ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240618

Merge-and-Shrink Heuristics for SSPs with Prune Transformations

Thorsten Klößner[a], Álvaro Torralba[b], Marcel Steinmetz[c] and Silvan Sievers[d]

[a]Saarland University, Germany
[b]Aalborg University, Denmark
[c]LAAS-CNRS, Université de Toulouse, France
[d]University of Basel, Switzerland

Abstract. The merge-and-shrink framework is a powerful tool for constructing state-of-the-art admissible heuristics in classical planning. Recent work has begun generalizing the complex theory behind this framework to probabilistic planning in forms of stochastic shortest-path problems (SSPs). There however remain two important gaps. Firstly, although the previous work makes substantial efforts, the probabilistic merge-and-shrink theory is still incomplete, lacking in particular prune transformations, i.e., transformations discarding uninteresting states, effectively reducing the size of the abstraction without losing relevant information. Secondly, an actual implementation and experimental evaluation of the merge-and-shrink framework for SSPs is so far missing. Here, we round off the previous work by contributing both a theoretical analysis of prune transformations, as well as an empirical evaluation of merge-and-shrink heuristics. Our results show that merge-and-shrink heuristics outperform previous single abstraction heuristics, but do not quite reach the performance of state-of-the-art additive combinations of such heuristics yet.

1 Introduction

Fully observable probabilistic planning is commonly viewed as a stochastic shortest-path problem (SSP) [2], where one aims at finding a policy that guarantees achieving the goal with the lowest expected cost possible. The current state of the art for solving SSPs optimally is using heuristic search algorithms [e.g., 7] guided with admissible probabilistic-abstraction heuristics [32, 15, 14]. Such heuristics derive lower bounds on the expected cost-to-goal by abstracting the SSP, collapsing together states so to make an exhaustive analysis feasible. In recent years, many different methods have been proposed to generate such abstractions automatically, the roots typically going back to classical planning, including pattern databases [5, 8, 14, 16] and counterexample-guided Cartesian abstractions [24, 17].

Merge-and-shrink abstractions have received significant attention in classical planning research [9, 10, 11, 26]. Merge-and-shrink is a generic framework for constructing highly informative abstractions, which is based on a factored representation of transition systems and operations on this representation, called *transformations* [26]. Merge-and-shrink constructs admissible abstraction heuristics, as long as only transformations are used which preserve specific compositional properties, regardless of their application order.

Recently, this framework has been generalised to probabilistic planning [18], by defining a suitable factored representation for probabilistic transition systems alongside suitable extensions of the core transformations: *merge*, *shrink*, and *label reduction* transformations. Klößner et al. [18] showed that these transformations preserve compositional properties needed to guarantee correctness of the abstraction construction. However, two gaps remain open. On the one hand, their theory did not cover *prune* transformations, which remove uninteresting (e.g., unreachable) states from the abstraction, effectively reducing the size of the abstraction without harming informativeness of the resulting abstraction heuristic. On the other hand, they provide no implementation and no empirical evaluation, so the practical applicability of this type of abstraction remains unknown.

In this paper, we round off the previous work. We formally introduce prune transformations, which in general no longer preserve the admissibility guarantee of the constructed heuristic. Nevertheless, we derive sufficient conditions for admissibility, and furthermore show that it suffices to preserve admissibility on *alive* states, (i.e., states part of some solution), in order for standard SSP heuristic search algorithms to retain their completeness and optimality guarantees. We derive prune transformations that detect non-alive states, strengthening the heuristic, while preserving the optimality guarantee of the heuristic search.

Moreover, we implemented the merge-and-shrink framework, including merge, shrink, label reduction and prune transformations. We compare merge-and-shrink heuristics against other state-of-the-art abstraction heuristics for probabilistic planning. Our experiments show that this framework can derive informative heuristics, competitive with both PDB heuristics and Cartesian abstraction heuristics.

The paper is structured as follows. We first give an overview over our probabilistic planning setting and the probabilistic merge-and-shrink framework in Section 2. In Section 3, we re-consider the heuristic properties needed to ensure optimality of SSP heuristic search algorithms, and introduce relaxed properties that allow for the pruning of uninteresting states. In Section 4, we define prune transformations, embed them into the compositional theory of merge-and-shrink, and develop practical strategies for pruning. We conclude with an experimental evaluation of our approaches in Section 5.

2 Background

A partial function from X to Y, written $f : X \rightharpoonup Y$, is associated with its domain $dom(f) \subseteq X$. If $dom(f) = X$, we write $f : X \to Y$. The set of stochastic functions over X is given by $\mathit{Dist}(X) :=$

$\{\delta : X \to [0,1] \mid \sum_{x \in X} \delta(x) = 1\}$. Likewise, the set of sub-stochastic functions over X is given by $\mathit{SDist}(X) := \{\delta : X \to [0,1] \mid \sum_{x \in X} \delta(x) \leq 1\}$. The support of $\delta \in \mathit{SDist}(X)$ is defined by $\mathit{supp}(\delta) := \{x \mid \delta(x) > 0\}$.

2.1 Probabilistic Transition Systems

A *probabilistic transition system* (abbreviated as TS) is a tuple $\Theta = \langle S_\Theta, L_\Theta, C_\Theta, T_\Theta, I_\Theta, G_\Theta \rangle$ consisting of a set of *states* S_Θ, a set of *action labels* L_Θ, a *label cost function* $C_\Theta : L_\Theta \to \mathbb{R}_0^+$, a set of *probabilistic transitions* $T_\Theta \subseteq T_\Theta^{all} := S_\Theta \times L_\Theta \times \mathit{Dist}(S_\Theta)$, a set of *initial states* $I_\Theta \subseteq S_\Theta$ and a set of *goal states* $G_\Theta \subseteq S_\Theta$. All sets are finite. For a state $s \in S_\Theta$, we define $T_\Theta(s) := \{\langle s, \ell, \delta \rangle \in T_\Theta\}$. To ease notation, we define $C_\Theta(\mathfrak{T}) := C_\Theta(\ell)$ for the cost and $\delta_\mathfrak{T} := \delta$ for the successor distribution of a transition $\mathfrak{T} = \langle s, \ell, \delta \rangle \in T_\Theta$.

A finite path of Θ is a finite alternating sequence $p = s_0 \mathfrak{T}_0 \ldots s_n$ of states $s_0, \ldots, s_n \in S_\Theta$ and transitions $\mathfrak{T}_0, \ldots, \mathfrak{T}_{n-1} \in T_\Theta$. We define $\mathit{last}(p) := s_n$ as the last state and $\mathsf{Cost}(p) := \sum_{i=0}^{n-1} C_\Theta(\mathfrak{T}_i)$ as the cost of such a finite path p. The set of finite paths of Θ is denoted by $\mathit{FPaths}(\Theta)$. Analogously, infinite paths are infinite alternating sequences $p = s_0 \mathfrak{T}_0 \ldots$ for which *last* is undefined and with their cost defined as $\mathsf{Cost}(p) := \sum_{i=0}^{\infty} C_\Theta(\mathfrak{T}_i)$.

Usually, *stationary and deterministic* (SD for short) policies $\pi : S_\Theta \rightharpoonup T_\Theta$ are used in the context of SSPs to model an agent who selects a transition $\pi(s)$ (if any) to execute next only based on the *current* state of the environment $s \in S_\Theta$. Unfortunately, such policies are not expressive enough for our developments, as we will see later. We consider *history-dependent* and *stochastic* policies $\pi : \mathit{FPaths}(\Theta) \to \mathit{SDist}(T_\Theta)$ with the following semantics. If $p \in \mathit{FPaths}(\Theta)$ is the execution history so far, π executes the transition $\mathfrak{T}$ next with probability $\pi(p)(\mathfrak{T})$, and terminates with probability $\mathit{term}_\pi(p) := 1 - \sum_{\mathfrak{T} \in T_\Theta} \pi(p)(\mathfrak{T})$. Only transitions of the current state may be chosen, i.e., $\mathit{supp}(\pi(p)) \subseteq T_\Theta(\mathit{last}(p))$. Given a starting state $s \in S_\Theta$, π induces a probability space over the finite and infinite paths (i.e., possible executions) of Θ [1]. The event space is the σ-algebra generated by all of the cylinder sets $\mathit{Cyl}(p) := \{p' \mid p \text{ is a prefix of } p'\}$, for $p \in \mathit{FPaths}(\Theta)$. The probability measure $\Pr_s^\pi$ over the paths is the unique extension of the pre-measure $\mathcal{P}[\mathit{Cyl}(s_0 \mathfrak{T}_0 \ldots s_n)] := [s_0 = s] \cdot \prod_{i=0}^{n-1} \pi(s_0 \mathfrak{T}_0 \ldots s_i)(\mathfrak{T}_i) \cdot \delta_{\mathfrak{T}_i}(s_{i+1})$ (defined using Iverson brackets) to the full σ-algebra.

