Heuristic Search for SSPs with Lexicographic Preferences over Multiple Costs

Shuwa Miura[1], **Kyle Hollins Wray**[2], **Shlomo Zilberstein**[1]

[1] Manning College of Information and Computer Sciences, University of Massachusetts Amherst
[2] Alliance Innovation Lab Silicon Valley
smiura@cs.umass.edu, kyle.wray@nissan-usa.com, shlomo@cs.umass.edu

Abstract

Real-world decision problems often involve multiple competing objectives. The Stochastic Shortest Path (SSP) with lexicographic preferences over multiple costs offers an expressive formulation for many practical problems. However, the existing solution methods either lack optimality guarantees or require costly computations over the entire state space. We propose the first heuristic search algorithm for this problem, based on the heuristic algorithm for Constrained SSPs. Our experiments show that our heuristic search algorithm can compute optimal policies while avoiding a large portion of the state space. We also analyze the theoretical properties of the problem, establishing the conditions under which SSPs with lexicographic preferences have a proper optimal policy.

Introduction

Many real-world stochastic planning problems inherently involve multiple competing objectives. For example, in building management, it is necessary to consider both power usage and personal comfort (Kwak et al. 2012). In hybrid engine activation planning, it is necessary to minimize energy expenditure while reducing engine mode transitions (Wray, Lui, and Pedersen 2021). However, there seldom exists a single policy that optimizes all the objectives and striking a desired balance between them is an active research challenge.

The field of multi-objective decision making offers several fruitful approaches to formalize and solve decision problems with multiple objectives. Which approach is suited best for a particular problem depends on the available prior knowledge about the problem. We briefly review the existing approaches in the related work section at the end. In short, previous approaches are based on combining all the objectives using *linear scalarization*, computing the *Pareto front* of the policy space (Roijers and Whiteson 2017), optimizing the primary objective under constraints on the secondary objectives (Altman 1999), or solving problems that involve *lexicographic preferences* (Mouaddib 2004). We focus in this paper on improving the latter approach.

The latter approach is exemplified by MDPs with lexicographic preferences (LMDPs) (Mouaddib 2004), where the objectives are strictly ordered according to their importance. LMDPs provide an intuitive formulation for problems where there is a clear ordering among objectives. Wray, Zilberstein, and Mouaddib (2015) extended LMDPs with slack parameters, which allow small deviations from optimal values. The use of slack enables us to ignore small differences in one objective in order to improve lower priority objectives. For example, in hybrid engine optimization, we can specify some small amount of energy we are willing to compromise in order to reduce the number of mode transitions.

Despite the simplicity of formulating multi-objective problems using LMDPs, the existing solution methods for LMDPs lack scalability and optimality guarantees. Pineda, Wray, and Zilberstein (2015) proposed solving LMDPs as a series of Constrained MDPs (CMDPs) (Altman 1999) using a naïve linear programming formulation. However, solving MDPs using linear programming requires computations over the entire state space. Another approach called *local action restriction* (LAR) allocates local values of slack to each state to restrict available actions to those satisfying local slack conditions (Wray, Zilberstein, and Mouaddib 2015). However, methods based on LAR overconstrain the available actions and therefore cannot guarantee optimality.

Given these challenges, we propose the first heuristic search algorithm for stochastic planning problems with lexicographic preferences over multiple costs. To that end, we use a stochastic shortest path formulation of the problem called L-SSP (Wray, Lui, and Pedersen 2021). To justify the use of L-SSPs, we establish the conditions under which the existence of an optimal proper policy is guaranteed (Theorem 1). Finally, we show that optimal policies are not necessarily deterministic (Remark 1), requiring optimal algorithms to consider stochastic policies.

Our proposed algorithm solves L-SSPs as a series of Constrained SSPs (C-SSPs). Instead of using a naïve linear programming formulation, however, our algorithm uses heuristic search for C-SSPs (Trevizan et al. 2016) to solve those C-SSPs, which avoids processing the entire state space. Furthermore, we propose a technique that exploits the fact that the series of C-SSPs are closely related (i.e., they share the initial state, state space and action space). We demonstrate that optimal policies for L-SSPs achieve better trade-offs among multiple objectives, and experimentally show that using heuristic search can avoid visiting irrelevant states and can find an optimal policy faster than the naïve linear programming formulation.

Background

SSPs

A *stochastic shortest path problem* (SSP) is a tuple $\langle S, A, T, C, s_0, G\rangle$ where: S is a set of states. A is a set of actions. $T : S\times A\times S \rightarrow [0, 1]$ is a transition function such that $T(s_t, a_t, s_{t+1}) = Pr(s_{t+1}|s_t, a_t)$. $C(s_t, a_t) : S \times A \rightarrow \mathbb{R}$ is the cost of performing a_t at s_t. s_0 is the initial state. $G \subset s$ is a set of goal states. We assume that goal states are absorbing and transitions out of goal states have zero costs. In this paper, we only consider finite sets of states and actions.

A solution of an SSP is a *policy*. A *deterministic policy* π maps a state s to an action $a \in A$. A *stochastic policy* π maps a state s to a probability distribution on A. A policy π induces a value function $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} C(S_t, A_t)|S_0 = s_0, \pi]$, which represents the expected cost of reaching a goal state from s by following π. An *optimal policy* π^* is a policy that minimizes $V^\pi(s_0)$. We restrict our attention to problems in which there exists at least one *proper policy*, which reaches the goal from all states with probability 1, and any improper policies incur infinite costs. Under this assumption, an SSP is guaranteed to have an optimal policy that is proper (Bertsekas and Tsitsiklis 1991).

Linear Programming for SSPs

An optimal policy for an SSP can be computed in polynomial time using linear programming. Here, we introduce the linear program in the dual form (d'Epenoux 1963) presented in (LP1) where: $x_{s,a}$ are the optimization variables known as *occupation measures*, representing the expected number of times action $a \in A$ is taken in $s \in S$; $in(s) = \sum_{s'\in S, a\in A} x_{s',a}P(s|s', a)$; and $out(s) = \sum_{a\in A} x_{s,a}$.

$$\min_x \sum_{s\in S, a\in A} x_{s,a}C(s, a) \qquad \text{s.t. (C1)-(C4)} \quad \text{(LP1)}$$

$$x_{s,a} \geq 0 \qquad \forall s \in S, a \in A \quad \text{(C1)}$$

$$out(s) - in(s) = 0 \qquad \forall s \in S \setminus (G \cup \{s_0\}) \quad \text{(C2)}$$

$$out(s_0) - in(s_0) = 1 \quad \text{(C3)}$$

$$\sum_{s\in G} in(s) = 1 \quad \text{(C4)}$$

An optimal policy π^* can be obtained from the optimal solution to LP1 ($\pi^*(s, a) = x^*_{s,a}/out(s)$). However, LP1 can be intractable for an SSP with a large number of states.

