Heuristic Search for Multi-Objective Probabilistic Planning

Dillon Z. Chen[1], **Felipe Trevizan**[1], **Sylvie Thiébaux**[1,2]

[1]School of Computing, The Australian National University
[2]LAAS-CNRS, ANITI, Université de Toulouse
{Dillon.Chen, Felipe.Trevizan, Sylvie.Thiebaux}@anu.edu.au

Abstract

Heuristic search is a powerful approach that has successfully been applied to a broad class of planning problems, including classical planning, multi-objective planning, and probabilistic planning modelled as a stochastic shortest path (SSP) problem. Here, we extend the reach of heuristic search to a more expressive class of problems, namely multi-objective stochastic shortest paths (MOSSPs), which require computing a coverage set of non-dominated policies. We design new heuristic search algorithms MOLAO* and MOLRTDP, which extend well-known SSP algorithms to the multi-objective case. We further construct a spectrum of domain-independent heuristic functions differing in their ability to take into account the stochastic and multi-objective features of the problem to guide the search. Our experiments demonstrate the benefits of these algorithms and the relative merits of the heuristics.

1 Introduction

Stochastic shortest path problems (SSPs) are the de facto model for planning under uncertainty. Solving an SSP involves computing a policy which maps states to actions so as to minimise the expected (scalar) cost to reach the goal from a given initial state. Multi-objective stochastic shortest path problems (MOSSPs) are a useful generalisation of SSPs where multiple objectives (e.g. time, fuel, risk) need to be optimised without the user being able to *a priori* weigh these objectives against each other (Roijers and Whiteson 2017). In this more complex case, we now aim to compute a *set* of non-dominated policies whose vector-valued costs represent all the possible trade-offs between the objectives.

There exist two approaches to solving MOMDPs (Roijers and Whiteson 2017), a special case of MOSSPs: *inner loop planning* which consists in extending SSP solvers by generalising single objective operators to the multi-objective (MO) case, and *outer loop planning* which consists in solving a scalarised MOSSP multiple times with an SSP solver. We focus on the former. The canonical inner loop method, MO Value Iteration (MOVI) (White 1982) and its variants (Wiering and de Jong 2007; Barrett and Narayanan 2008; Roijers and Whiteson 2017), require enumerating the entire state space of the problem and are unable to scale to the huge state spaces typically found in automated planning.

Therefore, this paper focuses on heuristic search methods which explore only a small fraction of the state space when guided with informative heuristics. Heuristic search has been successfully applied to a range of optimal planning settings, including single objective (SO) or constrained SSPs (Hansen and Zilberstein 2001; Bonet and Geffner 2003; Trevizan et al. 2017), and MO deterministic planning (Mandow and Pérez-de-la-Cruz 2010; Khouadjia et al. 2013; Ulloa et al. 2020). Moreover, a technical report by Bryce *et al.* (2007) advocated the need for heuristic search and outlined an extension of LAO* (Hansen and Zilberstein 2001) for finite horizon problems involving multiple objectives and partial observability. Convex Hull Monte-Carlo Tree-Search (Painter, Lacerda, and Hawes 2020) extends the Trial-Based Heuristic Tree Search framework (Keller and Helmert 2013) to the MO setting but applies only to finite-horizon MOMDPs. Yet to the best of our knowledge there is no investigation of heuristic search for MOSSPs.

This paper fills this gap in heuristic search for MOSSPs. First, we characterise the necessary conditions for MOVI to converge in the general case of MOSSPs. We then extend the well-known SSP heuristic search algorithms LAO* (Hansen and Zilberstein 2001) and LRTDP (Bonet and Geffner 2003) to the multi-objective case, leading to two new MOSSP algorithms, MOLAO* and MOLRTDP, which we describe along with sufficient conditions for their convergence. We also consider the problem of guiding the search of these algorithms with domain-independent heuristics. A plethora of domain-independent heuristics exist for classical planning, but works on constructing heuristics for (single objective) probabilistic or multi-objective (deterministic) planning are much more recent (Trevizan, Thiébaux, and Haslum 2017; Klößner and Hoffmann 2021; Geißer et al. 2022). Building on these recent works, we investigate a spectrum of heuristics for MOSSPs differing in their ability to account for the probabilistic and multi-objective features of the problem.

Finally, we conduct an experimental comparison of these algorithms and of the guidance obtained via these heuristics. We observe the superiority of heuristic search over value iteration methods for MOSSPs, and of heuristics that are able to account for the tradeoffs between competing objectives.

2 Background

A *multi-objective stochastic shortest path problem (MOSSP)* is a tuple $(S, s_0, G, A, P, \vec{C})$ where: S is a finite set of states, one of which is the initial state s_0, $G \subseteq S$ is a set of goal states, A is a finite set of actions, $P(s' \mid s, a)$ is the probability of reaching s' after applying action a in s, and $\vec{C}(a) \in \mathbb{R}^n_{\geq 0}$ is the n-dimensional vector representing the cost of action a. Two special cases of MOSSPs are stochastic shortest path problems (SSPs) and bi-objective SSPs which are obtained when n equals 1 and 2, respectively.

A solution for an SSP is a *deterministic policy* π, i.e., a mapping from states to actions. A policy π is *proper* or *closed w.r.t.* s_0 if the probability of reaching G when following π from s_0 is 1; if this probability of reaching G is less than 1, then π is an improper policy. We denote by $S^\pi \subseteq S$ the set of states visited when following a policy π from s_0. The expected cost of reaching G when using a proper policy π from a state $s \in S^\pi$ is given by the *policy value function* defined as

$$V^\pi(s) = C(\pi(s)) + \sum_{s' \in S} P(s'|s, \pi(s))V^\pi(s') \qquad (1)$$

for $s \in S^\pi \setminus G$ and $V^\pi(g) = 0$ for $g \in G$. An optimal policy for an SSP is any proper policy π^* such that $V^{\pi^*}(s_0) \leq V^{\pi'}(s_0)$ for all proper policies π'. Although π^* might not be unique, the *optimal value function* V^* is unique and equals V^{π^*} for any π^*.

Stochastic policies are a generalisation of deterministic policies which map states to probability distributions of actions. The definitions of proper, improper and policy value function are trivially generalised to stochastic policies. A key result for SSPs is that at least one of its optimal policies is deterministic (Bertsekas and Tsitsiklis 1991); thus it suffices to search for deterministic policies when solving SSPs.

Coverage Sets and Solutions for MOSSPs

In the context of MOSSPs, given a policy π, we denote by $\vec{V}^\pi : S \to \mathbb{R}^n_{\geq 0}$ the vector value function for π. The function $\vec{V}^\pi$ is computed by replacing V and C by $\vec{V}$ and $\vec{C}$ in (1), respectively. In order to define the solution of an MOSSP, we need to first define how to compare two different vectors: a cost vector $\vec{v}$ *dominates* $\vec{u}$, denoted as $\vec{v} \preceq \vec{u}$, if $\vec{v}_i \leq \vec{u}_i$ for $i = 1, \dots, n$. A *coverage set* for a set of vectors $\mathbf{V}$, denoted as $\mathrm{CS}(\mathbf{V})$, is any set satisfying $\forall \vec{v} \in \mathrm{CS}(\mathbf{V}), \nexists \vec{u} \in \mathrm{CS}(\mathbf{V})$ s.t. $\vec{u} \preceq \vec{v}$ and $\vec{u} \neq \vec{v}$.[1] An example of a coverage set is the *Pareto coverage set* (PCS) which is the largest possible coverage set. For the remainder of the paper we focus on the *convex coverage set* (CCS) (Barrett and Narayanan 2008; Roijers and Whiteson 2017) which is defined as the convex hull of the PCS. Details for computing the CCS of a set $\mathbf{V}$ with a linear program (LP) can be found in (Roijers and Whiteson 2017, Sec. 4.1.3). We say that a set of vectors $\mathbf{U}$ *dominates* another set of vectors $\mathbf{V}$, denoted by $\mathbf{U} \preceq \mathbf{V}$, if for all $\vec{v} \in \mathbf{V}$ there exists $\vec{u} \in \mathbf{U}$ such that $\vec{u} \preceq \vec{v}$.

