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\textsuperscript{+} formalism [Bäckström and Nebel, 1995]. Such a task is defined as \( \Pi = (V, O, \text{cost}, s_0, s_f) \). \( V \) is a finite set of state variables \( v \), each with a finite domain \( \text{dom}(v) \). A partial state \( s \) is an assignment over \( \text{vars}(s) \subseteq V \), mapping each \( v \) in \( \text{vars}(s) \) to a value \( \ell \in \text{dom}(v) \). It is a state if \( \text{vars}(s) = V \). State \( s \) satisfies partial state \( s' \), written \( s \models s' \), if \( s(v) = s'(v) \) for all \( v \) in \( \text{vars}(s') \). \( O \) is a finite set of operators \( o = (\text{pre}_o, \text{eff}_o) \), where \( \text{pre}_o \) and \( \text{eff}_o \) are partial states called the precondition and effect of \( o \). The cost function cost : \( O \rightarrow \mathbb{R}_0^+ \) maps each operator to its cost. Finally, \( s_0 \) is the initial state, and \( s_f \) a partial state called the goal.

For a state \( s \) and operator \( o \) with \( s \models \text{pre}_o \) the successor state \( s[O|o] \) is defined as \( s[O|o](v) = \text{eff}_o(v) \) for all \( v \) in \( \text{vars}(\text{eff}_o) \) and \( s[O|o](v) = s(v) \) for all other variables. A planning task \( \Pi \) induces a transition system \( \Theta(\Pi) = (S, L, \text{cost}, s_0, S_G) \), where \( S \) is the set of states over \( V \); \( L = O \) are labels with costs \( \text{cost}(\ell) = \text{cost}(o) \) for \( \ell \in L \); \( T \) are the transitions \( \{ s \xrightarrow{t} s, t \in S, \ell \in L, s \models \text{pre}_\ell, t = s[O|\ell] \} \); \( s_0 = s_0 \); and \( S_G = \{ s \in S \ | \ s \models s_0 \} \) are the goal states. For a transition system \( \Theta \), 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 \( s_0 \) 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 \( A^\ast \) algorithm with admissible heuristics. A heuristic for a transition system \( \Theta \) with labels \( L \) and cost function cost is a function \( h : S(\Theta) \rightarrow \mathbb{R} \cup \{\infty\} \) which estimates the cost of reaching a goal state from \( s \) in \( \Theta \). We write \( h(s, \text{cost}') \) for the evaluation of \( h \) on state \( s \in S(\Theta) \), however using an alternative cost function \( \text{cost}' \) instead of \( \text{cost} \). 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-
serves all plans but is typically smaller and easier to solve. Formally, an abstraction \( \alpha \) of a transition system \( \Theta = (S, L, cost, T, s_1, S_G) \) is a homomorphism defined on the states \( S\) of \( \Theta \). The induced abstract transition system is defined as \( \alpha(\Theta) = \{ (\alpha(s) | s \in S), L, cost, (\alpha(s) \xrightarrow{t} \alpha(t) | s \xrightarrow{t} T), \alpha(s_1), (\alpha(s) | s \in S_G) \} \).

M&S is applicable to transition systems which exhibit a so-called factored structure: a factored transition system (FTS) \( F = (\Theta_1, \ldots, \Theta_n) \) consists of transition systems \( \Theta_i \) (also called factors) sharing the same set of labels and the same cost function. It is a compact (“factored”) representation of its product system \( F = (S^\otimes, L, cost, T^\otimes, s_1^\otimes, S_G^\otimes) \), where \( S^\otimes = \prod_{i=1}^n S_i \) is the Cartesian product of the state sets; there is a transition \( (s^i, \ldots, s^n) \xrightarrow{t} (t^i, \ldots, t^n) \) in \( T^\otimes \) iff \( s^i \xrightarrow{t^i} t^i \) in \( T_i \) for all \( 1 \leq i \leq n \); the initial state is \( s_1^\otimes = (s_1, \ldots, s_n) \) and the set of goal states is \( S_G^\otimes = \prod_{i=1}^n S_{G_i} \). We write \( S(F) \) to denote the states of the product of \( F \).

