SCP}}_{\text{div}}$ | 25 | 25 | 24 | 28 | 23 | 26 | – | 36 | **967.4** | 0.70 |
| $h^{\text{OCP}}$ | 4 | 3 | 3 | 4 | 3 | 3 | 1 | – | 393.0 | – |

Table 3: Results for Cartesian abstractions. For an explanation of the data, see the caption of Table 1.

$h^{\text{SCP}}_{\text{div}}$ is again the method of choice for almost all domains. It is only bested by $h^{\text{CAN}}$, $h^{\text{PHO}}$ and $h^{\text{OCP}}$, and never in more than 3 domains. $h^{\text{SCP}}_{\text{div}}$ also has the highest overall coverage. Figure 5 compares the number of states expanded by $h^{\text{SCP}}_{\text{div}}$ and $h^{\text{PHO}}$ and reveals that $h^{\text{SCP}}_{\text{div}}$ has a similar, if not slightly higher, advantage as in the corresponding experiment with hill climbing PDBs.

## Cartesian Abstractions

Next, we consider Cartesian abstractions (Seipp and Helmert 2013) of the *landmark* and *goal* task decompositions (Seipp and Helmert 2014). Table 3 holds the domain-wise and total coverage numbers for $h^{\text{UCP}}$, $h^{\text{OUCP}}$, $h^{\text{GZOCP}}$, $h^{\text{SCP}}$ and $h^{\text{OCP}}$ on Cartesian abstractions. We can see that reusing costs significantly improves heuristic estimates: $h^{\text{OUCP}}_{\text{one}}$ solves more tasks than $h^{\text{UCP}}$ in 14 domains, while

| | $h^{\text{UCP}}$ | $h^{\text{OUCP}}_{\text{one}}$ | $h^{\text{GZOCP}}_{\text{one}}$ | $h^{\text{SCP}}_{\text{one}}$ | $h^{\text{OCP}}$ | coverage | std. dev. |
|---|---|---|---|---|---|---|---|
| $h^{\text{UCP}}$ | – | 1 | 7 | 4 | 26 | 877.0 | – |
| $h^{\text{OUCP}}_{\text{one}}$ | 1 | – | 7 | 3 | 27 | 876.2 | 0.63 |
| $h^{\text{GZOCP}}_{\text{one}}$ | 0 | 0 | – | 0 | 24 | 857.0 | 0.00 |
| $h^{\text{SCP}}_{\text{one}}$ | 7 | 8 | 11 | – | 29 | **888.1** | 0.32 |
| $h^{\text{OCP}}$ | 1 | 1 | 3 | 1 | – | 816.0 | – |

Table 4: Results for landmark heuristics. For an explanation of the data, see the caption of Table 1.

the opposite is never the case. Similarly, $h^{\text{SCP}}_{\text{one}}$ has a higher coverage than $h^{\text{GZOCP}}_{\text{one}}$ in 25 domains and again the opposite never occurs.

Again, the results show that assigning costs uniformly is only beneficial when using a single order and when not reusing unused costs. As above, $h^{\text{OCP}}$ is almost always outperformed by all other cost partitioning algorithms and $h^{\text{SCP}}_{\text{div}}$ almost always has the highest coverage in each domain. $h^{\text{SCP}}_{\text{div}}$ also solves significantly more tasks in total than all other compared cost partitioning algorithms (967.4 tasks).

## Landmark Heuristics

Our last comparison of cost partitioning algorithms considers landmark heuristics. To the best of our knowledge only two ways of combining landmark heuristics admissibly have been previously evaluated: optimal and uniform cost partitioning (Karpas and Domshlak 2009). We compare these two algorithms to opportunistic uniform cost partitioning, greedy zero-one cost partitioning and saturated cost partitioning. In contrast to the experiments above, we have to compute a cost partitioning for landmark heuristics in every evaluated state, instead of only once before the search starts. We therefore restrict our analysis to single order cost partitionings.

