Merge-and-Shrink Heuristics for Stochastic Shortest-Path Problems with Prune Transformations

Thorsten Klößner[1], Álvaro Torralba[2], Marcel Steinmetz[1], Silvan Sievers[3]

[1]Saarland University, Germany, [2]Aalborg University, Denmark, [3]LAAS-CNRS, France, [4]Basel University, Switzerland
kloessner@cs.uni-saarland.de, marcel.steinmetz@laas.fr, alto@cs.aau.dk, silvan.sievers@unibas.ch

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 are highly competitive with or even outperform previous single abstraction heuristics, but they 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 (SSP) problem (Bertsekas and Tsitsiklis 1991), where one aims to find 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 SSP heuristic search algorithms (e.g., Hansen and Zilberstein 2001) guided with admissible probabilistic-abstraction heuristics (Trevizan, Thiébaux, and Haslum 2017; Klößner et al. 2021; Klößner and Hoffmann 2021). Such abstraction 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 (deterministic) planning, including pattern databases (Edelkamp 2001; Haslum et al. 2007; Klößner and Hoffmann 2021; Klößner et al. 2022) and counterexample-guided cartesian abstractions (Seipp and Helmert 2018; Klößner, Seipp, and Steinmetz 2023).

Merge-and-shrink abstractions have also received significant attention in classical planning research (Helmert, Haslum, and Hoffmann 2007; Helmert et al. 2014; Helmert, Röger, and Sievers 2015; Sievers and Helmert 2021). Merge-and-shrink describes a generic framework for constructing highly informative abstractions, which is based on a factored representation of transition systems and operations manipulating this representation, called *transformations* (Sievers and Helmert 2021). Merge-and-shrink guarantees constructing admissible abstraction heuristics, as long as only transformations are used which preserve specific compositional properties, regardless of their order of application.

Recently, this framework has been generalized to probabilistic planning (Klößner et al. 2023), 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. showed that these transformations preserve compositional properties needed to guarantee correctness of the abstraction construction. However, two gaps remain open. On the one hand, the theory by Klößner et al. 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 that are part of a policy which guarantees to reach the goal with certainty), in order for standard SSP heuristic search algorithms to retain their completeness and optimality guarantees. We derive prune transformations that help to 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.

2 Background

We make use of the following notation. The domain X of a function $f : X \to Y$ is denoted by $dom(f)$. A partial function $f : X \rightharpoonup Y$ is a function such that $dom(f) \subseteq X$. A function $\delta : X' \to [0, 1]$ is (sub-) stochastic over $X \subseteq X'$, iff $\sum_{x \in X} \delta(x) = 1$ (resp. ≤ 1). The set of such functions is denoted $\mathit{Dist}(X)$ (resp. $\mathit{SubDist}(X)$). We define $supp(\delta) := \{x \in X \mid \delta(x) > 0\}$ as the *support* of $\delta \in \mathit{SubDist}(X)$.

Probabilistic Transition Systems

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

An (in-)finite path of Θ is an (in-)finite alternating sequence $p = s_0 \mathfrak{t}_0 \ldots$ of states $s_i \in S_\Theta$ and transitions $\mathfrak{t}_i \in T_\Theta$ for all valid indices i. If p is a finite path, p is alternatively denoted as a *history*, and must end in a state $last(p)$. The set of all histories is denoted $\mathit{Hist}(\Theta)$.

Usually, *stationary and deterministic* (SD in 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 execution $s \in S_\Theta$. Unfortunately, such policies are not expressive enough for our developments, as we will see later. We consider *history-dependent* and *stochastic* policies $\pi : \mathit{Hist}(\Theta) \to \mathit{SubDist}(T_\Theta)$. If h is the execution history so far, π executes the transition $\mathfrak{t}$ next with probability $\pi(h)(\mathfrak{t})$, and terminates with probability $term_\pi(h) := 1 - \sum_{\mathfrak{t} \in T_\Theta} \pi(h)(\mathfrak{t})$. Only transitions of the current state may be chosen, i.e., $supp(\pi(h)) \subseteq T_\Theta(last(h))$. Given a starting state $s \in S_\Theta$, π induces a probability space over the paths (i.e., possible executions) of Θ (Baier and Katoen 2008). The event space is the σ-algebra generated by the cylinder sets $\mathit{Cyl}(h) := \{p \mid h \text{ is a prefix of } p\}$, for $h \in \mathit{Hist}(\Theta)$. The probability measure $\Pr_s^\pi$ is the unique extension of the pre-measure $\mathcal{P}[\mathit{Cyl}(s_0 \ldots s_n)] := [s_0 = s] \cdot \prod_{i=0}^{n-1} \pi(s_0 \ldots s_i)(\mathfrak{t}_i) \cdot \delta_{\mathfrak{t}_i}(s_{i+1})$ (defined using Iverson brackets) to the full σ-algebra. In particular, the probability of a set of histories $E \subseteq \mathit{Hist}(\Theta)$ is given by the countable sum $\Pr_s^\pi[E] = \sum_{h \in E} \Pr_s^\pi[\mathit{Cyl}(h)] \cdot term_\pi(h)$.

A policy π *solves* $s \in S_\Theta$ for the target states $T \subseteq S_\Theta$ if $\Pr_s^\pi[\mathit{Finish}_\Theta(T)] = 1$, where $\mathit{Finish}_\Theta(T) := \{h \in \mathit{Hist}(\Theta) \mid last(h) \in T\}$. The set of such solutions is denoted by $\mathit{Sols}_\Theta(s, T)$. The *expected cost-to-goal* of $\pi \in \mathit{Sols}_\Theta(s, G_\Theta)$ for a starting state s is defined as $J_\Theta^\pi(s) := \mathbb{E}_s^\pi[\mathsf{Cost}]$, where $\mathsf{Cost}(s_0 \mathfrak{t}_0 \ldots s_n) := \sum_{i=0}^{n-1} C_\Theta(\mathfrak{t}_i)$ is a random variable measuring the accumulated cost of an execution. The *optimal expected cost-to-goal* of s is defined as $J_\Theta^*(s) := \inf_{\pi \in \mathit{Sols}_\Theta(s, G_\Theta)} J_\Theta^\pi(s)$. A solution $\pi^\star \in \mathit{Sols}_\Theta(s, G_\Theta)$ is optimal for $s \in S_\Theta$ if $J_\Theta^\pi(s) = J_\Theta^*(s)$. We want to find an optimal initial state $s_0 \in \arg\min_{s \in I_\Theta} J_\Theta^*(s)$ and an optimal policy for s_0, if they exist.