A planning task \( \Pi = (\mathcal{V}, \mathcal{O}, cost, s_0, s_f) \) induces the FTS \( F(\Pi) \) consisting of the atomic factors \( \Theta_i \) for all variables \( v \in \mathcal{V} \): \( \Theta_v \) is the projection of \( \Pi \) to \( v \), i.e. \( \Theta_v = \{ dom(v), \mathcal{O}, cost, T^v, s_0(v), S_{G_i}^\otimes \} \). The set of transitions \( T^v \) consists of the transitions \( s \xrightarrow{t} t \) for the state \( s = pre_v[v] \) if \( v \in \mathcal{vars}(pre_v) \) and for all states \( s \in dom(v) \) otherwise, and for the state \( t \) with \( t = eff_v[v] \) if \( v \in \mathcal{vars}(eff_v) \) and \( t = s \) otherwise. If \( v \) is not mentioned in the goal of \( \Pi \), all states of \( \Theta_v \) are goal states \( (S_{G_i}^\otimes = dom(v)) \), otherwise there is a single goal state \( S_{G_i}^\otimes = (s_v) \).

Figure 1 shows an example FTS \( F = (\Theta_1, \Theta_2, \Theta_3) \). 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 \( F = (\Theta_1, \ldots, \Theta_n) \), M&S iteratively applies transformations of \( F \). To do so, it keeps track of the current FTS \( F' \), a state mapping \( \Sigma : S(F) \rightarrow S(F') \) from states of \( F \) to states of \( F' \), and a label mapping \( \lambda : L \rightarrow L' \) from labels of \( F \) to labels of \( F' \). A transformation from \( F \) to \( F' \) specifies \( F' \), the state mapping \( \Sigma \), and the label mapping \( \lambda \). Applying such a mapping 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. \( \Theta_1 \) and \( \Theta_2 \), by their product. Formally: \( F' = (\Theta_1 \otimes \Theta_2, \Theta_3, \ldots, \Theta_n), \Sigma((s_1, \ldots, s_n)) = ((s_1, s_2), s_3, \ldots, s_n) \) (where \( (s_1, s_2) \) is the product state in \( \Theta_1 \otimes \Theta_2 \) that results from the previous states \( s_1 \) of \( \Theta_1 \) and \( s_2 \) of \( \Theta_2 \), and \( \lambda = \text{id} \)).

**Shrinking.** Shrinking applies an abstraction \( \alpha \) to one of the factors of \( F \), w.l.o.g. \( \Theta_1 \). Formally: \( F'' = (\alpha(\Theta_1), \Theta_2, \ldots, \Theta_n), \Sigma((s_1, \ldots, s_n)) = (\alpha(s_1), s_2, \ldots, s_n) \) and \( \lambda = \text{id} \). Shrinking is called \( h \)-preserving if \( h^*_\alpha(s) = h^*_\alpha(\alpha(s)) \) 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: \( \lambda : L \rightarrow L' \) is the label mapping of the reduction, \( \Sigma \) is the identity mapping, and \( F'' = (\Theta_1', \ldots, \Theta_n') \), where \( \Theta_i' \) is defined based on \( \Theta_i = (S^i, L, cost, T^i, s_1^i, S_{G_i}^\otimes) \) as follows: \( \Theta_i' = (S^i, L', \text{cost}', \{ s \xrightarrow{t} t' | s \xrightarrow{t} t \in T^i \}, s_1^i, S_{G_i}^\otimes) \) with \( \text{cost}'(t') = \min_{t' \in \text{cost}(t')} \text{cost}(t) \) for all \( t' \in L' \).

All three types of transformations guarantee that at any time of the M&S computation, each factor \( \Theta \) is an abstraction of the original product \( \otimes \).

**Example 1.** Consider the FTS \( F \) shown in Figure 1. In a first step, a label reduction transformation \( \lambda \) combines the labels \( \ell_1, \ell_2 \) into a new label \( \ell_4 \). The resulting transformed FTS \( F'' = (\Theta_1', \Theta_2', \Theta_3') \) is shown at the top of Figure 2.

In a second step, a shrink transformation applies an abstraction \( \alpha \) to \( \Theta_2' \) such that the two gray states are combined. The resulting FTS is \( F''' = (\Theta_1', \alpha(\Theta_2'), \Theta_3') \) 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 \( \Theta_1' \) and \( \alpha(\Theta_2') \) to \( \Theta_1'' \). The product is shown in the bottom right of Figure 2. The final FTS is \( F'''' = (\Theta_1'' \otimes \alpha(\Theta_3')), \Theta_3'' \).

