# Revisiting Multi-Objective MDPs with Relaxed Lexicographic Preferences

**Luis Pineda** and **Kyle Hollins Wray** and **Shlomo Zilberstein**
College of Information and Computer Sciences
University of Massachusetts Amherst
{lpineda,wray,shlomo}@cs.umass.edu

### Abstract

We consider stochastic planning problems that involve multiple objectives such as minimizing task completion time and energy consumption. These problems can be modeled as multi-objective Markov decision processes (MOMDPs), an extension of the widely-used MDP model to handle problems involving multiple value functions. We focus on a subclass of MOMDPs in which the objectives have a *relaxed lexicographic structure*, allowing an agent to seek improvement in a lower-priority objective when the impact on a higher-priority objective is within some small given tolerance. We examine the relationship between this class of problems and *constrained MDPs*, showing that the latter offer an alternative solution method with strong guarantees. We show empirically that a recently introduced algorithm for MOMDPs may not offer the same strong guarantees, but it does perform well in practice.

## Introduction

Many stochastic planning problems involve a trade-off between multiple, possibly competing objectives. For example, balancing speed and safety in autonomous vehicles, or balancing energy consumption and speed in mobile devices. Multi-objective planning has been studied extensively in the context of a diverse set of applications such as energy conservation in commercial buildings (Kwak *et al.* 2012), semi-autonomous driving (Wray *et al.* 2015), water reservoir control (Castelletti *et al.* 2008) and autonomous robot exploration and search (Calisi *et al.* 2007).

Many of these planning problems involve uncertainty and can thus be naturally modeled as *Markov decision processes* (MDPs) problems (Bertsekas and Tsitsiklis 1991). When involving multiple-objectives, this leads to a class of problems known as Multi-Objective Markov Decision Processes (MOMDPs) (White 1982).

A common approach for solving MOMDPs is to use a scalarization function that combines all the objectives into a single one. The resulting problem can then be solved by using single-objective methods. Unfortunately, this approach doesn't work well in general, since there might be multiple Pareto optimal solutions to explore, or the proper scalarization weights are not known in advance.

On the other hand, several approaches have leveraged the fact that many problems allow an inherent lexicographic ordering of the objectives. For example, while driving a semi-autonomous vehicle, the driver may wish to minimize driving time, but also minimize the effort associated with driving (i.e., maximize autonomous operation of the vehicle) (Zilberstein 2015). Mouadibb used a strict lexicographic ordering for MOMDPs (Mouaddib 2004; Wray *et al.* 2015), while others explored lexicographic ordering of value functions (Mitten 1974; Sobel 1975) using a technique called *ordinal dynamic programming*, which has also been explored within reinforcement learning (Gábor *et al.* 1998; Natarajan and Tadepalli 2005).

More recently, Wray et al. (Wray *et al.* 2015) introduced LMDPs, a lexicographic variant of MDPs that includes relaxed lexicographic preferences through the use of slack variables. The main idea is to allow small deviations in higher priority value functions with the hopes of obtaining large improvements in the secondary ones. The authors also introduced a novel extension of Value Iteration, LVI, to solve LMDPs using slack.

In this paper we offer a deeper study of LMDPs, by introducing a more rigorous problem formulation, and showing that LVI can be interpreted as an efficient solution method for a variant of LMDPs where the decision-making is indifferent to local state-specific deviations in value. We also explore connections between LMDP and Constrained MDPs (Altman 1999), and use these to study the computational complexity of solving LMDPs. We also propose a simple CMDP-based algorithm to obtain optimal randomized policies for LMDPs.

## Problem Definition

A multi-objective Markov decision process (MOMDP) (Wray *et al.* 2015) is a tuple $\langle S, A, \mathcal{T}, \mathbf{C} \rangle$, where:

- $S$ is a finite set of states.
- $A$ is a finite set of actions.
- $\mathcal{T} : S \times A \times S \to [0, 1]$ is a state transition function that specifies the probability $T(s'|s, a)$ of ending in state $s'$ when action $a$ is taken in state $s$.
- $\mathbf{C}$ is a vector $[C_1(s, a), ..., C_k(s, a)]$ that specifies the cost of taking action $a$ in state $s$ under $k$ different reward functions $C_i : S \times A \to \mathbb{R}$, for $i \in K = \{1, ..., k\}$.

