Domain-Independent Heuristics in Probabilistic Planning – Dissertation Abstract

Thorsten Klößner

Saarland University, Saarland Informatics Campus
Foundations of Artificial Intelligence Group
kloessner@cs.uni-saarland.de
Supervisor: Jörg Hoffmann

Abstract

It has been almost two decades since MDP heuristic search algorithms have been developed. These algorithms guarantee to find an optimal partial policy for the initial state for several optimization objectives without necessarily expanding the entire state space, if provided with a heuristic that provides optimistic objective value estimates for all states. While a large set of such domain-independent heuristic families is available in classical planning, the same cannot be said about probabilistic planning. So far, with the exception of occupation measure heuristics for (constrained) Stochastic Shortest-Path Problems, most of the research on domain-independent heuristic construction consists of using a classical heuristic on the all-outcomes determinization of the planning problem, in which the probabilistic effect of an action can be chosen at will. Because this approach is agnostic to the uncertainty in the underlying problem, these heuristics are often uninformative. In this thesis, I will develop new domain-independent heuristics for probabilistic planning which take the probabilistic nature of the problem into account. To this end, the main focus of the thesis lies in developing the foundations of abstraction heuristics for probabilistic planning, in particular Pattern Database heuristics and Merge-and-Shrink heuristics.

Introduction

AI planning is a long-standing discipline in artificial intelligence which deals with the automatic deduction of strategies for autonomous agents. In classical planning, the simplest form of planning, a single agent acts inside a fully observable and deterministic environment. Probabilistic Planning relaxes these assumptions to allow stochastic environments, where an action leads to one of multiple possible outcomes, each occurring with an associated probability that is known a priori. In this thesis, I will focus on fully observable problems, where problems are commonly modelled as a Markov Decision Process (MDP). In this setting, the behaviour of the agent is typically specified by a function from states to actions, called a *policy*.

There exist various optimization criteria can be considered to specify the desired behaviour of an agent. In this thesis, I focus on two settings in particular. In the *MaxProb* setting, we want to find a policy that maximizes the probability to reach a set of goal states when starting in the initial state of the problem. On the other hand, Stochastic Shortest-Path Problems (SSPs, Bertsekas and Tsitsiklis (1991)) associate each action application with a real-valued cost. The objective of the agent is to reach a set of goal states with probability one in the limit, while minimizing the expected accumulated cost to do so.

There exist a plethora of algorithms to solve MDPs both optimally and approximately for these settings. Heuristic search algorithms for SSPs (e.g. Hansen and Zilberstein (2001), Bonet and Geffner (2003), Trevizan et al. (2017)) have the potential to prevent the exhaustive generation of the whole state space of the problem. These algorithms require an *admissible* heuristic to ensure optimality, i.e. a function which underestimates the real minimal expected cost-to-goal of a state. These algorithms can also be extended to MaxProb (Kolobov et al. 2011), where they require an upper-bounding heuristic on the maximal goal probability of a state instead.

Although substantial effort has been invested into the development of MDP heuristic search algorithms themselves, research regarding admissible heuristics which can be supplied to these algorithms is, to this day, rather sparse. Most of the research utilizes the all-outcomes determinization (Yoon, Fern, and Givan 2007), in which the agent can simply choose the probabilistic outcome of an action. With the help of this transformation, any classical heuristic can be used to guide the search by delegating to the determinization.

While the determinization-based approach enables the use of a large arsenal of classical planning heuristics, these heuristics are often not very informative, since the uncertainty in the problem is completely ignored. On the other hand, occupation measure heuristics for (constrained) SSPs (Trevizan, Thiébaux, and Haslum 2017) actually make use of the probabilistic information. These LP-based heuristics can be seen as extensions of operator-counting heuristics (Pommerening et al. 2014) to probabilistic planning. Trevizan, Thiébaux, and Haslum experimental evaluations show that these heuristics lead to greatly decreased search effort compared to determinization-based heuristics. As of today, these heuristics are considered state-of-the-art.

