ReLOAD: Reinforcement Learning with Optimistic Ascent-Descent for Last-Iterate Convergence in Constrained MDPs

Ted Moskovitz [*1] **Brendan O'Donoghue** [2] **Vivek Veeriah** [2]
Sebastian Flennerhag [2] **Satinder Singh** [2] **Tom Zahavy** [2]

Abstract

In recent years, Reinforcement Learning (RL) has been applied to real-world problems with increasing success. Such applications often require to put constraints on the agent's behavior. Existing algorithms for constrained RL (CRL) rely on gradient descent-ascent, but this approach comes with a caveat. While these algorithms are guaranteed to converge *on average*, they do not guarantee *last-iterate* convergence, i.e., the current policy of the agent may never converge to the optimal solution. In practice, it is often observed that the policy alternates between satisfying the constraints and maximizing the reward, rarely accomplishing both objectives simultaneously. Here, we address this problem by introducing *Reinforcement Learning with Optimistic Ascent-Descent* (ReLOAD), a principled CRL method with guaranteed last-iterate convergence. We demonstrate its empirical effectiveness on a wide variety of CRL problems including discrete MDPs and continuous control. In the process we establish a benchmark of challenging CRL problems.

1. Introduction

From navigating stratospheric balloons through the atmosphere to coordinating plasma control for nuclear fusion research, reinforcement learning (RL; Sutton & Barto, 2018) has proved increasingly effective at solving real-world problems (Bellemare et al., 2020; Degrave et al., 2022). In RL, an agent interacts with an environment over a series of time steps with the goal of maximizing its expected cumulative reward. When developing intelligent systems which must learn and act in the real world, it's often the case that constraints are placed on agents to ensure that certain safety or efficiency requirements are satisfied. Examples range from training a robot to run while avoiding placing too much torque on its joints, to video compression (Mandhane et al., 2022), to maximizing the efficiency of commercial cooling systems under stability constraints (Luo et al., 2022).

A natural question, then, is how to solve such constrained tasks. In standard RL, we often look to the Reward Hypothesis, which postulates that all goals and purposes of an intelligent agent can be achieved by maximization of a scalar reward (Sutton, 2004). Szepesvári (2020) studied the implications of this hypothesis to CRL. Accordingly, constrained RL problems can be solved by integrating the constraints and task reward into a single, non-stationary reward signal (Altman, 1999). However, this approach carries a subtle but important challenge that can be easily overlooked: gradient-based optimization for such constrained problems only guarantees that the *average* of the agent's behavior over the course of training converges to an optimal solution, with no assurances for the final policy.

Unfortunately, simple solutions like averaging model parameters are ineffective when the policy is implemented as a deep neural network. While this problem has been studied in the context of GANs (Daskalakis et al., 2018; Balduzzi et al., 2018), it has not been addressed within CRL, which brings additional complications. As an illustration, consider the problem of training an agent in the `Walker` domain in DeepMind Control Suite (Tassa et al., 2018) to walk subject to varying upper limits on its height. In Fig. 1a, learning looks stable due to averaging across multiple seeds, but in Fig. 1b, we can see dramatic oscillations over the course of a single training run, with the agent either simply walking normally or lying on the ground.

In this work, we address this issue, introducing *Reinforcement Learning with Optimistic Ascent-Descent* (ReLOAD), a principled CRL method for last-iterate convergence (LIC). Specifically, we make the following contributions: **(1)** We analyze several existing methods for CRL and show that they fail to achieve LIC (Lemma 3.1, Lemma 3.2). We then prove LIC for a generalized form of optimistic mirror descent (Theorem 3.6). **(2)** Building on these theoretical insights, we introduce ReLOAD (Section 4) and demonstrate empirically that it achieves LIC in both the tabular and function approximation settings (Section 5). Furthermore, ReLOAD improves the performance of strong baseline al-

[*]Work done as an intern at DeepMind. [1]Gatsby Unit, UCL [2]DeepMind. Correspondence to: Ted Moskovitz <ted@gatsby.ucl.ac.uk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

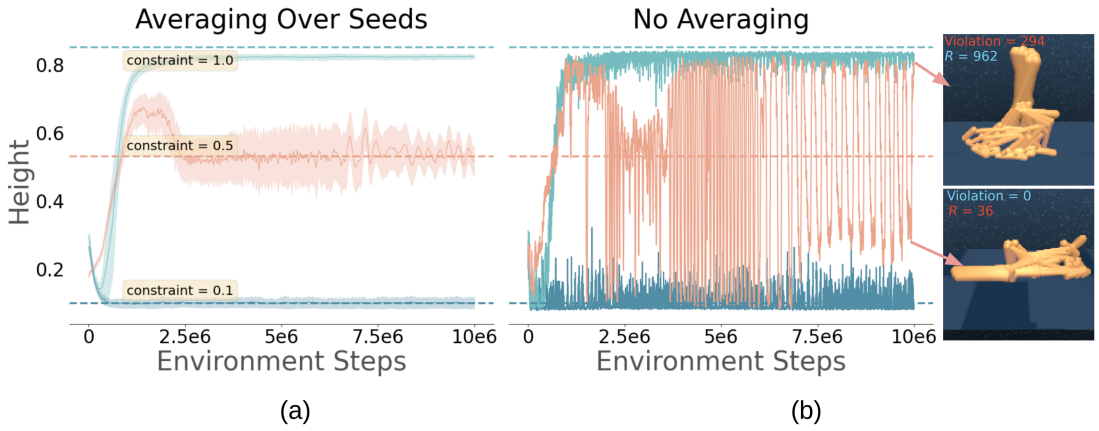

Figure 1. Standard gradient-based methods suffer from oscillations in constrained RL. (a) Training and agent to walk while keeping its height below different thresholds, learning looks stable when averaged across seeds. (b) Examining a single training run, we can see that learning oscillates dramatically, frequently leading to extreme behavior at the end of training (right).

gorithms in challenging CRL problems. **(3)** We identify constraints within a range of control tasks which are especially challenging for traditional methods. We list constraint and task details in the hope that these problems can be reused as a benchmark for CRL.

2. Reinforcement Learning with Constraints

In CRL, an agent not only seeks to maximize its cumulative reward, but must also obey constraints on its behavior. Typically, this problem is modeled as a *Constrained Markov Decision Process* (CMDP, Altman, 1999). An infinite horizon, discounted CMDP is a tuple $\mathcal{M}_C = \big(\mathcal{S}, \mathcal{A}, r_0, \gamma, \rho, \{r_n\}_{n=1}^N, \{\theta_n\}_{n=1}^N\big)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of available actions, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition kernel, $r_0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is a discount factor, $\rho \in \mathcal{P}(\mathcal{S})$ is the distribution over initial states, $\{r_n\}_{n=1}^N$ is the set of constraint rewards, and $\{\theta_n\}_{n=1}^N$ is the set of constraint thresholds. $\mathcal{P}(\cdot)$ denotes the set of distributions over a given space. At each time step, the agent samples an action from a stationary *policy* $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, which causes the environment to transition to a new state. This process induces a cumulative, discounted state-action occupancy measure (henceforth, simply "occupancy measure") associated with the policy:

$$d_\pi(s, a) \triangleq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \pi(a|s) P_\pi(s_t = s). \quad (1)$$

This probability measure lies within the following convex feasible set (a polytope in the discrete case):

$$\mathcal{K} \triangleq \big\{ d_\pi \mid d_\pi \geq 0, \sum_a d_\pi(s, a) = (1 - \gamma)\rho(s) + \gamma \sum_{s', a'} P(s|s', a') d_\pi(s', a') \big\}. \quad (2)$$

The agent's goal is to find a policy that maximizes its expected, cumulative, discounted reward while adhering to the designated constraints. This quantity is referred to as the policy's *value*: $v_0^\pi \triangleq \langle r_0, d_\pi \rangle$. Mathematically, this can be formalized as a constrained optimization problem:

$$\min_\pi -v_0^\pi \quad \text{s.t.} \quad v_n^\pi \leq \theta_n, \quad n = 1, \ldots, N, \quad (3)$$

where $v_n^\pi \triangleq \langle r_n, d_\pi \rangle$. On inspection, we can see that Eq. (3) defines a linear program in d_π:

$$\min_{d_\pi \in \mathcal{K}} -\langle r_0, d_\pi \rangle \ \text{s.t.} \ \langle r_n, d_\pi \rangle \leq \theta_n, \ \ n = 1, \ldots, N \quad (4)$$

Typically, CMDPs are solved via Lagrangian relaxation (Everett, 1963; Altman, 1999), which reframes the objective as a convex-concave min-max game:

$$\min_{d_\pi \in \mathcal{K}} \max_{\mu \geq 0} -\langle r_0, d_\pi \rangle + \sum_{n=1}^{N} \mu_n (\langle r_n, d_\pi \rangle - \theta_n) \triangleq \mathcal{L}(d_\pi, \mu). \quad (5)$$

We provide a more comprehensive review of relevant work on CMDPs in Appendix A.

2.1. The Scalarization Fallacy

Given Eq. (5), it would be tempting to simply solve the inner maximization problem over μ to find the optimal Lagrange multipliers $\mu^\star$. One could then "scalarize" the task and constraint rewards into a single stationary reward function $r^\star = r_0 - \sum_{n=1}^N \mu_n^\star r_n$ and solve the resulting standard MDP. However, the solution to this MDP is not typically the solution to the CMDP—one can easily define CMDPs for which the optimal policy must be stochastic (Altman, 1999; Szepesvári, 2020), but all MDPs admit deterministic optimal policies (Puterman, 2014). One notable exception occurs when one of the constraint thresholds θ_n is extreme (close to the highest or lowest possible v_n). In this case, one reward function r_n will dominate the optimization and the solution will closely match the optimal policy for an MDP with reward r_n, as seen in Fig. 1 for $\theta = 0.1$ and $\theta = 1.0$. Such cases are discussed in detail in Appendix B. For non-

Algorithm 1 Lagrange Optimization for Constrained MDPs

Input: Lagrangian $\mathcal{L} : \mathcal{K} \times \mathbb{R}^N_{\geq 0} \to \mathbb{R}$, $K \in \mathbb{N}_+$
for $k = 1, \ldots, K$ **do**
  $\mu^k = \mathrm{Alg}_\mu(d_\pi^1, \ldots, d_\pi^{k-1}; \mathcal{L})$
  $d_\pi^k = \mathrm{Alg}_\pi(\mu^1, \ldots, \mu^{k-1}; \mathcal{L})$
end for
Return d_π^K, μ^K (last-iterates) or $\bar{d}_\pi^K = \frac{1}{K}\sum_{k=1}^K d_\pi^k$, $\bar{\mu}^K = \frac{1}{K}\sum_{k=1}^K \mu^k$ (average-iterates)

trivial cases, however, we must turn to more specialized approaches for minimax optimization.

2.2. Average- vs. Last-Iterate Convergence

A *saddle-point* (SP), or equilibrium, of a two-player convex-concave zero-sum game like Eq. (5), is a pair $(d_\pi^\star, \mu^\star)$ such that $\mathcal{L}(d_\pi^\star, \mu) \leq \mathcal{L}(d_\pi^\star, \mu^\star) \leq \mathcal{L}(d_\pi, \mu^\star)\, \forall d_\pi, \mu \in \mathcal{K} \times \mathbb{R}^N_{\geq 0}$. A general procedure for finding such a point is presented in Algorithm 1, where at each round the Lagrange multipliers and occupancy measure are updated by procedures Alg_μ and Alg_π, respectively. Because both μ and d_π lie within convex sets and the objective is bilinear, standard results in game theory guarantee that when Alg_μ and Alg_π are no-regret algorithms (e.g., gradient ascent/descent), the averaged iterates $(\frac{1}{K}\sum_{k=1}^K \mu^k, \frac{1}{K}\sum_{k=1}^K d_\pi^k)$ converge to a SP of the Lagrangian (Freund & Schapire, 1997). Significantly, however, in general there is no guarantee regarding the *last* iterates of the optimization—that is, there is no reason to expect that (d_π^K, μ^K) will be close to $(d_\pi^\star, \mu^\star)$. We formalize the distinction between average- and last-iterate convergence with the following definitions:

Definition 2.1 (Average-Iterate Convergence (AIC))**.** Consider a SP problem $\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} \mathcal{L}(x, y)$ where $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$ with a nonempty set of equilibria $\mathcal{Z}^\star \subseteq \mathcal{X} \times \mathcal{Y}$. We say that a sequence $(x^1, y^1), (x^2, y^2), \ldots, (x^K, y^K)$ displays *average-iterate convergence* if $\left(\frac{1}{K}\sum_{k=1}^K x^k, \frac{1}{K}\sum_{k=1}^K y^k\right) \to (x^\star, y^\star) \in \mathcal{Z}^\star$ as $K \to \infty$.

Definition 2.2 (Last-Iterate Convergence (LIC))**.** In the same setting as Definition 2.1, we say that a sequence $(x^1, y^1), (x^2, y^2), \ldots, (x^K, y^K)$ displays *last-iterate convergence* if $\left(x^K, y^K\right) \to (x^\star, y^\star) \in \mathcal{Z}^\star$ as $K \to \infty$.

This discrepancy can be easily overlooked, but it has significant practical implications, as seen in Fig. 1. How can we address this? For inspiration, we look beyond RL for the moment to optimization theory.

3. Optimization in Min-Max Games

The CMDP Langrangian min-max game belongs to a larger family of problems for which standard GD-style approaches (in fact, any algorithm in the broad family of follow-the-regularized-leader approaches) fail to converge in the last-iterate, typically displaying cyclic behavior *around* an equilibrium rather than converging to it (Mertikopoulos et al., 2019). Specifically, GD approaches are instances of primal-dual *mirror descent* (MD), which for a generic SP problem with a differentiable objective $\min_{x\in\mathcal{X}} \max_{y\in\mathcal{Y}} \mathcal{L}(x, y)$ uses MD to update both x and y:

$$x^{k+1} = \operatorname*{argmin}_{x\in\mathcal{X}} \langle \nabla_x \mathcal{L}^k, x\rangle + \frac{1}{\eta^k} D_{\Omega_x}(x; x^k) \quad (6)$$

$$y^{k+1} = \operatorname*{argmax}_{y\in\mathcal{Y}} \langle \nabla_y \mathcal{L}^k, y\rangle - \frac{1}{\eta^k} D_{\Omega_y}(y; y^k), \quad (7)$$

where we use the notation $\nabla\mathcal{L}^k \triangleq \nabla\mathcal{L}(x^k, y^k)$ for brevity, η^k is the step-size at iteration k, and

$$D_\Omega(u;v) = \Omega(u) - \Omega(v) - \langle \nabla\Omega(v), u - v\rangle \quad (8)$$

is the *Bregman divergence* generated by a strictly convex, continuously differentiable function $\Omega(\cdot)$. Common choices for Ω include the squared Euclidean distance $\Omega(u) = \frac{1}{2}\|u\|^2$ which induces $D_\Omega(u;v) = \frac{1}{2}\|u - v\|^2$ and corresponds to standard GD, as well as the negative entropy $\Omega(u) = \langle u, \log u\rangle$, which induces $D_\Omega(u; v) = \mathsf{KL}[u||v]$, corresponding to multiplicative weights updating (MWU; Grigoriadis & Khachiyan, 1995). When $\mathcal{X}$ and $\mathcal{Y}$ are convex and $\mathcal{L}(x, y)$ is convex in x and concave in y, MD guarantees AIC, but may not converge in the last-iterate:

Lemma 3.1 (Insufficiency of MD; (Daskalakis & Panageas, 2018a))**.** *There exist convex-concave SP problems for which primal-dual mirror descent does not achieve LIC.*

3.1. Optimistic Mirror Descent

Fortunately, there exists an approach called *optimistic mirror descent* (OMD) which has been shown to achieve LIC for convex-concave games (Daskalakis & Panageas, 2018a;b; Daskalakis et al., 2018). OMD makes the following simple modification to standard MD (changes in blue):

$$x^{k+1} = \operatorname*{argmin}_{x\in\mathcal{X}} \langle 2\nabla_x \mathcal{L}^k - \nabla_x \mathcal{L}^{k-1}, x^k\rangle + \frac{1}{\eta^k} D_{\Omega_x}(x; x^k).$$

