State-dependent Cost Partitionings for Cartesian Abstractions in Classical Planning

Thomas Keller and **Florian Pommerening** and **Jendrik Seipp**
University of Basel, Switzerland
{tho.keller, florian.pommerening, jendrik.seipp}@unibas.ch

Florian Geißer and **Robert Mattmüller**
University of Freiburg, Germany
{geisserf, mattmuel}@informatik.uni-freiburg.de

Abstract

Abstraction heuristics are a popular method to guide optimal search algorithms in classical planning. *Cost partitionings* allow to sum heuristic estimates admissibly by distributing action costs among the heuristics. We introduce *state-dependent* cost partitionings which take context information of actions into account, and show that an *optimal state-dependent cost partitioning* dominates its state-independent counterpart. We demonstrate the potential of our idea with a state-dependent variant of the recently proposed *saturated cost partitioning*, and show that it has the potential to improve not only over its state-independent counterpart, but even over the optimal state-independent cost partitioning. Our empirical results give evidence that ignoring the context of actions in the computation of a cost partitioning leads to a significant loss of information.

1 Introduction

Abstraction heuristics based on pattern databases [Culberson and Schaeffer, 1998; Edelkamp, 2001] or the more general *Cartesian abstractions* [Seipp and Helmert, 2013] are popular methods to guide optimal heuristic search algorithms in classical planning. Since a single abstraction often provides poor guidance in challenging problems [Helmert and Mattmüller, 2008], we would like to combine the information from *several* abstractions admissibly. The simplest approach maximizes over a set of admissible heuristics. However, even though this often improves over the heuristic of a single abstraction, it does not really combine information of different sources but rather just selects the most accurate one.

Additive pattern databases [Korf and Felner, 2002; Felner *et al.*, 2004] are one technique for combining information from a set of abstractions. They are based on the idea that heuristic estimates can be added up and still form an admissible heuristic as long as the used abstractions are pairwise independent. The related canonical heuristic [Haslum *et al.*, 2007] combines both ideas: for a given set of abstractions, it computes all maximal subsets of pairwise independent abstractions and the sum of their heuristic values, and uses the maximum of these values to estimate the goal distance from a given state.

Cost partitioning [Katz and Domshlak, 2007; 2010] generalizes additive heuristics by replacing the requirement that the cost of each action may only be considered in a single abstraction with the requirement that the accumulated cost of each action over all abstractions may not exceed the original cost. Karpas and Domshlak [2009], for instance, distribute action costs among sub-tasks derived from a set of landmarks. In the context of Cartesian abstractions, Seipp and Helmert [2014] introduced *saturated cost partitioning*, where a cost partitioning is computed iteratively by "consuming" the minimum costs in each abstraction such that the costs of all shortest paths are preserved. Finally, *general cost partitioning* [Pommerening *et al.*, 2015] extends the formalism to allow negative costs, which leads to admissible estimates that can be higher than an *optimal cost partitioning* restricted to non-negative costs.

In this paper, we show that even more information can be extracted from an abstraction collection if *context information* is used. Accounting for complete context information corresponds to computing a cost partitioning over all transitions – or, equivalently, over all state-action pairs – rather than only over all actions. We define the concept of state-dependent cost partitionings and show that an optimal one dominates the optimal state-independent cost partitioning.

As the saturated state-independent cost partitioning maximizes over all transitions incurred by the same action, it loses valuable information. We show that the *saturated state-dependent cost partitioning*, where costs are consumed only in a given context, does not suffer from this loss of information. Our implementation follows the idea of Geißer *et al.* [2016] who use edge-valued multi-valued decision diagrams (EVMDDs) [Ciardo and Siminiceanu, 2002] to encode context information efficiently. Even though saturated state-dependent cost partitioning does not dominate optimal nor saturated state-independent cost partitioning, it has the potential to produce better heuristic estimates than both. A preliminary experimental evaluation shows that the concept is relevant in practice and able to improve heuristic estimates in many benchmark instances.

2 Background

Planning Tasks. We consider SAS$^+$ planning tasks [Bäckström and Nebel, 1995] with state-independent action costs, where a *planning task* is given as a tuple $\Pi = (\mathcal{V}, A, s_I, s_\star, c)$ that consists of the following components: $\mathcal{V}$ is a finite set of *state variables* v, each with an associated finite domain $\mathcal{D}_v$. A *fact* is a pair (v, d), where $v \in \mathcal{V}$ and $d \in \mathcal{D}_v$, and a *partial variable assignment* s over $\mathcal{V}$ is a set of facts that belong to different variables. If s assigns a value to each $v \in \mathcal{V}$, s is called a *state*. We use function notation $s(v) = d$ and set notation $(v, d) \in s$ interchangeably and denote the set of states of Π with S. Each action $a = \langle \mathit{pre}, \mathit{eff} \rangle$ in the set of *actions* A is a pair of partial variable assignments, called *precondition* and *effect*. The state $s_I \in S$ is called the *initial state*, and the partial variable assignment $s_\star$ specifies the *goal* condition. A state s is a goal state if $s_\star \subseteq s$. We denote the set of goal states by $S_\star$.

An action a is applicable in state s iff $\mathit{pre} \subseteq s$. Applying a in s yields the state s' with $s'(v) = \mathit{eff}(v)$ where $\mathit{eff}(v)$ is defined and $s'(v) = s(v)$ otherwise. We write $s[a]$ for s'. The (non-negative) *cost* of applying a is given by the cost function $c : A \to \mathbb{R}_0^+$ of Π as $c(a)$. However, at several places in this paper, we are interested in costs that are based on altered cost functions. An important aspect of this work are *general* and *state-dependent* cost functions $c : A \times S \to \mathbb{R}$ that determine transition costs $c(a, s)$ that depend on the state s in addition to the action a that is applied. Since state-dependent cost functions are more general, we define the following concepts in terms of state-dependent instead of regular cost functions unless we want to emphasize that the cost function of the original task is used. The resulting framework is similar to the formalism introduced by Geißer *et al.* [2015] for SAS$^+$planning tasks with state-dependent action costs.

