Multi-Objective POMDPs with Lexicographic Reward Preferences

Kyle Hollins Wray and **Shlomo Zilberstein**
College of Information and Computer Sciences
University of Massachusetts, Amherst, MA 01003
{wray, shlomo}@cs.umass.edu

Abstract

We propose a model, Lexicographic Partially Observable Markov Decision Process (LPOMDP), which extends POMDPs with lexicographic preferences over multiple value functions. It allows for slack–slightly less-than-optimal values–for higher-priority preferences to facilitate improvement in lower-priority value functions. Many real life situations are naturally captured by LPOMDPs with slack. We consider a semi-autonomous driving scenario in which time spent on the road is minimized, while maximizing time spent driving autonomously. We propose two solutions to LPOMDPs–Lexicographic Value Iteration (LVI) and Lexicographic Point-Based Value Iteration (LPBVI), establishing convergence results and correctness within strong slack bounds. We test the algorithms using real-world road data provided by Open Street Map (OSM) within 10 major cities. Finally, we present GPU-based optimizations for point-based solvers, demonstrating that their application enables us to quickly solve vastly larger LPOMDPs and other variations of POMDPs.

1 Introduction

Planning with multiple objectives is prevalent throughout numerous problem domains such as water reservoir control [Castelletti *et al.*, 2008], industrial scheduling [Aissani *et al.*, 2009], energy-conserving smart environments [Kwak *et al.*, 2012], anthrax outbreak detection [Soh and Demiris, 2011b], and semi-autonomous driving [Wray *et al.*, 2015]. Multi-objective Markov Decision Processes (MOMDPs) are commonly used to solve these sequential optimization problems, leveraging a scalarization function which combines the multiple objectives into a single one. However, finding a suitable scalarization function is non-trivial because there may be many valid Pareto optimal solutions to explore, and combining objective functions with differing units may be impossible. Furthermore, according to a recent survey, several existing algorithms are computationally expensive [Roijers *et al.*, 2013]. We leverage the observation that lexicographic ordering of objectives offers a natural way to describe numerous optimization problems [Mitten, 1974; Sobel, 1975; Gábor *et al.*, 1998; Rangcheng *et al.*, 2001; Wray *et al.*, 2015]. We optimize each objective function in order, only using subsequent objectives to break ties.

Recently, a partially observable model for Multi-Objective POMDPs (MOPOMDP) has been introduced, which generalizes POMDPs using a vector of rewards [Soh and Demiris, 2011a]. We define a new MOPOMDP model with a lexicographic preference over rewards, building upon previous work in this area [Rangcheng *et al.*, 2001] by introducing the notion of *slack* (allowing small deviation from optimal). This extends the original Lexicographic MDP (LMDP) model [Wray *et al.*, 2015]. Slack increases the space of possible actions for subsequent value functions to utilize, as long as the deviation caused by sub-optimal action selection is within the allotted slack. For example, the highest-priority objective may be to minimize execution time, but within some neighborhood of the optimal time, a secondary objective could be optimized. Work on MOPOMDPs has been sparse outside of evolutionary approaches [Soh and Demiris, 2011a; 2011b]. Constrained POMDPs (CPOMDP) present a similar framework, which states an additional constraint function with a budget [Isom *et al.*, 2008]. The expected constraint penalty must be less than a given limit. In essence, CPOMDPs present a related but distinct problem, since we characterize our problem using deviation from optimal (slack) and allow for many rewards following a preference.

We consider a semi-autonomous driving scenario, which requires the driver to take over control of the vehicle under some conditions [Zilberstein, 2015]. This scenario consists of a car capable of autonomous driving on selected well-mapped roads such as major highways. In order to ensure safety, the car monitors the state of the driver: *attentive* or *tired*. The system is not perfect, however, and obtains noisy estimates of the driver's true state. We seek to minimize the travel time while maximizing the time spent safely driving autonomously when the driver is tired. Since autonomous driving is only possible on some roads, it generally increases travel time, but could also increase safety when the driver is tired. The introduction of slack allows the car to select a slightly longer route in order to travel on safe roads which are autonomy-capable.

Our contributions include a new model entitled LPOMDP, which extends an LMDP by addressing partial observability. We present two algorithms for solving LPOMDPs: LVI and LPBVI. These are non-trivial extensions, due in part to our novel definition of slack. Without slack, lexicographic preference orderings would have been far less interesting to utilize because the likelihood of a tie among α-vectors at a belief point is highly unlikely in practice. Hence, without slack, the result would often be a unique policy that optimizes the highest-priority value function while ignoring all secondary optimization criteria. We present three propositions that prove convergence and the proper bounds on slack assignment. We experiment within a recently proposed semi-autonomous driving domain, which constructs POMDPs from real-world road data, using Open Street Map (OSM). Furthermore, we present a novel method for point-based POMDP and LPOMDP solvers that leverages the massive parallel processing power of a Graphics Processing Unit (GPU). We demonstrate performance of both CPU and GPU implementations within this realistic domain.

Section 2 formally introduces the problem description and relevant background material. Section 3 presents the Lexicographic Value Iteration (LVI) algorithm for LPOMDPs, as well as the Lexicographic Point-Based Value Iteration (LPBVI) variant which enables scalability. Section 4 introduces a convergence result and two strong bounds on slack for both LVI and LPBVI. Section 5 grounds our theoretical results within the semi-autonomous driving application. Additionally, this section covers the GPU optimization that vastly improves performance. Section 6 concludes with final thoughts.

