Cost-Partitioned Merge-and-Shrink Heuristics for Optimal Classical Planning

Silvan Sievers, **Florian Pommerening**, **Thomas Keller** and **Malte Helmert**
University of Basel, Switzerland
{silvan.sievers,florian.pommerening,tho.keller,malte.helmert}@unibas.ch

Abstract

Cost partitioning is a method for admissibly combining admissible heuristics. In this work, we extend this concept to merge-and-shrink (M&S) abstractions that may use labels that do not directly correspond to operators. We investigate how optimal and saturated cost partitioning (SCP) interact with M&S transformations and develop a method to compute SCPs during the computation of M&S. Experiments show that SCP significantly improves M&S on standard planning benchmarks.

1 Introduction

Classical planning [Ghallab *et al.*, 2004] aims to find a sequence of actions that leads from an initial world situation to a specified goal. A state-of-the-art approach to optimally solving classical planning problems is A* search [Hart *et al.*, 1968] with an admissible heuristic [Pearl, 1984]. Abstractions are a state-of-the-art class of admissible heuristics and contain the *merge-and-shrink (M&S)* framework [Dräger *et al.*, 2009; Helmert *et al.*, 2014; Sievers *et al.*, 2014], which repeatedly applies transformations to a given state space.

A common problem of abstractions is that they often contain useful information only for parts of a planning task, while completely ignoring the rest. A possible solution to this problem is the combination of different abstractions admissibly with *operator cost partitioning* [Katz and Domshlak, 2010; Seipp *et al.*, 2017a], which distributes the operator costs among the abstractions guaranteeing that the sum of their heuristic values is admissible. It has successfully been used with Cartesian abstractions [Seipp and Helmert, 2018] and pattern databases [Edelkamp, 2001], but not yet with M&S.

In this paper, we take a first step towards closing this gap. We transfer the notion of operator cost partitioning to M&S abstractions, thus defining *label cost partitionings*. We then analyze how the heuristic value of *optimal* [Katz and Domshlak, 2010; Pommerening *et al.*, 2015] and *saturated* cost partitioning (*SCP*) [Seipp and Helmert, 2018] is affected by different M&S transformations. Based on these insights, we present a practical approach that interleaves computing SCPs with the M&S algorithm. An experimental evaluation on standard benchmarks shows that combining M&S heuristics in SCPs significantly improves their performance.

2 Background

2.1 Classical Planning

Although our techniques are applicable to any transition system with a so-called factored structure, we show-case them using classical planning tasks in the SAS$^+$ formalism [Bäckström and Nebel, 1995]. Such a task is defined as $\Pi = \langle \mathcal{V}, \mathcal{O}, \mathit{cost}, s_0, s_\star \rangle$. $\mathcal{V}$ is a finite set of *state variables* v, each with a *finite domain* $\mathit{dom}(v)$. A *partial state* s is an assignment over $\mathit{vars}(s) \subseteq \mathcal{V}$, mapping each $v \in \mathit{vars}(s)$ to a value $d \in \mathit{dom}(v)$. It is a state if $\mathit{vars}(s) = \mathcal{V}$. State s *satisfies* partial state s', written $s \models s'$, if $s(v) = s'(v)$ for all $v \in \mathit{vars}(s')$. $\mathcal{O}$ is a finite set of *operators* $o = \langle \mathit{pre}_o, \mathit{eff}_o \rangle$, where pre_o and eff_o are partial states called the *precondition* and *effect* of o. The *cost function* $\mathit{cost} : \mathcal{O} \to \mathbb{R}_0^+$ maps each operator to its cost. Finally, s_0 is the *initial state*, and $s_\star$ a partial state called the *goal*.

For a state s and operator o with $s \models \mathit{pre}_o$ the *successor state* $s[\![o]\!]$ is defined as $s[\![o]\!](v) = \mathit{eff}_o(v)$ for all $v \in \mathit{vars}(\mathit{eff}_o)$ and $s[\![o]\!](v) = s(v)$ for all other variables. A planning task Π induces a *transition system* $\Theta(\Pi) = \langle S, L, \mathit{cost}, T, s_\mathrm{I}, S_\mathrm{G} \rangle$, where S is the set of states over $\mathcal{V}$; $L = \mathcal{O}$ are *labels* with costs $\mathit{cost}(\ell) = \mathit{cost}(o)$ for $\ell \in L$; T are the *transitions* $\{s \xrightarrow{\ell} t \mid s, t \in S, \ell \in L, s \models \mathit{pre}_\ell, t = s[\![\ell]\!]\}$; $s_\mathrm{I} = s_0$; and $S_\mathrm{G} = \{s \in S \mid s \models s_\star\}$ are the *goal states*. For a transition system Θ, we write $S(\Theta)$ for its set of states.

A solution for a planning task is a *plan*, i.e. a sequence of labels that induces a *path* in $\Theta(\Pi)$ from s_I to some state in S_G. The cost of a plan is the sum of its label costs. Optimal planning aims at finding a plan of minimal cost or proving that no plan exists. To optimally solve planning tasks, we use the A* algorithm with *admissible heuristics*. A heuristic for a transition system Θ with labels L and cost function *cost* is a function $h : S(\Theta) \to \mathbb{R} \cup \{\infty\}$ which estimates the cost of reaching a goal state from s in Θ. We write $h(s, \mathit{cost}')$ for the evaluation of h on state $s \in S(\Theta)$, however using an alternative cost function *cost*$'$ instead of *cost*. A heuristic h is admissible if for all states $s \in S(\Theta)$, $h(s) \leq h^*(s)$, where $h^*(s)$ is the true cost of reaching a goal state from s.

2.2 Merge-and-Shrink

Merge-and-shrink (M&S) is a framework for computing *abstractions* of transition systems. An abstraction maps a transition system to a different transition system that pre-

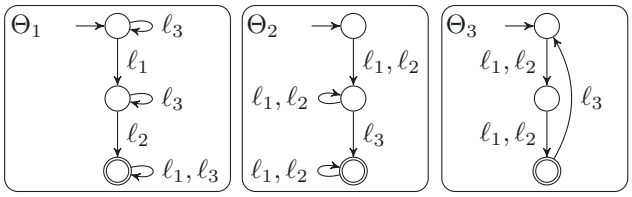

Figure 1: Factored transition system with 3 factors.

serves all plans but is typically smaller and easier to solve. Formally, an abstraction α of a transition system $\Theta = \langle S, L, \mathit{cost}, T, s_\mathrm{I}, S_\mathrm{G}\rangle$ is a homomorphism defined on the states S of Θ. The induced abstract transition system is defined as $\alpha(\Theta) = \langle\{\alpha(s) \mid s \in S\}, L, \mathit{cost}, \{\alpha(s) \xrightarrow{\ell} \alpha(t) \mid s \xrightarrow{\ell} t \in T\}, \alpha(s_\mathrm{I}), \{\alpha(s) \mid s \in S_\mathrm{G}\}\rangle$.

M&S is applicable to transition systems which exhibit a so-called *factored* structure: a *factored transition system (FTS)* $F = \langle\Theta_1, \ldots, \Theta_n\rangle$ consists of transition systems Θ_i (also called *factors*) sharing the same set of labels and the same cost function. It is a compact ("factored") representation of its *product system* $\bigotimes F = \langle S^\otimes, L, \mathit{cost}, T^\otimes, s_\mathrm{I}^\otimes, S_\mathrm{G}^\otimes\rangle$, where $S^\otimes = \prod_{i=1}^n S^i$ is the Cartesian product of the state sets; there is a transition $\langle s^1, \ldots, s^n\rangle \xrightarrow{\ell} \langle t^1, \ldots, t^n\rangle$ in $T^\otimes$ iff $s^i \xrightarrow{\ell} t^i \in T^i$ for all $1 \leq i \leq n$; the initial state is $s_\mathrm{I}^\otimes = \langle s_\mathrm{I}^1, \ldots, s_\mathrm{I}^n\rangle$; and the set of goal states is $S_\mathrm{G}^\otimes = \prod_{i=1}^n S_\mathrm{G}^i$. We write $S(F)$ to denote the states of the product of F.