Let $\pi = \langle a_1, \ldots, a_n \rangle$ be a sequence of actions from A. We call π *applicable* in s if there exist states $s_0, s_1, \ldots, s_n$ such that $s_0 = s$, a_i is applicable in s_{i-1} and $s_i = s_{i-1}[a_i]$ for all $i = 1, \ldots, n$ and write $s[\pi]$ for s_n. We call π an *s-plan* for Π if it is applicable in s and if $s[\pi] \in S_\star$. The *cost* of an s-plan π under cost function c is the sum of action costs along the induced state sequence, i.e., $c(\pi, s) = \sum_{i=1}^{n} c(a_i, s_{i-1})$. An *optimal s-plan* under c is an s-plan that minimizes $c(\pi, s)$. Its cost is denoted by $h^\star(s, c)$. If there is no s-plan then $h^\star(s, c) = \infty$. A *heuristic function* h estimates the cost of an optimal s-plan under cost function c with values $h(s, c) \in \mathbb{R} \cup \{-\infty, \infty\}$. Note that we allow negative heuristic values to support general cost partitioning [Pommerening *et al.*, 2015]. A heuristic h is called *admissible* if it never overestimates the true cost, i.e., $h(s, c) \leq h^\star(s, c)$ for all states $s \in S$.

A planning task Π and a cost function c induce a transition system $\mathcal{T} = (S, L, T, s_I, S_\star)$ with state space S, transition labels L, transition relation T, initial state s_I and goal states $S_\star$ as usual, except that besides source state s, target state t and transition label a, transitions in T also carry a weight $w \in \mathbb{R}$, which is determined by s and a as $w = c(s, a)$. For transitions we also write $s \xrightarrow{a,w} t$. It follows that a plan for s is a path from s to some $s_n \in S_\star$ and the plan is optimal if the sum of the weights along the path is minimal.

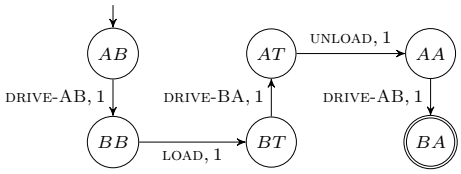


Figure 1: Transition system of our running example. States are of the form xy, where x is the location of the truck and y the location of the package.

Cartesian Abstractions. The core idea of abstraction heuristics is to collapse several states into a single abstract state, which reduces the size of the transition system and allows the computation of goal distances that can be used as admissible heuristic estimates. Given a planning task Π with induced transition system $\mathcal{T} = (S, L, T, s_I, S_\star)$, let α be an equivalence relation on S that describes equivalence classes of the form $D_1 \times D_2 \times \cdots \times D_n$, where $D_i \subseteq \mathcal{D}_{v_i}$ for $\mathcal{V} = \{v_1, \ldots, v_n\}$. The resulting abstract states are often called Cartesian abstract states, but since we only consider Cartesian abstractions, we call them *abstract states* in the following. $\mathcal{T}$ and α induce an abstract transition system $\mathcal{T}^\alpha = (S^\alpha, L, T^\alpha, \alpha(s_I), S_\star^\alpha)$ in the following way: each concrete state s is mapped to the abstract state $\alpha(s) \in S^\alpha$; the initial state s_I is mapped to the abstract initial state $\alpha(s_I)$ and goal states are mapped to abstract goal states $S_\star^\alpha = \{\alpha(s_\star) \mid s_\star \in S_\star\}$; there is a labelled abstract transition between $\alpha(s)$ and $\alpha(s')$ whenever there is a concrete transition with the same label between s and s'.

As long as action costs are state-independent, the weight of an abstract transition is simply the cost of the concrete action that induces it. In the presence of state-dependent action costs (which are not present initially in our input tasks, but can be introduced by state-dependent cost partitionings), determining the weight of an abstract transition is not as straightforward. Geißer *et al.* [2016] define the weight of an abstract transition between abstract states t and u with transition label a to be the *minimal* weight of all concrete transitions labeled with action a that start in a state s with $\alpha(s) = t$. Together with the fact that every plan in the concrete transition system is a plan in the abstract transition system, this definition ensures that the cost of each abstract plan may be used as an admissible heuristic estimate. For now, the definition of Geißer *et al.* is sufficient for our needs, but we present an enhanced method in Section 6 that estimates abstract transition weights more precisely, while still guaranteeing admissibility.

3 Running Example

Throughout the paper, we use a simple logistics task as a running example, where a truck that starts in location A has to fetch a package from location B and bring it to A. The truck must return to B after the delivery. The corresponding transition system is given in Figure 1.

The task is described by two variables $\mathcal{V} = \{T, P\}$ that encode the position of the truck and the package, respectively. The truck can be in each of the two locations ($\mathcal{D}_T = \{A, B\}$) and the package can be in either location or in the truck

($\mathcal{D}_P = \{A, B, T\}$). The truck can drive without load from A to B (DRIVE-AB) and with load from B to A (DRIVE-BA). There are actions to LOAD the package into the truck in B or UNLOAD it from the truck in A. All actions have unit cost.

Note that we explicitly exclude all other actions such as driving from B to A without the package or unloading the package at B. While this makes the task trivially solvable, the task is still sufficient to show some interesting properties.

Figure 2 shows three pairs of abstractions for our example task which are used in the remainder of this paper. In all examples, we assume that the applied heuristic is the goal distance function in each abstraction, denoted h_1 and h_2, respectively. Additionally, since all concrete states have exactly one applicable action, we also refer to state-action pairs (a, s) as $s^{\rightarrow}$. Given cost function c, we write $c(s^{\rightarrow})$ for $c(a, s)$.

4 State-dependent Cost Partitioning

Early work on additive admissible heuristics has mostly focused on techniques that allow to generate or identify heuristics that can be added up admissibly because each deals with a sub-problem of the planning task that can be regarded independently from the rest [Felner *et al.*, 2004; Haslum *et al.*, 2007]. An equivalent view on these techniques is to regard them as cost partitionings [Katz and Domshlak, 2007; 2010] that distribute action costs such that each operator is assigned its full cost in one heuristic and a cost of zero in all other. However, cost partitionings are more general as costs can be distributed arbitrarily between the heuristics as long as the sum over the individual costs does not exceed the original cost. Given such a cost partitioning, heuristic values are then computed on a copy of the planning task where actions cost only the fraction of the actual action cost that is assigned to the heuristic. Recently, Pommerening *et al.* [2015] showed that the framework can be extended from non-negative to general cost partitioning, which often allows to derive more accurate heuristic functions.

In this paper, we continue the process of developing more accurate cost partitioning techniques by presenting state-dependent cost partitionings, a generalization where context information of applied actions is additionally taken into account.