Heuristic search algorithms utilize heuristic functions to guide their search to only explore the relevant portion of the state space. For SSPs in particular, heuristic search algorithms such as LAO* (Hansen and Zilberstein 2001) can find an optimal policy without expanding the entire state space.

SSPs with Lexicographic Preferences

A *lexicographic stochastic shortest path* (L-SSP) problem (Wray, Lui, and Pedersen 2021) is a variant of SSP with lexicographic preferences over multiple costs. Formally, an L-SSP is a tuple $\mathcal{L} = \langle S, A, T, \overrightarrow{C}, s_0, G, \overrightarrow{\delta}\rangle$ where: S, A, T, s_0, and G are defined as in SSP. $\overrightarrow{C} : S \times A \rightarrow \mathbb{R}^k$ is a vector-valued cost function such that $\overrightarrow{C}(s, a) = [C_1(s, a), \cdots, C_k(s, a)]$, where each $C_i(s, a)$ denotes the cost for the i-th objective. $\overrightarrow{\delta} = [\delta_1, \cdots, \delta_{k-1}] \geq \overrightarrow{0}$ is slack parameters. As with a regular SSP, a policy π induces a value function $\overrightarrow{V}^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \overrightarrow{C}(S_t, A_t)|S_0 = s_0, \pi]$.

An optimal policy π for L-SSP with k objectives is:

$$\begin{aligned}&\arg\min_\pi V^\pi_k(s_0)\\ &\text{s.t. } V^\pi_i(s_0) - V^*_i(s_0) \leq \delta_i \qquad \forall i < k\end{aligned}$$

where $V^*_i(s_0)$ is the value of an optimal policy defined recursively. $V^*_1(s_0)$ coincides with the optimal value of C_1 for the unconstrained SSP. $V^*_2(s_0)$ is the optimal value of C_2 with the constraint $V^\pi_1(s_0) - V^*_1(s_0) \leq \delta_1$. Similarly, $V^*_3(s_0)$ is the optimal value of C_3 with the constraints $V^\pi_1(s_0) - V^*_1(s_0) \leq \delta_1$ and $V^\pi_2(s_0) - V^*_2(s_0) \leq \delta_2$.

Wray, Zilberstein, and Mouaddib (2015) proposed *local action restriction* (LAR) to approximately solve LMDPs (MDP counter-part of L-SSPs). LAR restricts the available actions via local slack $\overrightarrow{\eta} = [\eta_1, \cdots, \eta_{k-1}]^T$ at s by $A_{i+1}(s) = \{a \in A_i(s)|V^*_i(s) - Q^*_i(s, a) \leq \eta_i\}$ with $A_1 = A$. LVI is a value iteration algorithm based on LAR. However, solutions computed by LAR are not guaranteed to be optimal and can be arbitrarily worse than an optimal policy (Pineda, Wray, and Zilberstein 2015).

Constrained SSPs

Another closely related variant of multi-objective SSP is *constrained stochastic shortest path problem* (C-SSP). Formally, a C-SSP is a tuple $\mathcal{C} = \langle S, A, T, \overrightarrow{C}, s_0, G, \overrightarrow{c}, p\rangle$ where $\overrightarrow{c} = [\hat{c_1}, \cdots, \hat{c_k}]$ is the cost upper-bound vector and p is the primary objective to optimize. An optimal policy for a C-SSP is defined as:

$$\begin{aligned}&\arg\min_\pi V^\pi_p(s_0)\\ &\text{s.t. } V^\pi_i(s_0) \leq \hat{c_i} \qquad \forall i \neq p\end{aligned}$$

The linear program corresponding to a C-SSP can be obtained by adding (C5) to (LP1) and replacing C in the objective function with C_p.

$$\sum_{s\in S, a\in A(s)} x_{s,a}C_i(s, a) \leq \hat{c_i} \qquad \forall i \neq p \quad \text{(C5)}$$

We refer to this linear program LP1$'$.

Heuristic Search for Constrained SSPs

I-dual (Trevizan et al. 2016, 2017) is a heuristic search algorithm for C-SSPs. Similar to heuristic algorithms for unconstrained SSPs such as LAO* (Hansen and Zilberstein 2001), I-dual incrementally expands nodes on the current best solution graph. I-dual can find an optimal policy without evaluating the entire state space with the help of heuristic functions $\overrightarrow{H} = [H_1, \cdots, H_k]$, where H_i is a heuristic function for C_i.

Unlike LAO*, the current best solution for C-SSP needs to respect the constraints, thus I-dual repeatedly solves LP2 for partial problems $\langle \hat{S}, s_0, \hat{G}, T, \overrightarrow{C}, \overrightarrow{c}, \overrightarrow{H}, p\rangle$, where $\hat{S} \subset S$, $G \cap \hat{S} \subset \hat{G}$, $\hat{A} \subset A$, and $\overrightarrow{H}$ is the heuristic functions. LP2 computes the current best solution for the states generated so far. It treats fringe states (states not yet expanded) as artificial terminal states, and use heuristic values to derive cost estimates toward the terminal state. After computing the current best solution, the algorithm expands all the fringe states with some incoming flows.

$$\min_x \sum_{s\in\hat{S},a\in\hat{A}} x_{s,a}C_p(s,a) + \sum_{s\in\hat{G}} H_p(s)$$

s.t. (C1)-(C4), (C6) (LP2)

$$\sum_{x\in\hat{S},a\in\hat{A}} x_{s,a}C_i(s,a) + \sum_{s\in\hat{G}} in(s)H_i(s) \le \hat{c}_i \quad \forall i \neq p \tag{C6}$$

When $\overrightarrow{H}$ is admissible, I-dual returns an optimal policy (Trevizan et al. 2016). From an Operations Research point of view, I-dual can be thought of as a column and row generation algorithm (Trevizan et al. 2016, 2017). At each iteration of the algorithm, new variables $x_{s,a}$ (columns) and new flow constraints (rows) are added to LP2. The algorithm decides which variables (columns) to add based on the current best solution, which in turn is directed by the heuristic function.