We focus on *infinite horizon* MOMDP, with a discount factor $\gamma \in [0, 1)$. Additionally, without loss of generality, we will consider problems in which an initial state $s_0$ is given. A solution to a MOMDP is a policy $\pi : S \rightarrow A$ that maps states to actions. The total expected cost of a policy $\pi$ over cost function $i$ is defined as

$$V_i^{\pi}(s_0) = \mathbb{E}\left[\sum_{t=0}^{t=\infty} \gamma^t C_i(s_t, \pi(s_t)); s_0\right] \quad (1)$$

where $\mathbb{E}$ represents expectation, and $V_i^{\pi}$ is a value function $V_i^{\pi} : S \rightarrow \mathbb{R}$ that represents the expected cumulative cost obtained by following policy $\pi$ starting at state $s$. We denote the vector of all total expected cumulative costs, starting in $s_0$, as $\mathbf{V}_{s_0}^{\pi} = [V_1^{\pi}(s_0)\ V_2^{\pi}(s_0)\ ...\ V_k^{\pi}(s_0)]$.

In the presence of multiple cost functions there are many possible ways to define optimality of a solution. Common examples are minimizing some linear combination of the given cost functions or finding Pareto-optimal solutions. In an LMDP the reward functions $C_1, C_2, ..., C_k$ are ordered according to a lexicographic preference. That is, a solution to an LMDP minimizes the expected cumulative cost, following a lexicographic preference over the cost functions. Specifically, this means that $\forall i, j \in K\ \ s.t.\ \ i < j$, an arbitrarily small improvement in $C_i$ is preferred to an arbitrarily large improvement in $C_j$.

Given this lexicographic preference, we can compare the quality of any two policies $\pi_1$ and $\pi_2$ using the following operator:

$$\mathbf{V}_{s_0}^{\pi_1} \succ \mathbf{V}_{s_0}^{\pi_2} \Leftrightarrow \exists_{j \in K} \forall_{i<j},$$
$$\left(V_j^{\pi_1}(s_0) < V_j^{\pi_2}(s_0)\right) \wedge \left(V_i^{\pi_1}(s_0) = V_i^{\pi_2}(s_0)\right) \quad (2)$$

When neither $\mathbf{V}_{s_0}^{\pi_1} \succ \mathbf{V}_{s_0}^{\pi_2}$ nor $\mathbf{V}_{s_0}^{\pi_2} \succ \mathbf{V}_{s_0}^{\pi_1}$, we say that the policies are equally good (equally preferred). The $\succ$ operator imposes a partial order among policies. Hence an *optimal solution* to an LMDP (without slack) is a policy $\pi^*$ such that there is no policy $\pi$ s.t. $\mathbf{V}_{s_0}^{\pi} \succ \mathbf{V}_{s_0}^{\pi^*}$. We use the notation $\mathbf{V}_{s_0}^{*} = [V_1^{*}(s_0)\ V_2^{*}(s_0)\ ...\ V_k^{*}(s_0)]$ to refer to the vector of optimal expected costs.