The Merge-and-Shrink Framework

A heuristic is a function $h : S_\Theta \to \mathbb{R}_0^+ \cup \{\infty\}$. h is *admissible* if $h(s) \leq J_\Theta^*(s)$ holds for all states $s \in S_\Theta$, *goal-aware* if $h(s) = 0$ for goal states $s \in G_\Theta$, *consistent* if for every transition $\langle s, \ell, \delta \rangle \in T_\Theta$ it holds that $h(s) \leq C_\Theta(\ell) + \sum_{t \in S_\Theta} \delta(t) \cdot h(t)$ 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.

The probabilistic merge-and-shrink framework applies transformations on a factored representation of a TS, a so-called *annotated transition system* (ATS; Klößner et al. 2023). Adding a set of initial states for our developments, an ATS is a tuple $A = \langle S_A, L_A, C_A, E_A, D_A, T_A, I_A, G_A \rangle$. Here, E_A is a finite set of *label effects*, $D_A : L_A \to \mathit{Dist}(E_A)$ maps each label to a probability distribution over possible effects 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 $e \in supp(D_A(\ell))$ of ℓ with a successor state $\alpha(e) \in S_A$. An ATS is able to distinguish between different outcomes of an action leading to the same state. Dropping this information simplifies A to an ordinary TS $\Theta(A) := \langle S_A, L_A, C_A, T_{\Theta(A)}, I_A, G_A \rangle$ where $T_{\Theta(A)} := \{\langle s, \ell, t \mapsto \sum_{e \in \alpha^{-1}(t)} D_A(\ell)(e) \rangle \mid \langle s, \ell, \alpha \rangle \in T_A\}$. In $\Theta(A)$, the probability of a successor is the total probability of all label effects leading to this successor.

The framework maintains a *factored ATS*, which 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 label set L_F, effects E_F, effect probabilities D_F and label costs C_F. Such a factored ATS is an implicit representation of the synchronized product over its factors defined as $\bigotimes F := \langle \times_{i \in \mathcal{I}} S_{A_i}, L_F, C_F, E_F, D_F, T_{\bigotimes F}, \times_{i \in \mathcal{I}} I_{A_i}, \times_{i \in \mathcal{I}} G_{A_i} \rangle$, with $T_{\bigotimes F} := \{\langle \langle s_i \rangle_{i \in \mathcal{I}}, \ell, e \mapsto \langle \alpha_i(e) \rangle_{i \in \mathcal{I}} \rangle \mid \forall i \in \mathcal{I}.\, \langle s_i, \ell, \alpha_i \rangle \in T_{A_i}\}$. The factors representing an ATS of interest can be constructed from a probabilistic finite-domain representation (Trevizan, Thiébaux, and Haslum 2017).

Klößner et al. frame merge-and-shrink as a series of transformations from one factored ATS into another. A (non-annotated) TS transformation is a tuple $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$, where Θ is the *concrete TS*, Θ' is the *transformed TS*, $\sigma : S_\Theta \rightharpoonup S_{\Theta'}$ is a *state mapping* and $\lambda : L_\Theta \rightharpoonup L_{\Theta'}$ is a *label mapping*. τ induces the *transformation heuristic* $h^\tau(s) = J_{\Theta'}^*(\sigma(s))$ for $s \in dom(\sigma)$ and $h^\tau(s) = \infty$ else.

To prove properties like goal-awareness for h^τ, one can argue about structural properties of τ. To this end, one can extend the state mapping σ to a *state distribution* mapping $\sigma_{\mathit{Dist}} : \mathit{Dist}(dom(\sigma)) \to \mathit{Dist}(S_{\Theta'})$ defined by $\sigma_{\mathit{Dist}}(\delta) := s' \mapsto \sum_{s \in \sigma^{-1}(s')} \delta(s)$. Together, σ, λ and σ_{Dist} give rise to the *induced transition mapping* $ind_\tau : T_\Theta \cap (dom(\sigma) \times dom(\lambda) \times dom(\sigma_{\mathit{Dist}})) \to (S_{\Theta'} \times L_{\Theta'} \times \mathit{Dist}(S_{\Theta'}))$ given by $ind_\tau(\langle s, \ell, \delta \rangle) := \langle \sigma(s), \lambda(\ell), \sigma_{\mathit{Dist}}(\delta) \rangle$. With this, Klößner et al. define the following *conservativeness* properties:

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

Furthermore, they introduce the *inducedness* properties $\mathbf{IND_{S+L+C+T+G}}$ and the *refinedness* properties $\mathbf{REF_{C+T+G}}$, but we omit their detailed definitions here as we will not deal with them on a formal level. All of these properties are compositional: If two transformations $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ and $\tau' = \langle \Theta', \Theta'', \sigma', \lambda' \rangle$ satisfy the property, then their composition $\tau' \circ \tau := \langle \Theta, \Theta'', \sigma' \circ \sigma, \lambda' \circ \lambda \rangle$ also satisfies the property. It holds that for every conservative transformation τ, i.e., $\tau \in \mathbf{CONS_{S+L+C+T+G}}$, the heuristic h^τ is goal-aware, consistent and safe, and therefore admissible. On the other hand, if τ is refined, i.e., $\tau \in \mathbf{REF_{C+T+G}}$, then h^τ is pessimistic, i.e., $h^\tau \geq J^*_\Theta$. Lastly, if a transformation is *exact*, i.e., conservative and refined, then $h^\tau = J^*_\Theta$ is perfect.