In the context of planning, M&S is usually used to compute a heuristic. Let \( FTS = (\Theta_1, \ldots, \Theta_n) \) and state mapping \( \Sigma \) be the result obtained when executing M&S on some FTS. We define the factor heuristic \( h_\Theta \) for \( F \) as the perfect heuristic for the \( i \)-th factor of \( F' \): \( h_{\Theta_i}(s) = h^*_\Theta((\Sigma(s))) \). Then, the M&S heuristic for \( F \) is defined as the maximum over the factor heuristics: \( h_{\text{M&S}}^F(s) = \max_{1 \leq i \leq n} h_\Theta(s) \) for all \( s \in S(F) \). For the FTS \( F \) of Figure 1, we obtain \( h_{\text{M&S}}(s_1) = \max(h_{\text{M&S}}(s_1), h_{\text{M&S}}(s_1), h_{\text{M&S}}(s_1)) = \max(2, 2, 2) = 2 \). For the FTS \( F'''' \) from Example 1, we obtain \( h_{\text{M&S}}(s_1) = \max(3, 1) = 3 \).
As mentioned above, all transformations are such that the result is an abstraction. Therefore, $h^{\text{MKS}}$ 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 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 $s \xrightarrow{a} s'$, there exists $t \xrightarrow{a} t'$ with $s' \sim t'$. Shrinking is based on a bisimulation $\sim$ iff $\alpha(s) = \alpha(t)$ satisifes $s \sim t$. 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 $\lambda$ 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 $\text{cost}(\ell_1) = \text{cost}(\ell_2)$ and $\ell_1$ and $\ell_2$ are $\Theta$-combinable [Sieves et al., 2014]. Two labels $\ell_1$ and $\ell_2$ are called $\Theta$-combinable if they have parallel transitions in all factors except $\Theta$, i.e., if $s \xrightarrow{a} t \in T'$ iff $s \xrightarrow{\lambda a} t \in T' \forall \Theta' \in F \setminus \{\Theta\}$. Labels $\ell_1$ and $\ell_2$ combined in Example 1 are $\Theta_1$-combinable because they have parallel transitions in $\Theta_2$ and $\Theta_3$. 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 $\Theta_i, \Theta_j$ of $F'$ to merge in this iteration (line 5), apply exact label reduction (line 6), possibly shrink $\Theta_i$ and/or $\Theta_j$ to respect a size limit (lines 8 and 9), and finally perform the merge (line 10). The algorithm returns $h^{\text{MKS}}$ 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 $\Pi$ be a planning task with operator costs $\text{cost}$ and let $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$ be admissible heuristics for $\Pi$. The cost functions $\mathcal{C} = \langle \text{cost}_1, \ldots, \text{cost}_n \rangle$ form a cost partition if $\sum_{i=1}^n \text{cost}_i \leq \text{cost}$. The cost-partitioned heuristic $h_{\mathcal{H}, \mathcal{C}}(s) = \sum_{i=1}^n h_i(s, \text{cost}_i)$ 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 $\omega$. To simplify the presentation, we consider the order $\omega = \langle h_1, \ldots, h_n \rangle$ but in general, $\omega$ can be any permutation of the heuristics in $\mathcal{H}$. The algorithm keeps track of a remaining cost function $\text{rc}_i$ which is initialized with the cost function $\text{cost}_i$, i.e., $\text{rc}_0 = \text{cost}$. In each step $i = 1, \ldots, n$, the cost function $\text{rc}_{i-1}$ is set to a minimal cost function that satisfies $h_i(s, \text{rc}_{i-1}) = h_i(s, \text{cost}_i)$ for all $s$, called the saturated cost function. The remaining cost function is then reduced by $\text{cost}_i$, i.e., $\text{rc}_i = \text{rc}_{i-1} - \text{cost}_i$. 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: $\text{cost}_i(o) = \max_s \omega_s \in T (h_i(s, \text{rc}_{i-1}) - h_i(t, \text{rc}_{i-1}))$. The saturated cost partitioning heuristic is defined as $h_{\mathcal{H}, \mathcal{C}}(s)$ for the discovered cost partition $\mathcal{C} = \langle \text{cost}_1, \ldots, \text{cost}_n \rangle$.