### Relaxed lexicographic preferences

We can relax the strong lexicographic preference presented above by using *slack* variables (Wray *et al.* 2015). Concretely, we introduce a vector of slack variables $\delta = [\delta_1, ..., \delta_k]$ such that $\forall i \in K, \delta_i \geqslant 0$. The slack $\delta_i$ represents how much we are willing to compromise with respect to $V_i^*(s_0)$ in order to potentially improve the value of $V_{i+1}(s_0)$. Let $\Pi$ denote the set of all possible policies for a given LMDP, $\mathcal{L}$. Then, starting with $\Pi_0^{\mathcal{L}} = \Pi$, we define the relaxed preference as as follows:

$$\hat{V}_{i+1}(s_0) = \min_{\pi \in \Pi_i^{\mathcal{L}}} V_{i+1}^{\pi}(s_0) \quad (3)$$

$$\Pi_{i+1}^{\mathcal{L}} = \{\pi \in \Pi_i^{\mathcal{L}} | V_{i+1}^{\pi}(s_0) \leqslant \hat{V}_{i+1}(s_0) + \delta_i\} \quad (4)$$

with an optimal policy for a LMDP defined as any policy $\pi^* \in \Pi_k^{\mathcal{L}}$. In the rest of the paper we use the term LMDP to refers to problems that include such relaxed lexicographic preference.

**Algorithm 1:** A polynomial-time transformation from CMDPs to LMDPs

**cmdp2lmdp**
**input** : CMDP $\mathcal{C} = \langle S, A, \mathcal{T}, \mathbf{C}, \mathbf{T} \rangle$
**output**: $\pi$
1 **for** $i \leftarrow k, k-1, \ldots, 1$ **do**
2  $\bar{\mathbf{C}} \leftarrow [C_k\ \ C_{k-1}\ \ \ldots\ \ C_i]$
3  **for** $j = k, \ldots, i+1$ **do**
4  $\Delta_j \leftarrow T_j - \tilde{V}^j(s_0)$
5  $\bar{\delta} \leftarrow [\Delta_k\ \ \Delta_{k-1}\ \ \ldots\ \ \Delta_{i+1}\ \ 0]$
6  Create LMDP $\mathcal{L}^i \leftarrow \langle S, A, \mathcal{T}, \bar{\mathbf{C}} \rangle$
7  Solve $\mathcal{L}^i$ with slack $\bar{\delta}$ to obtain $\pi^{\mathcal{L}^i} \in \Pi_{k-i+1}^{\mathcal{L}^i}$ and cost $[V_k^{\mathcal{L}}(s_0)\ \ V_{k-1}^{\mathcal{L}}(s_0)\ \ \ldots\ \ \tilde{V}^i(s_0)]$
8  **if** $\tilde{V}^i(s_0) > T_i$ **then**
9  **return** *no solution*
10  $\pi \leftarrow \pi^{\mathcal{L}^i}$

## Connections with Constrained MDPs

In this section we establish some connections between solving LMDPs and a class of problems know as Constrained MDPs (CMDPs) (Altman 1999), and use these connections to assert some properties about the computational complexity of LMDPs. A CMDP is a MOMDP in which some of the cost functions are given specific budget limits, referred to as *targets*. In particular, a solution to a CMDP is a policy $\pi$ that satisfies:

$$\pi = \arg\min_{\pi \in \Pi_0} V_1^{\pi}(s_0)$$
$$\text{s.t. } V_j^{\pi}(s_0) \leqslant T_j \quad j = 2, \ldots, k$$

As it turns out, CMDPs and LMDPs have important computational connections, as shown by Algorithm 1, which describes a procedure for solving CMDPs as a sequence of $k$ LMDPs. We use the notation $\mathbf{T}$ to represent the vector of targets for the CMDP.

The algorithm proceeds by iterating over the constraints in the CMDP in reverse order, and creating an LMDP $\mathcal{L}^i$ in every iteration $i$ (line 6). The LMDPs are designed so that the optimal policy for $\mathcal{L}^i$ satisfies constraints $i, i+1, ..., k$ in the original CMDP. This is done by setting the slack vector in lines 3-5 appropriately, using the minimum values obtained for $V_{i+1}, \ldots, V_k$ to compute how much slack is necessary to maintain the constraint targets; the solution of LMDP $\mathcal{L}^i$ then minimizes $V_i$, while respecting targets $T_{i+1}, \ldots, T_k$. The value $\tilde{V}^j(s_0)$ used to compute $\Delta_j$ (line 4) is obtained from the solution to $\mathcal{L}^j$ (line 7) as the value that minimizes $V_j(s_0)$ subject to all the previous constraints. Note that even tough the value $\tilde{V}^k(s_0)$ is never computed, this is not an issue, since in the first iteration line 4 is skipped.