Table 4 compares the different cost partitioning algorithms for the BJOLP landmark heuristic (Domshlak et al. 2011). In difference to the results above, $h^{\text{OCP}}$ comes closer to the other cost partitioning algorithms. This is the case since the linear programs that have to be solved for optimal landmark cost partitioning are much smaller than the ones for general abstractions. Comparing the suboptimal cost partitioning algorithms, we see that reusing costs has no significant effect for uniform cost partitioning: $h^{\text{OUCP}}_{\text{one}}$ and $h^{\text{UCP}}$ outperform each other on one domain, respectively. For greedy cost assignments, reusing costs is very important: $h^{\text{SCP}}_{\text{one}}$ beats $h^{\text{GZOCP}}_{\text{one}}$ 11 to 0. Uniform cost partitioning again has an edge over greedy cost assignment if costs are not reused: $h^{\text{UCP}}$ beats $h^{\text{GZOCP}}_{\text{one}}$ 7 to 0. However, the picture is inverted when reusing costs: $h^{\text{SCP}}_{\text{one}}$ beats $h^{\text{OUCP}}_{\text{one}}$ 8 to 3.

## Comparison of Different Heuristics

In the experiments above, we compared different cost partitioning algorithms operating on the same set of heuristics. In all settings $h^{\text{SCP}}_{\text{div}}$ (respectively $h^{\text{SCP}}_{\text{one}}$ for landmark heuristics) had an edge over its competition. Comparing the four $h^{\text{SCP}}$-based planners allows us to shed some light on the relative accuracy of the underlying heuristics. Also, we are interested in how well these cost-partitioned heuristics fare against other planners. Table 5 compares the $h^{\text{SCP}}$-based planners to three heuristics from the literature: $h^{\text{LM-cut}}$ (Helmert and Domshlak 2009), merge-and-shrink using bisimulation and the DFP merge strategy ($h^{\text{M\&S}}$) (Helmert et al. 2014; Sievers, Wehrle, and Helmert 2014), and the state-equation heuristic ($h^{\text{SEQ}}$) (Bonet 2013).

Inspecting the results for the four $h^{\text{SCP}}$-based planners, we see that LM+$h^{\text{SCP}}_{\text{one}}$ usually solves fewer tasks per domain than the other three planners, even though it has the second highest total coverage. The ranking between HC+$h^{\text{SCP}}_{\text{div}}$, Sys2+$h^{\text{SCP}}_{\text{div}}$ and Cart.+$h^{\text{SCP}}_{\text{div}}$ is less clear, as each planner outperforms the others on roughly the same number of domains.

$h^{\text{M\&S}}$ and $h^{\text{SEQ}}$ are outperformed by the competition.

| | HC+$h^{\text{SCP}}_{\text{div}}$ | Sys2+$h^{\text{SCP}}_{\text{div}}$ | Cart.+$h^{\text{SCP}}_{\text{div}}$ | LM+$h^{\text{SCP}}_{\text{one}}$ | $h^{\text{LM-cut}}$ | $h^{\text{M\&S}}$ | $h^{\text{SEQ}}$ | coverage |
|---|---|---|---|---|---|---|---|---|
| HC+$h^{\text{SCP}}_{\text{div}}$ | – | 7 | 13 | 21 | 17 | 23 | 24 | 830.6 |
| Sys2+$h^{\text{SCP}}_{\text{div}}$ | 11 | – | 14 | 19 | 18 | 23 | 24 | 854.9 |
| Cart.+$h^{\text{SCP}}_{\text{div}}$ | 17 | 13 | – | 22 | 20 | 25 | 27 | **967.4** |
| LM+$h^{\text{SCP}}_{\text{one}}$ | 7 | 9 | 7 | – | 8 | 18 | 21 | 888.1 |
| $h^{\text{LM-cut}}$ | 14 | 12 | 10 | 14 | – | 23 | 24 | 882.0 |
| $h^{\text{M\&S}}$ | 7 | 7 | 6 | 14 | 9 | – | 19 | 743.0 |
| $h^{\text{SEQ}}$ | 7 | 7 | 7 | 8 | 6 | 13 | – | 734.0 |