**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 $\Pi$ with cost function $\text{cost}$, a state $s$, and admissible heuristics $\mathcal{H} = \langle h_1, \ldots, h_n \rangle$ is a cost partition $\mathcal{C}^* = \langle \text{cost}_1, \ldots, \text{cost}_n \rangle$ such that $h_{\mathcal{H}, \mathcal{C}^*}(s)$ is the objective value of the following linear program (LP) or if the LP is unbounded:

Maximize $\sum_{i=1}^n h_i^0 \text{subject to}$

- $h_i^0 \leq 0$ for all $i = 1, \ldots, n \quad (1)$
- $h_i^0 \leq \sum_{j \in L} C_{ij}^\ell + C_i^\ell$ for all $\ell \leq t \in T', 1 \leq i \leq n \quad (2)$
- $\sum_{i=1}^n C_i^\ell \leq \text{cost}(\ell)$ for all $\ell \in L \quad (3)$

**Example 2.** Consider the FTS of Figure 1 to be the induced FTS $\mathcal{F}'(\Pi)$ of a planning task $\Pi$. (Actually, it cannot be, but for simplicity we ignore this subtlety.) The factors are the atomic factors for $\Pi$, which are abstractions of $\Pi$ 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 $\text{rc}_0(\ell_i) = 1$ for $1 \leq i \leq 3$. The saturated costs are $\text{cost}_1(\ell_1) = \text{cost}_1(\ell_2) = 1$ and $\text{cost}_3(\ell_3) = 0$, and therefore $\text{rc}_1(\ell_i) = \text{rc}_1(\ell_2) = 0$ and $\text{rc}_2(\ell_i) = 1$. In the second iteration, we get saturated costs $\text{cost}_2(\ell_1) = \text{cost}_2(\ell_2) = 0$ and $\text{cost}_2(\ell_3) = 1$. All remaining costs are 0 now and cost $\text{cost}_3(\ell_2) = 0$ for $1 \leq i \leq 3$. Thus the heuristic value of the SCP heuristic is $h_1(s_0, \text{cost}_1) + h_2(s_0, \text{cost}_2) + h_3(s_0, \text{cost}_3) = 2 + 1 + 0 = 3$. If we consider the order $\omega = \langle h_3, h_1, h_3 \rangle$ instead, $h_2$ receives all costs and the heuristic value becomes $h_3(s_0, \text{cost}_1) + h_2(s_0, \text{cost}_2) + h_3(s_0, \text{cost}_3) = 0 + 2 + 0 = 2$

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 $\text{cost}_3(\ell_1) = 0$ for $1 \leq i \leq 3$; $\text{cost}_3(\ell_2) = \text{cost}_3(\ell_3) = 3$; and $\text{cost}_3(\ell_1) = \text{cost}_3(\ell_2) = 1$. Thus the heuristic value of the OCP heuristic is $h_1(s_0, \text{cost}_1) + h_2(s_0, \text{cost}_2) + h_3(s_0, \text{cost}_3) = 0 + 3 + 2 = 5$.

In the example, we used the fact that there is a one-to-one correspondence between the labels of an induced FTS $\mathcal{F}'(\Pi)$ and the operators of $\Pi$. 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 \( \text{cost} : L \rightarrow \mathbb{R}_+^n \) be a label cost function for a set of labels \( L \). The cost functions \( C = \langle \text{cost}_1, \ldots, \text{cost}_n \rangle \) with costi : \( L \rightarrow \mathbb{R} \) form a label cost partition if they satisfy \( \sum_{i=1}^n \text{cost}_i \leq \text{cost} \).

**Definition 2.** Let \( F = \langle \Theta_1, \ldots, \Theta_n \rangle \) be an FTS with labels \( L \) and a label cost function cost. Let \( C = \langle \text{cost}_1, \ldots, \text{cost}_n \rangle \) be a label cost partition. The label-cost-partitioned heuristic is \( h_{F,C}(s) = \sum_{i=1}^n h_{\Theta_i}(s, \text{cost}_i) \).

**Proposition 1.** Let \( \Theta \) be a transition system and let \( F = \langle \Theta_1, \ldots, \Theta_n \rangle \) and \( C \) be defined as above. If every \( h_{\Theta_i} \) is admissible for \( \Theta \), then \( h_{F,C} \) is an admissible heuristic for \( \Theta \).

**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 \( h_{\Theta_i} \) 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. We consider optimal and saturated cost partitioning here and define \( h^\text{SCP} \) and \( h^\text{FSCP} \) for all states \( s \in L \).

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

Then the functions \( C = \langle \text{cost}_1, \ldots, \text{cost}_n \rangle \) with costi : \( L_i \rightarrow \mathbb{R} \) form an extended label cost partition if \( \sum_{i=1}^n \text{cost}_i(\lambda_i(\ell)) \leq \text{cost}(\ell) \) holds for all \( \ell \in L \).