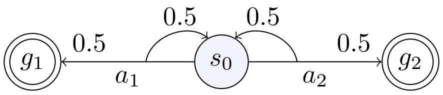

Figure 1: A MOSSP with action costs given by $\vec{C}(a_1) = [1, 0]$ and $\vec{C}(a_2) = [0, 1]$.

Given an MOSSP define the *optimal value function* $\mathbf{V}^*$ by $\mathbf{V}^*(s) = \mathrm{CCS}(\{\vec{V}^\pi(s) \mid \pi \text{ is a proper policy}\})$. Then we define a *solution* to the MOSSP to be any set of proper policies Π such that the function $\varphi : \Pi \to \mathbf{V}^*(s_0)$ with $\varphi(\pi) = \vec{V}^\pi(s_0)$ is a bijection. By choosing CCS as our coverage set operator, we may focus our attention to only non-dominated deterministic policies, where non-dominated stochastic policies are implicitly given by the points on the surface of the polyhedron drawn out by the CCS. In this way, we avoid having to explicitly compute infinitely many non-dominated stochastic policies.

To illustrate this statement, consider the MOSSP in Fig. 1 with actions a_1 and a_2 where $P(s_0|s_0, a_1) = P(g_1|s_0, a_1) = P(s_0|s_0, a_2) = P(g_2|s_0, a_2) = 0.5$ and $\vec{C}(a_1) = [1, 0]$ and $\vec{C}(a_2) = [0, 1]$. One solution consists of only two deterministic policies $\pi_1(s_0) = a_1$ and $\pi_2(s_0) = a_2$ with corresponding expected costs $[2, 0]$ and $[0, 2]$. Notice that there are uncountably many stochastic policies obtained by the convex combinations of π_1 and π_2, i.e., $\pi_t(a_1|s_0) = 1 - t, \pi_t(a_2|s_0) = t$ for $t \in [0, 1]$. The expected cost of each π_t is $[2 - 2t, 2t]$ and these stochastic policies do not dominate each other. Therefore, if the PCS is used instead of the CCS in the definition of optimal value function, then $\mathbf{V}^*(s_0)$ would be $\{[2 - 2t, 2t] \mid t \in [0, 1]\}$ and the corresponding solution would be $\{\pi_t \mid t \in [0, 1]\}$. The CCS allows us to compactly represent the potentially infinite PCS by storing only the deterministic policies: in this example, the actual solution is $\{\pi_1, \pi_2\}$ and the optimal value function is $\mathbf{V}^*(s_0) = \{[2, 0], [0, 2]\}$.

Value Iteration for SSPs

We finish this section by reviewing how to compute the optimal value function V^* for SSPs and extend these results to MOSSPs in the next section. The optimal (scalar) value function V^* for an SSP can be computed using the Value Iteration (VI) algorithm (Bertsekas and Tsitsiklis 1996): given an initial value function V^0 it computes the sequence $V^1, \dots, V^k$ where V^{t+1} is obtained by applying a Bellman backup, that is $V^{t+1}(g) = 0$ if $g \in G$ and, for $s \in S \setminus G$,

$$V^{t+1}(s) = \min_{a \in A} Q^{t+1}(s, a) \qquad (2)$$
$$Q^{t+1}(s, a) = C(a) + \sum_{s' \in S} P(s'|s, a)V^t(s').$$

VI guarantees that V^k converges to V^* as $k \to \infty$ under the following conditions: (i) for all $s \in S$, there exists a proper policy w.r.t. s; and (ii) all improper policies have infinite cost for at least one state. The former condition is known as the *reachability assumption* and it is equivalent

[1] We will denote sets of vectors or functions which map to sets of vectors with bold face, e.g. $\mathbf{V}$, and single vectors or functions which map to single vectors with vector notation, e.g. $\vec{V}$.

Algorithm 1: MOVI

```
  Data: MOSSP problem $P = (S, s_0, G, A, P, \vec{C})$,
        initial values $\mathbf{V}(s)$ for each state $s$ (default to
        $\mathbf{V}(s) = \{\vec{0}\}$), and consistency threshold $\varepsilon$.
1 while $\max_{s \in S} \mathit{res}(s) < \varepsilon$ do
2     for $s \in S$ do
3         if $s \in G$ then $\mathbf{V}_{\mathit{new}}(s) \leftarrow \{\vec{0}\}$ ;
4         else $\mathbf{V}_{\mathit{new}}(s) \leftarrow \mathrm{BellmanBackup}(s)$ ;
5         $\mathit{res}(s) \leftarrow D(\mathbf{V}, \mathbf{V}_{\mathit{new}})$
6     $\mathbf{V} \leftarrow \mathbf{V}_{\mathit{new}}$
7 return $\mathbf{V}$
```

to requiring that no dead ends exist, while the latter condition can be seen as preventing cycles with 0 cost. Notice that *finite-horizon Markov Decision Processes* (MDPs) and *infinite-horizon discounted MDPs* are special cases of SSPs in which all policies are guaranteed to be proper (Bertsekas and Tsitsiklis 1996).

3 Value Iteration for MOSSPs

Value Iteration has been extended to the MO setting in special cases of SSPs, e.g. infinite-horizon discounted MDPs (White 1982). In this section, we present the MO version of Value Iteration for the general MOSSP case. While the changes in the algorithm are minimal, the key contribution of this section is the generalisation of the assumptions needed for convergence. We start by generalising the Bellman backup operation from the SO, i.e. (2), to the MO case. For $s \in S \setminus G$ we have

$$\mathbf{V}^{t+1}(s) = \mathrm{CS}\Big(\bigcup_{a \in A} \mathbf{Q}^{t+1}(s,a)\Big) \tag{3}$$

$$\mathbf{Q}^{t+1}(s,a) = \{\vec{C}(a)\} \oplus \Big(\bigoplus_{s' \in S} P(s'|s,a)\mathbf{V}^{t}(s')\Big),$$

and $\mathbf{V}^{t+1}(g) = \{\vec{0}\}$ for $g \in G$ where $\oplus$ denotes the sum of two sets of vectors $\mathbf{V}$ and $\mathbf{U}$ defined as $\{\vec{u} + \vec{v} \mid \vec{u} \in \mathbf{U}, \vec{v} \in \mathbf{V}\}$, and $\bigoplus$ is the generalised version of $\oplus$ to several sets.

Alg. 1 illustrates the MO version of Value Iteration (MOVI) which is very similar to the single-objective VI algorithm with the notable difference that the convergence criterion is generalised to handle sets of vectors. The Hausdorff distance between two sets of vectors $\mathbf{U}$ and $\mathbf{V}$ is given by

$$D(\mathbf{U}, \mathbf{V}) = \max\big\{\max_{\vec{u} \in \mathbf{U}} \min_{\vec{v} \in \mathbf{V}} d(\vec{u}, \vec{v}), \max_{\vec{u} \in \mathbf{V}} \min_{\vec{v} \in \mathbf{U}} d(\vec{u}, \vec{v})\big\}$$

for some choice of metric d such as the Euclidean metric. We use this distance to define residuals in line 5. As with VI, MOVI converges to the optimal value function at the limit (White 1982; Barrett and Narayanan 2008) under certain strong assumptions presented in the next section. We can extract policies with the choice of a scalarising weight $\vec{w}$ from the value function (Barrett and Narayanan 2008).

Assumptions for the Convergence of MOVI

Similarly to the SO version of VI, MOVI requires that the reachability assumption holds; however, the assumption that improper policies have infinite cost needs to be generalised to the MO case. One option is to require that all improper policies cost $\vec{\infty}$ which we call the **strong improper policy assumption** and, if it holds with the reachability assumption, then MOVI (Alg. 1) is sound and complete for MOSSPs. The strong assumption is very restrictive because it implies that any cycle in an MOSSP must have a cost greater than zero in *all dimensions*; however, it is common for MOSSPs to have zero-cost cycles in one or more dimensions. For instance, in navigation domains where some actions such as wait, load and unload do not consume fuel and can be used to generate zero-fuel-cost loops.

