Pattern Databases for Stochastic Shortest Path Problems

Thorsten Klößner, Jörg Hoffmann
Saarland University, Saarland Informatics Campus, Germany
{kloessner, hoffmann}@cs.uni-saarland.de

Abstract

Stochastic shortest-path problems (SSP) are an important subclass of MDPs for which heuristic search algorithms exist since over a decade. Yet most known heuristic functions rely on determinization so do not actually take the transition probabilities into account. The only exceptions are Trevizan et al.'s heuristics h^{pom} and h^{roc}, which are geared at solving more complex (constrained) MDPs. Here we contribute pattern database (PDB) heuristics for SSPs, including an additivity criterion. These new heuristics turn out to be very competitive, even when using a simple systematic generation of pattern collections up to a fixed size. In our experiments, they beat determinization-based heuristics, and tend to yield better runtimes than h^{pom} and h^{roc}.

Introduction

Stochastic shortest-path problems (SSP) are an important subclass of MDPs, requiring to minimize the expected cost of reaching a goal. Costs are not discounted, and it is assumed that dead-ends are avoidable, i. e., there exists a policy reaching the goal with probability 1. Heuristic search algorithms for SSP are well known (Hansen and Zilberstein 2001; Bonet and Geffner 2003, 2006; Trevizan et al. 2017). These methods require an expected-cost heuristic function h, admissible in terms of lower-bounding the actual expected cost. Yet research on such heuristic functions h for SSP has been limited so far.

Most work on SSP heuristic search was concerned with the search algorithms, not the heuristic functions. In the context of probabilistic planning which we address here, authors relied on classical planning heuristics, applied to probabilistic planning via the all-outcomes determinization which assumes that one can choose the action outcome (e. g. (Bonet and Geffner 2005; Teichteil-Königsbuch, Vidal, and Infantes 2011; Kolobov, Mausam, and Weld 2010, 2012)). While this construction is elegant, the resulting heuristic functions ignore the transition probabilies characterizing the probabilistic setting. The only other known SSP heuristic functions are Trevizan et al.'s (2017) LP-based occupancy measure heuristics h^{pom} and h^{roc}. These are powerful in their native search algorithm i-dual (Trevizan et al. 2017). Yet i-dual is designed for the more complex setting of constrained MDPs (where several objectives must be reasoned about), and may not be the most effective way to tackle the simpler SSP case.

Here we devise a pattern database (PDB) heuristic for SSP, taking the transition probabilities into account. PDBs are a well-known class of heuristic functions based on problem projections (e. g. (Korf 1997; Culberson and Schaeffer 1998; Edelkamp 2001; Holte et al. 2006; Haslum et al. 2007; Pommerening, Röger, and Helmert 2013)). PDBs for probabilistic planning were previously considered (Keyder and Geffner 2008; Klößner et al. 2021), but only for a different class of MDPs, namely goal probability maximization in the presence of unavoidable dead-ends.

Defining PDBs as in prior work on planning, we obtain from Klößner et al.'s (2021) results that the resulting abstract state spaces are still MDPs (which is not the case for state abstractions in general as the transition probabilities may not be well-defined (Givan, Leach, and Dean 2000; Tagorti et al. 2013)). We show that, while these MDPs are not SSPs anymore, they can be solved with existing methods and the resulting heuristic functions are consistent. We also devise an additivity criterion, which transfers directly from corresponding criteria in deterministic search and classical planning (Edelkamp 2001; Korf and Felner 2002; Felner, Korf, and Hanan 2004; Haslum et al. 2007).

These new heuristics turn out to be very competitive, even when using a simple systematic generation of pattern collections, including all patterns up to a fixed size K. We run experiments on SSP probabilistic PDDL benchmark domains from the International Probabilistic Planning Competitions. The new PDB heuristics beat the corresponding determinized PDB heuristics, showing that taking probabilities into account can pay off. With small patterns of size 2, they are similarly informative as h^{pom} and h^{roc}, but are faster to compute and hence tend to yield better runtimes.