A planning task $\Pi = \langle\mathcal{V}, \mathcal{O}, \mathit{cost}, s_0, s_\star\rangle$ induces the FTS $F(\Pi)$ consisting of the *atomic* factors Θ_v for all variables $v \in \mathcal{V}$: Θ_v is the projection of Π to v, i.e., $\Theta_v = \langle \mathit{dom}(v), \mathcal{O}, \mathit{cost}, T^v, s_0[v], S_\mathrm{G}^v\rangle$. The set of transitions T^v consists of the transitions $s \xrightarrow{o} t$ for the state $s = \mathit{pre}_o[v]$ if $v \in \mathit{vars}(\mathit{pre}_o)$ and for all states $s \in \mathit{dom}(v)$ otherwise, and for the state t with $t = \mathit{eff}_o[v]$ if $v \in \mathit{vars}(\mathit{eff}_o)$ and $t = s$ otherwise. If v is not mentioned in the goal of Π, all states of Θ_v are goal states ($S_\mathrm{G}^v = \mathit{dom}(v)$), otherwise there is a single goal state $S_\mathrm{G}^v = \{s_\star[v]\}$.

Figure 1 shows an example FTS $F = \langle\Theta_1, \Theta_2, \Theta_3\rangle$. When drawing transition systems, we mark goal states with a double circle and the initial state with an incoming arrow without label. All other arrows are labeled and denote transitions of the transition system. Unless stated otherwise, we assume a constant label cost of 1 (called *unit cost*).

Starting from any given FTS $F = \langle\Theta_1, \ldots, \Theta_n\rangle$, M&S iteratively applies *transformations* of F. To do so, it keeps track of the current FTS F', a state mapping $\Sigma : S(F) \to S(F')$ from states of F to states of F', and a label mapping $\lambda : L \to L'$ from labels L of F to labels L' of F'. A transformation from F to F' specifies F', the state mapping Σ, and the label mapping λ. Applying such a transformation means to replace the current FTS F by F' and to *compose* the maintained mappings with those of the transformation. We consider three transformations of the M&S framework:

Merging. Merging replaces two factors of F, w.l.o.g. Θ_1 and Θ_2, by their product. Formally: $F' = \langle\Theta_1 \otimes \Theta_2, \Theta_3, \ldots, \Theta_n\rangle$, $\Sigma(\langle s_1, \ldots, s_n\rangle) = \langle\langle s_1, s_2\rangle, s_3, \ldots, s_n\rangle$ (where $\langle s_1, s_2\rangle$ is the product state in $\Theta_1 \otimes \Theta_2$ that results from the previous states s_1 of Θ_1 and s_2 of Θ_2), and $\lambda = \mathrm{id}$.

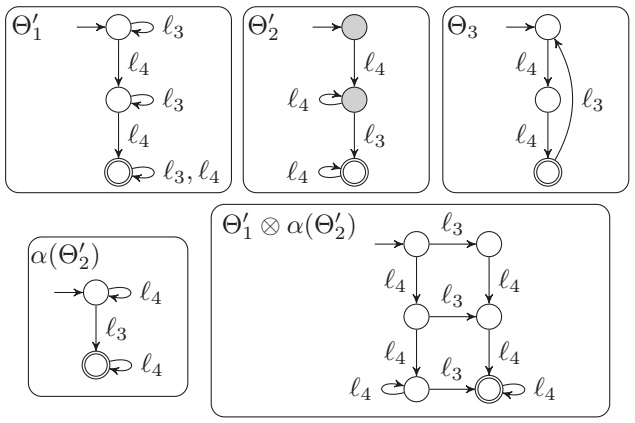

Figure 2: Top: FTS $F' = \langle\Theta_1, \Theta'_2, \Theta'_3\rangle$ obtained from F shown in Figure 1 through label reduction. Bottom: Factor $\alpha(\Theta'_2)$ obtained by shrinking Θ'_2 such that the gray states are combined; Factor $\Theta'_1 \otimes \alpha(\Theta'_2)$ obtained by merging Θ'_1 and $\alpha(\Theta'_2)$.

Shrinking. Shrinking applies an abstraction α to one of the factors of F, w.l.o.g. Θ_1. Formally: $F' = \langle\alpha(\Theta_1), \Theta_2, \ldots, \Theta_n\rangle$, $\Sigma(\langle s_1, \ldots, s_n\rangle) = \langle\alpha(s_1), s_2, \ldots, s_n\rangle$, and $\lambda = \mathrm{id}$. Shrinking is called h-preserving iff $h^*_{\Theta_1}(s) = h^*_{\alpha(\Theta_1)}(\alpha(s))$ for all $s \in S(\Theta_1)$.

Label reduction. Analogously to shrinking, label reduction applies a homomorphism to the set of labels of F. Formally: $\lambda : L \to L'$ is the label mapping of the reduction, Σ is the identity mapping, and $F' = \langle\Theta'_1, \ldots, \Theta'_n\rangle$, where Θ'_i is defined based on $\Theta_i = \langle S^i, L, \mathit{cost}, T^i, s_\mathrm{I}^i, S_\mathrm{G}^i\rangle$ as follows: $\Theta'_i = \langle S^i, L', \mathit{cost}', \{s \xrightarrow{\lambda(\ell)} t \mid s \xrightarrow{\ell} t \in T^i\}, s_\mathrm{I}^i, S_\mathrm{G}^i\rangle$ with $\mathit{cost}'(\ell') = \min_{\ell \in \lambda^{-1}(\ell')} \mathit{cost}(\ell)$ for all $\lambda' \in L'$.

All three types of transformations guarantee that at any time of the M&S computation, each factor Θ_i is an abstraction of the original product $\bigotimes F$.

Example 1. *Consider the FTS F shown in Figure 1. In a first step, a label reduction transformation λ combines the labels ℓ_1, ℓ_2 into a new label ℓ_4. The resulting transformed FTS $F' = \langle\Theta'_1, \Theta'_2, \Theta'_3\rangle$ is shown at the top of Figure 2.*

In a second step, a shrink transformation applies an abstraction α to Θ'_2 such that the two gray states are combined. The resulting FTS is $F'' = \langle\Theta'_1, \alpha(\Theta'_2), \Theta'_3\rangle$ and the factor $\alpha(\Theta'_2)$ is shown in the bottom left of Figure 2 (the other factors remain unchanged).

Finally, a merge transformation merges Θ'_1 and $\alpha(\Theta'_2)$ of F''. The product is shown in the bottom right of Figure 2. The final FTS is $F''' = \langle(\Theta'_1 \otimes \alpha(\Theta'_2)), \Theta'_3\rangle$.

In the context of planning, M&S is usually used to compute a heuristic. Let FTS $F' = \langle\Theta_1, \ldots, \Theta_n\rangle$ and state mapping Σ be the result obtained when executing M&S on some FTS F. We define the *factor heuristic* h_{Θ_i} for F as the perfect heuristic for the i-th factor of F': $h_{\Theta_i}(s) = h^*_{\Theta_i}(\Sigma(s)_i)$. Then, the M&S heuristic for F is defined as the maximum over the factor heuristics: $h^{\mathrm{M\&S}}_F(s) = \max_{1 \leq i \leq n} h_{\Theta_i}(s)$ for all $s \in S(F)$. For the FTS F of Figure 1, we obtain $h^{\mathrm{M\&S}}_F(s_\mathrm{I}) = \max(h_{\Theta_1}(s_\mathrm{I}), h_{\Theta_2}(s_\mathrm{I}), h_{\Theta_3}(s_\mathrm{I})) = \max(2, 2, 2) = 2$. For the FTS F''' from Example 1, we obtain $h^{\mathrm{M\&S}}_{F'''}(s_\mathrm{I}) = \max(3, 1) = 3$.

As mentioned above, all transformations are such that the result is an abstraction. Therefore, $h^{\text{M\&S}}$ is admissible. Merging is even an *exact* transformation, i.e., the perfect heuristic values of the product system of the FTSs are unchanged.