Let $T \subseteq S_\Theta$ be a set of target states. Let $\mathit{Finish}_\Theta(T) := \{p \in \mathit{FPaths}(\Theta) \mid \mathit{last}(p) \in T\}$ be the set of paths of Θ ending in T. The *solutions* of the state $s \in S_\Theta$ for T are denoted $\mathit{Sols}_\Theta(s, T) := \{\pi \mid \Pr_s^\pi[\mathit{Finish}_\Theta(T)] = 1\}$. The *expected cost* of a policy π for a state s is denoted $J_\Theta^\pi(s) := \mathbb{E}_s^\pi[\mathsf{Cost}]$. The *optimal expected cost-to-goal* of s is given by $J_\Theta^*(s) := \inf_{\pi \in \mathit{Sols}_\Theta(s, G_\Theta)} J_\Theta^\pi(s)$. A solution $\pi \in \mathit{Sols}_\Theta(s, G_\Theta)$ is optimal for s if $J_\Theta^\pi(s) = J_\Theta^*(s)$. If a solution exists, there is also an optimal SD policy [2, 23]. The objective is finding an initial state s_0 and optimal policy for s_0, if existent, where $J_\Theta^*(s_0)$ is minimal among all initial states.

2.2 Heuristics & Transformations

A heuristic is a function $h : S_\Theta \to \mathbb{R}_0^+ \cup \{\infty\}$ that is used by a heuristic search algorithm to estimate J_Θ^*. h is *admissible* if $h(s) \leq J_\Theta^*(s)$ for every state $s \in S_\Theta$, *goal-aware* if $h(s) = 0$ for every goal state $s \in G_\Theta$, *consistent* if $h(s) \leq C_\Theta(\ell) + \sum_{t \in S_\Theta} \delta(t) \cdot h(t)$ for every transition $\langle s, \ell, \delta \rangle \in T_\Theta$ and finally *safe* if $h(s) \neq \infty$ for every state s such that $\mathit{Sols}_\Theta(s, G_\Theta) \neq \emptyset$. Every heuristic that is consistent, goal-aware, and safe is also admissible.

A TS transformation [18] is a tuple $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$, consisting of a *concrete TS* Θ, a *transformed TS* Θ', and the *state and label mappings* $\sigma : S_\Theta \rightharpoonup S_{\Theta'}$ and $\lambda : L_\Theta \rightharpoonup L_{\Theta'}$. We call states of Θ *concrete states* and states of Θ' *transformed states* (similarly for labels, etc.). The *transformation heuristic* induced by τ is given by $h^\tau(s) := J_{\Theta'}^*(\sigma(s))$ if $s \in \mathit{dom}(\sigma)$ and $h^\tau(s) := \infty$ otherwise.

Every function $f : X \rightharpoonup Y$ can be lifted to a function $\mathit{lift}[f] : \mathit{Dist}(X) \rightharpoonup \mathit{Dist}(Y)$ with the domain $\mathit{dom}(\mathit{lift}[f]) := \{\delta \mid \mathit{supp}(\delta) \subseteq \mathit{dom}(f)\}$ by defining $\mathit{lift}[f](\delta)(y) := \sum_{x \in f^{-1}(y)} \delta(x)$. By making use of this, the state mapping σ can be naturally lifted to a *state distribution* mapping $\mathit{lift}[\sigma] : \mathit{Dist}(S_\Theta) \rightharpoonup \mathit{Dist}(S_{\Theta'})$. Consulting the associations σ, λ and $\mathit{lift}[\sigma]$ and applying them componentwise yields the *transition transformation* $\mathit{ttr}_\tau : T_\Theta \rightharpoonup T_{\Theta'}^{all}$ with the domain $\mathit{dom}(\mathit{ttr}_\tau) := \mathit{dom}(\sigma) \times \mathit{dom}(\lambda) \times \mathit{dom}(\mathit{lift}[\sigma])$ given by $\mathit{ttr}_\tau(\langle s, \ell, \delta \rangle) := \langle \sigma(s), \lambda(\ell), \mathit{lift}[\sigma](\delta) \rangle$. Based on this, Klößner et al. [18] define the following *conservativeness* properties:

$\mathbf{CONS_S}$	σ is total on S_Θ
$\mathbf{CONS_L}$	λ is total on L_Θ
$\mathbf{CONS_C}$	$\forall \ell \in \mathit{dom}(\lambda).\, C_{\Theta'}(\lambda(\ell)) \leq C_\Theta(\ell)$
$\mathbf{CONS_T}$	$\mathit{ttr}_\tau(T_\Theta) \subseteq T_{\Theta'}$
$\mathbf{CONS_G}$	$\sigma(G_\Theta) \subseteq G_{\Theta'}$

They also introduce other properties, but we omit them here as we do not deal with them formally. We treat such transformation properties as sets, writing $\tau \in \mathbf{X}$ if τ satisfies the property $\mathbf{X}$. To ease notation, we also write $\tau \in \mathbf{CONS_{S+L}}$ instead of $\tau \in \mathbf{CONS_S} \cap \mathbf{CONS_L}$ for example. The properties $\mathbf{X}$ from above are invariant under composition, i.e., if two transformations $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ and $\tau' = \langle \Theta', \Theta'', \sigma', \lambda' \rangle$ both satisfy $\mathbf{X}$, then their composition $\tau' \circ \tau := \langle \Theta, \Theta'', \sigma' \circ \sigma, \lambda' \circ \lambda \rangle$ satisfies $\mathbf{X}$ as well. For every conservative transformation τ, i.e., $\tau \in \mathbf{CONS_{S+L+C+T+G}}$, h^τ is goal-aware, consistent and safe, and admissible.

2.3 The Merge-and-Shrink Framework

The probabilistic merge-and-shrink framework [18] is defined via transformations on a *factored* TS representation. The base of this representation are *annotated transition systems* (ATS), which are tuples $A = \langle S_A, L_A, C_A, E_A, D_A, T_A, I_A, G_A \rangle$. Here, E_A is a finite set of *effect labels* modelling different action outcomes, $D_A : L_A \to \mathit{Dist}(E_A)$ maps each action label to a probability distribution over effect labels and $T_A \subseteq S_A \times L_A \times (E_A \rightharpoonup S_A)$ consists of *annotated transitions* $\langle s, \ell, \alpha \rangle \in T_A$ associating each possible effect label $e \in \mathit{supp}(D_A(\ell))$ of ℓ with a successor $\alpha(e) \in S_A$. As opposed to a regular TS, an ATS is able to distinguish between different action outcomes leading to the same state. Dropping this information yields the ordinary TS $\Theta(A) := \langle S_A, L_A, C_A, T_{\Theta(A)}, I_A, G_A \rangle$ with $T_{\Theta(A)} := \{\langle s, \ell, \mathit{lift}[\alpha](D_A(\ell)) \rangle \mid \langle s, \ell, \alpha \rangle \in T_A\}$. For the transitions of $\Theta(A)$, the probability of a successor state is the total probability of all effect labels leading to this successor.

A *factored ATS* is a tuple $F = \langle A_i \rangle_{i \in \mathcal{I}}$ of ATSs (its *factors*) where $\mathcal{I}$ is some finite index set. All factors have the same action labels L_F, effect labels E_F, effect probabilities D_F and costs C_F. The factored ATS F implicitly represents the synchronised product of its factors, formally defined as $\bigotimes F := \langle \bigtimes_{i \in \mathcal{I}} S_{A_i}, L_F, C_F, E_F, D_F, T_{\bigotimes F}, \bigtimes_{i \in \mathcal{I}} I_{A_i}, \bigtimes_{i \in \mathcal{I}} G_{A_i} \rangle$, where $T_{\bigotimes F} := \{\langle \langle s_i \rangle_{i \in \mathcal{I}}, \ell, \alpha \rangle \mid \forall i \in \mathcal{I}.\, \langle s_i, \ell, \alpha_i \rangle \in T_{A_i} \wedge \alpha(e) = \langle \alpha_i(e) \rangle_{i \in \mathcal{I}}\}$. For a state $s \in S_{\bigotimes F}$ and a factor index $i \in \mathcal{I}$, $s(i) \in S_{A_i}$ denotes the state in the corresponding factor. Probabilistic planning tasks in finite-domain representation [32, 30] can be straightforwardly mapped to a factored ATS containing one factor for each finite-domain variable (the variable's atomic projection).

Since the underlying TS of an ATS is only implicitly represented, merge-and-shrink transformations are specified on the factored ATS directly. To this end, a *factored mapping* (FM) from F to a set D is a function $\sigma : S_{\bigotimes F} \rightharpoonup D$ that is either an *atomic FM* $\mathrm{Atom}_{i,\alpha}(s) := \alpha(s(i))$ for factor index $i \in \mathcal{I}$ and $\alpha : S_{A_i} \rightharpoonup D$, or a *merge FM* $\mathrm{Merge}_{\sigma_1,\sigma_2,\alpha}(s) := \alpha(\sigma_1(s), \sigma_2(s))$ for child FMs $\sigma_i : S_{\bigotimes F} \rightharpoonup D_i, i \in \{1, 2\}$ and $\alpha : D_1 \times D_2 \rightharpoonup D$. A *factored-to-factored* mapping from F to another factored ATS $F' = \langle A_j \rangle_{j \in J}$ is a tuple $\Sigma = \langle \sigma_j \rangle_{j \in J}$ of FMs $\sigma_j : S_{\bigotimes F} \rightharpoonup S_{A_j}$, representing the function $[\![\Sigma]\!] : S_{\bigotimes F} \to S_{\bigotimes F'}, [\![\Sigma]\!](s) := \langle \sigma_j(s) \rangle_{j \in J}$.