Since the underlying TS is only implicitly represented by a factored ATS $F = \langle A_i \rangle_{i \in \mathcal{I}}$, merge-and-shrink transformations must be specified on this representation instead. 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* $\mathrm{Atom}[i, \alpha] := \alpha(s(i))$ for some factor index $i \in \mathcal{I}$ and $\alpha : S_{A_i} \rightharpoonup D$, or a *merge FM* $\mathrm{Merge}[\sigma_L, \sigma_R, \alpha] := \alpha(\sigma_L(s), \sigma_R(s))$ for the child FMs $\sigma_L : S_{\bigotimes F} \rightharpoonup D_L$, $\sigma_R : S_{\bigotimes F} \rightharpoonup D_R$ and $\alpha : D_L \times D_R \rightharpoonup D$. A *factored-to-factored* (F2F) 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 σ_j from F to 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 F2F 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 $\langle \Theta(\bigotimes F), \Theta(\bigotimes F'), [\![\Sigma]\!], \lambda \rangle$ between the represented transition systems of F and F'. Via this association, properties like $\mathbf{CONS_S}$ are said to hold for a factored transformation, if they hold for this corrresponding TS transformation.

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 behavior 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 (Fišer, Horčík, and Komenda 2020; Sievers and Helmert 2021). Here, we generalize these ideas to SSP heuristic search.

Before introducing our novel heuristic properties, we need some additional notation. 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 start and target states, respectively. We write $s \overset{\pi}{\rightsquigarrow} t$ if and only if $\Pr^\pi_s[Cyl(h)] > 0$ for some history $h \in Finish_\Theta(\{t\})$, and $s \rightsquigarrow t$ if and only if $s \overset{\pi}{\rightsquigarrow} t$ for some policy π. With this, 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 $Sols_\Theta(s, T) \neq \emptyset$. The set of states solvable for T is denoted $Solv_\Theta(T)$. There can be states which are neither solvable nor dead ends for T. Lastly, we say that a state s' 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 $s \overset{\pi}{\rightsquigarrow} s'$. 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 $s \overset{\pi}{\rightsquigarrow} s'$ implies $\pi \in Sols_\Theta(s', T)$. Yet, not every forward reachable and solvable state is necessarily alive.

The sets $Reach^{\rightarrow}_\Theta(S)$ and $Reach^{\leftarrow}_\Theta(T)$ can be computed by running a simple exhaustive forward/backwards exploration of Θ. $Solv_\Theta(T)$ can be computed by iteratively pruning dead ends (Baier and Katoen 2008). To this end, the projection of Θ onto a state set $K \subseteq S_\Theta$ is given by $\Theta|_K := \{K, L_\Theta, T_\Theta \cap (K \times L_\Theta \times \mathit{Dist}(K)), C_\Theta, I_\Theta \cap K, G_\Theta \cap K\}$. 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 $Alive_\Theta(S, T) = Reach^{\rightarrow}_{\Theta_k}(S)$.

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 new dead end s_2 and the transitions leading to s_2 (drawn with dashed lines) to obtain Θ_2. Finally, Θ_2 contains 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 solution for s_0 can reach these states. Note that s_9 can be reached by the history-dependent solution that first goes to s_9 from s_0 and behaves like an SD solution for s_9 afterwards for histories of length ≥ 1. However, an SD policy reaching s_9 from s_0 cannot be a solution, as it will produce an infinite cycle due to 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 $\Theta|_{\{s_0, s_3, s_4, s_9\}}$, this means we would not obtain a proper generalization 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.

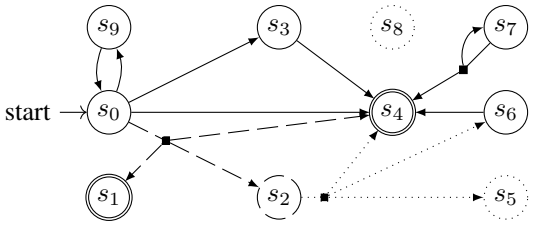

Figure 1: Transition system used in Example 1.

Definition 1. *Let Θ be a TS and let $h : S_\Theta \to \mathbb{R} \cup \{\infty\}$ be a heuristic. We say that h is* alive-

- *(i)* goal-aware *if $h(s) = 0$ for all $s \in Alive(\Theta) \cap G_\Theta$*
- *(ii)* consistent *if $h(s) \leq C_\Theta(\ell) + \sum_{t \in S_\Theta} \delta(t) \cdot h(t)$ for all $\langle s, \ell, \delta \rangle \in T_\Theta \cap Alive(\Theta) \times L_\Theta \times Dist(Alive(\Theta))$*
- *(iii)* safe *if $h(s) \neq \infty$ for all $s \in Alive(\Theta)$*
- *(iv)* admissible *if $h(s) \leq J^*_\Theta(s)$ for all $s \in Alive(\Theta)$*
- *(v)* perfect *if $h(s) = J^*_\Theta(s)$ for all $s \in Alive(\Theta)$ and $h(s') = \infty$ for all $s' \notin Solv(\Theta)$ such that there exists $\langle t, \ell, \delta \rangle \in T_\Theta$ with $t \in Alive(\Theta) \wedge s' \in supp(\delta)$*

As usual, the heuristic value $h(s) = \infty$ is used to signal the search to discard s from consideration. For alive-perfection (v), 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 estimate 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, and let h be the heuristic used in the search. If h is alive-safe, then $\mathcal{A}$ guarantees to return a solution for I, if one exists. If h is alive-admissible, then $\mathcal{A}$ guarantees to return an optimal solution for I, if a solution exists.*

Proof. Pruning non-alive states does by definition not affect the space of possible solutions, which suffices for the first part of the claim. Regarding the second part, if the algorithm guarantees to find an optimal solution with an admissible heuristic, the algorithm will also find an optimal solution if we make all suboptimal choices look worse than they actually are; intuitively, this can possibly only make the optimal choices look better. As h may be inadmissible only for states not part of any solution, h can hence only result in pessimistic estimations of suboptimal choices, concluding the proof. □

Our Theorem applies to most popular heuristic search algorithms for SSPs, including LAO* (Hansen and Zilberstein 2001), LRTDP (Bonet and Geffner 2003), and idual (Trevizan, Teichteil-Königsbuch, and Thiébaux 2017).