2 Problem Definition

LPOMDP encapsulates a multi-objective, partially observable, sequential decision problem such that for each state s the agent lexicographically prefers to improve the value of the state for objective i over objective $i+1$. The LPOMDP model extends the Lexicographic MDP (LMDP) model to include partial observability [Wray *et al.*, 2015].

Definition 1. A **Lexicographic POMDP (LPOMDP)** is represented by a 7-tuple $\langle S, A, \Omega, T, O, \mathbf{R}, \delta \rangle$:

- S is a finite set of n states
- A is a finite set of m actions, with $A(s)$ denoting the actions available in state $s \in S$
- Ω is a finite set of z observations
- $T : S \times A \times S \rightarrow [0,1]$ is a state transition function, with $T(s,a,s') = Pr(s'|s,a)$ denoting taking the action $a \in A$ in state $s \in S$, then transitioning to state $s' \in S$
- $O : A \times S \times \Omega \rightarrow [0,1]$ is an observation transition function, with $O(a,s',\omega) = Pr(\omega|a,s')$ denoting the probability of observing $\omega \in \Omega$ after taking the action $a \in A$ and entering successor state $s' \in S$
- $\mathbf{R} = [R_1, \dots, R_k]^T$ is a vector of k reward functions, with $R_i : S \times A \rightarrow \mathbb{R}$, $i \in K = \{1, \dots, k\}$, mapping a state $s \in S$ and action $a \in A$ to a reward $R_i(s,a)$
- $\delta = \langle \delta_1, \dots, \delta_k \rangle$ is a tuple of slack values, $\delta_i \geqslant 0, \forall i \in K$

The *horizon* h denotes the number of time steps in the stochastic process, namely *finite* ($h < \infty$) and *infinite* ($h = \infty$), with *discount factor* $\gamma \in [0,1)$. As with most point-based POMDP solvers, we will consider infinite horizon POMDPs and approximately solve them via a large finite horizon.

The agent maintains a *belief* over the current state, given the *history* of observations and actions. It is often useful to refer to a set of *beliefs* $B \subseteq \triangle^n$, within the standard n-simplex $\triangle^n$. At each time step, the agent has a belief $b \in B$, takes an action $a \in A$, makes an observation $\omega \in \Omega$, and updates the belief to $b' \in B$ for all states $s' \in S$:

$$b'(s'|b,a,\omega) = c\,O(a,s',\omega) \sum_{s \in S} T(s,a,s') b(s) \qquad (1)$$

with normalizing constant $c = Pr(\omega|b,a)^{-1}$ [Kaelbling *et al.*, 1998]. We often write $b' = [b'(s_1|b,a,\omega), \dots, b'(s_n|b,a,\omega)]^T$. The belief state is a sufficient statistic for a history. Note the belief does not depend on the reward vector.

Definition 1 is a direct extension of the original LMDP definition to include partial observability, although we have chosen to omit the state-dependent orderings, wherein the preference ordering could change depending on the *belief state*. Formally, we would include $\mathcal{S} = \{S_1, \dots, S_\ell\}$ to be an ℓ-partition over $B = \triangle^n$ and a tuple of strict preference orderings for each $o = \langle o_1, \dots, o_\ell \rangle$, with o_j denoting an ordering over the rewards. In other words, we will limit the scope of our investigation to $\ell = 1$, i.e., one preference ordering over all belief states: $\mathcal{S} = \{S_1\}$, $S_1 = \triangle^n$, $o = \langle o_1 \rangle$, and $o_1 = \langle 1, \dots, k \rangle$. The reason for this is that LPOMDPs are quite expressive with just a single preference-ordering and the inclusion of slack. The additional complexity of belief state-dependent orderings, which previously required a partition assumption for convergence, will be left to future work. In any case, the ordering may be naturally applied following the LMDP model, and our two algorithms for LPOMDPs may also be extended in the natural way.

Policy representations for LPOMDPs are the same as for POMDPs, e.g., *policy trees* or *policy graphs*. Formally, we define a *policy* $\pi : B \rightarrow A$ as a mapping from beliefs to actions [Sondik, 1978]. A *value function* maps belief states to a real value, such that for each reward $i \in K$, $V_i : B \rightarrow \mathbb{R}$. Let $\mathbf{V} = [V_1, \dots, V_k]^T$ refer to the vector of value functions. Each value function in belief space, for finite horizon LPOMDPs, is piecewise linear and convex (PWLC) [Smallwood and Sondik, 1973]. This can be represented as a set of r α-*vectors* $\Gamma_i = \{\alpha_{i1}, \dots, \alpha_{ir}\}, \forall i \in K$, such that $\forall j \in \{1, \dots, r\}$, $\alpha_{ij} = [\alpha_{ij}(s_1), \dots, \alpha_{ij}(s_n)]^T = [V_i(s_1), \dots, V_i(s_n)]^T$.