Definition 1. *Let Π be a planning task. A (general)* state-dependent cost partitioning *for Π is a tuple $P = \langle c_1 \dots, c_n \rangle$, where $c_i : A \times S \to \mathbb{R}$ for $1 \leq i \leq n$ and $\sum_{i=1}^{n} c_i(a, s) \leq c(a)$ for all $s \in S$ and $a \in A$. If P is* state-independent, *i.e., such that $c_i(a, s) = c_i(a, s')$ for all $s, s' \in S$, $a \in A$ and $1 \leq i \leq n$, we call P a general* state-independent cost partitioning *for Π.*

The introduction of state-dependent cost functions does not change the fact that admissible additive heuristic can be derived.

Theorem 1. *Let $h_1, \dots, h_n$ be admissible heuristics for a planning task Π and $P = \langle c_1 \dots, c_n \rangle$ be a state-dependent cost partitioning for Π. Then $h_P(s) = \sum_{i=1}^{n} h_i(s, c_i)$ is an admissible heuristic. If any term in the sum is ∞, the sum is defined as ∞, even if another term is $-\infty$.*

Proof sketch: The proof is a straightforward extension of the corresponding proof for state-independent cost partitionings (Theorem 1 in the work of Pommerening *et al.*, 2015). It can be found in the technical report [Keller *et al.*, 2016]. ∎

State-dependent cost partitionings differ from their state-independent counterpart in the fact that each state-action pair can have its own cost instead of a cost that is shared among all possible applications of an action. If abstraction heuristics are considered, this corresponds to transition systems where all transitions can have arbitrary weights on the one hand and transition systems where all transitions with the same label share the same weight on the other.

Definition 2. *Let $h_1, \dots, h_n$ be admissible heuristics for a planning task Π, $\mathbb{P}_D$ the space of possible state-dependent cost partitionings and $\mathbb{P}_I \subseteq \mathbb{P}_D$ the space of possible state-independent cost partitionings for Π. The* optimal state-dependent cost partitioning ($\mathrm{OCP_D}$) *heuristic estimate for $h_1, \dots, h_n$ in state s is $h^{ocp_D}(s) = \max_{P \in \mathbb{P}_D} h_P(s)$, and the* optimal state-independent cost partitioning ($\mathrm{OCP_I}$) *heuristic estimate for $h_1, \dots, h_n$ is $h^{ocp_I}(s) = \max_{P \in \mathbb{P}_I} h_P(s)$.*

State-dependent cost partitionings allow the computation of more accurate heuristics estimates, which is the most important theoretical contribution of this paper.

Theorem 2. *Let $h_1, \dots, h_n$ be admissible heuristics for a planning task Π. For all $s \in S$ it holds that $h^{ocp_D}(s) \geq h^{ocp_I}(s)$. Moreover, there are planning tasks where the inequality is strict for at least one state.*

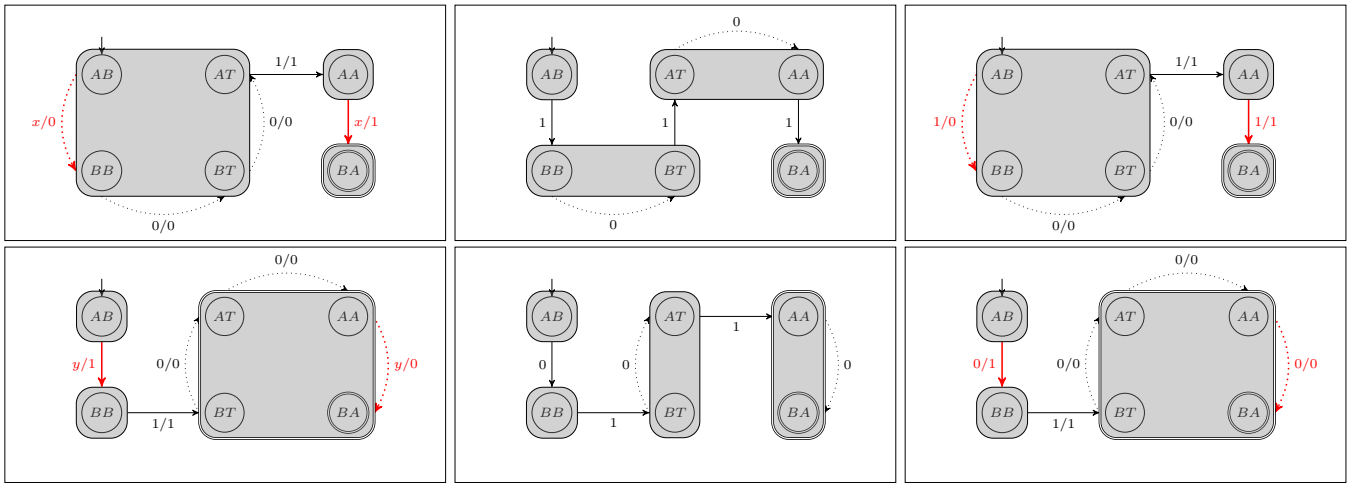
Proof: The statement that $h^{ocp_D}(s) \geq h^{ocp_I}(s)$ for all $s \in S$ holds as every state-independent cost partitioning is also a state-dependent cost partitioning. The second statement follows from the following example. ∎

Example 1. *Consider the two abstract transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$ in Figure 2a. The first edge label denotes the cost assigned by $\mathrm{OCP_I}$, and the second the cost assigned by $\mathrm{OCP_D}$. If an action induces only self-loops (depicted by dotted arcs) in one abstraction and must be part of a shortest path (depicted by solid arcs) in the other such as $AT^{\rightarrow}$ and $BB^{\rightarrow}$, all optimal cost partitionings use their full cost in the abstraction where the transition is part of the path. The difference between $\mathrm{OCP_I}$ and $\mathrm{OCP_D}$ is the distribution of the cost of action* DRIVE-AB*, which induces the two transitions $AB^{\rightarrow}$ and $AA^{\rightarrow}$ in our example task.*

State-independent cost partitionings must assign the same value to both transitions within each abstraction. Let this value be x for the first and y for the second abstraction in our example. The heuristic value of the initial state under an optimal state-independent cost partitioning is then the maximum of $(1 + x) + (y + 1)$ subject to $x + y \leq 1$, so $h^{ocp_I}(AB) = 3$.

