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 $R_1, \dots, R_m$, and that learn a policy π such that π maximises expected discounted R_1-reward, and among all policies that do so, π 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*

[*]Contact Author

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 $\langle S, A, T, I, R, \gamma \rangle$ where S is a set of states, A is a set of actions, $T : S \times A \rightsquigarrow S$ is a *transition function*, I is an initial state distribution over S, $R : S \times A \times S \rightsquigarrow \mathbb{R}$ 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, $f : X \rightsquigarrow Y$ 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 $\pi : S \rightsquigarrow A$ that specifies a distribution over the agent's actions in each state. The *value function* $v_\pi(s)$ of π is defined as the *expected γ-discounted cumulative reward* when following π 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 $J(\pi) := \sum_s I(s) v_\pi(s)$. Given a policy π we may also define the *q-function* $q_\pi(s, a) := \mathbb{E}_{s' \sim T(s,a)} \left[R(s, a, s') + v_\pi(s')\right]$ 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* $Q : S \times A \to \mathbb{R}$ 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]:

$$\begin{aligned} Q(s_t, a_t) \leftarrow & \big(1 - \alpha_t(s_t, a_t)\big) \cdot Q(s_t, a_t) \\ & + \alpha_t(s_t, a_t) \cdot \big(r_t + \gamma \max_a Q(s_{t+1}, a)\big), \end{aligned}$$

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 $\mathbb{E}_{a \sim \pi(s)}[Q(s_{t+1}, a)]$ (where π 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 $\theta \in \Theta \subset \mathbb{R}^x$, and θ is updated according to an objective $K(\theta)$. If using $K^{\text{A2C}}(\theta) := J(\theta)$ then we may estimate this using:

$$K^{\text{A2C}}(\theta) := \mathop{\mathbb{E}}_t \big[\log \pi(a_t \mid s_t; \theta) \cdot A_\theta(s_t, a_t)\big],$$

where A_θ is an estimate of a_θ. One often computes A_θ by approximating v_θ with a function V parameterised by some $w \in W \subset \mathbb{R}^y$, 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].[1]

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:

$$\begin{aligned} K^{\text{PPO}}(\theta) := \mathop{\mathbb{E}}_t \Big[& \frac{\pi(a_t \mid s_t; \theta)}{\pi(a_t \mid s_t; \theta_{\text{old}})} A_\theta(s_t, a_t) \\ & - \kappa \cdot \mathrm{D}_{\text{KL}}(\pi(s_t; \theta) \parallel \pi(s_t; \theta_{\text{old}})) \Big], \end{aligned}$$

where κ 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 $\langle S, A, T, I, \mathfrak{R}, \gamma \rangle$ that is defined analogously to an MDP, but where $\mathfrak{R} : S \times A \times S \rightsquigarrow \mathbb{R}^m$ returns a vector of m rewards, and $\gamma \in [0, 1]^m$ defines m discount rates. We define R_i as $(s, a, s) \mapsto \mathfrak{R}(s, a, s)_i$.

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 $\mathcal{M}$ with m rewards, we say that a policy π is (globally) *lexicographically ϵ-optimal* if $\pi \in \Pi^\epsilon_m$, where $\Pi^\epsilon_0 = \Pi$ is the set of all policies in $\mathcal{M}$, $\Pi^\epsilon_{i+1} := \{\pi \in \Pi^\epsilon_i \mid \max_{\pi' \in \Pi^\epsilon_i} J_i(\pi') - J_i(\pi) \leq \epsilon_i\}$, and $\mathbb{R}^{m-1} \ni \epsilon \succcurlyeq 0$. We similarly write Θ^ϵ_{i+1} to define global lexicographic ϵ-optima for parametrised policies, but also $\tilde{\Theta}^\epsilon_{i+1} := \{\theta \in \Theta^\epsilon_i \mid \max_{\theta' \in \mathcal{N}_i(\theta)} J_i(\theta') - J_i(\theta) \leq \epsilon_i\}$ to define *local* lexicographic ϵ-optima, where $\mathcal{N}_i(\theta) \subseteq \tilde{\Theta}^\epsilon_i$ is a compact local neighbourhood of θ, and $\Theta^\epsilon_0 = \tilde{\Theta}^\epsilon_{-1} = \Theta$. 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, $Q_1, \ldots, Q_m : S \times A \to \mathbb{R}$ 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 $\mathcal{B} : (S \times A \to \mathbb{R})^m \times S \times \mathbb{N} \rightsquigarrow A$, such that