In this thesis, I develop novel domain-independent heuristics for probabilistic planning which are not agnostic to the uncertainty of the problem. To this end, I extend several families of abstraction heuristics from classical planning to probabilistic planning, including Pattern Database heuristics (Korf 1997; Haslum et al. 2007; Pommerening,

Röger, and Helmert 2013), and Merge-and-Shrink heuristics (Helmert, Haslum, and Hoffmann 2007; Nissim, Hoffmann, and Helmert 2011; Helmert et al. 2014). Apart from an experimental evaluation, I investigate the theoretical relationships with previous heuristics, in particular with classical abstraction heuristics on the determinization as well as occupation measure heuristics.

Preliminaries

I consider probabilistic planning problems with full observability in the context of different optimization objectives. This thesis abstract focuses primarily on the MaxProb objective, which prioritizes goal probability maximization, as well as Stochastic Shortest-Path Problems (SSPs, Bertsekas and Tsitsiklis (1991)). To represent both settings in a uniform manner, the underlying probabilistic model will be kept separate from the considered optimization objective.

MDPs and Optimization Objectives As the baseline model, we define a Markov Decision Process (MDP) as a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_{\mathcal{I}} \rangle$. $\mathcal{S}$ is the finite, non-empty set of *states*, $\mathcal{A}$ is a finite, non-empty set of *actions*, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the *transition probability function* and $s_{\mathcal{I}} \in \mathcal{S}$ is the *initial state* of the problem. For any state-action pair $\langle s, a \rangle \in \mathcal{S} \times \mathcal{A}$, either $\sum_{t \in \mathcal{S}} \mathcal{T}(s, a, t) = 1$ (a is enabled in s) or $\mathcal{T}(s, a, t) = 0$ for all $t \in \mathcal{S}$ (a is disabled in s). The set of actions enabled in s is denoted $\mathcal{A}(s)$. We assume that $\mathcal{A}(s) \neq \emptyset$ for all states. We can easily introduce artificial self-loops to achieve this. Finally, a policy is a mapping $\pi : \mathcal{S} \to \mathcal{A}$ with $\pi(s) \in \mathcal{A}(s)$ for every state $s \in \mathcal{S}$.

The MaxProb optimization objective is specified by a set of goal states $\mathcal{S}_{\mathcal{G}} \subseteq \mathcal{S}$. The semantics of a policy in presence of this optimization objective is given by the policy state value function $\mathcal{V}^{\pi}_{\text{MP}} : \mathcal{S} \to [0, 1]$, where $\mathcal{V}^{\pi}_{\text{MP}}(s)$ represents the probability to reach the goal when starting in the state s and following policy π. It is defined as the (point-wise) *least* solution of the equation system

$$\mathcal{V}^{\pi}_{\text{MP}}(s) = \begin{cases} 1 & s \in \mathcal{S}_{\mathcal{G}}, \\ \sum_{t \in \mathcal{S}} \mathcal{T}(s, a, t) \mathcal{V}^{\pi}_{\text{MP}}(t) & s \notin \mathcal{S}_{\mathcal{G}}. \end{cases}$$

The optimal state value function $\mathcal{V}^{*}_{\text{MP}}$ is defined as $\mathcal{V}^{*}_{\text{MP}}(s) := \max_{\pi} \mathcal{V}^{\pi}_{\text{MP}}(s)$. A policy $\pi^{\star}$ is optimal if $\mathcal{V}^{\pi^{\star}}_{\text{MP}} = \mathcal{V}^{*}_{\text{MP}}$. For MaxProb, an optimal policy always exists. Moreover, we say that π is an s-proper policy, if $\mathcal{V}^{\pi}_{\text{MP}}(s) = 1$. If π is s-proper for all s, then π is proper.

The optimization objective considered for SSPs is given by a set of goal states $\mathcal{S}_{\mathcal{G}} \subseteq \mathcal{S}$ and an action cost function $c : \mathcal{A} \to \mathbb{R}$. This objective makes two additional assumptions: (i) There exists a proper policy and (ii) Every improper policy eventually accumulates infinite cost[1]. The policy state value function $\mathcal{V}^{\pi}_{\text{SSP}} : \mathcal{S} \to \mathbb{R}$ is only defined for proper policies π for this objective. $\mathcal{V}^{\pi}_{\text{SSP}}(s)$ gives the expected accumulated cost until the goal is reached when starting in state s and acting according to π. It is the unique solution of the equation system

$$\mathcal{V}^{\pi}_{\text{SSP}}(s) = \begin{cases} 0 & s \in \mathcal{S}_{\mathcal{G}} \\ c(\pi(s)) + \sum_{t \in \mathcal{S}} \mathcal{T}(s, a, t) \mathcal{V}^{\pi}_{\text{SSP}}(t) & s \notin \mathcal{S}_{\mathcal{G}} \end{cases}$$