Table 5: Results for different heuristics. For an explanation of the data, see the caption of Table 1.

| | HC+$h^{\text{SCP}}_{\text{div}}$ | Sys2+$h^{\text{SCP}}_{\text{div}}$ | Cart.+$h^{\text{SCP}}_{\text{div}}$ | LM+$h^{\text{SCP}}_{\text{one}}$ | $h^{\text{LM-cut}}$ | $h^{\text{M\&S}}$ | $h^{\text{SEQ}}$ | SymBA$^*_2$ | coverage |
|---|---|---|---|---|---|---|---|---|---|
| HC+$h^{\text{SCP}}_{\text{div}}$ | – | 7 | 9 | 19 | 15 | 21 | 25 | 17 | 845.0 |
| Sys2+$h^{\text{SCP}}_{\text{div}}$ | 10 | – | 11 | 18 | 18 | 23 | 24 | 16 | 878.5 |
| Cart.+$h^{\text{SCP}}_{\text{div}}$ | 19 | 14 | – | 24 | 19 | 24 | 28 | 17 | **1017.9** |
| LM+$h^{\text{SCP}}_{\text{one}}$ | 8 | 9 | 4 | – | 9 | 13 | 23 | 9 | 934.0 |
| $h^{\text{LM-cut}}$ | 15 | 11 | 8 | 14 | – | 20 | 23 | 10 | 927.0 |
| $h^{\text{M\&S}}$ | 8 | 7 | 5 | 14 | 10 | – | 20 | 6 | 797.0 |
| $h^{\text{SEQ}}$ | 5 | 5 | 6 | 6 | 8 | 12 | – | 7 | 779.0 |
| SymBA$^*_2$ | 20 | 18 | 16 | 23 | 20 | 23 | 27 | – | 1008.0 |

Table 6: Results for different heuristics using $h^2$ mutexes to prune irrelevant operators. For an explanation of the data, see the caption of Table 1.

Only $h^{\text{LM-cut}}$ is able to beat one of the planners using saturated cost partitioning, beating LM+$h^{\text{SCP}}_{\text{one}}$ 14 to 8. However, $h^{\text{LM-cut}}$ comes up short in the comparisons to HC+$h^{\text{SCP}}_{\text{div}}$ (17 to 14), Sys2+$h^{\text{SCP}}_{\text{div}}$ (18 to 12) and Cart.+$h^{\text{SCP}}_{\text{div}}$ (20 to 10).

To evaluate how close our cost partitioning algorithms are to the state of the art, we compare the heuristics from the last experiment to the winner of the IPC 2014 sequential optimization track, the symbolic search planner SymBA$^*_2$ (Torralba, Linares López, and Borrajo 2016). SymBA$^*_2$ preprocesses planning tasks by using $h^2$ mutexes to prune irrelevant operators (Alcázar and Torralba 2015). To allow for an unbiased comparison, we evaluate all algorithms from the previous experiment with this preprocessing step. We compare them to a version of SymBA$^*_2$ that is identical to the IPC version apart from some bug-fixes. Table 6 shows that the only algorithm on par with SymBA$^*_2$ is Cart.+$h^{\text{SCP}}_{\text{div}}$, scoring 17 to 16 against SymBA$^*_2$. In terms of total coverage, Cart.+$h^{\text{SCP}}_{\text{div}}$ has a very slight edge over SymBA$^*_2$ (1017.9 vs. 1008 tasks).

## Conclusion

We presented the first systematic theoretical and experimental comparison of cost partitioning algorithms for optimal classical planning. Our theoretical analysis shows that saturated cost partitioning dominates zero-one cost partitioning and suggested a new cost partitioning algorithm called opportunistic uniform cost partitioning, which dominates uniform cost partitioning.