Solving L-SSPs as a Series of Constrained SSPs

Existing methods for L-SSPs optimally transform an L-SSP to a series of C-SSPs (Pineda, Wray, and Zilberstein 2015)[1].

Algorithm 1: CM-Map (Pineda, Wray, and Zilberstein 2015)

```
Input: $\mathcal{L} = \langle S, A, T, \overrightarrow{C}, s_0, G, \overrightarrow{\delta} \rangle$
  $\overrightarrow{\hat{c}} \leftarrow []$
  for $i = 1, \cdots, k$ do
      Create C-SSP $\mathcal{C}_i \leftarrow \langle S, A, T, \overrightarrow{C}, s_0, G, \overrightarrow{\hat{c}}, i \rangle$
      $\pi^* \leftarrow$ Solve $\mathcal{C}_i$
      $\hat{c}_i \leftarrow V_i^*(s_0) + \delta_i$
  end for
  return $\pi^*$
```

Algorithm 1 illustrates the procedure (CM-Map) to transform an L-SSP to a series of C-SSPs. For each $i = 1, \cdots, k$ objectives, CM-Map creates a C-SSP $\mathcal{C}_i$ such that i is the primary objective ($p = i$) and $\hat{c}_j = V_j^*(s_0) + \delta_j$ for all $j < i$. Note that $\mathcal{C}_1$ is an unconstrained SSP optimizing C_1. $\mathcal{C}_2$ is a C-SSP optimizing C_2 with $\hat{c}_1 = V_1^*(s_0) + \delta_1$ and so on. When each $\mathcal{C}_i$ is solved optimally, CM-Map returns an optimal policy. Pineda, Wray, and Zilberstein (2015) used a naïve linear programming formulation (LP1′) to solve a series of C-SSPs. We denote the version of CM-Map that uses LP1′ to solve C-SSPs as CM-Map (LP).

Theoretical Analysis

The theoretical properties of L-SSPs (Wray, Lui, and Pedersen 2021) have not yet been analyzed. Hence, we begin with a theoretical analysis of some key properties of L-SSPs.

[1]The algorithm was originally proposed for LMDPs, but can be easily adapted for L-SSPs.

Existence of a Proper Optimal Policy

We first show the existence of a proper optimal policy for L-SSPs under the assumptions similar to SSPs. Indeed, for general L-SSPs, the existence of a proper optimal policy is not guaranteed (e.g., environment with zero-cost loops in terms of the primary objective (C_1)).

We show that L-SSPs have a proper optimal policy under the following assumptions:

Assumption 1. *There exists at least one proper policy.*

Assumption 2. *For the primary objective, all improper policies yield infinite cost from the initial state.*

Note that we only require Assumption 2 for the primary objective but not for the secondary objectives ($C_2, \cdots, C_k$). That is, we can have zero cost loops for the secondary objectives. In fact, allowing zero (negative) cost loops can be convenient for specifying real-world problems, where a secondary objective can be zero for many of the states. For example, when a secondary objective penalizes the agent for hitting some object, the agent may not incur that cost in the majority of the states.

Theorem 1. *Under Assumptions 1 and 2, an L-SSP has a proper optimal policy.*

Proof. We prove the theorem by induction on the number of objectives.

Base Case: when $k = 1$, the problem is a regular SSP. The existence of a proper optimal policy has been proven by Bertsekas and Tsitsiklis (1991) for this case.

Induction Step: we assume the claim is true for L-SSPs with k objectives. Given an L-SSP with $k+1$ objectives $\mathcal{L}_{k+1}$, let $\mathcal{L}_k$ be the corresponding L-SSP using the first k objectives. Then by the induction hypothesis, $\mathcal{L}_k$ has a proper optimal policy π_k^*. By definition, we have $V_j^*(s_0) - V_j^{\pi_k^*}(s_0) \le \delta_j$ for all $j < k$. Moreover, trivially $V_k^*(s_0) - V_k^{\pi_k^*}(s_0) = 0 \le \delta_k$ must hold. Therefore, by the definition of L-SSPs, π_k^* is a feasible policy for $\mathcal{L}_{k+1}$ as well. Consider LP1′ for $\mathcal{L}_{k+1}$. The linear program is feasible as the occupation measure induced by π_k^* is a feasible solution, satisfying all the flow constraints and cost constraints. Furthermore, the objective function is bounded from below. Therefore, LP1′ for $\mathcal{L}_{k+1}$ has an optimal solution. Let π_{k+1}^* be a policy induced by an optimal solution (occupation measure) from LP1′, then π_{k+1}^* is an optimal policy for $\mathcal{L}_{k+1}$.

We now argue that π_{k+1}^* must be proper. Note that π_{k+1}^* needs to have bounded cost in terms of the primary objective due to the cost constraint (C5). Therefore, π_{k+1}^* must be proper as otherwise, π_{k+1}^* would be an improper policy with bounded cost, which violates Assumption 2. □

We argue that the existence of a proper optimal policy is an advantage of L-SSPs over C-SSPs, where having a constraint that is too tight can make the problem infeasible.

Need for Stochastic Policies

We next point out an important characteristic of optimal policies for L-SSP, which justifies our approach to solving the problem.

Remark 1. *For L-SSPs, stochastic policies dominate deterministic policies.*

Example: Consider the L-SSP in Figure 1. The optimal policy for the primary objective is to go above to the goal with $V_1^*(s_0) = 0$. With $\delta_1 = 0.3$, the optimal stochastic policy takes the path below with the probability 0.3 and the path above with the probability 0.7, which results in $V_2^*(s_0) = 0.7$. On the other hand, the only feasible deterministic policy (π) is to take the path above, which results in $V_2^\pi(s_0) = 1.0$.

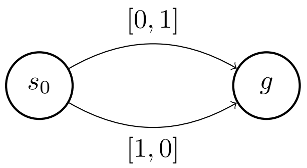


Figure 1: Sample L-SSP with a stochastic optimal policy

The result is in contrast to the existence of deterministic optimal policies for regular SSPs (Bertsekas and Tsitsiklis 1991), and essentially identical to the same result for CMDPs/C-SSPs (Altman 1999). Due to the need for randomization, previous methods that look for deterministic policies such as LVI or linear scalarization inherently cannot find optimal policies.