Preliminaries

At the surface level, we address probabilistic planning tasks formulated in PPDDL (Younes et al. 2005). In the planner's internal representation, these tasks come in a probabilistic finite-domain (FDR) notation (Bäckström and Nebel 1995; Trevizan, Thiébaux, and Haslum 2017), as follows. A probabilistic FDR task is a tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, s_{\mathcal{I}}, \mathcal{G}, c \rangle$. $\mathcal{V}$ denotes the *variables*, each v has a finite domain $\mathcal{D}_v$. A *variable assignment* is a partial function $\sigma : \mathcal{V} \mapsto \bigcup_{v \in \mathcal{V}} \mathcal{D}_v$ with

$\sigma(v) \in \mathcal{D}_v$, if defined. We write $\sigma(v) = \bot$ in case that $\sigma(v)$ remains undefined. We denote by $\mathcal{V}(\sigma)$ the set of all variables v with $\sigma(v) \neq \bot$. σ is *complete* if $\mathcal{V}(\sigma) = \mathcal{V}$. For a set of variables $U \subseteq \mathcal{V}(\sigma)$, we denote by $\sigma|_U$ the *projection of σ onto U*. Slightly abusing notation, we denote the *composition* of two variable assignments as $\sigma_2 \circ \sigma_1$ where $(\sigma_2 \circ \sigma_1)(v) = \sigma_2(v)$ for all $v \in \mathcal{V}(\sigma_1) \cap \mathcal{V}(\sigma_2)$, and $(\sigma_2 \circ \sigma_1)(v) = \sigma_1(v)$ for all $v \in \mathcal{V}(\sigma_1) \setminus \mathcal{V}(\sigma_2)$. The *states* $\mathcal{S}$ of Π are the complete variable assignments. The *initial state* $s_\mathcal{I}$ is a state. The *goal* $\mathcal{G}$ is a variable assignment. $\mathcal{A}$ is the set of *actions*. An action a specifies its *precondition* pre_a, and a set of *effects* eff_a, all variable assignments; as well as a probability distribution P_a over the effects. $c : \mathcal{A} \to \mathbb{R}_0^+$ is a non-negative cost function independent of the source state.

Π induces the MDP $\mathcal{M} = \langle \mathcal{S}, s_\mathcal{I}, \mathcal{T}, \mathcal{S}_\mathcal{G}, c \rangle$ whose states and initial state are those of Π. The *goal states* $\mathcal{S}_\mathcal{G}$ are all states $s_\mathcal{G}$ where $s_\mathcal{G}(v) = \mathcal{G}(v)$ for all $v \in \mathcal{V}(\mathcal{G})$. An action a is *applicable* in state s if $pre_a(v) = s(v)$ for all $v \in \mathcal{V}(pre_a)$. $\mathcal{A}(s)$ denotes the set of all actions applicable in s. We require $\mathcal{A}(s) \neq \emptyset$ for all $s \in \mathcal{S}$. The *transition function* $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is defined for $a \in \mathcal{A}(s)$ by $\mathcal{T}(s, a, t) = \sum_{e \in eff_a, t=(e \circ s)} P_a(e)$. To simplify equations, we require that every goal state has a zero-cost self-loop.

We require that $\mathcal{M}$ is a stochastic shortest path problem (SSP), precisely: (i) there is at least one policy for which the goal probability of every state is 1 (such a policy is called *proper*) and (ii) for improper policies, the total expected cost incurred until a goal is reached is infinite for at least one initial state. Since we consider non-negative action costs, (ii) reduces to the absence of zero-cost cycles.

The optimal value function $J^* : \mathcal{S} \to \mathbb{R}$ maps each state to its minimal (non-discounted) expected cost-to-goal among all proper policies. Formally, with S_i denoting a random variable representing the state in time step i:

$$J^*(s) := \min_{\pi \text{ proper}} \mathbb{E}\left[\sum_{i=0}^{\infty} c(\pi(S_i))\right]$$

J^* can be computed by *value iteration (VI)*, which starts with an initial value function J_0 and iteratively applies the Bellman update operator $\mathcal{B}$ defined by

$$(\mathcal{B}J)(s) = \min_{a \in \mathcal{A}(s)} c(a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s')J(s)$$

until convergence. Under SSP assumptions (i) and (ii), J^* is the unique fixed point of $\mathcal{B}$ within $\{J \mid \forall s \in \mathcal{S}_\mathcal{G}.\, J(s) = 0\}$. Moreover, $\mathcal{B}$ is monotonic, and $\lim_{k \to \infty} \mathcal{B}^k(J_0) = J^*$ for any initial guess J_0 where $J_0(s) = 0$ for all $s \in \mathcal{S}_\mathcal{G}$.