**Proposition 1.** *When Algorithm 1 terminates, policy $\pi$ is an optimal solution to the input problem $\mathcal{C}$.*

*Proof.* We start by establishing the following inductive

property: at the end of the for loop in lines 1-10 we have

$$\forall_{1 \leqslant j \leqslant k-i+1},\ \Pi_j^{\mathcal{L}^i} = \bigcap_{\ell=k-j+1}^{k} \Pi_\ell^{\mathcal{C}}$$

where $\Pi_\ell^{\mathcal{C}}$ represents the set of all policies $\pi'$ such that $V_\ell^{\pi'}(s_0) \leqslant T_\ell$. The base case is $i = k$, which is trivial given that $\mathcal{L}^k$ in line 6 just minimizes $C_k$. Consider the solution of $\mathcal{L}^i$ for any $i < k$. Note that $\mathcal{L}^i$ is exactly the same as $\mathcal{L}^{i+1}$, except for three changes: the addition of cost function $C_i$ at the end of $\bar{C}$, changing $\delta_{i+1}$ from 0 to $\Delta_{i+1}$, and appending 0 at the end of $\delta$. Therefore, we have that $\forall_{1 \leqslant j \leqslant k-i-1},\ \Pi_j^{\mathcal{L}^i} = \Pi_j^{\mathcal{L}^{i+1}}$ (see Eq. 4). Using the definition of $\Delta_{i+1}$ and Eq. 4, we get

$$\Pi_{k-i}^{\mathcal{L}^i} = \{\pi' \in \Pi_{k-i-1}^{\mathcal{L}^{i+1}} | V_{i+1}^{\pi'}(s_0) \leqslant T_{i+1}\}$$

Putting this definition and the inductive property together we get $\Pi_{k-i}^{\mathcal{L}^\ell} = \bigcap_{j=i+1}^{k} \Pi_j^{\mathcal{C}}$. Therefore, since the solution of $\mathcal{L}^i$ minimizes cost function $C_i$ with no slack, we have $\Pi_{k-i+1}^{\mathcal{L}^\ell} = \bigcap_{j=i}^{k} \Pi_j^{\mathcal{C}}$, which proves the inductive property. We can now complete the proof, by using the inductive property with $i = 1$ and noting that the last step just minimizes for $C_1$, as in the definition of the CMDP. □

Algorithm 1 allows us to establish some properties about the computational complexity of LMDPs. As it turns out, it's been shown that finding an optimal *deterministic* policy for a CMDP is an NP-hard problem in the maximum of the number of states, number of actions, or number of constraints, $k$ (Feinberg 2000). Thus, we can establish the following result about the computational complexity of solving LMDPs.

**Lemma 1.** *Finding an optimal deterministic policy for a LMDP is NP-hard in the maximum of the number of states, number of actions or number of cost functions.*

*Proof.* The proof follows directly from the complexity of CMDPs and the definition of Algorithm 1, since having an algorithm for solving LMDPs in polynomial time would imply a polynomial-time algorithm for solving CMDPs. □

Note that Lemma 1 implies that finding optimal deterministic policies for LMDPs is much harder than finding such policies for regular MDPs, which can be done in polynomial time in the number of states (Papadimitriou and Tsitsiklis 1987). Interestingly, we can obtain a polynomial-time algorithm for finding an optimal randomized policy for a LMDP by using a reverse transformation to the one shown in Algorithm 1. This idea is illustrated in Algorithm 2. We call this approach **CM-Map**, for Constrained MDP Mapping.