Algorithm 2: MOVI under Assumption 1

```
  Data: MOSSP problem $P = (S, s_0, G, A, P, \vec{C})$,
        initial values $\mathbf{V}(s)$ for each state $s$ (default to
        $\mathbf{V}(s) = \{\vec{0}\}$), consistency threshold $\varepsilon$ and
        upper bound $\vec{b}$.
1 while $\max_{s \in S} \mathit{res}(s) < \varepsilon$ do
2     for $s \in S$ do
3         if $s \in G$ then $\mathbf{V}_{\mathit{new}}(s) \leftarrow \{\vec{0}\}$ ;
4         else $\mathbf{V}_{\mathit{new}}(s) \leftarrow \mathrm{BellmanBackup}_B(s)$ ;
5         $\mathit{res}(s) \leftarrow D(\mathbf{V}, \mathbf{V}_{\mathit{new}})$
6     $\mathbf{V} \leftarrow \mathbf{V}_{\mathit{new}}$
7 for $s \in S$ do $\mathbf{V}(s) = \mathbf{V}(s) \setminus \{\vec{b}\}$ ;
8 return $\mathbf{V}$
```

Another possible generalisation for the cost of improper policies is that they cost infinity in *at least one dimension*. This requirement is not enough as illustrated by the (deterministic) MOSSP in Fig. 2. The only deterministic proper policy is $\pi(s_0) = a_g$ with cost $[0, 1]$, and the other deterministic policy is $\pi'(s_0) = a_1, \pi'(s_1) = a_2$ which is improper and costs $[\infty, 0]$. Since neither $[0, 1]$ or $[\infty, 0]$ dominate each other, two issues arise: MOVI tries to converge to infinity and thus never terminates; and even if it did, the cost of an improper policy would be wrongly added to the CCS.

These issues stem from VI and its generalisations not explicitly pruning improper policies. Instead they rely on the assumption that improper policies have infinite cost to implicitly prune them. This approach is enough in the SO case because any proper policy dominates all improper policies since domination is reduced to the '$\leq$' operator; however, as shown in this example, the implicit pruning approach for improper policies is not correct for the MO case.

We address these issues by providing a version of MOVI that explicitly prunes improper policies and to do so we need a new assumption:

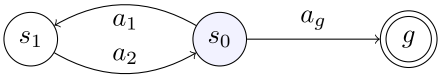

Figure 2: An MOSSP with action costs given by $\vec{C}(a_1) = [1, 0], \vec{C}(a_2) = [1, 0], \vec{C}(a_g) = [0, 1]$.

Assumption 1 (Weak Improper Policy Assumption)**.** *There exists a vector $\vec{b} \in \mathbb{R}^n_{\geq 0}$ such that for every improper policy π and $s \in S$, $\vec{V}^\pi(s) \not\preceq \vec{b}$, and for every proper policy π and $s \in S$, $\vec{V}^\pi(s) \preceq \vec{b}$ and $\vec{V}^\pi(s) \neq \vec{b}$.*

The weak assumption allows improper policies to have finite cost in some *but not all* dimensions at the expense of knowing an upper bound $\vec{b}$ on the cost of all proper policies. This upper bound $\vec{b}$ lets us infer that a policy π is improper whenever $\vec{V}^\pi(s) \not\preceq \vec{b}$, i.e., that $\vec{V}^\pi(s)$ is greater than $\vec{b}$ in at least one dimension.

Alg. 2 outlines the new MOVI where the differences can be found in lines 4 and 7. In line 4, we use a modified Bellman backup which detects vectors associated to improper policies and assigns them the upper bound $\vec{b}$ given by the weak improper policy assumption. The modified backup is given by

$$\mathbf{V}^{t+1}(s) = \mathrm{CS}_B\Big(\bigcup_{a\in A} \mathbf{Q}_B^{t+1}(s,a)\Big) \tag{4}$$

$$\mathbf{Q}_B^{t+1}(s,a) = \{\vec{C}(a)\} \oplus_B \Big({\bigoplus_{s'\in S}}_B P(s'|s,a) \star_B \mathbf{V}^t(s')\Big)$$

where $\star_B$ and $\oplus_B$ are generalisations to sets of vectors of

$$c \star_B \vec{v} = \begin{cases} \vec{b} & \text{if } \vec{v} \not\preceq \vec{b} \\ c\vec{v} & \text{else} \end{cases} \quad \vec{u} \oplus_B \vec{v} = \begin{cases} \vec{b} & \text{if } \vec{u} \oplus \vec{v} \not\preceq \vec{b} \\ \vec{u} \oplus \vec{v} & \text{else.} \end{cases}$$

Also define CS_B to be the modified coverage set operator which does not prune away $\vec{b}$ if it is in the input set. The modified backup (4) enforces a cap on value functions (i.e., $\mathbf{V}^t(s) \preceq \{\vec{b}\}$) through the operator $\oplus_B$. This guarantees that a finite fixed point exists for all $\vec{V}^\pi(s)$ and, as a result, that Alg. 2 terminates. Once the algorithm converges, we need to explicitly prune the expected cost of improper policies which is done in line 7. By Assumption 1, we have that no proper policy costs $\vec{b}$ thus we can safely remove $\vec{b}$ from the solution. Note that the overhead in applying this modified backup and post-processing pruning is negligible. Theorem 3.1 shows that MOVI converges to the correct solution.

Theorem 3.1. *Given an MOSSP in which the reachability and weak improper policy assumptions hold for an upper bound $\vec{b}$, and given a set of vectors $\mathbf{V}^0$ such that $\mathbf{V}^0 \preceq \mathbf{V}^*$, the sequence $\mathbf{V}^1, \ldots, \mathbf{V}^k$ computed by MOVI converges to the MOSSP solution $\mathbf{V}^*$ as $k \to \infty$.*

Proof sketch. By the assumption that $\mathbf{V}^0 \preceq \mathbf{V}^*$, we have that MOVI will not prune away vectors associated with proper policies which contribute to a solution. If $\mathbf{V}^0 \not\preceq \mathbf{V}^*$, e.g., $\mathbf{V}^0(s) = \{\vec{b}\}$ for all s, then Alg. 2 is not guaranteed to find the optimal value function since it will incorrectly prune proper policies. Otherwise, we have that $\mathbf{V}^1, \ldots, \mathbf{V}^k$ converges to $\mathbf{V}^\dagger = \mathbf{V}^* \cup \{\vec{b}\}$ if the original MOVI in Alg. 1 does not converge, and $\mathbf{V}^\dagger = \mathbf{V}^*$ otherwise. For example, $\mathbf{V}^* = \{[0,1]\}$ in the example seen in Fig. 2 but $\mathbf{V}^\dagger = \{[2,2],[0,1]\}$ if we choose $\vec{b} = [2,2]$.

This convergence result follows by noticing that, by definition of the modified backup in (4), every vector in $\mathbf{V}^t(s)$ for all t dominates $\vec{b}$. We may then apply the proof for the convergence of MOVI with convex coverage sets by Barrett and Narayanan (2008) which reduces to the convergence of scalarised SSPs using VI in the limit, of which there are finitely many since the number of deterministic policies is finite. Here, we have for every $\vec{w}$ that $\vec{w} \cdot \vec{b}$ is an upper bound for the problem scalarised by $\vec{w}$. Finally, we have that line 7 removes $\vec{b}$ from $\mathbf{V}^\dagger$ such that we correctly return $\mathbf{V}^*$. □

Note that the original proof for MOMDPs by Barrett and Narayanan (2008) does not directly work for MOSSPs as some of the scalarised SSPs have a solution of infinite expected cost such that VI never converges. The upper bound $\vec{b}$ we introduce solves this issue as achieving this bound is the same as detecting improper policies. The proof remains correct in the original setting applied to discounted MOMDPs in which there are no improper policies.