The optimal state value function $\mathcal{V}^{*}_{\text{SSP}}$ is defined by $\mathcal{V}^{*}_{\text{SSP}}(s) := \min_{\pi \text{ proper}} \mathcal{V}^{\pi}_{\text{SSP}}(s)$. Analogously, a policy $\pi^{\star}$ is optimal if $\mathcal{V}^{\pi^{\star}}_{\text{SSP}} = \mathcal{V}^{*}_{\text{SSP}}$ and always exists.

Probabilistic SAS$^+$ Tasks The planning problem is specified as a probabilistic SAS$^+$ task (Trevizan, Thiébaux, and Haslum 2017), except that the cost function is omitted since it is not needed for MaxProb. A probabilistic SAS$^+$ task is a tuple $\langle \mathcal{V}, \mathcal{A}, s_{\mathcal{I}}, \mathcal{G} \rangle$. $\mathcal{V}$ denotes the *state variables*, where each $v \in \mathcal{V}$ is associated with a finite domain $\mathcal{D}(v)$ of at least two values. A *partial state* is a partial function $s : \mathcal{V} \rightharpoonup \bigcup_{v \in \mathcal{V}} \mathcal{D}(v)$ with $s(v) \in \mathcal{D}(v)$ if defined. We denote the variables on which s is defined by $\mathcal{V}(s)$. s is a *state* if $\mathcal{V}(s) = \mathcal{V}$. The set of states of a probabilistic SAS$^+$ task Π is denoted $\mathcal{S}(\Pi)$. For a set of variables $P \subseteq \mathcal{V}$ and partial state s, we denote by $s[P]$ the *projection of s onto P* and define the set $S[P] := \{s[P] \mid s \in S\}$. We say *$s$ subsumes* t, written $t \subseteq s$, if $\mathcal{V}(s) \subseteq \mathcal{V}(t)$ and $s[\mathcal{V}(s)] = t[\mathcal{V}(s)]$. The *application* of partial state e onto partial state s is defined by $\text{appl}(s, e)(v) = e(v)$ if $v \in \mathcal{V}(e)$ and $s(v)$ otherwise. $\mathcal{A}$ is the set of *actions*. An action a specifies its *precondition* $\text{pre}(a)$, and a probability distribution $\Pr_a$ over effects, where an effect is a partial state. The possible effects of a are denoted $\text{Eff}(a) := \{e \mid \Pr_a(e) > 0\}$. Lastly, the *initial state* $s_{\mathcal{I}}$ is a state and the *goal* $\mathcal{G}$ is a partial state.

A probabilistic SAS$^+$ task $\Pi = \langle \mathcal{V}, \mathcal{A}, s_{\mathcal{I}}, \mathcal{G} \rangle$ induces the MDP $\langle \mathcal{S}(\Pi), \mathcal{A}, \mathcal{T}, s_{\mathcal{I}} \rangle$ where $\mathcal{T}(s, a, t)$ is defined as 0 if $\text{pre}(a) \not\subseteq s$ and by

$$\mathcal{T}(s, a, t) := \sum_{\substack{e \in \text{Eff}(a) \text{ s.t.} \\ \text{appl}(s,e) = t}} \Pr_a(e)$$

otherwise. The set of goal states for the MaxProb and SSP objective is given by $\mathcal{S}_{\mathcal{G}} = \{s \mid s \subseteq \mathcal{G}\}$.

