Lexicographic Multi-Objective Reinforcement Learning

Joar Skalse*, Lewis Hammond, Charlie Griffin and Alessandro Abate
Department of Computer Science, University of Oxford
{joar.skalse, lewis.hammond, charlie.griffin, aabate}@cs.ox.ac.uk

Abstract

In this work we introduce reinforcement learning techniques for solving lexicographic multi-objective problems. These are problems that involve multiple reward signals, and where the goal is to learn a policy that maximises the first reward signal, and subject to this constraint also maximises the second reward signal, and so on. We present a family of both action-value and policy gradient algorithms that can be used to solve such problems, and prove that they converge to policies that are lexicographically optimal. We evaluate the scalability and performance of these algorithms empirically, demonstrating their practical applicability. As a more specific application, we show how our algorithms can be used to impose safety constraints on the behaviour of an agent, and compare their performance in this context with that of other constrained reinforcement learning algorithms.

1 Introduction

Reinforcement learning (RL) algorithms learn to solve tasks in unknown environments by a process of trial and error, where the task typically is encoded as a scalar reward function. However, there are tasks for which it is difficult (or even infeasible) to create such a function. Consider, for example, Isaac Asimov’s three Laws of Robotics – the task of following these laws involves multiple (possibly conflicting) objectives, some of which are lexicographically (i.e. categorically) more important than others. There is, in general, no straightforward way to write a scalar reward function that encodes such a task without ever incentivising the agent to prioritise less important objectives. In such cases, it is difficult (and often unsuitable) to apply standard RL algorithms.

In this work, we introduce several RL techniques for solving lexicographic multi-objective problems. More precisely, we present both a family of action-value algorithms and a family of policy gradient algorithms that can accept multiple reward functions $R_1, \ldots, R_m$, and that learn a policy $\pi$ such that $\pi$ maximises expected discounted $R_1$-reward, and among all policies that do so, $\pi$ also maximises expected discounted $R_2$-reward, and so on. These techniques can easily be combined with a wide range of existing RL algorithms. We also prove the convergence of our algorithms, and benchmark them against state-of-the-art methods for constrained reinforcement learning in a number of environments.

1.1 Related Work

Lexicographic optimisation in Multi-Objective RL (MORL) has previously been studied by [Gábor et al., 1998], whose algorithm is a special case of one of ours (cf. Footnote 3). Our contribution extends this work to general, state-of-the-art RL algorithms. Unlike [Gábor et al., 1998], we also prove that our algorithms converge to the desired policies, and provide benchmarks against other state-of-the-art algorithms in more complex environments. Other MORL algorithms combine and trade off rewards in different ways; for an overview, see [Roijers et al., 2013; Liu et al., 2015].

Lexicographic optimisation more generally is a long-studied problem – see, e.g. [Mitten, 1974; Rentmeesters et al., 1996; Wray and Zilberstein, 2015].

A natural application of lexicographic RL (LRL) is to learn a policy that maximises a performance metric, subject to satisfying a safety constraint. This setup has been tackled with dynamic programming in [Lesser and Abate, 2018], and has also been studied within RL. For example, [Tessler et al., 2019] introduce an algorithm that maximises a reward subject to the constraint that the expectation of an additional penalty signal should stay below a certain threshold; [Chow et al., 2017] introduce techniques to maximise a reward subject to constraints on the value-at-risk (VaR), or the conditional value-at-risk (CVaR), of a penalty signal; and [Miryoosefi et al., 2019] discuss an algorithm that accepts an arbitrary number of reward signals, and learns a policy whose expected discounted reward vector lies inside a given convex set.

Our contributions add to this literature and, unlike the methods above, allow us to encode safety constraints in a principled way without prior knowledge of the level of safety that can be attained in the environment. Note that lexicographic optimisation of two rewards is qualitatively different from maximising one reward subject to a constraint on the second, and thus the limit policies of LRL and the algorithms above will not, in general, be the same. Other methods also emphasise staying safe while learning; see e.g. [Achiam et
2 Background

Reinforcement Learning. The RL setting is usually formalised as a Markov Decision Process (MDP), which is a tuple \((S, A, T, R, \gamma)\) where \(S\) is a set of states, \(A\) is a set of actions, \(T : S \times A \rightarrow S\) is a transition function, \(R : S \times A \times S \rightarrow \mathbb{R}\) a reward function, where \(R(s, a, s')\) is the reward obtained if the agent moves from state \(s\) to \(s'\) by taking action \(a\), and \(\gamma \in [0, 1]\) is a discount factor. Here, \(f : X \rightarrow Y\) denotes a probabilistic mapping \(f\) from \(X\) to \(Y\). A state is terminal if \(T(s, a) = s\) and \(R(s, a, s) = 0\) for all \(a\).

A (stationary) policy is a mapping \(\pi : S \rightarrow A\) that specifies a distribution over the agent’s actions in each state. The value function \(v_\pi(s)\) of \(\pi\) is defined as the expected \(\gamma\)-discounted cumulative reward following \(\pi\) from \(s\), i.e. \(v_\pi(s) := \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid s_0 = s \right]\). When \(\gamma = 1\), we instead consider the limit-average of this expectation.

The objective in RL can then be expressed as maximising \(J(\pi) := \sum_s v_\pi(s)\). Given a policy \(\pi\) we may also define the \(q\)-function \(q_\pi(s, a) := \mathbb{E}_{\pi'} \left[ R(s, a, s') + v_\pi(s') \right]\) and the advantage function \(a_\pi(s, a) := q_\pi(s, a) - v_\pi(s)\).

Value-Based Methods. A value-based agent has two main components: a \(Q\)-function \(Q : S \times A \rightarrow \mathbb{R}\) that predicts the expected future discounted reward conditional on taking a particular action in a particular state; and a \(b\)-function that is used to select actions in each state. The \(Q\)-function can be represented as a lookup table (in which case the agent is tabular), or as a function approximator.

There are many ways to update the \(Q\)-function. One popular rule is \(Q\)-Learning [Watkins, 1986]:

\[
Q(s_t, a_t) \leftarrow (1 - \alpha_t(s_t, a_t)) \cdot Q(s_t, a_t) + \alpha_t(s_t, a_t) \cdot \left( r_t + \gamma \max_a Q(s_{t+1}, a) \right),
\]

where \(t\) is the time step and \(\alpha_t(s_t, a_t)\) is a learning rate. One can replace the term \(\max_a Q(s_{t+1}, a)\) in the rule above with \(Q(s_{t+1}, a)\) or \(\mathbb{E}_{a \sim \pi(a)}[Q(s_{t+1}, a)]\) (where \(\pi\) is the policy that describes the agent’s current behaviour) to obtain the SARSA [Rummery and Niranjan, 1994] or the Expected SARSA [van Seijen et al., 2009] updates respectively.

Policy-Based Methods. In these methods, the policy \(\pi(\cdot; \theta)\) is differentiable with respect to some \(\theta \in \Theta \subset \mathbb{R}^2\), and \(\theta\) is updated according to an objective \(K(\theta)\). If using \(K^{\text{Actor-Critic}}(\theta) = J(\theta)\) then we may express this using:

\[
K^{\text{Actor-Critic}}(\theta) := \mathbb{E}_t \left[ \log \pi(a_t | s_t; \theta) \cdot A_\theta(s_t, a_t) \right],
\]

where \(A_\theta\) is an estimate of \(a_\theta\). One often computes \(A_\theta\) by approximating \(v_\theta\) with a function \(V\) parameterised by some \(w \in W \subset \mathbb{R}^2\), and using the fact that the expected temporal difference error \(\delta_t := r_t + \gamma v_\theta(s_{t+1}) - v_\theta(s_t)\) (or \(r_t + v_\theta(s_{t+1}) - v_\theta(s_t) - J(\theta)\) when \(\gamma = 1\)) equals \(a_\theta(s_t, a_t)\) [Bhatnagar et al., 2009]. Such algorithms are known as Actor-Critic (AC) algorithms [Konda and Tsitsiklis, 2000].

More recently, other policy gradient algorithms have used surrogate objective functions, which increase stability in training by penalising large steps in policy space, and can be viewed as approximating the natural policy gradient [Amari, 1998; Kakade, 2001]. One common such penalty is the Kullback–Leibler (KL) divergence between new and old policies, as employed in one version of Proximal Policy Optimisation (PPO) [Schulman et al., 2017], leading to:

\[
K^{\text{PPO}}(\theta) := \mathbb{E}_t \left[ \frac{\pi(a_t | s_t; \theta)}{\pi(a_t | s_t; \theta_{old})} A_\theta(s_t, a_t) \right],
\]

\[
-\lambda \cdot \text{D}_\text{KL}(\pi(s_t; \theta) \| \pi(s_t; \theta_{old})),
\]

where \(\lambda\) is a scalar weight. Such algorithms enjoy both state of the art performance and strong convergence guarantees [Hsu et al., 2020; Liu et al., 2019].

Multi-Objective Reinforcement Learning. MORL is concerned with policy synthesis under multiple objectives. This setting can be formalised as a multi-objective MDP (MOMDP), which is a tuple \((S, A, T, I, R, \gamma)\) that is defined analogously to an MDP, but where \(R : S \times A \times S \rightarrow \mathbb{R}^m\) returns a vector of \(m\) rewards, and \(\gamma \in [0, 1]^m\) defines \(m\) discount rates. We define \(R_i\) as \((s, a, s') \mapsto R(s, a, s_i)\).

3 Lexicographic Reinforcement Learning

In this section we present a family of value-based and policy-based algorithms that solve lexicographic multi-objective problems by learning a lexicographically optimal policy. Given a MOMDP \(M\) with \(m\) rewards, we say that a policy \(\pi\) is (lexicographically) \(\epsilon\)-optimal if \(\pi \in \Pi^m, \Pi^m = \{ \pi \in \Pi^m \mid R_i(\pi) \leq \epsilon_i \text{ for all } i \in [m] \}\), and \(\Pi^m\) is the set of all policies in \(M\).

We similarly write \(\Theta^m\) to define global lexicographic \(\epsilon\)-optima for parameterised policies.\footnote{Due to the choice of baseline [Sutton et al., 1999], we describe here the classic Advantage Actor-Critic (A2C) algorithm.}

3.1 Value-Based Algorithms

We begin by introducing bandit algorithms that take as input multiple \(Q\)-functions and converge to taking lexicographically optimal actions.

Definition 1 (Lexicographic Bandit Algorithm). Let \(S\) be a set of states, \(A\) a set of actions, \(Q_1, \ldots, Q_m : S \times A \rightarrow \mathbb{R}\) a sequence of \(Q\)-functions, and \(t \in \mathbb{N}\) a time parameter. A lexicographic bandit algorithm with tolerance \(\tau \in \mathbb{R}_{>0}\) is a function \(B : (S \times A \rightarrow \mathbb{R})^m \times S \times \mathbb{N} \rightarrow A\), such that

\[
\lim_{t \rightarrow \infty} \Pr(B(Q_1, \ldots, Q_m, s, t) = A) = 1,
\]

where \(\Delta_{s, \tau} = A\) and \(\Delta_{s, \tau+1} = \{a \in A \mid Q_i(s, a) \geq \max_{a' \in A} Q_i(s, a') - \tau\}\).
Intuitively, a lexicographic bandit algorithm will, in the limit, pick an action \(a\) such that \(a\) maximises \(Q_1\) (with tolerance \(\tau\)) and among all actions that do this, action \(a\) also maximises \(Q_2\) (with tolerance \(\tau\)), and so on. An example of a lexicographic bandit algorithm is given in Algorithm 1, where the exploration probabilities \(\epsilon_{s,a}\) should satisfy \(\lim_{t \to \infty} \epsilon_{s,a} = 0\) and \(\sum_{t=0}^{\infty} \epsilon_{s,a} = \infty\) for all \(s \in S\).