Relaxing the Reachability Assumption

The reachability assumption can also be relaxed by transforming an MOSSP with dead ends into a new MOSSP without dead ends. Formally, given an MOSSP $(S, s_0, G, A, P, \vec{C})$ with dead ends, let $A' = A \cup \{\text{give-up}\}$; $P(s_g|\text{give-up}, s) = 1$ for all $s \in S$ and any $s_g \in G$; $\vec{C}'(a) = [\vec{C}(a) : 0]$ for all $a \in A$; and $\vec{C}'(\text{give-up}) = [\vec{0} : 1]$.

Then the MOSSP $(S, s_0, G, A', P, \vec{C}')$ does not have dead ends (i.e., the reachability assumption holds) since the give-up action is applicable in all states $s \in S$. This transformation is similar to the *finite-penalty transformation* for SSPs (Trevizan, Teichteil-Königsbuch, and Thiébaux 2017); however it does not require a large penalty constant. Instead, the MOSSP transformation uses an extra cost dimension to encode the cost of giving up and the value of the $n+1$-th dimension of $\vec{C}'(\text{give-up})$ is irrelevant as long as it is greater than 0. Since the give-up action can only be applied once, defining the cost of give-up as 1 let us interpret the $n+1$-th dimension of any $\vec{V}^\pi(s)$ as the probability of giving up when following π from s.

4 Heuristic Search

Value iteration gives us one method for solving MOSSPs but is limited by the fact that it requires enumerating over the whole state space. This is impractical for planning problems, as their state space grows exponentially with the size of the problem encoding. This motivates the need for heuristic search algorithms in the vein of LAO* and LRTDP for SSPs (Hansen and Zilberstein 2001; Bonet and Geffner 2003), which perform backups on a promising subset of states at each iteration. To guide the choice of the states to expand and explore at each iteration, they use heuristic estimates of state values initialised when the state is first encountered. In this section, we extend the concept of heuristics to the more general MOSSP setting, discuss a range of ways these heuristics can be constructed, and provide multi-objective versions of LAO* and LRTDP.

Heuristics

For the remainder of the paper, we will call a set of vectors a value. Thus, a heuristic value for a state is a set of vectors $\mathbf{H}(s) \subset \mathbb{R}^n_{\geq 0}$. We implicitly assume that any heuristic value $\mathbf{H}(s)$ has already been pruned with some coverage set operator. The definition of an admissible heuristic for MOSSPs should be at most as strong as the definition for deterministic MO search (Mandow and Pérez-de-la-Cruz 2010).

Definition 4.1. A heuristic $\mathbf{H}$ for an MOSSP is *admissible* if $\forall s \in S \setminus G, \mathbf{H}(s) \preceq \mathbf{V}^*(s)$ where $\mathbf{V}^*$ is the optimal value function, and $\forall g \in G, \mathbf{H}(g) = \{\vec{0}\}$.

For example, if for some MOSSP we have $\mathbf{V}^*(s) = \{[0, 2]\}$ for a state s, then $\mathbf{H}(s) = \{[0, 1], [3, 0]\}$ and $\mathbf{H}(s') = \{\vec{0}\}$ for all other s' is an admissible heuristic. As in the single-objective case, admissible heuristics ensure that the algorithms below converge to near optimal value functions for $\varepsilon > 0$ with finitely many iterations and to optimal value functions with possibly infinitely many iterations when $\varepsilon = 0$.

Definition 4.2. A heuristic $\mathbf{H}$ for an MOSSP is *consistent* if we have for all $s \in S, a \in A$ that $\mathbf{H}(s) \preceq \{\vec{C}(a)\} \oplus (\bigoplus_{s' \in S} P(s'|s, a)\mathbf{H}(s'))$.

The definition of consistent heuristic is derived and generalised from the definition of consistent heuristics for deterministic search: $h(s) \leq c(s, a, s') + h(s')$. The main idea is that assuming non-negative costs, state costs increase monotonically which results in no re-expansions of nodes in A* search. Similarly, we desire monotonically increasing value functions for faster convergence.

We now turn to the question of constructing domain-independent heuristics satisfying these properties from the encoding of a planning domain. This question has only recently been addressed in the *deterministic* multi-objective planning setting (Geißer et al. 2022): with the exception of this latter work, MO heuristic search algorithms have been evaluated using "ideal-point" heuristics, which apply a single-objective heuristic h_i to each objective in isolation, resulting in a single vector $\mathbf{H}_{\textit{ideal}}(s) = \{[h_1(s), \dots h_n(s)]\}$. Moreover, informative domain-independent heuristics for single objective SSPs are also relatively new (Trevizan, Thiébaux, and Haslum 2017; Klößner and Hoffmann 2021): a common practice was to resort to classical planning heuristics obtained after determinisation of the SSP (Jimenez, Coles, and Smith 2006).

We consider a spectrum of options when constructing domain-independent heuristics for MOSSPs, which we instantiate using the most promising families of heuristics identified in (Geißer et al. 2022) for the deterministic case: critical paths and abstraction heuristics. All heuristics are consistent and admissible unless otherwise mentioned.

- The baseline option is to ignore both the MO and stochastic aspects of the problem, and resort to an ideal-point heuristic constructed from the determinised problem. In our experiments below, we apply the classical h^{max} heuristic (Bonet and Geffner 2001) to each objective and call this $\mathbf{H}^{\textit{max}}_{\textit{ideal}}$.
- A more elaborate option is to consider only the stochastic aspects of the problem, resulting in an ideal-point SSP heuristic. In our experiments, we apply the recent SSP canonical PDB abstraction heuristic by Klößner and Hoffmann (2021) to each objective which we call $\mathbf{H}^{\textit{pdb2}}_{\textit{ideal}}$ and $\mathbf{H}^{\textit{pdb3}}_{\textit{ideal}}$ for patterns of size 2 and 3, respectively.
- Alternatively, one might consider only the multi-objective aspects, by applying some of the MO deterministic heuristics (Geißer et al. 2022) to the determinised SSP. The heuristics we consider in our experiments are the MO extension of h^{max} and canonical PDBs: $\mathbf{H}^{\textit{comax}}_{\textit{mo}}$, $\mathbf{H}^{\textit{pdb2}}_{\textit{mo}}$, and $\mathbf{H}^{\textit{pdb3}}_{\textit{mo}}$. These extend classical planning heuristics by using MO deterministic search to solve subproblems and combining solutions by taking the maximum of two sets of vectors with the admissibility preserving operator comax (Geißer et al. 2022).
- The best option is to combine the power of SSP and MO heuristics. We do so with a novel heuristic $\mathbf{H}^{\textit{pdb}}_{\textit{mossp}}$ which extends the SSP PDB abstraction heuristic (Klößner and Hoffmann 2021) to the MO case by using an MO extension of topological VI (TVI) (Dai et al. 2014) to solve each projection and the comax operator to combine the results. Our experiments use $\mathbf{H}^{\textit{pdb2}}_{\textit{mossp}}$ and $\mathbf{H}^{\textit{pdb3}}_{\textit{mossp}}$.

(i)MOLAO*

Readers familiar with the LAO* heuristic search algorithm and the multi-objective backup can convince themselves that an extension of LAO* to the multi-objective case can be obtained by replacing the backup operator with the multi-objective version. This idea was first outlined by Bryce, Cushing, and Kambhampati (2007) for *finite-horizon MDPs* which is a special case of SSPs without improper policies. In the same vein as (Hansen and Zilberstein 2001), we provide MOLAO* alongside an 'improved' version, iMOLAO*, in Alg. 3 and 4 respectively.