Heuristics A heuristic h returns an estimate $h(s)$ for the optimal state value $\mathcal{V}^{*}_{\text{MP}}(s)$ or $\mathcal{V}^{*}_{\text{SSP}}(s)$ of a state s. For MaxProb, a heuristic is *admissible* if $h(s) \geq \mathcal{V}^{*}_{\text{MP}}(s)$, whereas it is admissible in the SSP setting if $h(s) \leq \mathcal{V}^{*}_{\text{SSP}}(s)$. For SSPs, a heuristic is *consistent* if the equation

$$h(s) \leq c(a) + \sum_{t \in \mathcal{S}} \mathcal{T}(s, a, t) h(t)$$

is satisfied for every $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and *goal-aware* if $h(s) = 0$ for goal states $s \in \mathcal{S}_{\mathcal{G}}$. These properties are convenient because a heuristic that is both consistent and goal-aware is admissible. Also, some SSP heuristics search algorithms like iLAO* (Hansen and Zilberstein 2001) can be optimized for consistent, goal-aware heuristics.

Lastly, a finite family of heuristics $(h_i)_{i \in I}$ (where I is some index set) is *additive* if $\sum_{i \in I} h_i(s) \leq \mathcal{V}^{*}_{\text{SSP}}(s)$ and *multiplicative* if $\prod_{i \in I} h_i(s) \geq \mathcal{V}^{*}_{\text{MP}}(s)$.

[1] More general SSP definitions exist (Kolobov et al. 2011; Guillot and Stauffer 2020), but I focus on this traditional definition for the sake of brevity.

Abstraction Heuristics

In classical planning, abstractions heuristics are a fairly versatile family of heuristics that has been studied extensively in various forms, for example through Pattern Databases (e.g. Korf (1997); Haslum et al. (2007); Pommerening, Röger, and Helmert (2013)), Cartesian Abstraction (Seipp and Helmert 2013) and Merge-and-Shrink Abstraction (e.g. Helmert, Haslum, and Hoffmann (2007); Nissim, Hoffmann, and Helmert (2011); Helmert et al. (2014)). In classical planning, an abstraction is typically specified by a surjective abstraction mapping $\alpha : \mathcal{S} \to \alpha(\mathcal{S})$, which associates each state with a corresponding abstract state $\alpha(s) \in \alpha(\mathcal{S})$. For a labelled transition system (LTS), the deterministic planning model usually assumed in classical planning, an abstraction induces an abstract LTS which overapproximates the behaviour of the original LTS. This abstract LTS can then be solved to obtain an admissible heuristic for the original problem. To do the same with respect to an MDP, a definition for the abstract MDP induced by an abstraction mapping needs to be proposed.

Projection Heuristics

In recent work (Klößner et al. 2021), we propose a definition for the specific case of *projections*. A projection is an abstraction mapping $s \mapsto s[P]$ which considers a subset of state variables (a *pattern*) $P \subseteq \mathcal{V}$ of the problem. The abstract MDP for a projection with respect to P is defined as $\langle S[P], A, \mathcal{T}_P, s_{\mathcal{I}}[P] \rangle$, where the transition probability $\mathcal{T}_P(\sigma, a, \tau)$ is defined as 0 if $\mathrm{pre}(a)[P] \not\subseteq \sigma$, otherwise

$$\mathcal{T}_P(\sigma, a, \tau) = \sum_{\substack{e \in \mathrm{Eff}(a) \text{ s.t.} \\ \mathrm{appl}(\sigma, e[P]) = \tau}} \Pr_a(e).$$

The probabilistic projection heuristic $h^P(s) := \mathcal{V}^*_{\mathrm{MP}}(s[P])$ is an admissible heuristic for MaxProb, and the analogous heuristic $h^P(s) := \mathcal{V}^*_{\mathrm{SSP}}(s[P])$ is even consistent and goal-aware for SSPs, when the abstract set of goal states for both objectives is defined as $\mathcal{S}_{\mathcal{G}}[P]$ and the cost function for SSPs is unchanged for the abstraction. Most importantly, these heuristics dominate the respective determinization-based projection heuristic on the same pattern.

Pattern Database Heuristics

In classical planning, Pattern Database (PDB) heuristics are a family of abstraction heuristics that use several projections in unison to achieve a more accurate heuristic. Given a collection of patterns $C \subseteq \mathcal{P}(\mathcal{V})$, the corresponding pattern database heuristic is constructed by precomputing a lookup table of heuristic values for each individual projection heuristic h^P for $P \in \mathcal{C}$. When an estimate for a state s is requested, these individual projection heuristics can then be combined by performing the necessary table lookups and taking the highest estimate: $h^{\max}_C(s) = \max_{P \in C}\{h^P(s)\}$. An even better approach is to employ additivity constraints to find sub-collections $D \subseteq C$ such that the heuristics $(h^P)_{P \in D}$ become additive (Haslum et al. 2007). Max'ing over these sub-collections then yields and an even stronger heuristic, called the *canonical PDB heuristic* $h^{\mathrm{can}}_C(s)$.