4 A Theory of Prune Transformations

In the following, we define prune transformations on a factored ATS and characterize them in the context of the heuristic properties introduced in the previous section. 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 set of* kept states*. The pruned ATS $A|_K$ for A and K is defined as $A|_K := \{K, L_A, C_A, E_A, D_A, T_{A|_K}, I_A \cap K, G_A \cap K\}$, where $T_{A|_K} := \{\langle s, \ell, \alpha \rangle \in T_A \mid s \in K \wedge \alpha(supp(D_A(\ell))) \subseteq K\}$.*

It is easy to see that $\Theta(A)|_K = \Theta(A|_K)$, i.e., it is unimportant whether we first prune states from an ATS and then translate it to its corresponding non-annotated transition system, or if we do this translation first and prune afterwards. With this, we formally define prune transformations on a factored ATS, which are transformations that prune states from a specific factor of the factored ATS.

Definition 3 (Prune Transformations)**.** *Let $F = \langle A_i \rangle_{i \in \mathcal{I}}$ be a factored ATS. Let $k \in \mathcal{I}$ be an index and let $K \subseteq S_{A_k}$. 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*

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

Since prune transformations essentially only discard some states (and their transitions), they are induced, refinable, and satisfy $\mathbf{CONS_{L+T+C+G}}$. However, they satisfy $\mathbf{CONS_S}$ only if $K = \emptyset$. In conclusion, they are generally not conservative, and can lead to inadmissible transformation heuristics. To nevertheless obtain sufficient criteria for admissibility and other heuristic properties, we replace $\mathbf{CONS_S}$ with weaker properties inspired by the classical planning theory.

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

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

The property $\mathbf{CONS_I}$ was omitted by Klößner et al. (2023) as it is not needed for reasoning about shrink and merge transformations or label reduction. All the transformations defined in their work satisfy $\mathbf{KEEP_G}$ and $\mathbf{CLOS}$ because they utilize a total state mapping, i.e., $dom(\sigma) = S_\Theta$. It is also easy to see that they satisfy $\mathbf{CONS_I}$.

The property $\mathbf{CLOS}$ requires 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 essentially guarantees that $dom(\sigma)$ is closed under predecessors, which matches the definition of the corresponding property in the classical case. The property $\mathbf{CLOS^{ALV}}$ is 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. $\mathbf{KEEP_G}$ requires that all goal states are kept, whereas $\mathbf{KEEP_G^{ALV}}$ preserves only alive goal states.

Policy Transformation

In classical planning, when reasoning over transformations, a re-occuring pattern is the need to simulate a concrete path in the original TS Θ with a corresponding path of the transformed TS Θ'. Such a simulation shows that a specific strategy of an agent, as modelled by the concrete path, can be simulated in Θ'. Before we can analyze the transformation properties introduced above, we need to generalize this idea for policies to be able to conduct proofs later on. All proofs for claims made in this sub-section can be found in our technical report (Klößner et al. 2024).

To this end, let $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ be a transformation. Analogous to the induced transition $ind_\tau(\mathfrak{t})$ of a concrete transition $\mathfrak{t}$, we define the *induced history* of $h = s_0\mathfrak{t}_0 \cdots \in Hist(\Theta)$ as $ind_\tau(s_0\mathfrak{t}_0 \dots) := \sigma(s_0)ind_\tau(\mathfrak{t}_0)\dots$, if $s_i \in dom(\sigma)$ and $\mathfrak{t}_i \in dom(ind_\tau)$ for all valid indices i, and undefined otherwise. This construction is essentially the simulation of a concrete path used in classical planning. To extend the simulation to policies we introduce the following construction, which uses the notation $Cyl(h\mathfrak{t}) := \bigcup_{s \in S_\Theta} Cyl(h\mathfrak{t}s)$.

Definition 5. *Let $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ be a transformation, let π be a policy for Θ and let $s \in S_\Theta$ be some starting state. The transformed policy $\pi_{\tau,s}$ of π for s is defined by*

$$\pi_{\tau,s}(h')(\mathfrak{t}') := \Pr_s^\pi[\bigcup_{\substack{h \in ind_\tau^{-1}(h') \\ \mathfrak{t} \in ind_\tau^{-1}(\mathfrak{t}')}} Cyl(h\mathfrak{t}) \mid \bigcup_{h \in ind_\tau^{-1}(h')} Cyl(h)]$$

if $\Pr_s^\pi[\bigcup_{h \in ind_\tau^{-1}(h')} Cyl(h)] > 0$*, else* $\pi_{\tau,s}(h')(\mathfrak{t}') := 0$.

Given a history h' and transition $\mathfrak{t}'$, the construction above calculates the conditional probability that π selects *some* concrete transition $\mathfrak{t} \in ind_\tau^{-1}(\mathfrak{t}')$ in the current time step, under the condition that a concrete history $h \in ind_\tau^{-1}(h')$ was observed. Note that these probabilities depend on the initial state s from which π is run. If no concrete history $h \in ind_\tau^{-1}(h')$ is possible, the policy terminates.

This construction has very convenient properties. Firstly, $\pi_{\tau,s}$ generates a prefix h' in Θ' with the same probability that π generates *some* matching prefix $h \in ind_\tau^{-1}(h')$, if both are executed from the starting states $\sigma(s)$ and s respectively. Moreover, the probability that $\pi_{\tau,s}$ produces some finite execution h' matches the probability that π produces some corresponding finite execution $h \in ind_\tau^{-1}(h')$, or an execution that starts with such a history and then continues along a non-induced transition. Formally, we have the following Lemma.