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

where $\Delta^\tau_{s,0} = A$ and $\Delta^\tau_{s,i+1} := \{a \in \Delta^\tau_{s,i} \mid Q_i(s, a) \geq \max_{a' \in \Delta^\tau_{s,i}} Q_i(s, a') - \tau\}$.

[1] 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 τ), and among all actions that do this, action a also maximises Q_2 (with tolerance τ), and so on. An example of a lexicographic bandit algorithm is given in Algorithm 1, where the exploration probabilities $\epsilon_{s,t}$ should satisfy $\lim_{t\to\infty} \epsilon_{s,t} = 0$ and $\sum_{t=0}^{\infty} \epsilon_{s,t} = \infty$ for all $s \in S$.

We can now introduce Algorithm 2 (VB-LRL), a value-based algorithm for lexicographic multi-objective RL. Here $\mathcal{B}$ 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*:

$$Q_i(s,a) \leftarrow \big(1 - \alpha_t(s,a)\big) \cdot Q_i(s,a) + \alpha_t(s,a) \cdot \big(R_i(s,a,s') + \gamma_i \max_{a \in \Delta^\tau_{s,i}} Q_i(s',a)\big),$$

where $\Delta^\tau_{s,0} = A$, $\Delta^\tau_{s,i+1} := \{a \in \Delta^\tau_{s,i} \mid Q_i(s,a) \geq \max_{a' \in \Delta^\tau_{s,i}} Q_i(s,a') - \tau\}$, 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_i^A, Q_i^B for each reward. To update the Q-values, with probability 0.5 we set:

$$Q_i^A(s,a) \leftarrow \big(1 - \alpha_t(s,a)\big) \cdot Q_i^A(s,a) + \alpha_t(s,a) \cdot \Big(R_i(s,a,s') + \gamma_i \cdot Q_i^B\big(s', \operatorname*{arg\,max}_{a' \in \Delta^\tau_{s,i}} Q_i^A(s',a')\big)\Big),$$

and else perform the analogous update on Q_i^B, and let $Q_i(s,a) := 0.5\big(Q_i^A(s,a) + Q_i^B(s,a)\big)$ 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 SARSA, Expected SARSA, or Lexicographic Q-Learning, then it will converge to a policy π that is lexicographically optimal if:*

1. *S and A are finite,*
2. *All reward functions are bounded,*
3. *Either $\gamma_1, \dots, \gamma_m < 1$, 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 τ 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 τ. 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 [Gábor *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 ϵ-Greedy

input: $Q_1, \dots, Q_m, s, t$
with probability $\epsilon_{s,t}$ **do** $a \sim \text{unif}(A)$
else
  $\Delta \leftarrow A$
  for $i \in \{1, \dots, m\}$ **do**
    $x \leftarrow \max_{a' \in \Delta} Q_i(s,a')$
    $\Delta \leftarrow \{a \in \Delta \mid Q_i(s,a) \geq x - \tau\}$
  $a \sim \text{unif}(\Delta)$
return a

Algorithm 2 Value-Based Lexicographic RL

input: $\mathcal{M} = \langle S, A, T, I, \mathfrak{R}, \gamma \rangle$
1: initialise $Q_1, \dots, Q_m$, $t \leftarrow 0$, $s \sim I$
2: **while** $Q_1, \dots, Q_m$ have not converged **do**
3:   $a \leftarrow \mathcal{B}(Q_1, \dots, Q_m, s, t)$ ▷ Algorithm 1
4:   $s' \leftarrow T(s,a)$
5:   **for** $i \in \{1, \dots, m\}$ **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** $\pi = s \mapsto \lim_{t\to\infty} \mathcal{B}(Q_1, \dots, Q_m, s, t)$

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 π 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 τ is small. We motivate this intuition with a formal guarantee about the behaviour of VB-LRL for arbitrary τ.