A *heuristic* is a function $h : \mathcal{S} \to \mathbb{R}$. We say that h is *admissible* if $h(s) \leq J^*(s)$ for all $s \in \mathcal{S}$. We say that h is *goal-aware* if $h(s) = 0$ for all $s \in \mathcal{S}_\mathcal{G}$. We say that h is *consistent* if $h \leq \mathcal{B}h$. Consistency and goal-awareness imply admissibility because $\mathcal{B}$ is monotonic: $h \leq \mathcal{B}h \leq \mathcal{B}^2h \leq \cdots \leq \lim_{k \to \infty} \mathcal{B}^k h = J^*$.

Since VI has to construct the entire state space, heuristic search algorithms (Hansen and Zilberstein 2001; Bonet and Geffner 2003, 2006; Trevizan et al. 2017) can be far more memory-efficient. These algorithms search forward from the initial state $s_\mathcal{I}$, and perform Bellman updates starting from an admissible heuristic h, thus maintaining an approximation J of J^*. In each search step, they focus on the so-called *greedy graph*, which contains only those actions that minimize $(\mathcal{B}J)(s)$. If h is accurate, many states are never reached in a greedy graph and thus will not be generated during search.

Expected-Cost PDBs

We follow Klößner et al. (2021) in the definition of *projections* of the induced MDP $\mathcal{M}$ onto a subset $V \subseteq \mathcal{V}$ of variables also denoted a *pattern*. The state set $\mathcal{S}_V$ is the set of complete variable assignments to V. The goal states are $\mathcal{S}_{\mathcal{G}V} := \{s_\mathcal{G}|_V \mid s_\mathcal{G} \in \mathcal{S}_\mathcal{G}\}$. For the transition probabilities between projected states σ, τ, we set $\mathcal{T}_V(\sigma, a, \tau) = 0$ for all τ if $pre_a|_V \not\subseteq \sigma$. Otherwise, we select any representative $s \in \mathcal{S}$ of σ, where $\sigma = s|_V$; and for each $a \in \mathcal{A}(s)$ and τ, we set $\mathcal{T}_V(\sigma, a, \tau) := \sum_{t \in \mathcal{S}.\ \tau = t|_V} \mathcal{T}(s, a, t)$. As Klößner et al. (2021) show, this is well-defined, i. e., the transition probability is the same regardless of which representative s of σ is selected. We denote the projected MDP by $\mathcal{M}_V = \langle \mathcal{S}_V, \mathcal{A}, \mathcal{T}_V, \mathcal{S}_{\mathcal{G}V} \rangle$.

Obviously, $\mathcal{M}_V$ can be constructed directly from Π in time polynomial in $|\Pi|$ and exponential only in $|V|$. Observe though that $\mathcal{M}_V$ is not necessarily an SSP. Condition (i) is satisfied since the projection can only increase goal reachability. But the projection may introduce zero-cost cycles, so condition (ii) does not necessarily hold anymore. However, $\mathcal{M}_V$ falls into the class of so-called generalized stochastic shortest path problems (GSSPs), and can be solved optimally using VI starting from an upper-bounding initial value function, or by eliminating these cycles in a preprocessing step or during search (Kolobov et al. 2011). In our experiments, we restrict ourselves to strictly positive actions costs.

We henceforth denote with J_V^* the expected cost value function J^* of $\mathcal{M}_V$, and with $\mathcal{B}_V$ the bellman update operator w.r.t. $\mathcal{M}_V$. Note that J_V^* is a fixed point of $\mathcal{B}_V$ (though not necessarily the only one). We define the expected cost PDB heuristic for pattern V as $h^V(s) := J_V^*(s|_V)$. By definition of $\mathcal{S}_{\mathcal{G}V}$, h^V is goal-aware. A fundamental observation is that h^V is also consistent for any pattern $V \subseteq \mathbb{V}$.

Theorem 1 *h^V is consistent.*

Proof. Let $s \in S$. We need to prove $h^V(s) \leq (\mathcal{B}h^V)(s)$.