Lemma 1. *Let $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ be a transformation, let π be a policy and let $s \in S_\Theta$ be a starting state with $s \in dom(\sigma)$. For every abstract history $h' \in Hist(\Theta')$:*

$$\Pr_{\sigma(s)}^{\pi_{\tau,s}}[Cyl(h')] = \Pr_s^\pi\left[\bigcup_{h \in ind_\tau^{-1}(h')} Cyl(h)\right] \quad (1)$$

$$\Pr_{\sigma(s)}^{\pi_{\tau,s}}[h'] = \Pr_s^\pi\left[ind_\tau^{-1}(h')\right] + \Pr_s^\pi\left[\bigcup_{\substack{h \in ind_\tau^{-1}(h') \\ \mathfrak{t} \notin dom(ind_\tau)}} Cyl(h\mathfrak{t})\right] \quad (2)$$

Importantly, the transformed policy can be used to simulate a solution in Θ within Θ'. For each concrete state that π reaches, $\pi_{\tau,s}$ will then reach the corresponding abstract state. To this end, we need to impose mild conservativeness assumptions, and require that all states reachable by the original policy are kept by the transformation.

Proposition 1. *Let $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle \in$ **CONS**$_{\mathbf{L+T}}$ and let π be a policy. For all states $s \in S_\Theta$ with $Reach^{\rightarrow}_{\Theta,\pi}(s) \subseteq dom(\sigma)$ and every set of target states $T \subseteq S_\Theta$:*

(i) $Reach^{\rightarrow}_{\Theta',\pi_{\tau,s}}(\sigma(s)) = \sigma(Reach^{\rightarrow}_{\Theta,\pi}(s))$ *and*

(ii) If $\pi \in Sols_\Theta(s, T)$*, then* $\pi_{\tau,s} \in Sols_{\Theta'}(\sigma(s), \sigma(T))$.

Compositionality & Heuristic Guarantees

The construction we just introduced will be key for the proofs given in the rest of the paper. First, we use it to prove compositionality of the properties defined in Definition 4, so they can be reasoned with in the context of the merge-and-shrink framework. Since these properties are extensions of their respective classical properties, we similarly have to assume mild side conditions to prove compositionality.

Theorem 2. *Let $\tau = \langle \Theta, \Theta', \sigma, \lambda \rangle$ and $\tau' = \langle \Theta', \Theta'', \sigma', \lambda' \rangle$ be two transformations. For the following properties* **X***, if $\tau, \tau' \in$* **X***, then $\tau' \circ \tau \in$* **X***, under the assumption that τ satisfies the following additional side requirements:*

(i) **CLOS** *requires* **CONS**$_{\mathbf{L+T}}$

(ii) **CLOS**$^{\mathbf{ALV}}$ *requires* **CONS**$_{\mathbf{L+T+I}}$

(iii) **KEEP**$_{\mathbf{G}}$ *requires* **CONS**$_{\mathbf{G}}$

(iv) **KEEP**$^{\mathbf{ALV}}_{\mathbf{G}}$ *requires* **CLOS**$^{\mathbf{ALV}}$ **+ CONS**$_{\mathbf{L+T+I+G}}$

Proof. For statement (iii), see Sievers and Helmert (2021).

Statement (i) Let $s \in Alive_\Theta(S_\Theta, dom(\sigma' \circ \sigma))$. Then there is a policy π solving s for $dom(\sigma' \circ \sigma) \subseteq dom(\sigma)$. Then we also have $Reach^{\rightarrow}_{\Theta,\pi}(s) \subseteq Alive_\Theta(S_\Theta, dom(\sigma)) \subseteq dom(\sigma)$ due to **CLOS**. Applying Proposition 1 with the assumption $\tau \in$ **CONS**$_{\mathbf{L+T}}$, we conclude that $\pi_{\tau,s}$ solves $\sigma(s)$ for the target states $\sigma(dom(\sigma' \circ \sigma)) \subseteq dom(\sigma')$. Thus, $\sigma(s) \in Alive_\Theta(S_{\Theta'}, dom(\sigma'))$. With $\tau' \in$ **CLOS**, we obtain $\sigma(s) \in dom(\sigma')$ and ultimately $s \in dom(\sigma' \circ \sigma)$.

Statement (ii) Let $s \in Alive_\Theta(I_\Theta, dom(\sigma' \circ \sigma))$. Then there is a policy π with $s_0 \overset{\pi}{\rightsquigarrow} s$ that solves an initial state $s_0 \in I_\Theta$ for $dom(\sigma' \circ \sigma) \subseteq dom(\sigma)$. We also have $Reach^{\rightarrow}_{\Theta,\pi}(s_0) \subseteq Alive_\Theta(I_\Theta, dom(\sigma)) \subseteq dom(\sigma)$ due to $\tau \in$ **CLOS**$^{\mathbf{ALV}}$. Applying Proposition 1 with the assumption $\tau \in$ **CONS**$_{\mathbf{L+T}}$, we conclude that π_{τ,s_0} solves $\sigma(s_0)$ for $\sigma(dom(\sigma' \circ \sigma)) \subseteq dom(\sigma')$, and $\sigma(s_0) \overset{\pi_{\tau,s_0}}{\rightsquigarrow} \sigma(s)$. Since we have $\sigma(s_0) \in I_{\Theta'}$ due to **CONS**$_{\mathbf{I}}$, ultimately $\sigma(s) \in Alive_\Theta(I_{\Theta'}, dom(\sigma')) \subseteq dom(\sigma')$ by **CLOS**$^{\mathbf{ALV}}$.