We published two papers (Klößner et al. 2021; Klößner and Hoffmann 2021) in which we transfer these concepts to probabilistic planning and construct pattern database heuristics which exploit a collection of MDP projections instead. In particular, we show that the well-known additivity constraints considered by Haslum et al. can be adapted and used in a straightforward manner to obtain additivity constraints for SSPs and even multiplicativity constraints for MaxProb.

In more detail, we say that an action *affects* a variable v if there is a possible effect $e \in \mathrm{Eff}(a)$ with $v \in \mathcal{V}(e)$ and $e(v) \neq \mathrm{pre}(a)(v)$. An action a affects a pattern P if any variable $v \in P$ is affected. We show that, for a collection of patterns C, if every action affects at most one pattern $P \in C$, the projection heuristics $(h^P)_{P \in C}$ are additive for the SSP objective, and multiplicative for the MaxProb objective.

This observation leads to a direct generalization of $h^{\mathrm{can}}_C(s)$ for both MaxProb and SSPs. We show that construction of this heuristic is analogous to the construction in classical planning: Finding the maximal additive sub-collections of C can still be accomplished by finding the maximal cliques in the graph where nodes are the pattern $P \in C$ and two patterns are connected if their projections are additive, which is easy to check for only two patterns. Our empirical evaluation shows a substantial improvement over the determinization-based canonical PDB heuristics.

In very recent work (Klößner et al. 2022b), we also deal with the question of how to construct reasonably construct the initial pattern collection C when the problem is no longer deterministic. We consider and extend two approaches that have been studied in classical planning: Pattern construction via Counter-example guided abstraction refinement (CEGAR, Rovner, Sievers, and Helmert (2019)) and pattern construction as a search problem solved using hill-climbing (Haslum et al. 2007). We reformulate both of these frameworks to operate on MDPs, as opposed to using the classical algorithm variants on the determinization. Compared to classical pattern construction techniques on the determinization, both algorithms have a significant advantage in particular problem domains. However, there also exist many domains in which we observe no benefit over determinization-based pattern construction, so these algorithms can by no means be seen as a universal answer to this research question. We might therefore revisit this topic in the future.

Merge-and Shrink Heuristics

The Merge-and-Shrink framework (Dräger, Finkbeiner, and Podelski 2006) is a framework that originates from model checking but has since found use in various forms in classical planning, in particular to compute abstraction heuristics. In a nutshell, the Merge-and-Shrink framework operates on a *factored transition system* which is a tuple of labelled transition systems (LTS) $F = \langle \Theta_1, \ldots, \Theta_n \rangle$ with a common set of labels. Each Θ_i is called a *factor*. These factors implicitly represent the LTS that is their synchronous product. If $\Theta_i = \langle \mathcal{S}^i, \mathcal{L}, \mathcal{T}^i, s^i_{\mathcal{I}} \rangle$, the synchronous product is the LTS $\bigotimes F = \langle \bigotimes_{i=1}^n \mathcal{S}^i, \mathcal{L}, \mathcal{T}^{\otimes}, \langle s^1_{\mathcal{I}}, \ldots s^n_{\mathcal{I}} \rangle \rangle$ where

$$\mathcal{T}^{\otimes} := \{\langle \langle s_1, \ldots, s_n \rangle, a, \langle t_1, \ldots, t_n \rangle \rangle \mid \forall i \in \{1, \ldots, n\}. \langle s_i, a, t_i \rangle \in \mathcal{T}^i\}.$$

At the start of Merge-and-Shrink, the tuple of atomic projections (to a single variable) of the LTS induced by the planning task yields the initial factored transition system and is an exact implicit representation of the state space. In each iteration, the algorithm applies one of four transformation to the factored transition system:

1. Merge: Select two factors Θ_1 and Θ_2 and replace them by their synchronous product $\Theta_1 \otimes \Theta_2$.
2. Shrink: Select a factor and apply an abstraction on top of it. Replace the old factor with the abstraction.
3. Prune: Select a factor and remove states which are not *alive*, i.e. which are unreachable or cannot reach the goal.
4. Label Reduction: Reduce the number of labels by aggregating multiple labels into a common label.