Furthermore, computing optimal deterministic policies for L-MDPs/L-SSPs is shown to be NP-hard (Pineda, Wray, and Zilberstein 2015), using the corresponding result for C-MDPs/C-SSPs (Feinberg 2000). Note, however, that computing deterministic optimal policies can still be useful when the policy needs to be interpretable.

Heuristic Search for Lexicographic SSPs

In this section, we propose a new heuristic search algorithm for L-SSPs. The basic idea is very simple; instead of solving a series of C-SSPs with LP1$'$, we use heuristic search to solve a series of C-SSPs. In particular, when a vanilla I-dual is used to solve C-SSPs, we denote the algorithm as CM-Map (I-dual).

From the optimality of CM-Map and I-dual, we immediately get the following result:

Remark 2. *When $\overrightarrow{H}$ is admissible, CM-Map (I-dual) finds an optimal policy for given L-SSPs.*

However, we observe that the C-SSPs being solved have some similarities. Hence, we propose a technique to improve CM-Map (I-dual) based on this property.

Using Unconstrained Problems as Heuristics

While CM-Map (I-dual) avoids expanding the entire state space, solving the linear programs (LP2) can still be expensive when the heuristic function is not informative. To remedy this issue we can reuse information obtained from solving $\mathcal{C}_1$ to solve $\mathcal{C}_2, \cdots, \mathcal{C}_k$. In particular, we propose to use lower bound value estimates $\underline{V}_i$ for an unconstrained problem ($\mathcal{U}_i$) as a heuristic function for $\mathcal{C}_i$ as any lower bound value estimate for $\mathcal{U}_i$ is trivially also a lower bound for the optimal value of $\mathcal{C}_i$.

Algorithm 2: CM-Map (LRTDP or LAO*, I-dual)

```
Input: $\mathcal{L} = \langle S, A, T, \overrightarrow{C}, s_0, G, \overrightarrow{\delta} \rangle$, $\overrightarrow{H}_{\mathcal{U}}$, $\overrightarrow{H}_{\mathcal{C}}$.
1: $\overrightarrow{\underline{V}} \leftarrow$ Solve $\mathcal{C}_1$ with LRTDP or LAO$^*$ using $\overrightarrow{H}_{\mathcal{U}}$
2: $\overrightarrow{H}' = \max(\overrightarrow{\underline{V}}, \overrightarrow{H}_{\mathcal{C}})$
3: $\overrightarrow{c} \leftarrow [V_1^* + \delta_1]$
4: for $i$ in $2, \cdots, k$ do
5:     Create C-SSP $\mathcal{C}_i \leftarrow \langle S, A, T, \overrightarrow{C}, s_0, G, \overrightarrow{c}, i \rangle$
6:     $\pi^* \leftarrow$ Solve $\mathcal{C}_i$ with I-dual using $\overrightarrow{H}'$
7:     $\hat{c}_i \leftarrow V_i^* + \delta_i$
8: end for
9: seturn $\pi^*$
```

To compute $\underline{V}_i$, we use standard SSP solvers that keep lower bounds on values, such as LRTDP (Bonet and Geffner 2003) or LAO* (Hansen and Zilberstein 2001), to update the heuristic values. We denote these variants CM-Map (LRTDP, I-dual) and CM-Map (LAO*, I-dual), respectively.

As shown in Algorithm 2, the algorithm takes as input L-SSP ($\mathcal{L}$), heuristic function for unconstrained SSPs ($\overrightarrow{H}_{\mathcal{U}}$), and heuristic function for corresponding constrained SSPs ($\overrightarrow{H}_{\mathcal{C}}$) if available. LRTDP or LAO* are first used to solve $\mathcal{C}_1 = \mathcal{U}_1$. We make a slight modification to LRTDP or LAO* so that whenever LRTDP or LAO* update their current value estimates with regard to C_1, they update the value estimates for the other objectives as well. This gives us lower bound value estimates for each objective $\overrightarrow{\underline{V}}$ to be used later (see further discussion below). We combine $\overrightarrow{\underline{V}}$ with $\overrightarrow{H_{\mathcal{C}}}$ to get a new heuristic function $\overrightarrow{H'}$. For $C_2, \cdots, C_k$, we solve C-SSPs with I-dual using $\overrightarrow{H'}$.

Note that we did not change LRTDP or LAO* except where they update their value estimates. For example, in both algorithms, greedy actions are selected based on C_1, and termination conditions are checked only for C_1. Therefore, our slight modification preserves theoretical properties of LRTDP or LAO* such as guaranteed termination in finite steps in case of LRTDP. As a consequence of selecting actions only based on C_1, however, $\underline{V}_i$ does not necessarily converge to the optimal values of unconstrained problems for $i = 2, \cdots, k$. Note also that $\overrightarrow{\underline{V}}$ does not necessarily correspond to $\overrightarrow{V}^{\pi^*}$, where π^* is an optimal policy for $\mathcal{U}_1$. Rather, $\overrightarrow{\underline{V}}$ is a lower bound value estimates for each objective. As LRTDP and LAO* can visit different set of states, $\overrightarrow{\underline{V}}$ can be different for the two algorithms. This incurs little overhead because the first iteration of CM-Map requires solving an unconstrained SSP ($\mathcal{C}_1 = \mathcal{U}_1$).

For LRTDP or LAO* to return lower bound value estimates, we need the following assumption in addition to Assumption 1 and 2:

Assumption 3. *All costs are non-negative.*

which is a standard assumption in AI planning setting. In Theorem 1, we considered a setting where costs can be negative for the sake of generality.

For CM-Map (LRTDP, I-dual) and CM-Map (LAO*, I-dual) to compute optimal policies, LRTDP and LAO* must return optimal values for C_1. However, both algorithms return ϵ-optimal policies. As constraints on C_1 are based on V_1^*, policies returned by CM-Map (LRTDP, I-dual) and CM-Map (LAO*, I-dual) can fail to utilize slack values by ϵ. In practice, and in all of our experiments, CM-Map (LRTDP, I-dual) and CM-Map (LAO*, I-dual) returned the same values as CM-Map (LP) by using small enough value of ϵ.