Statement (iv) Let $s \in Alive(\Theta) \cap G_\Theta$. Since $s \in Alive(\Theta)$, there is an initial state $s_0 \in I_\Theta$ and $\pi \in Sols_\Theta(s_0)$ with $s_0 \overset{\pi}{\rightsquigarrow} s$. We have $Reach^{\rightarrow}_{\Theta,\pi}(s_0) \subseteq Alive(\Theta) \subseteq Alive_\Theta(I_\Theta, dom(\sigma)) \subseteq dom(\sigma)$ due to **KEEP**$^{\mathbf{ALV}}_{\mathbf{G}}$ and **CLOS**$^{\mathbf{ALV}}$. Applying Proposition 1 with the assumption $\tau \in$ **CONS**$_{\mathbf{L+T}}$, we obtain that π_{τ,s_0} solves $\sigma(s_0)$ and $\sigma(s_0) \overset{\pi_{\tau,s_0}}{\rightsquigarrow} \sigma(s)$. Since $\sigma(s_0) \in I_{\Theta'}$ by **CONS**$_{\mathbf{I}}$, we have $\sigma(s) \in Alive(\Theta')$. Finally, $\sigma(s) \in G_{\Theta'}$ due to **CONS**$_{\mathbf{G}}$. □

Next, 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. (2023), h^τ is goal-aware, consistent and safe if τ is conservative. The new transformation properties now replace $\mathbf{CONS_S}$ so that h^τ still remains with these properties, or the corresponding weaker variants as defined in Definition 4.

Theorem 3. *Let τ be a transformation. The transformation heuristic h^τ satisfies the following heuristic properties, if τ satisfies the corresponding transformation properties:*

(i) goal-awareness: $\mathbf{CONS_G} + \mathbf{KEEP_G}$
(ii) consistency: $\mathbf{CONS_{L+C+T}} + \mathbf{CLOS}$
(iii) safety: $\mathbf{CONS_{L+T+G}} + \mathbf{CLOS} + \mathbf{KEEP_G}$
(iv) alive-goal-awareness: $\mathbf{CONS_G} + \mathbf{KEEP_G^{ALV}}$
(v) alive-cons.: $\mathbf{CONS_{L+C+T}} + \mathbf{CLOS^{ALV}} + \mathbf{KEEP_G^{ALV}}$
(vi) alive-safety: $\mathbf{CONS_{L+T+G}} + \mathbf{CLOS^{ALV}} + \mathbf{KEEP_G^{ALV}}$

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

Statement (ii) Let $s \in S_\Theta$ and $\langle s, \ell, \delta \rangle \in T_\Theta(s)$. We assume $\mathit{supp}(\delta) \subseteq \mathit{dom}(\sigma)$, as otherwise the right-hand side of the consistency inequation evaluates to ∞, making it trivial. We then have $s \in \mathit{dom}(\sigma)$ as well due to $\mathbf{CLOS}$, since we can reach $\mathit{dom}(\sigma)$ with certainty from s via this transition. We also have $\ell \in \mathit{dom}(\lambda)$ due to $\mathbf{CONS_L}$. Due to $\mathbf{CONS_T}$, it follows that $\langle \sigma(s), \lambda(\ell), \sigma_{\mathit{Dist}}(\delta) \rangle \in T_{\Theta'}(\sigma(s))$. The following inequality concludes the proof, where the first inequality follows because $J^*_{\Theta'}$ is consistent, and the second inequality follows by $\mathbf{CONS_C}$ and the definition of σ_{Dist}.

$$\begin{aligned} J^*_{\Theta'}(\sigma(s)) &\leq C_\Theta(\lambda(\ell)) + \sum_{t' \in S_{\Theta'}} \sigma_{\mathit{Dist}}(\delta)(t) \cdot J^*_{\Theta'}(t') \\ &\leq C_\Theta(\ell) + \sum_{t' \in S_{\Theta'}} \sum_{t \in \sigma^{-1}(t')} \delta(t) \cdot J^*_{\Theta'}(t') \\ &= C_\Theta(\ell) + \sum_{t \in S_\Theta} \delta(t) \cdot J^*_{\Theta'}(\sigma(t)) \end{aligned}$$

Statement (iii) 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. 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 $h^\tau(s) \neq \infty$.

Statement (v) Let $\langle s, \ell, \delta \rangle \in T_\Theta \cap (\mathit{Alive}(\Theta) \times L_\Theta \times \mathit{Dist}(\mathit{Alive}(\Theta)))$. We conclude $s \in \mathit{dom}(\sigma)$ and $\mathit{supp}(\delta) \subseteq \mathit{dom}(\sigma)$ using (B). The rest of the proof is analogous to (ii).

Statement (vi) Analogous to (iii), substituting $\mathit{Solv}(\Theta)$ with $\mathit{Alive}(\Theta)$ and (A) with (B). □

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 $\mathbf{CLOS}$, 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 3. Furthermore, composing such transformations with each other or with exact transformations also yields perfect or alive-perfect heuristics by Theorem 2.

Theorem 4. *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 $K \subseteq S_{A_k}$. Then:*

(i) $\tau \in \mathbf{CLOS} \cap \mathbf{KEEP_G}$ if $K = \mathit{Reach}^{\leftarrow}(\Theta(A_k))$
(ii) $\tau \in \mathbf{CLOS^{ALV}} \cap \mathbf{KEEP_G^{ALV}}$ if $K = \mathit{Reach}^{\rightarrow}(\Theta(A_k))$

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 $\langle \Theta, \Theta', \sigma, \mathit{id} \rangle$, where $s \in \mathit{dom}(\sigma)$ if and only if $s(k) \in K$.

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

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

Theorem 4 establishes two reachability-based prune transformations. 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 transformations at the expense of additional computation time. Note that we can simulate a prune transformation on factor A_k with $K = \mathit{Solv}(\Theta(A_k))$, keeping only solvable states, via a series of prune transformations with $K = \mathit{Reach}^{\leftarrow}(\Theta(A_k))$. Likewise, to simulate $K = \mathit{Alive}(\Theta(A_k))$, we can apply a prune transformation with $K = \mathit{Solv}(\Theta(A_k))$, followed by a prune transformation with $K = \mathit{Reach}^{\rightarrow}(\Theta(A_k))$. By compositionality (Theorem 2) 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 $K \subseteq S_{A_k}$. Then:*

(i) $\tau \in \mathbf{CLOS} \cap \mathbf{KEEP_G}$ if $K = \mathit{Solv}(\Theta(A_k))$
(ii) $\tau \in \mathbf{CLOS^{ALV}} \cap \mathbf{KEEP_G^{ALV}}$ if $K = \mathit{Alive}(\Theta(A_k))$