We start by exploiting that J_V^* is a fixed point of $\mathcal{B}_V$.

$$\begin{aligned} h^V(s) &= J_V^*(s|_V) = (\mathcal{B}_V J_V^*)(s|_V) \\ &= \min_{a \in \mathcal{A}(s|_V)} c(a) + \sum_{\tau \in \mathcal{S}^V} \mathcal{T}_V(s|_V, a, \tau) J_V^*(\tau) \end{aligned}$$

Next, we apply the definition of $\mathcal{T}_V$ and only consider actions in $\mathcal{A}(s) \subseteq \mathcal{A}(s|_V)$.

$$\begin{aligned} &\leq \min_{a \in \mathcal{A}(s)} c(a) + \sum_{\tau \in \mathcal{S}^V} \sum_{\substack{t \in \mathcal{S} \\ t|_V = \tau}} \mathcal{T}(s, a, t) J_V^*(\tau) \\ &= \min_{a \in \mathcal{A}(s)} c(a) + \sum_{t \in \mathcal{S}} \mathcal{T}(s, a, t) J_V^*(t|_V) \end{aligned}$$

$$= \min_{a \in \mathcal{A}(s)} c(a) + \sum_{t \in \mathcal{S}} \mathcal{T}(s,a,t) h^V(t) = (\mathcal{B}h^V)(s)$$

This concludes the proof. □

Additivity

Given a collection (a set) $\mathcal{C}$ of patterns, the *sum heuristic* $h^{\text{add}}_{\mathcal{C}}(s) := \sum_{V \in \mathcal{C}} h^V(s)$ sums over the individual pattern heuristics. Sufficient conditions for admissibility can be identified based on *orthogonality*, aka additivity (Korf and Felner 2002; Felner, Korf, and Hanan 2004). In particular, such a condition has been devised in classical planning (Edelkamp 2001; Haslum et al. 2007). We next show that this orthogonality criterion can be used analogously in the expected-cost setting.

We say that an action $a \in \mathcal{A}$ *affects* a pattern $V \subseteq \mathbb{V}$ if $\mathcal{M}_V$ contains non-loop a-transitions, i.e. if $\mathcal{T}_V(\sigma, a, \tau) > 0$ for some $\sigma \neq \tau$. Similarly to the classical case, this property can be checked syntactically on the planning task: a affects V if and only if there is an effect e with $P_a(e) > 0$ and a variable $v \in V$ for which $e(v) \neq \bot$ and $e(v) \neq pre_a(v)$.

Definition 1 (Orthogonality) *A pattern collection $\mathcal{C}$ is* orthogonal *if every action in $\mathcal{A}$ affects at most one $V \in \mathcal{C}$.*

We next show that, just like in classical planning, $h^{\text{add}}_{\mathcal{C}}$ is admissible if $\mathcal{C}$ is orthogonal. Observe first that $h^{\text{add}}_{\mathcal{C}}$ is trivially goal-aware, as every individual heuristic is. Furthermore, $h^{\text{add}}_{\mathcal{C}}$ is consistent:

Theorem 2 *If $\mathcal{C}$ is orthogonal, then $h^{\text{add}}_{\mathcal{C}}$ is consistent.*

Proof. Let $s \in \mathcal{S}$. We need to show $h^{\text{add}}_{\mathcal{C}}(s) \leq (\mathcal{B}h^{\text{add}}_{\mathcal{C}})(s)$.

Let $a \in \mathcal{A}(s)$ be an arbitary action. Since $\mathcal{C}$ is orthogonal, there is a pattern $W_a \in \mathcal{C}$ dependent on a such that all other patterns $V \in \mathcal{C} \setminus \{W_a\}$ are unaffected by a. We know that $h^{W_a}(s) \leq c(a) + \sum_{t \in \mathcal{S}} \mathcal{T}(s,a,t) h^{W_a}(t)$ because h^{W_a} is consistent as previously shown. Therefore

$$h^{\text{add}}_{\mathcal{C}}(s) \leq \left(c(a) + \sum_{t \in \mathcal{S}} \mathcal{T}(s,a,t) h^{W_a}(t) \right) + \sum_{V \in \mathcal{C} \setminus \{W_a\}} h^V(s)$$

Now we rewrite the right summand by multiplying with $\sum_{t \in \mathcal{S}} \mathcal{T}(s,a,t) = 1$. Acknowledge that $\mathcal{T}(s,a,t) > 0$ implies $s|_V = t|_V$ and in particular $h^V(s) = h^V(t)$ for any $V \in \mathcal{C} \setminus \{W_a\}$ since a does not affect V.