Each α-vector has an associated action, we denote as $a^\alpha \in A_i(b) \subseteq A$, with $A_i(b)$ denoting the set of actions available at belief point $b \in B$. Finite horizon LPOMDPs have a Γ^t, for each time step t. In the exhaustive case, each Γ^t denotes the α-vectors for all possible histories of length $2t$ (actions and observations). Infinite horizon LPOMDPs are only approximated by a set of α-vectors, as with POMDPs [Sondik, 1978], explained in the next section.

We write Bellman's equation for the induced continuous-state MDP [Kaelbling *et al.*, 1998] at horizon h, following policy π, given an initial belief state $b \in B$ [Sondik, 1978]:

$$\mathbf{V}_\pi^h(b) = \mathbb{E}\Big[\sum_{t=0}^{h} \gamma^t \mathbf{R}(b^t, \pi^t(b^t)) \Big| b^0 = b, \pi\Big]$$

Examining any individual value function $i \in K$ at time t, we may write the Bellman optimality equation with:

$r_{ia} = [R_i(s_1, a), \dots, R_i(s_n, a)]^T$:

$$V_i^t(b) = \max_{\alpha \in \Gamma_i^t} b \cdot \alpha = \max_{a \in A_i^t(b)} Q_i^t(b, a) \tag{2}$$

$$Q_i^t(b, a) = b \cdot r_{ia} + \gamma \sum_{\omega \in \Omega} Pr(\omega | b, a) V_i^{t-1}(b') \tag{3}$$

Applying a lexicographic preference ordering over value functions follows similar logic as LMDPs. In an LMDP, for a particular value function $i \in K$, following the ordering, and a state $s \in S$, we allow for a *one-step* deviation from the optimal value at that state defined by $\eta_i \geqslant 0$. This enables a restriction on the set of actions available at this state for the next value function $i + 1$ following the ordering. Each value function restricts the actions for subsequent ones, until all value functions have been optimized under these constraints.

This process differs for an LPOMDP, since we do this for a belief state, meaning the set of actions must be appropriately restricted for *all belief points*. At each time step, for each belief point, we retain α-vectors' actions which are dominated so long as they are within η_i of the maximal value. So long as *there exists* a belief point for which the α-vector is within η_i of optimal, we must keep it in the set Γ_i^t.

Formally, for a value function $i \in \{1, \dots, k-1\}$, we restrict the set of actions for a belief point $b \in B$, by defining the set of actions available to the belief point $A_i(b) \subseteq A$:

$$A_{i+1}(b) = \{a \in A_i(b) | \max_{a' \in A_i(b)} Q_i(b, a') - Q_i(b, a) \leqslant \eta_i\} \tag{4}$$

with converged values Q_i. Since LPOMDPs have an equivalent representation in terms of α-vectors (Equation 2), we may also write this as a restriction to a new set $\bar{\Gamma}_{i+1}$ given the original set of α-vectors Γ_{i+1}:

$$\begin{aligned} \bar{\Gamma}_{i+1} = \{\alpha \in \Gamma_{i+1} | & \max_{\alpha' \in \bar{\Gamma}_i} b \cdot \alpha' - b \cdot \alpha'' \leqslant \eta_i \\ & \text{for } \alpha'' \in \bar{\Gamma}_i \text{ and } a_\alpha = a_{\alpha''}\} \end{aligned} \tag{5}$$

Figure 1 illustrates this action restriction. It depicts two belief points b_1 and b_2; two value functions V_1 and V_2; and four α-vectors for each value function. The unknown true (infinite horizon) value of $V_1(b)$ and $V_2(b)$ are shown, as well as the original $V_1'(b)$ and $V_2'(b)$ which would have been selected if slack was not introduced. First we examine V_1. For belief b_1, we observe two α-vectors: α_{11} and α_{14}. The difference between $b_1 \cdot \alpha_{11}$ and $b_1 \cdot \alpha_{14}$ is greater than the allowed slack η_1. Therefore, α_{14} is thrown out of Γ_1^t. Conversely, for belief b_2, the $b_2 \cdot \alpha_{12}$ and $b_2 \cdot \alpha_{13}$ are within η_i so Γ_1^t contains both. These two sets of α-vectors define the actions available to V_2. We make three observations about V_2. First, α_{24} corresponds to the *potential* α-vector that *would* have been available had we not removed α_{14}'s action. Its actual value was higher, but the first value function restricted the set of available actions for the second value function. Second, α_{13}'s action inclusion for the actions available at b_2 in V_2 enabled it to obtain a *higher* value (with α_{23}) than it would have if we had only allowed the maximal action to be taken (with α_{22}). Third, the infinite horizon values of $V_1'(b)$ decreased slightly to $V_1(b)$ because we allowed for slack in order to *greatly improve* the second value function from $V_2'(b)$ to $V_2(b)$.

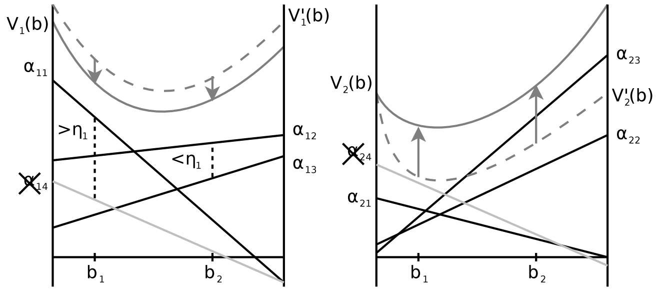

Figure 1: Examples of action restriction and slack affordance.