**Definition 4.** Let \( F \) be an FTS with labels \( L \) and a label cost function cost : \( L \rightarrow \mathbb{R}_+^n \). For \( 1 \leq i \leq n \), let \( F_i \) be an FTS that results from \( F \) with an M&S transformation using state mapping \( \Sigma_i : S(F) \rightarrow S(F_i) \) and label mapping \( \lambda_i : L \rightarrow L_i \), and let \( \Theta_i \) be a factor of \( F_i \). Finally, let \( C = \langle \text{cost}_1, \ldots, \text{cost}_n \rangle \) be an extended label cost partition according to Definition 3. Then the label-cost-partitioned heuristic for \( F = \langle \Theta_1, \ldots, \Theta_n \rangle \) is \( h_{F,F,C}(s) = \sum_{i=1}^n h_{\Theta_i}(\Sigma_i(s), \text{cost}_i) \).

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 \( h^\text{M&S}_{F,F,C} \) is admissible for \( \otimes F \).

**Proof.** The functions \( \text{cost}_i(\ell) = \text{cost}_i(\lambda_i(\ell)) \) form a label cost partition for \( F \). For each transition system \( \Theta_i \) in \( F \), consider the transition system \( \Theta_i' \) that results from \( \Theta_i \) by replacing every transition \( s \xrightarrow{\ell} t \) with \( s \xrightarrow{\ell'} t \) for some \( \ell' \in \lambda_i^{-1}(\ell) \) with minimal \( \text{cost}_i(\ell') \). It is easy to see that \( h_{\Theta_i'}(s, \text{cost}_i') \leq h_{\Theta_i}(s, \text{cost}_i) \) for all states \( s \). Since \( h_{\Theta_i} \) is admissible for \( \otimes F_i \), which is in turn an abstraction of \( \otimes F \), \( h_{\Theta_i} \) must be admissible for \( \otimes F \) under the cost function \( \text{cost}_i' \). 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 \( F \). 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 \( rc : L \rightarrow \mathbb{R} \). We compute the saturated cost function \( \text{cost}_i : L_i \rightarrow \mathbb{R} \) under the cost function \( rc_{L_i} : L_i \rightarrow \mathbb{R} \) with \( rc_{L_i}(\ell_i) = \min_{\ell \in \lambda_i^{-1}(\ell_i)} rc_{\ell}(\ell) \). We then define \( h^\text{SCP} \) and \( h^\text{SCP} \).

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 \( h^\text{OCP} \) 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 \( F = \langle \Theta_1, \ldots, \Theta_n \rangle \) be an FTS with labels \( L \) and let \( F' \) be the FTS that results from \( F \) with a label reduction \( \lambda : L \rightarrow L' \). Given a solution \( H', C' \) of OCP(F'), we can construct a solution \( H, C \) of OCP(F) with the same value.

**Proof.** The solution is defined as \( H = H' \) and \( C_i = C_i'(\lambda_i(\ell)) \) for all \( 1 \leq i \leq n \) and \( \ell \in L \). With these values, constraints (1)–(3) of OCP(F) are satisfied and since \( H = H' \), the value of the solution is the same.

We conclude that label reduction cannot improve \( h^\text{OCP} \).

**Theorem 1.** Let \( F \) and \( F' \) be FTSs such that \( F' \) results from \( F \) with label reduction. Then \( h^\text{OCP}_F \geq h^\text{OCP}_{F'} \).

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 \( \ell \) 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 \( h^\text{OCP} \) 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 $\lambda : L \rightarrow L'$. Given a solution $H, C$ of OCP($F$), we can construct a solution $H', C'$ of OCP($F'$) with the same value.

Proof. We write $[\ell]$ for the equivalence class $\{\ell \in L \mid \lambda(\ell) = \lambda(\ell')\}$ and first show that we can construct an alternative solution $H, \hat{C}$ for OCP($F$) where all labels in an equivalence class have the same cost. We define $\hat{H} = H$ and $\hat{C}_i = \min_{j \in [\ell]} C_j^i$ in cases where all labels in $[\ell]$ have parallel transitions in $\Theta_i$. There can only be one factor $\Theta_i$, where the labels do not necessarily have parallel transitions as the label reduction is exact. In this factor, we set $\hat{C}_i = \text{cost}(\ell) - \sum_{j \neq i} \hat{C}_j^i$, 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 OCP($F$) are still satisfied and since $H = \hat{H}$, the value of the solution is the same. Based on $H, \hat{C}$, we can define the solution of OCP($F'$) as $H' = H$ and $C'_i = \hat{C}_i^i$ for any label $\ell$ mapped to $\ell'$. The choice of $\ell$ does not matter as $\hat{C}_i^i$ has the same value for all choices.