MOLAO* begins in line 1 by lazily assigning an initial value function $\mathbf{V}$ to each state with the heuristic function, as opposed to explicitly initialising all initial values at once. Π is a dictionary representing our partial solution which maps states to sets of optimal actions corresponding to their current value function. The main while loop of the algorithm terminates once there are no nongoal states on the frontier as described in line 2, at which point we have achieved a set of closed policies.

The loop begins with lines 3-5 by removing a nongoal state s from the frontier representing a state on the boundary of the partial solution graph, and adding it to the interior set I. The nodes of the partial solution graph are partial solution states, and the edges are the probabilistic transitions under all partial solution actions. Next in line 6 we add the set of yet unexplored successors of s to the frontier: any state $s' \in S \setminus I$ such that $\exists a \in A, P(s'|s, a) > 0$. Then we extract the set Z of all ancestor states of s in the partial solution Π using graph search in line 7. We run MOVI to ε-consistency on the MOSSP problem restricted to the set of states Z and update the value functions for the corresponding states in line 8. The partial solution is updated in line 9 by extracting

Algorithm 3: MOLAO*

```
   Data: MOSSP problem $P = (S, s_0, G, A, P, \vec{C})$,
         heuristic $\mathbf{H}$, and consistency threshold $\varepsilon$
1  $\mathbf{V} \leftarrow \mathbf{H}$; $\Pi \leftarrow \emptyset$; $F \leftarrow \{s_0\}$ ; $I \leftarrow \emptyset$; $N \leftarrow \{s_0\}$
2  while $(F \cap N) \setminus G \neq \emptyset$ do
3      $s \leftarrow$ any element from $(F \cap N) \setminus G$
4      $F \leftarrow F \setminus \{s\}$
5      $I \leftarrow I \cup \{s\}$
6      $F \leftarrow F \cup (\text{successors}(s) \setminus I)$
7      $Z \leftarrow \text{ancestorStates}(s, \Pi)$
8      $\mathbf{V}|_Z \leftarrow \text{MOVI}(P|_Z, \mathbf{V}|_Z, \varepsilon)$
9      for $s \in Z$ do $\Pi(s) \leftarrow \text{getActions}(s, \mathbf{V})$ ;
10     $N \leftarrow \text{solutionGraph}(s_I, \Pi)$
11 return V
```

Algorithm 4: iMOLAO*

```
   Data: MOSSP problem $P = (S, s_0, G, A, P, \vec{C})$,
         heuristic $\mathbf{H}$, and consistency threshold $\varepsilon$
1  $\mathbf{V} \leftarrow \mathbf{H}$; $\Pi \leftarrow \emptyset$; $F \leftarrow \{s_0\}$ ; $I \leftarrow \emptyset$; $N \leftarrow \{s_0\}$
2  while $((F \cap N) \setminus G \neq \emptyset) \wedge (\max_{s \in N} res(s) < \varepsilon)$ do
3      $F = \emptyset$
4      $N \leftarrow \text{postorderTraversalDFS}(s_I, \Pi)$
5      for $s \in N$ in the computed order do
6          $\mathbf{V}(s) \leftarrow \text{BellmanBackup}(s)$
7          $\Pi(s) = \text{getActions}(s, \mathbf{V})$
8          if $s \notin I$ then $F = F \cup \{s\}$ ;
9          $I = I \cup \{s\}$
10 return V
```

the actions corresponding to the value function with

$$\text{getActions}(s, \mathbf{V}) = \{a \in A \mid \mathbf{Q}(s, a) \cap \mathbf{V}(s) \neq \emptyset\} .$$

This function can be seen as a generalisation of $\arg\min$ to the MO case where here we select the actions whose $\mathbf{Q}$ value at s contribute to the current value $\mathbf{V}(s)$. Next, we extract the set of states N corresponding to all states reachable from s_0 by the partial solution Π in line 10.

The completeness and correctness of MOLAO* follows from the same reasoning as LAO* extended to the MO case. Specifically, we have that given an admissible heuristic the algorithm achieves ε-consistency upon termination.

One potential issue with MOLAO* is that we may waste a lot of backups while running MOVI to convergence several times on partial solution graphs which do not end up contributing to the final solution. The original authors of LAO* proposed the iLAO* algorithm to deal with this. We provide the MO extension, namely iMOLAO*. The main idea with iMOLAO* is that we only run one set of backups every time we (re-)expand a state instead of running VI to convergence in the loop in lines 5 to 9. Backups are also performed asynchronously using DFS postorder traversal of the states in the partial solution graph computed in line 4, allowing for faster convergence times.

Algorithm 5: MOLRTDP

```
   Data: MOSSP problem $P = (S, s_0, G, A, P, \vec{C})$,
         heuristic $\mathbf{H}$, and consistency threshold $\varepsilon$
   procedure MOLRTDP$(P, \varepsilon, \mathbf{H})$
1      $\mathbf{V} \leftarrow \mathbf{H}$
2      while $\neg s_0.solved$ do
3          $visited \leftarrow \emptyset$
4          $s \leftarrow s_0$
5          while $\neg s.solved$ do
6              $visited.push(s)$
7              if $s \in G$ then break ;
8              $\mathbf{V}(s) \leftarrow \text{BellmanBackup}(s)$
9              $a \leftarrow \text{sampleUnsolvedGreedyAction}(s)$
10             $s \leftarrow \text{sampleUnsolvedNextState}(s, a)$
11         while $\neg visited.empty()$ do
12             $s \leftarrow visited.pop()$
13             if $\neg\text{checkSolved}(s)$ then break ;
14     return V
   routine checkSolved$(s)$
1      $rv \leftarrow true$; $open \leftarrow \emptyset$; $closed \leftarrow \emptyset$
2      if $\neg s.solved$ then $open.\text{push}(s)$ ;
3      while $\neg open.empty()$ do
4          $s \leftarrow open.pop()$
5          if $res(s) > \varepsilon$ then
6              $rv \leftarrow false$
7              continue
8          for $a \in \text{getActions}(s, \mathbf{V})$ do
9              for $s' \in \text{successors}(s, a)$ do
10                 if $\neg s'.solved \wedge s' \notin open \cup closed$
                     then $open.push(s')$ ;
11     if $rv$ then for $s \in closed$ do $s.solved = true$ ;
12     else
13         while $closed \neq \emptyset$ do
14             $s \leftarrow closed.pop()$
15             $\mathbf{V}(s) \leftarrow \text{BellmanBackup}(s)$
16     return $rv$
```

To summarise, the two main changes required to extend (i)LAO* to the MO case are (1) replacing the backup operator with the MO version, and (2) storing possibly more than one greedy action at each state corresponding to incomparable vector $\mathbf{Q}$-values, resulting in a larger set of successors associated to each state. These ideas can be applied to other asynchronous VI methods such as Prioritised VI (Wingate and Seppi 2005) and Topological VI (Dai et al. 2014).

MOLRTDP

LRTDP (Bonet and Geffner 2003) is another heuristic search algorithm for solving SSPs. The idea of LRTDP is to perform random trials using greedy actions based on the current value function or heuristic for performing backups, and labelling states as solved when the consistency threshold has been reached in order to gradually narrow down the search space and achieve a convergence criterion. A multi-objective extension of LRTDP is not as straightforward as extending the backup operator given that each state has possibly more than one greedy action to account for. The two main changes from the original LRTDP are (1) the sampling of a random greedy action before sampling successor states in the main loop, and (2) defining the descendants of a state by considering successors of all greedy actions (as opposed to just one in LRDTP). Alg. 5 outlines the MO extension of LRTDP.

MOLRTDP consists of repeatedly trialing paths through the state space and performing backups until the initial state is marked as solved. Trials are run by randomly sampling a greedy action a from $\mathrm{getActions}(s, \mathbf{V})$ at each state s followed by a next state sampled from the probability distribution of a until a goal state is reached as in the first inner loop from lines 5 to 10. The second inner loop from lines 11 to 13 calls the $\mathrm{checkSolved}$ routine in the reverse trial order to label states as solved by checking whether the residual of all its descendant states under greedy actions are less than the convergence criterion.