We can now introduce Algorithm 2 (VB-LRL), a value-based algorithm for lexicographic multi-objective RL. Here \(\tau\) is any lexicographic bandit algorithm. The rule for updating the \(Q\)-values (on line 6) can be varied. We call the following update rule Lexicographic \(Q\)-Learning:

\[
Q_i(s,a) \leftarrow (1 - \alpha_i(s,a)) \cdot Q_i(s,a) + \\
\alpha_i(s,a) \cdot (R_i(s,a,s') + \gamma_i \max_{a' \in A_i} Q_i(s',a'),
\]

where \(\Delta^i_{s,0} = A, \Delta^i_{s,t+1} := \{a \in \Delta^i_{s,t} \mid Q_i(s,a) \geq \max_{a' \in \Delta^i_{s,t}} Q_i(s,a') - \tau\}\), and \(\tau \in \mathbb{R}_{>0}\) is the tolerance parameter.\(^2\) This rule is analogous to \(Q\)-Learning, but where the max-operator is restricted to range only over actions that (approximately) lexicographically maximise all rewards of higher priority. We can also use SARSA or Expected SARSA. Alternatively, we can adapt Double \(Q\)-Learning [Hasselt, 2010] for VB-LRL. To do this, we let the agent maintain two \(Q\)-functions \(Q^A, Q^B\) for each reward. To update the \(Q\)-values, with probability 0.5 we set:

\[
Q^A_i(s,a) \leftarrow (1 - \alpha_i(s,a)) \cdot Q^A_i(s,a) + \\
\alpha_i(s,a) \cdot (R_i(s,a,s') + \gamma_i \cdot Q^A_i(s',\text{arg max}_{a' \in A_i} Q^A_i(s',a'))),
\]

and else perform the analogous update on \(Q^B_i\), and let \(Q_i(s,a) := 0.5(Q^A_i(s,a) + Q^B_i(s,a))\) in the bandit algorithm. Varying the bandit algorithm or \(Q\)-value update rule in VB-LRL produces a family of algorithms with different properties.\(^3\) We can now give our core result for Algorithm 2. All our proofs are included in the supplementary material.\(^4\)

**Theorem 1.** In any MOMDP \(\mathcal{M}\), if VB-LRL uses a lexicographic bandit algorithm and either SARSAs, Expected SARSA, or Lexicographic Q-Learning, then it will converge to a policy \(\pi\) that is lexicographically optimal if:

1. \(S\) and \(A\) are finite,
2. All reward functions are bounded,
3. Either \(\gamma_1, \ldots, \gamma_m < 1\), or every policy leads to a terminal state with probability one,
4. The learning rates \(\alpha_i(s,a) \in [0,1]\) satisfy the conditions \(\sum \alpha_i(s,a) = \infty\) and \(\sum \alpha_i(s,a)^2 < \infty\) with probability one, for all \(s \in S, a \in A\),
5. The tolerance \(\tau\) satisfies the condition that \(0 < \tau < \min_{i,s,a \neq a'} |q_i(s,a) - q_i(s,a')|\).

\(^2\)There are several places where VB-LRL makes use of a tolerance parameter \(\tau\). In the main text of this paper, we assume that the same tolerance parameter is used everywhere, and that it is a constant. In the supplementary material, we relax these assumptions.

\(^3\)The LRL algorithm in [Gabor et al., 1998] is equivalent to Algorithm 2 with Algorithm 1, Lexicographic Q-Learning, and \(\tau = 0\).

\(^4\)Available at https://github.com/lrhammond/lmorl.

---

### Algorithm 1 Lexicographic \(\epsilon\)-Greedy

**input:** \(Q_1, \ldots, Q_m, s, t\)

1. **with probability** \(\epsilon_{s,t}\) do \(a \sim \text{unif}(A)\)
2. **else**
3. \(\Delta \leftarrow A\)
4. **for** \(i \in \{1, \ldots, m\}\) **do**
5. \(x \leftarrow \max_{a' \in A} Q_i(s,a')\)
6. \(\Delta \leftarrow \{a \in A \mid Q_i(s,a) \geq x - \tau\}\)
7. \(a \sim \text{unif}(\Delta)\)
8. **return** \(a\)

### Algorithm 2 Value-Based Lexicographic RL

**input:** \(M = (S,A,T,I,\mathcal{R},\gamma)\)

1. initialise \(Q_1, \ldots, Q_m\)
2. **while** \(Q_1, \ldots, Q_m\) have not converged **do**
3. \(a \leftarrow B(Q_1, \ldots, Q_m, s, t)\) **\(\triangleright\)** Algorithm 1
4. \(s' \leftarrow T(s,a)\)
5. **for** \(i \in \{1, \ldots, m\}\) **do**
6. update \(Q_i\)
7. if \(s'\) is terminal then \(s \sim I\) **else** \(s \leftarrow s'\)
8. \(t \leftarrow t + 1\)
9. **return** \(\pi = s \rightarrow \lim_{t \to \infty} B(Q_1, \ldots, Q_m, s, t)\)

We also show that VB-LRL with Lexicographic Double \(Q\)-Learning converges to a lexicographically optimal policy.

**Theorem 2.** In any MOMDP, if VB-LRL uses Lexicographic Double \(Q\)-Learning then it converges to a lexicographically optimal policy \(\pi\) if conditions 1–5 in Theorem 1 hold.

Condition 4 requires that the agent takes every action in every state infinitely often. Condition 5 is quite strong – the upper bound on this range can in general not be determined a priori. However, we expect VB-LRL to be well-behaved as long as \(\tau\) is small. We motivate this intuition with a formal guarantee about the behaviour of VB-LRL for arbitrary \(\tau\).

**Proposition 1.** In any MOMDP, if VB-LRL has tolerance \(\tau > 0\), uses SARSAs, Expected SARSA, or Lexicographic Q-Learning, and conditions 1–4 in Theorem 1 are met, then:

1. \(J_1(\pi^*) - J_1(\pi_t) \leq \frac{\tau}{1 - \gamma} - \lambda_t\), for some sequence \(\{\lambda_t\}_{t \in \mathbb{N}}\) such that \(\lim_{t \to \infty} \lambda_t = 0\),
2. \(J_2(\pi^*) - J_2(\pi_t) \leq \frac{\tau}{1 - \gamma} - \eta_t\), for some sequence \(\{\eta_t\}_{t \in \mathbb{N}}\) such that \(\lim_{t \to \infty} \eta_t = 0\),

where \(\pi_t\) is the policy at time \(t\) and \(\pi^*\) is a lexicographically optimal policy.

Proposition 1 shows that we can obtain guarantees about the limit behaviour of VB-LRL without prior knowledge of the MOMDP when we only have two rewards, or are primarily interested in the two most prioritised rewards. We discuss this issue further in the supplementary material. Note also that while Algorithm 2 is tabular, it is straightforward to combine it with function approximators, making it applicable to high-dimensional state spaces.
3.2 Policy-Based Algorithms

We next introduce a family of lexicographic policy gradient algorithms. These algorithms use one objective function \( K_i \) for each reward function, and update the parameters of \( \pi(\cdot; \theta) \) with a multi-timescale approach whereby we first optimise \( \theta \) using \( K_1 \), then at a slower timescale optimise \( \theta \) using \( K_2 \) while adding the condition that the loss with respect to \( K_1 \) remains bounded by its current value, and so on. To solve these problems we use the well-known Lagrangian relaxation technique [Bertsekas, 1999].

Suppose that we have already optimised \( \theta' \) lexicographically with respect to \( K_1, \ldots, K_i-1 \) and we wish to now lexicographically optimise \( \theta \) with respect to \( K_i \). Let \( k_j := K_j(\theta') \) for each \( j \in \{1, \ldots, i \} \). Then we wish to solve the constrained optimisation problem given by:

\[
\text{maximise } K_i(\theta), \\
\text{subject to } K_j(\theta) \geq k_j - \tau, \quad \forall j \in \{1, \ldots, i \},
\]

where \( \tau > 0 \) is a small constant tolerance parameter, included such that there exists some \( \tau \) strictly satisfying the above constraints; in practice, while learning we set \( \tau = \tau_t \) to decay as \( t \to \infty \). This constraint qualification (Slater’s condition [Slater, 1950]) ensures that we may instead solve the dual of the problem by computing a saddle point \( \min_{\lambda > 0} \max_\theta L_i(\theta, \lambda) \) of the Lagrangian relaxation [Bertsekas, 1999] where:

\[
L_i(\theta, \lambda) := K_i(\theta) + \sum_{j=1}^{i-1} \lambda_j (K_j(\theta) - k_j + \tau).
\]

A natural approach would be to solve each optimisation problem for \( L_i \), where \( i \in \{1, \ldots, m\} \), in turn. While this would lead to a correct solution, when the space of lexicographic optima for each objective function is large or diverse, this process may end up being slow and sample-inefficient. Our key observation here is that by instead updating \( \theta \) at different timescales, we can solve this problem synchronously, guaranteeing that we converge to a lexicographically optimal solution as if done step-by-step under fixed constraints.

We set the learning rate \( \eta \) of the Lagrange multiplier to \( \eta^i \) after convergence with respect to the \( i^{th} \) objective, and assume that for all learning rates \( \alpha \in \{\alpha, \alpha^1, \ldots, \alpha^m\} \) and all \( i \in \{1, \ldots, m\} \) we have:

\[
t \in [0, 1], \sum_{t=0}^{\infty} \alpha^i t = \infty, \sum_{t=0}^{\infty} (\alpha^i t)^2 < \infty \quad \text{and} \\
\lim_{t \to \infty} \frac{\alpha^i}{\alpha^i t} = \lim_{t \to \infty} \frac{\eta^i}{\eta^i t} = 0.
\]

We also assume that \( \tau_t = o(\beta_t^m) \) in order to make sure that Slater’s condition holds in the limit with respect to all learning rates. Using learning rates \( \beta^i \) and \( \eta = \eta^i \) we may compute a saddle point solution to each Lagrangian \( L_i \) via the following (estimated) gradient-based updates:

\[
\theta \leftarrow \Gamma_0 \left[ \theta + \beta^i_0 \left( \nabla_\theta K_i(\theta) + \sum_{j=1}^{i-1} \lambda_j \nabla_\theta K_j(\theta) \right) \right], \\
\lambda_j \leftarrow \Gamma_\lambda \left[ \lambda_j + \eta \left( k_j - \tau_t - K_j(\theta) \right) \right] \forall j \in \{1, \ldots, i \}.
\]

Finally, in many policy-based algorithms we use a critic \( V_i \) (or \( Q_i \)) to estimate each \( K_i \) and so must also update the parameters \( w_i \) of each critic. This is typically done on a faster timescale using the learning rate \( \alpha \), for instance via the TD(0) update for \( V_i \) given by \( w_i \leftarrow w_i + \alpha \delta^i_t \nabla w_i V_i \), where \( \delta^i_t \) is the TD error for \( V_i \) at time \( t \) [Sutton and Barto, 2018]. A general scheme for policy-based LRL (PB-LRL) is shown in Algorithm 3, which may be instantiated with a wide range of objective functions \( K_i \) and update rules for each \( w_i \).

Below, we show that PB-LRL inherits the convergence guarantees of whatever (non-lexicographic) algorithm corresponds to the objective function used. Note that when \( m = 1 \), PB-LRL reduces to whichever algorithm is defined by the choice of objective function, such as A2C when using \( K_1^{A2C} \), or PPO when using \( K_1^{PPO} \). By using a standard stochastic approximation argument [Borkar, 2008] and proceeding by induction, we prove that any such algorithm that obtains a local (or global) \( \epsilon \)-optimum when \( m = 1 \) obtains a lexicographically local (or global) \( \epsilon \)-optimum when the corresponding objective function is used in PB-LRL.