A *factored transformation* is a tuple $\langle F, F', \Sigma, \lambda \rangle$ where F is the original factored ATS, F' is the transformed factored ATS, Σ is a factored-to-factored mapping from F to F' and $\lambda : L_F \rightharpoonup L_{F'}$ is a label mapping. Such a factored transformation implicitly represents the TS transformation $\langle \Theta(\bigotimes F), \Theta(\bigotimes F'), [\![\Sigma]\!], \lambda \rangle$. Properties like **CONS$_\mathbf{S}$** are said to hold for a factored transformation, if they hold for this corresponding TS transformation. Merge-and-shrink iteratively applies factored transformations, terminating with a single factor left. Thanks to the compositional invariance, that final factor is guaranteed to yield a goal-aware, consistent, and admissible heuristic, if each individual transformation satisfied the properties from Section 2.2.

3 Revisiting SSP Heuristic Properties

In forward heuristic search, one will only ever encounter states *reachable* from the initial state. For the completeness and optimality properties of the search, the behaviour of the heuristic on unreachable states is hence irrelevant. Classical planning literature has exploited this observation through according relaxations of heuristic properties like admissibility, which allowed constructing higher quality heuristics [6, 26]. Here, we generalise these ideas to SSP heuristic search.

To be able to formulate these relaxed heuristic properties, we first define some additional concepts and notation, then follow up with an example that showcases the new definitions. In the following, let Θ be a TS, let π be a policy, and let $S \subseteq S_\Theta$ and $T \subseteq S_\Theta$ be a set of starting states and target states, respectively.

We write $s \overset{\pi}{\rightsquigarrow} t$ if there is a path $p = s \dots t$ with $\mathrm{Pr}^\pi_s[Cyl(p)] > 0$, and $s \rightsquigarrow t$ if there is a policy π with $s \overset{\pi}{\rightsquigarrow} t$. The states *forward reachable* from S and *backward reachable* from T under π are denoted $Reach^{\rightarrow}_{\Theta,\pi}(S) := \{t \mid \exists s \in S.\, s \overset{\pi}{\rightsquigarrow} t\}$ and $Reach^{\leftarrow}_{\Theta,\pi}(T) := \{s \mid \exists t \in T.\, s \overset{\pi}{\rightsquigarrow} t\}$ respectively. Likewise, the set of *all* forward respectively backward reachable states are denoted $Reach^{\rightarrow}_{\Theta}(S) := \{t \mid \exists s \in S.\, s \rightsquigarrow t\}$ and $Reach^{\leftarrow}_{\Theta}(T) := \{s \mid \exists t \in T.\, s \rightsquigarrow t\}$.

The set of dead ends for the targets T is given by $Dead_\Theta(T) := S_\Theta \setminus Reach^{\leftarrow}_{\Theta}(T)$, and denotes states which cannot reach T under any circumstance. A state s is *solvable* for T if it has a solution $\pi \in Sols_\Theta(s, T)$. The set of states solvable for T is denoted $Solv_\Theta(T)$. Lastly, we say that a state u is *alive* for S and T, if there is a state $s \in S$ and a solution $\pi \in Sols_\Theta(s, T)$ such that $s \overset{\pi}{\rightsquigarrow} u$. The set of all states alive for S and T is given by $Alive_\Theta(S, T)$. All alive states are forward reachable from S and solvable, as $\pi \in Sols_\Theta(s, T)$ and $s \overset{\pi}{\rightsquigarrow} u$ implies $\pi \in Sols_\Theta(u, T)$.

The sets $Reach^{\rightarrow}_{\Theta}(S)$ and $Reach^{\leftarrow}_{\Theta}(T)$ can be computed by a simple exhaustive forward/backwards exploration of Θ from S or T respectively. $Solv_\Theta(T)$ can be computed by repeatedly pruning dead ends until a fixpoint is reached [1]. More specifically, let the projection of Θ onto a state set $K \subseteq S_\Theta$ be $\Theta|_K := \{K, L_\Theta, T_{\Theta|_K}, C_\Theta, I_\Theta \cap K, G_\Theta \cap K\}$, where $T_{\Theta|_K} := \{\langle s, \ell, \delta|_K \rangle \mid \langle s, \ell, \delta \rangle \in T_\Theta \wedge s \in K \wedge supp(\delta) \subseteq K\}$. Starting with $\Theta_0 := \Theta$, one iteratively computes the TS $\Theta_{i+1} := \Theta_i|_{Reach^{\leftarrow}_{\Theta_i}(T)}$ until $\Theta_{i+1} = \Theta_i$. The solvable states are the remaining states of the final transition system Θ_k. Furthermore, we have $Alive_\Theta(S, T) = Reach^{\rightarrow}_{\Theta_k}(S)$.

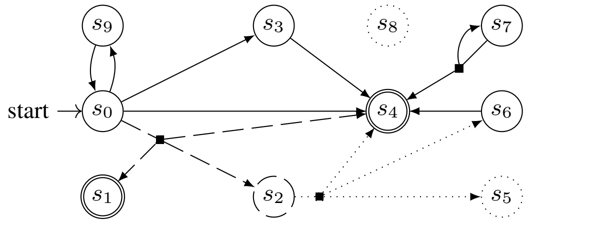

Figure 1. Transition system Θ used in Example 1.

In the following, we mostly use the above definitions with respect to the set of source states $S = I_\Theta$ and the target states $T = G_\Theta$ in the context of a TS Θ. We therefore use the corresponding shorthand notations $Reach^{\rightarrow}(\Theta) := Reach^{\rightarrow}_{\Theta}(I_\Theta)$ and $Reach^{\leftarrow}(\Theta) := Reach^{\leftarrow}_{\Theta}(G_\Theta)$, and analogously $Dead(\Theta)$, $Solv(\Theta)$ and $Alive(\Theta)$.

Example 1. *In the transition system Θ depicted in Fig. 1, we can see that $Reach^{\rightarrow}(\Theta) = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_9\}$ and $Dead(\Theta) = \{s_5, s_8\}$. To determine the solvable states of Θ, we first compute $\Theta_1 = \Theta|_{Reach^{\leftarrow}(\Theta)}$, which is the TS from which the dead ends of Θ and transitions leading to them are removed (drawn with dotted lines in Fig. 1). Repeating the procedure, we next prune the dead end s_2 and the transitions leading to s_2 (drawn with dashed lines) to obtain Θ_2. Finally, Θ_2 has no dead ends, so the algorithm terminates with the set of solvable states $Solv(\Theta) = S_{\Theta_2} = \{s_0, s_1, s_3, s_4, s_6, s_7, s_9\}$. In particular, s_2 is neither solvable nor a dead end, as we can reach the goal from s_2, but not with certainty. The set of alive states for s_0 can be computed as $Alive(\Theta) = Reach^{\rightarrow}(\Theta_2) = \{s_0, s_3, s_4, s_9\}$. In particular, the states s_1 and s_6 are forward reachable from s_0 and solvable, but not alive for s_0, because no policy that reaches these states from s_0 can reach the goal with certainty. Note that s_9 can be reached by a history-dependent solution that first goes to s_9 from s_0 and then behaves like a solution for s_0 after one step. However, an SD policy reaching s_9 from s_0 would produce an infinite cycle due to always repeating the same choice in s_0 regardless of the history. Since s_9 would also be alive from the view of the classical theory in the deterministic TS restricted to the states $\{s_0, s_3, s_4, s_9\}$, we would not obtain a proper generalisation of this concept if we only considered SD policies.*

As in classical planning, SSP heuristic search algorithms retain their correctness properties regardless of the heuristic estimates for unreachable states. It however turns out that for SSPs, one can even further relax the heuristic properties.