State-dependent cost partitionings, on the other hand, allow that two transitions that are induced by the same action are assigned different costs within the same abstraction – as long as the sum of assigned costs for each transition does not exceed that transitions original cost. In our example, a possible optimal state-dependent cost partitioning is to assign the full cost to $AA^{\rightarrow}$ in the first and to $AB^{\rightarrow}$ in the second abstraction, which combines to a heuristic value of $h^{ocp_D}(AB) = 2 + 2 = 4$

Even though Theorem 2 provides an encouraging result, its practical impact is limited without further work. This



(a) Example 1 (b) Example 2 (c) Example 3

Figure 2: Abstractions for our running example. Circles depict concrete states, abstract states are rectangular, solid arcs depict the shortest path from the abstract initial state to an abstract goal state and dotted arcs are abstract self-loops. We denote the transition system at the top (bottom) of each column with $\mathcal{T}_1$ ($\mathcal{T}_2$). Edge labels denote costs of different cost partitionings.

is mostly because the computation of an optimal state-dependent cost partitioning with a method that is used to compute a state-independent cost partitioning [Katz and Domshlak, 2010; Bonet and van den Briel, 2014; Pommerening *et al.*, 2015] would require a compilation with one action for each state-action pair, a number that is linear in the number of states and hence exponential in the number of variables. Even though there are techniques like context splitting [Röger *et al.*, 2014] that allow to compute a more compact compilation, the worst-case exponential blowup cannot be avoided in general. We therefore turn our attention to saturated cost partitioning [Seipp and Helmert, 2014], a cost partitioning that is tractable in practice.

5 Saturated Cost Partitioning

Seipp and Helmert [2014] introduced the concept of *cost saturation*. Iteratively, they compute an abstraction, reduce the action costs such that all goal distances are preserved, and use the remaining costs for subsequent abstractions. The result is known as a *saturated cost partitioning*. Due to the greedy nature of the procedure, the resulting cost partition usually provides worse estimates than the optimal cost partition. However, we can compute the saturated cost partitioning much faster and do not need to hold all abstractions in memory simultaneously.

Saturated State-independent Cost Partitioning. In order to show that saturated cost partitioning yields an admissible heuristic even in the presence of general costs, we extend the underlying lemma from the original paper to general costs.

Lemma 1 (Lemma 5 of Seipp and Helmert [2014] extended to general costs)**.** *Let h and h' denote the goal distance functions in two transition systems $\mathcal{T}$ and $\mathcal{T}'$ that differ only in the weight of a single transition $s \to s'$, which is $w \in \mathbb{R}$ in $\mathcal{T}$ and $w' \in \mathbb{R}$ in $\mathcal{T}'$. If $h(s) - h(s') \leq w' \leq w$, then $h = h'$.*

Proof sketch: We only sketch this proof here and refer to the technical report [Keller *et al.*, 2016] for a full proof.

The only difference to Lemma 5 of Seipp and Helmert [2014] is that we allow general costs and hence negative weights in transition systems. With their proof, it is easy to see that negative weights are unproblematic unless there is a negative-cost cycle including the transition $s \to s'$ in $\mathcal{T}'$ but not in $\mathcal{T}$ (because then $h(s') \neq h'(s')$). Assume there is a cyclic path $s \xrightarrow{w} s' \xrightarrow{d}{}^{*} s$, that is only negative in $\mathcal{T}'$, i.e. $w + d > 0$ and $w' + d < 0$. Since h estimates goal distances, we get $h(s') \leq h(s) + d$ and hence $0 \leq h(s) - h(s') + d$. However, from $h(s) - h(s') \leq w'$ and $w' + d < 0$, we get $h(s) - h(s') + d < 0$, which is a contradiction. ∎

With the preceding lemma we can define saturated state-independent cost partitioning for general costs while preserving admissibility.

Definition 3. *Let Π be a planning task with cost function c and $\alpha_1, \ldots, \alpha_n$ abstractions. Let $\langle c_1, \ldots, c_n \rangle$ and $P = \langle \hat{c}_1, \ldots, \hat{c}_n \rangle$ be tuples of cost functions with the following properties: $c_1 = c$; $\hat{c}_i(a) = \max_{s \in S} h_i(\alpha_i(s)) - h_i(\alpha_i(s[a]))$, where h_i is the goal distance function of $\mathcal{T}^{\alpha_i}$ with cost function c_i; and $c_{i+1} = c_i - \hat{c}_i$. We call c_i the* remaining cost *for $\mathcal{T}^{\alpha_i}$, $\hat{c}_i$ the* saturated cost *of $\mathcal{T}^{\alpha_i}$ and P the* saturated state-independent cost partitioning *for $\alpha_1, \ldots, \alpha_n$.*

We denote the associated heuristic by h^{scp_I}. Seipp and Helmert [2014] show that the saturated cost function preserves the goal distances of all abstract states in all abstractions, and is minimal among all distance-preserving cost functions. Next, we illustrate the computation of SCP_I in our example domain.

Example 2. *Consider the two abstract transition systems of our running example that are shown in Figure 2b. The goal distances of $\mathcal{T}_1$ are $h_1(\{AB\}) = 3$, $h_1(\{BB, BT\}) = 2$, $h_1(\{AT, AA\}) = 1$, and $h_1(\{BA\}) = 0$. The saturated*

cost $\hat{c}_1$ of $\mathcal{T}_1$, which annotates the edge labels, assigns the full cost of 1 to both DRIVE-AB *(which induces both $AB^{\rightarrow}$ and $AA^{\rightarrow}$) and* DRIVE-BA *(i.e., $BT^{\rightarrow}$), and 0 to all other actions. The remaining costs c_2, which are used to compute the goal distances in $\mathcal{T}_2$, are hence 0 for all actions except $c_2(BB^{\rightarrow}) = c_2(AT^{\rightarrow}) = 1$. With this, the saturated state-independent cost partitioning for the initial state of our example is $h^{scp_I}(AB) = 3 + 2 = 5$.*

Saturated State-dependent Cost Partitioning. As state-independent cost functions do not allow that costs are assigned to actions in the context of the current state, saturated cost functions are computed by maximizing over all weights of transitions that are labeled with the same action. State-dependent cost partitioning offers an opportunity to overcome this weakness by allowing to reduce the costs of state-action pairs rather than actions.