Experiments

In the first set of experiments, we compare qualities of optimal policies for L-SSPs against suboptimal policies computed using LVI. In the second set of experiments, we compare planning time among algorithms that compute optimal policies for L-SSPs.

Domains

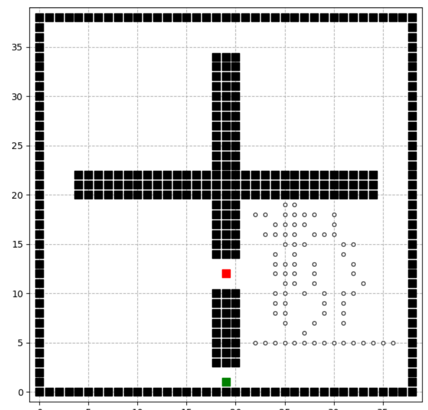
We experimented with the following three problem domains.

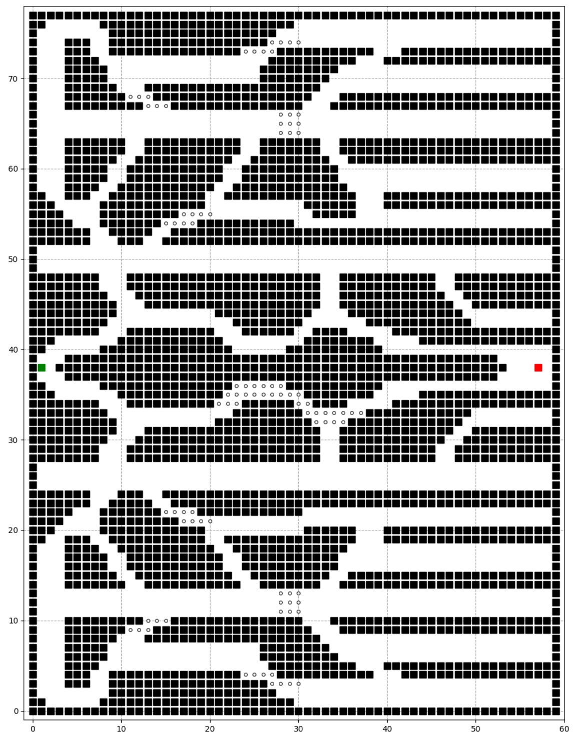
RaceTrack This problem is a multi-objective variant of the RaceTrack problem introduced by Pineda, Wray, and Zilberstein (2015). The racetrack problem is a simulation of a racing car, which starts from the starting line and ends at the finishing line. The state of the car (agent) consists of the current position as well as its velocity. At each state, the agent can change its velocity in each dimension by at most 1, resulting in 9 possible actions in total. Figure 2 shows the problem instances used in the experiments.

The multi-objective version of the problem has three cost functions (C_1, C_2, and C_3). C_1 is the number of time-steps taken to reach the finish line. C_2 penalizes changes in the car's velocity. C_3 penalizes entering unsafe locations.

SearchRescue (Trevizan et al. 2016) This environment consists of an $n \times n$ grid. The goal is to find a single survivor, bring her on board the vehicle, and transport her to safety. There is a known location at Hamming distance d from the starting point that has a survivor. The environment has $r\%$ of locations where the presence of survivors are initially unknown. These locations initially have low (5%), middle (10%), or high (20%) probability of having a survivor. The primary objective C_1 is to minimize the time to rescue a survivor. The time needed to move among grid locations depends on the speed of the vehicle as well as the load of the vehicle. The secondary objective C_2 is to minimize fuel consumption. The problem was originally defined as a C-SSP, where the constraint on C_2 is half of the fuel the vehicle would spend if it optimized only C_1. We reformulated the problem as an L-SSP. As in the original paper, we experimented with random instances.

Route and Engine Activation Planning This problem is a variation of the Engine Activation Planning problem for series hybrid electric vehicles (HEVs) (Wray, Lui, and Pedersen 2021). Series HEVs use gas powered engines to power a compact battery that an electric motor uses to control the wheels. The problem then is to come up with a route plan with low expected travel time (C_1) in minutes as well as a gas engine activation plan to reduce different costs such as energy expenditure (C_2) and engine mode transitions (C_3). The states are the current position and the battery level of the vehicle. We use the same model of the engine as in the original paper. Due to the data availability, we used synthetic data based on a map in Figure 3. The vehicle makes route and engine activation decisions at intersections.


(a) blank


(b) sym

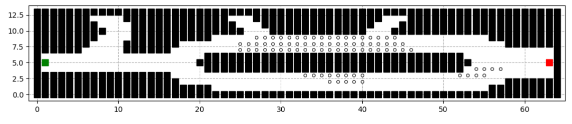
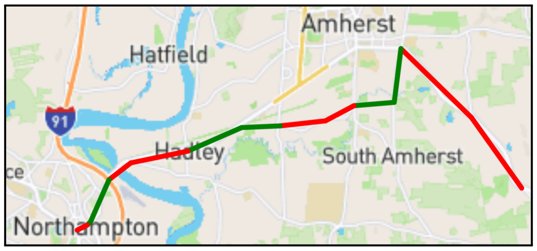
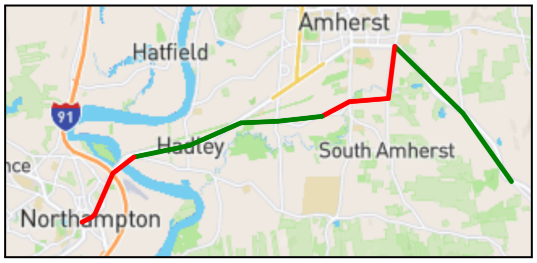

(c) track1

Figure 2: RaceTrack Problems. The green and red squares represent the start and goal locations, respectively. Locations with grey dots are unsafe locations for the car.



(a) Minimizing energy expenditure only.



(b) Optimizing the lexicographic objective.

Figure 3: Instance (Amherst-Northampton) of Route and Engine Activation. Red/green lines indicate engine is on/off.