Definition 1. *A heuristic $h : S_\Theta \to \mathbb{R} \cup \{\infty\}$ for the TS Θ is*

a) alive-goal-aware *if $h(s) = 0$ for every $s \in Alive(\Theta) \cap G_\Theta$,*
b) alive-consistent *if $h(s) \leq C_\Theta(\ell) + \sum_{t \in S_\Theta} \delta(t) \cdot h(t)$ for every $\langle s, \ell, \delta \rangle \in T_\Theta$ with $s \in Alive(\Theta)$ and $supp(\delta) \subseteq Alive(\Theta)$,*
c) alive-safe *if $h(s) \neq \infty$ for every $s \in Alive(\Theta)$,*
d) alive-admissible *if $h(s) \leq J^*_\Theta(s)$ for every $s \in Alive(\Theta)$ and*
e) alive-perfect *if $h(s) = J^*_\Theta(s)$ for every $s \in Alive(\Theta)$ and $h(u) = \infty$ for every $u \notin Solv(\Theta)$ such that there exists $\langle t, \ell, \delta \rangle \in T_\Theta$ with $t \in Alive(\Theta) \wedge u \in supp(\delta)$.*

As usual, the heuristic value $h(s) = \infty$ is used to signal the search to discard s from consideration. For alive-perfection 1e), we want the heuristic to effectively restrict the search to the alive states only, which is guaranteed if the unsolvable successors of alive states are detected as such. When pruned, beyond those unsolvable states, no other non-alive states will be visited, making their heuristic estimates irrelevant. Note that, as is the case for the unrestricted versions, if a heuristic is alive-admissible, it is also alive-goal-aware and alive-safe. If a heuristic is alive-goal-aware, alive-safe, and alive-consistent, then it is alive-admissible. If a heuristic is alive-perfect, then it satisfies also all the other properties. Importantly, the alive properties leave the correctness properties of heuristic search intact:

Theorem 1. *Let $\mathcal{A}$ be an optimal SSP heuristic search algorithm applied on the TS Θ with initial state s_I, and let h be the search heuristic. If h is alive-safe, then $\mathcal{A}$ returns a solution for s_I, if existent. If h is even alive-admissible, this solution is optimal.*

Proof (Sketch). Pruning non-alive states does not affect the set of possible solutions, which suffices for the first claim. For the second part, since $\mathcal{A}$ always returns an optimal solution with an admissible heuristic, $\mathcal{A}$ will also find an optimal solution if we make all suboptimal choices look worse than they actually are; intuitively, this can only make the optimal choices look better. As h is only inadmissible for states not part of any solution, h can only result in pessimistic estimations of suboptimal choices, concluding the proof. □

Our argument applies to most popular heuristic search algorithms for SSPs, including LAO* [7], LRTDP [3], and i-dual [31].

4 A Theory of Prune Transformations

In this section, we formally define prune transformations on a factored ATS and embed them into the compositional theory of merge-and-shrink. As we will see, all interesting prune transformations are not conservative. We will therefore replace some of the conservativeness properties with less strict structural properties. We proceed as follows. After introducing the relaxed properties, we first show that they are sufficient to enforce the relaxed heuristic properties introduced in Section 3. Afterwards, we prove that these properties are invariant under composition under additional assumptions, which however hold for prune transformations as well as the other three core M&S transformations. We conclude the section with practical prune strategies that respect these properties.

We start by defining a pruning operation on a single annotated TS.

Definition 2. *Let A be an ATS and let $K \subseteq S_A$ be a subset of* kept states. *The pruned ATS $A|_K$ for A and K is defined as the ATS $A|_K := \{K, L_A, C_A, E_A, D_A, T_{A|_K}, I_A \cap K, G_A \cap K\}$, where $T_{A|_K} := \{\langle s, \ell, \alpha\rangle \in T_A \mid s \in K \wedge \alpha(supp(D_A(\ell))) \subseteq K\}$.*

With this, prune transformations on a factored ATS are defined as transformations that prune states from one specific factor.

Definition 3 (Prune Transformations)**.** *Let $F = \langle A_i\rangle_{i\in\mathcal{I}}$ be a factored ATS. Let $k \in \mathcal{I}$ be a factor index and let $K \subseteq S_{A_k}$ be a set of kept states. Lastly, let id_X be the identity function for domain X. The* prune transformation *for A_k and K is the factored transformation $\langle F, \langle A'_i\rangle_{i\in\mathcal{I}}, \langle\sigma_i\rangle_{i\in\mathcal{I}}, id_{L_F}\rangle$ where*

$$A'_i := \begin{cases} A_i & i \neq k \\ A_i|_K & i = k \end{cases} \qquad \sigma_i := \begin{cases} \text{Atom}(i, id_{S_{A_i}}) & i \neq k \\ \text{Atom}(i, id_K) & i = k \end{cases}$$

One can easily show that prune transformations satisfy all conservativeness properties except $\mathbf{CONS_S}$, which is only satisfied if $K = \emptyset$, i.e., no states are pruned at all. Unsurprisingly, they can yield inadmissible transformation heuristics. To nevertheless obtain sufficient criteria for admissibility, consistency, etc., we replace $\mathbf{CONS_S}$ with weaker properties inspired by the classical theory [26].

Definition 4. *Let $\tau = \langle\Theta, \Theta', \sigma, \lambda\rangle$ be a TS transformation. We define the following transformation properties on τ.*

$\mathbf{CONS_I}$	$\sigma(I_\Theta) \subseteq I_{\Theta'}$
$\mathbf{CLOS}$	$Alive_\Theta(S_\Theta, dom(\sigma)) \subseteq dom(\sigma)$
$\mathbf{CLOS^{ALV}}$	$Alive_\Theta(I_\Theta, dom(\sigma)) \subseteq dom(\sigma)$
$\mathbf{KEEP_G}$	$G_\Theta \subseteq dom(\sigma)$
$\mathbf{KEEP_G^{ALV}}$	$Alive(\Theta) \cap G_\Theta \subseteq dom(\sigma)$

Obviously, $\mathbf{CONS_S}$ ($S_\Theta = dom(\sigma)$) is stronger than any of the properties $\mathbf{CLOS}$, $\mathbf{CLOS^{ALV}}$, $\mathbf{KEEP_G}$ and $\mathbf{KEEP_G^{ALV}}$. The property $\mathbf{CLOS}$ requires that K is closed under probabilistic transitions in the sense that, if a state s is able to reach a set of kept states $K \subseteq dom(\sigma)$ with probability one, then s needs to be kept as well. For deterministic transition systems, this property guarantees that $dom(\sigma)$ is closed under predecessors, which matches the definition of the corresponding property denoted $\mathbf{CLOS}_{pred}$ in the classical case. The property $\mathbf{CLOS^{ALV}}$ is even weaker and requires only that if there is a policy that reaches a set of kept states $K \subseteq dom(\sigma)$ with certainty from an initial state, then all states reached by this policy must also be kept. It matches the classical property $\mathbf{CLOS}^{\rightarrow}_{pred}$. Moreover, $\mathbf{KEEP_G}$ requires that all goal states are kept, whereas the weaker $\mathbf{KEEP_G^{ALV}}$ preserves only alive goal states. For deterministic transition systems, $\mathbf{KEEP_G^{ALV}}$ keeps forward reachable goal states, thus matching $\mathbf{KEEP_G^{\rightarrow}}$ in the classical theory.

4.1 Heuristic Guarantees

First, we make the connection between the transformation properties of Definition 4 and the heuristic properties introduced in Section 3, in the context of the transformation heuristic h^τ. As shown by Klößner et al. [18], h^τ is goal-aware, consistent and safe if τ is conservative. The new transformation properties replace $\mathbf{CONS_S}$ so that h^τ still remains with these properties, or the corresponding weaker variants as defined in Definition 4, depending on the replacement properties.

Theorem 2. *Let τ be a transformation. Then h^τ is*

a) goal-aware, if $\tau \in \mathbf{CONS_G} \cap \mathbf{KEEP_G}$
b) consistent, if $\tau \in \mathbf{CONS_{L+C+T}} \cap \mathbf{CLOS}$
c) safe, if $\tau \in \mathbf{CONS_{L+T+G}} \cap \mathbf{CLOS} \cap \mathbf{KEEP_G}$
d) alive-goal-aware, if $\tau \in \mathbf{CONS_G} \cap \mathbf{KEEP_G^{ALV}}$
e) alive-consistent, if $\tau \in \mathbf{CONS_{L+C+T}} \cap \mathbf{CLOS^{ALV}} \cap \mathbf{KEEP_G^{ALV}}$
f) alive-safe, if $\tau \in \mathbf{CONS_{L+T+G}} \cap \mathbf{CLOS^{ALV}} \cap \mathbf{KEEP_G^{ALV}}$

Proof. Let $\tau = \langle\Theta, \Theta', \sigma, \lambda\rangle$. Theorems 2a) and 2d) are straightforward. For the other statements, we first argue that (A) $\mathbf{CLOS} + \mathbf{KEEP_G}$ implies $Solv(\Theta) \subseteq dom(\sigma)$ and (B) $\mathbf{CLOS^{ALV}} + \mathbf{KEEP_G^{ALV}}$ implies $Alive(\Theta) \subseteq dom(\sigma)$. Regarding (A), we have $Solv(\Theta) = Solv_\Theta(G_\Theta) \subseteq Solv_\Theta(dom(\sigma)) \subseteq dom(\sigma)$, where the first inclusion follows from $\mathbf{KEEP_G}$ and the second follows from $\mathbf{CLOS}$. For (B), first acknowledge that $Alive(\Theta) = Alive_\Theta(I_\Theta, Alive(\Theta) \cap G_\Theta)$. We conclude $Alive(\Theta) \subseteq Alive_\Theta(I_\Theta, dom(\sigma)) \subseteq dom(\sigma)$, where the inclusions similarly follow from $\mathbf{KEEP_G^{ALV}}$ and $\mathbf{CLOS^{ALV}}$.