## Comparison with LVI

Wray *et al.* [2015] introduced an algorithm called LVI (Lexicographic Value Iteration) for solving a variant of LMDPs based on Value Iteration. Interestingly, LVI could provide approximate solutions to solving LMDPs with relaxed lexicographic preferences of the type considered in this paper.

We begin the comparison by describing the main ideas behind LVI. The LVI algorithm works iteratively, in increasing order of lexicographic preference, by using Value Iteration to solve for a single cost function at a time. After finding an optimal policy for the *i-th* cost function, the algorithm prunes the set of available actions at each state $s \in S$ so that a bounded deviation from optimal is guaranteed, thus maintaining a relaxed lexicographic preference. Note that this pruning of actions implicitly establishes a new space of policies, which we denote $\hat{\Pi}_i$, over which the optimization for cost function $i + 1$ will be performed.

**Algorithm 2: CM-Map**: A CMDP-based algorithm for solving LMDPs with relaxed preferences.

**lmdp2cmdp**
**input** : LMDP $\mathcal{L} = \langle S, A, \mathcal{T}, \mathbf{C} \rangle$
**output**: $\pi$
1 $\mathbf{T} \leftarrow []$
2 **for** $i = 1, ..., k$ **do**
3 $\quad \bar{\mathbf{C}} \leftarrow [C_i, C_1, \cdots, C_{i-1}]$
4 $\quad$ Create CMDP $\mathcal{C} \leftarrow \langle S, A, \mathcal{T}, \bar{\mathbf{C}}, \mathbf{T} \rangle$
5 $\quad$ Solve $\mathcal{C}$ to obtain policy $\pi$
6 $\quad t_i \leftarrow V_i^\pi(s_0) + \delta_i$
7 $\quad \mathbf{T} \leftarrow [\mathbf{T}\ t_i]$

Concretely, LVI prunes actions according to:

$$A_{i+1}(s) = \{a \in A_i(s) | Q_i^{\hat{\Pi}_{i-1}}(s, a) \leqslant V_i^{\hat{\Pi}_{i-1}}(s) + \eta_i\} \quad (5)$$

where $\eta_i$ is a *local* slack term that relaxes the lexicographic preferences with respect to all possible actions that can be taken in state $s$. We use the notation $V_i^{\hat{\Pi}_i}$ to represent the best possible expected cumulative cost that can be obtained in state $s$ by following policies in $\hat{\Pi}_i$. Similarly, the term $Q_i^{\hat{\Pi}_i}(s, a)$ represents the expected cost of taking action $a$ in state $s$ and following the best policy in $\hat{\Pi}_i$ afterward. The set $\hat{\Pi}_i$ is defined as $\hat{\Pi}_i \leftarrow \times_{s \in S} A_{i+1}$, where initially $A_1 \leftarrow A$. Essentially, the slack $\eta_i$ is a one-step relaxation of the lexicographic preference. Note that, for all states $s \in S$, the algorithm is allowed to deviate from the optimal value up to a slack of $\eta_i$, without accounting from deviation in the rest of states. Therefore, the final policy $\pi$ returned by LVI need not satisfy $V_i^\pi(s) < V_i^{\hat{\Pi}_{i-1}}(s) + \eta_i$, since deviations accumulate.

The authors of LVI address this issue by establishing the following proposition:

**Proposition 2.** *(Wray* et al. *2015) If $\eta_i = (1 - \gamma)\delta_i$, then $V_i^\pi(s) \leqslant V_i^{\hat{\Pi}_{i-1}}(s) + \delta_i$ for all states $s \in S$.*

Note that an implication of this proposition is that, if $\hat{\Pi}_{i-1} = \Pi_{i-1}^{\mathcal{L}}$, where $\Pi_i^{\mathcal{L}}$ is as defined in Equation 2, then $\hat{\Pi}_i \subseteq \Pi_i^{\mathcal{L}}$. In other words, LVI doesn't break any lexicographic preferences already implied by $\Pi_{i-1}^{\mathcal{L}}$.