3 Solving LPOMDPs

We will describe two general methods of solving LPOMDPs: Lexicographic Value Iteration and Point-Based Algorithms.

3.1 Lexicographic Value Iteration

Fortunately, the original formulations of value iteration for POMDPs [Smallwood and Sondik, 1973; Sondik, 1978; Kaelbling *et al.*, 1998] can be directly extended to LPOMDPs. For all $i \in K$, we may write an equivalent equation to Equation 2 following the stochastic process. This result follows exactly as Sondik's formulation.

$$Q_i^t(b, a) = b \cdot r_{ia} + \sum_{\omega \in \Omega} \max_{\alpha \in \Gamma_i^{t-1}} \sum_{s \in S} b(s) V_i^t(s, a, \omega, \alpha)$$

$$V_i^t(s, a, \omega, \alpha) = \gamma \sum_{s' \in S} O(a, s', \omega) T(s, a, s') \alpha(s') \tag{6}$$

Similarly, we define the set of α-vectors, Γ_i^t, using a *Minkowski sum* (cross-sum) operator $\oplus$.[1] We define sets Γ_{ia*}^t, $\Gamma_{ia\omega}^t$, for all $a \in A_i(b)$ and $\omega \in \Omega$:

$$\begin{aligned} \Gamma_{ia*}^t &= \{r_{ia}\} = \{[R_i(s_1, a), \dots, R_i(s_n, a)]^T\} \\ \Gamma_{ia\omega}^t &= \{[V_i^t(s_1, a, \omega, \alpha), \dots, V_i^t(s_n, a, \omega, \alpha)]^T | \alpha \in \Gamma_i^{t-1}\} \end{aligned}$$

Finally, we are able to define Γ^t:

$$\Gamma_{ia}^t = \Gamma_{ia*}^t \oplus \Gamma_{ia\omega_1}^t \oplus \cdots \oplus \Gamma_{ia\omega_z}^t \tag{7}$$

$$\Gamma_i^t = \bigcup_{a \in A} \Gamma_{ia}^t \tag{8}$$

This form constructs all possible combinations of action selections and resulting α-vectors following the policy tree.

We always select our initial vector α's values to be $\alpha(s) = R_i^{min}/(1 - \gamma)$ for all $s \in S$; this guarantees that α-vectors form a lower bound, remaining the same or improving at each time step [Lovejoy, 1991; Pineau *et al.*, 2006]. Additionally, as others have shown, for each $i \in K$ we may assign horizon T to ensure we are within $\epsilon > 0$ such that T satisfies: $\gamma^T (R_i^{max} - R_i^{min}) < \epsilon$, with $R_i^{max} = \max_{s \in S} \max_{a \in A} R_i(s, a)$ and $R_i^{min} = \min_{s \in S} \min_{a \in A} R_i(s, a)$.

[1] For sets $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_m\}$, $A \oplus B = \{a_1 + b_1, a_1 + b_2, \dots, a_1 + b_m, a_2 + b_1, \dots, a_n + b_m\}$

3.2 Point-Based Variants

Due to the complexity of exact value iteration in POMDPs, point-based methods have been developed, which perform Bellman updates to a set of α-vectors for a particular set of belief points. Algorithms differ in their selection, addition, and update frequency of belief points: Point-Based Value Iteration (PBVI) [Pineau *et al.*, 2006], Heuristic Search Value Iteration (HSVI2) [Smith and Simmons, 2005], Perseus [Spaan and Vlassis, 2005], and SARSOP [Kurniawati *et al.*, 2008]. At their core, however, they all exploit a new Bellman update that only operates over belief points (Equations 9 and 10).

$$\Gamma_{ib}^{t} = \{r_{ia} + \sum_{\omega \in \Omega} \operatorname*{argmax}_{\alpha \in \Gamma_{ia\omega}^{t}} \alpha \cdot b | \forall a \in A_i(b)\}, \quad \forall b \in B \tag{9}$$

$$\Gamma_{i}^{t} = \{\operatorname*{argmax}_{\alpha \in \Gamma_{ib}^{t}} \alpha \cdot b | \forall b \in B\} \tag{10}$$

4 Theoretical Analysis

We will now provide a series of theoretical results for LPOMDPs, LVI, and LPBVI. First, we show that LVI converges to a unique fixed point in the limit in Proposition 1. Propositions 2 and 3 give a bound on slack for LVI and LPBVI, respectively. Additionally, they provide a correctness result for both, which guarantees that the returned policy is within the allotted slack.

Proposition 1 (Convergence). Lexicographic Value Iteration (LVI) for LPOMDPs converges to a unique fixed point.

Proof. For all $i \in K$, in order, we have a fixed set of actions for each belief point $b \in B$, restricted from the previous value function $i - 1$. Therefore, value iteration converges to a unique fixed point for V_i [Smallwood and Sondik, 1973; Sondik, 1978; Wray *et al.*, 2015]. Since this is true for all $i \in K$, once $i = k$, it converges to a unique fixed point, returning a final policy π. Thus, LVI converges to a unique fixed point $\mathbf{V}^{\pi}$. □

With convergence established, we now are able to state a strong slack guarantee in Proposition 2. The proposition guarantees that LVI returns a policy that satisfies the slack constraints. The proof is similar to the one for LMDPs [Wray *et al.*, 2015], with a few adjustments for the belief-state MDPs induced by each value function in an LPOMDP.