Consider Theorem 2b). Let $\langle s, \ell, \delta\rangle \in T_\Theta$. We assume $supp(\delta) \subseteq dom(\sigma)$, as otherwise the right-hand side of the consistency inequation evaluates to ∞, making it trivial. We then have $s \in dom(\sigma)$ as well due to $\mathbf{CLOS}$, since we can reach $dom(\sigma)$ with certainty from s via this transition. We also have $\ell \in dom(\lambda)$ due to $\mathbf{CONS_L}$. Due to $\mathbf{CONS_T}$, it follows that $\langle s', \ell', \delta' \rangle := \langle \sigma(s), \lambda(\ell), \mathit{lift}[\sigma](\delta) \rangle \in T_{\Theta'}(\sigma(s))$. The following inequality concludes the proof.

$$\begin{aligned} J^*_{\Theta'}(s') &\le C_\Theta(\ell') + \sum_{t' \in S_{\Theta'}} \delta'(t') \cdot J^*_{\Theta'}(t') \quad \text{(consistency of } J^*_{\Theta'}\text{)} \\ &\le C_\Theta(\ell) + \sum_{t' \in S_{\Theta'}} \sum_{t \in \sigma^{-1}(t')} \delta(t) \cdot J^*_{\Theta'}(t') \quad \text{(def. } \mathit{lift}[\sigma]\text{)} \\ &= C_\Theta(\ell) + \sum_{t \in S_\Theta} \delta(t) \cdot J^*_{\Theta'}(\sigma(t)) \quad (\mathit{supp}(\delta) \subseteq \mathit{dom}(\sigma)) \end{aligned}$$

Next, consider Theorem 2c). Let $s \in \mathit{Solv}(\Theta)$. Consider the transformation $\tau' = \langle \Theta|_{\mathit{Solv}(\Theta)}, \Theta', \sigma|_{\mathit{Solv}(\Theta)}, \lambda \rangle$. Because $\tau \in \mathbf{CONS_{L+T+G}}$ and $\mathit{Solv}(\Theta) \subseteq \mathit{dom}(\sigma)$ by (A), it is easy to see that $\tau' \in \mathbf{CONS_{S+L+T+G}}$, which implies that $h^{\tau'}$ is safe [18]. Since $s \in \mathit{Solv}(\Theta) = \mathit{Solv}(\Theta|_{\mathit{Solv}(\Theta)})$, we conclude $\sigma(s) \in \mathit{Solv}(\Theta')$ by safety of $h^{\tau'}$ and therefore $h^\tau(s) \neq \infty$.

Now, consider Theorem 2e). Let $\langle s, \ell, \delta \rangle \in T_\Theta$, $s \in \mathit{Alive}(\Theta)$ and $\mathit{supp}(\delta) \subseteq \mathit{Alive}(\Theta)$. We conclude $s \in \mathit{dom}(\sigma)$ and $\mathit{supp}(\delta) \subseteq \mathit{dom}(\sigma)$ using (B). The proof continues analogous to Theorem 2b).

Lastly, Theorem 2f) is analogous to Theorem 2c), when substituting $\mathit{Solv}(\Theta)$ with $\mathit{Alive}(\Theta)$ and (A) with (B). □

4.2 Policy Transformation

In order to show that our new properties are indeed invariant under composition, we have to have means to relate policies in the original and in the transformed transition systems. In the classical theory of transformations, given a transformation $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$, there is a useful construction that transforms a concrete path $p = s_0 \mathfrak{T}_0 \dots s_n$ of the original TS Θ to a transformed path of Θ'. This *path transformation* is given by $\mathit{tpath}_\tau(p) := \sigma(s_0)\mathit{ttr}_\tau(\mathfrak{T}_0) \dots \sigma(s_n)$, and is defined for p if $s_0, \dots, s_n \in \mathit{dom}(\sigma)$ and $\mathfrak{T}_0, \dots, \mathfrak{T}_{n-1} \in \mathit{dom}(\mathit{ttr}_\tau)$. Before proceeding to our compositionality proof, here we will generalise this path transformation to policies. Since a path has an integrated starting state, which a policy does not, we have to explicitly assume one in the context of this transformation.

Definition 5. *Let $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ be a transformation, let $s \in S_\Theta$ be a starting state and let π be a policy for Θ. The transformed policy $\mathit{tpol}_{\tau,s}(\pi)$ of π for s is defined by $\mathit{tpol}_{\tau,s}(\pi)(p')(\mathfrak{T}') := 0$ if $\sum_{p \in \mathit{tpath}_\tau^{-1}(p')} \Pr^\pi_s[\mathit{Cyl}(p)] = 0$ and otherwise*

$$\mathit{tpol}_{\tau,s}(\pi)(p')(\mathfrak{T}') := \frac{\sum\limits_{p \in \mathit{tpath}_\tau^{-1}(p')} \left[\Pr^\pi_s[\mathit{Cyl}(p)] \cdot \sum\limits_{\mathfrak{T} \in \mathit{ttr}_\tau^{-1}(\mathfrak{T}')} \pi(p)(\mathfrak{T}) \right]}{\sum\limits_{p \in \mathit{tpath}_\tau^{-1}(p')} \Pr^\pi_s[\mathit{Cyl}(p)]}.$$

To compute the probability to choose the transition $\mathfrak{T}'$ after the history p' was observed, the construction calculates a conditional probability: Given that the original policy π generated some unknown corresponding concrete path $p \in \mathit{tpath}_\tau^{-1}(p')$ so far, what is the probability that π chooses a corresponding concrete transition $\mathfrak{T} \in \mathit{ttr}_\tau^{-1}(\mathfrak{T}')$ next? The term $\sum_{\mathfrak{T} \in \mathit{ttr}_\tau^{-1}(\mathfrak{T}')} \pi(p)(\mathfrak{T})$ calculates the probability that π chooses a corresponding concrete transition for a given path p. Since we only know that π generated some path $p \in \mathit{tpath}_\tau^{-1}(p')$, but not which one, we embed this term into a sum such that it is weighted with the probability of the path it is based on. Finally, we normalise by the total probability of these paths, since we know that such a path has been generated. Note that all probabilities depend on the initial state s from which π is run. If no concrete path $p \in \mathit{tpath}_\tau^{-1}(p')$ is possible, the policy terminates.

We will use this construction to transform solutions in Θ to solutions in Θ'. This can be done under mild conservativeness assumptions, specifically label and transition conservativeness ($\mathbf{CONS_{L+T}}$), and under the requirement that all states reachable from the considered starting state s by the original policy π are kept by the transformation ($\mathit{Reach}^{\rightarrow}_{\Theta,\pi}(s) \subseteq \mathit{dom}(\sigma)$). In this case, for each concrete state t that the original policy π reaches from s, the transformed policy $\mathit{tpol}_{\tau,s}(\pi)$ will reach the corresponding transformed state $\sigma(t)$ from $\sigma(s)$. Vice versa, for each transformed state t' that $\mathit{tpol}_{\tau,s}(\pi)$ reaches from $\sigma(s)$, π reaches some concrete state $t \in \sigma^{-1}(t')$. Formally, we have the following (proof is available in the appendix [20]):

Theorem 3. *Let $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ be a transformation, let π be a policy, let $s \in S_\Theta$ be a starting state and let $T \subseteq S_\Theta$ be a set of target states. If $\tau \in$ $\mathbf{CONS_{L+T}}$ and $\mathit{Reach}^{\rightarrow}_{\Theta,\pi}(s) \subseteq \mathit{dom}(\sigma)$, then*

a) $\mathit{Reach}^{\rightarrow}_{\Theta',\mathit{tpol}_{\tau,s}(\pi)}(\sigma(s)) = \sigma(\mathit{Reach}^{\rightarrow}_{\Theta,\pi}(s))$ and
b) If $\pi \in \mathit{Sols}_\Theta(s, T)$, then $\mathit{tpol}_{\tau,s}(\pi) \in \mathit{Sols}_{\Theta'}(\sigma(s), \sigma(T))$.

4.3 Compositionality

The tools from Section 4.2 empower us to prove the compositionality of the properties from Definition 4, under mild side conditions similar to the classical planning setting [26].