Definition 4. *Let Π be a planning task with cost function c and $\alpha_1, \ldots, \alpha_n$ abstractions. Let $\langle c_1, \ldots, c_n \rangle$ and $P = \langle \hat{c}_1, \ldots, \hat{c}_n \rangle$ be tuples of cost functions with the following properties: $c_1(a, s) = c(a)$ for all $a \in A$ and $s \in S$; $\hat{c}_i(a, s) = h_i(\alpha(s)) - h_i(\alpha(s[a]))$, where h_i is the goal distance function of $\mathcal{T}^{\alpha_i}$ with cost function c_i; and $c_{i+1} = c_i - \hat{c}_i$. We call c_i the* remaining cost *for $\mathcal{T}^{\alpha_i}$, $\hat{c}_i$ the* saturated cost *of $\mathcal{T}^{\alpha_i}$ and P the* saturated state-dependent cost partitioning *for $\alpha_1, \ldots, \alpha_n$.*

We denote the associated heuristic by h^{scp_D}. Let us investigate the differences between both versions of cost saturation.

Example 3. *Consider the two abstract transition systems in Figure 2c, where the first transition label denotes the goal distances of* $\mathrm{SCP_I}$ *and the second the goal distances of* $\mathrm{SCP_D}$*. Due to the state-independence of* $\mathrm{SCP_I}$*, a cost of 1 is assigned to both occurrences of* DRIVE-AB *in the first abstraction, such that no cost is left in the second one and $h^{scp_I}(AB) = 2 + 1 = 3$. Because* $\mathrm{SCP_D}$ *is state-dependent, the cost of $AB^{\rightarrow}$ can be used in the second abstraction and the heuristic estimate is $h^{scp_D}(AB) = 2 + 2 = 4$.*

In analogy to Theorem 2, Example 3 hints at a theoretical dominance of $\mathrm{SCP_D}$ over $\mathrm{SCP_I}$. However, it turns out that this is not the case due to the inaccuracy caused by the greedy nature of saturated cost partitionings.

Theorem 3. *There are planning tasks Π and Π' with states $s \in S$ and $s' \in S'$ such that $h^{scp_D}(s) > h^{scp_I}(s)$ and $h^{scp_I}(s') > h^{scp_D}(s')$.*

Proof sketch: Example 3 shows that there are instances where $h^{scp_D}(s) > h^{scp_I}(s)$. For $h^{scp_I}(s) > h^{scp_D}(s)$, consider the following example. ∎

Example 4. *Consider the planning task and the three abstract transition systems that are shown in Figure 3. The abstractions in the example are not Cartesian, which is necessary to keep it small. An example that follows the same idea but uses Cartesian abstractions is contained in the technical report [Keller* et al.*, 2016].*

First, consider the state-independent case (first label in the figure): in $\mathcal{T}_1$, only the cost of a_1 is required to maintain the heuristic values, i.e., the saturated cost of $\mathcal{T}_1$ is $\hat{c}_1(a_1) = 1$ and 0 for all other actions. The remaining cost of a_1 for $\mathcal{T}_2$ thus is 0, so the abstract initial state has a heuristic value of 0 in $\mathcal{T}_2$ and the saturated cost of $\mathcal{T}_2$ is 0 for all actions. Since a_0, a_2, and a_3 all still have their original cost in $\mathcal{T}_3$, the abstract initial state of $\mathcal{T}_3$ has a heuristic value of 2. Therefore, the overall heuristic value is $h^{scp_I}(I) = 0 + 0 + 2 = 2$.

Now consider the state-dependent case (second label in the figure): here, the saturated cost of $\mathcal{T}_1$ is $\hat{c}_1(a_1, B) = 1$ but $\hat{c}_1(a_1, A) = 0$, so the remaining cost for $\mathcal{T}_2$ is $c_2(a_1, A) = 1$ and the heuristic value for the abstract initial state of $\mathcal{T}_2$ is 1 instead of 0. This means that the saturated cost of $\mathcal{T}_2$ is $\hat{c}_2(a_0, I) = \hat{c}_2(a_1, A) = \hat{c}_2(a_2, C) = \hat{c}_2(a_3, E) = 1$ and 0 for all other actions, and that no cost remains for a_2 and a_3 in $\mathcal{T}_3$. Therefore, the overall heuristic value is $h^{scp_D}(I) = 0 + 1 + 0 = 1 < 2 = h^{scp_I}(I)$.

In Theorems 2 and 3 we have investigated the relationship between $\mathrm{OCP_D}$ and $\mathrm{OCP_I}$ on the one hand and $\mathrm{SCP_D}$ and $\mathrm{SCP_I}$ on the other. What is left is the relationship between $\mathrm{SCP_D}$ and $\mathrm{OCP_I}$.

Corollary 1. *There are a planning tasks Π and Π' with states $s \in S$ and $s' \in S'$ such that $h^{scp_D}(s) > h^{ocp_I}(s)$ and $h^{ocp_I}(s') > h^{scp_D}(s')$.*

Proof: For the first part, consider Examples 1 and 3 where $h^{scp_D}(AB) = 4$ and $h^{ocp_I}(AB) = 3$. The second part follows from Definition 2 and Theorem 3. ∎

Figure 4 shows a summary of our theoretical results (where $A \succ B$ means A dominates B). While optimal state-dependent cost partitioning clearly combines the best of both worlds, we leave it for future work, since computing it is exponential. On the other hand, saturated state-dependent cost partitioning may not always result in better heuristic estimates, but it has the potential to surpass optimal state-independent cost partitioning, which warrants further investigation. We therefore discuss practical considerations for the computation of saturated state-dependent cost partitioning in the following section.

6 Implementation Details of h^{scp_D}

Given a set of abstractions, Section 5 reveals a general workflow to compute saturated state-dependent cost partitionings. For each abstraction we have to apply the following four steps: (1) determine the abstract transition weights (with the current remaining cost function), (2) compute the abstract goal distances, (3) compute the saturated state-dependent cost function, and (4) determine the remaining cost function. Note that the cost function for the first abstraction is the (state-independent) original cost function, while subsequent abstractions use the remaining costs, computed in the previous iteration. In general, it is not important how we obtain the abstractions, although some abstractions or abstraction orders may result in better heuristics than others. We will briefly discuss this in Section 7. In fact, we do not even require Cartesian abstractions. When we describe the implementation of the four steps above in the remainder of this section, we will, however, see that Cartesian abstractions are necessary to guarantee some important properties.