Theorem 3. Let \( M \) be a MOMDP, \( \pi \) a policy that is twice continuously differentiable in its parameters \( \theta \), and assume that the same form of objective function is chosen for each \( K_i \) and that each reward function \( R_i \) is bounded. If using a critic, let \( V_i \) (or \( Q_i \)) be (action-value) functions that are continuously differentiable in \( w_i \) for \( i \in \{1, \ldots, m\} \) and suppose that if PB-LRL is run for \( T \) steps there exists some limit point \( w^*_i(\theta) = \lim_{T \to \infty} \mathbb{E}_t[w_i] \) for each \( w_i \) when \( \theta \) is held
fixed under some set of conditions $\mathcal{C}$ on $\mathcal{M}$, $\pi$, and each $V_i$. If $\lim_{T \to \infty} E_i[\theta] \in \Theta^*_m$ (respectively $\Theta^*_m$) under conditions $\mathcal{C}$ when $m = 1$, then for any fixed $m \in \mathbb{N}$ we have that $\lim_{T \to \infty} E_i[\theta] \in \Theta^*_m$ (respectively $\Theta^*_m$), where each $\epsilon_i \geq 0$ is a constant that depends on the representational power of the parametrisations of $\pi$ (and $V_i$ or $Q_i$, if using a critic).

In the remainder of the paper, we consider two particular variants of Algorithm 3, in which we use $K^{\text{A2C}}_\infty$ and $K^{\text{PPO}}_\infty$ respectively, for each $i$. We refer to the first as Lexicographic A2C (LA2C) and the second as Lexicographic PPO (LPPO). We conclude this section by combining Theorem 3 with certain conditions $\mathcal{C}$ that are sufficient for the local and global convergence of A2C and PPO respectively, in order to obtain the following corollaries. The proofs of these corollaries contain further discussion and references regarding the conditions required in each case.

**Corollary 1.** Suppose that each critic is linearly parametrised as $V_i(s) = w^T_i \phi(s)$ for some choice of state features $\phi$ and is updated using a semi-gradient TD(0) rule, and that:

1. $S$ and $A$ are finite, and each reward function $R_i$ is bounded.
2. For any $\theta \in \Theta$, the induced Markov chain over $S$ is irreducible.
3. For any $s \in S$ and $a \in A$, $\pi(a \mid s; \theta)$ is twice continuously differentiable.
4. Letting $\Phi$ be the $|S| \times c$ matrix with rows $\phi(s)$, then $\Phi$ has full rank (i.e. the features are independent), $c \leq |S|$, and there is no $w \in W$ such that $\Phi w = 1$.

Then for any MOMDP with discounted or limit-average objectives, LA2C almost surely converges to a policy in $\Theta^*_m$.

**Corollary 2.** Let $\pi(a \mid s; \chi)$ be the $|S| \times c$ matrix with rows $\phi(s)$, then $\hat{\phi}^T(s, a; \theta)$ and suppose that both $\pi$ and the action-value critics $Q_i$ are parametrised using two-layer neural networks (where $\chi$ is a temperature parameter), that a semi-gradient TD(0) rule is used to update $Q_i$, and that $Q_i$ replaces $A_i$ in the standard PPO loss $K^{\text{PPO}}_\infty$, both of which updates use samples from the discounted steady state distribution. Further, let us assume that:

1. $S$ is compact and $A$ is finite, with $S \times A \subseteq \mathbb{R}^d$ for some finite $d > 0$, and each reward function $R_i$ is bounded.
2. The neural networks have widths $\mu_f$ and $\mu_Q$, respectively with ReLU activations, initial input weights drawn from a normal distribution with mean 0 and variance $\frac{1}{2}$, and initial output weights drawn from $\text{unif}([-1, 1])$.
3. We have that $q^T(a, \cdot) \in \{Q_i(s, a, \cdot) \mid w_i \in \mathbb{R}^y\}$ for any $\pi \in \Pi$.
4. There exists $c > 0$ such that for any $z \in \mathbb{R}^d$ and $\zeta > 0$ we have that $\mathbb{E}_\pi[|z^T(s, a)| \leq \zeta] \leq \frac{c}{\epsilon^2}$ for any $\pi \in \Pi$.

Then for any MOMDP with discounted objectives, LPPO almost surely converges to a policy in $\Theta^*_m$. Furthermore, if the coefficient of the KL divergence penalty $\kappa > 1$ then $\lim_{\mu_f, \mu_Q, \kappa \to \infty} \epsilon = 0$.

4 **Experiments**

In this section we evaluate our algorithms empirically. We first show how the learning time of LRL scales with the number of reward functions. We then compare the performance of VB-LRL and PB-LRL against that of other algorithms for solving constrained RL problems. Further experimental details and additional experiments are described in the supplementary material, and documented in our codebase.  

4.1 Scaling with the Number of Rewards

Our first experiment (shown in Figure 1) shows how the learning time of LRL scales in the number of rewards. The data suggest that the learning time grows sub-linearly as additional reward functions are added, meaning that our algorithms can be used with large numbers of objectives.

4.2 Lexicographic RL for Safety Constraints

Many tasks are naturally expressed in terms of both a performance metric and a safety constraint. Our second experiment compares the performance of LRL against RCPO [Tessler et al., 2019], AproPO [Miryoosefi et al., 2019], and the actor-critic algorithm for VaR-constraints in [Chow et al., 2017], in a number of environments with both a performance metric and a safety constraint. These algorithms synthesise slightly different kinds of policies, but are nonetheless sufficiently similar for a relevant comparison to be made. We use VB-LRL with a neural network and a replay buffer, which we call LDQN, and the PB-LRL algorithms we evaluate are LA2C and LPPO. The results are shown in Figure 2.

The CartSafe environment from *gym-safety* is a version of the classic CartPole environment. The agent receives more reward the higher up the pole is, whilst incurring a cost if the cart is moved outside a safe region. Here the LRL algorithms, RCPO, and AproPO all learn quite safe policies, but VaR-AC struggles. Of the safer policies LDQN gets the most reward.

---

Available at https://github.com/lrhammond/lmorl.

Available at https://github.com/jemaw/gym-safety.
(roughly matching DQN and A2C), followed by RCPO and AproPO, and then LA2C and LPPO. The latter two minimise cost more aggressively, and thus gain less reward.

Finally, in the Intersection environment from highway-env, the agent must guide a car through an intersection with dense traffic. We give the agent a reward of 10 if it reaches its destination, and a cost of 1 for each collision that occurs (which is slightly different from the environment’s original reward structure). This task is challenging, and all the algorithms incur approximately the same cost as a random agent. However, they still manage to increase their reward, with LA2C and RCPO obtaining the most reward out of the constrained algorithms (roughly matching that of DQN and A2C). This shows that if optimising the first objective is too difficult, then the LRL algorithms fail gracefully by optimising the second objective, even if it has lexicographically lower priority.

5 Discussion and Conclusions

We introduced two families of RL algorithms for solving lexicographic multi-objective problems, which are more general than prior work, and are justified both by their favourable theoretical guarantees and their compelling empirical performance against other algorithms for constrained RL. VB-LRL converges to a lexicographically optimal policy in the tabular setting, and PB-LRL inherits convergence guarantees as a function of the objectives used, leading to locally and globally lexicographically ϵ-optimal policies in the case of LA2C and LPPO respectively. The learning time of the algorithms grows sub-linearly as reward functions are added, which is an encouraging result for scalability to larger problems. Further, when used to impose safety constraints, the LRL algorithms generally compare favourably to the state of the art, both in terms of learning speed and final performance.

We conclude by noting that in many situations, LRL may be preferable to constrained RL for reasons beyond its strong performance, as it allows one to solve different kinds of problems. For example, we might want a policy that is as safe as possible, but lack prior knowledge of what level of safety can be attained in the environment. LRL could also be used e.g. to guide learning by encoding prior knowledge in extra reward signals without the risk of sacrificing optimality with respect to the primary objective(s). These applications, among others, provide possible directions for future work.

Acknowledgments

Hammond acknowledges the support of an EPSRC Doctoral Training Partnership studentship (Reference: 2218880).

References

[Achiam et al., 2017] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In Proceedings of the 34th International Conference on Machine Learning, pages 22–31, 2017.

[Amari, 1998] Shun-ichi Amari. Natural gradient works efficiently in learning. Neural Computation, 10(2):251–276, 1998.

Available at https://github.com/eleurent/highway-env.
[Ben-Israel and Mond, 1986] A. Ben-Israel and B. Mond. What is invexity? The Journal of the Australian Mathematical Society. Series B. Applied Mathematics, 28(1):1–9, 1986.

[Bertsekas, 1999] Dimitri Bertsekas. Nonlinear Programming. Athena Scientific, 1999.

[Bhatnagar et al., 2009] Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. Automatica, 45(11):2471–2482, 2009.

[Borkar, 2008] Vivek S. Borkar. Stochastic Approximation. Hindustan Book Agency, 2008.

[Chow et al., 2017] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. Journal of Machine Learning Research, 18(1):6070–6120, 2017.

[Chow et al., 2018] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, pages 8103–8112, 2018.

[Craven, 1981] B.D. Craven. Invex functions and constrained local minima. Bulletin of the Australian Mathematical Society, 24(3):357–366, 1981.

[Gábor et al., 1998] 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.

[Hanson, 1981] Morgan A Hanson. On sufficiency of the kuhn-tucker conditions. Journal of Mathematical Analysis and Applications, 80(2):545–550, 1981.

[Hasselt, 2010] Hado V. Hasselt. Double q-learning. In Proceedings of the 24th International Conference on Neural Information Processing Systems, pages 2613–2621, 2010.

[Hsu et al., 2020] Chloe Ching-Yun Hsu, Celestine Mendler-Dünn, and Moritz Hardt. Revisiting design choices in proximal policy optimization. arXiv:2009.10897, 2020.

[Kakade, 2001] Sham Kakade. A natural policy gradient. In Proceedings of the 14th International Conference on Neural Information Processing Systems, pages 1531–1538, 2001.

[Konda and Tsitsiklis, 2000] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In Proceedings of the 13th International Conference on Neural Information Processing Systems, pages 1008–1014, 2000.

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

[Liu et al., 2015] C. Liu, X. Xu, and D. Hu. Multiobjective reinforcement learning: A comprehensive overview. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45(3):385–398, 2015.

[Liu et al., 2019] Boyi Liu, Qi Cai, Zhouran Yang, and Zhaoran Wang. Neural trust region/proximal policy optimization attains globally optimal policy. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 10564–10575, 2019.

[Miryoosefi et al., 2019] Sobhan Miryoosefi, Kianté Brantly, Hal Daume III, Miroslav Dudík, and Robert E. Schapire. Reinforcement learning with convex constraints. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 14070–14079, 2019.

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

[Paternain et al., 2019] Santiago Paternain, Luiz F. O. Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 7553–7563, 2019.

[Polymenakos et al., 2019] K. Polymenakos, A. Abate, and S. Roberts. Safe policy search using gaussian process models. In Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems, pages 1565–1573, 2019.

[Rentmeesters et al., 1996] M.J. Rentmeesters, W.K. Tsai, and Kwei-Jay Lin. A theory of lexicographic multi-criteria optimization. In Proceedings of the 2nd IEEE International Conference on Engineering of Complex Computer Systems, 1996.

[Rojiers et al., 2013] D. M. Roijers, P. Vamplew, S. White, and R. Dazeley. A survey of multi-objective sequential decision-making. Journal of Artificial Intelligence Research, 48:67–113, 2013.

[Rummery and Niranjan, 1994] Gavin Rummery and Mehsan Niranjan. On-line q-learning using connectionist systems. Technical report, University of Cambridge, 1994.

[Schulman et al., 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv:1707.06347, 2017.

[Singh et al., 2000] Satinder Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. Machine Learning, 38:287–308, 2000.

[Slater, 1950] Morton Slater. Lagrange multipliers revisited. Cowles Commission Discussion Paper No. 403, 1950.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, 2018.

[Sutton et al., 1999] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In Proceedings of the 12th International Conference on Neural Information Processing Systems, pages 1057–1063, 1999.
6 General Form of VB-LRL

There are several places where VB-LRL makes use of a tolerance parameter $\tau$. In the main text of this paper, we assume that the same tolerance parameter is used everywhere, and that it is a constant. Here we provide a more general form of VB-LRL, which is also the form that will be used in our proofs. First, let a tolerance function $\tau: \mathbb{S} \times (\mathbb{S} \times \mathbb{A} \times \mathbb{T} \times \mathbb{N}) \to \mathbb{R}_{>0}$ be a function that takes as input a state, reward priority, time step, and sequence of Q-functions, and returns a (positive) variable corresponding to a tolerance. In other words, $\tau(s, i, t, Q_1, \ldots, Q_m)$ is the tolerance for the $i$th reward when selecting an action in state $s$ at time $t$, given the current Q-functions. We also require that $\lim_{t \to \infty} \tau(s, i, t, Q_1, \ldots, Q_m)$ exists. We will revisit the specification of VB-LRL using such tolerance functions, starting with a more general definition of lexicographic bandit algorithms.