Theorem 4. *Let $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ and $\tau' = \langle \Theta', \Theta'', \sigma', \lambda' \rangle$ be two transformations. For the following properties* $\mathbf{X}$*, if $\tau, \tau' \in$* $\mathbf{X}$*, then $\tau' \circ \tau \in$* $\mathbf{X}$ *under the following additional side requirements:*

a) $\mathbf{CLOS}$ *requires $\tau \in$* $\mathbf{CONS_{L+T}}$
b) $\mathbf{CLOS^{ALV}}$ *requires $\tau \in$* $\mathbf{CONS_{L+T+I}}$
c) $\mathbf{KEEP_G}$ *requires $\tau \in$* $\mathbf{CONS_G}$
d) $\mathbf{KEEP_G^{ALV}}$ *requires $\tau \in$* $\mathbf{CONS_{L+T+I+G}} \cap \mathbf{CLOS^{ALV}}$

Proof. For Theorem 4c), we refer to Sievers and Helmert [26].

Consider Theorem 4a). Let $s \in \mathit{Alive}_\Theta(S_\Theta, \mathit{dom}(\sigma' \circ \sigma))$. Then there is a policy π solving s for $\mathit{dom}(\sigma' \circ \sigma) \subseteq \mathit{dom}(\sigma)$. Then we also have $\mathit{Reach}^{\rightarrow}_{\Theta,\pi}(s) \subseteq \mathit{Alive}_\Theta(S_\Theta, \mathit{dom}(\sigma)) \subseteq \mathit{dom}(\sigma)$ due to $\mathbf{CLOS}$. Applying Theorem 3 with the assumption $\tau \in \mathbf{CONS_{L+T}}$, we conclude that $\mathit{tpol}_{\tau,s}(\pi)$ solves $\sigma(s)$ for the target states $\sigma(\mathit{dom}(\sigma' \circ \sigma)) \subseteq \mathit{dom}(\sigma')$. Thus, $\sigma(s) \in \mathit{Alive}_\Theta(S_{\Theta'}, \mathit{dom}(\sigma'))$. With $\tau' \in \mathbf{CLOS}$, we obtain $\sigma(s) \in \mathit{dom}(\sigma')$ and ultimately $s \in \mathit{dom}(\sigma' \circ \sigma)$.

Consider Theorem 4b). Let $s \in \mathit{Alive}_\Theta(I_\Theta, \mathit{dom}(\sigma' \circ \sigma))$. Then there is a policy π with $s_0 \overset{\pi}{\rightsquigarrow} s$ that solves an initial state $s_0 \in I_\Theta$ for $\mathit{dom}(\sigma' \circ \sigma) \subseteq \mathit{dom}(\sigma)$. We also have $\mathit{Reach}^{\rightarrow}_{\Theta,\pi}(s_0) \subseteq \mathit{Alive}_\Theta(I_\Theta, \mathit{dom}(\sigma)) \subseteq \mathit{dom}(\sigma)$ due to $\tau \in \mathbf{CLOS^{ALV}}$. Applying Theorem 3 with the assumption $\tau \in \mathbf{CONS_{L+T}}$, we conclude that $\mathit{tpol}_{\tau,s_0}(\pi)$ solves $\sigma(s_0)$ for $\sigma(\mathit{dom}(\sigma' \circ \sigma)) \subseteq \mathit{dom}(\sigma')$, and $\sigma(s_0) \overset{\mathit{tpol}_{\tau,s_0}(\pi)}{\rightsquigarrow} \sigma(s)$. Since we have $\sigma(s_0) \in I_{\Theta'}$ due to $\mathbf{CONS_I}$, we get $\sigma(s) \in \mathit{Alive}_\Theta(I_{\Theta'}, \mathit{dom}(\sigma')) \subseteq \mathit{dom}(\sigma')$ by $\mathbf{CLOS^{ALV}}$.

Lastly, Theorem 4d). Let $s \in \mathit{Alive}(\Theta) \cap G_\Theta$. As $s \in \mathit{Alive}(\Theta)$, there is an initial state $s_0 \in I_\Theta$ and $\pi \in \mathit{Sols}_\Theta(s_0)$ with $s_0 \overset{\pi}{\rightsquigarrow} s$. We have $\mathit{Reach}^{\rightarrow}_{\Theta,\pi}(s_0) \subseteq \mathit{Alive}(\Theta) \subseteq \mathit{Alive}_\Theta(I_\Theta, \mathit{dom}(\sigma)) \subseteq \mathit{dom}(\sigma)$ due to $\mathbf{KEEP_G^{ALV}}$ and $\mathbf{CLOS^{ALV}}$. Applying Theorem 3 with the assumption $\tau \in \mathbf{CONS_{L+T}}$, we obtain that $\mathit{tpol}_{\tau,s_0}(\pi)$ solves $\sigma(s_0)$ and $\sigma(s_0) \overset{\mathit{tpol}_{\tau,s_0}(\pi)}{\rightsquigarrow} \sigma(s)$. Since $\sigma(s_0) \in I_{\Theta'}$ by $\mathbf{CONS_I}$, we have $\sigma(s) \in \mathit{Alive}(\Theta')$. Finally, $\sigma(s) \in G_{\Theta'}$ due to $\mathbf{CONS_G}$. □

4.4 Prune Strategies

Finally, we discuss how the set of kept states K of a prune transformation can be practically chosen so that either the properties

$\mathbf{KEEP_G}$ and **CLOS**, or alternatively $\mathbf{KEEP_G^{ALV}}$ and $\mathbf{CLOS^{ALV}}$ are satisfied. Since prune transformations always satisfy $\mathbf{CONS_{L+T+C+I+G}}$ and are refinable, these two categories will yield perfect or alive-perfect heuristics, respectively, by Theorem 2. Furthermore, composing such transformations with each other or with exact transformations also yields perfect or alive-perfect heuristics by Theorem 4.

Theorem 5. *Let $F = \langle A_i \rangle_{i \in \mathcal{I}}$ be a factored ATS and let τ be a prune transformation for A_k and $K \subseteq S_{A_k}$.*

a) If $K = Reach^{\leftarrow}(\Theta(A_k))$, then $\tau \in \mathbf{CLOS} \cap \mathbf{KEEP_G}$.
b) If $K = Reach^{\rightarrow}(\Theta(A_k))$, then $\tau \in \mathbf{CLOS^{ALV}} \cap \mathbf{KEEP_G^{ALV}}$.

Proof. Let $\Theta := \Theta(\bigotimes F)$, $\Theta' := \Theta(\bigotimes F')$ and $\Theta_i := \Theta(A_i)$ for $i \in \mathcal{I}$ in the following. We need to consider the TS transformation $\langle \Theta, \Theta', \sigma, id \rangle$, where $s \in dom(\sigma)$ if and only if $s(k) \in K$.

Let $K = Reach^{\leftarrow}(\Theta_k)$. For $\mathbf{KEEP_G}$, note that $s \in G_\Theta = \bigtimes_{i \in \mathcal{I}} G_{\Theta_i}$, clearly implies $s(k) \in G_{\Theta_k} \subseteq Reach^{\leftarrow}_{\Theta}(G_{\Theta_k}) = K$. For **CLOS**, let $s \in Alive_\Theta(S_\Theta, dom(\sigma))$. Then s must necessarily be backwards reachable from a state $t \in dom(\sigma)$. Concludingly, $s(k)$ is backwards reachable from $t(k) \in K = Reach^{\leftarrow}_{\Theta}(G_\Theta)$. By transitivity, $s(k) \in Reach^{\leftarrow}_{\Theta}(G_{\Theta_k}) = K$.

Now, let $K = Reach^{\rightarrow}(\Theta_k)$. For $\mathbf{KEEP_G^{ALV}}$, let $s \in Alive(\Theta) \cap G_\Theta$. Then s must be forward reachable from an initial state $s_0 \in I_\Theta = \bigtimes_{i \in \mathcal{I}} I_{\Theta_i}$. Concludingly, $s(k)$ must be forward reachable from $s_0(k) \in I_{\Theta_k}$, which shows $s(k) \in K$. Since $G_\Theta = \bigtimes_{i \in \mathcal{I}} G_{\Theta_i}$, we also have $s(k) \in G_{\Theta_k}$. For $\mathbf{CLOS^{ALV}}$, let $s \in Alive_\Theta(I_\Theta, dom(\sigma))$. Then, s is forward reachable from $s_0 \in I_\Theta$. By repeating the arguments as above, we conclude $s(k) \in K$. □

Theorem 5 establishes two reachability-based prune strategies. In practice, the backwards-reachable or forward-reachable states of a factor are easy to compute, but may not lead to substantial pruning depending on the topology of the state space. We can however extend this result to obtain stronger prune strategies at the expense of additional computation time. Note that we can simulate a prune transformation on factor A_k with $K = Solv(\Theta(A_k))$, keeping only solvable states, via a series of prune transformations with $K = Reach^{\leftarrow}(\Theta(A_k))$. Likewise, to simulate $K = Alive(\Theta(A_k))$, we can apply a prune transformation with $K = Solv(\Theta(A_k))$, followed by a prune transformation with $K = Reach^{\rightarrow}(\Theta(A_k))$. By compositionality (Theorem 4) and the obvious facts $\mathbf{CLOS} \subseteq \mathbf{CLOS^{ALV}}$ and $\mathbf{KEEP_G} \subseteq \mathbf{KEEP_G^{ALV}}$, we conclude the following.