By combining Propositions 3 and 4, we conclude that $h^{\text{OCP}}$ 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 $h^{\text{OCP}}_F = h^{\text{OCP}}_{F'}$.

4.2 Shrinking

We first show that the heuristic value of $h^{\text{OCP}}$ 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 $F = (\Theta_1, \ldots, \Theta_n)$ be an FTS with states $S$ and let $F' = (\alpha(\Theta_1), \Theta_2, \ldots, \Theta_n)$ be the FTS $F$ after shrinking a factor (w.l.o.g. $\Theta_1$) with an abstraction $\alpha$. Given a solution $H, C$ of OCP($F$), we can construct a solution $H', C'$ of OCP($F'$) with the same value.

Proof. The solution is defined as $C = C'$ and for all $s \in S$, $H_i^1 = H_i^{\alpha(s)}$ and $H_i^j = H_i^j$ 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 $\Theta_1$ 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 $C = C'$ and the value of the solution is the same.

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

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

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 $F = (\Theta_1, \Theta_2)$ and $F' = (\Theta_1, \Theta_2)$, where $\Theta_1$ results from combining the two gray states with identical heuristic value in $\Theta_1$. The OCP for $F$ is $\text{cost}_1 = \{1, 2, 1, -1\}$ and $\text{cost}_2 = (0, -1, 0, 2)$ with heuristic value 4, and $\text{cost}_1' = \{0, 0, 0, 0\}$ and $\text{cost}_2' = \{1, 1, 1, 1\}$ for $F'$ with heuristic value 3.

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

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

Proof sketch. Instead of $H, C$, we consider a solution $H, C$, where $H_i^j = H_i'$ for all $i \neq 1$ but $H_1$ is the optimal plan cost of $s$ in $\Theta_1$ under the costs encoded in $C^1$ for all states $s$ of $\Theta_1$. This is the intuitive meaning of the variables $H$ but the LP can admit solutions with lower values. It can be checked that $H, C$ remains a solution and has at least the same value as $H, C$. For two bisimilar states $s \sim t$, we then have $H_i^1 = H_i^1$ because they have identical plans. We define the desired solution to be like $H, C$ in all components except $H_1$, where we set $H_1^{\sim} = H_1^+$ for any state $s'$ mapped to $s$. As bisimulation shrinking can only combine bisimilar states, $H_1^{\sim}$ is well-defined. Constraints for goal states and transitions in $\Theta_1$ are validated by the constraints for the goal states and transitions from $\Theta_1$ 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 $h^{\text{OCP}}$ is invariant under bisimulation shrinking.

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

4.3 Merging

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

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

Proof. We construct the solution as $H_i^{\sim} = H_i^1 + H_i^2$ and $C_i^{\sim} = C_i^1 + C_i^2$, 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 \( \Theta_m \) 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 \( h_{\text{SCP}}^{\text{OCP}} \leq h_{\text{SCP}}^{\text{OCP}} \).

## 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 \( h_{\text{SCP}} \). 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 \( F = (\Theta_1, \ldots, \Theta_n) \) with labels \( L \) and the FTS \( F' = (\Theta_1', \ldots, \Theta_n') \) with labels \( L' \) that results from applying label reduction \( \lambda : L \to L' \) to \( F \). As all \( \Theta_i' \) in \( F' \) are derived from \( \Theta_i \) via \( \lambda \), we compare \( h_{\text{SCP}} \) before and after label reduction with orders \( \omega \) and \( \omega' \), where \( \omega' \) can be derived from \( \omega \) by replacing each \( \Theta_i \) with \( \Theta_i' \).

We first show that the value of \( h_{\text{SCP}} \) can decrease or increase after label reduction. Therefore, we show that exact label reduction does not affect the value of \( h_{\text{SCP}} \).