In other words, rather than use the gradient at iteration k, OMD uses the *optimistic* gradient obtained by doubling the current gradient and subtracting the previous one. For concision, we'll denote the optimistic gradient of $\mathcal{L}$ at step k by $\widetilde{\nabla}\mathcal{L}^k = 2\nabla\mathcal{L}^k - \nabla\mathcal{L}^{k-1} = \nabla\mathcal{L}^k + (\nabla\mathcal{L}^k - \nabla\mathcal{L}^{k-1})$. In general, optimistic optimization attempts to use a "hint" about the next gradient to augment the current update. OMD can be seen as setting the hint to be the current gradient $\nabla\mathcal{L}^k$ and then removing the previous hint $\nabla\mathcal{L}^{k-1}$. It belongs to a broader family of so-called "single-call" extra-gradient methods (Hsieh et al., 2019) that only requires one gradient estimate and a single projection into the constraint set at each step, making it particularly scalable to high-dimensional problems. For more discussion of this family of methods, see Appendix E, and for other approaches to LIC in min-max games, see Appendix A. As a demonstration, we applied gradient descent-ascent (GDA) and

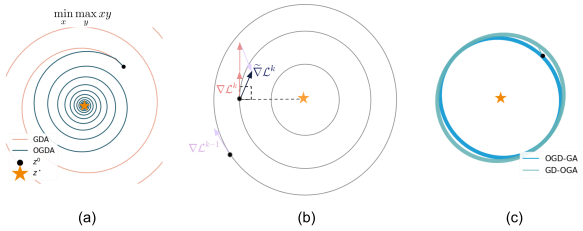

Figure 2. Optimistic gradients achieve LIC. (a) GDA spirals outwards, while OGDA reaches the optimum. (b) The optimistic update $\widetilde{\nabla}\mathcal{L}^k$ bends the trajectory inwards. (c) One optimistic player is not enough for LIC.

optimistic GDA (OGDA) to the simple bilinear problem $\min_{x\in\mathbb{R}} \max_{y\in\mathbb{R}} xy$, for which GDA can be shown to diverge for any positive learning rate (Daskalakis & Panageas, 2018a). (Note that (O)GD is (O)MD with $\Omega(\cdot) = \frac{1}{2}\|\cdot\|^2$.) While both achieve AIC, Fig. 2 shows that that GDA's iterates diverge while OGDA converges to the SP at $(0, 0)$. To get an intuition for why OMD works, we can look to the loss surface in Fig. 2b. By adding the current hint $\nabla\mathcal{L}^k$ and subtracting the previous one $\nabla\mathcal{L}^{k-1}$ to the standard gradient update, the optimistic gradient $\widetilde{\nabla}\mathcal{L}^k$ bends inwards towards the optimum. In contrast, we can see that $\nabla\mathcal{L}^k$ alone is always orthogonal to the radius pointing towards the optimum, and thus never converges to it.

Anticipating the application of OMD to RL, a natural question when building on top of existing approaches is how much can be left unchanged. Since existing CRL approaches use standard gradients when performing primal-dual optimization or focus on stabilizing only the Lagrange multiplier (Calian et al., 2020; Stooke et al., 2020), it would simplify matters if we only required one player in Algorithm 1 to use optimistic gradients—particularly Alg_μ, as this would allow the Alg_π player to simply implement a standard RL algorithm. Unfortunately, this is not the case:

Lemma 3.2 (Being Singly-Optimistic is Not Enough)**.** *There exists a convex-concave objective $\mathcal{L}$ for which no combination of one OMD and one MD player achieves LIC.*

All proofs are provided in Appendix D. This result is reflected in Fig. 2c, where we can see that combining one GD player with one OGD player cycles rather than reaching LIC. We then need to show that OMD's guarantees carry over to the specific setting we require. Typically, convergence results for (O)MD set $\Omega_x = \Omega_y$, but in practice $\mathcal{X}$ and $\mathcal{Y}$ may be such that optimization is more natural with different MD algorithms for x and y. For example, $\mathcal{X}$ may be the probability simplex, for which MWU is a natural approach, while $\mathcal{Y}$ may be the non-negative reals, for which projected GD is more appropriate. Because the OMD-inspired CRL algorithm we derive in Section 4 is more naturally implemented with $\Omega_x \neq \Omega_y$, we'd like to show that MD achieves LIC in this setting. To do so, we turn to monotone operator theory (Bauschke & Combettes, 2011).

3.1.1. Analysis via Monotone Operators

Here, we'll prove the LIC of a generalized form of OMD which permits different Bregman divergences by casting OMD as the application of several *monotone operators*. In the next section, we'll show that the CMDP Lagrangian can be solved by this method. We first provide a few definitions:

Definition 3.3 (Set-Valued Operator)**.** An operator F on a Hilbert space $\mathcal{H}$ is said to be *set-valued* if F maps a point in $\mathcal{H}$ to subset of $\mathcal{H}$. We denote this by $F : \mathcal{H} \rightrightarrows \mathcal{H}$.

Definition 3.4 (Graph)**.** The *graph* of an operator F is

$$\mathrm{Gra}\ F = \{(x, u) \mid u \in F(x)\} \subseteq \mathcal{H} \times \mathcal{H}.$$

Definition 3.5 (Monotone Operator)**.** An operator F on a Hilbert space $\mathcal{H}$ is said to be m*-strongly monotone* if

$$\langle F(x_1) - F(x_2), x_1 - x_2\rangle \geq \frac{m}{2}\|x_1 - x_2\|^2 \quad \forall x_1, x_2 \in \mathcal{H},$$

with $m > 0$. If the inequality holds with $m = 0$, F is simply called monotone. F is *maximal monotone* if there is no other monotone operator G such that $\mathrm{Gra}\ F \subset \mathrm{Gra}\ G$.

Mirror Descent as Fixed Point Iteration Let $\ell(x) \triangleq \mathcal{L}(x, y^k)$. Then Eq. (6) can be written as

$$x^{k+1} = \operatorname*{argmin}_{x\in\mathbb{R}^d} \langle\nabla\ell(x^k), x\rangle + \frac{1}{\eta^k} D_\Omega(x; x^k) + \mathbb{I}_\mathcal{X}(x),$$

where $\mathbb{I}_\mathcal{X}(x)$ is the indicator function which equals 0 if $x \in \mathcal{X}$ and $+\infty$ if $x \notin \mathcal{X}$. Because this problem is convex, we only need a first-order optimality condition, and it is equivalent to solving the following inclusion problem:

$$\begin{aligned} 0 &\in \nabla\ell(x^k) + \frac{1}{\eta^k}(\nabla\Omega(x) - \nabla\Omega(x^k)) + N_\mathcal{X}(x) \\ &\iff \nabla\Omega(x^k) - \eta^k\nabla\ell(x^k) \in (\nabla\Omega + \eta^k N_\mathcal{X})(x) \\ &\iff x \in \mathrm{Prt}^\Omega_{\eta^k N_\mathcal{X}}(\nabla\Omega(x^k) - \eta^k\nabla\ell(x^k)), \end{aligned}$$

where $N_\mathcal{X} = \partial\mathbb{I}_\mathcal{X}$ is the normal cone operator for $\mathcal{X}$ and $\mathrm{Prt}^\Omega_{\eta^k N_\mathcal{X}} = (\nabla\Omega + \eta^k N_\mathcal{X})^{-1}$ is the *proto-resolvent* of $\eta^k N_\mathcal{X}$ relative to Ω (Reich & Sabach, 2011). Thus, mirror descent can be seen as performing fixed point iteration:

$$x^{k+1} = \mathrm{Prt}^\Omega_{\eta^k N_\mathcal{X}}(\nabla\Omega - \eta^k\nabla\ell)(x^k). \tag{9}$$

For optimistic mirror descent, we write the update as

$$\begin{aligned} x^{k+1} = \mathrm{Prt}^\Omega_{\eta^k N_\mathcal{X}}(&\nabla\Omega(x^k) - \eta^k\nabla\ell(x^k) \\ &- \eta^{k-1}(\nabla\ell(x^k) - \nabla\ell(x^{k-1}))). \end{aligned}$$

In general, we can replace $N_\mathcal{X}$ with any maximal monotone operator A and $\nabla\ell$ with any monotone and L-Lipschitz operator B:

$$\begin{aligned} x^{k+1} = \mathrm{Prt}^\Omega_{\eta^k A}(&\nabla\Omega(x^k) - \eta^k B(x^k) \\ &- \eta^{k-1}(B(x^k) - B(x^{k-1}))). \end{aligned} \tag{10}$$

When $\Omega(\cdot) = \frac{1}{2}\|\cdot\|^2$, this is *forward-reflected-backward splitting* (Malitsky & Tam, 2018). We now generalize the convergence result of Malitsky & Tam (2018) to OMD for

split monotone inclusion problems of the form:

$$\text{find } x \in \mathcal{H} \quad \text{s.t.} \quad 0 \in (A+B)(x). \tag{11}$$

Theorem 3.6 (Convergence)**.** *Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximal monotone and let $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz and suppose that $(A+B)^{-1}(0) \neq \varnothing$. Suppose that $(\eta^k) \subseteq [\varepsilon, \frac{1-2\varepsilon}{2L}]$ for some $\varepsilon > 0$. Given $x^0, x^{-1} \in \mathcal{H}$, define the sequence (x^k) according to Eq.* (10) *with Ω σ-strongly convex, $\sigma \geq 1$. Then (x^k) converges weakly to a point contained in $(A+B)^{-1}(0)$.*

Theorem 3.7 (Convergence Rate)**.** *In the setting of Theorem 3.6, if A or B is m-strongly monotone and Ω has an $L_{\nabla\Omega}$-Lipschitz continuous gradient, then (x^k) converges at a rate $\mathcal{O}(1/\alpha^k)$ to the unique element $x^\star \in (A+B)^{-1}(0)$, where $\alpha > 1$ is a constant.*

In the following section, we'll show that this approach can be adapted to the RL setting to achieve LIC for CMDPs.

4. Applying Optimistic Optimization to RL

Next, we will show that the guarantees we developed in the previous section apply to OMD in CMDPs. We begin by writing the CMDP Lagrangian problem as follows:

$$\min_{d_\pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}} \max_{\mu \in \mathbb{R}^N} \mathbb{I}_{\mathcal{K}}(d_\pi) + \mathcal{L}(d_\pi, \mu) + \mathbb{I}_{\mathbb{R}^N_{\geq 0}}(\mu). \tag{12}$$

We can then express the CMDP problem in the form of Eq. (11) by noting that a SP must satisfy:

$$\text{find } \begin{bmatrix} d_\pi \\ \mu \end{bmatrix} \text{ s.t. } \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \begin{bmatrix} \partial \mathbb{I}_{\mathcal{K}}(d_\pi) \\ \partial \mathbb{I}_{\mathbb{R}^N_{\geq 0}}(\mu) \end{bmatrix} + \begin{bmatrix} \nabla_{d_\pi} \mathcal{L}(d_\pi, \mu) \\ -\nabla_\mu \mathcal{L}(d_\pi, \mu) \end{bmatrix} \tag{13}$$

Define $\nabla_{d_\pi} \mathcal{L} = -r_0 + \sum_{n=1}^N \mu_n r_n \triangleq r_\mu$ as the *mixed reward vector* for Lagrange multipliers μ. Substituting this and $\nabla_\mu \mathcal{L}^k = v_{1:N} - \theta$ into the OMD updates above yields:

$$\begin{aligned} {d_\pi}^{k+1} &= \operatorname*{argmin}_{d_\pi \in \mathcal{K}} \langle \tilde{r}_\mu^k, d_\pi \rangle + \frac{1}{\eta} D_{\Omega_\pi}(d_\pi; {d_\pi}^k) \\ \mu^{k+1} &= \operatorname*{argmax}_{\mu \geq 0} \langle \tilde{v}_{1:N}^k - \theta, \mu \rangle - \frac{1}{\eta} D_{\Omega_\mu}(\mu; \mu^k). \end{aligned} \tag{14}$$

where $\tilde{r}_\mu^k \triangleq 2{r_\mu}^k - {r_\mu}^{k-1}$ and $\tilde{v}_{1:N}^k \triangleq 2v_{1:N}^k - v_{1:N}^{k-1}$. Hereafter, we'll refer to this general family of approaches (determined by different choices of Ω_π and Ω_μ) as *Reinforcement Learning with Optimistic Ascent-Descent* (ReLOAD). Based on the above, we can guarantee the LIC of ReLOAD.

Corollary 4.1 (ReLOAD Convergence)**.** *The sequence $((d_\pi^k, \mu^k))$ generated by Eq.* (14) *converges in the last-iterate for $\eta \in (0, 1/2)$.*

At first, glance, one might think that Corollary 4.1 violates the Scalarization Fallacy (Szepesvári, 2020),(Zahavy et al., 2021b, Lemma 1)—as μ^k converges to $\mu^\star$, the policy is optimizing an increasingly stationary reward. However, this is not the case: (d_π^k, μ^k) are jointly converging (last-iterate) to the optimal SP $(d_\pi^\star, \mu^\star)$. This implies that $\pi^\star$ is an optimal policy w.r.t to the $\mu^\star$-weighted reward $r_{\mu^\star}$. However, there might exist other policies that are optimal w.r.t to the $r_{\mu^\star}$ that are not in Nash equilibrium with $\mu^\star$. ReLOAD is guaranteed to converge in last iterate to $\pi^\star$ and not to these other policies. An algorithm that maximizes the stationary reward $r_{\mu^\star}$, on the other hand, will be optimal w.r.t to $r_{\mu^\star}$ but will not necessarily return $\pi^\star$ and therefore will not be in Nash equilibrium with $\mu^\star$. We revisit this observation in our experiments, with Fig. 8 showing that simply optimizing $r_{\mu^\star}$ fails to match ReLOAD's performance.

4.1. Policy-Based ReLOAD

In larger settings, it is easier to optimize policies than occupancy measures. Accordingly, virtually all scalable RL algorithms either learn a policy directly or define one implicitly, e.g., via q-learning, such that the resulting occupancy measure is guaranteed to lie in the feasible set. Rewriting the value in terms of the policy $v_\pi = \langle r, d_\pi \rangle = \langle q_\pi, \pi \rangle$, where $q_\pi(s,a) \triangleq \mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t r_t | s_t = s, a_t = a]$, is not convex in π. Nevertheless, policy optimization methods based on non-convex relaxations of convex objectives often converge to the optimal solution (Shani et al., 2020) and in particular, OMD has been shown to reach first-order—and occasionally global—equilibria in non-convex settings (Cai & Zheng, 2022). We can thus rewrite the Lagrangian as

$$\begin{aligned} \mathcal{L}(d_\pi, \mu) &= \Big\langle -r_0 + \sum_n \mu_n r_n, d_\pi \Big\rangle - \langle \mu, \theta \rangle \\ &= \Big\langle -q_0^\pi + \sum_n \mu_n q_n, \pi \Big\rangle - \langle \mu, \theta \rangle = \mathcal{L}(\pi, \mu), \end{aligned}$$

where $q_n \triangleq q_{\pi, r_n}$. We note that in the non-parametric setting, $\nabla_\pi \mathcal{L} = -q_0 + \sum_{n=1}^N \mu_n q_n \triangleq q_\mu$. In other words, gradient estimation is equivalent to policy evaluation, resulting in the mixed q-value q_μ. As before, the gradient with respect to the Lagrange multipliers is the vector of constraint violations: $\nabla_\mu \mathcal{L} = v_{1:N} - \theta$. When dealing with probability measures, a natural choice for Ω_π is the negative entropy $\Omega_\pi(u) = \langle u, \log u \rangle$, and setting $\Omega_\mu(u) = \frac{1}{2}\|u\|_2^2$, Applying OMD, ReLOAD then performs the following updates:

$$\begin{aligned} \pi^{k+1} &= \operatorname*{argmin}_{\pi \in \Pi} -\langle 2q_\mu^k - q_\mu^{k-1}, \pi \rangle + \frac{1}{\eta} \mathsf{KL}[\pi || \pi^k] \\ &= \frac{\pi^k \exp\left((2q_\mu^k - q_\mu^{k-1})/\eta_\pi^k\right)}{\left\langle \pi^k \exp\left((2q_\mu^k - q_\mu^{k-1})/\eta_\pi^k\right), \mathbf{1} \right\rangle} \\ \mu^{k+1} &= \operatorname*{argmax}_{\mu \geq 0} \langle 2v^k - v^{k-1} - \theta, \mu \rangle - \frac{1}{2\eta} \|\mu - \mu^k\|_2^2 \\ &= \max\{\mu^k + \eta_\mu^k (2v_{1:N}^k - v_{1:N}^{k-1} - \theta),\ 0\}, \end{aligned}$$

with the full algorithm presented in Algorithm 6. Here, accurate policy evaluation is especially important, as estimating q amounts to computing the gradient of the Lagrangian.