Corollary 1. *Let $F = \langle A_i \rangle_{i \in \mathcal{I}}$ be a factored ATS and let τ be a prune transformation for A_k and $K \subseteq S_{A_k}$.*

a) If $K = Solv(\Theta(A_k))$, then $\tau \in \mathbf{CLOS} \cap \mathbf{KEEP_G}$.
b) If $K = Alive(\Theta(A_k))$, then $\tau \in \mathbf{CLOS^{ALV}} \cap \mathbf{KEEP_G^{ALV}}$.

5 Experiments

We empirically evaluate the probabilistic merge-and-shrink framework in different configurations. Our implementation is based on our version of Probabilistic Fast Downward [30]. The code is publicly available [19]. We stuck closely to the classical-planning merge-and-shrink implementation. The top-level algorithm takes an abstract state limit M as a parameter, as well as a merge, shrink, prune and a label reduction strategy. We compute the abstractions' J^* values using a variant of topological value iteration [4] that supports zero-cost actions and unsolvable states.

Regarding shrink strategies, we implemented ATS bisimulation shrinking as described by Klößner et al. [18], following the standard partition-refinement based algorithm for computing bisimulations. To respect the state limit, the refinement stops early if splitting an equivalence class would result in more than M classes. In that case, the shrink transformation will not be exact. For label reduction, we implemented exact label reduction based on (A, ϵ)-combinability. Here, we keep an ordering of the possible effects of each label, and unify only labels with the same amount of possible effects and for which the effect probabilities according to this order match. Finally, we implemented two prune strategies, one that keeps only the solvable states, and one that keeps only the alive states of a factor.

We compare SSP M&S heuristics with other heuristics in the context of heuristic search to compute an optimal policy for a given stochastic shortest-path problem. We focus on improved LAO* [7] as the heuristic search algorithm in our experiments, extended with a trap-elimination procedure [21, 30] to support zero-cost actions.

The experiments were run with Downward Lab [25] on a cluster with Intel Xeon E5–2650 v3 processors CPUs @2.30 GHz. We used a memory limit of 4GiB and a time limit of 30 minutes for all configurations. We use the benchmark set by Klößner et al. [17], containing 9 probabilistic PDDL domains with 20 problems each, some of them containing traps or unsolvable states.

5.1 SSP M&S Versus Determinization-based M&S

First, we compare SSP M&S heuristics with their classical counterpart to assess the benefit of taking the stochasticity into account. To this end, we apply the all-outcomes determinization on the given task to obtain a classical planning problem, and then compute a classical M&S heuristic to be used for the heuristic search. While determinization-based heuristics are faster to construct, they discard all probabilities, trading construction time for heuristic accuracy.

We ran both M&S variants with an abstract state limit of $50k$, which turned out to be effective in preliminary experiments. Both variants use their respective notion of bisimulation shrinking. We selected two merge strategies from classical planning: The reverse level linear merge strategy [22] which orders variables closer to the root of the causal graph first and the state-of-the-art strategy SCC-DFP [28]. For label reduction, we use exact label reduction based on Θ-combinability [27], respectively (A, ϵ)-combinability [18]. Here, we sequentially select each factor as the pivot for the combinability relation, and then collapse all combinable labels. This is done until no more labels can be combined. Finally, we consider three prune strategies: Keeping all, only solvable and only alive states.

Table 1 shows the coverage table. Here, the superscript M&S represents the SSP variant, while dM&S represents the classical variant. The subscripts All, Solv and Aliv denote the prune strategy. SSP M&S heuristics consistently cover more instances than their classical relatives for matching algorithm configurations. Looking at the number of evaluated states for our best configurations in Fig. 2a, we see that there are many problems for which our variant generates a significantly smaller search space, while the opposite is rare. However, the SSP variant takes considerably longer in most problems, as we can see in Fig. 2c. As evident from Fig. 2b, this is due to the heuristic's construction, which is considerably more expensive for the SSP variants. Here, the factored representation contains more information that needs to be maintained during each transformation. Moreover, computing the abstract states' optimal expected costs is computationally much more demanding than computing the costs in the deterministic setting. Despite being about an order of magnitude slower

	Linear Reverse Level						SCC-DFP										
Domains	$h^{\text{M\&S}}_{\text{Aliv}}$	$h^{\text{M\&S}}_{\text{All}}$	$h^{\text{M\&S}}_{\text{Solv}}$	$h^{\text{dM\&S}}_{\text{Aliv}}$	$h^{\text{dM\&S}}_{\text{All}}$	$h^{\text{dM\&S}}_{\text{Solv}}$	$h^{\text{M\&S}}_{\text{Aliv}}$	$h^{\text{M\&S}}_{\text{All}}$	$h^{\text{M\&S}}_{\text{Solv}}$	$h^{\text{dM\&S}}_{\text{Aliv}}$	$h^{\text{dM\&S}}_{\text{All}}$	$h^{\text{dM\&S}}_{\text{Solv}}$	h^{blind}	h^{PDB}_{10k}	h^{Cart}_{10k}	h^{roc}	h^{PDB}_{HC}
BLOCKSWORLD	**9**	**9**	**9**	7	7	7	**9**	**9**	**9**	7	7	7	7	7	7	7	**9**
BOXWORLD	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	4	4	4	4	6
ELEVATORS	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	13	15	14	12	**18**
PROB-PARC-PRINTER	14	15	15	12	13	13	16	14	12	12	10	9	8	8	8	**20**	16
RANDOM	12	11	11	12	12	12	12	11	11	12	12	12	14	17	16	15	**18**
SCHEDULE	11	11	11	11	11	11	11	10	10	11	11	11	**12**	**12**	**12**	11	**12**
SYSADMIN	11	**12**	11	**12**	**12**	**12**	11	11	11	11	11	**12**	11	11	11	11	11
TRIANGLE-TIREWORLD	7	7	7	6	6	6	**8**	6	**8**	7	7	7	5	**8**	6	7	**8**
ZENOTRAVEL	9	9	9	8	8	8	9	9	9	8	8	8	5	8	8	7	**10**
Sum (180)	98	99	98	93	94	94	101	95	95	93	91	91	79	90	86	94	**108**

Table 1. Coverage results for all tested configurations. Each number reports the number of solved instances for the respective domain and configuration. The highest coverage per domain is highlighted in boldface. All domains have 20 problem instances.

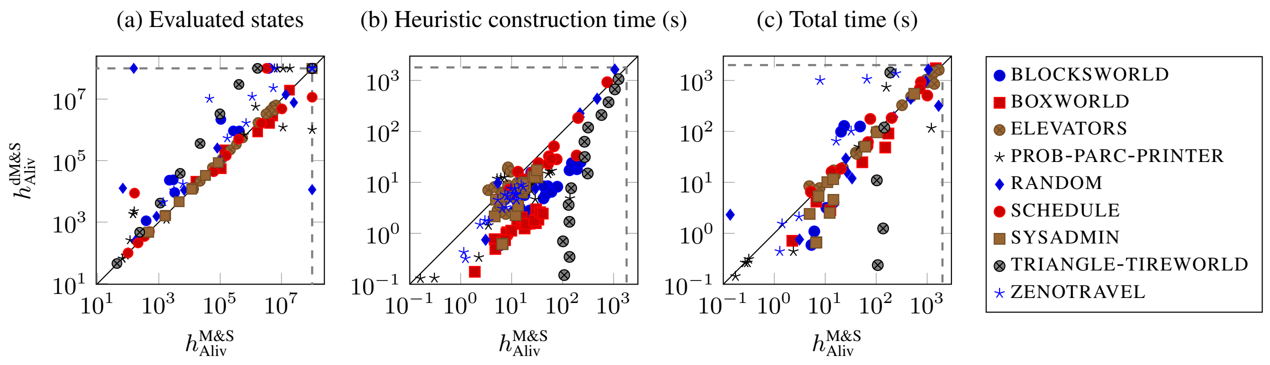

Figure 2. Probabilistic $h^{\text{M\&S}}_{\text{Aliv}}$ (x-axis) versus deterministic $h^{\text{dM\&S}}_{\text{Aliv}}$ (y-axis), both using the SCC-DFP merge strategy.

in many instances of BLOCKSWORLD, the construction pays off for larger instances (see Fig. 2c) and the SSP heuristic is able to solve two additional instances in this domain. In TRIANGLE-TIREWORLD, the trade-off also becomes more favourable as the size of the problem grows, which results in one additional solved instance.

The bottlenecks of the algorithm vary greatly across different configurations and benchmark instances. The maintenance of J^* for all factors adds a significant overhead. The time spent computing the label abstraction for label reduction is negligible in 6 domains, but becomes dominant in the domains RANDOM and TRIANGLE-TIREWORLD, where it is the most expensive process by far. The time spent computing ATS bisimulations is usually on the lower end of the spectrum, the highest relative overhead for the SCC-DFP variant of $h^{\text{M\&S}}_{\text{Aliv}}$ is introduced in BLOCKSWORLD with 29% of the algorithm runtime (similarly for other configurations). The actual transformations take negligible time in relation to these processes.