Shrinking with abstraction α is not exact in general, but is exact if it combines states according to a bisimulation relation $\sim$ between states [Nissim *et al.*, 2011]. Such a relation satisfies $s \sim t$ ("s and t are bisimilar") iff 1) either both s, t are goal states or both are not goal states, and 2) for all $s \xrightarrow{\ell} s'$, there exists $t \xrightarrow{\ell} t'$ with $s' \sim t'$. Shrinking is based on a bisimulation $\sim$ if $\alpha(s) = \alpha(t)$ iff $s \sim t$. It is guaranteed to be h-preserving. The two states combined in Example 1 are not bisimilar for any $\sim$, and the transformation is not exact.

Label reduction with label mapping λ is also not exact in general, but is exact if there is $\Theta \in F$ such that for all labels $\ell_1, \ell_2 \in L$ with $\lambda(\ell_1) = \lambda(\ell_2)$, we have $\mathit{cost}(\ell_1) = \mathit{cost}(\ell_2)$ and ℓ_1 and ℓ_2 are Θ-*combinable* [Sievers *et al.*, 2014]. Two labels ℓ_1 and ℓ_2 are called Θ-combinable if they have parallel transitions in all factors except Θ, i.e., if $s \xrightarrow{\ell_1} t \in T'$ iff $s \xrightarrow{\ell_2} t \in T'$ for the transitions T' of all $\Theta' \in F \setminus \{\Theta\}$. Labels ℓ_1 and ℓ_2 combined in Example 1 are Θ_1-combinable because they have parallel transitions in Θ_2 and Θ_3. Since they also have the same cost, the transformation is exact.

To obtain a concrete M&S algorithm, we must decide when to perform which transformation. We follow the literature and define the algorithm as shown in Algorithm 1 in Section 6, for simplicity without showing state and label mappings. (Ignore H, which is related to cost partitioning, discussed later.) The main loop (lines 4–11) runs until the current FTS F' only consists of one element or a time limit is reached. One iteration of the loop does the following: decide which two factors Θ_i, Θ_j of F' to merge in this iteration (line 5), apply exact label reduction (line 6), possibly shrink Θ_i and/or Θ_j to respect a size limit (lines 8 and 9), and finally perform the merge (line 10). The algorithm returns $h_F^{\text{M\&S}}$ computed over F' (line 12 shows how the extended algorithm computes a cost-partitioned heuristic instead). All transformations are parameterized by so-called *transformation strategies*.

2.3 Cost Partitioning

Operator cost partitioning [Katz and Domshlak, 2010] allows to admissibly combine admissible heuristics by distributing operator costs among them. Let Π be a planning task with operator costs *cost* and let $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$ be admissible heuristics for Π. The cost functions $\mathcal{C} = \langle \mathit{cost}_1, \ldots, \mathit{cost}_n \rangle$ form a *cost partition* if $\sum_{i=1}^n \mathit{cost}_i \leq \mathit{cost}$. The *cost-partitioned heuristic* $h_{\mathcal{H},\mathcal{C}}(s) = \sum_{i=1}^n h_i(s, \mathit{cost}_i)$ is admissible. We discuss two methods to design cost partitions.

Saturated cost partitioning (SCP). The SCP method [Seipp *et al.*, 2020] greedily assigns costs to the heuristics $\mathcal{H}$ in a given *order* ω. To simplify the presentation, we consider the order $\omega = \langle h_1, \ldots, h_n \rangle$ but in general, ω can be any permutation of the heuristics in $\mathcal{H}$. The algorithm keeps track of a remaining cost function rc_i which is initialized with the cost function *cost*, i.e., $\mathit{rc}_0 = \mathit{cost}$. In each step $i = 1, \ldots, n$, the cost function cost_i is set to a minimal cost function that satisfies $h_i(s, \mathit{rc}_{i-1}) = h_i(s, \mathit{cost}_i)$ for all s, called the *saturated cost function*. The remaining cost function is then reduced by cost_i, i.e., $\mathit{rc}_i = \mathit{rc}_{i-1} - \mathit{cost}_i$. In abstraction heuristics where the heuristic value is defined as the cheapest path in a transition system with transitions T, the saturated cost function is unique: $\mathit{cost}_i(o) = \max_{s \xrightarrow{o} t \in T}(h_i(s, \mathit{rc}_{i-1}) - h_i(t, \mathit{rc}_{i-1}))$. The saturated cost partitioning heuristic is defined as $h_{\mathcal{H},\mathcal{C}}(s)$ for the discovered cost partition $\mathcal{C} = \langle \mathit{cost}_1, \ldots, \mathit{cost}_n \rangle$.

Optimal cost partitioning (OCP). In contrast to suboptimal cost partitioning methods like SCP, OCP distributes the costs in an optimal way [Katz and Domshlak, 2010]. An OCP for a planning task Π with cost function *cost*, a state s, and admissible heuristics $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$ is a cost partition $\mathcal{C}^* = \langle \mathit{cost}_1, \ldots, \mathit{cost}_n \rangle$ such that $h_{\mathcal{H},\mathcal{C}^*}(s) \geq h_{\mathcal{H},\mathcal{C}}(s)$ for all cost partitions $\mathcal{C}$. The optimal cost partitioning heuristic is defined as $h_{\mathcal{H},\mathcal{C}^*}(s)$. Note that C^* depends on s, so the heuristic can use a different cost partition in each state. If the heuristics $\mathcal{H}$ are abstraction heuristics with induced abstract transition systems $\Theta_i = \langle S^i, L, T^i, s^i_{\mathrm{I}}, S^i_{\mathrm{G}} \rangle$, the value $h_{\mathcal{H},\mathcal{C}^*}(s)$ is the objective value of the following linear program (LP) or ∞ if the LP is unbounded:

$$\text{Maximize } \textstyle\sum_{i=1}^n \mathsf{H}^i_s \text{ subject to}$$

$$\mathsf{H}^i_{s_\star} \leq 0 \qquad \text{for all } s_\star \in S^i_{\mathrm{G}}, 1 \leq i \leq n \quad (1)$$

$$\mathsf{H}^i_s \leq \mathsf{H}^i_t + \mathsf{C}^i_\ell \qquad \text{for all } s \xrightarrow{\ell} t \in T^i, 1 \leq i \leq n \quad (2)$$

$$\sum_{i=1}^n \mathsf{C}^i_\ell \leq \mathit{cost}(\ell) \qquad \text{for all } \ell \in L \quad (3)$$

Example 2. *Consider the FTS of Figure 1 to be the induced FTS $F(\Pi)$ of a planning task Π. (Actually, it cannot be, but for simplicity we ignore this subtlety.) The factors are the atomic factors for Π, which are abstractions of Π where each label corresponds to an operator. The corresponding abstraction heuristics $\mathcal{H} = \langle h_1, h_2, h_3 \rangle$ are defined as the shortest path in each abstraction.*

To compute the the SCP heuristic of $\mathcal{H}$ with $\omega = \mathcal{H}$, we first set $\mathit{rc}_0(\ell_i) = 1$ for $1 \leq i \leq 3$. The saturated costs are $\mathit{cost}_1(\ell_1) = \mathit{cost}_1(\ell_2) = 1$ and $\mathit{cost}_1(\ell_3) = 0$, and therefore $\mathit{rc}_1(\ell_1) = \mathit{rc}_1(\ell_2) = 0$ and $\mathit{rc}_1(\ell_3) = 1$. In the second iteration, we get saturated costs $\mathit{cost}_2(\ell_1) = \mathit{cost}_2(\ell_2) = 0$ and $\mathit{cost}_2(\ell_3) = 1$. All remaining costs are 0 now and $\mathit{cost}_3(\ell_i) = 0$ for $1 \leq i \leq 3$. Thus the heuristic value of the SCP heuristic is $h_1(s_0, \mathit{cost}_1) + h_2(s_0, \mathit{cost}_2) + h_3(s_0, \mathit{cost}_3) = 2 + 1 + 0 = 3$. If we consider the order $\omega = \langle h_2, h_1, h_3 \rangle$ instead, h_2 receives all costs and the heuristic value becomes $h_1(s_0, \mathit{cost}_1) + h_2(s_0, \mathit{cost}_2) + h_3(s_0, \mathit{cost}_3) = 0 + 2 + 0 = 2$

An OCP can be computed by solving the LP above. Note that the LP does not exclude negative costs and they are useful here. An optimal solution is $\mathit{cost}_1(\ell_i) = 0$ for $1 \leq i \leq 3$; $\mathit{cost}_2(\ell_1) = \mathit{cost}_2(\ell_2) = 0$, $\mathit{cost}_2(\ell_3) = 3$; and $\mathit{cost}_3(\ell_1) = \mathit{cost}_3(\ell_2) = 1$, $\mathit{cost}_3(\ell_3) = -2$. The heuristic value of the OCP heuristic is $h_1(s_0, \mathit{cost}_1) + h_2(s_0, \mathit{cost}_2) + h_3(s_0, \mathit{cost}_3) = 0 + 3 + 2 = 5$.