Definition 1 (Lexicographic Bandit Algorithm). Let $S$ be a set of states, $A$ a set of actions, $Q_1, \ldots, Q_m : S \times A \to \mathbb{R}$ a sequence of $Q$-functions, and $t \in \mathbb{N}$ a time parameter. Then a lexicographic bandit algorithm with tolerance function $\tau_B : S \times (\{1, \ldots, m\} \times \mathbb{N} \times (S \times A \to \mathbb{R})^m \to \mathbb{R}_{>0}$ is a function $B : (S \times A \to \mathbb{R})^m \times \mathbb{S} \times \mathbb{N} \to A$, such that:

$$\lim_{t \to \infty} \Pr(B(Q_1, \ldots, Q_m, s, t) \in \Delta_{s, m}^\tau) = 1,$$

where $\Delta_{s, 0}^\tau = A$ and $\Delta_{s, i+1}^\tau := \{a \in \Delta_{s, i}^\tau \mid Q_i(s, a) \geq \max_{a' \in \Delta_{s, i}^\tau} Q_i(s, a') - \lim_{t \to \infty} \tau_B(s, i+1, t, Q_1, \ldots, Q_m)\}$.

We next generalise the definition of lexicographic Q-learning in a similar way. Given a tolerance function $\tau_Q$, we say that lexicographic Q-learning with tolerance $\tau_Q$ is the update rule:

$$Q_i(s, a) \leftarrow (1 - \alpha_i(s, a)) \cdot Q_i(s, a) + \alpha_i(s, a) \cdot \left(R_i(s, a, s') + \gamma_i \max_{a' \in A} Q_i(s', a')\right),$$

where $\Delta_{s, 0}^{\tau_Q} = A$ and $\Delta_{s, i+1}^{\tau_Q} := \{a \in \Delta_{s, i}^{\tau_Q} \mid Q_i(s, a) \geq \max_{a' \in \Delta_{s, i}^{\tau_Q}} Q_i(s, a') - \tau_Q(s, i+1, t, Q_1, \ldots, Q_m)\}$. Note that $\tau_B$ and $\tau_Q$ may, but need not, be the same.

These generalisations may seem excessive, but they let us set the tolerance in a much more flexible way. For example, if we do not know the scale of the reward, then it might be beneficial to express the tolerance as a proportion rather than a constant, such as by setting $\tau(s, i, t, Q_1, \ldots, Q_m) := \sigma \cdot \max_{a \in A} Q_i(s, a)$, where $\sigma \in (0, 1)$. If we do this, then the tolerance will depend on $s, i, t, Q_1, \ldots, Q_m$. We might also wish to decrease over time, in which case the tolerance will additionally depend on $t$. To give another example, we could implement the tolerance by casting the $Q$-values to low-precision floats, letting the rounding error correspond to the tolerance. The benefit of the more general tolerance functions we define above is that they easily can capture these (and similar) kinds of setups.

7 VB-LRL Convergence Proof

We now prove that VB-LRL converges to a lexicographically optimal policy, making use of the following lemma.
Lemma 1. Let $(\zeta_t, \beta_t, F_t)$ be a stochastic process where $\zeta_t, \beta_t, F_t : X \to \mathbb{R}$ satisfy:

$$\beta_{t+1}(x) = (1 - \zeta_t(x_t)) \cdot \beta_t(x_t) + \zeta_t(x_t) \cdot F_t(x_t),$$

with $x_t \in X$ and $t \in \mathbb{N}$. Let $P_t$ be a sequence of increasing $\sigma$-fields such that $\zeta_t$ and $\beta_t$ are $P_t$-measurable and $\zeta_t, \beta_t$ and $F_{t-1}$ are $P_t$-measurable, and $t \geq 1$. Then $\beta_t$ converges to 0 with probability one if:

1. $X$ is finite,
2. $\zeta_t(x_t) \in [0,1]$ and $\forall x \neq x_t$ we have $\zeta_t(x) = 0$,
3. $\sum_t \zeta_t(x_t) = \infty$ and $\sum_t \zeta_t(x_t)^2 < \infty$ with probability one,
4. $\text{Var}(F_t(x_t) \mid P_t) \leq K(1 + \kappa \|\beta_t\|_\infty)^2$ for some $K \in \mathbb{R}$ and $\kappa \in [0,1],$
5. $\|\mathbb{E}(F_t \mid P_t)\|_\infty \leq \kappa \|\beta_t\|_\infty + c_t,$ where $c_t \to 0$ with probability one as $t \to \infty,$

where $\| \cdot \|_\infty$ is a (potentially weighted) maximum norm.

Proof. See [Singh et al., 2000].

We next give the general statement of the Theorem:

Theorem 1. In any MOMDP $M$, if VB-LRL uses SARSA, Expected SARSA, or Lexicographic Q-Learning, then it will converge to a policy $\pi$ that is lexicographically optimal if:

1. $S$ and $A$ are finite,
2. All reward functions are bounded,
3. Either $\gamma_1, \ldots, \gamma_m < 1$, or every policy leads to a terminal state with probability one,
4. The learning rates $\alpha_t(s,a) \in [0,1]$ satisfy the conditions $\sum_t \alpha_t(s,a) = \infty$ and $\sum_t \alpha_t(s,a)^2 < \infty$ with probability one, for all $s \in S, a \in A$,
5. The tolerance function $\tau_B$ used in the lexicographic bandit algorithm satisfies the condition that $0 < \lim_{t \to \infty} \tau_B(s,i,t,Q_1,\ldots,Q_m) < \inf_{a \neq a'} q_t(s,a) - q_t(s,a')$; if lexicographic Q-learning is used then its tolerance function $\tau_Q$ must also satisfy this condition.

Note that condition 4 requires that the agent visits every state-action pair infinitely many times. Before giving the proof, we need to provide a few definitions. We say that the $Q$-functions $Q_1, \ldots, Q_m$ are lexicographically optimal if they correspond to the $q$-values of (globally) lexicographically optimal policies, and we will also use $q_1^t, \ldots, q_m^t$ to denote these $Q$-functions. We define $\text{lexmax}$ as:

$$\text{lexmax}_a q_t^t(s,a) = \max_{a \in A} q_t^t(s,a),$$

with $\Delta_{s,0} = A$ and $\Delta_{s,t+1} = \arg \max_{a \in \Delta_{s,t}} q_{t+1}(s,a).$ Moreover, given a function $\sigma : S \times \{1 \ldots m\} \times (S \times A \to \mathbb{R})^m \to \mathbb{R}_{>0}$ that takes as input a state, reward priority, and sequence of $Q$-functions, and returns a non-negative real value, we define $\sigma$-slack $\text{lexmax}$ as:

$$\sigma\text{-slack lexmax}_a Q_t(s,a) = \max_{a \in \Delta_{s,t}^\sigma} Q_t(s,a),$$

where $\Delta_{s,0}^\sigma = A$ and $\Delta_{s,t+1}^\sigma = \{a \in \Delta_{s,t}^{\sigma} \mid Q_t(s,a) \geq \text{max}_{a' \in \Delta_{s,t}^{\sigma}} Q_t(s,a') - \sigma(s,i,Q_1,\ldots,Q_m)\}$. In other words,

$$\text{lexmax}_a q_t^t(s,a)$$

is the maximal value of $q_t^t$ at $s$ if $a$ is chosen from the actions that lexicographically maximise $q_1^t, \ldots, q_m^t,$ and $\sigma$-slack $\text{lexmax}_a Q_t(s,a)$ is the maximal value of $Q_t$ at $s$ if $a$ is chosen from the actions that lexicographically maximise $Q_1, \ldots, Q_m$ with tolerance $\sigma$. Note that $\sigma$ does not depend on the time $t,$ whereas the tolerance functions $(\tau)$ which we defined earlier are allowed to depend on $t.$ A $Q$-value update rule is a temporal difference (TD) update rule if it has the following form:

$$Q_t(s_t,a_t) \leftarrow (1 - \alpha_t(s_t,a_t)) \cdot Q_t(s,a) + \alpha_t(s_t,a_t) \cdot (r_{t,t} + \gamma_t \hat{v}_{n,t}(s_{t+1})),$$

where $\hat{v}_{n,t}(s)$ estimates the value on $s$ under $R_n.$ Note that Lexicographic Q-Learning, SARSA, and Expected SARSA all are TD update rules. We use $Q_{i,t}$ to refer to the agent’s $Q$-function for the $i$th reward at time $t.$ Note that $q_{i,t}^t$ is the “true” $Q$-function for the $i$th reward, whereas $Q_{i,t}$ is the agent’s estimate of $q_{i,t}^t$ at time $t.$ We can now finally give the proof.

Proof. The proof proceeds by induction on the reward signals. For $n \in \{0 \ldots m\},$ let $P(n)$ signify that $Q_1,\ldots,Q_{n,t}$ converge to $q_1^t,\ldots,q_n^t$ as $t \to \infty.$ The base case $P(0)$ holds vacuously. As for the inductive step $P(n) \to P(n+1),$ let:

- $X = S \times A,$
- $\zeta_t(s,a) = \alpha_t(s,a),$
- $\beta_t(s,a) = Q_{n+1,t}(s,a) - q_{n+1}(s,a),$
- $F_t(s,a) = r_{n+1,t} + \gamma_{n+1} \hat{v}_{n,t+1}(s_{t+1}) - q_{n+1}(s,a).$

Note that $\hat{v}_{n,t+1}(s_{t+1})$ comes from the $Q$-value update rule. Now assumptions 1, 2, and 4 of this theorem imply conditions 1–4 in Lemma 1. It remains to show that condition 5 holds, which we can do algebraically. If Lexicographic Q-Learning is used then let:

$$\sigma(s,i,Q_1,t \ldots Q_m,t) = \lim_{t' \to \infty} \tau_Q(s,i,t',Q_1,t \ldots Q_m,t),$$

where $\tau_Q$ is the tolerance function used in the lexicographic Q-learning update rule. Otherwise, if SARSA or Expected SARSA is used, let:

$$\sigma(s,i,Q_1,t \ldots Q_m,t) = \lim_{t' \to \infty} \tau_B(s,i,t',Q_1,t \ldots Q_m,t),$$

where $\tau_B$ is the tolerance function used in the lexicographic bandit algorithm. It is important to note that $\sigma$ is defined as the limit of $\tau_Q$ or $\tau_B$ when the time parameter that $\tau_Q$ or $\tau_B$ takes as input goes to $\infty,$ and that this limit is not applied to the other arguments of $\tau_Q$ or $\tau_B.$ In other words, it is not a typo that the $Q_{1,t} \ldots Q_{m,t}$ are parameterised by $t$ rather than $t'.$ Indeed, we have not yet shown that $\lim_{t' \to \infty} Q_{i,t}$ even exists. With this noted, we begin as follows:
\[
\|E(F_t|P_t)\|_\infty = \max_{s,a} \mathbb{E}\left[r_{n+1,t} + \gamma_{n+1} \max_{a'} Q_{n+1,t+1}(s_{t+1}, a') - q_{n+1}^l(s, a) + \gamma_{n+1} \max_{a'} Q_{n+1,t+1}(s_{t+1}, a')\right]
\]
\[
\leq \max_{s,a} \mathbb{E}\left[\gamma_{n+1} \max_{a'} Q_{n+1,t+1}(s_{t+1}, a')\right]
\]
\[
\gamma_{n+1} \max_{a'} Q_{n+1,t+1}(s_{t+1}, a') + \gamma_{n+1} \max_{a'} Q_{n+1,t+1}(s_{t+1}, a') - q_{n+1}^l(s, a)
\]

The second term is bounded above by:
\[
\max_{s,a} \mathbb{E}\left[\hat{v}_{n+1,t+1}(s) - \sigma \max_{a'} Q_{n+1,t+1}(s, a)\right].
\]

