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, e.g. [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 [Miryoosfet 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, in general, be the same. Other methods also emphasise staying safe while learning; see e.g. [Achiam et
al., 2017; Thomas et al., 2013; Polymenakos et al., 2019]. In contrast, our algorithms do not guarantee safety while learning, but rather learn a safe limit policy.

2 Background

Reinforcement Learning. The RL setting is usually formalised as a Markov Decision Process (MDP), which is a tuple \((S, A, T, I, R, \gamma)\) where \(S\) is a set of states, \(A\) is a set of actions, \(T : S \times A \rightarrow S\) is a transition function, \(I\) is an initial state distribution over \(S\), \(R : S \times A \times S \rightarrow \mathbb{R}\) 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 policy \(\pi\) from \(s\), i.e. \(v_\pi(s) := \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s]\). When \(\gamma = 1\), we instead consider the limit-average of this expectation. The objective in RL can then be expressed as maximising \(J(\pi) := \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]\). Given a policy \(\pi\) we may also define the \(Q\)-function \(Q_\pi(s, a) := \mathbb{E}_{\pi \sim T\pi}[R(s, a, s') + v_\pi(s')]\) 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 bandit algorithm 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]:

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

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 \(Q(s_{t+1}, a_{t+1})\) or \(\mathbb{E}_{a_t \sim \pi}(Q(s_{t+1}, a))\) (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 \(K^{AC}(\theta) := J(\theta)\) then we may express this using:

\[
K^{AC}(\theta) := \mathbb{E}_t \left[ \log \pi_\theta(s_t) \cdot A_\theta(s_t, a_t) \right],
\]

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 \(w \in W \subset \mathbb{R}\), and using the fact that the expected temporal difference error \(d_t := r_t + \gamma v_\theta(s_{t+1}) - v_\theta(s_t)\) (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:

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

where \(\kappa\) 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 \((S, A, T, I, R, \gamma, \epsilon)\) 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 \(\epsilon \in [0, 1]^m\) defines \(m\) discount rates. We define \(R_t\) as \((s, a, s') \mapsto R(s, a, s')\).

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 (globally) lexicographically \(\epsilon\)-optimal if \(\pi \in \Pi^\epsilon\), where \(\Pi_0 = \Pi\) is the set of all policies in \(M\), \(\Pi_{t+1} := \{\pi \in \Pi_t \mid \max_{\pi' \in \Pi_t} J_t(\pi') - J_t(\pi) \leq \epsilon_t\}\), and \(\mathbb{R}^{m-1} \ni \epsilon \geq 0\). We similarly write \(\Theta_{t+1}^\epsilon\) to define global lexicographic \(\epsilon\)-optima for parameterised policies, but also \(\tilde{\Theta}_{t+1}^\epsilon = \{\theta \in \Theta_t \mid \max_{\theta' \in \tilde{\Theta}_{t+1}^\epsilon} J_t(\theta') - J_t(\theta) \leq \epsilon_t\}\) to define local lexicographic \(\epsilon\)-optima, where \(\tilde{\Theta}_{t+1}^\epsilon \subset \Theta_t^\epsilon\) is a compact local neighbourhood of \(\theta\), and \(\Theta_0^\epsilon = \Theta_0^\epsilon = \Theta\). When \(\epsilon = 0\) we drop it from our notation and refer to lexicographic optima and lexicographically optimal policies simpliciter.

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

\[
\lim_{t \rightarrow \infty} \text{Pr}(B(Q_1, \ldots, Q_m, s, t) \in \Delta^\tau_{s,0}) = 1,
\]

where \(\Delta_{s,0}^\tau = A\) and \(\Delta_{s,\tau} = \{a \in \Delta^\tau_{s,i} \mid Q_i(s, a) \geq \max_{a' \in \Delta^\tau_{s,i}} Q_i(s, a') - \tau\}\).

\footnote{Due to the choice of baseline [Sutton et al., 1999], we describe here the classic Advantage Actor-Critic (A2C) algorithm.}
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 \( B \) 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:

\[
Q_i(s,a) := (1-\alpha_i(s,a)) \cdot Q_i(s,a) + \alpha_i(s,a) \cdot \left( R_i(s,a,s') + \gamma_i \max_{a' \in A_{\tau, i}} Q_i(s',a') \right),
\]

where \( \Delta_{s,0}^\tau = A_i, \Delta_{s,t+1}^\tau := \{ a \in \Delta_{s,t}^\tau \mid Q_i(s,a) \geq \max_{a' \in A_{\tau, i}} Q_i(s,a') - \tau \}, \) and \( \tau \in \mathbb{R}_{>0} \) is the tolerance parameter. 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:

\[
Q_A(s,a) := (1-\alpha_A(s,a)) \cdot Q_A(s,a) + \alpha_A(s,a) \cdot \left( R_A(s,a,s') + \gamma_A \max_{a' \in A_{\tau, A}} Q_A(s',a') \right),
\]

and else perform the analogous update on \( Q_B \), and let \( Q_i(s,a) := 0.5(Q_A(s,a) + Q_B(s,a)) \) in the bandit algorithm. Varying the bandit algorithm or \( Q \)-value update rule in VB-LRL produces a family of algorithms with different properties.

We can now give our core result for Algorithm 2. All our proofs are included in the supplementary material.