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 π_t is the policy at time t and π^* 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 θ using K_1, then at a slower timescale optimise θ 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 θ' lexicographically with respect to $K_1, \ldots, K_{i-1}$ and we wish to now lexicographically optimise θ 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:

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

where $\tau > 0$ is a small constant tolerance parameter, included such that there exists some θ 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 $\min_{\lambda \succcurlyeq 0} \max_\theta L_i(\theta, \lambda)$ of the Lagrangian relaxation [Bertsekas, 1999] where:

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

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 θ 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 η of the Lagrange multiplier to η^i after convergence with respect to the i^{th} objective, and assume that for all learning rates $\iota \in \{\alpha, \beta^1, \ldots, \beta^m, \eta^0, \ldots, \eta^m\}$ and all $i \in \{1, \ldots, m\}$ we have:

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

$$\lim_{t\to\infty} \frac{\beta^i_t}{\alpha_t} = \lim_{t\to\infty} \frac{\eta^i_t}{\beta^i_t} = \lim_{t\to\infty} \frac{\beta^i_t}{\eta^{i-1}_t} = 0.$$

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

$$\theta \leftarrow \Gamma_\theta\Big[\theta + \beta^i_t\Big(\nabla_\theta \hat{K}_i(\theta) + \sum_{j=1}^{i-1} \lambda_j \nabla_\theta \hat{K}_j(\theta)\Big)\Big],$$

$$\lambda_j \leftarrow \Gamma_\lambda\Big[\lambda_j + \eta_t\big(\hat{k}_j - \tau_t - \hat{K}_j(\theta)\big)\Big] \ \forall\, j \in \{1, \ldots, i-1\},$$

Algorithm 3 Policy-Based Lexicographic RL

input: $\mathcal{M} = \langle S, A, T, I, \mathfrak{R}, \gamma \rangle$
initialise $\theta, w_1, \ldots, w_m, \lambda_1, \ldots, \lambda_m$
$t \leftarrow 0, \quad \eta \leftarrow \eta^0, \quad s \sim I$
while θ has not converged **do**
  $t \leftarrow t+1, \quad a \sim \pi(s), \quad s' \sim T(s,a)$
  for $i \in \{1, \ldots, m\}$ **do**
    if $\hat{K}_i(\theta)$ has not converged **then** $\hat{k}_i \leftarrow \hat{K}_i(\theta)$
    else $\eta \leftarrow \eta^i$
    update w_i (if using a critic) and λ_i
  update θ
  if s' is terminal **then** $s \sim I$ **else** $s \leftarrow s'$
return θ

where $\Gamma_\lambda(\cdot) = \max(\cdot, 0)$, Γ_θ projects θ to the nearest point in Θ, and $\hat{}$ is used to denote a Monte Carlo estimate. We next note that by collecting the terms involved in the updates to θ for each i, at time t we are effectively performing the simple update $\theta \leftarrow \Gamma_\theta[\theta + \nabla_\theta \hat{K}(\theta)]$, 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 $\sum_{j=m+1}^{m} \beta^j_t = 0$. 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 α, for instance via the TD(0) update for V_i given by $w_i \leftarrow w_i + \alpha_t\big(\delta^i_t \nabla_{w_i} V_i\big)$, where δ^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) ϵ-optimum when $m = 1$ obtains a *lexicographically* local (or global) ϵ-optimum when the corresponding objective function is used in PB-LRL.