Unfortunately, the local slack principle used by LVI offers no guarantees about the quality of the policy set $\hat{\Pi}_i$ over the $(i + 1)$-*th* cost function, and, in fact, $V_{i+1}^{\hat{\Pi}_i}(s)$ could be arbitrarily worse than $V_{i+1}^{\Pi_i^{\mathcal{L}}}(s)$.

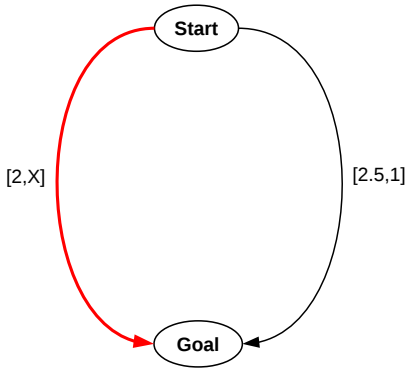


Figure 1: Example where LVI results in an arbitrarily large increase in cost with respect to the optimal value respecting the relaxed lexicographic preference.

This situation, where pruning actions according to the local slack rule leads to bad consequences globally, is illustrated in Figure 1. Using $\delta = 1$ and $\gamma = 0.9$, we get $\eta = 0.1$ and therefore action *right* will be pruned by LVI. But note that the resulting value $V_2(s_0) = X$ can be arbitrarily worse than the optimal value for $V_2$ *within* the specified slack for $V_1$. The reason is that, to be able to maintain the lexicographic preference, LVI prunes the policy space too harshly.

Despite these lack of guarantees with respect to the LMDP problem with lexicographic preferences, LVI has the advantage of being able to find deterministic policies relatively quickly. Therefore, it can be seen as an efficient solution method for a variant of LMDP where the decision-making is instead indifferent to local state-specific deviations in value. When applied to the stricter version of the problem considered in this paper, LVI provides a fast approximation method, which is important, given the complexity results established in Lemma 1. Moreover, as shown in the experimental results section, by ignoring the strong slack bound proposed in Proposition 2, we can interpret the vector of local slacks, $\eta = [\eta_1 \ \cdots \ \eta_k]$, as a set of parameters that can be tuned to achieve a result closer to optimal.

## Experimental Results

In this section we compare the CMDP-based solution approach described above, CM-Map, with LVI for the solution of the LMDP problem. As a testbed we use a multi-objective variant of the Racetrack Problem, a widely-used reinforcement learning benchmark (Sutton and Barto 1998).

The racetrack problem involves a simulation of a race car on a discrete track of some length and shape, where a starting line has been drawn on one end and a finish line on the opposite end of the track. The state of the car is determined by its location and its two-dimensional velocity. The car can change its speed in each of the two dimensions by at most 1 unit, resulting in a total of nine possible actions. After applying an action there is a probability $p_s$ that the resulting acceleration is zero, simulating failed attempts to accelerate/decelerate because of unpredictably slipping on the track.

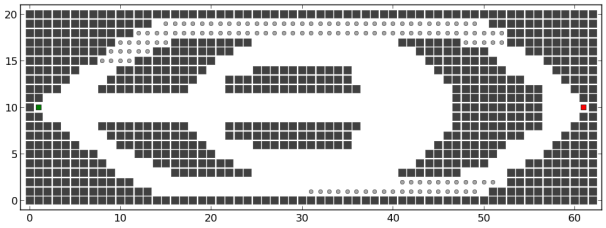


Figure 2: Example of a race track used in the experiments.