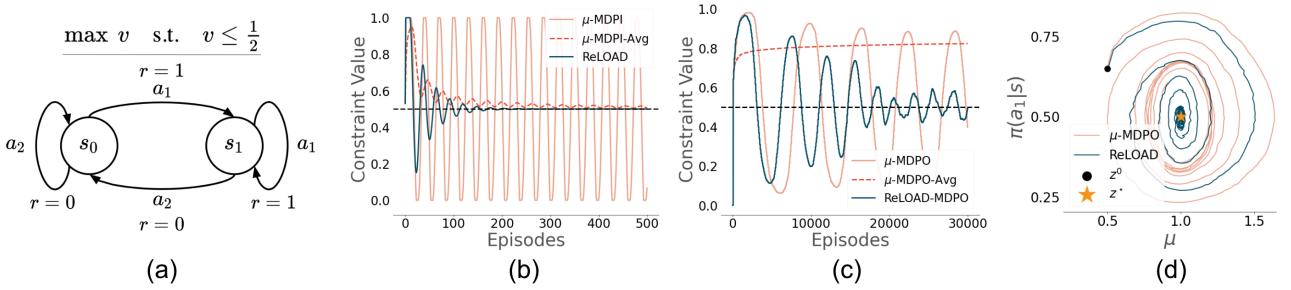

Figure 3. Optimization in a simple CMDP. (a) Schematic of a CMDP whose constraint and reward conflict. (b) Adding ReLOAD to tabular policy iteration damps oscillations. (c) ReLOAD also damps oscillations with function approximation, a setting in which averaging parameters no longer works. (d) Unlike the baseline, ReLOAD successfully converges to the SP with function approximation.

4.2. ReLOAD with Function Approximation

For large state and action spaces, function approximation, i.e., deep RL (DRL), becomes preferable. To implement ReLOAD using DRL, we require the following ingredients: (1) optimistic value estimates and (2) a trust region for the policy update. Fortunately, both requirements are easy to satisfy for most state-of-the-art policy optimization methods. For (1), a crucial factor is that with function approximation, rather than compute q-value estimates for every state-action pair in the environment, policy evaluation is performed over minibatches of sampled experience. This means that the agent can't simply store past gradients to compute optimistic values, as those gradients may have been obtained from the values of different (s, a) pairs from those used to compute the current gradient. Instead, the agent must maintain a copy of the previous value network so that the optimistic q-values can be computed from the same samples: $\tilde{q}^k_\mu(s,a) = 2q^k_\mu(s,a) - q^{k-1}_\mu(s,a)$. Similarly, μ must also be updated using value estimates from the same data. This is a complication which makes using OMD directly, as optimistic GAN methods do (Daskalakis et al., 2018), nontrivial for RL. As for (2), many high-performing policy optimization algorithms employ trust regions as a means of stabilizing learning and improving sample efficiency. Examples include TRPO (Schulman et al., 2015), MDPO (Tomar et al., 2020), and MPO (Abdolmaleki et al., 2018).

5. Experiments

Next, we study the LIC of ReLOAD empirically on a variety of CMDPs with discrete and continuous state and action spaces. We augmented ReLOAD with popular DRL algorithms including: MD Policy Iteration (MDPI; Geist et al., 2019), MD Policy Optimization (MDPO; Tomar et al., 2020), IMPALA (Espeholt et al., 2018), and distributional MPO (DMPO; Abdolmaleki et al., 2020). **Notation:** In the following, when augmenting unconstrained methods to perform Lagrangian optimization, we prefix the method name with "μ-". For more detail on all algorithms, see Appendix F. To measure performance, we use the *weighted reward*: the task reward minus the multiplier-weighted constraint overshoot, given by $r_0 - \sum_{n=1}^{N} \hat{\mu}^\star_n \max\{r_n - \theta_n, 0\}$, where $\hat{\mu}^\star_n$ is the normalized optimal Lagrange multiplier (Stooke et al., 2020). $\hat{\mu}^\star$ was calculated by averaging the Lagrange multipliers learned by the non-optimistic baseline agents (details in Appendix G). Experiments were repeated over 8 random seeds, and error bars denote one standard error. Additional plots and result tables can be found in Appendix H. Videos of trained agents can be found at `https://tedmoskovitz.github.io/ReLOAD/`.

Toy Example: A Paradoxical CMDP In many real-world applications of CMDPs, constraints are introduced to ensure the system's integrity is maintained. For example, a robot may be trained to run as fast as possible but restrained so that it does not place sufficient torque on its joints to break them. To capture these conflicting goals in a simple setting, we tested tabular ReLOAD-MDPI (Algorithm 6) on the two-state CMDP in Fig. 3a. In this task, there is a single constraint reward which is *equal* to the primary reward $r_0 = r_1 = r$, so that $r = 1$ when the agent takes action a_1 placing it in s_1, and $r = 0$ for action a_2 which moves the agent to s_2. The constraint $\theta = 1/2$ was chosen so that the agent may only choose a_1 at most half of the time. Plotting the value over the course of learning in Fig. 3b, we can see that ReLOAD converges, while μ-MDPI oscillates and fails to converge in the last-iterate. However, this approach does achieve AIC ("μ-MDPI-Avg"). To test performance with

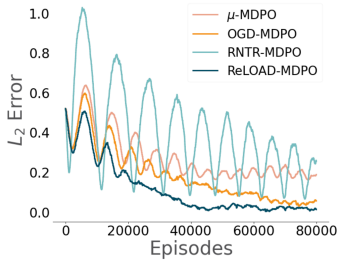

Figure 4. Ablations in the toy CMDP. Both μ-MDPO and RNTR-MDPO fail to converge. However, OGD nearly matches ReLOAD: there are only two states, so using the gradient from the previous minibatch will be close to the current gradient.

function approximation, we then applied ReLOAD-MDPO and μ-MDPO to the same problem. As in the tabular case, ReLOAD significantly dampens oscillations compared to its non-optimistic counterpart (Fig. 3c). (However, some noise remains due to noise in the approximate policy evaluation.) Importantly, μ-MDPO-Avg does not converge in this setting, as it corresponds to averaging the parameters of a nonlinear network. Examining the optimization trajectories in Fig. 3d, we can see that ReLOAD-MDPO converges to the SP, while μ-MDPO gets "stuck," circling but never reaching the optimum. In Fig. 4, we compare ReLOAD-MDPO against various ablations by measuring the L_2 distance from the SP over the course of training. Agents which use OGD directly on the policy parameters are prefixed by "OGD-", and ReLOAD with no trust region is denoted by "RNTR-". Both μ-MDPO and RNTR-MDPO fail to converge. However, performing OGD directly on the policy parameters, rather than via optimistic value estimates, performs nearly as well as ReLOAD. This is because the CMDP only has two states, so the gradient computed from the previous minibatch will be close to the current gradient. This difference becomes significant on large-scale problems.

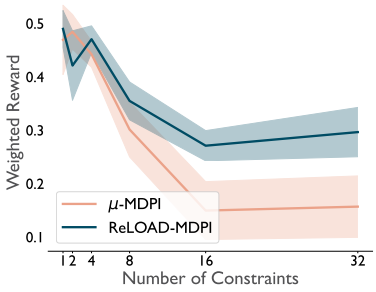

Figure 5. ReLOAD better maintains its performance as more constraints are added.

Multiple Constraints To test ReLOAD's performance with $N > 1$ constraints, we generated random constraint rewards in a small CMDP analogous to the one depicted in Fig. 3 but with three states such that taking action a_i deterministically transitions the agent to state s_i for $i \in \{1, 2, 3\}$. Rewards for each state $r_n(s_i)$ were randomly set between 0 and 1, and the resulting CMDP was tested for feasibility using the CVXPY package (Diamond & Boyd, 2016). We measured the performance of μ-MDPI and ReLOAD-MDPI for $N \in \{1, 2, 4, 8, 16, 32\}$, with each N repeated for 10 seeds. As discussed in Appendix B, it's less likely for random constraints to result in oscillations for lower N, as it's easier for one reward to dominate the others. However, as N increases, it's more probable to have parity among the rewards, making the optimization process more prone to oscillations. Accordingly, the results (Fig. 5) indicate that while performance is comparable for low N, ReLOAD is much better able to maintain performance as the number of constraints increases.

Catch We then applied μ-MDPO and ReLOAD-MDPO to a constrained version of Bsuite's Catch (Osband et al., 2019). In the standard version of this task, the agent moves a paddle left or right to catch a falling ball. To convert this problem into a CMDP, we added a constraint reward r_1 which was 0.2 in the leftmost three columns of the environment and 0 elsewhere, with the constraint $\theta_1 = 1.0$. To both obey the constraint and catch the ball, the agent could only effectively make one trip per episode to the left side of the arena. In Fig. 6a, ReLOAD successfully damps oscillations and achieves LIC, while μ-MDPO only achieves AIC. As an illustration of the consequences, we can see that when the Lagrange multiplier spikes upwards, indicating a constraint violation, the agent successfully catches the ball but lingers on the left side of the environment and thus violates the constraint (Fig. 6b, top). Conversely, a drop in the Lagrange multiplier indicates that the while the constraint is satisfied, performance suffers—the agent simply ignores balls falling on the left side of the arena (Fig. 6b, bottom). ReLOAD learns to stay to the right until the last moment, catching the ball while obeying the constraint (Fig. 6b, middle).

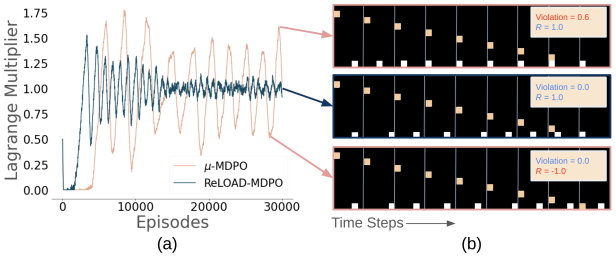

Figure 6. LIC in Catch. (a) ReLOAD reduces oscillations. (b) ReLOAD catches the ball and obeys the constraint, while the standard method only does one or the other.

The Real-World RL Suite (RWRL; Dulac-Arnold et al., 2019) is a collection of DeepMind Control Suite tasks modified with constraints as well as a variety of other real-world challenges which has become a benchmark for applied RL agents (Dulac-Arnold et al., 2020; Huang et al., 2022; Calian et al., 2020; Brunke et al., 2022).

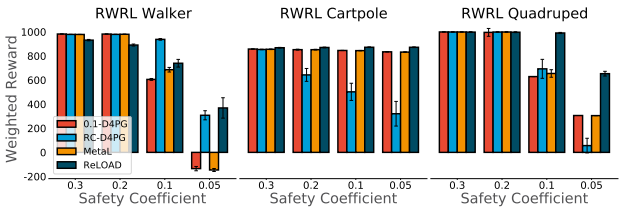

Figure 7. ReLOAD outperforms baselines on the RWRL Suite for the most challenging safety coefficients (lower = harder).

In addition to the choice of constraint threshold, each task in the RWRL Suite has a *safety coefficient*, where low values of this coefficient indicate that it's harder to satisfy the constraints. We trained ReLOAD-DMPO on three

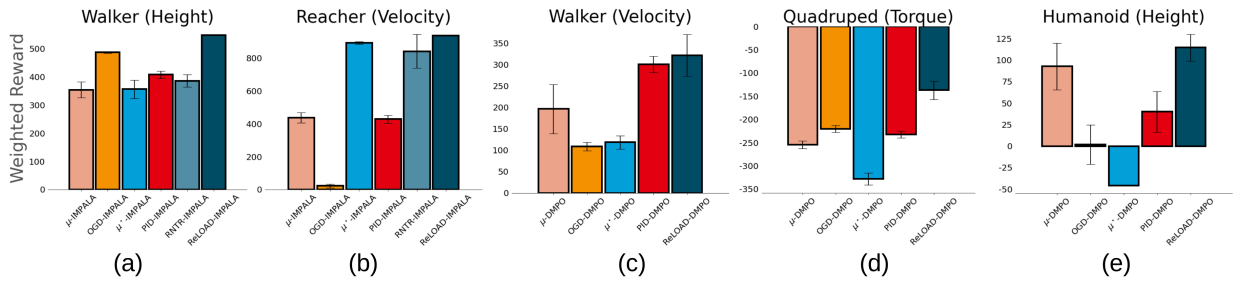

Figure 8. ReLOAD (dark blue) achieves the strongest performance on a variety of CMDPs which induce oscillations in control suite.

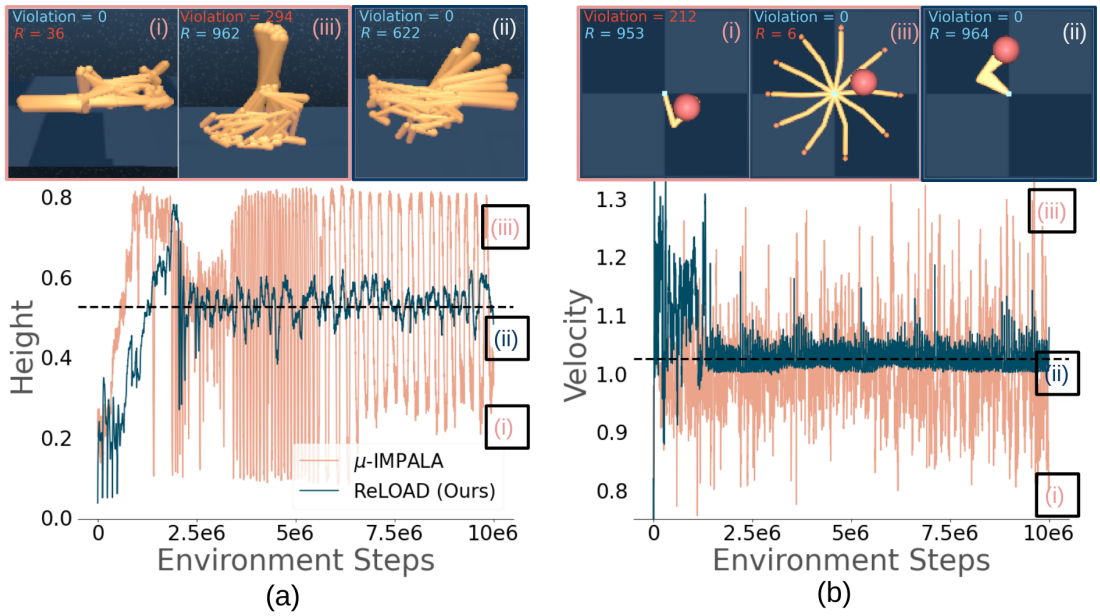

Figure 9. ReLOAD significantly damps oscillations, resulting in agents which perform the desired task while obeying constraints.

challenging tasks: `RWRL-Walker`, `RWRL-Cartpole`, and `RWRL-Quadruped` across the same three constraint thresholds and four safety coefficients for each task used by Calian et al. (2020). As baselines, we applied MetaL, a tuned, fixed-Lagrange method (0.1-D4PG), and a primal-dual D4PG variant similar to μ-D4PG (RC-D4PG), all as in Calian et al. (2020). We also compared ReLOAD against μ-DMPO and multi-objective DMPO (Huang et al., 2022) on the easier safety coefficient settings used by Huang et al. (2022) (see Appendix F and Fig. 14 for further details). As we can see in Fig. 7, ReLOAD solves nearly all the CMDPs at least as well as the baselines and outperforms them for the most challenging safety coefficients. Interestingly, we found that the benchmark constraint thresholds introduced by Calian et al. (2020) were selected to be extreme so as to avoid oscillations. The RWRL suite therefore serves as a useful sanity check that ReLOAD performs strongly even without the threat of oscillations, but we would still like to test it on high-dimensional CMDPs which carry this threat.