Theorem 3. *Let $\mathcal{M}$ be a MOMDP, π a policy that is twice continuously differentiable in its parameters θ, 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 θ is held* *fixed under some set of conditions $\mathcal{C}$ on $\mathcal{M}$, π, and each V_i. If $\lim_{T\to\infty}\mathbb{E}_t[\theta] \in \Theta_1^{\epsilon}$ (respectively $\tilde{\Theta}_1^{\epsilon}$) under conditions $\mathcal{C}$ when $m = 1$, then for any fixed $m \in \mathbb{N}$ we have that $\lim_{T\to\infty}\mathbb{E}_t[\theta] \in \Theta_m^{\epsilon}$ (respectively $\tilde{\Theta}_m^{\epsilon}$), where each $\epsilon_i \geq 0$ is a constant that depends on the representational power of the parametrisations of π (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_i^{A2C} and K_i^{PPO} 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 $V_i(s) = w_i^\top \phi(s)$ for some choice of state features ϕ and is updated using a semi-gradient TD(0) rule, and that:*

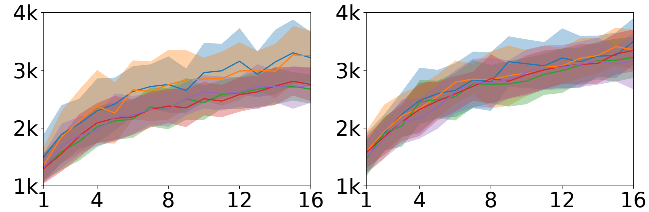
1. *S and A are finite, and each reward function R_i is bounded,*
2. *For any $\theta \in \Theta$, the induced Markov chain over S is irreducible,*
3. *For any $s \in S$ and $a \in A$, $\pi(a \mid s;\theta)$ is twice continuously differentiable,*
4. *Letting Φ be the $|S| \times c$ matrix with rows $\phi(s)$, then Φ 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 $\tilde{\Theta}_m^{\epsilon}$.

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 χ 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}, 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 μ_f and μ_{Q_i} 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* $\mathrm{unif}([-1,1])$,
3. *We have that $q_i^{\pi}(\cdot,\cdot) \in \left\{Q_i(\cdot,\cdot;w_i) \mid w_i \in \mathbb{R}^y\right\}$ 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 $\mathbb{E}_\pi\left[\mathbf{1}(|z^\top(s,a)| \leq \zeta)\right] \leq \frac{c\zeta}{\|z\|_2}$ for any $\pi \in \Pi$.*

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



(a) 256 states (b) 512 states

Figure 1: We plot the learning time of LRL as the number of episodes until convergence (y axis) against the number of reward signals (x axis). We use randomly generated MOMDPs with 256 or 512 states, four actions, and a varying number of rewards. For each trial we generate 30 MOMDPs, and record the number of episodes it takes for each agent's long-run average reward per episode to converge to a stable value. The algorithms are Lexicographic Q-Learning (blue), Lexicographic Expected SARSA (orange), Lexicographic Double Q-Learning (green), Lexicographic PPO (red), and Lexicographic A2C (purple).

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.[5]