In the example, we used the fact that there is a one-to-one correspondence between the labels of an induced FTS $F(\Pi)$ and the operators of Π. However, after performing M&S transformations (in particular label reduction), we lose this correspondence. In the following, we introduce a more general form of cost partitioning, which we then use to discuss cost partitioning on an arbitrary FTS.

3 Label Cost Partitioning

Operator cost partitioning partitions operator cost functions, so it is not directly applicable to factored transition systems with arbitrary labels. However, we can easily lift the notion to general (and in particular to factored) transition systems.

Definition 1. *Let $cost : L \to \mathbb{R}_0^+$ be a label cost function for a set of labels L. The cost functions $\mathcal{C} = \langle cost_1, \dots, cost_n \rangle$ with $cost_i : L \to \mathbb{R}$ form a* label cost partition *if they satisfy $\sum_{i=1}^n cost_i \leq cost$.*

Definition 2. *Let $F = \langle \Theta_1, \dots, \Theta_n \rangle$ be an FTS with labels L and a label cost function $cost$. Let $\mathcal{C} = \langle cost_1, \dots, cost_n \rangle$ be a label cost partition. The* label-cost-partitioned heuristic *is $h_{F,\mathcal{C}}(s) = \sum_{i=1}^n h_{\Theta_i}(s, cost_i)$.*

Proposition 1. *Let Θ be a transition system and let $F = \langle \Theta_1, \dots, \Theta_n \rangle$ and $\mathcal{C}$ be defined as above. If every h_{Θ_i} is admissible for Θ, then $h_{F,\mathcal{C}}$ is an admissible heuristic for Θ.*

Proof. The proof is analogous to the one for operator cost partitioning [Pommerening *et al.*, 2015]. □

As a special case, we can conclude that any label cost partition over the factors of an FTS F created by M&S induces an admissible heuristic: we know that the factor heuristics h_{Θ_i} are admissible heuristics for the product system of F, which in turn is an abstraction of the original planning task.

As with other cost-partitioned heuristics, there are several methods to compute the partition $\mathcal{C}$. We consider optimal and saturated cost partitioning here and define $h_F^{\text{OCP}}(s)$ as $h_{F,\mathcal{C}^*}(s)$ for a label cost partition C^* that is optimal for s, and $h_{F,\omega}^{\text{SCP}}(s)$ as $h_{F,\mathcal{C}}(s)$ for a saturated label cost partition $\mathcal{C}$ for s under order ω. Both heuristics are computed analogously to their counterparts for planning tasks. In particular, h_F^{OCP} is defined by the LP from Section 2.3 except that L is the set of labels of F. We denote this LP by $\text{OCP}(F)$.

We also consider a more general form of label cost partitioning, where factors of different FTSs are used. We use this to combine factors from different stages of the M&S computation. The main issue with this is that the factors may use different labels. However, for M&S each factor comes from a transformed FTS of a common base FTS (e.g., $F(\Pi)$). In such cases, we can treat the combination as a label cost partitioning of the base FTS and translate labels accordingly.

Definition 3. *Let L be a set of labels and $cost : L \to \mathbb{R}_0^+$ be a label cost function. For $1 \leq i \leq n$, let L_i be a different set of labels and $\lambda_i : L \to L_i$ a label mapping function.*

Then the functions $\mathcal{C} = \langle cost_1, \dots, cost_n \rangle$ with $cost_i : L_i \to \mathbb{R}$ form an extended label cost partition *if $\sum_{i=1}^n cost_i(\lambda_i(\ell)) \leq cost(\ell)$ holds for all $\ell \in L$.*

Definition 4. *Let F be an FTS with labels L and a label cost function $cost : L \to \mathbb{R}_0^+$. For $1 \leq i \leq n$ let F_i be an FTS that results from F with an M&S transformation using state mapping $\Sigma_i : S(F) \to S(F_i)$ and label mapping $\lambda_i : L \to L_i$, and let Θ_i be a factor of F_i. Finally, let $\mathcal{C} = \langle cost_1, \dots, cost_n \rangle$ be an extended label cost partition according to Definition 3. Then the label-cost-partitioned heuristic for $\mathcal{F} = \langle \Theta_1, \dots, \Theta_n \rangle$ is $h_{F,\mathcal{F},\mathcal{C}}^{\text{M\&S}}(s) = \sum_{i=1}^n h_{\Theta_i}(\Sigma_i(s), cost_i)$.*

The benefit of an extended label cost partitioning over a regular one is that it allows us to derive admissible heuristics from factors of different FTSs (even if they do not use the same labels).

Proposition 2. *The heuristic $h_{F,\mathcal{F},\mathcal{C}}^{\text{M\&S}}$ is admissible for $\bigotimes F$.*

Proof. The functions $cost_i'(\ell) = cost_i(\lambda_i(\ell))$ form a label cost partition for F. For each transition system Θ_i in $\mathcal{F}$, consider the transition system Θ_i' that results from Θ_i by replacing every transition $s \xrightarrow{\ell} t$ with $s \xrightarrow{\ell'} t$ for some $\ell' \in \lambda_i^{-1}(\ell)$ with minimal $cost_i'(\ell')$. It is easy to see that $h_{\Theta_i'}(s, cost_i') \leq h_{\Theta_i}(s, cost_i)$ for all states s. Since h_{Θ_i} is admissible for $\bigotimes F_i$, which is in turn an abstraction of $\bigotimes F$, $h_{\Theta_i'}$ must be admissible for $\bigotimes F$ under the cost function $cost_i'$. Using Proposition 1, the sum of the heuristics is admissible. □

Analogous to the cases above, we can define OCP and SCP for a collection of factors $\mathcal{F}$. An OCP can be computed by replacing constraint (3) with the constraint from Definition 3 in the LP from Section 2.3. For SCP, we keep track of the remaining cost of every label in L in $rc : L \to \mathbb{R}$. We compute the saturated cost function $cost_i : L_i \to \mathbb{R}$ under the cost function $rc_{L_i} : L_i \to \mathbb{R}$ with $rc_{L_i}(\ell_i) = \min_{\ell \in \lambda^{-1}(\ell_i)} rc_i(\ell)$ and update $rc_i(\ell) = rc_{i-1}(\ell) - cost_i(\lambda(\ell))$ for all $\ell \in L$ to guarantee that the $cost_i$ form an extended label cost partition.

4 Interaction of OCP with M&S

In this section, we show how OCP interacts with M&S transformations. In particular, we want to know if the OCP over an FTS before applying a specific M&S transformation achieves a higher or lower value compared to computing it over the transformed FTS.

4.1 Label Reduction

We first show that the value of h^{OCP} cannot increase after label reduction but can decrease in general. We then show that in the special case of exact label reduction, it cannot decrease.

Proposition 3. *Let $F = \langle \Theta_1, \dots, \Theta_n \rangle$ be an FTS with labels L and let F' the FTS that results from F with a label reduction $\lambda : L \to L'$. Given a solution H', C' of $\text{OCP}(F')$, we can construct a solution H, C of $\text{OCP}(F)$ with the same value.*

Proof. The solution is defined as $\mathsf{H} = \mathsf{H}'$ and $\mathsf{C}_\ell^i = \mathsf{C}'^i_{\lambda(\ell)}$ for all $1 \leq i \leq n$ and $\ell \in L$. With these values, constraints (1)–(3) of $\text{OCP}(F)$ are satisfied and since $\mathsf{H} = \mathsf{H}'$, the value of the solution is the same. □

We conclude that label reduction cannot improve h^{OCP}.