Oscillating Control Suite. Finally, we identified tasks and constraint settings in DeepMind Control Suite which cause standard agents to oscillate. We trained ReLOAD on the following tasks: `Walker, Walk` with a constraint on the height of the agent, `Reacher, Easy` with a velocity constraint, `Walker, Walk` with a velocity constraint, `Quadruped, Walk` with a constraint on the torque applied to its joints, and `Humanoid, Walk` with a height constraint. We call this set of tasks and constraints, whose details can be found in Appendix G, the *Oscillating Control Suite*, and we believe it can serve as a challenging benchmark of CMDPs. To test its generality, we paired ReLOAD with both IMPALA and DMPO base agents, and compared it against the corresponding μ- agent, OGD, an agent which uses the fixed optimal Lagrange multiplier obtained by averaging the final iterates of the associated μ- across seeds ($\mu^\star$-), and PID control (PID-; Stooke et al., 2020). For IMPALA, we also tested RNTR. We found that in all cases, ReLOAD achieves higher average weighted reward at the end of training than the baselines (Fig. 8). Im-

portantly, the performance gap between ReLOAD and OGD is much greater than in Fig. 4, as the data distribution varies more significantly from batch to batch in high-dimensional problems. Fig. 9 depicts example training curves and final behaviors for μ-IMPALA and ReLOAD-IMPALA on `Walker` and `Reacher`, with curves for other domains in Appendix Fig. 13. We can see that ReLOAD significantly dampens oscillations. For `Walker` (Fig. 9a), it produces an agent which moves forward with a modified, kneeling walk (panel (ii)), while μ-IMPALA typically either ends up lying down (panel (i))—thus obeying the constraint but not performing the task—or walking normally and ignoring the constraint (panel (iii)). We see a similar pattern with `Reacher` (Fig. 9b), with the ReLOAD agent moving quickly while keeping the tip of its arm in the rewarded area (panel (ii)), while μ-IMPALA either stops moving within the rewarded area (panel (i)) or maximizes velocity while swinging in a circle and ignoring the task (panel (iii)).

6. Conclusion

In this work, we introduced ReLOAD, an RL framework for LIC in constrained MDPs. We examined the challenges in achieving LIC in this setting from a formal perspective, derived a convergence guarantee for a generalized form of OMD for whom a special case is the convex formulation of ReLOAD, and demonstrated ReLOAD's strong empirical performance on a range of challenging CMDPs. One shortcoming of the current analysis is a lack of theoretical understanding of non-convex ReLOAD, and in the future we'd like to experiment with more constraints. We believe that ReLOAD may offer insights into policy optimization with non-stationary rewards more generally, as well as the oscillations which are known to plague standard RL combined with function approximation (Young & Sutton, 2020; Gopalan & Thoppe, 2022), improving optimization within the primal-dual formulation of RL (Bas-Serrano et al., 2021), and extensions to convex MDPs (Zahavy et al., 2021b).

Acknowledgements Work funded by DeepMind. The authors would like to thank Abbas Abdolmaleki, Steven Bohez, Edouard Leurent, Daniel Mankowitz, Dan Calian, Lior Shani, Yash Chandak, Chris Lu, Robert Lange, DJ Strouse, Jack Parker-Holder, Kate Baumli, Kevin Waugh, and other colleagues on the Discovery team and at DeepMind for helpful discussions and feedback over the course of this project.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation, 2018. URL `https://arxiv.org/abs/1806.06920`.

Abdolmaleki, A., Huang, S., Hasenclever, L., Neunert, M., Song, F., Zambelli, M., Martins, M., Heess, N., Hadsell, R., and Riedmiller, M. A distributional view on multi-objective policy optimization. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11–22. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/abdolmaleki20a.html`.

Abernethy, J., Lai, K. A., and Wibisono, A. Last-iterate convergence rates for min-max optimization: Convergence of hamiltonian gradient descent and consensus optimization. In *Algorithmic Learning Theory*, pp. 3–47. PMLR, 2021.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 22–31. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/achiam17a.html`.

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *The Journal of Machine Learning Research*, 22(1):4431–4506, 2021.

Altman, E. Constrained markov decision processes, 1999.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T. The mechanics of n-player differentiable games. In *International Conference on Machine Learning*, pp. 354–363. PMLR, 2018.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. Distributed distributional deterministic policy gradients, 2018. URL `https://arxiv.org/abs/1804.08617`.

Bas-Serrano, J., Curi, S., Krause, A., and Neu, G. Logistic q-learning. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th Interna-*

tional Conference on Artificial Intelligence and Statistics, volume 130 of *Proceedings of Machine Learning Research*, pp. 3610–3618. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/bas-serrano21a.html.

Bauschke, H. H. and Combettes, P. L. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Publishing Company, Incorporated, 1st edition, 2011. ISBN 1441994661.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning, 2017. URL https://arxiv.org/abs/1707.06887.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

Bhatnagar, S. and Lakshmanan, K. An online actor–critic algorithm with function approximation for constrained markov decision processes. *Journal of Optimization Theory and Applications*, 153(3):688–708, 2012.

Bohez, S., Abdolmaleki, A., Neunert, M., Buchli, J., Heess, N., and Hadsell, R. Success at any cost: value constrained model-free continuous control, 2019. URL https://openreview.net/forum?id=rJlJ-2CqtX.

Borkar, V. S. An actor-critic algorithm for constrained markov decision processes. *Systems & control letters*, 54 (3):207–213, 2005.

Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J., and Schoellig, A. P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2023/01/11 2022.

Bura, A., Hasanzadezonuzy, A., Kalathil, D., Shakkottai, S., and Chamberland, J.-F. Dope: Doubly optimistic and pessimistic exploration for safe reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Cai, Y. and Zheng, W. Accelerated single-call methods for constrained min-max optimization. *arXiv preprint arXiv:2210.03096*, 2022.

Calian, D. A., Mankowitz, D. J., Zahavy, T., Xu, Z., Oh, J., Levine, N., and Mann, T. Balancing constraints and rewards with meta-gradient d4pg, 2020. URL https://arxiv.org/abs/2010.06324.

Chambolle, A. and Pock, T. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

Chiang, C.-K., Yang, T., Lee, C.-J., Mahdavi, M., Lu, C.-J., Jin, R., and Zhu, S. Online optimization with gradual variations. In *COLT '12: Proceedings of the 25th Annual Conference on Learning Theory*, 2012.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A lyapunov-based approach to safe reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/4fe5149039b52765bde64beb9f674940-Paper.pdf.

Cui, S. and Shanbhag, U. V. On the analysis of reflected gradient and splitting methods for monotone stochastic variational inequality problems. In *CDC '16: Proceedings of the 57th IEEE Annual Conference on Decision and Control*, 2016.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Daskalakis, C. and Panageas, I. The limit points of (optimistic) gradient descent in min-max optimization, 2018a. URL https://arxiv.org/abs/1807.03907.

Daskalakis, C. and Panageas, I. Last-iterate convergence: Zero-sum games and constrained min-max optimization, 2018b. URL https://arxiv.org/abs/1807.04252.

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. Training GANs with optimism. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJJySbbAZ.

Dayan, P. and Sejnowski, T. J. Exploration bonuses and dual control. *Machine Learning*, 25(1):5–22, 1996.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de las Casas, D., Donner, C., Fritz, L., Galperti, C., Huber, A., Keeling, J., Tsimpoukelli, M., Kay, J., Merle, A., Moret, J.-M., Noury, S., Pesamosca, F., Pfau, D., Sauter, O., Sommariva, C., Coda, S., Duval, B., Fasoli, A., Kohli, P., Kavukcuoglu, K., Hassabis, D., and Riedmiller, M. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Diamond, S. and Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning, 2019. URL https://arxiv.org/abs/1904.12901.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. An empirical investigation of the challenges of real-world reinforcement learning, 2020. URL https://arxiv.org/abs/2003.11881.

Efroni, Y., Mannor, S., and Pirotta, M. Exploration-exploitation in constrained mdps, 2020. URL https://arxiv.org/abs/2003.02189.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL http://arxiv.org/abs/1802.01561.

Everett, H. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Oper. Res.*, 11(3):399–417, jun 1963. URL https://doi.org/10.1287/opre.11.3.399.

Flennerhag, S., Schroecker, Y., Zahavy, T., van Hasselt, H., Silver, D., and Singh, S. Bootstrapped meta-learning. *arXiv preprint arXiv:2109.04504*, 2021.

Flennerhag, S., Zahavy, T., O'Donoghue, B., van Hasselt, H., György, A., and Singh, S. Optimistic meta-gradients. *arXiv preprint arXiv:2301.03236*, 2023.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. URL https://www.sciencedirect.com/science/article/pii/S002200009791504X.

Geist, M., Scherrer, B., and Pietquin, O. A theory of regularized Markov decision processes. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2160–2169. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/geist19a.html.

Gidel, G., Berard, H., Vignoud, G., Vincent, P., and Lacoste-Julien, S. A variational inequality perspective on generative adversarial networks. In *ICLR '19: Proceedings of the 2019 International Conference on Learning Representations*, 2019.

Gopalan, A. and Thoppe, G. Approximate q-learning and sarsa (0) under the epsilon-greedy policy: a differential inclusion analysis. *arXiv preprint arXiv:2205.13617*, 2022.

Grigoriadis, M. D. and Khachiyan, L. G. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58, 1995. URL https://www.sciencedirect.com/science/article/pii/0167637795000320.

Hsieh, Y.-G., Iutzeler, F., Malick, J., and Mertikopoulos, P. On the convergence of single-call stochastic extra-gradient methods, 2019. URL https://arxiv.org/abs/1908.08465.

Huang, S., Abdolmaleki, A., Vezzani, G., Brakel, P., Mankowitz, D. J., Neunert, M., Bohez, S., Tassa, Y., Heess, N., Riedmiller, M., and Hadsell, R. A constrained multi-objective reinforcement learning framework. In Faust, A., Hsu, D., and Neumann, G. (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 883–893. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/huang22a.html.

Korpelevich, G. M. The extragradient method for finding saddle points and other problems. In *Ekonomika i Matematicheskie Metody*, volume 12, pp. 747–756, 1976.

Kumar, S., Kumar, A., Levine, S., and Finn, C. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *Advances in Neural Information Processing Systems*, 33:8198–8210, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning, 2015. URL https://arxiv.org/abs/1509.02971.

Liu, T., Zhou, R., Kalathil, D., Kumar, P., and Tian, C. Learning policies with zero or bounded constraint violation for constrained mdps. In *Thirty-fifth Conference on Neural Information Processing Systems*, 2021.

Luo, J., Paduraru, C., Voicu, O., Chervonyi, Y., Munns, S., Li, J., Qian, C., Dutta, P., Davis, J. Q., Wu, N., et al. Controlling commercial cooling systems using reinforcement learning. *arXiv preprint arXiv:2211.07357*, 2022.

Malitsky, Y. and Tam, M. K. A forward-backward splitting method for monotone inclusions without cocoercivity, 2018. URL https://arxiv.org/abs/1808.04162.

Mandhane, A., Zhernov, A., Rauh, M., Gu, C., Wang, M., Xue, F., Shang, W., Pang, D., Claus, R., Chiang, C.-H., Chen, C., Han, J., Chen, A., Mankowitz, D. J., Broshear, J., Schrittwieser, J., Hubert, T., Vinyals, O., and Mann, T. Muzero with self-competition for rate control in vp9 video compression, 2022. URL https://arxiv.org/abs/2202.06626.

Mertikopoulos, P., Lecouat, B., Zenati, H., Foo, C.-S., Chandrasekhar, V., and Piliouras, G. Optimistic mirror descent in saddle-point problems: Going the extra(-gradient) mile. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg8jjC9KQ.

Moskovitz, T., Arbel, M., Huszar, F., and Gretton, A. Efficient wasserstein natural gradients for reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OHgnfSrn2jv.

Moskovitz, T., Arbel, M., Parker-Holder, J., and Pacchiano, A. Towards an understanding of default policies in multitask policy optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 10661–10686. PMLR, 2022.

Nemirovski, A. S. Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

O'Donoghue, B. Variational Bayesian reinforcement learning with regret bounds. *Advances in Neural Information Processing Systems*, 34:28208–28221, 2021.

O'Donoghue, B. and Lattimore, T. Variational Bayesian optimistic sampling. *Advances in Neural Information Processing Systems*, 34:12507–12519, 2021.

O'Donoghue, B., Lattimore, T., and Osband, I. Stochastic matrix games with bandit feedback. *arXiv preprint arXiv:2006.05145*, 2020.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., Van Roy, B., Sutton, R., Silver, D., and Van Hasselt, H. Behaviour suite for reinforcement learning, 2019. URL https://arxiv.org/abs/1908.03568.

Pacchiano, A., Parker-Holder, J., Tang, Y., Choromanski, K., Choromanska, A., and Jordan, M. Learning to score behaviors for guided policy optimization. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7445–7454. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/pacchiano20a.html.

Paternain, S., Chamon, L., Calvo-Fullana, M., and Ribeiro, A. Constrained reinforcement learning has zero duality gap. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/c1aeb6517a1c7f33514f7ff69047e74e-Paper.pdf.

Perolat, J., Munos, R., Lespiau, J.-B., Omidshafiei, S., Rowland, M., Ortega, P., Burch, N., Anthony, T., Balduzzi, D., De Vylder, B., et al. From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In *International Conference on Machine Learning*, pp. 8525–8535. PMLR, 2021.

Popov, L. D. A modification of the arrow-hurwicz method for search of saddle points. *Mathematical Notes of the Academy of Sciences of the USSR*, 28(5):845–848, 1980.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Reich, S. and Sabach, S. Existence and approximation of fixed points of bregman firmly nonexpansive mappings in reflexive banach spaces. In *Springer Optimization and Its Applications*, chapter Chapter 15, pp. 301–316. Springer, 2011. URL https://EconPapers.repec.org/RePEc:spr:spochp:978-1-4419-9569-8_15.

Ryu, E. K. and Yin, W. *Large-Scale Convex Optimization: Algorithms & Analyses via Monotone Operators*. Cambridge University Press, 2022. doi: 10.1017/9781009160865.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In Bach, F. R. and Blei, D. M. (eds.), *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/schulman15.html.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shani, L., Efroni, Y., and Mannor, S. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5668–5675, 2023/01/16 2020.

Shani, L., Zahavy, T., and Mannor, S. Online apprenticeship learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8240–8248, 2023/01/13 2022.

Simão, T. D., Jansen, N., and Spaan, M. T. Alwayssafe: Reinforcement learning without safety constraint violations during training. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2021.

Sokota, S., D'Orazio, R., Kolter, J. Z., Loizou, N., Lanctot, M., Mitliagkas, I., Brown, N., and Kroer, C. A unified approach to reinforcement learning, quantal response equilibria, and two-player zero-sum games. *arXiv preprint arXiv:2206.05825*, 2022.

Stooke, A., Achiam, J., and Abbeel, P. Responsive safety in reinforcement learning by pid lagrangian methods, 2020. URL https://arxiv.org/abs/2007.03964.

Strehl, A. L. and Littman, M. L. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Sutton, R. The reward hypothesis, 2004. URL http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Szepesvári, C. Constrained mdps and the reward hypothesis, Mar 2020. URL http://readingsml.blogspot.com/2020/03/constrained-mdps-and-reward-hypothesis.html.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. Deepmind control suite, 2018. URL https://arxiv.org/abs/1801.00690.