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

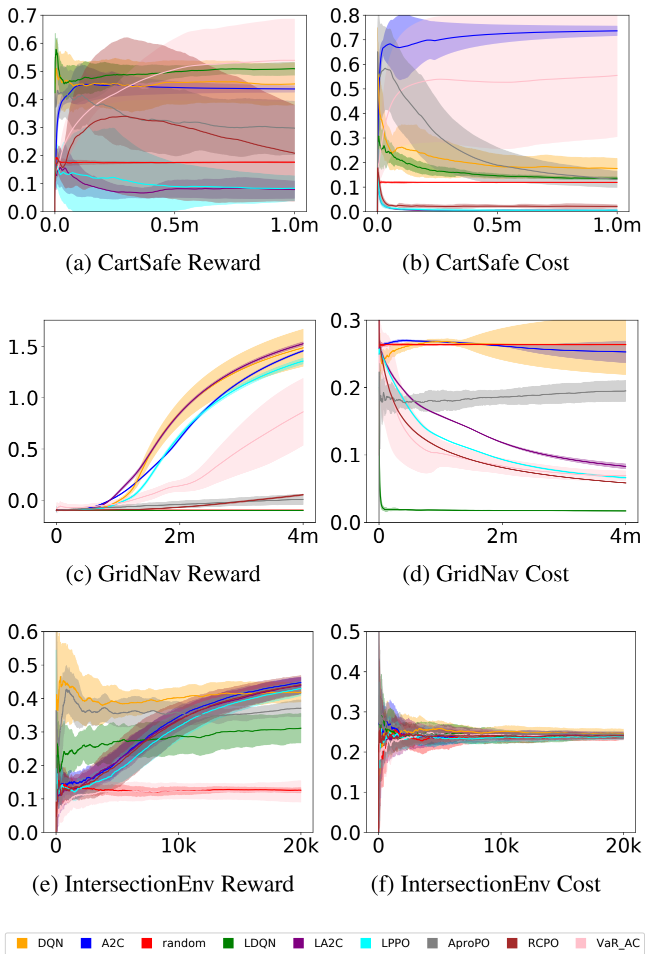
Many tasks are naturally expressed in terms of both a *performance metric* and a *safety constraint*. Our second experiment compares the performance of LRL against RCPO [Tessler *et al.*, 2019], AproPO [Miryoosefi *et al.*, 2019], and the actor-critic algorithm for VaR-constraints in [Chow *et al.*, 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*[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

[5] 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.



(a) CartSafe Reward (b) CartSafe Cost

(c) GridNav Reward (d) GridNav Cost

(e) IntersectionEnv Reward (f) IntersectionEnv Cost

Figure 2: We plot the average reward and cost (y axis) against the number of environment interactions (x axis). In each environment, RCPO, AproPO, and VaR_AC were tasked with maximising reward subject to a constraint on the cost, bounds on both the reward and cost, or a bound on the probability of the cost exceeding a certain constant. The LRL algorithms were tasked with minimising cost and, subject to that, maximising reward. Each algorithm was run ten times in each environment.

The GridNav environment, again from *gym-safety* (based on an environment in [Chow *et al.*, 2018]), is a large grid-world with a goal region and a number of "unsafe" squares. The agent is rewarded for reaching the goal quickly, and incurs a cost if it enters an unsafe square. Moreover, at each time step, the agent is moved in a random direction with probability 0.1. Here LDQN is the safest algorithm, but it also fails to obtain any reward. LA2C, LPPO, RCPO, and VaR_AC are similar in terms of safety, but LA2C and LPPO obtain the most reward, VaR_AC a fairly high reward, and RCPO a low reward. AproPO has low safety *and* low 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

[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.

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

[7] Available at https://github.com/eleurent/highway-env.

[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.

[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.

[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-Dünner, 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. Multi-objective 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, Zhuoran 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 Brantley, Hal Daumé 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.

[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.

[Roijers *et al.*, 2013] D. M. Roijers, P. Vamplew, S. Whiteson, 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 Mahesan 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.

[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.

[Tessler *et al.*, 2019] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[Thomas *et al.*, 2013] Philip S. Thomas, William Dabney, Sridhar Mahadevan, and Stephen Giguere. Projected natural actor-critic. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 2337–2345, 2013.

[van Seijen *et al.*, 2009] Harm van Seijen, Hado van Hasselt, Whiteson Shimon, and Marco Wiering. A theoretical and empirical analysis of expected sarsa. *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184, 2009.

[Watkins, 1986] Chris Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, 1986.

[Wray and Zilberstein, 2015] Kyle Hollins Wray and Shlomo Zilberstein. Multi-objective pomdps with lexicographic reward preferences. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1719–1725, 2015.