Proposition 2 (LVI Slack Bound; Correctness). For $i \in K$, assume $1, \ldots i - 1$ have converged. Let V_i^{η} be the value functions returned by value iteration. Let V_i^{π} be the value function returned by the policy π returned after running LVI over all K. If $\eta_i = (1 - \gamma)\delta_i$, then $\forall b \in B$, $V_i^{\eta}(b) - V_i^{\pi}(b) \leqslant \delta_i$.

Proof. First, let $\rho_i : B \times A \to \mathbb{R}$ and $\tau_i : B \times A \times B \to [0, 1]$ denote the reward and state transition for the induced belief state MDP, respectively [Kaelbling *et al.*, 1998]. Let us examine the full expansion of t time steps ($t \to \infty$) of Bellman's equation for $V_i^{\pi}(b^t)$.

$$V_i^{\pi}(b^t) = \rho(b^t, \pi(b^t)) + \gamma \int_{b'} \tau(b^t, \pi(b^t), b') V_i^{\pi}(b') \mathrm{d}b'$$

$$= \rho(b^t, \pi(b^t)) + \gamma \int_{b^{t-1}} \tau(b^t, \pi(b^t), b^{t-1}) \Big(\cdots \Big(\rho(b^1, \pi(b^1))$$

$$+ \gamma \int_{b^0} \tau(b^1, \pi(b^1), b^0) V_i^0(b^0) \mathrm{d}b^0 \Big) \cdots \Big) \mathrm{d}b^{t-1}$$

By Proposition 1 there is exactly one unique fixed point, so the initial value V_i^0 does not matter. Following the policy, we assign $V_i^0(b^0) = V_i^{\eta}(b^0)$. In doing so, we recognize the existence of $Q_i^{\eta}(b^1, \pi(b^1))$ (the $\rho(\cdot)$ plus the integral over b^0). By Equation 4, $V_i^{\eta}(b^1) - Q_i^{\eta}(b^1, \pi(b^1)) \leqslant \eta_i$, since $\pi(b^1) \in A_k(b^1) \subseteq \cdots \subseteq A_{i+1}(b^1)$. Rewriting this, we obtain $Q_i^{\eta}(b^1, \pi(b^1)) \geqslant V_i^{\eta}(b^1) - \eta_i$. Applying this equation, recognizing that $\gamma\eta_i$ is constant, and $\int_{b^0} \tau(b^1, \pi(b^1), b^0)\mathrm{d}b^0 = 1$, we get:

$$\geqslant \rho(b^t, \pi(b^t)) + \gamma \int_{b^{t-1}} \tau(b^t, \pi(b^t), b^{t-1}) \Big(\cdots \Big(\rho(b^1, \pi(b^1))$$

$$+ \gamma \int_{b^1} \tau(b^2, \pi(b^2), b^1) V_i^{\eta}(b^1) \mathrm{d}b^1 - \gamma\eta_i \Big) \cdots \Big) \mathrm{d}b^{t-1}$$

We apply this process for all $t \to \infty$ (let $b^{\infty} = b$), noting that each time an extra $\gamma\eta_i$ falls out, and the previous ones gain an extra product of γ. We also may subtract an extra $\eta_i \geqslant 0$ to obtain a geometric series.

$$V_i^{\pi}(b) \geqslant V_i^{\eta}(b) - \sum_{t=0}^{\infty} \gamma^t \eta_i \geqslant V_i^{\eta}(b) - \frac{\eta_i}{1 - \gamma}$$

$$V_i^{\eta}(b) - V_i^{\pi}(b) \leqslant \frac{\eta_i}{1 - \gamma}$$

Assign $\eta_i = (1 - \gamma)\delta_i$ to obtain our desired inequality. □

LVI is guaranteed to converge to a unique fixed point within the afforded slack; now we may analyze LPBVI. Like PBVI, LPBVI introduces an error from the approximation, i.e., using belief points B instead of $\triangle^n$. This affects the set of actions available to the next value function, since some actions may have been included which should not have been. We can, however, account for this in our selection of η in order to guarantee all actions passed to the next value function are valid; Proposition 3 formally states this bound.

Let the density of belief points δ_B be defined in the same manner as Pineau *et al.* [2006], albeit with slightly overloaded notation: $\delta_B = \max_{b' \in \triangle^n} \min_{b \in B} \|b - b'\|_1$. For $i \in K$, assume $1, \ldots, i - 1$ has converged. Let the maximal (worst-case) error after horizon t be ϵ_i^t.

Proposition 3 (LPBVI Slack Bound; Correctness). For $i \in K$, assume $1, \ldots, i-1$ have converged. Let $V_i^{\eta B}$ be the value function returned by PBVI for belief set B. Let $V_i^{\pi B}$ be the value function returned by the policy π returned after running LPBVI over all K. If

$$\eta_i = \max\Big\{0, (1 - \gamma)\delta_i - \frac{R_i^{max} - R_i^{min}}{1 - \gamma}\delta_B\Big\} \tag{11}$$

then $\forall b \in B$, $V_i^{\eta B}(b) - V_i^{\pi B}(b) \leqslant \delta_i$.