**Theorem 1.** In any MOMDP \( \mathcal{M} \), if VB-LRL uses a lexicographic bandit algorithm and either 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 \( \alpha_i(s,a) \in [0,1] \) 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 \( 0 < \tau < \min_{i,s,a \neq a'} |q_i(s,a) - q_i(s,a')| \).

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 SARSA, 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.

---

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

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

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

**Algorithm 2 Value-Based Lexicographic RL**

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

1. initialise \( Q_1, \ldots, Q_m, \ t \leftarrow 0, \ s \sim I \)
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 \( \pi = s \rightarrow \lim_{t \to \infty} B(Q_1, \ldots, Q_m, s, t) \)
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 $j \in \{1, \ldots, i-1\}$. Then we wish to solve the constrained optimisation problem given by:

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

where $\tau > 0$ is a small constant tolerance parameter, included such that there exists some $\theta$ 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 $\alpha \in \{\alpha, \beta^1, \ldots, \beta^m, \eta^0, \ldots, \eta^m\}$ and all $i \in \{1, \ldots, m\}$ we have:

\[
\begin{align*}
t_i & \in [0, 1], \\
\sum_{t=0}^{\infty} t_i = \infty, \\
\sum_{t=0}^{\infty} (t_i)^2 & < \infty \quad \text{and}
\end{align*}
\]

\[
\lim_{t \to \infty} \frac{\beta_i^t}{\alpha_t} = \lim_{t \to \infty} \frac{\eta_i^t}{\alpha_t} = \lim_{t \to \infty} \frac{\beta_i^t}{\eta_i^t} = 0,
\]

we also assume that $\tau_i = o(\beta_i^t)$ 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:

\[
\begin{align*}
\theta & \leftarrow \Gamma_0 \left[ \theta + \beta_i^t \left( \nabla_\theta \hat{K}_i(\theta) + \sum_{j=1}^{i-1} \lambda_j \nabla_\theta \hat{K}_j(\theta) \right) \right], \\
\lambda_j & \leftarrow \Gamma_1 \left[ \lambda_j + \eta_t (\hat{k}_j - \tau_t - \hat{K}_j(\theta)) \right] \quad \forall j \in \{1, \ldots, i-1\},
\end{align*}
\]

where $\Gamma_i(\cdot) = \max(\cdot, 0), \Gamma_0$ projects $\theta$ to the nearest point in $\Theta$, and $\tau$ is used to denote a Monte Carlo estimate. We next note that by collecting the terms involved in the updates to $\theta$ for each $i$, at time $t$ we are effectively performing the simple update $\theta \leftarrow \Gamma_0[\theta + \nabla_\theta \hat{K}(\theta)]$, where:

\[
\hat{K}(\theta) := \sum_{i=1}^{m} c_i^t \hat{K}_i(\theta) \quad \text{and} \quad c_i^t := \beta_i^t + \lambda_i \sum_{j=i+1}^{m} \beta_j^t,
\]

and where we assume that $\sum_{i=1}^{m} \beta_i^t = 0$. It is therefore computationally simple to formulate a lexicographic optimisation problem from any collection of objective functions by updating a small number of coefficients $c_i^t$ at each timestep and then linearly combining the objective functions.

Finally, in many policy-based algorithms we use a critic $V_i$ (or $Q_i$) to estimate each $\hat{K}_i$ and so must also update the parameters $w_i$ of each critic. This is typically done on a faster timescale using the learning rate $\alpha_i$, for instance via the TD(0) update for $V_i$ given by $w_i \leftarrow w_i + \alpha_i(\delta_i^t \nabla_{w_i} V_i)$, 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 $K_1^{\text{A2C}}$, or PPO when using $K_1^{\text{PPO}}$. 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 $w_i^*(\theta) = \lim_{T \to \infty} E_t[w_i]$ 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 $\lim_{T \to \infty} E[V_i] \in \Theta^*_i$ (respectively $\tilde{\Theta}^*_i$) under conditions $\mathcal{C}$ when $m = 1$, then for any fixed $m \in \mathbb{N}$ we have that $\lim_{T \to \infty} E[V_i] \in \Theta^*_m$ (respectively $\tilde{\Theta}^*_m$), 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 $K^{A2C}_a$ and $K^{PPO}_b$ 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 $V_i(s) = w_i^T \phi(s)$ 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), 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 $\tilde{\Theta}^*_m$.

**Corollary 2.** Let $\pi(a \mid s; \theta, \chi) \propto \exp \left( \chi^{-1} f(s, a; \theta) \right)$ and suppose that both $f$ 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 $K^{PPO}_b$, 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 $q_i^\pi(\cdot, \cdot) \in \{Q_i(\cdot, \cdot; w_i) \mid w_i \in \mathbb{R}^p\}$ 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 $E[1(|z^T(s, a)| \leq \zeta)] \leq \frac{K^e}{\|z\|^2}$ for any $\pi \in \Pi$.

Then for any MOMDP with discounted objectives, LPPO almost surely converges to a policy in $\tilde{\Theta}^*_m$. Furthermore, if the coefficient of the KL divergence penalty $\kappa > 0$ 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 [Miryoossefi 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\(^6\) 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.

\(^6\)Available at https://github.com/lrhammond/lmorl.

\(^{6}\)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\(^7\), 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


\(^7\)Available at https://github.com/eleurent/highway-env.