Depending on how the framework is used, the procedure either stops when there is only one factor left or when a memory or time limit is reached.

The algorithms has several important properties. If we ignore label reduction, each factor represents an abstraction of the original state space at any point in time in the algorithm (modulo non-alive states). The algorithm can therefore be used to construct abstraction heuristics. Furthermore, without label reduction and if bisimulation is used as a shrinking strategy, then each factor represents a bisimulation the factored LTS implicitly represents a bisimulation of the original LTS at any point in time (modulo non-alive states).

As of yet, it remains an open question whether the Merge-and-Shrink framework can be formulated for probabilistic planning. The first approach that comes to mind is to model each factor as an MDP and to initialize the factored representation with all atomic (MDP) projections. A reasonable definition of a product between MDPs should again have the property that the product of all atomic projections yield the original state space. Regrettably, this is impossible to accomplish. Consider a planning task with variables $v, w \in \{0, 1\}$ and a single action a with the following effects:

$$\Pr_a(\{v \mapsto 1\}) = 0.25 \quad \Pr_a(\{w \mapsto 1\}) = 0.25$$
$$\Pr_a(\{v \mapsto 1, w \mapsto 1\}) = 0.5$$

Unfortunately, this planning task has exactly the same projections onto v and w as the planning task where the effect probabilities are changed to

$$\Pr_a(\{\}) = \frac{1}{16}$$
$$\Pr_a(\{v \mapsto 1\}) = \frac{3}{16} \quad \Pr_a(\{w \mapsto 1\}) = \frac{3}{16}$$
$$\Pr_a(\{v \mapsto 1, w \mapsto 1\}) = \frac{9}{16}$$

Consequentially, we cannot define a merging operation that reconstructs the original MDP of the planning task from the atomic projections, as there are already multiple MDPs with the same atomic projections. The problem arises because we do no longer remember the individual action outcomes in the atomic projections. Whereas in classical Merge-and-Shrink we must only synchronize those transitions with the same label when building the product, we have an additional layer of synchronization in the probabilistic setting: We must now also synchronize on the outcomes of an action, as every factor must be subject to the same outcome. Therefore, the planning model used to represent a factor must remember the individual outcomes and their probabilities.

Regarding shrinking strategies, previous considerations in classical planning dealt with variants of bisimulation (Nissim, Hoffmann, and Helmert 2011; Katz, Hoffmann, and Helmert 2012). Therefore, a natural candidate to investigate for shrinking strategies in probabilistic planning is probabilistic bisimulation (Larsen and Skou 1991), as well as possibly relaxed variations of this concept. Alternatively, traditional shrinking strategies are still applicable by considering the determinization of a factor. In particular, any bisimulation on the determinization is a probabilistic bisimulation, although not necessarily the coarsest bisimulation.

Finally, label reduction is a transformation that becomes useful when using bisimulation as a shrinking strategy. Often, using bisimulation only leads to a small reduction in the size of a factor, making this shrinking strategy barely effective in isolation. However, collapsing multiple labels into a common label usually has positive effects on bisimulation, as less labels mean less restrictions for the bisimulation relation. If the label reduction is exact, i.e. it does not introduce any spurious transitions in the represented transition system, then this transformation can even be safely applied without changing the cost-to-goal estimates the factors. It is therefore important to also consider label reduction when using a variant of bisimulation in the probabilistic setting.

Other Contributions and Research Ideas

Although this thesis mainly focuses on abstraction heuristics, any topic related to domain-independent heuristic construction for MaxProb and SSPs falls into the broader scope of the thesis. In a recent publication (Klößner et al. 2022a), we propose a theory of cost partitioning (Katz and Domshlak 2010) for SSPs. We found out that Trevizan, Thiébaux, and Haslum's projection occupation measure heuristic h^{pom} essentially computes an optimal cost-partitioning over atomic projections. This has major implications, as it means that optimal cost partitioning over PDB heuristics is theoretically superior to h^{pom}. An obvious candidate for future work is an experimental evaluation of different cost-partitioning techniques and different sets of combined heuristics.