Our experimental evaluation revealed that uniform cost partitioning is only preferable to assigning costs greedily if a single order is used and costs are not reused. In all other cases it is beneficial to reuse unused costs, to assign them greedily and to use multiple orders. We also showed that saturated cost partitioning is the method of choice in all tested settings, outperforming the previous best cost partitioning methods for all tested heuristics. Except for hill climbing PDBs, saturated cost partitioning even has the highest total coverage when using only a single order. The resulting heuristics are competitive with and often outperform the state of the art in optimal classical planning.

In future work, we would like to use saturated cost partitioning to combine non-abstraction-based heuristics.

## Acknowledgments

We thank Álvaro Torralba for providing us with the fixed SymBA$^*_2$ version.

This work was supported by the Swiss National Science Foundation (SNSF) as part of the project "Reasoning about Plans and Heuristics for Planning and Combinatorial Search" (RAPAHPACS).

## References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2–6. AAAI Press.

Ball, T.; Podelski, A.; and Rajamani, S. K. 2001. Boolean and Cartesian abstraction for model checking C programs. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001)*, 268–283.

Bonet, B. 2013. An admissible heuristic for SAS$^+$ planning obtained from the state equation. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2268–2274.

Domshlak, C.; Helmert, M.; Karpas, E.; and Markovitch, S. 2011. The SelMax planner: Online learning for speeding up optimal planning. In *IPC 2011 planner abstracts*, 108–112.

Edelkamp, S. 2006. Automated creation of pattern database search heuristics. In *Proceedings of the 4th Workshop on Model Checking and Artificial Intelligence (MoChArt 2006)*, 35–50.

Felner, A.; Korf, R.; and Hanan, S. 2004. Additive pattern database heuristics. *Journal of Artificial Intelligence Research* 22:279–318.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*, 1007–1012. AAAI Press.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, 1163–1168. AAAI Press.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM* 61(3):16:1–63.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Holte, R.; Felner, A.; Newton, J.; Meshulam, R.; and Furcy, D. 2006. Maximizing over multiple pattern databases speeds up heuristic search. *Artificial Intelligence* 170(16–17):1123–1136.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In Boutilier, C., ed., *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 1728–1733.

Karpas, E.; Katz, M.; and Markovitch, S. 2011. When optimal is just not good enough: Learning fast informative action cost partitionings. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 122–129. AAAI Press.

Katz, M., and Domshlak, C. 2008. Optimal additive composition of abstraction-based admissible heuristics. In Rin-

tanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E., eds., *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*, 174–181. AAAI Press.

Katz, M., and Domshlak, C. 2010. Optimal admissible composition of abstraction heuristics. *Artificial Intelligence* 174(12–13):767–798.

Keller, T.; Pommerening, F.; Seipp, J.; Geißer, F.; and Mattmüller, R. 2016. State-dependent cost partitionings for cartesian abstractions in classical planning. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3161–3169.

Korf, R. E., and Felner, A. 2002. Disjoint pattern database heuristics. *Artificial Intelligence* 134(1–2):9–22.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From non-negative to general operator cost partitioning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3335–3341. AAAI Press.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the most out of pattern databases for classical planning. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2357–2364.

Seipp, J., and Helmert, M. 2013. Counterexample-guided Cartesian abstraction refinement. In Borrajo, D.; Kambhampati, S.; Oddi, A.; and Fratini, S., eds., *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling (ICAPS 2013)*, 347–351. AAAI Press.

Seipp, J., and Helmert, M. 2014. Diverse and additive Cartesian abstraction heuristics. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, 289–297. AAAI Press.

Seipp, J.; Keller, T.; and Helmert, M. 2017. Narrowing the gap between saturated and optimal cost partitioning for classical planning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3651–3657. AAAI Press.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized label reduction for merge-and-shrink heuristics. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.

Torralba, Á.; Linares López, C.; and Borrajo, D. 2016. Abstraction heuristics for symbolic bidirectional search. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3272–3278.

Yang, F.; Culberson, J.; Holte, R.; Zahavi, U.; and Felner, A. 2008. A general theory of additive state space abstractions. *Journal of Artificial Intelligence Research* 32:631–662.