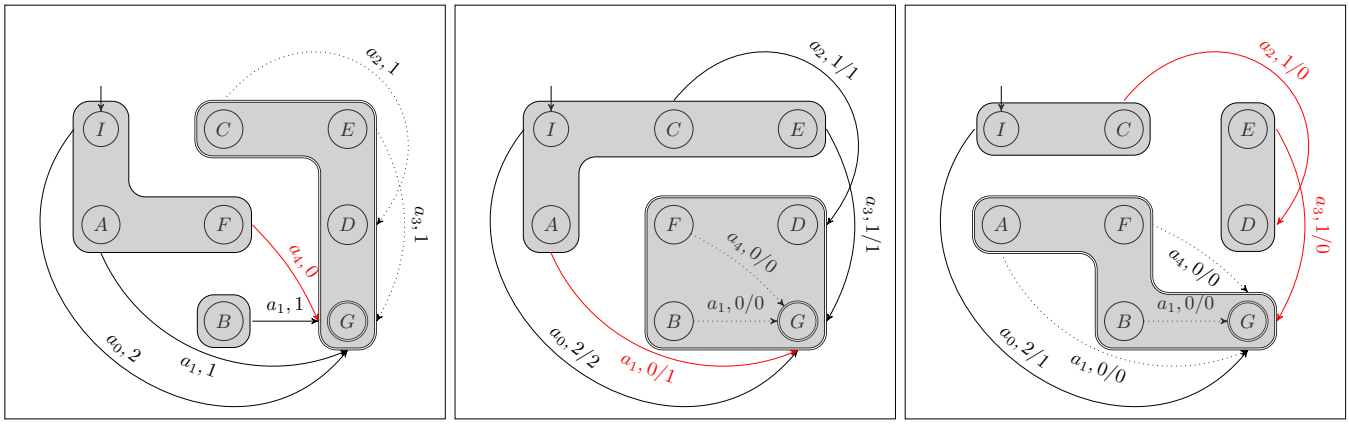


Figure 3: Example for a planning task and abstractions where $h^{scp_I}(I) > h^{scp_D}(I)$. Circles depict concrete states, abstract states are rectangular, and dotted arcs are abstract self-loops. A shortest path from the initial abstract state to the goal abstract state is highlighted in red. We denote the transition systems with $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$ from left to right. Edge labels denote costs with different cost partitionings.

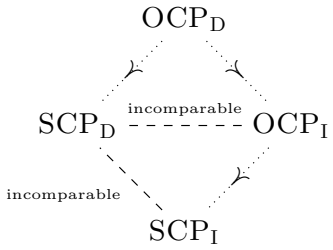


Figure 4: Summary of theoretical dominance results.

Edge-valued Decision Diagrams. To encode state-dependent cost functions, we need a suitable representation. We chose edge-valued multi-valued decision diagrams, because they are (i) often *compact*, albeit worst-case exponential, they (ii) allow *efficient computation* of *abstract* cost values as long as the abstract states are *Cartesian* [Geißer *et al.*, 2016], and they (iii) admit reasonably *efficient arithmetic operations*, in particular subtraction of the current saturated costs from the current remaining costs that is needed to determine the remaining costs for the next iteration. Taking binary sums and differences is linear in the product of the sizes of the input EVMDDs, a result that can be easily generalized from a result by Lai *et al.* [1996] on EVBDDs. EVMDDs work as follows: like BDDs [Bryant, 1986], EVMDDs contain decision nodes. At each decision node, the diagram branches over the value of the associated decision variable. There are three differences from BDDs: first, branchings can be n-ary instead of only binary, second, the values returned by evaluating an EVMDD can come from a finite range of values instead of just being true or false, and third, the return values are associated with the *edges* of the diagram instead of the *leaves*. When traversing an EVMDD for a single valuation s, the resulting value is the sum of edge weights along the unique path corresponding to s.

Determining Abstract Transition Weights. Assume that we have an EVMDD representing the remaining cost function for the current abstraction of some action a, and that we want to use this EVMDD to determine abstract transition weights for the next abstraction we build. The main challenge is this: with the EVMDD, we can easily determine what a costs in any *concrete* state. Geißer *et al.* [2016] show that we can use the same EVMDD to efficiently determine what a costs in any *abstract* state as well, as long as the abstract state is Cartesian. Here, we extend their technique in order to obtain more accurate transition weights depending not only on the source state of the transition, but also on its target state. Note that for deterministic transition systems, this does not make a difference, but for non-deterministic transition systems it does, and that the abstractions we consider here may indeed introduce non-determinism. More specifically, we want to be able to assign two *different* weights to two different abstract transitions $t \xrightarrow{a} u_1$ and $t \xrightarrow{a} u_2$ with the same abstract source state t and the same action label a depending on whether it leads to u_1 or u_2. This gives us more accurate abstract goal distances. In order to accomplish this, we (i) identify via regression the subset t_1 of t where a is applicable and leads to u_1, and (ii) restrict t to that subset; and similarly for u_2. Both regression and intersection of Cartesian sets preserve Cartesianness [Seipp and Helmert, 2013], so we can still safely assume that t_1 and t_2 are Cartesian sets and hence admit efficient EVMDD operations. Note that any two such t_1, t_2 that are computed by regression are indeed disjoint, even if they originate from the same abstract state t via regression through the same action a from two different states u_1 and u_2. Figure 5 illustrates the idea of improving the abstraction via regression. The abstract state t contains four concrete states where a is applicable, at costs 2, 3, 4, and 5, respectively. Simply minimizing the cost of a over all four states would require us to assign cost 2 to a in the entire set t. However, by partitioning the four concrete states into the two states leading to successors in u_1 at cost 3 and 5 (represented by the Cartesian set t_1), and into those two states leading to successors in

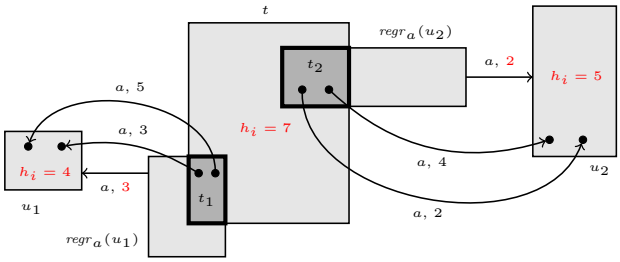


Figure 5: Abstract transition weights.

u_2 at cost 2 and 4 (represented by the Cartesian set t_2), we can now assign cost 2 to a in t_2, and the higher cost of 3 to a in t_1.