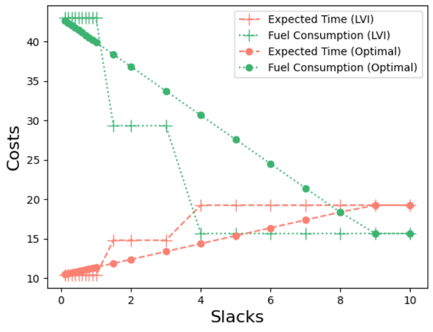
Comparing Solution Qualities against LVI

We first compare qualities of optimal policies for L-SSPs and deterministic policies computed by LVI. Recall that LVI restricts local deviation from optimal value by $\overrightarrow{\eta}$. Note that optimal policies for L-SSPs allow some slack values, thus values for $C_1, \cdots, C_{k-1}$ are not unique for optimal policies. We used CM-Map (LRTDP, I-dual) to compute the values in the plots; other CM-Map variants returned similar values.

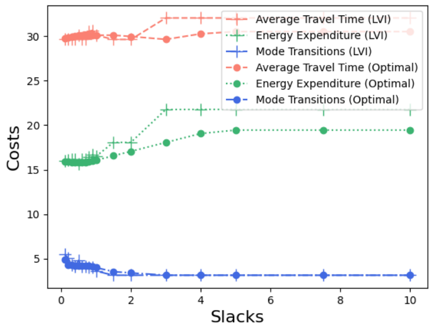
Figure 4a shows values computed with different values of δ_1 and η_1 for an instance of SearchRescue. Deterministic policies computed by LVI resulted in sudden jumps in objectives as we changed η_1. On the other hand, methods that compute optimal stochastic policies for L-SSPs resulted in a more intricate balance between C_1 and C_2. Note that with $\delta_1 = \eta_1$, having a global slack δ_1 is a stronger constraint. With large enough values of δ_1 and η_1, both algorithms return an optimal policy for C_2.

Figure 5 shows values computed with different values of $\overrightarrow{\delta}$ and $\overrightarrow{\eta}$ for instances of RaceTrack. While slacks are two-dimensional in RaceTrack problems, we used vectors with the same values. In Figure 5a, LVI failed to sacrifice C_1 in order to improve the other objectives. But the resulting values for C_2 and C_3 were not significantly different.

Figure 5b and 5c show that while both LVI and optimal policies had similar values for each objective, the output of LVI tended to fluctuate more. Note that it is possible for values to fluctuate: as we increase the slack for C_1, we can have a better policy in terms of C_2, which in turn makes the constraint when optimizing C_3 tighter. On the other hand, the values for optimal policies change more smoothly. When the constraints on costs (C5) are tight, increasing δ often meant moving the solution in the same direction. We observed a similar trend for an instance of Route and Engine Activation Planning (Figure 4b).



(a) SearchRescue with $n = 4$, $d = 3$, and $r = 0.5$.



(b) Instance (Amherst-Northampton) of Route and Engine Activation Planning.

Figure 4: Values corresponding different $\overrightarrow{\delta}$ and $\overrightarrow{\eta}$

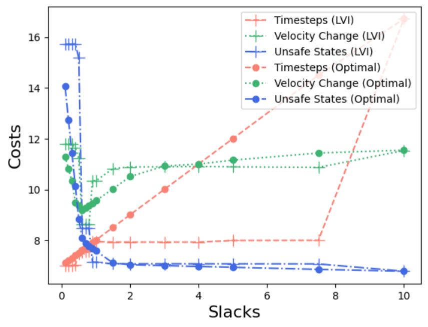
Comparing Planning Times

We next compare planning times to compute optimal policies among: a naïve linear programming encoding (CM-Map (LP)), CM-Map (I-dual), CM-Map (LRTDP, I-dual), and CM-Map (LAO*, I-dual). We refer to these algorithms LP, I-dual, LRTDP, and LAO*, respectively in tables.

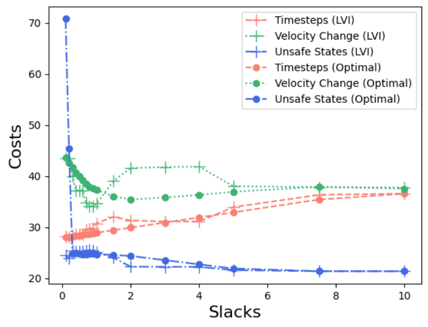
For the heuristic functions, we used h_{min} heuristic, computed as needed by LRTA* (Bonet and Geffner 2003). Note that for problems that admit factored state representation such as PPDDL (Younes and Littman 2004), other heuristics such as h_{max} that exploit the representation exist. Most notably, Trevizan, Thiébaux, and Haslum (2017) proposed heuristics for (C-)SSPs represented in probabilistic SAS$^+$. We used h_{min} as some of the domains used in the experiments do not have efficient factored state representation.

To solve linear programs, we used the Gurobi linear programming solver v9.5 (Gurobi Optimization, LLC 2022), using a single thread. All the algorithms were implemented by us and solved on Intel Xeon E2-2680v4 computers. All problem instances were solved with 60 minutes CPU-time and 2GB memory limit. For LRTDP and LAO* we used $\epsilon = 10^{-2}$ when checking convergence.

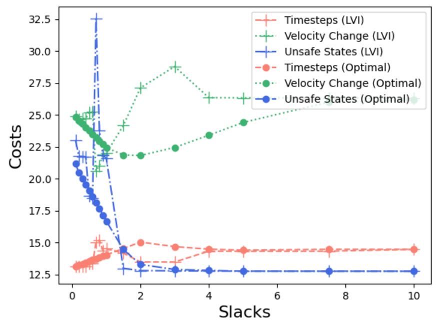
Table 1 shows the results for RaceTrack problems. For blank, all the heuristic search variants solved the problem much faster than CM-Map (LP), and expanded less than a tenth of the entire state space. Among heuristic search variants, CM-Map (LRTDP, I-dual) was slower as it took more time for LRTDP to solve $\mathcal{C}_1$.



(a) blank



(b) sym



(c) track1

Figure 5: Values corresponding different $\overrightarrow{\delta}$ and $\overrightarrow{\eta}$ for RaceTrack Problems.