Proof. Let V_i^{η} and V_i^{π} denote the true values following value iteration and LVI, respectively, as defined in Proposition 2. We know that $\forall b \in B$:

$$V_i^{\eta B}(b) \leqslant V_i^{\eta}(b) \quad \text{and} \quad V_i^{\eta B}(b) \geqslant V_i^{\pi B}(b).$$

For all $b \in B$, we begin with $V_i^{\eta B}(b) - V_i^{\pi B}(b)$:

$$\begin{aligned}
&V_i^{\eta B}(b) - V_i^{\pi B}(b) && \\
&\leqslant V_i^{\eta}(b) - V_i^{\pi B}(b) && \text{by construction} \\
&\leqslant V_i^{\eta}(b) - (V_i^{\pi}(b) - \epsilon_i^t) && \text{worst-case upper bound} \\
&\leqslant (V_i^{\eta}(b) - V_i^{\pi}(b)) + \epsilon_i^t && \text{rearrange} \\
&\leqslant \frac{\eta_i}{1-\gamma} + \epsilon_i^t && \text{by Proposition 2} \\
&\leqslant \frac{\eta_i}{1-\gamma} + \frac{R_i^{max} - R_i^{min}}{(1-\gamma)^2}\delta_B && \text{by Pineau } \textit{et al.} \text{ [2006]}
\end{aligned}$$

The worse-case upper bound value comes from the fact that $V_i^{\pi B}(b) \in [V_i^{\pi}(b) - \epsilon_i^t, V_i^{\pi}(b)]$. In order to guarantee exactly δ_i slack, we solve for η_i. We also select the maximum of 0 and this value in order to satisfy the constraint:

$$V_i^{\eta B}(b) - V_i^{\pi B}(b) \geqslant 0.$$

$$\delta_i = \frac{\eta_i}{1-\gamma} + \frac{R_i^{max} - R_i^{min}}{(1-\gamma)^2}\delta_B$$

$$\eta_i = \max\left\{0, (1-\gamma)\delta_i - \frac{R_i^{max} - R_i^{min}}{1-\gamma}\delta_B\right\} \qquad \square$$

Interestingly, this bound has the desired property that as we improve the density of our belief points (i.e., $B \rightarrow \triangle^n$ and $\delta_B \rightarrow 0$), the acceptable value of each η_i converges to the result from Proposition 2. Additionally, the bound makes intuitive sense: η_i describes the acceptable slack for one iteration's action's deviation from optimal. It turns out that the adjustment of $\frac{R_i^{max} - R_i^{min}}{1-\gamma}\delta_B$ is *exactly* the definition of PBVI's one-step error [Pineau *et al.*, 2006], which obviously must be accounted for in the tolerable amount of one-step slack η_i.

Equation 11 raises one issue: We cannot compute δ_B without solving a linear program. Therefore, we will use an approximation $\hat{\delta}_B = \max_{b' \in B} \min_{b \in B} \|b - b'\|_1$ to create a weaker slack variable $\hat{\eta}_i$. It is easy to show that the approximation we will use is an upper bound: $\eta_i \leqslant \hat{\eta}_i$ because $\hat{\delta}_i \leqslant \delta_i$. The density of belief points $\hat{\delta}_B$ will be updated upon every expansion of B and weakly monotonically decrease.

5 Experimentation

Semi-autonomous systems require collaboration between a human and an agent in order to achieve a goal [Cohn *et al.*, 2011; Zilberstein, 2015]. We experiment within one such domain–semi-autonomous driving–in which a car may only drive autonomously on some subset of *autonomy-capable* roads, requiring manual driving on the remaining roads. In our scenario the driver may be *attentive* or *tired*, extending the original problem formulation by Wray *et al.* [2015].

States are defined as all pairs of intersections, which implicitly represent location and direction, as well as two Boolean values, indicating whether the driver is tired and if autonomy is enabled. All actions are therefore taken at intersections and simply refer to which road to take next. Additionally, autonomy can be enabled or disabled at an intersection; however, the car can only drive autonomously on major roads, not minor back roads. Following the model introduced by Wray *et al.*, we use a real-world dataset and allow for autonomy whenever the speed limit is greater than 30 miles per hour. State transitions capture the likelihood (0.1 probability) that the human driver will drift from attentive to tired.

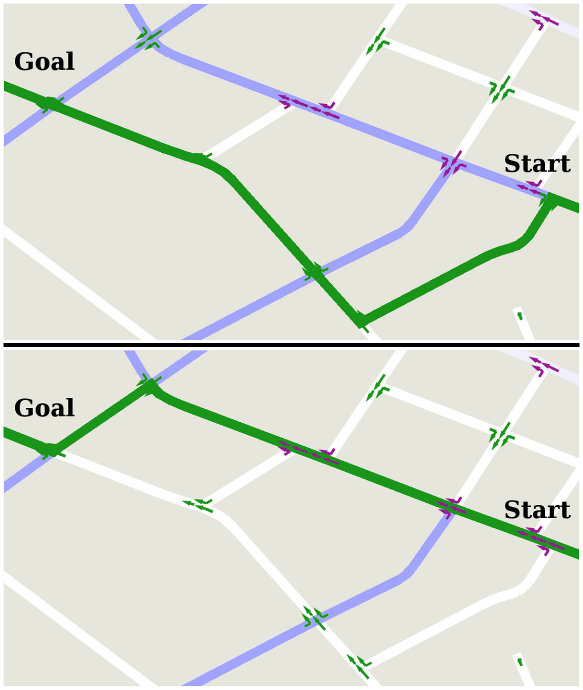

Figure 2: An example policy for Boston with driver tiredness belief probability of 0.2 (top) and 0.8 (bottom).