Conclusion

Abstraction heuristics for MaxProb and SSPs are promising candidates to extend the landscape of admissible heuristics for these settings and enable more effective use of MDP heuristic search algorithms. So far, we focused in particular on Pattern Database heuristics, for which we observe a clear advantage over determinization-based heuristics. When combined with cost-partitioning, these heuristics even have the potential to outperform occupation measure heuristics, which are the most powerful heuristics for SSPs at present. In future work, we aim to transfer the Merge-and-Shrink framework to probabilistic planning and take a detailed look at various cost-partitioning techniques for SSPs.

References

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 Giunchiglia, E.; Muscettola, N.; and Nau, D., eds., *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*, 12–21. Trento, Italy: AAAI Press.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2006. Directed Model Checking with Distance-Preserving Abstractions. In Valmari, A., ed., *Proceedings of the 13th International SPIN Workshop (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, 19–34. Springer-Verlag.

Guillot, M.; and Stauffer, G. 2020. The Stochastic Shortest Path Problem: A polyhedral combinatorics perspective. *European Journal of Operational Research*, 285(1): 148–158.

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 Howe, A.; and Holte, R. C., eds., *Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07)*, 1007–1012. Vancouver, BC, Canada: AAAI Press.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible Abstraction Heuristics for Optimal Sequential Planning. In Boddy, M.; Fox, M.; and Thiebaux, S., eds., *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, 176–183. Providence, Rhode Island, USA: Morgan Kaufmann.

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.

Katz, M.; and Domshlak, C. 2010. Optimal admissible composition of abstraction heuristics. *Artificial Intelligence*, 174(12–13): 767–798.

Katz, M.; Hoffmann, J.; and Helmert, M. 2012. How to Relax a Bisimulation? In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 101–109. AAAI Press.

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

Klößner, T.; Pommerening, F.; Keller, T.; and Röger, G. 2022a. Cost Partitioning Heuristics for Stochastic Shortest Path Problems. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22)*, 193–202. AAAI Press.

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

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

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic Search for Generalized Stochastic Shortest Path MDPs. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS'11)*. AAAI Press.

Korf, R. E. 1997. Finding Optimal Solutions to Rubik's Cube Using Pattern Databases. In Kuipers, B. J.; and Webber, B., eds., *Proceedings of the 14th National Conference of the American Association for Artificial Intelligence (AAAI'97)*, 700–705. Portland, OR: MIT Press.

Larsen, K. G.; and Skou, A. 1991. Bisimulation through probabilistic testing. *Information and Computation*, 94(1): 1–28.

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., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, 1983–1990. AAAI Press/IJCAI.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the Most Out of Pattern Databases for Classical Planning. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. AAAI Press/IJCAI.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-Based Heuristics for Cost-Optimal Planning. In Chien, S.; Do, M.; Fern, A.; and Ruml, W., eds., *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, 226–234. AAAI Press.

Rovner, A.; Sievers, S.; and Helmert, M. 2019. Counterexample-Guided Abstraction Refinement for Pattern Selection in Optimal Classical Planning. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS'19)*, 362–367. AAAI Press.

Seipp, J.; and Helmert, M. 2013. Counterexample-guided Cartesian Abstraction Refinement. In Borrajo, D.; Fratini, S.; Kambhampati, S.; and Oddi, A., eds., *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*, 347–351. Rome, Italy: AAAI Press.

Trevizan, F. W.; Thiébaux, S.; and Haslum, P. 2017. Occupation Measure Heuristics for Probabilistic Planning. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS'17)*, 306–315. AAAI Press.

Trevizan, F. W.; Thiébaux, S.; Santana, P. H.; and Williams, B. 2017. I-dual: Solving Constrained SSPs via Heuristic Search in the Dual Space. In Sierra, C., ed., *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, 4954–4958. AAAI Press/IJCAI.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A Baseline for Probabilistic Planning. In Boddy, M.; Fox, M.; and Thiebaux, S., eds., *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, 352–359. Providence, Rhode Island, USA: Morgan Kaufmann.