Tessler, C., Mankowitz, D. J., and Mannor, S. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkfrvsA9FX.

Thomas, P. S., da Silva, B. C., Barto, A. G., and Brunskill, E. On ensuring that intelligent machines are well-behaved, 2017. URL https://arxiv.org/abs/1708.05448.

Tomar, M., Shani, L., Efroni, Y., and Ghavamzadeh, M. Mirror descent policy optimization, 2020. URL https://arxiv.org/abs/2005.09814.

Xu, Z., van Hasselt, H. P., and Silver, D. Meta-gradient reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/2715518c875999308842e3455eda2fe3-Paper.pdf.

Young, K. and Sutton, R. S. Understanding the pathologies of approximate policy evaluation when combined with greedification in reinforcement learning, 2020. URL https://arxiv.org/abs/2010.15268.

Zahavy, T., Cohen, A., Kaplan, H., and Mansour, Y. Apprenticeship learning via frank-wolfe. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6720–6728, 2020a.

Zahavy, T., Xu, Z., Veeriah, V., Hessel, M., Oh, J., van Hasselt, H. P., Silver, D., and Singh, S. A self-tuning actor-critic algorithm. *Advances in neural information processing systems*, 33:20913–20924, 2020b.

Zahavy, T., O'Donoghue, B., Barreto, A., Mnih, V., Flennerhag, S., and Singh, S. Discovering diverse nearly optimal policies withsuccessor features. *arXiv preprint arXiv:2106.00669*, 2021a.

Zahavy, T., O'Donoghue, B., Desjardins, G., and Singh, S. Reward is enough for convex MDPs. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021b. URL https://openreview.net/forum?id=ELndVeVA-TR.

Zahavy, T., Schroecker, Y., Behbahani, F., Baumli, K., Flennerhag, S., Hou, S., and Singh, S. Discovering policies with domino: Diversity optimization maintaining near optimality, 2022. URL https://arxiv.org/abs/2205.13521.

A. Additional Related Work

Beyond that which is covered in the main text, there is a rich history of work on CMDPs. Borkar (2005) was the first to analyze an actor-critic approach to CMDPs, while Bhatnagar & Lakshmanan (2012) was the first to expand this approach to function approximation. Tessler et al. (2019); Achiam et al. (2017); Efroni et al. (2020); Bohez et al. (2019); Chow et al. (2018); Paternain et al. (2019) all focus on integrating constraints into sequential decision problems (Altman, 1999), commonly done, as noted, via Lagrangian relaxation (Tessler et al., 2019). Calian et al. (2020) argue for a *soft-constraint* approach to CRL, wherein the solution is not absolutely required to satisfy a particular constraint, but rather is penalized in proportion to its violation (Thomas et al., 2017; Dulac-Arnold et al., 2020). This philosophy lies in contrast to *hard-constraint* approaches, for which a solution is marked as invalid if there is any constraint violation (Dalal et al., 2018; Bura et al., 2022). In zero-violation approaches, methods are initialized in the feasible region, and only updated in ways that are guaranteed not to leave the feasible region (Liu et al., 2021; Simão et al., 2021). In applications, soft constraints may be preferable when violations do not result in catastrophic system failure, but rather simply undesirable behavior (e.g., inefficiency). Calian et al. (2020) attempt to stabilize the learning process in CMDPs by using meta-gradients (Xu et al., 2018; Zahavy et al., 2020b) to adapt the learning rate of the Lagrange multiplier online. More recently, Bootstrapped meta gradients (Flennerhag et al., 2021) have been analyzed and shown to provide a form of optimism (Flennerhag et al., 2023); thus, it would be interesting to see if they can help to achieve LIC in CMDPs. Stooke et al. (2020) apply a principled approach to damping the dynamics of the Lagrange multiplier via PID control with the goal of reducing constraint overshoots over the course of training. However, neither of these two approaches guarantees LIC—this may have a connection to Lemma 3.2, which shows that only one optimistic player (e.g., the Lagrange player) is in general not sufficient to guarantee LIC. Efroni et al. (2020) address the role of exploration in CMDPs, proving bounds for both the linear programming formulation and the primal-dual formulation of the problem, though they only consider standard gradients and do not focus on LIC vs. AIC.

Rather than formulate CRL as a CMDP, Huang et al. (2022) and Abdolmaleki et al. (2020) consider a multi-objective approach. That is, rather than integrate the constraints and the task reward into a single, non-stationary reward, they optimize the task reward and each constraint reward independently, and then search over a pareto front which balances among them.

There are a number of recent applications of CRL which have achieved impressive practical successes. One example is the use of MuZero (Schrittwieser et al., 2020) for video compression by Mandhane et al. (2022). Specifically, the agent was trained on a constrained MDP to maximize video quality with a constraint on the allowed bit rate. Others involve using CRL for quality-diversity optimization where the agent is trying to find a set of diverse skill while all of the skills are required to satisfy a near-optimality constraint on the reward (Zahavy et al., 2021a; 2022; Kumar et al., 2020).

Beyond extra-gradient methods (see Appendix C below), there are other approaches which aim to achieve LIC in min-max games. The symplectic gradient adjustment (Balduzzi et al., 2018) uses a signed additive term to the gradient to push the dynamics away from unstable equilibria and towards stable ones. Hamiltonian gradient descent updates the iterates in the direction of the (negative) gradient of the squared norm of the signed partial derivatives, and has been shown to achieve LIC in a variety of min-max games (Abernethy et al., 2021). However, when applied to CMDPs, minimizing the squared gradient with respect to the Lagrange multiplier(s) is equivalent to an apprenticeship learning problem (Abbeel & Ng, 2004; Zahavy et al., 2020a; Shani et al., 2022), which is itself a convex MDP representing a challenging optimization problem (Zahavy et al., 2021b). Perolat et al. (2021) instead augment the objective with an adaptive regularizer, solving the resulting convex/concave (but biased) problem exactly before iteratively refitting with progressively lesser regularization.

One aspect we have not touched on is the question of exploration. When the agent has uncertainty about the rewards or the constraints, how can it 'explore' sufficiently well in order to solve the CMDP? This would require information seeking behaviour to discover the rewards and constraints in the environment. A common heuristic for exploration is 'optimism in the face of uncertainty' (Dayan & Sejnowski, 1996; Strehl & Littman, 2008; O'Donoghue, 2021), however there is little work applying optimism to CMDPs. Some preliminary work in constrained bandits (O'Donoghue & Lattimore, 2021) or more generally in two-player zero sum matrix games (O'Donoghue et al., 2020) is encouraging, but the presented algorithms are not online and therefore questions like LIC are not applicable. Similarly, classic adversarial algorithms like EXP3 (Auer et al., 2002) typically only guarantee AIC. Clearly more work is required in this area.

As noted in the main text, many policy optimization methods in deep RL currently use a form of trust region to stabilize optimization. KL-based trust regions are discussed in detail by Agarwal et al. (2021); Geist et al. (2019); Moskovitz et al. (2022); Shani et al. (2020). While not explored further here, it would also be interesting to study policy optimization methods which use trust regions generated by other divergence measures or distances, such as the Wasserstein distance (Moskovitz et al., 2021; Pacchiano et al., 2020).

B. Extreme Constraints and Oscillations

As noted in the main text, so-called "extreme" constraints don't often induce oscillations in practice. Here, we present a simple, relatively informal argument to provide an intuition for why this is the case. Consider a simple CMDP with one constraint and where values lie in the range $[0, B]$, where $B < \infty$. (Note this bounding can be easily obtained for any reward function which is upper- and lower-bounded by adding the lower bound to rewards to make them non-negative.) The Lagrangian in this case is

$$\mathcal{L}(d_\pi, \mu) = -\langle r_0, d_\pi \rangle + \mu(\langle r_1, d_\pi \rangle - \theta). \tag{15}$$

An *extreme* constraint threshold θ, then, is one which is close to 0 or B, as those are the bounds on the value. Say that $\theta = 0$. Then the Lagrangian reduces to

$$\mathcal{L}(d_\pi, \mu) = -\langle r_0, d_\pi \rangle + \mu\langle r_1, d_\pi \rangle, \tag{16}$$

with the gradients being

$$\begin{aligned} \nabla_{d_\pi} \mathcal{L} &= \mu r_1 - r_0 \\ \nabla_\mu \mathcal{L} &= \langle r_1, d_\pi \rangle = v_1. \end{aligned}$$

Because $v_1 \geq 0$, the Lagrange multiplier μ will always be increasing, which, because of $\nabla_{d_\pi} \mathcal{L}$, means that the occupancy measure will increasingly be updated to align with the constraint reward r_1 and ignore the task reward r_0. Therefore, the constraint reward will dominate the optimization of d_π, and there is no "back and forth" between optimizing the constraint reward and the task reward. Similarly, if $\theta = B$, $v_1 - B \leq 0$, so μ will be non-increasing, eventually dropping to 0, so that d_π will only optimize the task reward r_1.

This also provides motivation for why problems with "intermedaite" constraints which induce oscillations are the most valuable/interesting CMDPs—because extreme constraints cause one reward function to dominate, the problem essentially reduces to an MDP, and can reasonably be solved with standard RL methods.

C. Extra-Gradient Methods

The minimax problems studied in the paper all fall within the broader class of *variational inequality* (VI) problems, which can be written as

$$\text{find } x^\star \in \mathcal{X} \subseteq \mathbb{R}^d \quad \text{s.t.} \quad \langle F(x^\star), x - x^\star \rangle \geq 0 \; \forall x \in \mathcal{X}$$

for some single-valued operator $F : \mathbb{R}^d \to \mathbb{R}^d$. For example, if $F = \nabla \mathcal{L}$ for some differentiable loss $\mathcal{L}$, the solution to the VI problem is a critical point of $\mathcal{L}$. The *extra-gradient* algorithm (EG; Korpelevich, 1976) is as follows:

$$\begin{aligned} x^{k+1/2} &= \Pi_{\mathcal{X}}(x^k - \eta^k F(x^k)) \\ x^{k+1} &= \Pi_{\mathcal{X}}(x^k - \eta^k F(x^{k+1/2})), \end{aligned}$$

where $\Pi_{\mathcal{X}}(z) \triangleq \min_{x \in \mathcal{X}} \|x - z\|$ is the projection operator onto $\mathcal{X}$. The Bregman variant of EG is called the Mirror-Prox method (MP; Nemirovski, 2004). While EG achieves the optimal $\mathcal{O}(1/k)$ convergence rate when V is monotone and Lipschitz-continuous, it is difficult to scale, as each update requires two gradient computations and two projections into $\mathcal{X}$. In addition to the optimistic gradient method, there are a number of so-called "single-call" EG methods (one of which is the optimistic gradient):

- *Past EG* (PEG; Chiang et al., 2012; Gidel et al., 2019; Popov, 1980):

$$\begin{aligned} x^{k+1/2} &= \Pi_{\mathcal{X}}(x^k - \eta^k F(x^{k-1/2})) \\ x^{k+1} &= \Pi_{\mathcal{X}}(x^k - \eta^k F(x^{k+1/2})) \end{aligned} \tag{17}$$

- *Reflected Gradient* (RG; Chambolle & Pock, 2011; Cui & Shanbhag, 2016):

$$\begin{aligned} x^{k+1/2} &= x^k - (x^{k-1} - x^k) \\ x^{k+1} &= \Pi_{\mathcal{X}}(x^k - \eta^k F(x^{k+1/2})). \end{aligned}$$

Both PEG and RG are very closely related to OGD, and are identical in the unconstrained case. As an additional note, the term "optimistic gradient" is frequently applied broadly within the literature, and is often used to refer to the EG method or any of its single-call variants. For a more comprehensive discussion and analysis of these methods, we refer the reader to Hsieh et al. (2019).

Algorithm 2 PEG-MDPI

Require: CMDP $\mathcal{M}_C$, step sizes $\{\eta_\pi, \eta_\mu\} > 0$
Initialize π^1, μ^1, π^0, μ^0
for $k = 1, \dots, K$ **do**
  Half-step:

$$\pi^{k+1/2} = \frac{\pi^k \exp\left(q^{\mu^{k-1/2}}_{\pi^{k-1/2}}/\eta_\pi\right)}{\left\langle \pi^k \exp\left(q^{\mu^{k-1/2}}_{\pi^{k-1/2}}/\eta_\pi\right), 1\right\rangle}$$
$$\mu^{k+1/2} = \max\{\mu^k - \eta_\mu(v^{k-1/2}_{1:N} - \theta), 0\}$$

  $\{q^{k+1/2}_n\}_{n=0}^N \leftarrow \texttt{PolicyEval}(\mathcal{M}_C, \pi^{k+1/2})$
  $q^{k+1/2}_{\mu^{k+1/2}} \leftarrow -q^{k+1/2}_0 + \sum_{n=1}^N \mu^{k+1/2}_n q^{k+1/2}_n$ (mixed q-values)
  $v^{k+1/2}_{1:N} \leftarrow [\langle q^{k+1/2}_1, \pi^{k+1/2}\rangle, \dots, \langle q^{k+1/2}_N, \pi^{k+1/2}\rangle]^\top$
  Full-step:

$$\pi^{k+1} = \frac{\pi^k \exp\left(q^{\mu^{k+1/2}}_{\pi^{k+1/2}}/\eta_\pi\right)}{\left\langle \pi^k \exp\left(q^{\mu^{k+1/2}}_{\pi^{k+1/2}}/\eta_\pi\right), 1\right\rangle}$$
$$\mu^{k+1} = \max\{\mu^k - \eta_\mu(v^{k+1/2}_{1:N} - \theta), 0\}$$

end for
return π^K, μ^K

C.1. PEG Policy Iteration

As described in Section 4, policy evaluation in the tabular case is equivalent to gradient computation, and a trust region step equates to a projection step in OMD. While OMD only requires one of these each per update, PEG (Eq. (17)) requires two projections and one gradient call per update, which makes it less scalable than OMD. Nonetheless, we can derive a Bregman PEG-based optimization method for optimizing CMDPs which carries similar guarantees to ReLOAD.

Let $\ell(x) \triangleq \mathcal{L}(x, y^k)$. Then we can write the PEG update as

$$x^{k+1/2} = \operatorname*{argmin}_{x\in\mathcal{X}} \langle \nabla\ell(x^{k-1/2}), x\rangle + \frac{1}{\eta^k} D_\Omega(x; x^k)$$
$$x^{k+1} = \operatorname*{argmin}_{x\in\mathcal{X}} \langle \nabla\ell(x^{k+1/2}), x\rangle + \frac{1}{\eta^k} D_\Omega(x; x^k).$$

Setting Ω_π as the negative entropy and Ω_μ as the squared Euclidean norm, we can derive the policy iteration-style algorithm in Algorithm 2. We verified its LIC on the paradoxical CMDP from Fig. 3, with results plotted in Appendix H.

D. Theoretical Results

D.1. Impossibility Results

Lemma 3.1 (Insufficiency of MD; (Daskalakis & Panageas, 2018a))**.** *There exist convex-concave SP problems for which primal-dual mirror descent does not achieve LIC.*

Proof. This proof is originally due to Daskalakis et al. (2018), but we repeat it here for completeness.