Theorem 1. *Let F and F' be FTSs such that F' results from F with label reduction. Then $h_F^{\text{OCP}} \geq h_{F'}^{\text{OCP}}$.*

In general, this inequality can be strict, i.e., the heuristic value can decrease because of label reduction. For example, we have seen in Example 2 that the OCP heuristic value of the factors in Figure 1 is 5. Mapping all labels in that example to a single label ℓ would mean that the cost of this label has to be split between the factors for a total heuristic value of 2.

However, we can show that the heuristic value of h^{OCP} is not influenced if the label reduction is exact.

Proposition 4. *Let F be an FTS with labels L and let F' be the FTS that results from F with an exact label reduction $\lambda : L \to L'$. Given a solution H, C of $\mathrm{OCP}(F)$, we can construct a solution H', C' of $\mathrm{OCP}(F')$ with the same value.*

Proof. We write $[\ell]$ for the equivalence class $\{\hat{\ell} \in L \mid \lambda(\ell) = \lambda(\hat{\ell})\}$ and first show that we can construct an alternative solution $\hat{\mathsf{H}}, \hat{\mathsf{C}}$ for $\mathrm{OCP}(F)$ where all labels in an equivalence class have the same cost. We define $\hat{\mathsf{H}} = \mathsf{H}$ and $\hat{\mathsf{C}}^i_\ell$ as $\min_{\hat{\ell} \in [\ell]} \mathsf{C}^i_{\hat{\ell}}$ in cases where all labels in $[\ell]$ have parallel transitions in Θ_i. There can only be one factor Θ_i where the labels do not necessarily have parallel transitions as the label reduction is exact. In this factor, we set $\hat{\mathsf{C}}^i_\ell = \mathit{cost}(\ell) - \sum_{j \neq i} \hat{\mathsf{C}}^j_\ell$, i.e., all remaining costs for this label. Labels in one equivalence class have the same cost in all factors of the former case and since the same amount of the cost remains also in the latter case. Constraints (1)–(3) of $\mathrm{OCP}(F)$ are still satisfied and since $\hat{\mathsf{H}} = \mathsf{H}$, the value of the solution is the same. Based on $\hat{\mathsf{H}}, \hat{\mathsf{C}}$, we can define the solution of $\mathrm{OCP}(F')$ as $\mathsf{H}' = \hat{\mathsf{H}}$ and $\mathsf{C}'^i_{\ell'} = \hat{\mathsf{C}}^i_\ell$ for any label ℓ mapped to ℓ'. The choice of ℓ does not matter as $\hat{\mathsf{C}}^i_\ell$ has the same value for all choices. □

By combining Propositions 3 and 4, we conclude that h^{OCP} is invariant under exact label reduction.

Theorem 2. *Let F and F' be FTSs such that F' results from F with exact label reduction. Then $h^{\mathrm{OCP}}_F = h^{\mathrm{OCP}}_{F'}$.*

4.2 Shrinking

We first show that the heuristic value of h^{OCP} cannot increase after shrinking but can decrease in general and even with h-preserving shrinking. We then show that the value does not decrease in the special case of bisimulation shrinking.

Proposition 5. *Let $F = \langle \Theta_1, \ldots, \Theta_n \rangle$ be an FTS with states S and let $F' = \langle \alpha(\Theta_1), \Theta_2, \ldots, \Theta_n \rangle$ be the FTS F after shrinking a factor (w.l.o.g. Θ_1) with an abstraction α. Given a solution H', C' of $\mathrm{OCP}(F')$, we can construct a solution H, C of $\mathrm{OCP}(F)$ with the same value.*

Proof. The solution is defined as $\mathsf{C} = \mathsf{C}'$ and for all $s \in S$, $\mathsf{H}^1_s = \mathsf{H}'^1_{\alpha(s)}$ and $\mathsf{H}^i = \mathsf{H}'^i$ for all $2 \leq i \leq n$. Constraints (1) and (2) remain satisfied, even for the first factor, because $\alpha(\Theta_1)$ is an abstraction of Θ_1 and for each concrete transition or goal state the corresponding abstract transition or goal state shows the constraint is satisfied. Constraint (3) is satisfied since $\mathsf{C} = \mathsf{C}'$ and the value of the solution is the same. □

We conclude that shrinking cannot improve h^{OCP}.

Theorem 3. *Let F and F' be FTSs such that F' results from F with shrinking. Then $h^{\mathrm{OCP}}_F \geq h^{\mathrm{OCP}}_{F'}$.*

In general, this inequality can be strict, i.e., the heuristic value can decrease because of shrinking. This is even the case if we consider only h-preserving shrinking. To illustrate this, consider the factors depicted in Figure 3 and the FTSs $F = \langle \Theta_1, \Theta_2 \rangle$ and $F' = \langle \Theta'_1, \Theta_2 \rangle$, where Θ'_1 results from combining the two gray states with identical heuristic value in Θ_1. The OCP for F is $\mathit{cost}_1 = \langle 1, 2, 1, -1 \rangle$ and $\mathit{cost}_2 = \langle 0, -1, 0, 2 \rangle$ with heuristic value 4, and $\mathit{cost}'_1 = \langle 0, 0, 0, 0 \rangle$ and $\mathit{cost}'_2 = \langle 1, 1, 1, 1 \rangle$ for F' with heuristic value 3.

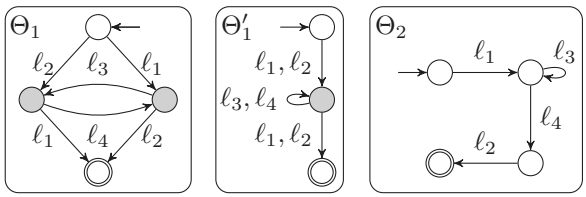

Figure 3: Interaction of h-preserving shrinking and OCP.

OCP does not lose information with the stronger restriction to bisimulation shrinking, so the heuristic value is unaffected.

Proposition 6. *Let $F = \langle \Theta_1, \ldots, \Theta_n \rangle$ be an FTS and let $F' = \langle \Theta'_1, \Theta_2, \ldots, \Theta_n \rangle$ be the FTS F after shrinking a factor (w.l.o.g. Θ_1) with a bisimulation α. Given a solution H, C of $\mathrm{OCP}(F)$ we can construct a solution H', C' of $\mathrm{OCP}(F')$ with at least the same value.*

Proof sketch. Instead of H, C, we consider a solution $\hat{\mathsf{H}}, \mathsf{C}$, where $\hat{\mathsf{H}}^i = \mathsf{H}^i$ for all $i \neq 1$ but $\hat{\mathsf{H}}^1_s$ is the optimal plan cost of s in Θ_1 under the costs encoded in C^1 for all states s of Θ_1. This is the intuitive meaning of the variables H but the LP can admit solutions with lower values. It can be checked that $\hat{\mathsf{H}}, \mathsf{C}$ remains a solution and has at least the same value as H, C. For two bisimilar states $s \sim t$, we then have $\hat{\mathsf{H}}^1_s = \hat{\mathsf{H}}^1_t$ because they have identical plans. We define the desired solution to be like $\hat{\mathsf{H}}, \mathsf{C}$ in all components except H'^1, where we set $\mathsf{H}'^1_s = \hat{\mathsf{H}}^1_{s'}$ for any state s' mapped to s. As bisimulation shrinking can only combine bisimilar states, H'^1 is well-defined. Constraints for goal states and transitions in Θ'_1 are validated by the constraints for the goal states and transitions from Θ_1 that induce them. □

Note that the proof uses a property that is similar to h-preserving shrinking but the difference is that bisimulation preserves h values under *any* cost function, in particular under the cost function encoded in C^1, while h-preserving shrinking only considers h-values under the original cost function.

By combining Propositions 5 and 6, we conclude that h^{OCP} is invariant under bisimulation shrinking.

Theorem 4. *Let F and F' be FTSs such that F' results from F with bisimulation shrinking. Then $h^{\mathrm{OCP}}_F = h^{\mathrm{OCP}}_{F'}$.*

4.3 Merging

The previous two sections showed transformations that can potentially decrease the value of h^{OCP}. In contrast to these results, we show in this section that the heuristic value of h^{OCP} can only increase after merging.