The $\mathrm{checkSolved}$ routine begins with inserting s into the open set if it has not been solved, and returns true otherwise due to line 2. The main loop in lines 3 to 10 collects the descendent states under all greedy actions (lines 8-10) and checks whether the residual of all the descendent states are small (lines 5-7). If true, all descendents are labelled as solved as in line 11. Otherwise, backups are performed in the reverse order of explored states as in lines 12 to 15.

The completeness and correctness of MOLRTDP follows similarly from its single objective ancestor. We note specifically that the labelling feature works similarly in the sense that whenever a state is marked as solved, it is known for certain that all its descendents' values are ε-consistent and remain unchanged when backups are performed elsewhere.

5 Experiments

In this section we empirically evaluate the different algorithms and heuristics for MOSSPs in several domains. Since no benchmark domains for MOSSPs exist, we adapt domains from a variety of sources to capture challenging features of both SSPs and MO deterministic planning. Our benchmark set is introduced in the next section and it is followed by our experiments setup and analysis of the results.

Benchmarks

k-d Exploding Blocksworld Exploding Blocksworld (ExBw) was first introduced by Younes et al. (2005) and later slightly modified for the IPPC'08 (Bryce and Buffet 2008). In ExBw, a robot has to pick up and stack blocks on top of each other to get a target tower configuration. Blocks have a chance of detonating and destroying blocks underneath or the table. We consider a version with no dead ends using an action to repair the table (Trevizan et al. 2017). We extend ExBw to contain multi-objective costs. ExBw-2d has two objectives: the number of actions required to stack the blocks and number of times we need to repair the table. ExBw-3d has an additional action to repair blocks with an objective to minimise the number of block repairs.

MO Search and Rescue Search and Rescue (SAR-n) was first introduced by Trevizan et al. (2017) as a *Constrained* SSP. The goal is to find, board and escort to a safe location any one survivor in an $n \times n$ grid as quickly as possible, constrained by a fuel limit. Probabilities are introduced when modelling fuel consumption and partial observability of whether a survivor exists in a given location. The location of only one survivor is known for certain. We extend the problem by making fuel consumption as an additional objective instead of a constraint. A solution for a SAR MOSSP is a set of policies with different trade-offs between fuel and time.

MO Triangle Tireworld Triangle Tireworld, introduced by Little and Thiébaux (2007) as a probabilistically interesting problem, consists of a triangular grid of locations. The goal is to travel from an initial to a target location where each location has a probability of getting a flat tire. Some locations contain a spare tire which the agent can load into the car for future use. These problems have dead ends and they occur when a tire is flat and no spare is available. We apply the give-up transformation from Section 3 resulting in an MOSSP with two objectives: total number of actions and probability of using the give-up action.

Probabilistic VisitAll The deterministic MO VisitAll (Geißer et al. 2022) is a TSP problem on a grid, where the agents must collectively visit all locations, and each agent's objective is to minimise its own number of moves. This problem is considered MO interesting because any admissible ideal-point heuristic returns the zero vector for all states since it is possible for a single agent to visit all locations while no actions are performed by the other agents. We extend this domain by adding a probabilistic action *move-risky* which has probability 0.5 of acting as the regular move action and 0.5 of teleporting the agent back to its initial location. The cost of the original move action was changed to 2 while the cost of the move-risky action is 1.

VisitAllTire This domain combines features of both the probabilistic Tireworld and the deterministic MO VisitAll domains into one that is probabilistically and MO interesting. The underlying structure and dynamics is the same as the deterministic MO VisitAll except that the move action now can result in a flat tire with probability 0.5. We also added the actions for loading and changing tires for each of the agents. Similarly to Triangle Tireworld, the problems in this domain can have dead ends when a flat tire occurs and no spare tire is available. Applying the give-up transformation from Section 3 to this domain results in $k + 1$ cost functions where k is the number of agents.

Setup

We implemented the MO versions of the VI, TVI, (i)LAO* and LRTDP algorithms and the MO version of the PDB abstraction heuristics ($\mathbf{H}^{pdb}_{mossp}$) in C++.[2] PDB heuristics are computed using TVI, $\varepsilon = 0.001$ and $\vec{b} = 1\vec{00}$. We include in our experiments the following heuristics for deterministic

[2] Code at https://github.com/DillonZChen/cpp-mossp-planner

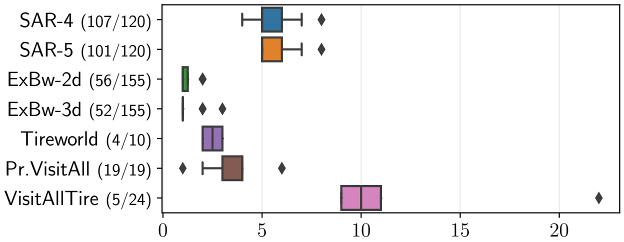

Figure 3: Boxplot of CCS size of instances across domains which have been solved at least once. Number of solved and total instances for each domain is indicated in parentheses.

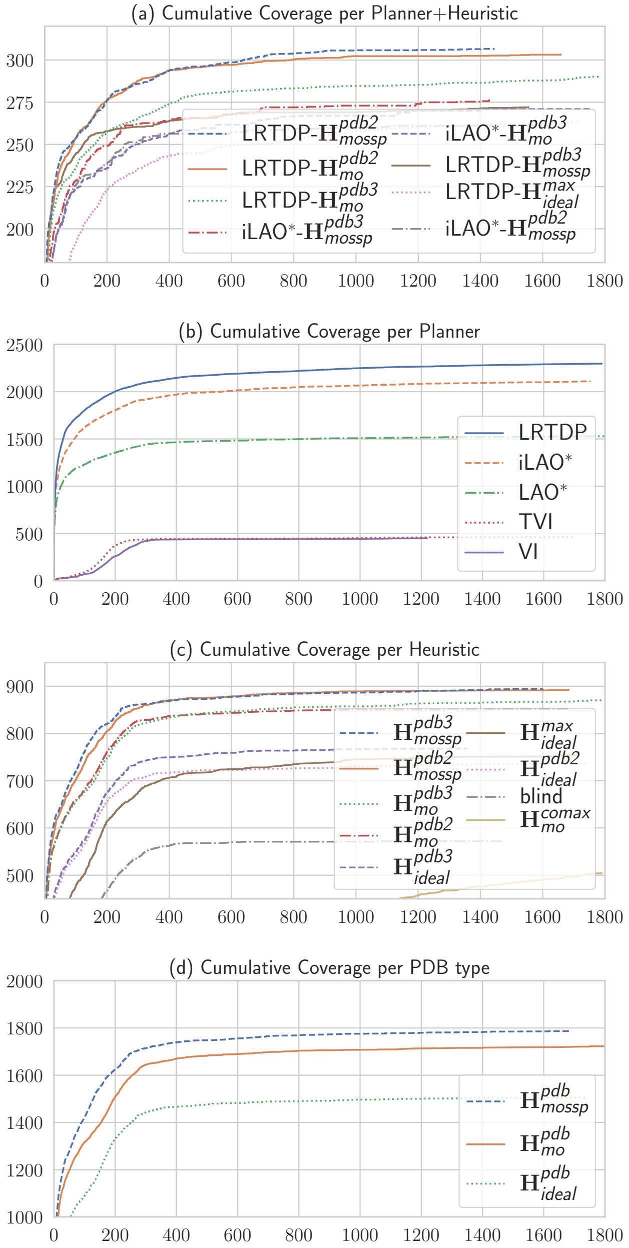

Figure 4: Cumulative coverage of: (a) planners and heuristics combinations (low-performing planners omitted); (b) planners only, i.e., summation across different heuristics; (c) heuristics only, i.e. summation across different planners; (d) PDB approaches considered, i.e., summation across the ideal-point, MO, and MOSSP approaches. Notice that the x-axis is the same for all plots but the y-axis is different and might not start at 0.