Consider the problem

$$\min_{x\in\mathbb{R}} \max_{y\in\mathbb{R}} xy.$$

The gradient descent-ascent updates at step k are:

$$x_{k+1} = x_k - \eta y_k$$
$$y_{k+1} = y_k + \eta x_k.$$

For simplicity of notation we consider a fixed learning rate η, but the same result is obtained for variable learning rates. This problem has a unique SP at $(x^\star, y^\star) = (0, 0)$. Consider the squared distance from the origin at step k, $\Delta_k \triangleq x_k^2 + y_k^2$. We then have

$$\begin{aligned}\Delta_{k+1} &= x_{k+1}^2 + y_{k+1}^2 \\ &= (x_k - \eta y_k)^2 + (y_k + \eta x_k)^2 \\ &= x_k^2 - 2\eta x_k y_k + \eta^2 y_k^2 + y_k^2 + 2\eta x_k y_k + \eta^2 x_k^2 \\ &= \Delta_k + \eta^2 \Delta_k \\ &= (1 + \eta^2)\Delta_k.\end{aligned}$$

Therefore, for any $\eta > 0$, the distance from the SP grows with every step of gradient ascent-descent. □

Lemma 3.2 (Being Singly-Optimistic is Not Enough)**.** *There exists a convex-concave objective $\mathcal{L}$ for which no combination of one OMD and one MD player achieves LIC.*

Proof. Consider $\min_{x_\mathbb{R}} \max_{y \in \mathbb{R}} \; xy$. The OGD-GA (singly-optimistic) dynamics are given by

$$x_{k+1} = x_k - 2\eta y_k + \eta y_{k-1}$$
$$y_{k+1} = y_k + \eta x_k.$$

Stability analysis: To facilitate analysis, we can rewrite this (discrete-time) dynamical system by introducing an extra variable z which tracks y:

$$\begin{aligned}x_{k+1} &= x_k - 2\eta y_k + \eta z_k \\ y_{k+1} &= y_k + \eta x_k \\ z_{k+1} &= y_k.\end{aligned}$$

The Jacobian is given by

$$J = \begin{pmatrix} 1 & -2\eta & \eta \\ \eta & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Its eigenvalues λ are obtained by solving

$$\det(J - \lambda I) = -\lambda^3 + 2\lambda^2 - (2\eta^2 + 1)\lambda + \eta^2 = 0. \tag{18}$$

Two of the three eigenvalues have modulus greater than one for all $\eta > 0$, so $\rho(J) > 1$ and the system is unstable. □

D.2. Analysis of Mixed-Bregman OMD via Monotone Operators

CMDPs as Monotone Inclusion Problems The theory of monotone operators provides a general, powerful framework for analyzing the behavior of convex optimization problems (Ryu & Yin, 2022). While monotone operator theory's generality often facilitates relatively simple proofs of convergence, it can also make it more challenging to show *rates* of convergence. Nonetheless, as the main goal of our theoretical analysis is to simply demonstrate the fundamental soundness of our proposed approach, monotone operator theory is a useful choice. Specifically, we cast convex-concave SP problems as *monotone inclusion problems*. Monotone inclusion problems take the form

$$\text{find } x \in \mathcal{H} \quad \text{s.t.} \quad 0 \in F(x)$$

where $F : \mathcal{H} \to \mathcal{H}$ is a *monotone* operator and $\mathcal{H}$ is a Hilbert space. A Hilbert space is a vector space upon which an inner product $\langle \cdot, \cdot \rangle$ is defined and for which the distance induced by this inner product is a complete metric space (one which contains every Cauchy sequence within it). One example of a vector space is the d-dimensional reals $\mathbb{R}^d$, for which the inner product is the standard dot product and the induced metric is the Euclidean distance. A monotone operator is an operator for which the following inequality holds:

$$\langle F(x) - F(y), x - y \rangle \geq 0$$

for all $x, y \in \mathcal{H}$. Often inclusion problems can be made simpler to solve if the operator F can be *split* into two (or more) operators $F = A + B$, as A and B may be more tractable to use and evaluate separately. We therefore consider inclusion problems of the form

$$\text{find } x \in \mathcal{H} \quad \text{s.t.} \quad 0 \in (A + B)(x), \tag{19}$$

where $A : \mathcal{H} \rightrightarrows \mathcal{H}$ and $B : \mathcal{H} \to \mathcal{H}$ are monotone operators. The notation $\rightrightarrows$ indicates A is (in the general case) a set-valued operator, one which maps a point in $\mathcal{H}$ to a (possibly empty) subset of $\mathcal{H}$. A wide variety of problems can be expressed in this form, and in particular we consider the Lagrangian of the CMDP problem expressed as follows:

$$\min_{d_\pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}} \max_{\mu \in \mathbb{R}^N} \mathbb{I}_{\mathcal{K}}(d_\pi) + \mathcal{L}(d_\pi, \mu) + \mathbb{I}_{\mathbb{R}^N_{\geq 0}}(\mu), \tag{20}$$

where $\mathbb{I}_{\mathcal{X}}(\cdot)$ is the indicator function which equals 0 inside the set $\mathcal{X} \subseteq \mathcal{H}$ and $+\infty$ outside of $\mathcal{X}$. We can therefore express the CMDP problem in the form of Eq. (11) by noting that a SP must satisfy the first-order optimality condition:

$$\text{find } \begin{bmatrix} d_\pi \\ \mu \end{bmatrix} \quad \text{s.t.} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \begin{bmatrix} \partial \mathbb{I}_{\mathcal{K}}(d_\pi) \\ \partial \mathbb{I}_{\mathbb{R}^N_{\geq 0}}(\mu) \end{bmatrix} + \begin{bmatrix} \nabla_{d_\pi} \mathcal{L}(d_\pi, \mu) \\ -\nabla_\mu \mathcal{L}(d_\pi, \mu) \end{bmatrix} \tag{21}$$

where we can note that $\partial \mathbb{I}_{\mathcal{X}} = N_{\mathcal{X}}$, where $N_{\mathcal{X}}$ is the normal cone operator for $\mathcal{X}$ (Ryu & Yin, 2022).

(Optimistic) Mirror Descent as Fixed Point Iteration The mirror descent update for the Bregman divergence $D_\Omega(\cdot;\cdot)$ generated by the strictly convex, continuously differentiable function $\Omega(\cdot)$ and applied to a differentiable loss function $\ell(\cdot)$ is given by

$$x^{k+1} = \operatorname*{argmin}_{x \in \mathcal{X}} \langle \nabla \ell(x^k), x \rangle + \frac{1}{\eta^k} D_\Omega(x; x^k) \tag{22}$$

$$= \operatorname*{argmin}_{x \in \mathbb{R}^d} \langle \nabla \ell(x^k), x \rangle + \frac{1}{\eta^k} D_\Omega(x; x^k) + \mathbb{I}_{\mathcal{X}}(x). \tag{23}$$

This is equivalent to solving the following inclusion problem:

$$0 \in \nabla \ell(x^k) + \frac{1}{\eta^k}(\nabla \Omega(x) - \nabla \Omega(x^k)) + N_{\mathcal{X}}(x) \tag{24}$$

$$\iff \nabla \Omega(x^k) - \eta^k \nabla \ell(x^k) \in (\nabla \Omega + \eta^k N_{\mathcal{X}})(x) \tag{25}$$

$$\iff x \in (\nabla \Omega + \eta^k N_{\mathcal{X}})^{-1}(\nabla \Omega(x^k) - \eta^k \nabla \ell(x^k)) = \operatorname{Prt}^{\Omega}_{\eta^k N_{\mathcal{X}}}(\nabla \Omega(x^k) - \eta^k \nabla \ell(x^k)), \tag{26}$$

where $\operatorname{Prt}^{\Omega}_{\eta^k N_{\mathcal{X}}} = (\nabla \Omega + \eta^k N_{\mathcal{X}})^{-1}$ is the *proto-resolvent* of $\eta^k N_{\mathcal{X}}$ relative to Ω (Reich & Sabach, 2011). Thus, mirror descent can be seen as performing fixed point iteration (FPI) as follows:

$$x^{k+1} = \operatorname{Prt}^{\Omega}_{\eta^k N_{\mathcal{X}}}(\nabla \Omega - \eta^k \nabla \ell)(x^k). \tag{27}$$

For optimistic mirror descent, we write the update as

$$x^{k+1} = \operatorname{Prt}^{\Omega}_{\eta^k N_{\mathcal{X}}}(\nabla \Omega(x^k) - \eta^k \nabla \ell(x^k) - \eta^{k-1}(\nabla \ell(x^k) - \nabla \ell(x^{k-1}))). \tag{28}$$

More generally, for SP problems like Eq. (13), we can think of $x^k = [d_\pi^k, \mu^k]^\mathsf{T} \in \mathcal{K} \times \mathbb{R}_{\geq 0} = \mathcal{X}$, where

$$A = \begin{bmatrix} N_{\mathcal{K}} \\ N_{\mathbb{R}^N_{\geq 0}} \end{bmatrix} \qquad B = \begin{bmatrix} \nabla_{d_\pi} \mathcal{L}(d_\pi, \mu) \\ -\nabla_\mu \mathcal{L}(d_\pi, \mu) \end{bmatrix} \qquad \nabla \Omega = \begin{bmatrix} \nabla \Omega_\pi \\ \nabla \Omega_\mu \end{bmatrix}. \tag{29}$$

As a note, in the following analysis we'll frequently consider the Bregman divergence generated by Ω:

$$D_\Omega(x_1; x_2) \triangleq \Omega(x_1) - \Omega(x_2) - \langle \nabla \Omega(x_2), x_1 - x_2 \rangle. \tag{30}$$

In the "stacked"/SP case, we can consider this to be the *stacked* divergence where $\Omega = [\Omega_\pi, \Omega_\mu]^\top$ and $\nabla \Omega$ is as above. We can then write the OMD update as

$$x^{k+1} = \operatorname{Prt}^{\Omega}_{\eta^k A}(\nabla \Omega(x^k) - \eta^k B(x^k) - \eta^{k-1}(B(x^k) - B(x^{k-1}))). \tag{31}$$

When $\Omega_\pi(\cdot) = \Omega_\mu(\cdot) = \frac{1}{2}\|\cdot\|^2$, this is equivalent to *forward-reflected-backward splitting* (Malitsky & Tam, 2018).

D.3. Convergence Analysis

Before showing convergence, we require several prepatory lemmas. In the following, we assume that $A : \mathcal{H} \rightrightarrows H$ is maximal monotone and $B : \mathcal{H} \to \mathcal{H}$ is monotone and L-Lipschitz.

Lemma D.1. *Let $F : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximal monotone, and let $d_1, u_1, u_0, v_2, v_1 \in \mathcal{H}$ be arbitrary. Define d_2 as*

$$d_2 = \mathrm{Prt}_F^{\Omega}(\nabla\Omega(d_1) - u_1 - (v_1 - u_0)).$$

Then, $\forall x \in \mathcal{H}$ and $u \in -F(x)$, we have

$$\begin{aligned} D_\Omega(x; d_2) + \langle v_2 - u_1, x - d_2\rangle \le D_\Omega(x; d_1) + \langle v_1 - u_0, x - d_1\rangle \\ + \langle v_1 - u_0, d_1 - d_2\rangle - D_\Omega(d_2; d_1) - \langle v_2 - u, d_2 - x\rangle. \end{aligned}$$

Proof. From the definition of the proto-resolvent, we have

$$\nabla\Omega(d_2) + F(d_2) \ni \nabla\Omega(d_1) - u_1 - (v_1 - u_0).$$

Then by the monotonicity of F,

$$\begin{aligned} 0 &\le \langle -u - \nabla\Omega(d_1) + u_1 + (v_1 - u_0) + \nabla\Omega(d_2), x - d_2\rangle \\ &= -\langle \nabla\Omega(d_1) - \nabla\Omega(d_2), x - d_2\rangle + \langle u + v_1 - u_0, x - d_2\rangle + \langle u, d_2 - x\rangle \\ &\overset{(1)}{=} -D_\Omega(x; d_2) - D_\Omega(d_2; d_1) + D_\Omega(x; d_1) + \langle u + v_1 - u_0, x - d_2\rangle + \langle u, d_2 - x\rangle, \end{aligned}$$

where (1) is due to the Bregman 3-point identity. Then we can simply add and subtract both $\langle v_2, x - d_2\rangle$ and $\langle d_1, v_1 - u_0\rangle$ and rearrange to get the desired result. □

Lemma D.2. *Let $x \in (A + B)^{-1}(0)$ and let (x^k) be given by Eq. (31). Suppose that $(\eta^k) \subseteq [\varepsilon, \frac{1-2\varepsilon}{2L}]$ for some $\varepsilon > 0$. Then for all $k \in \mathbb{N}$ we have*

$$\begin{aligned} D_\Omega(x; x^{k+1}) + \eta^k\langle B(x^{k+1}) - B(x^k), x - x^{k+1}\rangle + D_\Omega(x^{k+1}; x^k) + \left(\frac{\varepsilon}{2} - \frac{1}{4}\right)\|x^{k+1} - x^k\|^2 \\ \le D_\Omega(x; x^k) + \eta^{k-1}\langle B(x^k) - B(x^{k-1}), x - x^k\rangle + \frac{1}{4}\|x^k - x^{k-1}\|^2. \end{aligned}$$

Proof. Apply Lemma D.1 with

$$\begin{aligned} F \triangleq \eta^k A \quad d_1 \triangleq x^k \quad u_0 \triangleq \eta^{k-1}B(x^{k-1}) \quad v^1 \triangleq \eta^{k-1}B(x^k) \\ u \triangleq \eta^k B(x) \quad d_2 \triangleq x^{k+1} \quad u_1 \triangleq \eta^k B(x^k) \quad v_2 \triangleq \eta^k B(x^{k+1}) \end{aligned}$$

to get

$$\begin{aligned} D_\Omega(x; x^{k+1}) &+ \eta^k\langle B(x^{k+1}) - B(x^k), x - x^{k+1}\rangle + D_\Omega(x^{k+1}; x^k) \\ &\le D_\Omega(x; x^k) + \eta^{k-1}\langle B(x^k) - B(x^{k-1}), x - x^k\rangle \\ &\quad + \eta^{k-1}\underbrace{\langle B(x^k) - B(x^{k-1}), x^k - x^{k+1}\rangle}_{(1)} - \eta^k\underbrace{\langle B(x^{k+1}) - B(x), x^{k+1} - x\rangle}_{(2)}. \end{aligned}$$

Since B is monotone, (2) is non-negative and can be dropped. Since B is also L-Lipschitz, (1) can be written as

$$\begin{aligned} \langle B(x^k) - B(x^{k-1}), x^k - x^{k+1}\rangle &\le L\|x^k - x^{k-1}\|\|x^k - x^{k+1}\| \\ &\le \frac{L}{2}(\|x^k - x^{k-1}\|^2 + \|x^k - x^{k+1}\|^2). \end{aligned}$$

Applying these changes and rearranging gives

$$\begin{aligned} D_\Omega(x; x^{k+1}) + \eta^k\langle B(x^{k+1}) - B(x^k), x - x^{k+1}\rangle + D_\Omega(x^{k+1}; x^k) - \frac{1}{2}\eta^{k-1}L\|x^{k+1} - x^k\|^2 \\ \le D_\Omega(x; x^k) + \eta^{k-1}\langle B(x^k) - B(x^{k-1}), x - x^k\rangle + \frac{1}{2}\eta^{k-1}L\|x^k - x^{k-1}\|^2. \end{aligned}$$

Then since $\eta^{k-1}L \le 1/2$ and

$$-\frac{1}{2}\eta^{k-1}L \ge -\frac{1 - 2\varepsilon}{4} = -\frac{1}{4} + \frac{\varepsilon}{2},$$

we get the desired bound. □

Now we are able to prove convergence.