**Theorem 6.** Given \( F, F', \omega \) and \( \omega' \) as described above, \( h_{F,\omega}^{\text{SCP}} \) and \( h_{F',\omega'}^{\text{SCP}} \) 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 \( h_{F,\omega}^{\text{SCP}} = h_{F',\omega'}^{\text{SCP}} \). Now consider an FTS \( F = (\Theta_1, \Theta_2, \Theta_3) \) with factors as depicted in Figure 4 and the order \( \omega = F \). We have \( h_{F,\omega}^{\text{SCP}}(s_1) = 2 \) if all labels are mapped to the same label, we get \( h_{F,\omega}^{\text{SCP}}(s_1) = 1 \), and if only \( \ell_2 \) and \( \ell_3 \) are mapped to the same label, we get \( h_{F,\omega}^{\text{SCP}}(s_1) = 3 \).

**Theorem 7.** Given \( F, F', \omega \) and \( \omega' \) as described above. If \( \lambda \) is an exact label reduction, then \( h_{F',\omega'}^{\text{SCP}} = h_{F,\omega}^{\text{SCP}} \).

**Proof sketch.** As the label reduction is exact, we know that all combined labels are \( \Theta_k \)-combinable. We prove the following statements by an induction over the order \( \Theta \):

\[
\begin{align*}
    h_{\Theta_i}(s, r_{i-1}) &= h_{\Theta_i}(s, r_{i-1}) \\
    r'_{i}(\ell') &= \begin{cases} r_{i}(\ell) & \text{if } i < k \\
                        \min_{\ell \in \lambda^{-1}(\ell')} r_{k}(\ell) & \text{otherwise} \end{cases} \\
    \text{cost}'_{i}(\ell') &= \begin{cases} \text{cost}_{i}(\ell) & \text{if } i \neq k \\
                        \max_{\ell \in \lambda^{-1}(\ell')} \text{cost}_{k}(\ell) & \text{otherwise} \end{cases}
\end{align*}
\]

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].

### 5.2 Shrinking

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

**Theorem 8.** Given \( F, F', \omega \) and \( \omega' \) as described above, \( h_{F,\omega}^{\text{SCP}} \) and \( h_{F',\omega'}^{\text{SCP}} \) 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 \( h_{F,\omega}^{\text{SCP}} = h_{F',\omega'}^{\text{SCP}} \). Now consider an FTS \( F = (\Theta_1, \Theta_2) \) with factors as depicted in Figure 4 and the order \( \omega = F \). We have \( h_{F,\omega}^{\text{SCP}}(s_1) = 1 \). If \( \Theta_3 \) is shrunk by combining the black and white states, we get \( h_{F',\omega'}^{\text{SCP}}(s_1) = 2 \), but if \( \Theta_3 \) is shrunk by combining the gray and white states, we get \( h_{F',\omega'}^{\text{SCP}}(s_1) = 0 \).

**Theorem 9.** Given \( F, F', \omega \) and \( \omega' \) as described above. If \( \{\text{cost}_1, \ldots, \text{cost}_n\} \) is the cost partition computed by \( h_{F,\omega}^{\text{SCP}} \) and \( \alpha \) is \( h \)-preserving under \( \text{cost}_i \) then \( h_{F,\omega}^{\text{SCP}} \geq h_{F',\omega'}^{\text{SCP}} \).

**Proof sketch.** Saturated costs in \( \Theta_i \) are computed solely with the \( h \)-values of states under \( \text{cost}_i \), which are preserved by \( \alpha \) for all \( s \in S(\alpha(\Theta_1)) \).

### 5.3 Merging

Consider FTS \( F = (\Theta_1, \ldots, \Theta_n) \) and a merge transformation which w.l.o.g. replaces the factors \( \Theta_1 \) and \( \Theta_2 \) by their product \( \Theta_{1 \odot 2} \) to obtain the FTS \( F' \). There are two compatible orders: \( \omega' \) replaces \( \Theta_1 \) with \( \Theta_{1 \odot 2} \) and removes \( \Theta_2 \), while \( \omega'' \) replaces \( \Theta_2 \) with \( \Theta_{1 \odot 2} \) and removes \( \Theta_1 \).

**Theorem 10.** Given \( F, F', \omega \) and \( \omega' \) as described above, \( h_{F,\omega}^{\text{SCP}}, h_{F',\omega'}^{\text{SCP}} \), and \( h_{F',\omega''}^{\text{SCP}} \) are incomparable.