5.2 *SSP M&S Versus other SSP Abstraction Heuristics*

Next, we compare our approach against previously considered SSP abstraction heuristics. We consider SSP pattern database (PDB) heuristics [14], SSP Cartesian abstraction heuristics [17] and the occupation measure heuristic h^{roc} [32]. We construct single-abstraction PDB and Cartesian abstraction heuristics via policy-based counter-example guided abstraction refinement, with a limit of $10k$ states for the final abstraction. In preliminary experiments, we observed worse results for higher limits. We also include a state-of-the-art configuration that computes the canonical PDB heuristic over multiple PDBs constructed via hill-climbing search over the space of PDB collections ($h^{\text{PDB}}_{\text{HC}}$) [16]. We run hill-climbing for 180 seconds, with a collection size limit of 10 million abstract states.

The coverage is reported in Table 1. Overall, all SSP M&S configurations achieve a higher coverage than their single-abstraction sibling heuristics, as well as h^{roc}, showing that the expressiveness of the framework is highly beneficial. Yet, they do not quite reach the performance of the state-of-the-art heuristic $h^{\text{PDB}}_{\text{HC}}$. However, in contrast to $h^{\text{PDB}}_{\text{HC}}$, our configurations do not use a construction time limit to ensure that the search always starts, as the strictness of the time limit for the M&S algorithm is hard to enforce and may affect comparability of the M&S configurations. This leads to three timeouts of our best configuration in problems solved by blind search. Moreover, $h^{\text{PDB}}_{\text{HC}}$ uses multiple abstractions. The combination of multiple M&S heuristics has already been considered in classical planning [29], and achieves a better performance overall than the single-abstraction approach. It is likely that these developments can be extended to the SSP setting to achieve state-of-the-art performance.

6 Conclusion

In this paper, we rounded off the existing theory of SSP M&S with a formal analysis of prune transformations. We have established transformation properties that generalise those considered by Sievers and Helmert [26] in the context of prune transformations and proved their correctness using a notion of policy transformation. We propose several prune strategies that trade off computational effort with overall effectiveness. Our experiments show that SSP M&S heuristics perform better than their determinization-based variant, as well as previously considered SSP single-abstraction heuristics. Our work leaves room for many possible continuations following the footsteps of classical planning research on this topic, e.g., an exploration of merge or shrink strategies [13, 12, 28], or the construction of an cost-partitioned ensemble of M&S heuristics [29].

Acknowledgements

This work was funded by DFG grant 389792660 as part of TRR 248 (see https://perspicuous-computing.science).

References

[1] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press, 2008.

[2] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16:580–595, 1991.

[3] B. Bonet and H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In E. Giunchiglia, N. Muscettola, and D. Nau, editors, *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*, pages 12–21, Trento, Italy, 2003. AAAI Press.

[4] P. Dai, Mausam, D. S. Weld, and J. Goldsmith. Topological value iteration algorithms. *Journal of Artificial Intelligence Research*, 42:181–209, 2011.

[5] S. Edelkamp. Planning with pattern databases. In A. Cesta and D. Borrajo, editors, *Proceedings of the 6th European Conference on Planning (ECP'01)*, pages 13–24. Springer-Verlag, 2001.

[6] D. Fišer, R. Horčík, and A. Komenda. Strengthening potential heuristics with mutexes and disambiguations. In *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS'20)*, pages 124–133. AAAI Press, 2020.

[7] E. A. Hansen and S. Zilberstein. LAO*: a heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.

[8] P. Haslum, A. Botea, M. Helmert, B. Bonet, and S. Koenig. Domain-independent construction of pattern database heuristics for cost-optimal planning. In A. Howe and R. C. Holte, editors, *Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07)*, pages 1007–1012, Vancouver, BC, Canada, July 2007. AAAI Press.

[9] M. Helmert, P. Haslum, and J. Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In M. Boddy, M. Fox, and S. Thiebaux, editors, *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, pages 176–183, Providence, Rhode Island, USA, 2007. Morgan Kaufmann.

[10] M. Helmert, P. Haslum, J. Hoffmann, and R. Nissim. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the Association for Computing Machinery*, 61(3): 16:1–16:63, 2014.

[11] M. Helmert, G. Röger, and S. Sievers. On the expressive power of non-linear merge-and-shrink representations. In R. Brafman, C. Domshlak, P. Haslum, and S. Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, pages 106–114. AAAI Press, 2015.

[12] J. Hoffmann, P. Kissmann, and Á. Torralba. "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In T. Schaub, editor, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, pages 441–446, Prague, Czech Republic, Aug. 2014. IOS Press. doi: 10.3233/978-1-61499-419-0-441.

[13] M. Katz, J. Hoffmann, and M. Helmert. How to relax a bisimulation? In B. Bonet, L. McCluskey, J. R. Silva, and B. Williams, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 101–109. AAAI Press, 2012.

[14] T. Klößner and J. Hoffmann. Pattern Databases for Stochastic Shortest Path Problems. In *Proceedings of the 14th Annual Symposium on Combinatorial Search (SOCS'21)*, pages 131–135. AAAI Press, 2021.

[15] T. Klößner, Á. Torralba, M. Steinmetz, and J. Hoffmann. Pattern Databases for Goal-Probability Maximization in Probabilistic Planning. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS'21)*, pages 80–89. AAAI Press, 2021.

[16] T. Klößner, M. Steinmetz, À. Torralba, and J. Hoffmann. Pattern Selection Strategies for Pattern Databases in Probabilistic Planning. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22)*, page 184–192. AAAI Press, 2022.

[17] T. Klößner, J. Seipp, and M. Steinmetz. Cartesian Abstractions and Saturated Cost Partitioning in Probabilistic Planning. In *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI'23)*, pages 1272–1279, Kraków, Poland, Sept. 2023. IOS Press.

[18] T. Klößner, Á. Torralba, M. Steinmetz, and S. Sievers. A Theory of Merge-and-Shrink for Stochastic Shortest Path Problems. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS'23)*, pages 203–211. AAAI Press, 2023.

[19] T. Klößner, Á. Torralba, M. Steinmetz, and S. Sievers. Code for the Paper "Merge-and-Shrink for Stochastic Shortest-Path Problems with Prune Transformations" (ECAI '24). https://doi.org/10.5281/zenodo.12699090, 2024.

[20] T. Klößner, Á. Torralba, M. Steinmetz, and S. Sievers. Proof Appendix of the Paper "Merge-and-Shrink for Stochastic Shortest-Path Problems with Prune Transformations" (ECAI '24). https://doi.org/10.5281/zenodo.12705958, 2024.

[21] A. Kolobov, Mausam, D. S. Weld, and H. Geffner. Heuristic search for generalized stochastic shortest path MDPs. In F. Bacchus, C. Domshlak, S. Edelkamp, and M. Helmert, editors, *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS'11)*. AAAI Press, 2011.

[22] R. Nissim, J. Hoffmann, and M. Helmert. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, pages 1983–1990. AAAI Press/IJCAI, 2011.

[23] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887.

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

[25] J. Seipp, F. Pommerening, S. Sievers, and M. Helmert. Downward Lab. https://doi.org/10.5281/zenodo.790461, 2017.

[26] S. Sievers and M. Helmert. Merge-and-shrink: A compositional theory of transformations of factored transition systems. *Journal of Artificial Intelligence Research*, 71:781–883, 2021.

[27] S. Sievers, M. Wehrle, and M. Helmert. Generalized label reduction for merge-and-shrink heuristics. In C. E. Brodley and P. Stone, editors, *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, pages 2358–2366, Austin, Texas, USA, Jan. 2014. AAAI Press.

[28] S. Sievers, M. Wehrle, and M. Helmert. An analysis of merge strategies for merge-and-shrink heuristics. In A. Coles, A. Coles, S. Edelkamp, D. Magazzeni, and S. Sanner, editors, *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS'16)*, pages 294–298. AAAI Press, 2016.

[29] S. Sievers, F. Pommerening, T. Keller, and M. Helmert. Cost-partitioned merge-and-shrink heuristics for optimal classical planning. pages 4152–4160, 7 2020. doi: 10.24963/ijcai.2020/574. URL https://doi.org/10.24963/ijcai.2020/574.

[30] M. Steinmetz, J. Hoffmann, and O. Buffet. Goal probability analysis in MDP probabilistic planning: Exploring and enhancing the state of the art. *Journal of Artificial Intelligence Research*, 57:229–271, 2016.

[31] F. W. Trevizan, F. Teichteil-Königsbuch, and S. Thiébaux. Efficient solutions for stochastic shortest path problems with dead ends. In G. Elidan, K. Kersting, and A. T. Ihler, editors, *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI'17)*. AUAI Press, 2017.

[32] F. W. Trevizan, S. Thiébaux, and P. Haslum. Occupation measure heuristics for probabilistic planning. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*, pages 306–315. AAAI Press, 2017.