$$= c(a) + \sum_{t \in \mathcal{S}} \mathcal{T}(s,a,t) h^{W_a}(t) + \sum_{t \in \mathcal{S}} \mathcal{T}(s,a,t) \sum_{V \in \mathcal{C} \setminus \{W_a\}} h^V(t)$$

$$= c(a) + \sum_{t \in \mathcal{S}} \mathcal{T}(s,a,t) h^{\text{add}}_{\mathcal{C}}(t)$$

This inequality holds for all actions, hence

$$h^{\text{add}}_{\mathcal{C}}(s) \leq \min_{a \in \mathcal{A}(s)} c(a) + \sum_{t \in \mathcal{S}} \mathcal{T}(s,a,t) h^{\text{add}}_{\mathcal{C}}(t) = (\mathcal{B}h^{\text{add}}_{\mathcal{C}})(s)$$

This concludes the proof. □

Given a pattern collection $\mathcal{C}$, we exploit orthogonality analogously to Haslum et al. (2007) in the classical setting, maximizing over all maximal orthogonal subcollections of $\mathcal{C}$. The resulting heuristic is called the *canonical heuristic*, defined as $h^{\text{can}}_{\mathcal{C}}(s) := \max_{D \in Q} \sum_{V \in D} h^D(s)$ where Q is the set of maximal orthogonal subcollection of $\mathcal{C}$.

Experiments

Our implementation is based on the probabilistic extension of Fast Downward (Helmert 2006; Steinmetz, Hoffmann, and Buffet 2016), with PPDDL as the input language. The experiments were run on a cluster of Intel Xeon E5-2650 v3 processors @2.30 GHz, using a memory limit of 4 GB and a time limit of 30 minutes. The LP solver we use is Soplex 3.1.1. All experiments are run using the Downward Lab toolkit (Seipp et al. 2017). In the following, we shortly decribe the algorithms we tested as well as the benchmarks used, before summarizing our empirical results.

Heuristics & Search Algorithms Compared

We experimented with LAO*and LRTDP as the underlying heuristic search algorithms. As LRTDP conducts its trials probabilistically, we run each LRTDP search configuration with 10 different random seeds and average the results. The results for both search algorithms are qualitatively very similar, so we report the data only for LRTDP below.

We experiment with both the additive and non-additive variant of expected-cost PDBs (ecpdbs). To compare against the analogous determinization-based approach, we also experiment with canonical PDBs (cpdbs) on the determinization. To solve projections (i. e. to construct expected-cost PDBs), we use focused topological value iteration (Dai et al. 2011). All algorithms are run until ϵ-convergence, i. e. the difference between successive Bellman backups becomes less than ϵ. We fix $\epsilon = 10^{-5}$ throughout our experiments.

Pattern generation for the expected cost setting is in itself a non-trivial research question, which remains open for future work. For our experiments, we adopt a simple systematic pattern generation procedure previously suggested in classical planning (Pommerening, Röger, and Helmert 2013). We enumerate pattern collections up to a fixed pattern size K (pruning ones that can easily be recognized to

						$K=2$			$K=3$		
Domain	N	-	h^{lmc}	h^{pom}	h^{roc}	cpdb	ecpdb	add. ecpdb	cpdb	ecpdb	add. ecpdb
blocksw	30	9	9	**21**	**21**	12	9	18	12	9.7	17
elevators	15	13	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	11
random	15	3	4	7	**8.8**	4	7	7	4	7	7
schedule	30	7	**9**	7	7	**9**	**9**	**9**	**9**	**9**	**9**
tri-tirew	10	5	5	5	5	5	**6**	**6**	5	**6**	**6**
zenotrvl	30	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**
Σ	130	45	49	62	**63.8**	52	53	62	52	53.7	58

Table 1: Coverage for LRTDP. Highest coverage highlighted in boldface. K denotes the size bound of the systematic pattern generator.