An LPOMDP differs from an LMDP in that the true state of the system is unobservable. We do assume that the car's location and autonomy indicator are perfectly observable, and include noisy observations regarding the driver's state. Thus, observations simply denote if the driver is attentive or tired. The observation model captures the inaccuracy of the system monitoring human attentiveness, e.g., an eye tracking system [Pradhan *et al.*, 2005]. In the experiments, we use a 0.75 probability for accurately determining the true state of the driver, with a 0.25 error rate.

We include two reward functions: *time* and *autonomy*. The first seeks to reduce time spent to reach the goal, i.e., shortest path. More concretely, it penalizes the agent for the time spent on each road (distance divided by speed in seconds). The second function only penalizes the agent if: (1) autonomy was not enabled while the driver was tired, or (2) the agent did not enable autonomy if it had the option to do so. At the goal state, both reward functions provide zero penalty.

Figure 2 demonstrates an example policy for a section of Boston. At each intersection, the action is shown with arrows. The blue roads are autonomy-capable and white roads

City	$\lvert S\rvert$	$\lvert A\rvert$	$\lvert\Omega\rvert$	$\lvert B\rvert$	$V_1^\eta(b^0)$	$V_2^\eta(b^0)$	CPU ($h = 10$)	GPU ($h = 500$)	Improvement
Austin	92	8	2	230	57.4	35.9	14.796	3.798	× 194.79
San Franc.	172	8	2	430	97.8	53.8	51.641	8.056	× 320.51
Denver	176	8	2	440	123.7	77.3	60.217	8.299	× 362.80
Baltimore	220	8	2	550	56.2	43.9	104.031	11.782	× 441.48
Pittsburgh	268	10	2	670	148.0	142.2	169.041	19.455	× 434.44
L.A.	380	8	2	950	167.9	114.4	298.794	25.535	× 585.07
Chicago	404	10	2	1010	67.4	31.6	399.395	36.843	× 542.02
Seattle	432	10	2	1080	111.2	66.9	497.061	48.204	× 515.58
N.Y.C.	1064	12	2	2660	108.1	73.7	n/a	351.288	n/a
Boston	2228	12	2	5570	109.3	79.2	n/a	2424.961	n/a

Table 1: Computation time (seconds), and the initial belief's values (negated travel time; seconds) over 10 cities for LPBVI on the CPU ($h = 10$) and GPU ($h = 500$) and the improvement ratio: $(50 * CPU)/GPU$ adjusted for the horizon difference.

are not. Since values and actions exist within an impossible-to-visualize high-dimensional belief space, we simply took two "slices" over this space. We assumed the physical state and autonomy were known, assigning a probability of 0.2 to being attentive and 0.8 to being tired.

5.1 GPU Optimization

To improve the scalability of (L)PBVI and other point-based algorithms, we develop a GPU-based parallel variant for (L)POMDPs.[2] We implemented it within CUDA[3] to demonstrate that CUDA provides a *vast performance improvement* over CPU-based versions. This is a relatively unexplored, fertile area of research. GPU-based MDP solvers have been developed in the past [Jóhannsson, 2009; Wray *et al.*, 2015], as well as a Monte Carlo Value Iteration GPU solver for continuous state POMDPs [Lee and Kim, 2013]. Crook *et al.* [2011] parallelized only the belief update equation within a spoken dialog system domain. To the best of our knowledge, no one has explored our parallel approach for point-based solvers that leverages a GPU's parallel computational capabilities.

To start, we divide the update step of (L)PBVI at a horizon into two subproblems, each executed as a *kernel* (i.e., a program which runs on the GPU). Equation 10 requires a maximization and a summation over states (dot product). This will form our first kernel, which computes the maximal α-vectors over each belief separately in parallel with a simple reduction. In order to do this, however, we must compute the α-vectors within Γ_{ib}^t. This will be our second kernel; it is computed by Equation 9, which contains a summation over observations, a maximization over α-vectors (the size of belief space B), and a summation over states (dot product). Furthermore, $\Gamma_{ia\omega}^t$ must be computed from Equation 6, which is another summation over states. This is done in parallel over beliefs, actions, and observations. The result is cached for use in the first kernel.

Table 1 shows performance over 10 cities, as well as the values of both objective functions at the initial belief point, which has uniform belief over the fatigue of the driver. The improvement ratio of $(50{*}CPU)/GPU$ is shown, too. These are point-based algorithms without expansion, so each horizon step is the same operation. For that reason, the improvement multiplies the CPU time by 50 as it accounts for the horizon difference, shifting them to the same scale.

Experiments were conducted with an Intel(R) Core(TM) i7-4702HQ CPU at 2.20GHz, 8GB of RAM, and an Nvidia(R) GeForce GTX 870M graphics card using C++ and CUDA(C) 6.5. The results demonstrate that the GPU implementation produces a speedup of more than two orders of magnitude over the CPU implementation. Note, however, that our CPU implementation used STL objects, whereas the GPU used arrays. While these results are preliminary and are specific to our implementations, the trend is evident from our initial experimentation in parallelizing (L)PBVI.