**Proof.** By choosing \( \omega \) so \( \Theta_1 \) and \( \Theta_2 \) appear consecutively, we have \( \omega' = \omega'' \) and it is sufficient to give an example for all three possible cases. If \( \Theta_2 \) is a factor with a single (goal) state \( s \), we have \( h_{\Theta_2}(s, \text{cost}_2) = 0 \) and \( \Theta_{1 \odot 2} = \Theta_1 \) and hence \( h_{F,\omega}^{\text{SCP}} = h_{F',\omega'}^{\text{SCP}} \).

Now consider an FTS \( F = (\Theta_1, \Theta_2, \Theta_3) \) 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 \( \text{M&SwithSCP}(F) \)
2: \( F' \leftarrow F \)
3: \( H \leftarrow \emptyset \)
4: while not \( \text{Terminate}(\langle F', \Sigma, \lambda \rangle) \) do
5: \( i, j \leftarrow \text{MergeStrategy}(F') \)
6: \( F' \leftarrow \text{LabelReductionStrategy}(F') \)
7: \( H \leftarrow H \cup \text{ComputeSnapshot}(F') \)
8: \( \Theta'_i, \Theta'_j \leftarrow \text{ShrinkStrategy}(F', i, j) \)
9: \( F' \leftarrow (F' \setminus \{\Theta'_i, \Theta'_j\}) \cup \{\Theta'_i \odot \Theta'_j\} \)
10: \( F' \leftarrow (F' \setminus \{\Theta'_i, \Theta'_j\}) \cup \{\Theta'_i \odot \Theta'_j\} \)
11: end while
12: return \( \max_{h \in H} h_{\text{SCP}} \)
13: end function

\( h_{\text{SCP}}(s_1) = 3 \). If \( \Theta_1 \) and \( \Theta_2 \) are merged to \( \Theta\odot\Theta \), we get \( h_{\text{SCP}}(s_1) = 2 \). If we consider an FTS \( F'\langle \Theta_1, \Theta_2 \rangle \) instead, we have \( h_{\text{SCP}}(s_1) = 1 \) but \( h_{\text{SCP}}(s_1) = 2 \) 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 \( \Theta \) of the current FTS \( F' \) induce the factor heuristics \( h_{\Theta} \) 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 \( h_{\text{SCP}} \)). 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 \( i \)th iteration of M&S. That is, in each \( i \)th 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 \( i \) 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 \( h_{\text{SCP}} \) 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 \( h_{\text{off}} \), and computing the maximum heuristic over several SCP heuristics computed for diverse orders obtained with the algorithm by Seipp et al. [2020], denoted \( h_{\text{off-msc}} \). 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 factorized 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 an 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 \( \text{rnd} \); creation time of the factors \( \text{otn} \) for old to new; the inverse of \( \text{otn} \) \( \text{nito} \); the greedy orders of Seipp et al. [2020] called maximize heuristic \( \text{nmc} \), minimize stolen cost \( \text{msc} \), and maximize heuristic per stolen cost \( \text{mhsc} \). Greedy orders are computed for the initial state.
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 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.

![Initial h values](image)

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.

<table>
<thead>
<tr>
<th>$i$</th>
<th>full</th>
<th>shallow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>922</td>
<td>933</td>
</tr>
<tr>
<td>2</td>
<td>923</td>
<td>930</td>
</tr>
<tr>
<td>5</td>
<td>923</td>
<td>926</td>
</tr>
<tr>
<td>10</td>
<td>916</td>
<td>917</td>
</tr>
</tbody>
</table>

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

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 $h^{\text{SCP}}_{\text{int}}$ with a random order ($h^{\text{SCP}}_{\text{int, rnd}}$) 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.

### 8 Conclusions

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 $h^{\text{SCP}}_{\text{int, rnd}}$. 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 $h^{\text{SCP}}_{\text{int}}$. 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 $h^{\text{SCP}}_{\text{off}}$ and $h^{\text{SCP}}_{\text{off-div}}$. Interestingly, even computing a single SCP heuristic ($h^{\text{SCP}}_{\text{off}}$) with orders nto and mh already achieves the same coverage as plain $h^{\text{M&S}}$. Computing diverse orders ($h^{\text{SCP}}_{\text{off-div}}$) strictly improves coverage compared to $h^{\text{SCP}}_{\text{off}}$, but only marginally for all order strategies except rnd, which is the best strategy like for the interleaved approach. Compared to $h^{\text{SCP}}_{\text{int}}$, $h^{\text{SCP}}_{\text{off-div}}$ 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.

### 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