Proposition 7. *Let $F = \langle \Theta_1, \Theta_2, \ldots, \Theta_n \rangle$ be an FTS and let $F' = \langle \Theta_m, \Theta_3, \ldots, \Theta_n \rangle$ be the FTS where two factors (w.l.o.g. Θ_1 and Θ_2) have been replaced with their synchronized product $\Theta_m = \Theta_1 \otimes \Theta_2$. Given a solution H, C of $\mathrm{OCP}(F)$, we can construct a solution H', C' of $\mathrm{OCP}(F')$ with the same value.*

Proof. We construct the solution as $\mathsf{H}'^m_{\langle s_1, s_2 \rangle} = \mathsf{H}^1_{s_1} + \mathsf{H}^2_{s_2}$ and $\mathsf{C}'^m_\ell = \mathsf{C}^1_\ell + \mathsf{C}^2_\ell$, leaving all other components of H, C

the same. The new solution forms a cost partition for F' and has the same objective value. Constraints (1) and (2) for Θ_m are validated for a goal state or transition by summing the constraints for the goal states or transitions inducing them in the synchronized product. □

From Proposition 7 we can conclude that merging can only be beneficial for optimal cost partitioning.

Theorem 5. *Let F and F' be FTSs such that F' results from F with merging two factors. Then $h^{\text{OCP}}_F \leq h^{\text{OCP}}_{F'}$.*

5 Interaction of SCP with M&S

The results on the interaction of OCP and M&S do not allow to draw conclusions for other cost partitioning techniques. In this section, we analyze how applying M&S transformations affects h^{SCP}. To avoid that a difference in the heuristic value is only caused by a different order, we compare compatible orders before and after a M&S transformation.

5.1 Label Reduction

In this section, we consider an FTS $F = \langle \Theta_1, \ldots, \Theta_n \rangle$ with labels L and the FTS $F' = \langle \Theta'_1, \ldots, \Theta'_n \rangle$ with labels L' that results from applying label reduction $\lambda : L \to L'$ to F. As all Θ'_i in F' are derived from Θ_i via λ, we compare h^{SCP} before and after label reduction with orders ω and ω', where ω' can be derived from ω by replacing each Θ_i with Θ'_i.

We first show that the value of h^{SCP} can decrease or increase after label reduction. Thereafter, we show that exact label reduction does not affect the value of h^{SCP}.

Theorem 6. *Given F, F', ω and ω' as described above, $h^{\text{SCP}}_{F,\omega}$ and $h^{\text{SCP}}_{F',\omega'}$ are incomparable.*

Proof. We provide examples for all three cases. If $\lambda(\ell) = \ell$ for all $\ell \in L$, we have $F = F'$ and $\omega = \omega'$ and hence $h^{\text{SCP}}_{F,\omega} = h^{\text{SCP}}_{F',\omega'}$. Now consider an FTS $F = \langle \Theta_2, \Theta_5, \Theta_3 \rangle$ with factors as depicted in Figure 4 and order $\omega = F$. We have $h^{\text{SCP}}_{F,\omega}(s_I) = 2$. If all labels are mapped to the same label, we get $h^{\text{SCP}}_{F',\omega'}(s_I) = 1$, and if only ℓ_2 and ℓ_3 are mapped to the same label, we get $h^{\text{SCP}}_{F',\omega'}(s_I) = 3$. □

Theorem 7. *Given F, F', ω and ω' as described above. If λ is an exact label reduction, then $h^{\text{SCP}}_{F,\omega} = h^{\text{SCP}}_{F',\omega'}$.*

Proof sketch. As the label reduction is exact, we know that all combined labels are Θ_k-combinable. We prove the following statements by an induction over the order ω:

$$h_{\Theta'_i}(s, rc'_{i-1}) = h_{\Theta_i}(s, rc_{i-1})$$

$$rc'_i(\ell') = \begin{cases} rc_i(\ell) & \text{if } i < k \\ \min_{\ell \in \lambda^{-1}(\ell')} rc_k(\ell) & \text{otherwise} \end{cases}$$

$$cost'_i(\ell') = \begin{cases} cost_i(\ell) & \text{if } i \neq k \\ \max_{\ell \in \lambda^{-1}(\ell')} cost_i(\ell) & \text{otherwise} \end{cases}$$

for all $1 \leq i \leq n$, all states s, all $\ell' \in L'$, and all $\ell \in \lambda^{-1}(\ell')$. The theorem then follows from the first statement. The full proof can be found in a technical report [Sievers *et al.*, 2020b]. □

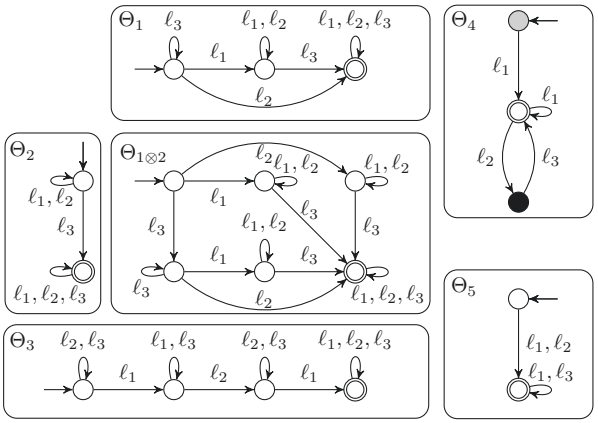

Figure 4: Factors used in Section 5.

5.2 Shrinking

Consider two FTSs $F = \langle \Theta_1, \ldots \Theta_n \rangle$ and $F' = \langle \alpha(\Theta_1), \ldots \Theta_n \rangle$ where w.l.o.g. Θ_1 of F is shrunk with abstraction α. The order ω' for F' can be derived from the order ω for F by replacing Θ_1 with $\alpha(\Theta_1)$.

Theorem 8. *Given F, F', ω and ω' as described above, $h^{\text{SCP}}_{F,\omega}$ and $h^{\text{SCP}}_{F',\omega'}$ are incomparable.*

Proof. We provide examples for all three cases. If $\alpha = \text{id}$ and hence $\alpha(\Theta_1) = \Theta_1$, we have $\omega = \omega'$ and $h^{\text{SCP}}_{F,\omega} = h^{\text{SCP}}_{F',\omega'}$. Now consider an FTS $F = \langle \Theta_4, \Theta_2 \rangle$ with factors as depicted in Figure 4 and the order $\omega = F$. We have $h^{\text{SCP}}_{F,\omega}(s_I) = 1$. If Θ_4 is shrunk by combining the black and white states, we get $h^{\text{SCP}}_{F',\omega'}(s_I) = 2$, but if Θ_4 is shrunk by combining the gray and white states, we get $h^{\text{SCP}}_{F',\omega'}(s_I) = 0$. □

Theorem 9. *Given F, F', ω and ω' as described above. If $\langle cost_1, \ldots, cost_n \rangle$ is the cost partition computed by $h^{\text{SCP}}_{F,\omega}$ and α is h-preserving under $cost_1$ then $h^{\text{SCP}}_{F,\omega} = h^{\text{SCP}}_{F',\omega'}$.*

Proof sketch. Saturated costs in Θ_i are computed solely with the h-values of states under rc_i, which are preserved by α for all $s \in S(\alpha(\Theta_1))$ according to the condition on α. □

5.3 Merging

Consider FTS $F = \langle \Theta_1, \ldots, \Theta_n \rangle$ and a merge transformation which w.l.o.g. replaces the factors Θ_1 and Θ_2 by their product $\Theta_{1\otimes 2}$ to obtain the FTS F'. There are two compatible orders: ω' replaces Θ_1 with $\Theta_{1\otimes 2}$ and removes Θ_2, while ω'' replaces Θ_2 with $\Theta_{1\otimes 2}$ and removes Θ_1.