Theorem 3.6 (Convergence)**.** *Let $A : \mathcal{H} \rightrightarrows \mathcal{H}$ be maximal monotone and let $B : \mathcal{H} \to \mathcal{H}$ be monotone and L-Lipschitz and suppose that $(A + B)^{-1}(0) \neq \varnothing$. Suppose that $(\eta^k) \subseteq [\varepsilon, \frac{1-2\varepsilon}{2L}]$ for some $\varepsilon > 0$. Given $x^0, x^{-1} \in \mathcal{H}$, define the*

sequence (x^k) *according to Eq.* (10) *with* Ω σ*-strongly convex,* $\sigma \geq 1$*. Then* (x^k) *converges weakly to a point contained in* $(A+B)^{-1}(0)$*.*

Proof. Let $x \in (A+B)^{-1}(0)$. We can telescope Lemma D.2 to get

$$\begin{aligned} D_\Omega(x; x^{k+1}) + \eta^k \langle B(x^{k+1}) - B(x^k), x - x^{k+1} \rangle + D_\Omega(x^{k+1}; x^k) - \frac{1}{4}\|x^{k+1} - x^k\|^2 + \frac{\varepsilon}{2}\sum_{i=0}^{k} \|x^{i+1} - x^i\|^2 \\ \leq D_\Omega(x; x^0) + \eta^{-1}\langle B(x^0) - B(x^{-1}), x - x^0 \rangle + \frac{1}{4}\|x^0 - x^{-1}\|^2. \end{aligned} \tag{32}$$

Because B is L-Lipschitz, we have

$$\begin{aligned} \eta^k \langle B(x^{k+1}) - B(x^k), x - x^{k+1} \rangle &\geq -\eta^k L \|x^{k+1} - x^k\| \|x - x^{k+1}\| \\ &\geq -\frac{1}{2}\eta^k L(\|x^{k+1} - x^k\|^2 + \|x - x^{k+1}\|). \end{aligned}$$

Then, because $\frac{1}{2}\eta^k L \leq \frac{1-2\varepsilon}{4} < \frac{1}{4}$ and since our choice of Ω implies $D_\Omega(w; z) \geq \frac{1}{2}\|w - z\|^2$, we can write

$$D_\Omega(x^{k+1}; x^k) - \frac{1}{4}\|x^{k+1} - x^k\|^2 \geq \frac{1}{2}\|x^{k+1} - x^k\|^2 - \frac{1}{4}\|x^{k+1} - x^k\|^2 = \frac{1}{4}\|x^{k+1} - x^k\|^2.$$

Combining this with Eq. (32) gives

$$D_\Omega(x; x^{k+1}) + \frac{\varepsilon}{2}\sum_{i=0}^{k} \|x^{i+1} - x^i\|^2 \leq D_\Omega(x; x^0) + \eta^{-1}\langle B(x^0) - B(x^{-1}), x - x^0 \rangle + \frac{1}{4}\|x^0 - x^{-1}\|^2.$$

Therefore, we can see that (x^k) is bounded above and moreover that $\|x^{k+1} - x^k\| \to 0$. Let $\bar{x}$ be a sequential weak cluster point of (x^k). We know that

$$x^{k+1} = (\nabla\Omega + \eta^k A)^{-1}(\nabla\Omega(x^k) - \eta^k B(x^k)) - \eta^{k-1}(B(x^k) - B(x^{k-1}))$$

so that

$$\nabla\Omega(x^{k+1}) - \nabla\Omega(x^k) + \eta^k B(x^k) + \eta^{k-1}(B(x^k) - B(x^{k-1})) \in -\eta^k A(x^{k+1}).$$

By subtracting $\eta^k B(x^{k+1})$ from both sides and rearranging, we get

$$\frac{1}{\eta^k}\left(\nabla\Omega(x^k) - \nabla\Omega(x^{k+1}) + \eta^k(B(x^{k+1}) - B(x^k)) + \eta^{k-1}(B(x^{k-1}) - B(x^k))\right) \in (A+B)(x^{k+1}). \tag{33}$$

Since $A + B$ is maximal monotone, its graph is demiclosed. Take the limit along a subsequence of (x^k) which converges to $\bar{x}$ in Eq. (33) and, noting that $\eta^k \geq \varepsilon\ \forall k \in \mathbb{N}$, we can see that the differences on the left side of Eq. (33) vanish in the limit so that $0 \in (A+B)(\bar{x})$.

To show that (x^k) is weakly convergent to $\bar{x}$, note that by combining Lemma D.2 and the Lipschitzness of B, we can see that the limit

$$\lim_{k\to\infty} D_\Omega(\bar{x}; x^k) + \eta^{k-1}\langle B(x^k) - B(x^{k-1}), \bar{x} - x^k \rangle + \frac{1}{4}\|x^k - x^{k-1}\|^2$$

exists. Since (x^k) and (η^k) are bounded, $\|x^k - x^{k-1}\| \to 0$, and since B is continuous, the limit is equal to $\lim_{k\to\infty} D_\Omega(\bar{x}; x^k)$. Since the cluster point $\bar{x}$ was chosen arbitrarily, (x^k) is weakly convergent by Malitsky & Tam (2018) Lemma 2.2, and the proof is complete. □

This result then immediately implies the following corollary when the learning rate η is fixed.

Corollary D.3. *Let* $A : \mathcal{H} \rightrightarrows \mathcal{H}$ *be maximal monotone and let* $B : \mathcal{H} \to \mathcal{H}$ *be monotone and* L*-Lipschitz and suppose that* $(A+B)^{-1}(0) \neq \varnothing$*. Suppose that* $\eta \in (0, \frac{1}{2L})$*. Given* $x^0, x^{-1} \in \mathcal{H}$*, define the sequence* (x^k) *according to*

$$x^{k+1} = \mathrm{Prt}^{\Omega}_{\eta A}(\nabla\Omega(x^k) - 2\eta B(x^k) + \eta B(x^{k-1})) \tag{34}$$

Then (x^k) *converges weakly to a point contained in* $(A+B)^{-1}(0)$*.*

Theorem 3.7 (Convergence Rate)**.** *In the setting of Theorem 3.6, if* A *or* B *is* m*-strongly monotone and* Ω *has an* $L_{\nabla\Omega}$*-Lipschitz continuous gradient, then* (x^k) *converges at a rate* $\mathcal{O}(1/\alpha^k)$ *to the unique element* $x^\star \in (A+B)^{-1}(0)$*, where* $\alpha > 1$ *is a constant.*

Proof. Without loss of generality, let A be m-strongly monotone. Note that we do not lose generality here because if B

were m-strongly monotone instead, we could write $A + B = (A + mI) + (B - mI)$, thereby making $A' = A + mI$ strongly monotone while preserving the monotonicity of $B' = B - mI$. Let $x \in (A + B)^{-1}(0)$. Then with a constant learning rate $\eta \in (0, 1/2L)$ as in Corollary D.3 we can use the strong monotonicity of A in place of standard monotonicity in Lemma D.1 and propagate the resulting inequality through Lemma D.2 to get

$$\eta m\|x - x^{k+1}\|^2 + D_\Omega(x; x^{k+1}) + \eta\langle B(x^{k+1} - B(x^k), x - x^{k+1}\rangle + D_\Omega(x^{k+1}; x^k) - \frac{1}{2}L\eta\|x^{k+1} - x^k\|^2$$
$$\leq D_\Omega(x; x^k) + \eta\langle B(x^k) - B(x^{k-1}), x - x^k\rangle + \frac{1}{4}\|x^k - x^{k-1}\|^2.$$

Then because Ω is σ-strongly convex and has $L_{\nabla\Omega}$-Lipschitz smooth gradient, by (Bauschke & Combettes, 2011) Theorem 18.5, we have

$$\eta m\|x - x^{k+1}\|^2 + \frac{\sigma}{2}\|x - x^{k+1}\|^2 + \eta\langle B(x^{k+1} - B(x^k), x - x^{k+1}\rangle + \frac{\sigma}{2}\|x^{k+1} - x^k\|^2 - \frac{1}{2}L\eta\|x^{k+1} - x^k\|^2$$
$$\leq \frac{L_{\nabla\Omega}}{2}\|x - x^k\|^2 + \eta\langle B(x^k) - B(x^{k-1}), x - x^k\rangle + \frac{1}{4}\|x^k - x^{k-1}\|^2$$

Simplifying gives

$$(\sigma + 2\eta m)\|x - x^{k+1}\|^2 + 2\eta\langle B(x^{k+1} - B(x^k), x - x^{k+1}\rangle + (\sigma - L\eta)\|x^{k+1} - x^k\|^2$$
$$\leq L_{\nabla\Omega}\|x - x^k\|^2 + 2\eta\langle B(x^k) - B(x^{k-1}), x - x^k\rangle + \frac{1}{2}\|x^k - x^{k-1}\|^2.$$

Define sequences (a^k) and (b^k) such that

$$a^k \triangleq \frac{L_{\nabla\Omega}}{2}\|x - x^k\|^2$$
$$b^k \triangleq \frac{L_{\nabla\Omega}}{2}\|x - x^k\|^2 + 2\eta\langle B(x^k) - B(x^{k-1}), x - x^k\rangle + \frac{1}{2}\|x^k - x^{k-1}\|^2.$$

We'd like to rewrite the LHS of the inequality in terms of a^{k+1} and b^{k+1}. Observe that

$$(\sigma + 2\eta m)\|x - x^{k+1}\|^2 = \frac{2}{L_{\nabla\Omega}}(\sigma + 2\eta m)\frac{L_{\nabla\Omega}}{2}\|x - x^{k+1}\|^2 = \frac{2}{L_{\nabla\Omega}}(\sigma + 2\eta m)a^{k+1}$$
$$= \left(\frac{2\sigma}{L_{\nabla\Omega}} + \frac{4\eta m}{L_{\nabla\Omega}}\right)a^{k+1} = \frac{L_{\nabla\Omega}}{2}\|x - x^{k+1}\|^2 + \left(\frac{2\sigma}{L_{\nabla\Omega}} + \frac{4\eta m}{L_{\nabla\Omega}} - 1\right)a^{k+1},$$

and that

$$(\sigma - L\eta)\|x^{k+1} - x^k\|^2 = \frac{1}{2}\|x^{k+1} - x^k\|^2 + \left(\sigma - L\eta - \frac{1}{2}\right)\|x^{k+1} - x^k\|^2.$$

Then we have

$$\left(\frac{2\sigma}{L_{\nabla\Omega}} + \frac{4\eta m}{L_{\nabla\Omega}} - 1\right)a^{k+1} + b^{k+1} + \left(\sigma - L\eta - \frac{1}{2}\right)\|x^{k+1} - x^k\|^2 \leq a^k + b^k.$$

Set $\varepsilon \triangleq \min\left\{\frac{4\sigma - 2L_{\nabla\Omega} + 8\eta m}{L_{\nabla\Omega} + 2}, \sigma - L\eta - \frac{1}{2}\right\}$. Then we can write

$$\left(\frac{2\sigma}{L_{\nabla\Omega}} + \frac{4\eta m}{L_{\nabla\Omega}} - 1\right)a^{k+1} + b^{k+1} + \varepsilon\|x^{k+1} - x^k\|^2 \leq a^k + b^k.$$

Consider the LHS. Add and subtract $\varepsilon b^{k+1}/2$ to get

$$\left(\frac{2\sigma}{L_{\nabla\Omega}} - 1 + \frac{4\eta m}{L_{\nabla\Omega}} - \frac{\varepsilon}{2}\right)a^{k+1} + \left(1 + \frac{\varepsilon}{2}\right)b^{k+1} + \frac{3\varepsilon}{4}\|x^{k+1} - x^k\|^2 - \varepsilon\eta\langle B(x^{k+1}) - B(x^k), x - x^{k+1}\rangle$$
$$\overset{(i)}{\geq} \left(\frac{2\sigma}{L_{\nabla\Omega}} - 1 + \frac{4\eta m}{L_{\nabla\Omega}} - \frac{\varepsilon}{2}\right)a^{k+1} + \left(1 + \frac{\varepsilon}{2}\right)b^{k+1} + \frac{3\varepsilon}{4}\|x^{k+1} - x^k\|^2 - \varepsilon\eta L\|x^{k+1} - x^k\|\|x - x^{k+1}\|$$
$$\overset{(ii)}{\geq} \left(\frac{2\sigma}{L_{\nabla\Omega}} - 1 + \frac{4\eta m}{L_{\nabla\Omega}} - \frac{\varepsilon}{2}\right)a^{k+1} + \left(1 + \frac{\varepsilon}{2}\right)b^{k+1} + \frac{3\varepsilon}{4}\|x^{k+1} - x^k\|^2 - \frac{\varepsilon\eta L}{2}\left(\|x^{k+1} - x^k\|^2 + \|x - x^{k+1}\|^2\right)$$
$$\geq \left(\frac{2}{L_{\nabla\Omega}} - 1 + \frac{4\eta m}{L_{\nabla\Omega}} - \frac{\varepsilon}{2} - \frac{\varepsilon}{2L_{\nabla\Omega}}\right)a^{k+1} + \left(1 + \frac{\varepsilon}{2}\right)b^{k+1} + \frac{\varepsilon}{2}\|x^{k+1} - x^k\|^2,$$

where (i) is because B is L-Lipschitz and (ii) is due to Young's inequality. Now, set $\alpha \triangleq$

$\min\left\{\frac{2}{L_{\nabla\Omega}} - 1 + \frac{4\eta m}{L_{\nabla\Omega}} - \frac{\varepsilon}{2} - \frac{\varepsilon}{2L_{\nabla\Omega}}, 1 + \frac{\varepsilon}{2}\right\} > 1$ due to $\varepsilon \leq \frac{4\sigma - 2L_{\nabla\Omega} + 8\eta m}{L_{\nabla\Omega}+2}$. We then get

$$\alpha\left(a^{k+1} + b^{k+1}\right) + \frac{\varepsilon}{2}\|x^{k+1} - x^k\|^2 \leq a^k + b^k$$
$$\Rightarrow \alpha(a^{k+1} + b^{k+1}) \leq a^k + b^k,$$

which, iterating, yields

$$a^{k+1} + b^{k+1} \leq \frac{1}{\alpha}(a^k + b^k) \leq \frac{1}{\alpha}\left(\frac{1}{\alpha}(a^{k-1} + b^{k-1})\right) \leq \cdots \leq \frac{1}{\alpha^k}(a^0 + b^0).$$

Therefore, $x^k \to x$ with rate $\mathcal{O}(1/\alpha^k)$, and since x was arbitrarily chosen, it is unique. □

Corollary 4.1 (ReLOAD Convergence)**.** *The sequence* $((d_\pi^k, \mu^k))$ *generated by Eq.* (14) *converges in the last-iterate for* $\eta \in (0, 1/2)$.

Proof. To connect this problem to Theorem 3.6, we can can take advantage of the fact that monotonicity is preserved by concatenation (Ryu & Yin, 2022) and set $x^k = [d_\pi^k, \mu^k]^\mathsf{T} \in \mathcal{K} \times \mathbb{R}_{\geq 0} = \mathcal{X}$ such that

$$A = \begin{bmatrix} N_\mathcal{K} \\ N_{\mathbb{R}^N_{\geq 0}} \end{bmatrix} \quad B = \begin{bmatrix} \nabla_{d_\pi}\mathcal{L} \\ -\nabla_\mu \mathcal{L} \end{bmatrix} \quad \nabla\Omega = \begin{bmatrix} \nabla\Omega_\pi \\ \nabla\Omega_\mu \end{bmatrix}. \tag{35}$$

We can further confirm that the normal cone operator is maximal monotone (Ryu & Yin, 2022) and that $\nabla\mathcal{L}$ is monotone and 1-Lipschitz, as the problem is bilinear. We then get the desired result by applying Corollary D.3. □

E. Overview of Extragradient Approaches

F. Further Algorithm Details

Below, we provide a more detailed overview of the algorithms used in the paper, both ReLOAD variants and baseline approaches.

Mirror Descent Policy Iteration (MDPI) MDPI (Geist et al., 2019) is a modified policy iteration algorithm which, rather than apply a hard/greedy improvement step to the policy, uses a soft-improvement step:

$$\pi^{k+1} = \frac{\pi^k \exp(q^k/\eta_\pi)}{\langle \pi^k \exp(q^k/\eta_\pi), 1\rangle}$$

To adapt MDPI for Lagrangian optimization, we update the policy using the mixed q-value $q^k \to q_\mu^k$ as described in the main text and update the Lagrange multiplier(s) with projected gradient ascent. This is μ-MDPI, and is similar to the procedure used in Shani et al. (2022). To add ReLOAD, we simply replace the mixed q-values with optimistic mixed q-values. This procedure is summarized in Algorithm 6.