MO planning from (Geißer et al. 2022): the ideal-point version of $h^{\max}$ ($\mathbf{H}^{max}_{ideal}$); the MO extension of $h^{\max}$ ($\mathbf{H}^{comax}_{mo}$); and the MO canonical PDBs of size 2 and 3 ($\mathbf{H}^{pdb2}_{mo}$ and $\mathbf{H}^{pdb3}_{mo}$). The SO PDB heuristics for SSPs from (Klößner and Hoffmann 2021) of size 2 and 3 ($\mathbf{H}^{pdb2}_{ideal}$ and $\mathbf{H}^{pdb3}_{ideal}$) are also included in our experiments. All problem configurations are run with a timeout of 1800 seconds, memory limit of 4GB, and a single CPU core. The experiments are run on a cluster with Intel Xeon 3.2 GHz CPUs. We used CPLEX version 22.1 as the LP solver for computing CCS. The consistency threshold is set to $\varepsilon = 0.001$ and we set $\vec{b} = \vec{100}$. Each experimental configuration is run 6 times and averages are taken to reduce variance in the data.

We also modify the inequality constraint in Alg. 4.3 from Roijers and Whiteson 2017 for computing CCS to $\vec{w} \cdot (\vec{v} - \vec{v}') + x \leq -\lambda, \forall \vec{v}' \in \mathbf{V}$ with $\lambda = 0.01$ to deal with inevitable numerical precision errors when solving the LP. If the λ term is not added, the function may fail to prune unnecessary points such as those corresponding to stochastic policies in the CCS, resulting in slower convergence. However, an overly large λ may return incorrect results by discarding important points in the CCS. One may alternatively consider the term as an MO parameter for trading off solution accuracy and search time, similarly to the ε parameter.

Fig. 3 shows the distribution of CCS sizes for different domains. Recall that the CCS implicitly represents a potentially infinite set of stochastic policies and their value functions. As a result, a small CCS is sufficient to dominate a large number of solutions, for instance, in Triangle Tireworld, a CCS of size 3 to 4 is enough to dominate all other policies even though the number of deterministic policies for this domain is double exponential in its parameter n.

Results

Fig. 4 summarises our results by showing the cumulative coverage of different planners and heuristics across all considered problems. The following is a summary of our findings (we omit the MO in the planner names for simplicity):

What is the best planner and heuristic combination? Referring to the overall cumulative coverage plot in Fig. 4a and coverage tables in Tab. 1, we notice that LRTDP+$\mathbf{H}^{pdb2}_{mossp}$ performs best followed by LRDTP+$\mathbf{H}^{pdb2}_{mo}$, LRTDP+$\mathbf{H}^{pdb3}_{mo}$ and iLAO*+$\mathbf{H}^{pdb3}_{mossp}$. The ranking of the top 3 planners remain the same if we normalise the coverage by domain.

We note that the considered Exbw3 problems had several improper policies resulting in slow convergence of TVI for solving PDBs of size 3. This is due to Assumption 1 and the

planner + heuristic		coverage
LRTDP	$\mathbf{H}^{pdb2}_{mossp}$	307.6 ± 2.7
LRTDP	$\mathbf{H}^{pdb2}_{mo}$	302.8 ± 2.1
LRTDP	$\mathbf{H}^{pdb3}_{mo}$	291.2 ± 1.3
iLAO*	$\mathbf{H}^{pdb3}_{mossp}$	276.7 ± 0.7
iLAO*	$\mathbf{H}^{pdb3}_{mo}$	272.0 ± 0.0
LRTDP	$\mathbf{H}^{pdb3}_{mossp}$	271.2 ± 1.7
LRTDP	$\mathbf{H}^{max}_{ideal}$	263.2 ± 0.7
iLAO*	$\mathbf{H}^{pdb2}_{mo}$	262.0 ± 0.0

heuristic	coverage
$\mathbf{H}^{pdb3}_{mossp}$	893.5 ± 1.9
$\mathbf{H}^{pdb2}_{mossp}$	893.3 ± 3.2
$\mathbf{H}^{pdb3}_{mo}$	871.2 ± 1.8
$\mathbf{H}^{pdb2}_{mo}$	851.8 ± 2.1
$\mathbf{H}^{pdb3}_{ideal}$	768.7 ± 0.7
$\mathbf{H}^{max}_{ideal}$	755.2 ± 0.7
$\mathbf{H}^{pdb2}_{ideal}$	737.2 ± 1.2
blind	572.9 ± 1.4

planner	coverage
LRTDP	2297.3 ± 3.0
iLAO*	2111.7 ± 0.7
LAO*	1529.0 ± 1.2
TVI	460.0 ± 0.5
VI	450.3 ± 0.7

PDB	coverage
$\mathbf{H}^{pdb}_{mossp}$	1786.8 ± 1.6
$\mathbf{H}^{pdb}_{mo}$	1723.0 ± 2.0
$\mathbf{H}^{pdb}_{ideal}$	1505.8 ± 1.9

Table 1: Average marginalised cumulative coverages with 95% confidence intervals. Higher values are better.

chosen $\vec{b} = \vec{100}$ for our experiments which that requires the cost of improper policies in each PDB to cost at least 100 in one of its dimensions. Ignoring the Exbw3 domain, the top 3 configurations are LRTDP+$\mathbf{H}^{pdb3}_{mossp}$, LRTDP+$\mathbf{H}^{pdb2}_{mossp}$ and iLAO*+$\mathbf{H}^{pdb3}_{mossp}$ and their 95% confidence intervals all overlap. The ranking of the top few planners remain the same if the coverage is normalised.

What planner is best? To answer this question, we look at the cumulative coverage marginalised over heuristics, that is, we sum the coverage of a planner across the different heuristics. The results are shown in Fig. 4b and Tab. 1 and the best three planners in order are LRTDP, iLAO* and LAO*. Notice that the difference in coverage between LRTDP and iLAO* is at 185.6 solved instances while the difference between TVI and VI is only 9.7. The ranking between planners remains the same when the coverage is normalised per domain. Considering domains individually, LRTDP is the best planner for Exbw and VisitAllTire while for iLAO* is the best planner for SAR and Probabilistic VisitAll. For MO Tireworld, both LRTDP and iLAO* are tied in first place.

What heuristic is best? The cumulative coverage marginalised over planners for selected heuristics is shown in Fig. 4c and Tab. 1. The MOSSP PDB heuristic of size 3 ($\mathbf{H}^{pdb3}_{mossp}$) has the best performance followed by $\mathbf{H}^{pdb2}_{mossp}$, $\mathbf{H}^{pdb3}_{mo}$ and $\mathbf{H}^{pdb2}_{mo}$. We note there is a large overlap in the 95% confidence intervals of $\mathbf{H}^{pdb2}_{mossp}$ and $\mathbf{H}^{pdb3}_{mossp}$ due to the slow computation of $\mathbf{H}^{pdb3}_{mossp}$ on Exbw3. The overlap disappears when we remove the Exbw3 results with coverage and confidence intervals given by 862.9 ± 3.0 and 807.6 ± 2.9 for $\mathbf{H}^{pdb3}_{mossp}$ and $\mathbf{H}^{pdb2}_{mossp}$, respectively. We further quantify heuristic accuracy using the directed Hausdorff distance between heuristic and optimal values in Tab. 2. We notice that $\mathbf{H}^{comax}_{mo}$ achieves strong accuracy relative to other heuristics but has the worst coverage of 504.5 due to its high computation time.