Let \(k_t\) denote this expression — we will argue that \(k_t \to 0\) as \(t \to \infty\). First recall that:
\[
\sigma \max_{a'} Q_{n+1,t+1}(s, a) = \max_{a \in \Delta_{s,n+1}} Q_{n+1,t+1}(s, a)
\]

If VB-LRL uses Lexicographic Q-Learning (with \(\tau_B\)) then:
\[
\hat{v}_{n+1,t+1}(s) = \max_{a \in \Delta_{s,n+1}} Q_{n+1,t+1}(s, a),
\]

and also recall that:
\[
\sigma(s, n + 1, Q_{1,t+1} \ldots Q_{m,t+1}) = \lim_{t' \to \infty} \tau_B(s, n + 1, t', Q_{1,t+1} \ldots Q_{m,t+1}).
\]

If assumption 5 holds, and the inductive assumption holds, then this implies that there is a time \(t'\) such that:
\[
\Delta_{s,n+1} = \Delta_{s,n+1}^\tau_B\]

for all \(t'' \geq t'\), which also means that \(k_{t''} = 0\) for all \(t'' \geq t'\). Hence, if VB-LRL uses Lexicographic Q-Learning then \(k_t \to 0\) as \(t \to \infty\). If instead VB-LRL uses SARSA or Expected SARSA then we have that:
\[
\mathbb{E}\left[\hat{v}_{n+1,t+1}(s)\right] = \max_{a \in \Delta_{s,n+1}} Q_{n+1,t+1}(s, a),
\]

where \(B\) is the bandit algorithm that is used, and:
\[
\sigma(s, n + 1, Q_{1,t+1} \ldots Q_{m,t+1}) = \lim_{t' \to \infty} \tau_B(s, n + 1, t', Q_{1,t+1} \ldots Q_{m,t+1}),
\]

where \(\tau_B\) is the tolerance of \(B\). From the properties of lexicographic bandit algorithms, we have that:
\[
\lim_{t \to \infty} \Pr(B(Q_1, \ldots, Q_m, s, t) \in \Delta_{s,n+1}^\tau_B) = 1.
\]

Moreover, since in this case \(\sigma = \lim_{t' \to \infty} \tau_B\), we straightforwardly have that: \(\Delta_{s,n+1} = \Delta_{s,n+1}^\tau\). This means that:
\[
\lim_{t' \to \infty} \Pr(\hat{v}_{n+1,t'}(s) = \max_{a \in \Delta_{s,n+1}} Q_{n+1,t+1}(s, a)) = 1.
\]

Since the action space \(A\) is finite, this also means that:
\[
\lim_{t' \to \infty} \mathbb{E}[\hat{v}_{n+1,t'}(s) - \max_{a \in \Delta_{s,n+1}} Q_{n+1,t+1}(s, a)] = 0.
\]

In other words, if VB-LRL uses SARSA or Expected SARSA then \(k_t \to 0\) as \(t \to \infty\). This exhausts all cases, so we can conclude that \(k_t \to 0\) as \(t \to \infty\). We thus have:
\[
\|E(F_t|P_t)\|_\infty \leq \max_{s,a} \mathbb{E}\left[r_{n+1,t} + \gamma_{n+1} \max_{a'} Q_{n+1,t+1}(s_{t+1}, a') - q_{n+1}^l(s, a) + \gamma_{n+1} \max_{a'} Q_{n+1,t+1}(s_{t+1}, a')\right] + k_t.
\]

Recall that:
\[
\sigma \max_{a} q_{n+1}^l(s, a) = \max_{a \in \Delta_{s,n+1}} Q_{n+1,t+1}(s, a),
\]
\[
\sigma \max_{a} Q_{n+1,t}(s, a) = \max_{a \in \Delta_{s,n+1}} Q_{n+1,t}(s, a).
\]

The inductive assumption states that \(Q_{1,t} \ldots Q_{n,t}\) converge to \(q_1^l \ldots q_n^l\) as \(t \to \infty\). This, together with assumption 5, implies that there is a time \(t'\) such that \(\Delta_{s,n+1} = \Delta_{s,n}^\tau\) for all \(t'' \geq t'\). This in turn implies that:
\[
\sigma \max_{a} q_{n+1,t''}(s, a) - \sigma \max_{a} q_{n+1}^l(s, a) \leq \max_{a} \left[Q_{n+1,t''}(s, a) - q_{n+1,1}^l(s, a)\right],
\]
for all $t'' \geq t'$. We can therefore continue as follows:

$$
\leq \gamma_{n+1} \max_{s,a} \sum_{s' \sim T(s,a)} \max_{a'} |Q_{n+1,t}(s', a') - q_{n+1}(s', a')| + d_t(s) + k_t
$$

where $c_t = \max_s d_t(s) + k_t$, and $d_t(s)$ is an error correction term given by:

$$
d_t(s) = \max_a \left( 0, \sigma \text{-slack} \max_a q_{n+1,t}(s, a) - \min_a |Q_{n+1,t}(s,a) - q_{n+1}(s,a)| \right).
$$

Note that $d_t(s)$ is 0 after $t'$, so $c_t \to 0$ as $t \to \infty$. Now if $\gamma_{n+1} < 1$ we furthermore have:

$$
\leq \gamma_{n+1} \max_{s,a} |\beta(s, a)| + c_t = \gamma_{n+1} \|\beta\|_\infty + c_t.
$$

This means that:

$$
\|\mathbb{E}\{F_t \mid P_t\}\|_\infty \leq \gamma_{n+1} \|\beta\|_\infty + c_t,
$$

where $\gamma_{n+1} \in [0, 1)$ and $c_t \to 0$ as $t \to \infty$. Thus by Lemma 1 we have that $Q_{n+1}$ converges to $g_{n+1}$, and so $P(n+1)$ holds.

Now suppose $\gamma_{n+1} = 1$, and that every policy leads to a terminal state with probability one. For this case we will need to use a specific maximum norm. Let:

- $S_1$ be the set of terminal states in $\mathcal{M}$,
- $S_{i+1} = \{ s \in S_1 \cup \cdots \cup S_i \mid \forall a \in A : P[T(s,a) \in S_1 \cup \cdots \cup S_i] > 0 \}$,
- $\eta$ be the largest integer such that $S_\eta \neq \emptyset$,
- $S_{a,b} = S_a \cup S_{a+1} \cup \cdots \cup S_b$ (where $a, b \in \mathbb{N}$ with $a < b$),
- $d : S \to \mathbb{N}$ be the function such that $d(s) = i$ if $s \in S_i$,
- $\epsilon = \min_{r \in \{2, \ldots, \eta\}} \min_{a \in A} \min_{s \in S_a} P[T(s, a) \in S_1 \cup \cdots \cup S_{r-1}]$,
- If $\epsilon = 1$ then $w_i = i$, else $w_1 = 1$, and $w_{i+1} = 0.5(w_i + \min_{j \in \{1, \ldots, i\}} \{ w_j + \frac{\epsilon}{1-\epsilon}(w_j - w_{j-1}) \})$,
- $\| \cdot \|_W$ be the maximum norm over $S \times A$ where the weight of $(s, a)$ is $w_d(s)$.

Intuitively, $S_1, \ldots, S_\eta$ represent different “distances” to a terminal state. For example, if $s \in S_1$ and an agent is at $s$ then the agent will reach a terminal state with positive probability after $i - 1$ steps, regardless of which actions it selects. The constant $\eta$ gives the maximal “distance”, which must be finite if every policy leads to a terminal state with probability one. The constant $\epsilon$ gives a (positive) lower bound on the probability of moving “closer” to a terminal state after taking an action, and $w_1, \ldots, w_\eta$ are the weights we will use to define our maximum norm.

By straightforward induction we have that $w_1 < \cdots < w_\eta$ and that:

$$
\left( \frac{w_\eta}{w_1} \right) (1 - \epsilon) + \left( \frac{w_{\eta-1}}{w_1} \right) \epsilon < 1.
$$

Let $\kappa = \max_{i \in \{1, \ldots, \eta\}} \left( \frac{w_i}{w_1} \right) (1 - \epsilon) + \left( \frac{w_{i-1}}{w_1} \right) \epsilon$. We can now go back to the inequality:

$$
\gamma_{n+1} \max_{s,a} \sum_{s' \in S} P[T(s, a) = s'] \max_{a'} |\beta(s', a')| + c_t
$$

$$
\leq \max_{s,a} \sum_{s' \in S} P[T(s, a) = s'] \max_{a'} |\beta(s', a')|
$$

$$
\cdot \left( \frac{w_d(s')}{w_d(s')} \right) + c_t
$$

$$
\leq \max_{s,a} \left( \frac{w_d(s)}{w_d(s')} \right) .
$$

$$
\leq \gamma_{n+1} \max_{s,a} \sum_{s' \in S} P[T(s, a) = s'] \left( \frac{w_d(s')}{w_d(s')} \right) + c_t.
$$

By factoring the sum, and using the fact that $w_1 < \cdots < w_\eta$, we obtain:

$$
\leq \|\beta_t\|_W \max_{s,a} \left[ \sum_{s' \in S_{d(s)-1}} P[T(s, a) = s'] \left( \frac{w_d(s')}{w_d(s')} - \frac{w_\eta}{w_d(s')} \right) \right]
$$

$$
+ \sum_{s' \in S_{d(s), \eta}} P[T(s, a) = s'] \left( \frac{w_\eta}{w_d(s')} \right) + c_t
$$

$$
\leq \|\beta_t\|_W \max_{s,a} \left[ \epsilon \left( \frac{w_d(s')}{w_d(s')} - \frac{w_\eta}{w_d(s')} \right) + \left( \frac{w_\eta}{w_d(s')} \right) \right] + c_t.
$$

This means that:

$$
\|\mathbb{E}\{F_t \mid P_t\}\|_\infty \leq \kappa \|\beta_t\|_W + c_t,
$$

where $\kappa \in [0, 1)$ and $c_t \to 0$ as $t \to \infty$. Thus Lemma 1 implies that $Q_{n+1}$ converges to $g_{n+1}$, so $P(n+1)$ holds.

This exhausts all cases, which means that the inductive step holds, and so $Q_1, \ldots, Q_m$ must all converge to lexicographically optimal values. From the properties of lexicographic bandit algorithms, and assumption 5, we then see that LRL must converge to a lexicographically optimal policy. $\square$
We next provide a separate proof that VB-LRL converges to a lexicographically optimal policy if it is used with Lexicographic Double Q-Learning.

**Lemma 2.** In any MDP, if conditions 1–4 in Theorem 1 are satisfied then Double Q-Learning converges to an optimal policy.

Proof. See [Hasselt, 2010].

**Theorem 2.** In any MOMDP $M$, if VB-LRL is used with Lexicographic Double Q-Learning and conditions 1–5 in Theorem 1 are satisfied then it will converge to a lexicographically optimal policy.

Proof. We prove this by induction on the reward signals. For $n \in \{0, \ldots, m\}$, let $P(n)$ signify that $Q_{A}^{1, t}, \ldots, Q_{A}^{n, t}$ and $Q_{B}^{1, t}, \ldots, Q_{B}^{n, t}$ converge to $q_{1}^{i}, \ldots, q_{m}^{i}$ as $t \to \infty$.

The base case $P(0)$ is vacuously true. For the inductive step $P(n) \to P(n + 1)$ we have that assumption 5 and the inductive assumption together imply that there is a time $t'$ such that for all $t'' \geq t'$ we have that:

$$\Delta_{s,n+1} = \Delta_{s,n+1}.$$  

Let $M^{-}$ be the MOMDP that is created if in each state of $M$ the action set is restricted to $\Delta_{s,n+1}$. Note that after time $t'$ the agent will update $Q_{A}^{1, t'}$ and $Q_{B}^{1, t'}$ like a Double Q-Learning agent with reward $R_{n+1}$ running in $M^{-}$ (starting from a particular initialisation). Thus, by Lemma 2, $Q_{A}^{1, t'}$ and $Q_{B}^{1, t'}$ converge to the optimal Q-values for $R_{n+1}$ in $M^{-}$. These values correspond to the lexicographically optimal values for $R_{n+1}$ in $M$, and so $P(n) \to P(n + 1)$ holds. We thus have that $Q_{1,t}, \ldots, Q_{m,t}$ converge to $q_{1}^{i}, \ldots, q_{m}^{i}$ as $t \to \infty$. Assumption 5 then implies that Lexicographic Double Q-learning converges to a lexicographically optimal policy. This completes the proof.