6 Conclusion

We present a model for a lexicographic multi-objective POMDP with slack entitled LPOMDP, which naturally captures many real-world problem domains with multiple, prioritized objectives. In fact, it is our novel definition of slack that enables the use of lexicographic orderings in MOPOMDPs. It allows for higher-preference objective functions to slightly deviate from optimal, up to the allotted slack, in order to hopefully improve lower-preference objective functions.

We present two algorithms that solve this problem: LVI, which extends value iteration to LPOMDPs, and LPBVI, which to the best of our knowledge is the first multi-objective point-based algorithm for MOPOMDPs. We prove convergence, as well as bounds on slack for both, thereby proving their correctness. To illustrate our model's applicability to real-world problems, we examine its performance within the recently proposed semi-autonomous driving domain. Our experiments show that our algorithms can solve practical LPOMDPs with large state-spaces–in line with the capabilities of state-of-the-art POMDP solvers. Furthermore, we present a GPU-based version of point-based solvers that reduces runtime by orders of magnitude, and is immediately applicable to a range of existing POMDP solvers.

In future work, we will expand our investigation of LPOMDPs and their applications. Additionally, we plan to make our domain-specific tools public, in addition to both our CPU and GPU source code, to facilitate community-wide development of realistic-scale domains and algorithms for planning under partial observability.

[2] Trivially, this also works perfectly with point-based methods for POMDPs, yielding the same kind of performance improvements.

[3] https://developer.nvidia.com/about-cuda

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This research was supported in part by NSF grant number IIS-1405550.

References

[Aissani *et al.*, 2009] Nassima Aissani, Bouziane Beldjilali, and Damien Trentesaux. Dynamic scheduling of maintenance tasks in the petroleum industry: A reinforcement approach. *Engineering Applications of Artificial Intelligence*, 22(7):1089–1103, 2009.

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

[Cohn *et al.*, 2011] Robert Cohn, Edmund Durfee, and Satinder Singh. Comparing action-query strategies in semi-autonomous agents. In *Proc. of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1287–1288, 2011.

[Crook *et al.*, 2011] Paul A. Crook, Brieuc Roblin, Hans-Wolfgang Loidl, and Oliver Lemon. Parallel computing and practical constraints when applying the standard POMDP belief update formalism to spoken dialogue management. In *Proc. of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop*, pages 189–201. Springer, 2011.

[Gábor *et al.*, 1998] Zoltán Gábor, Zsolt Kalmár, and Csaba Szepesvári. Multi-criteria reinforcement learning. In *Proc. of the 15th International Conference on Machine Learning (ICML)*, pages 197–205, 1998.

[Isom *et al.*, 2008] Joshua D. Isom, Sean P. Meyn, and Richard D. Braatz. Piecewise linear dynamic programming for constrained POMDPs. In *Proc. of the 23rd International Conference on Artificial Intelligence (AAAI)*, pages 291–296, 2008.

[Jóhannsson, 2009] Ársæll Þór Jóhannsson. GPU-based Markov decision process solver. Master's thesis, School of Computer Science, Reykjavík University, 2009.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Journal of Artificial Intelligence Research*, 101(1):99–134, 1998.

[Kurniawati *et al.*, 2008] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. of Robotics: Science and Systems*, pages 65–72, 2008.

[Kwak *et al.*, 2012] 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 *Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 21–28, 2012.

[Lee and Kim, 2013] Taekhee Lee and Young J. Kim. GPU-based motion planning under uncertainties using POMDP. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4576–4581, 2013.

[Lovejoy, 1991] William S. Lovejoy. Computationally feasible bounds for partially observed Markov decision processes. *Operations Research*, 39(1):162–175, 1991.

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

[Pineau *et al.*, 2006] Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

[Pradhan *et al.*, 2005] Anuj Kumar Pradhan, Kim R. Hammel, Rosa DeRamus, Alexander Pollatsek, David A Noyce, and Donald L. Fisher. Using eye movements to evaluate effects of driver age on risk perception in a driving simulator. *Human Factors*, 47(4):840–852, 2005.

[Rangcheng *et al.*, 2001] Jia Rangcheng, Ding Yuanyao, and Tang Shaoxiang. The discounted multi-objective Markov decision model with incomplete state observations: lexicographically order criteria. *Mathematical Methods of Operations Research*, 54(3):439–443, 2001.

[Roijers *et al.*, 2013] Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[Smallwood and Sondik, 1973] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[Smith and Simmons, 2005] Trey Smith and Reid Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 542–547, 2005.

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

[Soh and Demiris, 2011a] Harold Soh and Yiannis Demiris. Evolving policies for multi-reward partially observable Markov decision processes (MR-POMDPs). In *Proc. of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 713–720, 2011.

[Soh and Demiris, 2011b] Harold Soh and Yiannis Demiris. Multi-reward policies for medical applications: Anthrax attacks and smart wheelchairs. In *Proc. of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO)*, pages 471–478, 2011.

[Sondik, 1978] Edward J. Sondik. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.

[Spaan and Vlassis, 2005] Matthijs T. J. Spaan and Nikos A. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

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

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