Instance	Algorithm	$\overrightarrow{\delta} = [0.1, 0.1]$		$\overrightarrow{\delta} = [1.0, 1.0]$		$\overrightarrow{\delta} = [5.0, 5.0]$	
		Time(s)	$\lvert S\rvert$	Time(s)	$\lvert S\rvert$	Time(s)	$\lvert S\rvert$
blank	LP	361.8	74636	812.6	74636	883.5	74636
	Idual	**22.6**	3766	30.0	4401	**18.5**	2632
	LRTDP	45.5	2096	51.4	4297	50.1	2526
	LAO	26.4	**1952**	**26.0**	**3429**	26.1	**1885**
sym	LP	57.9	61862	142.3	61862	246.8	61862
	Idual	378.7	9184	193.1	8830	144.6	8830
	LRTDP	78.8	3354	67.1	**3338**	74.3	5037
	LAO	**45.2**	**3161**	**47.7**	3803	**55.2**	**4336**
track1	LP	54.2	14270	84.7	14270	159.3	14270
	Idual	160.0	8268	187.0	8601	59.6	7337
	LRTDP	65.7	7397	**79.9**	7980	20.9	6374
	LAO	**53.0**	**7039**	81.1	**7818**	**17.7**	**6114**

Table 1: Results for RaceTrack Domains. $|S|$ represents the number of states generated.

For sym, CM-Map (I-dual) performed worst. As the heuristic was not informative enough, it had to generate about a sixth of the entire state space. On the other hand, CM-Map (LRTDP, I-dual) and CM-Map (LAO*, I-dual) generated less than a tenth of the state space, and solved problem faster than CM-Map (LP).

For track1, the heuristic variants were not significantly better than CM-Map (LP) except when $\delta = 5.0$. Due to the structure of the problem instance, heuristic search had to expand more than half of the entire space. As heuristic search needs to solve LP2 in each iteration of the algorithm, it has a large overhead compared to CM-Map (LP) when it cannot exclude many of the states.

Table 2 shows the results for SearchRescue problems. Overall, heuristic search performed much better than CM-Map (LP) in this domain. CM-Map (LP) could not solve any problem instances with $n = 5$. On the other hand, heuristic search was able to solve problems with millions of states as it only needs to explore a small portion of the entire state space. Note that as we increase r (the percentage of) locations where the presence of survivors are initially unknown, the number of possible states increases. The other three heuristic search variants solved mostly the same set of problems. Due to the more informed heuristic, CM-Map (LRTDP, I-dual) generally solved these problems slightly faster than the other two with fewer generated states.

Table 3 shows the results for Route and Engine Activation Planning. As in the previous two domains, heuristic search computed an optimal policy faster than CM-Map (LP) with fewer generated states. However, there was no significant difference among the heuristic search variants.

Related Work

As discussed earlier, LMDPs/L-SSPs can be thought of as a way to specify a series of CMDPs/C-SSPs. However, formulating problems as CMDPs/C-SSPs requires the designer to know the appropriate constraints a priori. If the constraints are too hard to satisfy, the problem would be infeasible. In contrast, LMDPs/L-SSPs require specifying ordering among objectives and slack values (allowable deviations from optimal costs). The implicit assumption is that for some problems, the deviations from the optimal costs are easier to specify than hard constraints on costs. We showed that L-SSPs always have a proper optimal policy whenever the corresponding SSPs have one (Theorem 1). While some problems are easier to specify with CMDPs/C-SSPs, problems with no clear a priori budget constraints can be easier to specify with LMDPs/L-SSPs.

When the exact trade-offs among different objectives are known a priori, we can linearly combine the multiple objectives into one (*linear scalarization*). The advantage of this approach is that we can make use of the existing solution methods for single objective MDPs. However, specifying the coefficients to strike the right balance among objectives is hard, and could result in unexpected behaviors of the agent. Furthermore, we showed (Remark 1) that optimal policies for L-SSPs are not necessarily deterministic, but most algorithms look for a single-objective produce deterministic policies.

When the desired trade-offs among the different objectives cannot be determined a priori, another approach to handle multiple objectives is to compute the Pareto front of different objectives (Roijers and Whiteson 2017). The Pareto front consists of all policies that are Pareto optimal, that is, policies for which no single objective can be improved without compromising another objective. Unlike the other approaches, Pareto-based approaches return a set of policies

n	d	r	Algorithm	$\overrightarrow{\delta} = [0.1]$			$\overrightarrow{\delta} = [1.0]$			$\overrightarrow{\delta} = [5.0]$		
				Time(s)	C	$\lvert S\rvert$	Time(s)	C	$\lvert S\rvert$	Time(s)	C	$\lvert S\rvert$
5	3	0.5	LP	NA	0	88550400.4	NA	0	88645803.5	NA	0	88611512.6
			Idual	283.2	26	3948.7	366.2	27	4441.8	665.1	27	6024.5
			LRTDP	**134.8**	**28**	**3312.2**	**224.0**	**28**	**3508.7**	**498.2**	**28**	**5601.7**
			LAO	248.1	26	3804.3	348.2	26	4227.7	622.4	**28**	6007.1
		0.75	LP	NA	0	NA	NA	0	NA	NA	0	NA
			Idual	111.1	**22**	3863.4	272.0	23	5339.1	324.6	22	5298.9
			LRTDP	**98.6**	**22**	**3328.5**	**259.4**	**24**	**4573.0**	**239.7**	**24**	**5063.0**
			LAO	104.2	**22**	3795.0	275.1	23	5315.0	327.0	22	5363.1
	4	0.5	LP	NA	0	88952184.4	NA	0	88966559.9	NA	0	88966559.9
			Idual	420.5	21	4496.7	560.1	22	5332.9	625.9	22	5987.0
			LRTDP	**273.9**	**22**	**3703.0**	**419.8**	**23**	**4351.2**	**544.1**	23	**5368.3**
			LAO	399.0	**22**	4300.0	494.2	22	5169.2	602.5	**24**	5967.2
		0.75	LP	NA	0	NA	NA	0	NA	NA	0	NA
			Idual	373.9	18	7023.4	760.9	20	10507.7	533.7	20	8553.0
			LRTDP	**278.1**	**21**	**5526.4**	**602.6**	**21**	**8697.2**	**449.0**	**21**	**7878.1**
			LAO	341.6	18	6594.7	697.6	20	10209.8	495.0	17	8442.1

Table 2: Results for SearchRescue Domains with 30 random instances. C represents the number of instances solved. $|S|$ is the average number of generated states. Time(s) is the average time. The averages are computed for instances that all heuristic search variants solved.