8 VB-LRL Tolerance

In this section, we prove a formal guarantee about the limit behaviour of VB-LRL with respect to its two most priori sation reward signals, which can be obtained without any prior knowledge of the reward functions. We also further comment on the problem of selecting the tolerance parameters.

We begin by arguing that although the convergence of $\tau_{B}$ and $\tau_{Q}$ to 0 is not technically sufficient to guarantee convergence to a lexicographically optimal policy, such a policy should be found as long as these tolerances approach 0 sufficiently slowly. The reason for this insufficiency is that if $\tau_{B}$ and $\tau_{Q}$ approach 0 faster than $Q_{1}, \ldots, Q_{m}$ approach their limit values, then this might lead the sets $\Delta_{s,n}^{t}$ – which are quantified over in the Q-value update rule – to forever remain too small, after a certain time. This is because we might have $q_{1}^{i}(s,a_{1}) = q_{1}^{i}(s,a_{2})$ and a tolerance that shrinks faster than $Q_{1,t}(s,a_{1}) - Q_{1,t}(s,a_{2})$; in that case, $a_{1}$ or $a_{2}$ might erroneously be considered suboptimal according to $R_{t}$ during the updating of $Q_{1,t}, \ldots, Q_{m,t}$.

If the tolerance parameters decrease more slowly than the $Q_{t}$-values converge, however, then this will not happen. It should therefore be possible to reliably obtain convergence in practice by simply letting $\tau_{B}$ and $\tau_{Q}$ decrease to 0 very slowly. Moreover, we also expect VB-LRL to in practice be likely to converge to a policy that is approximately lexicographically optimal as long as the limit values of $\tau_{B}$ and $\tau_{Q}$ are small (but positive). Evidence for this is provided by Proposition 1, for which we require the following lemma.

**Lemma 3.** Let $\pi$ and $\pi'$ be two policies in an MDP $M = (S,A,T,I,R,\gamma)$, and let $q_{s}(\pi,s') \geq q_{s}(\pi',s)$ for all $s \in S$. Then $J(\pi) \geq J(\pi') - \frac{\gamma}{1-\gamma}$. Moreover, this bound is tight.

Proof. Let $\pi_{n}$ be the (non-stationary) policy that selects its first $n$ actions according to $\pi$, and all its subsequent actions according to $\pi'$. By a straightforward inductive argument we have that $J(\pi_{n}) \geq J(\pi) - \sum_{i=1}^{n} \gamma^{i} \delta_{i}$, for all $n \in N$. Then, since $\lim_{n \to \infty} J(\pi_{n}) = J(\pi)$ and $\lim_{n \to \infty} J(\pi) - \sum_{i=1}^{n} \gamma^{i} \delta_{i}$, we have that $J(\pi') \geq J(\pi) - \frac{\delta}{1-\gamma}$. To see that the bound is tight, consider the MDP $M = (\{s\}, \{a_{1}, a_{2}\}, T, I, R, \gamma)$ where $R(s,a_{1},s) = \delta$, $R(s,a_{2},s) = 0$, $T(s') = s'$, $I(s) = 1$, and the policies $\pi : s \to a_{1}$ and $\pi' : s \to a_{2}$.

**Proposition 1.** Suppose VB-LRL either uses SARSA or Expected SARSA with any lexicographic bandit algorithm with tolerance $\tau_{B}$, or that it uses Lexicographic Q-Learning and Lexicographic $\epsilon$-Greedy with tolerance $\tau_{\epsilon}$ and $\tau_{B}$. Moreover, suppose that with probability one there exists a $t \in N$ such that:

$$\tau_{Q}(s,i,t',Q_{1}, \ldots, Q_{m}) \leq \tau_{B}(s,i,t'+1, Q_{1}, \ldots, Q_{m})$$

for all $t' \geq t$, and further that conditions 1–4 in Theorem 1 are satisfied. Then, in any MOMDP, we have that:

1. $J_{1}(\pi) - J_{1}(\pi_{1}) \leq \max_{i \in N} \tau_{B}^{i} - \lambda_{i}$, for some sequence $\{\lambda_{i}\}_{i \in N}$ such that $\lim_{i \to \infty} \lambda_{i} = 0$.

2. $J_{2}(\pi') - J_{2}(\pi_{1}) \leq \max_{i \in N} \tau_{B}^{i} - \eta_{i}$, for some sequence $\{\eta_{i}\}_{i \in N}$ such that $\lim_{i \to \infty} \eta_{i} = 0$.

where $\pi'$ is a lexicographically optimal policy, and $\max_{i} \tau_{B}^{i}$ is max$s, Q_{s}, \ldots, Q_{m}$ $\lim_{i \to \infty}$ $\tau_{B}(s,i,t, Q_{1}, \ldots, Q_{m})$.

**Sketch.** The updating of $Q_{1,t}$ is unaffected by the slack parameter, so $Q_{1,t}$ converges to $q_{1}^{i}$. This, together with Lemma 3 and the properties of lexicographic bandit algorithms, implies the first point of the proposition.

Since $Q_{1,t}$ converges to $q_{1}^{i}$, we have that there is a time $t'$ such that $\Delta_{s,n}^{t} \subseteq \Delta_{s,n}$ for all $s$ and all $t'' \geq t'$. This means that eventually $Q_{2,t}$ will always be updated in the direction of values that are at least as high as the corresponding $q_{2}^{i}$-values. We therefore have that $Q_{2,t}(s,a) \geq q_{2}^{i}(s,a) - \eta_{i}$ for some $\eta_{i}$ where $\lim_{i \to \infty} \eta_{i} = 0$ (this can be shown via Lemma 1, alternatively, one could perhaps design a scheme whereby the rate at which $\tau_{B}$ and $\tau_{Q}$ decrease depends on the rate at which $Q_{1}, \ldots, Q_{m}$ change. We conjecture that such a scheme could be used to derive a formal guarantee of VB-LRL’s convergence to a lexicographically optimal policy with high probability, but in this work we did not investigate this possibility due to the extra computational and theoretical overhead it would introduce.)
but we omit the full derivation for the sake of brevity). This, together with the conditions on $\tau_g$ and $\tau_Q$, Lemma 3, and the properties of lexicographic bandit algorithms, implies the second point of the proposition.

Note that the conditions that Proposition 1 imposes on $\tau_g$ and $\tau_Q$ are sufficient, rather than necessary, and that there might be additional choices of $\tau_g$ and $\tau_Q$ that also lead to the described behaviour. Note also that Proposition 1 does not state that VB-LRL’s policy $\pi$ will converge—it establishes certain conditions on the limit value of $\pi$, but does not rule out the possibility that $\pi$ itself continues to oscillate. This is presumably unlikely, but might perhaps happen if $q_i(s,a) = q_1(s,a) - \lim_{i \to \infty} \tau_Q(s,i,t,Q_1,\ldots,Q_m)$, $a_1 = \arg \max_a q_i(s,a)$, and $q_{i+1}(s,a_1) < q_i(s,a_2)$, for some $s,a_1,a_2$, and $i$.

Note also that Proposition 1 cannot be strengthened to make any claims about $J_2(\pi),\ldots,J_m(\pi)$ without additional assumptions. This is because we eventually have that $\Delta_{s,0}^{T,q,t} = \Delta_{s,0}$ and $\Delta_{s,1}^{T,q,t} \subseteq \Delta_{s,1}$, there is no guaranteed relationship between $\Delta_{s,1}^{T,q,t}$ and $\Delta_{s,1}$ for $i \geq 3$. To see this, consider what happens as we increase the tolerance. The $Q_1$-values will remain accurate. For $Q_2$, we now have fewer restrictions on what actions we can choose, which means that the $Q_2$-values should increase (or stay the same). However, this might lead the algorithm to choose an action $a$ where $a \not\in \Delta_{s,1}$, with $q_1(s,a) \approx \max_a q_1(s,a)$ and $q_2(s,a) > \max_a q_1(s,a)$. Then $q_1(s,a),\ldots,q_m(s,a)$ could have any arbitrary values relative to those of the lexicographically optimal actions.

9 PB-LRL Convergence Proof

The proof of PB-LRL’s convergence proceeds via a standard multi-timescale stochastic approximation argument; see Chapter 6 in [Borkar, 2008] for a full exposition. We begin by restating, for reference, the update rules for the parameters $\theta$ and $\lambda$ associated with Lagrangian $L_i$:

$$
\theta \leftarrow \Gamma_0 \left[ \theta + \beta_i \nabla_{\theta} \tilde{K}_i(\theta) + \sum_{j=0}^{i-1} \lambda_j \nabla_{\theta} \tilde{K}_j(\theta) \right],
$$

$$
\lambda_j \leftarrow \Gamma_\lambda \left[ \lambda_j + \eta_i \tilde{K}_j(\theta) - K_j(\theta) \right] \quad \forall j < i.
$$

We also restate the conditions on our learning rates $\eta_i \in \{\alpha,\beta_1,\ldots,\beta_m,\eta_1,\ldots,\eta_m\}$, where for all $i \in \{1,\ldots,m\}$ we have:

$$
\alpha_i \in [0,1], \quad \sum_{i=0}^{\infty} \alpha_i = \infty, \quad \sum_{i=0}^{\infty} (\alpha_i)^2 < \infty \quad \text{and}
$$

$$
\lim_{i \to \infty} \alpha_i = \lim_{i \to \infty} \eta_i = \lim_{i \to \infty} \beta_i = 0.
$$

Our main theorem for PB-LRL is then given as follows, where we make some minor observations before beginning the proof.

**Theorem 3.** Let $M$ be a MOMDP, $\pi$ a policy that is twice continuously differentiable in its parameters $\theta$, and assume that the same form of objective function is chosen for each $K_i$ and that each reward function $R_i$ is bounded. If using a critic, let $V_i$ (or $Q_i$) be (action-)value functions that are continuously differentiable in $w_i$ for $i \in \{1,\ldots,m\}$ and suppose that if PB-LRL is run for $T$ steps there exists some limit point $w^*_i(\theta) = \lim_{T \to \infty} E[w_i]$ for each $w_i$ when $\theta$ is held fixed under conditions $C$ on $M$, $\pi$, and each $V_i$. If $\lim_{T \to \infty} E[\theta_i] = \Theta_1$ (respectively $\Theta_1$) under conditions $C$ when $m = 1$, then for any fixed $m \in \mathbb{N}$ we have that $\lim_{T \to \infty} E[\theta_i] = \Theta_{m}$ (respectively $\Theta_{m}$), where each $\epsilon_i \geq 0$ is a constant that depends on the representational power of the parametrisations of $\pi$ (and $V_i$ or $Q_i$, if using a critic).

In the proof below we use $\epsilon_0 \geq 0$ to quantify the representational power of our policy parametrisation as follows:

$$
\min_{\theta \in \Theta} \max_{\pi,s} \sum_a |\pi(a | s) - \pi(a | s; \theta)| \leq \epsilon_0,
$$

where, as we will see, each $\epsilon_i$ is a function of $\epsilon_0$. In the tabular setting or when $\pi$ is represented using, say, an over-parametrised neural network, then we have that $\epsilon_0 = 0$ [Patermain et al., 2019].

Similarly, for any fixed $\theta$ it will not always be possible to guarantee that the approximation $V_i^*$ of each $v_i^*$ defined by $w_i^*(\theta)$ will have zero error and thus, if each estimated objective $\bar{K}_i$ is computed using $V_i$, we may only prove convergence to within a small neighbourhood of lexicographic optima. In particular, let us define $	ilde{J}_i(\theta) = \sum_s I(s)V_i^*(s)$. When $\max_{\theta,s} E[\theta_i][v_i^*(s) - V_i^*(s)] \leq \epsilon_i$, we have that $\max_{\theta \in \Theta_{i-1}} \tilde{J}_i(\theta) - \tilde{J}_i(\theta^*) \leq \epsilon_i$, where $\theta^* \in \arg \max_{\theta \in \Theta_{i-1}} \tilde{J}_i(\theta)$. As such, for the majority of the proof we consider convergence with respect to each $\tilde{J}_i$ instead of each $J_i$.