5 Experiments

We conclude with an empirical evaluation of M&S heuristics for SSPs. We implemented them in our version of Probabilistic Fast Downward (Steinmetz, Hoffmann, and Buffet 2016) starting with the classical Fast Downward (Helmert 2006; Sievers 2018) implementation and adjusting it only where necessary. The top-level algorithm takes a maximal number of abstract states M as a parameter, as well as a merge, shrink, prune and a label reduction strategy. Initially, the atomic factors of the given probabilistic planning task are constructed, with uninteresting states discarded by the prune strategy. In each iteration of the algorithm, two factors to be merged are selected by the merge strategy. Afterwards, label reduction is applied, and both factors are shrunk by the shrink strategy, such that the merge results in a new factor with at most M states. Afterwards, both factors are merged. After each merge, the prune strategy discards uninteresting states from the resulting factor. The process continues until only one factor remains and has no time limit. The overall factored transformation is maintained and the corresponding heuristic is extracted by computing J^* for the final factor using a variant of topological value iteration (Dai et al. 2011) that supports zero cost actions and unsolvable states.

Regarding shrink strategies, we implemented ATS bisimulation shrinking as described by Klößner et al. (2023). To compute an ATS bisimulation, we follow the classical Fast Downward implementation of bisimulation shrinking, which runs a partition refinement algorithm. Initially, all states with the same J^* value are put into the same equivalence class, with goal states being in a distinct class. To this end, J^* is maintained for all factors, analogous to h^* in the classical implementation. The procedure iteratively splits equivalence classes until the underlying relation is an ATS bisimulation. To respect the state limit, the refinement stops early if splitting an equivalence class would result in more than M classes, and does not split the class at all in this case. 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 a prune strategy that keeps only 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 their usage in a heuristic search algorithm to compute an optimal policy for a given stochastic shortest-path problem. We focus on improved LAO* (Hansen and Zilberstein 2001) as the heuristic search algorithm in our experiments, extended with a trap-elimination procedure (Kolobov et al. 2011) to support zero-cost actions.

The experiments were run with Downward Lab (Seipp et al. 2017) 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, Seipp, and Steinmetz (2023), containing 9 probabilistic PDDL domains with 20 problems each, some of them containing traps or unsolvable states.

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 variants of M&S with an abstract state limit of 50000, which turned out to be an effective limit in preliminary experiments. Both variants use their respective notion of exact bisimulation shrinking. We selected two different merge stategies: The reverse level linear merge strategy (Nissim, Hoffmann, and Helmert 2011) which orders variables closer to the root of the causal graph first and the state-of-the-art strategy SCC-DFP from classical planning (Sievers, Wehrle, and Helmert 2016). For label reduction, we use the exact label reduction strategy based on Θ-combinability (Sievers, Wehrle, and Helmert 2014), respectively (A, ϵ)-combinability. To this end, 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. 2b, since the factored representation contains more information that needs to be maintained during each transformation, and solving an SSP is considerably more expensive than solving a classical planning problem. Despite being about an order of magnitude slower 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 similarly becomes more favorable 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 search problems. The maintenance of J^* for all factors consistently contributes to the runtime, taking a moderate amount of time. 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.

Domains	Linear Reverse Level						SCC-DFP											
	$h^{\text{M\&S}}_{\text{Aliv}}$	$h^{\text{M\&S}}_{\text{All}}$	$h^{\text{M\&S}}_{\text{Solv}}$	$h^{\text{dM\&S}}_{\text{Aliv}}$	$h^{\text{dM\&S}}_{\text{All}}$	$h^{\text{dM\&S}}_{\text{Solv}}$	$h^{\text{M\&S}}_{\text{Aliv}}$	$h^{\text{M\&S}}_{\text{All}}$	$h^{\text{M\&S}}_{\text{Solv}}$	$h^{\text{dM\&S}}_{\text{Aliv}}$	$h^{\text{dM\&S}}_{\text{All}}$	$h^{\text{dM\&S}}_{\text{Solv}}$	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.

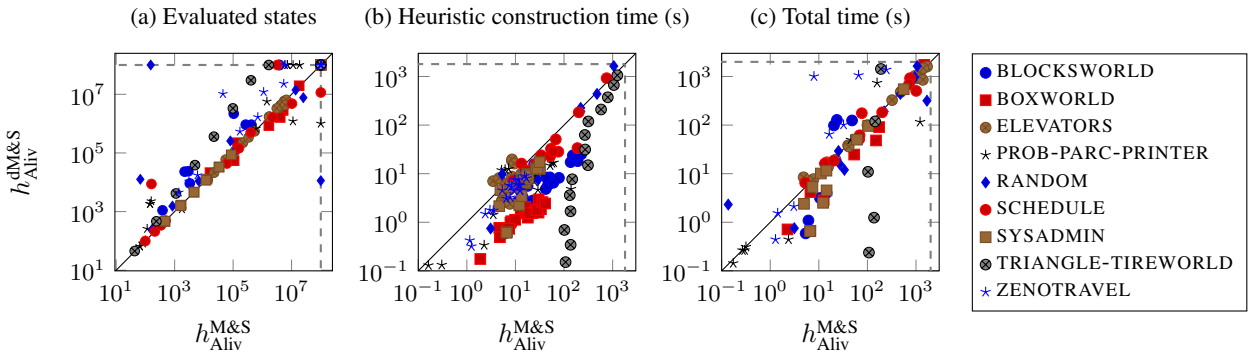

Figure 2: Probabilistic $h^{\text{M\&S}}_{\text{Aliv}}$ (x-axis) versus deterministic $h^{\text{dM\&S}}_{\text{Aliv}}$ (y-axis), both using the SCC-DFP merge strategy.

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 (Klößner and Hoffmann 2021), SSP cartesian abstraction heuristics (Klößner, Seipp, and Steinmetz 2023) and the occupation measure heuristic h^{roc} (Trevizan, Thiébaux, and Haslum 2017). We construct single-abstraction PDB and cartesian abstraction heuristics via policy-based counter-example guided abstraction refinement, with a limit of 10000 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 PDB abstractions constructed via hill-climbing search over the space of PDB collections ($h^{\text{PDB}}_{\text{HC}}$), following Klößner et al. (2022). We run hill-climbing for 180 seconds, with a collection size limit of 10 million abstract states.