		Crit. Path		Abstractions					
	blind	$\mathbf{H}^{max}_{ideal}$	$\mathbf{H}^{comax}_{mo}$	$\mathbf{H}^{pdb2}_{ideal}$	$\mathbf{H}^{pdb3}_{ideal}$	$\mathbf{H}^{pdb2}_{mo}$	$\mathbf{H}^{pdb3}_{mo}$	$\mathbf{H}^{pdb2}_{mossp}$	$\mathbf{H}^{pdb3}_{mossp}$
SAR-4	100	97	44	93	92	44	44	38	**26**
SAR-5	100	97	43	93	92	43	43	37	**27**
ExBw-2d	100	59	**22**	51	45	51	45	51	45
ExBw-3d	100	58	**22**	52	44	52	44	52	44
Tireworld	100	100	68	100	100	68	68	57	**13**
Pr.VisitAll	100	100	54	100	100	61	53	22	**12**
VisitAllTire	100	100	50	100	100	66	**45**	66	**45**

Table 2: The mean relative error (%) of the heuristic value relative to the optimal value at the initial state computed with the *directed* Hausdorff distance divided by the norm of the largest vector in the optimal value: $\max_{\vec{v}\in\mathbf{V}^*} \min_{\vec{u}\in\mathbf{H}} d(\vec{v},\vec{u}) / \max_{\vec{v}\in\mathbf{V}^*} \|\vec{v}\|$. Only solved instances for which all heuristics were computed for the initial state within the time limit were considered. Best value in each row is indicated in bold. Lower values are better.

What feature of MOSSPs is more important to capture in a heuristic? To answer this question, consider the performance of the 3 classes of PDB heuristics: probabilistic-only PDBs ($\mathbf{H}^{pdb}_{ideal}$), MO only PDBs ($\mathbf{H}^{pdb}_{mo}$), and MO probabilistic PDBs ($\mathbf{H}^{pdb}_{mossp}$). The cumulative coverage marginalised over the different PDB heuristics shown in Fig. 4d and Tab. 1 highlights the effectiveness of MO PDBs. $\mathbf{H}^{pdb}_{mossp}$ is able to solve 63.8 and 281 more instances than $\mathbf{H}^{pdb}_{mo}$ and $\mathbf{H}^{pdb}_{ideal}$, respectively. The ranking remains the same when the coverage is normalised per domain. Moreover, notice in Tab. 2 that $\mathbf{H}^{pdb}_{mo}$ heuristics are at least as informative as $\mathbf{H}^{pdb}_{ideal}$ heuristics on all domains. Lastly, we note that an admissible ideal point heuristic is upper bounded by $\{\vec{u}\}$ where $\vec{u}_i = \min_{\vec{v}\in\mathbf{V}^*} \vec{v}_i$ for $i = 1, \ldots, n$. These results suggest that it is more important for a heuristic to maintain the MO cost structure of MOSSPs than the stochastic structure of actions.

6 Conclusion

In this work we define a new general class of problems, namely multi-objective stochastic shortest path problems (MOSSPs). We adapt MOVI which was originally constructed for solving multi-objective Markov Decision Processes to our MOSSP setting with conditions for convergence. We further design new, more powerful heuristic search algorithms (i)MOLAO* and MOLRTDP for solving MOSSPs. The algorithms are complemented by a range of MOSSP heuristics and an extensive set of experiments on several benchmarks with varying difficulty and features. Through our evaluation, we can conclude that MOLRTDP is the best performing MOSSP solver, and abstraction heuristics which consider both the MO and probabilistic aspects of MOSSPs the best performing MOSSP heuristic. Our future work includes adding probabilistic LTL constraints as additional multi-objectives in order to compute solutions to MO-PLTL SSPs (Baumgartner, Thiébaux, and Trevizan 2018) that are robust to the choice of probability thresholds.

Acknowledgements

This work was supported by ARC project DP180103446, *On-line planning for constrained autonomous agents in an uncertain world*. The computational resources for this project were provided by the Australian Government through the National Computational Infrastructure (NCI) under the ANU Startup Scheme.

References

Barrett, L.; and Narayanan, S. 2008. Learning All Optimal Policies with Multiple Criteria. In *Proc. 25th International Conference on Machine Learning (ICML)*, 41–47.

Baumgartner, P.; Thiébaux, S.; and Trevizan, F. 2018. Heuristic Search Planning With Multi-Objective Probabilistic LTL Constraints. In *Proc. of 16th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*.

Bertsekas, D. P.; and Tsitsiklis, J. N. 1991. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research*, 16(3): 580–595.

Bertsekas, D. P.; and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*, volume 3 of *Optimization and neural computation series*. Athena scientific.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artif. Intell.*, 129(1-2): 5–33.

Bonet, B.; and Geffner, H. 2003. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *Proc. 13th International Conference on Automated Planning and Scheduling (ICAPS)*, 12–21.

Bryce, D.; and Buffet, O. 2008. 6th Int. Planning Competition: Uncertainty Track. *3rd Int. Probabilistic Planning Competition*.

Bryce, D.; Cushing, W.; and Kambhampati, S. 2007. Probabilistic Planning is Multi-Objective. Technical Report 07-006, ASU CSE.

Dai, P.; Mausam; Weld, D. S.; and Goldsmith, J. 2014. Topological Value Iteration Algorithms. *J. Artif. Intell. Res.*, 42: 181–209.

Geißer, F.; Haslum, P.; Thiébaux, S.; and Trevizan, F. 2022. Admissible Heuristics for Multi-Objective Planning. In *Proc. 32nd International Conference on Automated Planning and Scheduling (ICAPS)*, 100–109.

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

Jimenez, S.; Coles, A.; and Smith, A. 2006. Planning in Probabilistic Domains using a Deterministic Numeric Planner. In *Proc. 25th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG)*, 74–79.

Keller, T.; and Helmert, M. 2013. Trial-Based Heuristic Tree Search for Finite Horizon MDPs. In *Proc. 23rd International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI.

Khouadjia, M.; Schoenauer, M.; Vidal, V.; Dréo, J.; and Savéant, P. 2013. Pareto-Based Multiobjective AI Planning. In *Proc. 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2321–2327.

Klößner, T.; and Hoffmann, J. 2021. Pattern Databases for Stochastic Shortest Path Problems. In Ma, H.; and Serina, I., eds., *Proc. 14th International Symposium on Combinatorial Search (SOCS)*, 131–135.

Little, I.; and Thiébaux, S. 2007. Probabilistic Planning vs Replanning. In *Proc. ICAPS Workshop International Planning Competition: Past, Present and Future*.

Mandow, L.; and Pérez-de-la-Cruz, J. 2010. Multiobjective A* search with consistent heuristics. *J. ACM*, 57(5): 27:1–27:25.

Painter, M.; Lacerda, B.; and Hawes, N. 2020. Convex Hull Monte-Carlo Tree-Search. In *Proc. 30th International Conference on Automated Planning and Scheduling (ICAPS)*, 217–225.

Roijers, D. M.; and Whiteson, S. 2017. *Multi-Objective Decision Making*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Trevizan, F.; Teichteil-Königsbuch, F.; and Thiébaux, S. 2017. Efficient Solutions for Stochastic Shortest Path Problems with Dead Ends. In *Proc. 33rd International Conference on Uncertainty in Artificial Intelligence (UAI)*.

Trevizan, F.; Thiébaux, S.; Santana, P.; and Williams, B. 2017. I-dual: Solving Constrained SSPs via Heuristic Search in the Dual Space. In *Proc. 26th International Joint Conference on Artificial Intelligence (IJCAI)*.

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

Ulloa, C. H.; Yeoh, W.; Baier, J. A.; Zhang, H.; Suazo, L.; and Koenig, S. 2020. A Simple and Fast Bi-Objective Search Algorithm. In *Proc. 30th International Conference on Automated Planning and Scheduling (ICAPS)*, 143–151.

White, D. J. 1982. Multi-objective infinite-horizon discounted Markov decision processes. *Journal of Mathematical Analysis and Applications*, 89(2): 639–647.

Wiering, M. A.; and de Jong, E. D. 2007. Computing Optimal Stationary Policies for Multi-Objective Markov Decision Processes. In *Proc. 1st IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 158–165.

Wingate, D.; and Seppi, K. D. 2005. Prioritization Methods for Accelerating MDP Solvers. *J. Mach. Learn. Res.*, 6: 851–881.

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