Instance	Algorithm	$\overrightarrow{\delta} = [1.0, 1.0]$		$\overrightarrow{\delta} = [1.0, 2.0]$	
		Time(s)	$\lvert S\rvert$	Time(s)	$\lvert S\rvert$
Amherst-Northampton	LP	195.4	34777	226.4	34777
	Idual	82.7	2641	68.0	2023
	LRTDP	**67.2**	2647	58.8	1982
	LAO	67.6	**2605**	**50.7**	**1925**
Northampton-Holyoke	LP	83.5	34945	115.0	34945
	Idual	49.0	2279	50.2	2265
	LRTDP	**37.9**	2273	**36.0**	**2260**
	LAO	41.4	**2241**	38.8	2295

Table 3: Results for Route and Engine Activation. $|S|$ is the number of generated states. Time(s) is the planning time.

and let users choose a policy from the Pareto front. This could be a benefit in some applications, but when the set of policies is very large, it may be hard to present.

Decision-making with lexicographic preferences offers a natural representation of objectives that has proved useful in several application domains, including safe and cost-efficient operations of smart buildings (Lesser and Abate 2018), planning the distribution of emergency goods to victims of disasters (Laguna-Salvadó et al. 2019), and mitigating undesirable side effects of deployed AI systems (Saisubramanian, Kamar, and Zilberstein 2020).

LMDPs/L-SSPs are special cases of Lexicographic Multi-Objective Linear Programming (LMOLP) (Isermann 1982). In this sense, our heuristic search algorithm can be viewed as a column and row generation algorithm for a special case of LMOLP with the guidance of the heuristic function. While some existing algorithms are tailored for LMOLP (Pourkarimi and Zarepisheh 2007; Cococcioni, Pappalardo, and Sergeyev 2018), they work in settings with strict lexicographic ordering and are not directly applicable to LMDPs/L-SSPs with slacks.

Conclusions

We propose the first heuristic search algorithm for stochastic planning problems with lexicographic preferences called L-SSPs. Our approach builds on the heuristic search algorithm for constrained SSPs. Unlike the naïve linear programming encoding of the problem in the literature, which expands the entire state space, the algorithm gradually generates states based on the current best solution. Unlike previous methods that are based on local action restriction, our algorithm computes an optimal policy. Furthermore, we propose a variant of the algorithm that reuses value estimates as heuristic functions for later iterations.

Our experiments show that by computing optimal policies using our heuristic search, we can strike a finer balance among multiple objectives compared to the existing state-of-the-art method. Moreover, when the number of relevant states is much smaller than the number of possible states, out algorithm can compute optimal policies much faster than previous methods.

We additionally show that L-SSPs are guaranteed to have a proper optimal policy under the same conditions as unconstrained SSPs (Theorem 1). This is in contrast to C-SSP, where setting arbitrary constraints can render the problem infeasible. The result suggests an advantage of L-SSP over C-SSP when there are no clear cost constraints. While a direct consequence of the same result for CMDP/C-SSP, we note that stochastic policies dominate deterministic policies in L-SSP (Remark 1). The result provides a further indication that prior methods based on local action restriction or linear scalarization may not find optimal policies.

Acknowledgments

This research was supported by the National Science Foundation grants number IIS-1724101 and IIS-1954782, and by the Alliance Innovation Lab Silicon Valley.

References

Altman, E. 1999. *Constrained Markov Decision Processes*, volume 7. CRC Press.

Bertsekas, D. P.; and Tsitsiklis, J. N. 1991. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3): 580–595.

Bonet, B.; and Geffner, H. 2003. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proceedings of the Thirteenth International Conference on International Conference on Automated Planning and Scheduling*, 12–21.

Cococcioni, M.; Pappalardo, M.; and Sergeyev, Y. D. 2018. Lexicographic multi-objective linear programming using grossone methodology: Theory and algorithm. *Applied Mathematics and Computation*, 318: 298–311.

d'Epenoux, F. 1963. A probabilistic production and inventory problem. *Management Science*, 10(1): 98–108.

Feinberg, E. A. 2000. Constrained discounted Markov decision processes and Hamiltonian cycles. *Mathematics of Operations Research*, 25(1): 130–140.

Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual.

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

Isermann, H. 1982. Linear lexicographic optimization. *OR Spectrum*, 4(4): 223–228.

Kwak, J.; Varakantham, P.; Maheswaran, R.; Tambe, M.; Jazizadeh, F.; Kavulya, G.; Klein, L.; Becerik-Gerber, B.; Hayes, T.; and Wood, W. 2012. SAVES: A sustainable multiagent application to conserve building energy considering occupants. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems*, 21–28.

Laguna-Salvadó, L.; Lauras, M.; Okongwu, U.; and Comes, T. 2019. A multicriteria Master Planning DSS for a sustainable humanitarian supply chain. *Annals of Operations Research*, 283(1-2): 1303–1343.

Lesser, K.; and Abate, A. 2018. Multiobjective optimal control with safety as a priority. *IEEE Transactions on Control Systems Technology*, 26(3): 1015–1027.

Mouaddib, A. 2004. Multi-objective decision-theoretic path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, 2814–2819.

Pineda, L. E.; Wray, K. H.; and Zilberstein, S. 2015. Revisiting multi-objective MDPs with relaxed lexicographic preferences. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*.

Pourkarimi, L.; and Zarepisheh, M. 2007. A dual-based algorithm for solving lexicographic multiple objective programs. *European Journal of Operational Research*, 176(3): 1348–1356.

Roijers, D. M.; and Whiteson, S. 2017. Multi-objective decision making. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 11(1): 1–129.

Saisubramanian, S.; Kamar, E.; and Zilberstein, S. 2020. A multi-objective approach to mitigate negative side effects. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 354–361.

Trevizan, F.; Thiébaux, S.; Santana, P.; and Williams, B. 2017. I-dual: Solving constrained SSPs via heuristic search in the dual space. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 4954–4958.

Trevizan, F. W.; Thiébaux, S.; and Haslum, P. 2017. Occupation measure heuristics for probabilistic planning. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling*, 306–315.

Trevizan, F. W.; Thiébaux, S.; Santana, P. H.; and Williams, B. C. 2016. Heuristic search in dual space for constrained stochastic shortest path problems. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling*, 326–334.

Wray, K. H.; Lui, R.; and Pedersen, L. 2021. Engine activation planning for series hybrid electric vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 238–244.

Wray, K. H.; Zilberstein, S.; and Mouaddib, A.-I. 2015. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 3418–3424.

Younes, H. L. S.; and Littman, M. L. 2004. PPDDL 1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical report.