The coverage is again reported in Table 1. Overall, all SSP M&S heuristics we have tested 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 individual M&S configurations. This leads to three timeouts of our best configuration in problems even solved by blind search. Moreover, $h^{\text{PDB}}_{\text{HC}}$ makes use of multiple abstractions. The combination of multiple M&S heuristics has already been considered for classical planning (Sievers et al. 2020), 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 generalize those considered by Sievers and Helmert (2021) 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 (Katz, Hoffmann, and Helmert 2012; Hoffmann, Kissmann, and Torralba 2014; Sievers, Wehrle, and Helmert 2016), or the construction of an cost-partitioned ensemble of M&S heuristics (Sievers et al. 2020).

References

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

Bertsekas, D. P.; and Tsitsiklis, J. N. 1991. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research*, 16: 580–595.

Bonet, B.; and Geffner, H. 2003. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *Proc. ICAPS'03*, 12–21.

Dai, P.; Mausam; Weld, D. S.; and Goldsmith, J. 2011. Topological Value Iteration Algorithms. *Journal of Artificial Intelligence Research*, 42: 181–209.

Edelkamp, S. 2001. Planning with Pattern Databases. In *Proc. ECP'01*, 13–24.

Fišer, D.; Horčík, R.; and Komenda, A. 2020. Strengthening Potential Heuristics with Mutexes and Disambiguations. In *Proc. ICAPS'20*, 124–133.

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

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *Proc. AAAI'07*, 1007–1012.

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

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible Abstraction Heuristics for Optimal Sequential Planning. In *Proc. ICAPS'07*, 176–183.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. 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.

Helmert, M.; Röger, G.; and Sievers, S. 2015. On the Expressive Power of Non-Linear Merge-and-Shrink Representations. In *Proc. ICAPS'15*, 106–114.

Hoffmann, J.; Kissmann, P.; and Torralba, Á. 2014. "Distance"? Who Cares? Tailoring Merge-and-Shrink Heuristics to Detect Unsolvability. In *Proc. ECAI'14*, 441–446.

Katz, M.; Hoffmann, J.; and Helmert, M. 2012. How to Relax a Bisimulation? In *Proc. ICAPS'12*, 101–109.

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

Klößner, T.; Seipp, J.; and Steinmetz, M. 2023. Cartesian Abstractions and Saturated Cost Partitioning in Probabilistic Planning. In *Proc. ECAI'23*, 1272–1279.

Klößner, T.; Steinmetz, M.; Torralba, À.; and Hoffmann, J. 2022. Pattern Selection Strategies for Pattern Databases in Probabilistic Planning. In *Proc. ICAPS'22*, 184–192.

Klößner, T.; Torralba, Á.; Steinmetz, M.; and Hoffmann, J. 2021. Pattern Databases for Goal-Probability Maximization in Probabilistic Planning. In *Proc. ICAPS'21*, 80–89.

Klößner, T.; Torralba, Á.; Steinmetz, M.; and Sievers, S. 2023. A Theory of Merge-and-Shrink for Stochastic Shortest Path Problems. In *Proc. ICAPS'23*, 203–211.

Klößner, T.; Torralba, Á.; Steinmetz, M.; and Sievers, S. 2024. Merge-and-Shrink for Stochastic Shortest-Path Problems with Pruning Transformations — Technical Report. Available at https://fai.cs.uni-saarland.de/kloessner/papers/kloessner-etal-hsdip24-tr.pdf.

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic Search for Generalized Stochastic Shortest Path MDPs. In *Proc. ICAPS'11*.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011. Computing Perfect Heuristics in Polynomial Time: On Bisimulation and Merge-and-Shrink Abstraction in Optimal Planning. In Walsh, T., ed., *Proc. the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, 1983–1990. AAAI Press/IJCAI.

Seipp, J.; and Helmert, M. 2018. Counterexample-Guided Cartesian Abstraction Refinement for Classical Planning. *Journal of Artificial Intelligence Research*, 62: 535–577.

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

Sievers, S. 2018. Merge-and-Shrink Heuristics for Classical Planning: Efficient Implementation and Partial Abstractions. In Bulitko, V.; and Storandt, S., eds., *Proc. the 11th Annual Symposium on Combinatorial Search (SOCS'18)*, 90–98. AAAI Press.

Sievers, S.; and Helmert, M. 2021. Merge-and-Shrink: A Compositional Theory of Transformations of Factored Transition Systems. *Journal of Artificial Intelligence Research*, 71: 781–883.

Sievers, S.; Pommerening, F.; Keller, T.; and Helmert, M. 2020. Cost-Partitioned Merge-and-Shrink Heuristics for Optimal Classical Planning. In *Proc. IJCAI'20*, 4152–4160.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized Label Reduction for Merge-and-Shrink Heuristics. In *Proc. AAAI'14*, 2358–2366.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An Analysis of Merge Strategies for Merge-and-Shrink Heuristics. In *Proc. ICAPS'16*, 294–298.

Steinmetz, M.; Hoffmann, J.; and Buffet, O. 2016. Goal Probability Analysis in MDP Probabilistic Planning: Exploring and Enhancing the State of the Art. *Journal of Artificial Intelligence Research*, 57: 229–271.

Trevizan, F. W.; Teichteil-Königsbuch, F.; and Thiébaux, S. 2017. Efficient solutions for Stochastic Shortest Path Problems with Dead Ends. In Elidan, G.; Kersting, K.; and Ihler, A. T., eds., *Proc. the 33rd Conference on Uncertainty in Artificial Intelligence (UAI'17)*. AUAI Press.

Trevizan, F. W.; Thiébaux, S.; and Haslum, P. 2017. Occupation Measure Heuristics for Probabilistic Planning. In *Proc. ICAPS'17*, 306–315.