Finally, as noted in the main body of the paper, when $m = 1$ then PB-LRL reduces to whatever algorithm is defined by the choice of objective function, such as A2C when using $L_2A2C$, or PPO when using $L_2PPO$. As, by assumption, PB-LRL converges to an optimum (either local or global) of $\tilde{J}_i$ by following (an estimate of) the gradient of $\bar{K}_i$, we see that $\nabla_{\theta} K_i(\theta)$ serves as a faithful surrogate for the true policy gradient. Thus (by a simple inductive argument), in the general setting we may in fact focus on convergence according to the objectives $K_1,\ldots,K_m$ as opposed to $\tilde{J}_1,\ldots,\tilde{J}_m$, safe in the knowledge that a lexicographic optimum with respect to the former is sufficient for a lexicographic optimum with respect to the latter.

**Proof.** The proof proceeds by induction on $i \in \{1,\ldots,m\}$ for some fixed $m$, using a multi-timescale approach. Due to the learning rates chosen, we may consider those more slowly updated parameters fixed for the purposes of analysing the convergence of the more quickly updated parameters [Borkar, 2008]. Thus, for the updates to each vector $w_i$, the parameter $\theta$ and $\lambda$ may be viewed as fixed. By assumption, under conditions $C$ we have that this process thus converges to some fixed $w_i^*(\theta)$ for each $w_i$. 

---

9In the following remarks and proof, for ease of exposition, we focus primarily on the case of a value rather than action-value critic, but analogous arguments apply in the case of the latter.
Let us next consider, as the base case $i = 1$ for our inductive argument, the convergence of $\theta$ with respect to the Lagrangian $L_1(\theta, \lambda)$ objective. As all of the critic parameters are updated on a faster timescale we may view them as having converged for the purpose of our analysis. Further, by the above argument we have that each $w_i = w_i^*(\theta)$. Similarly, as the updates to $\theta$ with respect to $L_j(\theta, \lambda)$ for $j > i$ occur at a slower timescale then we may view $\theta$ as static with respect to these latter updates. Note that at this step we may also ignore updates to the sets of Lagrange multipliers $\lambda$ as they do not feature in $L_i(\theta, \lambda) = K_i(\theta)$. As such, this reduces to the case with $m = 1$ where, by assumption, we that $\theta$ converges to either a local or global optimum of $K_1$ (and thus an $\epsilon$-optimum, for $\epsilon_1 = \epsilon_{w_1}$, of $J_i$) under conditions $C$, as required.

Let us next assume as our inductive hypothesis, not only that $\theta$ has converged with respect to objectives $K_1, \ldots, K_{i-1}$ (as entailed by of our choice of learning rates), but has converged, under conditions $C$, to some policy $\pi^*$, so may view each of them as static. Thus, when using a critic or when estimating the gradient using another unbiased method (such as Monte Carlo estimates generated from trajectories) then $E_\theta[\tilde{K}_i(\theta)] = K_i(\theta)$ for each $i \in \{1, \ldots, m\}$ and so:

$$E_\theta[\tilde{K}_i(\theta)] = K_i(\theta) = \sum_{j=0}^{i-1} \lambda_j \nabla_{\theta} \tilde{K}_j(\theta) = \nabla_{\theta} K_i(\theta)$$

In other words, the update rule (1) uses an unbiased estimate of the gradient of the Lagrangian $L_i$ with respect to $\theta$ and can thus be seen as a discretisation of the ODE:

$$\dot{\theta} = \Gamma_\theta(\nabla_{\theta} L_i(\theta^t, \lambda)),$$

where $t$ indexes the values of $\theta$ over time and the projection operator $\Gamma_\theta$ ensures that iterates governed by this ODE remain in a compact set.

Given that gradient descent on $K_1$ converges to a locally or globally optimal stationary point when $m = 1$, then it must be the case that $K_1$ is either locally or globally invert respectively [Ben-Israel and Mond, 1986], where recall that a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is (globally) invert if and only if there exists a function $g : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ such that $f(x_1) - f(x_2) \geq g(x_1, x_2) \cdot \nabla f(x_2)$ for all $x_1, x_2 \in \mathbb{R}^n$ [Hanson, 1981], with a generalisation to local invert by straightforward [Craven, 1981]. Further, as each objective $K_i$ is of the same form (by assumption) then each $K_i$ is similarly locally or globally invert, and thus so too is each $L_i$ in $\theta$, being a linear combination of $K_1, \ldots, K_{i-1}$ with the addition of a single scalar term. Hence, the stationary point $\theta^*(\lambda)$ of the above ODE for fixed Lagrange multipliers $\lambda$ is such that $L_i(\theta^*(\lambda), \lambda)$ is is either a local or global maximum of $L_i$ respectively.

What remains is to consider the convergence of said Lagrange multipliers at a slower timescale given by $\eta = \eta_i$. As the actor and critics are updated at a faster timescale we may again view them as having converged to $\theta^*(\lambda)$ and $w_i^*(\theta^*)$ respectively. Similarly to above we have that:

$$E_\theta [\tilde{K}_j(\theta; \tau_1 - K_j(\theta)) = k_j - \tau - K_j(\theta) = \nabla_{\lambda} L_i(\theta, \lambda),$$

for each $j \in \{1, \ldots, m - 1\}$. As such, the gradient update in (2) uses unbiased estimates and thus forms a discrete approximation of the following ODE:

$$\lambda_j^t = -\nabla_{\lambda} L_i(\theta(\lambda^t), \lambda^t),$$

where $t'$ indexes the values of each $\lambda_j$ over time, $\theta(\lambda^t)$ is the limit of the $\theta$ recursion with static parameters $\lambda^t$, and $\Gamma_\lambda$ ensures that $\lambda > 0$. As shown above, $\theta^t(\lambda^t) \to \theta^*$, and by a similar argument we have $\lambda_j^t(\theta^*) \to \lambda_j^*$, where $\lambda_j^t(\theta)$ is the limit of the $\lambda_j^t$ recursion with static parameters $\theta$ and thus that $L_i(\theta^*, \lambda^t)$ is a global minimum (as $L_i$ is convex in $\lambda$), given $\theta^*$.

Thus, $(\theta^*, \lambda^*)$ forms a local or global saddle point of the Lagrangian $L_i$, representing a solution to the dual formulation of our original constrained optimisation problem. Furthermore, it can be shown that (given Slater’s condition) the duality gap between this solution and a locally or globally optimal solution to the primal formulation is bounded above by [Patermain et al., 2019]:

$$\epsilon_\theta := \epsilon_{\theta} \cdot \left( \max_{s,a} R_i(s, a) + \|\lambda\|_1 \max_{j < i, s,a} R_j(s, a) / (1 - \gamma_j) \right),$$

where:

$$\lambda_\epsilon = \arg \min_{\lambda > 0} \max_{\theta} L_i(\theta, \lambda) - \xi \sum_{j=1}^{i} \lambda_j$$

$$\xi = \max_{j < i, s,a} \epsilon_{\theta} \cdot \frac{R_j(s, a)}{1 - \gamma_j}.$$
1. $S$ and $A$ are finite, and each reward function $R_i$ is bounded.

2. For any $\theta \in \Theta$, the induced Markov chain over $S$ is irreducible.

3. For any $s \in S$ and $a \in A$, $\pi(a \mid s; \theta)$ is twice continuously differentiable.

4. Letting $\Phi$ be the $|S| \times c$ matrix with rows $\phi(s)$, then $\Phi$ has full rank (i.e. the features are independent), $c \leq |S|$, and there is no $w \in W$ such that $\Phi w = 1$.

Then for any MOMDP with discounted or limit-average objectives, LA2C almost surely converges to a policy in $\Theta\epsilon^m$.

Proof. Note that the critic updates are equivalent to the well-known Linear Semi-Gradient TD(0) algorithm [Sutton and Barto, 2018] under the state distribution $d_i^\theta$ that depends on $\gamma_i$,\(^10\) given by:

$$d_i^\theta(s) := \lim_{T \to \infty} \frac{1}{\gamma_i^T} \sum_{t=0}^{T-1} \gamma_i^t \Pr(s_t = s \mid s_0, \theta) \Phi(s_0).$$

Under the assumed conditions this algorithm is known to converge with probability one to some $w_i^*(\theta)$ where $V_i = \Phi w_i^*(\theta)$ is the (unique) TD fixed point minimising the mean squared Bellman error. In the discounted case, see [Tsitsiklis and Van Roy, 1997] for the original proof (Theorem 1) and [Sutton and Barto, 2018] for a more recent exposition (Section 9.4). In the limit-average case see, for one particularly clear and concise proof, Lemma 5 in [Bhatnagar et al., 2009].

Similarly, under the conditions listed above the standard A2C algorithm is known to converge asymptotically to a local $\epsilon_{\omega_1}$-optimum $J_1$ when $m = 1$. See [Bhatnagar et al., 2009] for the limit-average setting, the proof for which can be readily adapted to the discounted setting assuming that proper discounting is applied to the various gradient updates in order to reflect $d_i^\theta$, and that the definition of $J_1$ is based on a specific initial distribution $I$, rather than the steady state distribution under the policy $\theta$ [Sutton et al., 1999; Thomas, 2014]. With these facts in hand we may apply Theorem 3 to obtain our result.

$\square$

Corollary 2. Let $\pi(a \mid s; \theta, \chi) \propto \exp(\chi^{-1} f(s, a; \theta))$ and supposed that both $f$ and the action-value critic $Q_i$ are parametrised using two-layer neural networks (where $\chi$ is a temperature parameter), that a semi-gradient TD(0) rule is used to update $Q_i$, and that $Q_i$ replaces $A_i$ in the standard PPO loss $\mathcal{L}_{\text{PPO}}$, both of which updates use samples from the discounted steady state distribution. Further, let us assume that:

1. $S$ is compact and $A$ is finite, with $S \times A \subseteq \mathbb{R}^d$ for some finite $d > 0$, and each reward function $R_i$ is bounded.

2. The neural networks have widths $\mu_f$ and $\mu_Q$, respectively with ReLU activations, initial input weights drawn from a normal distribution with mean 0 and variance $\frac{1}{d}$, and initial output weights drawn from $\text{unif}([-1, 1])$.

3. We have that $q^\pi_i(\cdot, \cdot) \in \{Q_i(\cdot, \cdot; w_i) \mid w_i \in \mathbb{R}^y\}$ for any $\pi \in \Pi$.

4. There exists $c > 0$ such that for any $z \in \mathbb{R}^d$ and $\zeta > 0$ we have that $\mathbb{E}_{\pi} \left[ 1(|z^\top(s, a) - \zeta|) \right] \leq \frac{c}{\zeta^2}$ for any $\pi \in \Pi$.

Then for any MOMDP with discounted objectives, LPPO almost surely converges to a policy in $\Theta\epsilon^m$. Furthermore, if the coefficient of the KL divergence penalty $\kappa > 1$ then $\lim_{\mu_f, \mu_Q, \epsilon \to \infty} \epsilon = 0$.

Proof. Our proof makes use of recent results about the global optimality of PPO when using over-parametrised networks. See [Liu et al., 2019] for the original detailed result that we make use of here and [Hsu et al., 2020] for a more intuitive exposition based on the theory of multiplicative weights. Our arguments are essentially identical to those found in [Liu et al., 2019] with minor adaptions to accommodate: a) our use of diminishing rather than constant learning rates; b) our focus on asymptotic convergence; and c) our assumption that samples are taken from the discounted steady state distribution $d_i^\theta(s)$ defined in the previous proof. We refer the reader to this paper for a full exposition.