					$K=2$			$K=3$		
Domain	N	h^{lmc}	h^{pom}	h^{roc}	cpdb	ecpdb	add. ecpdb	cpdb	ecpdb	add. ecpdb
blocksworld	9	1,067.9	200.7	**196.8**	956.2	940.7	270.7	950.6	462.5	258.3
elevators	11	**9,473.3**	26,755.4	26,755.4	21,174.0	32,447.2	21,508.0	21,053.0	29,923.7	20,164.1
random	4	42,731.7	**778.7**	**778.7**	60,772.3	59,931.8	59,931.8	961.7	**778.7**	**778.7**
schedule	7	9,904.7	233,382.1	231,562.4	9,932.3	10,092.4	9,323.4	8,109.7	**7,804.3**	**7,804.3**
tri-tireworld	5	461,472.6	212,439.0	212,439.0	486,091.5	303,349.7	303,349.7	461,472.6	**201,582.5**	**201,582.5**
zenotravel	8	139,947.2	119,216.7	119,216.7	157,196.0	105,067.2	71,946.7	147,117.6	27,485.1	**18,554.1**
Mean	44	110,766.2	98,795.4	98,491.5	122,687.0	85,304.9	77,721.7	106,610.9	44,672.8	**41,523.7**

Table 2: Evaluated States. Mean taken over instances covered by all competing configurations (and over ten random seeds for LRTDP). Lowest values highlighted in boldface. N is the number of common instances, K is the size bound of the systematic pattern generator.

not be useful, e. g. not containing a goal variable). We experiment with $K \in \{2, 3\}$. For $K > 3$, PDB construction is infeasible due to the number of patterns. The pattern collection is identical across all PDB configurations.

We compare against h^{roc} and h^{pom}, the only prior SSP heuristic that take transition probabilities into account. As a canonical representative of the determinization-based approach, we compare against LM-cut (h^{lmc}) (Helmert and Domshlak 2009).

Benchmarks

We run experiments on benchmarks from the International Probabilistic Planning Competitions (IPPC). Specifically, we use the subset of benchmarks that are stochastic shortest path problems, and that are formulated in our input language PPDDL (Younes et al. 2005). These benchmarks are from the IPPC editions 2006 and 2008 as later IPPCs switched to RDDL (Sanner 2010).[1] The benchmark domains are blocksworld, elevators, schedule, random, triangle-tireworld, and zenotravel. In all these domains, the SSP conditions are met because dead-ends are avoidable and action costs are unit 1.

Results

Tables 1 and 2 show coverage (number of problem instances solved) and search space size data. Consider first the comparison of our new heuristics to determinization-based approaches. LM-cut h^{lmc} lags behind almost consistently (the exception being elevators), and the same is true for determinized PDBs (cpdb) vs. their expected-cost counterparts (add. ecpdb). The data hence provides strong evidence that taking probabilities into account is important. The extent of the advantage varies per domain as one would expect, with dramatic improvements in blocksworld and random, significant improvements in triangle-tireworld and zenotravel, and more moderate improvements elsewhere.

The comparison to h^{pom} and h^{roc} is more subtle. Our heuristics have a substantial advantage in schedule, h^{pom} and h^{roc} have a substantial advantage in blocksworld. Elsewhere, both coverage and number of evaluated states are roughly on par (except for zenotravel with $K = 3$ which we discuss below). Our heuristics are faster to compute however, as evaluating them does not involve solving an LP. Figure 1 shows that they pay off in runtime almost consistently.

Considering again search space size as a measure of informedness, it appears that both types of heuristics often capture similar structure. This indeed makes sense for very small patterns of size $K = 2$, as h^{pom} and h^{roc} are based on individual-variable ($K = 1$) projections augmented with constraints over variable pairs. It is interesting to observe in this context the substantial reduction in evaluated states on zenotravel for $K = 3$; this gives evidence that, with larger patterns, expected-cost PDBs can reach beyond h^{pom} and h^{roc} as one would expect. More targeted pattern-selection strategies are required for this as the overhead of our simple strategy for $K = 3$ already outweighs this advantage.

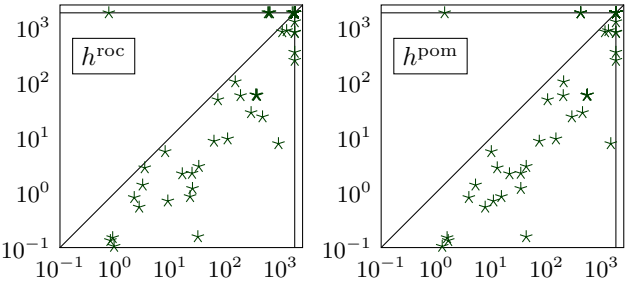

Figure 1: Total Time (in seconds) for LRTDP. Additive expected-cost PDBs with $K = 2$ (y-axis in both plots) vs. h^{roc} and h^{pom} (x-axes).