Theorem 10. *Given F, F', ω and ω' as described above, $h^{\text{SCP}}_{F,\omega}$, $h^{\text{SCP}}_{F',\omega'}$ and $h^{\text{SCP}}_{F',\omega''}$ are incomparable.*

Proof. By choosing ω so Θ_1 and Θ_2 appear consecutively, we have $\omega' = \omega''$ and it is sufficient to give an example for all three possible cases. If Θ_2 is a factor with a single (goal) state s, we have $h_{\Theta_2}(s, cost_2) = 0$ and $\Theta_{1\otimes 2} = \Theta_1$ and hence $h^{\text{SCP}}_{F,\omega} = h^{\text{SCP}}_{F',\omega'}$.

Now consider an FTS $F = \langle \Theta_1, \Theta_2, \Theta_3 \rangle$ with factors as depicted in Figure 4 and the order $\omega = F$. We have

Algorithm 1 Merge-and-shrink algorithm extended to compute SCP heuristics.

Input: FTS F, transformation strategies, size limit, time limit, order strategy
Output: Heuristic for F

```
1: function M&SWITHSCP(F)
2:     F' ← F
3:     H ← ∅
4:     while not TERMINATE(⟨F', Σ, λ⟩) do
5:         i, j ← MERGESTRATEGY(F')
6:         F' ← LABELREDUCTIONSTRATEGY(F')
7:         H ← H ∪ COMPUTESNAPSHOT(F')
8:         Θ'_i, Θ'_j ← SHRINKSTRATEGY(F', i, j)
9:         F' ← (F' \ {Θ_i, Θ_j}) ∪ {Θ'_i, Θ'_j}
10:        F' ← (F' \ {Θ'_i, Θ'_j}) ∪ {Θ'_i ⊗ Θ'_j}
11:    end while
12:    return max_{h^SCP ∈ H} h^SCP
13: end function
```

$h^{\text{SCP}}_{F,\omega}(s_I) = 3$. If Θ_1 and Θ_2 are merged to $\Theta_{1\otimes 2}$, we get $h^{\text{SCP}}_{F',\omega'}(s_I) = 2$. If we consider an FTS $F\langle\Theta_1, \Theta_2\rangle$ instead, we have $h^{\text{SCP}}_{F,\omega}(s_I) = 1$ but $h^{\text{SCP}}_{F',\omega'}(s_I) = 2$ after merging. □

6 Implementation

We now describe our practical approach of integrating label cost partitioning into the M&S algorithm. We focus on SCPs because computing OCPs over large abstractions is often infeasible in practice, but our approach can be used with OCPs in the same way.

Recall that at each step of the M&S algorithm on some FTS F, the factors Θ_i of the current FTS F' induce the factor heuristics h_{Θ_i} for F. The first and most simple approach simply takes the final factor heuristics of the M&S computation and, instead of computing the maximum over them like $h^{\text{M\&S}}$ does, combines them into an SCP heuristic (denoted $h^{\text{SCP}}_{\mathit{final}}$). However, if there is enough time, the M&S algorithm only terminates when there is only one factor left, in which case neither maximizing nor cost partitioning is beneficial.

Therefore, we instead suggest to compute SCP heuristics over the factor heuristics of any intermediate FTS F' and to compute the maximum over these SCP heuristics to form the final heuristic. This approach requires to decide *when* and *how often* to compute such SCP heuristics. Concerning *how often*, for simplicity, we decided to take a single *snapshot* of each ith iteration of M&S. That is, in each ith iteration, we compute a single SCP heuristic over the factor heuristics induced by the current FTS. We experimentally evaluate different values for i. Regarding the choice of *when* in a given iteration this should be done, our theoretical analysis showed that each transformation can have a positive or negative impact, so we consider computing a snapshot after label reduction, after shrinking, or after merging.

Algorithm 1 shows the M&S algorithm extended to compute SCP heuristics. Compared to regular M&S, it additionally stores the computed SCP heuristics in a set H (line 7). Depending on when to compute the snapshot, line 7 is moved after line 9 (shrinking) or after line 10 (merging). At the end (line 12), instead of computing $h^{\text{M\&S}}$ over the final F', it computes the maximum heuristic over all SCP heuristics in H, which we denote by $h^{\text{SCP}}_{\mathit{int}}$ for this *interleaved* approach of computing M&S and SCPs. The parameter *order strategy* specifies in which order to compute SCP heuristics and we explore different choices in the experiments.

A potential drawback of the interleaved approach is that it misses the opportunity to combine factor heuristics of different M&S iterations with extended label cost partitions. We therefore also suggest an *offline* approach, which, rather than computing an SCP during an iteration, only stores the factor heuristics of that iteration for later use. We exclude factors which have not changed since the last snapshot to avoid duplicates. At the end, we need to use the stored factor heuristics to compute a heuristic. We consider two alternatives: computing a single SCP heuristic, denoted $h^{\text{SCP}}_{\mathit{off}}$, and computing the maximum heuristic over several SCP heuristics computed for diverse orders obtained with the algorithm by Seipp *et al.* [2020], denoted $h^{\text{SCP}}_{\mathit{off\text{-}div}}$. This makes the offline approach more comparable to the interleaved approach that also maximizes over several SCP heuristics. A potential disadvantage of the offline approach is that it needs to store the transition system of each factor heuristic until it computed the SCPs.

Finally, we remark that for all SCP heuristics, we only store useful (non-constant zero) heuristics as suggested by Seipp *et al.* [2020]. Furthermore, for each factor heuristic used in an SCP, we need to copy the relevant state mapping because further M&S transformations possibly modify it. However, M&S stores state mappings in a tree-like data structure called *factored state mappings* [Sievers, 2018] which is expensive to copy. Fortunately, only their root node is subject to potential change through M&S transformations. We exploit this by computing shallow copies, reusing subtrees which are guaranteed to not change anymore. We experimentally compare this to the alternative of making full copies.

7 Experiments

We implemented our techniques on top of the M&S implementation of Fast Downward [Helmert, 2006], version 19.12. To evaluate them, we use the following state-of-the-art M&S configuration: shrinking is based on bisimulation and uses a size limit of 50000 on transition systems (allowing exact shrinking also when shrinking is not necessary to respect the size limit); the merge strategy is SCC-DFP [Sievers *et al.*, 2016]; we use exact label reduction; and the main loop is limited to 900s. As benchmarks, we use the tasks of all optimal tracks of all International Planning Competitions, a set consisting of 1827 tasks across 65 domains. Experiments were run on Intel Xeon Silver 4114 CPUs, using Downward Lab [Seipp *et al.*, 2017b]. Each planner run is limited to 1800s and 3.5 GiB. The code, benchmarks, and experimental data are published online [Sievers *et al.*, 2020a].

We use the following order strategies for computing SCPs: random (*rnd*); creation time of the factors (*otn* for old to new); the inverse of otn (*nto*); the greedy orders of Seipp *et al.* [2020] called maximize heuristic (*mh*), minimize stolen cost (*msc*), and maximize heuristic per stolen cost (*mhsc*). Greedy orders are computed for the initial state.

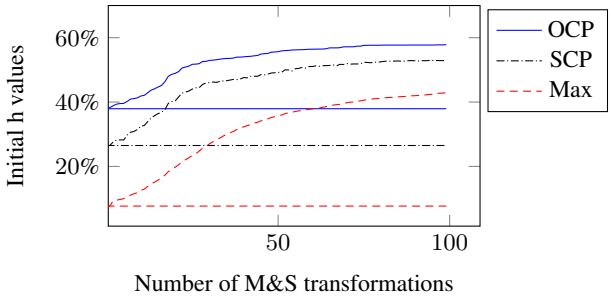

Figure 5: Geometric mean over initial heuristic values of the factor heuristics when combined with maximization (Max), SCP with a random order, or OCP. All values are normalized with h^* and plotted over the first 100 M&S transformations.

	after label reduction			
	$i=1$	$i=2$	$i=5$	$i=10$
full	922	923	923	916
shallow	**933**	930	926	917

$i=1$, shallow	
after label r.	**933**
after shrinking	**933**
after merging	925

Table 1: Coverage of $h^{\text{SCP}}_{\textit{int, rnd}}$ computing an SCP every ith iteration after different M&S transformations using shallow or full copies of the factored state mappings.