The aforementioned result relies on bounding the error for a generic network $u(\cdot, \cdot; \omega)$ of the same form as that representing $f$ and each $Q_i$ based on the updates given by:

$$\omega_{t+1} \leftarrow \omega_t + \nabla_{\omega} \mathbb{E}_{\nu} \left[ (u(s, a; \omega_t) - g(s, a) - h \cdot u(s', a'; \omega_t)) \right]$$

for a fixed learning rate $\epsilon$, and some fixed function $g$ and constant $h$, where $\Gamma_{\nu} \omega' := \arg \min_{\omega \in B^0(r_u)} \|\omega' - \omega\|_2$ projects $\omega'$ to the ball $B^0(r_u) := \{\omega \mid \|\omega - \omega_0\|_2 \leq r_u\}$ of radius $r_u$ about the initial parameters $\omega_0$. If we instead replace $\epsilon$ with a series of learning rates $\{\epsilon_t\}_{t \in \mathbb{Z}}$ such that $\epsilon_t \in [0, 1]$, $\sum_{t=0}^{\infty} \epsilon_t = \infty$, and $\sum_{t=0}^{\infty} (\epsilon_t)^2 < \infty$ then at time $t$ we have the following error bound:

$$\mathbb{E}_{\nu} \left[ (u(s, a; \omega_t') - u^0(s, a; \omega^*) \right)^2 \left| \omega_t \right| \leq \frac{1}{\epsilon_t(1-h) - 4\epsilon_t \left( \|\omega_t - \omega^*\|_2^2 - \mathbb{E}_{\nu} \left[ \|\omega_{t+1} - \omega^*\|_2^2 \right] \right)} + C_1(r_u) \epsilon_t^2 + C_2(r_u, \mu_u),$$

for some choice of distribution $\nu$ over $S \times A$, and where $u^0$ is the local linearisation of $u$ at $\psi_0$, $\omega^*$ is such that:

$$\mathbb{E}_{\nu} \left[ (u^0(s, a; \omega^*) - g(s, a) - h \cdot u^0(s', a'; \omega^*) \right)^\top (\omega - \omega^*) \geq 0,$$

for any $\omega \in B^0(r_u)$, and $C_1(r_u)$ and $C_2(r_u, \mu_u)$ are constants of size $O(r_u^2)$ and $O(r_u^{5/2} \mu_u^{1/4} + r_u^3 / \mu_u^{1/2})$, respectively. Then telescoping up to timestep $T$ and dividing by $\sum_{t=0}^{T-1} (\epsilon_t(1-h)}$.

\(^10\)Said dependence is often ignored in the literature, we refer the reader to [Thomas, 2014] for further discussion.
\[ h(t) = 4t^2 \] we have:
\[
\mathbb{E}_{\nu, I_w, t} \left[ \left( u(s, a; \omega_t) - u^0(s, a; \omega^*) \right)^2 \right]
\leq \left( \mathbb{E}_{I_w} \left[ \| \omega_0 - \omega^* \|^2 + \sum_{t=0}^{T-1} C_r(r_u) t^2 \right] \right) + C_2(r_u, \mu_u) + \sum_{t=0}^{T-1} \left( u(t(1-h) - 4t^2) \right)
\leq \left( r_u + \sum_{t=0}^{T-1} C_1(r_u) t^2 \right) + C_2(r_u, \mu_u),
\]
where \( t \) is sampled from \([0, T - 1]\) with probability \( \epsilon_0(t(1-h) - 4t^2) \), \( \omega_0 \) is sampled from \( I_w \) (where \( I_w \) is the distribution described by condition 2 of the corollary), and in the second inequality we use the fact that \( \| \omega_0 - \omega^* \|^2 \leq r_u \). By using different instantiations of \( g, h, r_u, \) and \( t \), we may instantiate the update rules for both \( w_1 \) and \( \theta \), as described in [Liu et al., 2019].

Let the error bounds for these cases be given by \( \epsilon_{Q_1}(T) \) and \( \epsilon_f(T) \) respectively, and note that due to the conditions on \( \alpha \) and \( \beta \) then we have \( \lim_{T \to \infty} \epsilon_{Q_1}(T) = C_2(r_Q, \mu_Q) \) and \( \lim_{T \to \infty} \epsilon_f(T) = C_2(r_f, \mu_f) \) respectively. Supposing that \( \kappa \) and \( \alpha \) are updated every \( T \) steps (where recall that \( \alpha \) is the coefficient of the KL divergence penalty term in the PPO update rule), and that this is repeated \( K \) times, then we have the following bound:
\[
J_1(\theta^*) - \mathbb{E}_{\theta_k} \left[ J_1(\theta_k) \right] \leq \epsilon_1 \cdot \frac{\log |A| + M \sum_{k=0}^{K-1} (\kappa_k^{-2} + \epsilon_k)}{(1 - \gamma_1) \sum_{k=0}^{K-1} \kappa_k^{-1}},
\]
where \( k \) is sampled from \([0, K - 1]\) with probability \( \kappa_k^{-1} \) and we have:
\[
M := 2 \left( \mathbb{E}[\max_{a \in A} Q(s, a; w_0)^2] + \sum_{f} r_f^2 \right),
\]
\[
\epsilon_k := \chi_{k+1}^2 \epsilon_f(T) \phi_k^0 + \chi_k^{-1} \epsilon_{Q_1}(T) \omega_k^0 + |A| \chi_{k+1}^{-1} \epsilon_f(T)^2,
\]
for constants \( \phi_k^0 \) and \( \omega_k^0 \) that represent density ratios – see equation 4.2 in [Liu et al., 2019] for their precise definitions. Letting \( \epsilon_1 := \lim_{T \to \infty} \epsilon_1 \) allows us to apply Theorem 3, where note that if \( \kappa_k > 1 \) then for each \( i \) we have:
\[
\mu_{Q_i}, \mu_f \to \infty \implies \epsilon_i \to 0,
\]
which thus results in a global lexicographic optimum.

10 Hyperparameters

In this section we list the hyperparameters used in Section 4, as well as other noteworthy implementation details.

In Section 4.1, the VB-LRL algorithms are tabular, with a learning rate of \( \alpha_t = 0.01 \). All VB-LRL algorithms use Lexicographic \( \epsilon_t \) Greedy with \( \alpha_t = 0.05 \), and slack \( \tau(s, i) = \xi(s, i, t) = 0.01 \cdot \max_{a \in A_{\tau}} Q_1(s, a) \). The PB-LRL algorithms take the state as a one-hot encoded vector, and use fully-connected neural networks without hidden layers or bias terms. The code used to generate the MOMDPs is based on MDPToolBox\(^\text{11}\) but with Gaussian noise (\( \sigma = 0.2 \)) added to the reward.

In the CartSafe environment, RCPO’s constraint was \( J_2(\pi) \leq 1 \), AproPO’s constraints were \( J_1(\pi) \in [90, 350] \), \( J_2(\pi) \in [0, 10] \), and VaR-AC’s constraint was to keep the cost under 10 with probability at least 0.95. In the GridNav environment, RCPO’s constraint was \( J_2(\pi) \leq 5 \), AproPO’s constraints were \( J_1(\pi) \in [90, 100] \), \( J_2(\pi) \in [0, 5] \), and VaR-AC’s constraint was to keep the cost under 5 with probability at least 0.95. Finally, in the Intersection environment, RCPO’s constraint was \( J_2(\pi) \leq 1 \), AproPO’s constraints were \( J_1(\pi) \in [9, 10] \), \( J_2(\pi) \in [0, 1] \), and VaR-AC’s constraint was to keep the cost under 1 with probability at least 0.95. The lexicographic algorithms were trying to minimise cost and, subject to that, maximise reward. Each algorithm was run ten times in each environment.

In Section 4.2, we use VB-LRL with Lexicographic \( \epsilon_t \)-Greedy with \( \epsilon_t = 0.05 \), Lexicographic Q-Learning, and slack \( \tau(s, i, t) = \xi(s, i, t) = 0.05 \cdot \max_{a \in A_{\tau}} Q_1(s, a) \) in GridNav and Intersection, and 0.001 \cdot \max_{a \in A_{\tau}} Q_1(s, a) \) in CartSafe. All agents use fully connected networks, and for the algorithms with parameterised policies, separate networks were used for prediction and control. AproPO’s best-response oracle was a DQN algorithm. In GridNav and Intersection, the algorithms update their networks every 32 steps, and use 32 as their batch size, but in CartSafe they update every 8 steps, and have a batch size of 8. DQN and LDQN have a replay buffer of size 100,000, and AproPO performs an update every 8 episodes in GridNav and Intersection, and every episode in CartSafe. The networks were updated with the Adam optimiser, with a learning rate of 0.001 in CartSafe and Intersection, and 0.01 in GridNav. In the CartSafe environment the agents use networks with two hidden layers with 24 neurons each, in the GridNav environment they use networks without hidden layers or bias terms, and in the Intersection environment they use networks with two hidden layers with 512 neurons each.

11 A Hyperparameter Study for VB-LRL’s Slack

Here we present some additional data that investigates the practical significance of VL-LRL’s slack parameter. In particular, we run LDQN in CartSafe, and vary \( \tau(s, i) = \xi(s, i, t) = 0.01 \cdot \max_{a \in A_{\tau}} Q_1(s, a) \) from 0 to 1 in steps of 0.2. The result of this experiment is shown in Figure 3. We can see that the choice of slack parameter does have an impact on the performance of the algorithm, but that this impact is fairly gradual. The hyperparameters used in this experiment were similar to those used for the experiment in Section 4.2.

12 A Comparison of Different Settings of PB-LRL

Here we present data that compares different settings of PB-LRL. Specifically, we run LPPO and LA2C in CartSafe,\(^{11}\)Available at https://github.com/sawcordwell/pymdptoolbox.
Figure 3: This experiment studies the impact of LDQN’s tolerance parameter in the CartSafe environment.

Figure 4: This experiment compares different versions of PB-LRL in the CartSafe environment.

with and without second-order derivatives in the loss, and with and without sequential optimisation of the objectives, as opposed to the simultaneous method presented in the paper. We also include LDQN for comparison. The result of this experiment is shown in Figure 4. We can see that these versions of PB-LRL all are quite similar in terms of the cost they attain, and that LA2C achieves higher reward than LPPO. Moreover, it seems that the use of second-order terms or sequential optimisation has an impact on the performance, but that this impact is not that large. The other hyperparameters used in this experiment were similar to those used for the experiment in Section 4.2.

13 Policy Visualisations

Here we present some additional experimental data. Figure 5 displays visualisations of the policies learned by all the algorithms in the CartSafe environment during the experiment presented in Section 4, and Figure 6 does the same for the GridNav environment. We have not created any visualisations of the policies learned in the Intersection environment, since it would be difficult to display such a policy as a static 2D plot.

Looking at the plots for the CartSafe environment, we can see that A2C and VaR_AC seem to reliably learn to simply go straight to the right or straight to the left. LA2C and LPPO are good at staying inside the safe region, but also don’t get a lot of reward. LDQN learns policies that generate more reward, but sometimes strays outside the safe region, and RCPO will sometimes learn policies that are safe but overly cautious, and sometimes learn policies that get more reward, but sometimes go outside the safe region. Note that this environment allows for policies that get high reward, but never stray outside the safe region.

Looking at the plots for the GridNav environment, we can see that LA2C, LPPO, and VaR_AC all seem to mostly reach the goal. LDQN mostly stays around the start square, without reaching the goal. RCPO wanders around the board a lot, but only sometimes reaches the goal. AproPO seems to have a preference for going in the same direction, and sometimes reaches the goal.
Figure 5: This figure displays visualisations of the policies learned by the algorithms in the CartSafe environment. Each plot corresponds to a policy learned by an algorithm, and is labelled with the random seed used in that run of the experiment. They were generated by running each policy ten times, and drawing trajectory of the cart body (in green) and the tip of the pole (in orange).

Figure 6: This figure displays visualisations of the policies learned by the algorithms in the GridNav environment. Each plot corresponds to a policy learned by an algorithm, and is labelled with the random seed used in that run of the experiment. They were generated by running each policy ten times, and drawing the path of the algorithm through the environment (in orange).