Conclusion

Considering the amount of work that has gone into pattern databases, and the attention that has been devoted to heuristic search on stochastic shortest path problems, it is surprising that the two have not been put together before. We do so here and the results clearly point to the potential benefits. Major questions for future work are more clever strategies for pattern collection as already discussed, as well as the use of cost partitioning (Katz and Domshlak 2010; Seipp, Keller, and Helmert 2020) for more powerful additivity.

[1] The first IPPC edition in 2004 also used PPDDL, but had very limited benchmarks (20 instances in total), of which most were reused/extended in 2006 anyway.

Acknowledgments

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

References

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS^+ Planning. *Computational Intelligence* 11(4): 625–655.

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

Bonet, B.; and Geffner, H. 2005. mGPT: A Probabilistic Planner Based on Heuristic Search. *JAIR* 24: 933–944.

Bonet, B.; and Geffner, H. 2006. Learning Depth-First Search: A Unified Approach to Heuristic Search in Deterministic and Non-Deterministic Settings, and Its Application to MDPs. In *Proc. ICAPS'06*, 142–151.

Culberson, J. C.; and Schaeffer, J. 1998. Pattern Databases. *Computational Intelligence* 14(3): 318–334.

Dai, P.; Mausam; Weld, D. S.; and Goldsmith, J. 2011. Topological Value Iteration Algorithms. *JAIR* 42: 181–209.

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

Felner, A.; Korf, R.; and Hanan, S. 2004. Additive Pattern Database Heuristics. *JAIR* 22: 279–318.

Givan, R.; Leach, S. M.; and Dean, T. 2000. Bounded-parameter Markov decision processes. *AI* 122(1–2): 71–109.

Hansen, E. A.; and Zilberstein, S. 2001. LAO^*: A heuristic search algorithm that finds solutions with loops. *AI* 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. *JAIR* 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proc. ICAPS'09*, 162–169.

Holte, R. C.; Felner, A.; Newton, J.; Meshulam, R.; and Furcy, D. 2006. Maximizing over multiple pattern databases speeds up heuristic search. *AI* 170(16-17): 1123–1136. doi: 10.1016/j.artint.2006.09.002.

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

Keyder, E.; and Geffner, H. 2008. The HMDP Planner for Planning with Probabilities. In *IPC 2008 planner abstracts*.

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.

Kolobov, A.; Mausam; and Weld, D. S. 2010. Classical Planning in MDP Heuristics: with a Little Help from Generalization. In *Proc. ICAPS'10*, 97–104.

Kolobov, A.; Mausam; and Weld, D. S. 2012. Discovering Hidden Structure in Factored MDPs. *Artificial Intelligence* 189: 19–47.

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

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

Korf, R. E.; and Felner, A. 2002. Disjoint pattern database heuristics. *AI* 134(1–2): 9–22.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the Most Out of Pattern Databases for Classical Planning. In *Proc. IJCAI'13*.

Sanner, S. 2010. Relational Dynamic Influence Diagram Language (RDDL): Language Description. Available at http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *JAIR* 67: 129–167.

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

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

Tagorti, M.; Scherrer, B.; Buffet, O.; and Hoffmann, J. 2013. Abstraction Pathologies in Markov Decision Processes. In *Proceedings of the 8th Journées Francophones Planification, Décision, et Apprentissage (JFPDA-13)*.

Teichteil-Königsbuch, F.; Vidal, V.; and Infantes, G. 2011. Extending Classical Planning Heuristics to Probabilistic Planning with Dead-Ends. In Burgard, W.; and Roth, D., eds., *Proc. the 25th National Conference of the American Association for Artificial Intelligence (AAAI'11)*. San Francisco, CA, USA: AAAI Press.

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

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 *Proc. IJCAI'17*, 4954–4958.

Younes, H. L. S.; Littman, M. L.; Weissman, D.; and Asmuth, J. 2005. The First Probabilistic Track of the International Planning Competition. *JAIR* 24: 851–887.