Computing Saturated State-dependent Cost Functions. Computing abstract goal distances is straightforward, e. g., using Dijkstra's algorithm in a backward manner. We skip discussing that and move on to computing saturated state-dependent cost functions. We already know how to *compute* them from the theoretical part of this paper. Here, we discuss how to efficiently *represent* them using EVMDDs. Assume that after regression and saturation, action a costs δ_k in Cartesian set t_k, $k = 1, \ldots, m$. Then we can obtain an EVMDD that encodes the saturated cost function of a as follows: for each set t_k, we can efficiently compute an EVMDD that encodes the *characteristic function* of t_k weighted with its cost δ_k. This EVMDD branches on all relevant variables in sequence, routing branches inconsistent with t_k immediately to the sink and branches consistent with it to the next decision node/variable. All edge weights are zero except for those leading from the last decision variable to the sink that are consistent with t_k. Those carry weight δ_k. Moreover, we can compute an EVMDD representing the *sum* of those characteristic-function EVMDDs for $k = 1, \ldots, m$, which is exactly the saturated cost function we are after. Taking this sum is exponential only in the number of Cartesian sets m and nothing else. In general, this cannot be avoided, not even in the Cartesian setting. Note that for abstract sets which do not lead to the goal we can assign sufficiently small saturated cost values to concrete states mapped to them. This results in arbitrarily large remaining costs, which mirrors the observation that applying the action in such a state makes the task unsolvable. Finally, representing the new remaining cost function of a as an EVMDD is trivial. We have EVMDDs for the previous remaining cost function and for the current saturated cost function of a. Taking the difference of two functions is a simple EVMDD operation and returns the desired EVMDD.

7 Empirical Evaluation

Even though the focus of this paper is on theoretical aspects of state-dependent cost partitionings, we have performed a brief empirical evaluation as well. Its aim is to investigate whether the potentially exponential blowup of the EVMDD size is critical in practice and how relevant state-dependent cost partitionings are in practice. It is important to note that we are not evaluating an elaborate algorithm that is optimized for the computation of accurate heuristic values with state-dependent cost partitionings. The creation of abstractions that are well suited to be used in combination with state-dependent cost partitionings is in particular not the focus of this paper, but plays an important role when accurate heuristic estimates are desired. Moreover, we did not implement any form of variable ordering in the decision diagrams [Rudell, 1993], which would further reduce their exponential blowup.

We have implemented the saturated state-dependent cost partitioning algorithm that is sketched in the previous section on top of the CEGAR implementation of Seipp and Helmert [2013] in the Fast Downward planning system [Helmert, 2006]. Our algorithms use the same parameter settings as the configuration that performed best for saturated state-independent cost partitionings in the original work on the topic [Seipp and Helmert, 2014] with two exceptions: first, we used a maximal number of states (10000) instead of a timeout to decide when the heuristic computation terminates in order to make the abstraction generation process deterministic. And second, we do not interleave the generation of an abstraction based on the latest remaining cost function with the computation of its saturated cost function but compute all abstractions with the original cost function instead. Combined, both changes make sure that all algorithms compute their respective cost partitionings on identical abstract transition systems.

We have performed experiments on all supported IPC 1998–2014 benchmarks on Intel Xeon E5-2660 CPUs running at 2.2 GHz with a time limit of 30 minutes and a memory limit of 2 GB. The exponential blowup of the EVMDD size is critical in only six out of 57 domains, namely FREECELL, MPRIME, MYSTERY, PIPESWORLD-NOTANKAGE, PIPESWORLD-TANKAGE, and TIDYBOT 2011. However, a simple procedure that uses saturated state-independent costs for all abstractions where the number of transitions with non-zero cost is too large leads to an algorithm that preserves the advantages of state-dependent cost partitionings in most domains and mitigates the risk of unmanageable EVMDDs successfully.

Table 1 gives the number of instances per domain where the heuristic value of the initial state with h^{scp_D} is larger, smaller and derived from a different sum of heuristic values in comparison to h^{scp_I}. The state-dependent version improves over the state-independent one in 228 out of 1667 instances, it performs worse in 76, and there are another 25 instances where both compute the same heuristic estimate for the initial state but based on different sums. It is a promising result that h^{scp_D} is already better in approximately $\frac{1}{7}$ of the instances even when the considered abstractions are not tailored to exploit context information. We believe that the fact that h^{scp_I} performs better on an instance is a sign that state-dependent costs are relevant in that instance, and it may very well only be the greediness of saturated cost partitionings, the poor abstraction generation or even just the abstraction order that prevent a result where h^{scp_D} is better than h^{scp_I} instead of worse.

Even when the estimates do not differ, we assume that it is often only the choice of abstractions that prevents an improved heuristic estimate with state-dependent cost partitionings. A look at the domains where the heuristic estimates

Domain	$h^{scp_D} > h^{scp_I}$	$h^{scp_D} < h^{scp_I}$	different sum
AIRPORT (50)	1	1	2
BARMAN 2011 (20)	16	0	16
BARMAN 2014 (14)	8	1	9
BLOCKS (35)	21	4	26
DEPOT (22)	4	3	7
DRIVERLOG (20)	1	5	7
ELEVATORS 2008 (30)	19	1	22
ELEVATORS 2011 (20)	12	1	15
FLOORTILE 2011 (20)	7	4	13
FLOORTILE 2014 (20)	7	5	14
GRID (5)	4	0	4
HIKING 2014 (20)	12	2	15
MPRIME (35)	1	0	2
MYSTERY (30)	0	0	1
NOMYSTERY 2011 (20)	4	1	5
OPENSTACKS (30)	2	0	2
OPENSTACKS 2008 (30)	10	0	10
OPENSTACKS 2011 (20)	6	0	6
OPENSTACKS 2014 (20)	3	0	3
PARCPRINTER 2008 (20)	7	1	8
PARCPRINTER 2011 (20)	5	1	6
PARKING 2011 (20)	0	1	1
PATHWAYS-NONEG (30)	3	16	22
PSR-SMALL (50)	11	0	11
ROVERS (40)	3	4	8
SATELLITE (36)	7	3	11
SCANALYZER 2008 (30)	1	0	1
SCANALYZER 2011 (20)	1	0	1
SOKOBAN 2008 (30)	6	0	6
SOKOBAN 2011 (20)	3	0	3
TETRIS 2014 (17)	1	1	2
TPP (30)	8	2	11
TRANSPORT 2008 (30)	14	4	20
TRANSPORT 2011 (20)	11	6	18
TRANSPORT 2014 (20)	9	9	18
TRUCKS (30)	0	0	1
WOODWORKING 2008 (30)	0	0	1
WOODWORKING 2011 (20)	0	0	1
Remaining domains (723)	0	0	0
Total (1667)	228	76	329

Table 1: Number of instances per domain where $h^{scp_D}(s_I)$ is larger, smaller or the result of a different sum than $h^{scp_I}(s_I)$.

differ in at least one instance reveals that similarly structured problems (like, for instance, route or transportation problems) are either present with a large number of representatives or not at all. This hints at the fact that the relevance of context information is domain- and not instance-dependent (i.e., relevant in all instances that are part of a domain with at least one instance were the heuristic estimates differ). Under the assumption that this is the case, there are 944 instances among the 1667 and hence more than half of them where state-dependent cost partitionings have the potential to improve over state-independent ones. We believe that this is a promising result that encourages future work on the topic, e.g., on well-suited abstraction generation methods or other state-dependent cost partitionings.