Table 1: Performance of CM-Map and LVI on a racetrack with 14,271 states (Track1).

| $\delta$ | $V_1(s_0)$ | $V_2(s_0)$ | $V_3(s_0)$ | Time (sec.) |
|---|---|---|---|---|
| Without any slack | | | | |
| N/A | 13.73 | 24.25 | 28.51 | N/A |
| CM-Map | | | | |
| 1.0 | 14.46 | 22.16 | 16.77 | 209 |
| 2.0 | 14.87 | 23.25 | 15.32 | 140 |
| 5.0 | 14.91 | 26.19 | 15.03 | 170 |
| LVI | | | | |
| 1.0 | 13.74 | 23.90 | 28.16 | 8 |
| 2.0 | 13.74 | 23.88 | 28.07 | 8 |
| 5.0 | 13.74 | 23.78 | 30.77 | 8 |

Our multi-objective variant of the racetrack problem involves three cost functions. The first is the usual racetrack cost function, which minimizes the number of steps to reach the goal. The second objective penalizes changes in direction, and the third tries to avoid a set of pre-defined "unsafe" locations. An example of a racetrack used in our experiments is shown in Figure 2, where the unsafe locations are represented by gray dots.

Tables 1 and 2 show the results of applying CM-Map and LVI to two racetrack problems of different shapes and sizes. We experimented with different values of $\delta$ and in all cases used $\gamma = 0.99$. For CM-Map we solved the generated CMDPs using the Gurobi linear programming solver (Gurobi Optimization, Inc. 2015). For LVI, we used the value of $\eta$ suggested by Proposition 2.

As expected, increasing the value of $\delta$ leads to CM-Map favoring lower priority cost functions. In both problems, using a slack of $\delta = 1$ already leads to significant gains in the second and third cost functions with respect to using no slack at all. In general, the significance of these gains will obviously be problem-dependent. However, in all cases these values will be guaranteed to be the optimal values of the lower priority cost functions, within the desired slack of the higher priority ones. Note that in these experiments, increasing the slack $\delta_1$ doesn't necessarily translate into a lower cost $V_2$, since we are also increasing $\delta_2$. This is why CM-Map produces higher values for $V_2$ when $\delta = 5$ is used.

Table 2: Performance of CM-Map and LVI on a racetrack with 24,602 states (Track2).

| $\delta$ | $V_1(s_0)$ | $V_2(s_0)$ | $V_3(s_0)$ | Time (sec.) |
|---|---|---|---|---|
| Without any slack | | | | |
| N/A | 19.64 | 55.39 | 43.86 | N/A |
| CM-Map | | | | |
| 1.0 | 20.66 | 42.31 | 34.73 | 3,611 |
| 2.0 | 21.64 | 40.58 | 33.61 | 827 |
| 5.0 | 21.69 | 42.98 | 31.42 | 732 |
| LVI | | | | |
| 1.0 | 19.62 | 55.34 | 43.82 | 16 |
| 2.0 | 19.61 | 55.31 | 43.81 | 16 |
| 5.0 | 19.64 | 54.64 | 44.02 | 16 |

Table 3: Performance of LVI on Track1 using different values of $\eta$.

| $\eta$ | $V_1(s_0)$ | $V_2(s_0)$ | $V_3(s_0)$ | Time (sec.) |
|---|---|---|---|---|
| 0.5 | 14.04 | 21.69 | 21.57 | 9 |
| 1.0 | 14.24 | 21.77 | 19.08 | 30 |
| 2.0 | 14.49 | 23.59 | 16.99 | 32 |
| 5.0 | 14.94 | 28.08 | 15.16 | 40 |

On the other hand, LVI failed to make any significant change in costs, with respect to using no slack, when using the values of $\eta$ suggested by Proposition 2. As mentioned before, the reason for this is the strong bound on the local slack needed to guarantee that the slack $\delta$ is respected. However, note that the running time for LVI is orders of magnitude faster than CM-Map.

To leverage the faster computational cost of LVI, we used $\eta$ as a free parameter that can be tuned to vary the trade-offs between optimizing higher and lower priority functions. We evaluated LVI on the same two race tracks using values of $\eta$ of 0.5, 1, 2 and 5, which correspond to values of $\delta$ equal to 50, 100, 200 and 500, respectively. The results of these experiments are shown in Tables 3 and 4. Note that increasing the value of $\eta$ allowed LVI to obtain lower values for the second and third cost functions, while still maintaining values of the first cost function close to optimal. Moreover, in most cases the algorithm still ran several times faster than CM-Map.