First, we analyze how much better compared to $h^{\text{M\&S}}$ we can get. Figure 5 reports the development of initial h-values of the factor heuristics after each M&S transformation, when combined with maximization (Max), SCP with a random order, or OCP. In this plot, we only consider the 833 tasks where M&S terminated (including computing the cost partitions). As expected, the combination with cost partitioning dominates the combination with maximization, but SCP is reasonably close to OCP, which justifies its higher practical relevance compared to the more expensive OCP.

Second, we compare $h^{\text{M\&S}}$ against $h^{\text{SCP}}_{\textit{final}}$. There are 162 tasks for which M&S computes *more* than one factor heuristic because it reaches the time limit. Of these tasks, $h^{\text{M\&S}}$ solves 50 compared to 56 tasks solved by $h^{\text{SCP}}_{\textit{final}}$ regardless of the used order. This is the result we expected: the combination with SCP is usually stronger than the maximization used by $h^{\text{M\&S}}$, but since these tasks are the most difficult ones (indicated by M&S reaching the time limit), the difference is small.

Next, we investigate the interleaved SCP approach by varying the parameter i. We also compare *full* against *shallow* copies of factored states mappings. The left part of Table 1 shows coverage (number of solved tasks) of $h^{\text{SCP}}_{\textit{int}}$ with a random order ($h^{\text{SCP}}_{\textit{int, rnd}}$) when computing snapshots after label reduction. Compared to $h^{\text{M\&S}}$, which achieves a coverage of 905, we can see that all variants improve coverage, even when computing a snapshot each iteration ($i=1$). While with *full* copying, there is no visible difference for $i=1,2,5$, using the memory-efficient *shallow* copying significantly improves coverage of those variants. For $i=10$, performance is similar for *full* and *shallow* copies, which can be explained by a lower heuristic quality compared to smaller values of i due to computing fewer SCP heuristics. We use the best configuration $i=1$ and *shallow* copies for the remaining evaluation.

	rnd	otn	nto	mhsc	mh	msc
$h^{\text{SCP}}_{\textit{int}}$	**933**	932	928	928	927	930
$h^{\text{SCP}}_{\textit{off}}$	841	836	905	869	905	837
$h^{\text{SCP}}_{\textit{off-div}}$	915	841	904	887	907	838

Table 2: Coverage with interleaved or offline computation using different order strategies.

We continue with evaluating when to compute snapshots: after label reduction, shrinking or merging. The right part of Table 1 shows how this affects the coverage of $h^{\text{SCP}}_{\textit{int, rnd}}$. We observe no difference between the first two choices but a clear benefit over the third. This matches our theoretical results: the value of an SCP heuristic is not affected by exact label reduction and h-preserving shrinking, whereas merging can affect the SCP value arbitrarily. We compute the snapshots after label reduction for the rest of the evaluation.

Finally, we evaluate different order strategies. The first row of Table 2 shows coverage of $h^{\text{SCP}}_{\textit{int}}$. Somewhat surprisingly, the random order (*rnd*) performs best. A possible reason is that diversification of orders with interleaved SCPs is obtained through computing SCPs over different factor heuristics in each iteration. Another reason might be that the two basic orders we introduced (*otn* and *nto*) as well as the greedy orders by Seipp *et al.* [2020] (*mhsc*, *mh*, *msc*) are not well suited for our setting.

The latter hypothesis is backed by the evaluation of the offline approach under the same setting as the interleaved approach ($i=1$, shallow copies, snapshots after label reduction). The last two rows of Table 2 show coverage of $h^{\text{SCP}}_{\textit{off}}$ and $h^{\text{SCP}}_{\textit{off-div}}$. Interestingly, even computing a single SCP heuristic ($h^{\text{SCP}}_{\textit{off}}$) with orders *nto* and *mh* already achieves the same coverage as plain $h^{\text{M\&S}}$. Computing diverse orders ($h^{\text{SCP}}_{\textit{off-div}}$) strictly improves coverage compared to $h^{\text{SCP}}_{\textit{off}}$, but only marginally for all order strategies except *rnd*, which is the best strategy like for the interleaved approach. Compared to $h^{\text{SCP}}_{\textit{int}}$, $h^{\text{SCP}}_{\textit{off-div}}$ is strictly dominated in terms of total coverage. Again, the most likely explanation is that the order strategies are not strong enough to help the order diversification algorithm to find better orders compared to the diversification implicitly obtained by the interleaved approach.

8 Conclusions

In this paper, we contribute the first combination of cost partitioning with M&S heuristics. Our theoretical analysis investigates the interaction of OCP and SCP with M&S transformations, and our practical implementation significantly improves regular M&S heuristics. In future work, we want to investigate better order strategies for SCPs in the context of M&S and to formalize cost partitioning as a transformation of the M&S framework.

Acknowledgements

We have received funding for this work from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 817639).

References

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

[Dräger *et al.*, 2009] Klaus Dräger, Bernd Finkbeiner, and Andreas Podelski. Directed model checking with distance-preserving abstractions. *International Journal on Software Tools for Technology Transfer*, 11(1):27–37, 2009.

[Edelkamp, 2001] Stefan Edelkamp. Planning with pattern databases. In Amedeo Cesta and Daniel Borrajo, editors, *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, pages 84–90. AAAI Press, 2001.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Helmert *et al.*, 2014] Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM*, 61(3):16:1–63, 2014.

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

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

[Nissim *et al.*, 2011] Raz Nissim, Jörg Hoffmann, and Malte Helmert. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 1983–1990. AAAI Press, 2011.

[Pearl, 1984] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[Pommerening *et al.*, 2015] Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 3335–3341. AAAI Press, 2015.

[Seipp and Helmert, 2018] Jendrik Seipp and Malte Helmert. Counterexample-guided Cartesian abstraction refinement for classical planning. *Journal of Artificial Intelligence Research*, 62:535–577, 2018.

[Seipp *et al.*, 2017a] Jendrik Seipp, Thomas Keller, and Malte Helmert. A comparison of cost partitioning algorithms for optimal classical planning. In Laura Barbulescu, Jeremy Frank, Mausam, and Stephen F. Smith, editors, *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, pages 259–268. AAAI Press, 2017.

[Seipp *et al.*, 2017b] Jendrik Seipp, Florian Pommerening, Silvan Sievers, and Malte Helmert. Downward Lab. https://doi.org/10.5281/zenodo.790461, 2017.

[Seipp *et al.*, 2020] Jendrik Seipp, Thomas Keller, and Malte Helmert. Saturated cost partitioning for optimal classical planning. *Journal of Artificial Intelligence Research*, 67:129–167, 2020.

[Sievers *et al.*, 2014] Silvan Sievers, Martin Wehrle, and Malte Helmert. Generalized label reduction for merge-and-shrink heuristics. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, pages 2358–2366. AAAI Press, 2014.

[Sievers *et al.*, 2016] Silvan Sievers, Martin Wehrle, and Malte Helmert. An analysis of merge strategies for merge-and-shrink heuristics. In Amanda Coles, Andrew Coles, Stefan Edelkamp, Daniele Magazzeni, and Scott Sanner, editors, *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, pages 294–298. AAAI Press, 2016.

[Sievers *et al.*, 2020a] Silvan Sievers, Florian Pommerening, Thomas Keller, and Malte Helmert. Code, benchmarks and experiment data for the IJCAI 2020 paper "Cost-Partitioned Merge-and-Shrink Heuristics for Optimal Classical Planning". https://doi.org/10.5281/zenodo.3775871, 2020.

[Sievers *et al.*, 2020b] Silvan Sievers, Florian Pommerening, Thomas Keller, and Malte Helmert. Cost-partitioned merge-and-shrink heuristics for optimal classical planning: Technical report. Technical Report CS-2020-001, University of Basel, Department of Mathematics and Computer Science, 2020.

[Sievers, 2018] Silvan Sievers. Merge-and-shrink heuristics for classical planning: Efficient implementation and partial abstractions. In Vadim Bulitko and Sabine Storandt, editors, *Proceedings of the 11th Annual Symposium on Combinatorial Search (SoCS 2018)*, pages 90–98. AAAI Press, 2018.