8 Conclusion

We generalized the concept of cost partitionings even further and showed that additional information can be extracted from a set of abstractions if context information of applied actions is taken into account. We showed that an optimal state-dependent cost partitioning dominates all state-independent cost partitionings and that there are planning tasks where the dominance is strict. As it is unclear how an optimal state-dependent cost partitioning can be computed efficiently in practice, we applied the idea to the efficiently computable saturated cost partitioning. We showed that saturated state-dependent cost partitioning does not dominate its state-independent sibling, but may still surpass optimal state-independent cost partitioning. We discussed practical considerations regarding saturated state-dependent cost partitioning and reasoned that EVMDDs are a suitable representation for the resulting cost functions as long as the abstractions are Cartesian. A baseline implementation of our approach revealed that there are indeed many instances among the IPC benchmarks where the heuristic estimates improve under a state-dependent cost partitioning. It furthermore allows to conjecture that taking context information into account can pay off in the majority of IPC benchmark tasks.

This work also opens up several possible directions for future work. Saturated state-dependent cost partitionings can be improved by reasoning about the order of the considered abstract transitions systems or with methods that generate Cartesian abstractions that are tailored to exploit context information. Dynamic variable orderings for decision diagrams are an enhancement that has the potential to further minimize the risk that an EVMDD's size gets out of hand. And finally, it should be possible to narrow the gap between the (error-prone) saturated state-dependent cost partitionings on the one hand and the optimal ones (which are intractable in practice) on the other. Context splitting, for instance, seems promising as it provides a tool that allows an iterative compilation of the input task, which might give rise to near-optimal state-dependent cost partitionings in the future.

Acknowledgments

This work was supported by the European Research Council as part of the project "State Space Exploration: Principles, Algorithms and Applications" and by BMBF grant 02PJ2667 as part of the KARIS PRO project.

References

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

[Bonet and van den Briel, 2014] Blai Bonet and Menkes van den Briel. Flow-based heuristics for optimal planning: Landmarks and merges. In *Proc. ICAPS 2014*, pages 47–55, 2014.

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[Ciardo and Siminiceanu, 2002] Gianfranco Ciardo and Radu Siminiceanu. Using edge-valued decision diagrams for symbolic generation of shortest paths. In *Proceedings of the Fourth International Conference on Formal Methods in Computer-Aided Design (FMCAD 2002)*, pages 256–273. Springer Berlin Heidelberg, 2002.

[Culberson and Schaeffer, 1998] Joseph C. Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.

[Edelkamp, 2001] Stefan Edelkamp. Planning with pattern databases. In *Proc. ECP 2001*, pages 84–90, 2001.

[Felner *et al.*, 2004] Ariel Felner, Richard Korf, and Sarit Hanan. Additive pattern database heuristics. *JAIR*, 22:279–318, 2004.

[Geißer *et al.*, 2015] Florian Geißer, Thomas Keller, and Robert Mattmüller. Delete relaxations for planning with state-dependent action costs. In *Proc. IJCAI 2015*, pages 1573–1579, 2015.

[Geißer *et al.*, 2016] Florian Geißer, Thomas Keller, and Robert Mattmüller. Abstractions for planning with state-dependent action costs. In *Proc. ICAPS 2016*, 2016. To appear.

[Haslum *et al.*, 2007] Patrik Haslum, Adi Botea, Malte Helmert, Blai Bonet, and Sven Koenig. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proc. AAAI 2007*, pages 1007–1012, 2007.

[Helmert and Mattmüller, 2008] Malte Helmert and Robert Mattmüller. Accuracy of admissible heuristic functions in selected planning domains. In *Proc. AAAI 2008*, pages 938–943, 2008.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *JAIR*, 26:191–246, 2006.

[Karpas and Domshlak, 2009] Erez Karpas and Carmel Domshlak. Cost-optimal planning with landmarks. In *Proc. IJCAI 2009*, pages 1728–1733, 2009.

[Katz and Domshlak, 2007] Michael Katz and Carmel Domshlak. Structural patterns of tractable sequentially-optimal planning. In *Proc. ICAPS 2007*, pages 200–207, 2007.

[Katz and Domshlak, 2010] Michael Katz and Carmel Domshlak. Optimal admissible composition of abstraction heuristics. *AIJ*, 174(12–13):767–798, 2010.

[Keller *et al.*, 2016] Thomas Keller, Florian Pommerening, Jendrik Seipp, Florian Geißer, and Robert Mattmüller. State-dependent cost partitionings for cartesian abstractions in classical planning: Full proofs. Technical Report CS-2016-002, University of Basel, Department of Mathematics and Computer Science, 2016.

[Korf and Felner, 2002] Richard E. Korf and Ariel Felner. Disjoint pattern database heuristics. *AIJ*, 134(1–2):9–22, 2002.

[Lai *et al.*, 1996] Yung-Te Lai, Massoud Pedram, and Sarma B. K. Vrudhula. Formal verification using edge-valued binary decision diagrams. *IEEE Transactions on Computers*, 45(2):247–255, 1996.

[Pommerening *et al.*, 2015] Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. In *Proc. AAAI 2015*, pages 3335–3341, 2015.

[Röger *et al.*, 2014] Gabriele Röger, Florian Pommerening, and Malte Helmert. Optimal planning in the presence of conditional effects: Extending LM-Cut with context splitting. In *Proc. ECAI 2014*, pages 80–87, 2014.

[Rudell, 1993] Richard Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Proc. ICCAD 1993*, pages 42–47, 1993.

[Seipp and Helmert, 2013] Jendrik Seipp and Malte Helmert. Counterexample-guided Cartesian abstraction refinement. In *Proc. ICAPS 2013*, pages 347–351, 2013.

[Seipp and Helmert, 2014] Jendrik Seipp and Malte Helmert. Diverse and additive Cartesian abstraction heuristics. In *Proc. ICAPS 2014*, pages 289–297, 2014.