## Conclusion

In this paper we introduce a rigorous formulation of the MDP problem with relaxed lexicographic preferences. The problem is defined as a sequential problem that increasingly restricts the set of available policies over which optimization will be performed. For each cost function $V_i$, in decreasing order of preference, the set of policies is restricted to the subset of the previous set of policies whose expected cost is within a slack of the optimal value $V_i$ in this set.

We explore connections between this problem formulation and Constrained MDPs. In particular, we show that both problems are computationally equivalent, and use this result to show that computing optimal deterministic policies for LMDPs is NP-hard in the maximum of the number of states, number of actions and number of cost functions. We also use these connections to introduce C-Map, a CMDP-based algorithm for finding optimal randomized policies for LMDPs.

Table 4: Performance of LVI on Track2 using different values of $\eta$.

| $\eta$ | $V_1(s_0)$ | $V_2(s_0)$ | $V_3(s_0)$ | Time (sec.) |
|---|---|---|---|---|
| 0.5 | 21.59 | 39.36 | 58.76 | 21 |
| 1.0 | 21.34 | 39.56 | 39.14 | 629 |
| 2.0 | 21.01 | 43.03 | 34.05 | 749 |
| 5.0 | 20.83 | 47.04 | 32.57 | 916 |

We also offer a deeper study of the Lexicographic Value Iteration (LVI) algorithm proposed by Wray et al. (Wray *et al.* 2015) and show that it can be seen as tailored for a variant of LMDP where the decision maker is indifferent to local state-specific deviations in value. We compare its performance with C-Map and observe that it can produce good results with much faster computational time than C-Map, although without the same theoretical guarantees.

In future work we want to leverage the insight gained by this novel formulation in order to devise fast approximation algorithms to solve the LMDP problem. We are particular interested in developing other extensions of value iteration, as well as algorithms based on heuristic search, such as LAO* (Hansen and Zilberstein 1998; 2001).

## Acknowledgments

This work was supported in part by the National Science Foundation grant number IIS-1405550.

## References

Eitan Altman. *Constrained Markov Decision Processes*. Stochastic Modeling Series. Taylor & Francis, 1999.

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

Daniele Calisi, Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Multi-objective exploration and search for autonomous rescue robots. *Journal of Field Robotics*, 24(8-9):763–777, 2007.

Andrea Castelletti, Francesca Pianosi, and Rodolfo Soncini-Sessa. Water reservoir control under economic, social and environmental constraints. *Automatica*, 44(6):1595–1607, 2008.

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

Zoltán Gábor, Zsolt Kalmár, and Csaba Szepesvári. Multi-criteria reinforcement learning. In *Proceedings of the 15th International Conference on Machine Learning*, pages 197–205, 1998.

Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual. http://www.gurobi.com, 2015.

Eric A. Hansen and Shlomo Zilberstein. Heuristic search in cyclic AND/OR graphs. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 412–418, 1998.

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

Jun-young Kwak, Pradeep Varakantham, Rajiv Maheswaran, Milind Tambe, Farrokh Jazizadeh, Geoffrey Kavulya, Laura Klein, Burcin Becerik-Gerber, Timothy Hayes, and Wendy Wood. SAVES: A sustainable multiagent application to conserve building energy considering occupants. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 21–28, 2012.

L. G. Mitten. Preference order dynamic programming. *Management Science*, 21(1):43–46, 1974.

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

Sriraam Natarajan and Prasad Tadepalli. Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd International Conference on Machine learning*, pages 601–608, 2005.

Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Matthew J. Sobel. Ordinal dynamic programming. *Management science*, 21(9):967–975, 1975.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998.

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

Kyle Hollins Wray, Shlomo Zilberstein, and Abdel-Illah Mouaddib. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 3418–3424, 2015.

Shlomo Zilberstein. Building strong semi-autonomous systems. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 4088–4092, 2015.