Mirror Descent Policy Optimization (MDPO) MDPO (Tomar et al., 2020) is a DRL approach designed to apply a scalable approximation of MD to the policy update, and is closely related to both TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017). There are both on- and off-policy versions of MDPO. Here, we focus on the on-policy version, for which at step the algorithm approximately performs the following update to the policy parameters θ:

$$\theta^{k+1} \leftarrow \operatorname*{argmax}_{\theta\in\Theta} \Psi(\theta, \theta^k) \quad \text{where}$$
$$\Psi(\theta, \theta^k) = \mathbb{E}_{s\sim d_{\pi_{\theta^k}}}\left[\mathbb{E}_{a\sim\pi_{\theta^k}} A^{\pi_{\theta^k}}(s,a) - \frac{1}{\eta^k}\mathsf{KL}[\pi_\theta(\cdot \mid s)||\pi_{\theta^k}(\cdot \mid s)]\right], \tag{36}$$

where $A^\pi(s,a) \triangleq q^\pi(s,a) - v^\pi(s)$ is the advantage function for π. Rather than solve this optimization exactly, for each update to the policy, MDPO performs an inner loop consisting of m SGD steps on Ψ using (s,a) pairs obtained from rollouts using the current policy π_{θ^k}. To adapt MDPO for Lagrangian optimization (μ-MDPO), we computed the advantages for the task and constraint rewards $A_n^\pi(s,a) = q_n^\pi(s,a) - v_n^\pi(s)$ and then mixed them using the Lagrange multiplier(s) μ^n to create mixed advantages:

$$A_{\mu^k}^{\theta^k}(s,a) = A_0^{\theta^k}(s,a) - \sum_{n=1}^N \mu_n A_n^{\theta^k}(s,a).$$

Note that this requires a value function estimate for each constraint reward in addition to the task reward. We then substituted the mixed advantages for the standard advantages in Eq. (36) to update the policy. To compute the Lagrange multiplier update, we computed the average constraint violation (gradient) on a minibatch of L states obtained from rollouts of the current policy

$$\Delta_n^k = \frac{1}{L}\sum_{\ell=1}^{L}(v^\pi(s_\ell) - \theta_n)$$

and updated the Lagrange multiplier(s) as

$$\mu_n^{k+1} = \max\{\mu_n^k + \eta_\mu \Delta_n^k, 0\} \quad \text{for } n = 1, 2, \ldots, N.$$

To add ReLOAD, we repeated the above steps but used the optimistic mixed advantages

$$\tilde{A}^{\pi_\mu^k}_{\mu^k}(s,a) = 2A^{\pi_\mu^k}_{\mu^k}(s,a) - A^{\pi_\mu^{k-1}}_{\mu^{k-1}}(s,a)$$

and optimistic values to compute the constraint violations:

$$\tilde{v}_n^{\theta^k}(s_\ell) = 2v_n^{\theta^k}(s_\ell) - v_n^{\theta^{k-1}}(s_\ell) \quad \text{for } n = 1, 2, \ldots, N.$$

To compute the optimistic advantages and values, we maintain a copy of the previous parameters θ^{k-1} to evaluate on the new trajectories collected by π_{θ^k}. In practice, we also use entropy regularization on the policy, making our implementation of μ-MDPO equivalent to *magnetic mirror-descent* (MMD; Sokota et al., 2022).

IMPALA The Importance-Weighted Actor-Learner Architecture (IMPALA Espeholt et al., 2018) is a distributed actor-critic agent which uses a set of actors to generate trajectories of experience which are then used by one or more learners to update the policy and value function parameters off-policy. To correct for the disjunction between the actor parameters and the learner parameters, updates are computed using an importance-weighting correction technique called V-trace. Unlike the other base agents with which we paired ReLOAD, IMPALA does not by default use a trust region approach for policy optimization. Therefore, our first modification was to add one via an inner loop in the style of MDPO. We then also added modifications for Lagrangian optimization and optimism as in MDPO. However, since we applied the IMPALA-based agents to large-scale tasks, optimization was inherently less stable, and so we used a sigmoid function to bound the Lagrange multiplier as done in Zahavy et al. (2021b); Stooke et al. (2020), leading to mixed advantages of the form:

$$A^{\theta^k}_{\mu^k}(s,a) = \left(N - \sum_{n=1}^{N}\sigma(\mu_n^k)\right)A_0^{\theta^k}(s,a) + \sum_{n=1}^{N}\sigma(\mu_n^k)A_n^{\theta^k}(s,a).$$

The overall procedure for the learner is summarized in Algorithm 3, Algorithm 4, and Algorithm 5. Code for V-trace can be found at this link: `https://github.com/deepmind/rlax/blob/master/rlax/_src/vtrace.py`.

Distributional MPO (DMPO) Maximum a posteriori policy optimization (MPO; Abdolmaleki et al., 2018) is a deep policy iteration-style algorithm whose improvement step can be broken into two stages which are reminiscent of the "E" and "M" steps of the EM algorithm. In the E-step, a non-parametric variational "policy" $\nu^k(a|s)$ is constructed using off-policy samples as follows:

$$\nu^k(a|s) \propto \pi_{\theta^k}(a|s)\exp\left(\frac{q_{\theta^k}(s,a)}{\omega^\star}\right) \tag{37}$$

where $\omega^\star$ is the minimizer of the convex dual function

$$g(\omega) = \omega\epsilon + \omega\mathbb{E}_{s\sim d_\pi}\mathbb{E}_{a\sim\pi_{\theta^k}}\exp\left(q_{\theta^k}(s,a)/\omega\right)$$

where ϵ is a constant defining the radius of a KL trust region around the current policy. This (soft-)improved policy is then distilled into the parametric policy in an approximate "M" step using another KL trust region:

$$\theta^{k+1} = \operatorname*{argmax}_\theta \mathbb{E}_{d_\nu}\mathbb{E}_{\nu^k}\log\pi_\theta(a|s) \quad \text{s.t.} \quad \mathbb{E}_{d_\nu}\mathsf{KL}[\pi_{\theta^k}(\cdot|s)||\pi_\theta(\cdot|s)] < \epsilon.$$

In *distributional* MPO (DMPO; Abdolmaleki et al., 2020), in order to provide more accurate value estimates, the standard critic is replaced with a distributional critic using a categorical representation of the return distribution (Bellemare et al., 2017). To add Lagrangian optimization to DMPO (μ-DMPO), we used the same strategy as in IMPALA to update both the Lagrange multipliers and to compute the mixed q-values $q^\mu_{\theta^k}$ (rather than advantages in this case) to plug into Eq. (37). Similarly, for ReLOAD we computed the optimistic mixed q-values

$$\tilde{q}^{\mu^k}_{\theta^k}(s,a) = q^{\mu^k}_{\theta^k}(s,a) - q^{\mu^{k-1}}_{\theta^{k-1}}(s,a)$$

Algorithm 3 ReLOAD-IMPALA `LearnerStep`

Require: network parameters θ^k, previous parameters θ^{k-1}, Lagrange multiplier μ^k, previous Lagrange multiplier μ^{k-1}, total learner updates K, trajectory rollouts $\{\tau_n\}_{n=1}^N$, number of inner loop updates m, constraint value φ_e
Compute trust region step-size: $\eta^k \leftarrow 1 - k/K$
$\pi_{\theta^k}(\cdot|s_t), \pi_{\theta^{k-1}}(\cdot|s_t), v^0_{\theta^{k-1}}(s_t), v^1_{\theta^{k-1}}(s_t), \log \nu(a_t|s_t), \rho_{\theta^{k-1}}(s_t, a_t) \leftarrow$`GetFixedOutputs`$(\theta^k, \theta^{k-1}, \{\tau_n\}_{n=1}^N)$
Set inner loop parameters $\theta^k_{(0)} \leftarrow \theta^k$
for $i = 0, \ldots, m-1$ **do**

$$\theta^k_{(i+1)} \leftarrow \theta^k_{(i)} - \epsilon \nabla_\theta \mathcal{L}(\theta^k_{(i)}, \eta^k, \mu^k, \pi_{\theta^k}(\cdot|s_t), \pi_{\theta^{k-1}}(\cdot|s_t), v^0_{\theta^{k-1}}(s_t), v^1_{\theta^{k-1}}(s_t), \log \nu(a_t|s_t), \rho_{\theta^{k-1}}(s_t, a_t), \mu^{k-1}, \{\tau_n\}_{n=1}^N)$$

end for
Reset network parameters $\theta^{k+1} \leftarrow \theta^k_{(m)}$
Update Lagrange multiplier:

$$\tilde{\Delta}^k_1 \leftarrow 2(v^1_{\theta^{k+1}}(s_t) - \varphi_1) - (v^1_{\theta^{k-1}}(s_t) - \varphi_1) \quad \text{(optimistic constraint violation)}$$
$$\mu^{k+1} \leftarrow \max\{\mu^k + \epsilon_\mu \tilde{\Delta}^k_1,\ 0\} \quad \text{(projected gradient ascent)}$$

Algorithm 4 ReLOAD-IMPALA `GetFixedOutputs`

// A function to compute everything that does not change within the inner loop.
Require: network parameters θ^k, previous parameters θ^{k-1}, trajectory rollouts $\{\tau_n\}_{n=1}^N$
Unroll current parameters: $\pi_{\theta^k}(\cdot|s_t), \_, \_ \leftarrow$ `Unroll`$(\{\tau_n\}, \theta^k)$
Unroll previous parameters: $\pi_{\theta^{k-1}}(\cdot|s_t), v^0_{\theta^{k-1}}(s_t), v^1_{\theta^{k-1}}(s_t) \leftarrow$ `Unroll`$(\{\tau_n\}, \theta^{k-1})$
Compute behavior policy log-probs $\log \nu(a_t|s_t)$
Compute previous policy importance weights: $\rho_{\theta^{k-1}}(s_t, a_t) = \frac{\pi_{\theta^{k-1}}(a_t|s_t)}{\nu^k(a_t|s_t)}$
Return $\pi_{\theta^k}(\cdot|s_t), \pi_{\theta^{k-1}}(\cdot|s_t), v^0_{\theta^{k-1}}(s_t), v^1_{\theta^{k-1}}(s_t), \log \nu(a_t|s_t), \rho_{\theta^{k-1}}(s_t, a_t)$

and plugged them into Eq. (37). The MPO policy update code we used can be found here: `https://github.com/deepmind/rlax/blob/master/rlax/_src/mpo_ops.py`.

Multi-Objective MPO (MO-MPO) Rather than scalarize the task and constraint rewards into a single, non-stationary reward via Lagrangian optimization, multi-objective MPO (MO-MPO; Abdolmaleki et al., 2020) instead estimates a separate variational policy for each task and constraint reward $\nu^k_n(a|s)$ by solving

$$\max_{\nu^k_n} \mathbb{E}_{d_{\pi_{\theta^k}}, \nu^k_n} q^k_n(s, a) \quad \text{s.t.} \quad \mathbb{E}_{d_{\pi_{\theta^k}}} \mathsf{KL}[\nu^k_n(a|s) || \pi_{\theta^k}(a|s)] < \epsilon_n$$

for $n = 0, 1, \ldots, N$, where ϵ_n encodes the preferences/thresholds/desired trade-offs among the task and constraint rewards. MO-MPO then distills each of these improved policies into the parametric policy:

$$\theta^{k+1} = \operatorname*{argmax}_\theta \sum_{n=0}^N \mathbb{E}_{d_{\nu^k_n}} \mathbb{E}_{\nu^k_n} \log \pi_\theta(a|s) \quad \text{s.t.} \quad \mathbb{E}_{d_\nu} \mathsf{KL}[\pi_{\theta^k}(\cdot|s) || \pi_\theta(\cdot|s)] < \beta.$$

where $\beta \in \mathbb{R}_+$ is the radius of the trust region.

Reward-Constrained D4PG (RC-D4PG) Deep deterministic policy gradients (DDPG; Lillicrap et al., 2015) is a DRL algorithm which uses a deterministic policy $\pi_{\theta_\pi}(s)$ to maximize the objective $J(\theta) = \mathbb{E}[q_{\theta_q}(s, a) \mid s = s_t, a = \pi_{\theta_\pi}(s_t)]$ (where $\theta = \{\theta_\pi, \theta_q\}$) using the deterministic policy gradient estimate $\nabla_{\theta_\pi} J(\theta) = \mathbb{E}[\nabla_a q_{\theta_q}(s, a) \nabla_{\theta_\pi} \pi_{\theta_\pi}(s_t)]$. Distributed distributional DDPG (D4PG; Barth-Maron et al., 2018) is a distributed algorithm which augments DDPG with distributional critics using a categorical representation for the return distribution (Bellemare et al., 2017). Reward-constrained D4PG (RC-D4PG; Calian et al., 2020) augments D4PG for Lagrangian optimization in the same way as μ-IMPALA and μ-MDPO augment their respective base algorithms, with the exception that rather than update the Lagrange multiplier by using constraint value estimates obtained from samples from the most recent policy, RC-D4PG maintains a buffer of constraint

Algorithm 5 ReLOAD-IMPALA `Loss`:
$\mathcal{L}(\theta^k_{(i)}, \eta^k, \mu^k, \pi_{\theta^k}(\cdot|s_t), \pi_{\theta^{k-1}}(\cdot|s_t), v^0_{\theta^{k-1}}(s_t), v^1_{\theta^{k-1}}(s_t), \log \nu(a_t|s_t), \rho_{\theta^{k-1}}(s_t, a_t), \mu^{k-1}, \{\tau_n\}_{n=1}^N)$

1: Unroll trajectories:

$$\{\pi_{\theta^k_{(i)}}(\cdot|s_t), v^0_{\theta^k_{(i)}}(s_t), v^1_{\theta^k_{(i)}}(s_t)\} \leftarrow \texttt{Unroll}(\{\tau_n\}, \theta^k_{(i)})$$

2: Compute importance ratios wrt behavior policy ν^k:

$$\rho_{\theta^k_{(i)}}(s_t, a_t) = \frac{\pi_{\theta^k_{(i)}}(a_t|s_t)}{\nu^k(a_t|s_t)}$$

3: Compute rewards: $r^0_t(\theta^k_{(i)}), r^1_t(\theta^k_{(i)}) \leftarrow \texttt{ComputeRewards}(r_t, \theta^{(i)}_k)$

4: Compute TD errors and advantages:

$$\delta^0_{(i)}, A^0_{\theta^k_{(i)}} \leftarrow \texttt{V-Trace}(v^0_{\theta^k_{(i)}}, r^0_t(\theta^k_{(i)}), \rho_{\theta^k_{(i)}}); \quad \delta^1_{(i)}, A^1_{\theta^k_{(i}} \leftarrow \texttt{V-Trace}(v^1_{\theta^k_{(i)}}, r^1_t(\theta^k_{(i)}), \rho_{\theta^k_{(i)}})$$
$$\_, A^0_{\theta^{k-1}} \leftarrow \texttt{V-Trace}(v^0_{\theta^{k-1}}, r^0_t(\theta^k_{(i)}), \rho_{\theta^{k-1}}); \quad \_, A^1_{\theta^{k-1}} \leftarrow \texttt{V-Trace}(v^1_{\theta^{k-1}}, r^1_t(\theta^k_{(i)}), \rho_{\theta^{k-1}})$$

5: Weight cumulants:

$$A^{\mu^k}_{\theta^k_{(i)}} \leftarrow (1 - \sigma(\mu^k))A^0_{\theta^k_{(i)}} - \sigma(\mu^k)A^1_{\theta^k_{(i)}}; \quad A^{\mu^{k-1}}_{\theta^{k-1}} \leftarrow (1 - \sigma(\mu^{k-1}))A^0_{\theta^{k-1}} - \sigma(\mu^{k-1})A^1_{\theta^{k-1}}$$

6: Compute optimistic advantages: $\tilde{A}^{\mu^k}_{\theta^k_{(i)}} \leftarrow 2A^{\mu^k}_{\theta^k_{(i)}} - A^{\mu^{k-1}}_{\theta^{k-1}}$

7: Policy loss:

$$\mathcal{L}_\pi = \sum_t -\rho_{\theta^k_{(i)}}(s_t, a_t)\tilde{A}^{\mu^k}_{\theta^k_{(i)}}(s_t, a_t)\log \pi_{\theta^k_{(i)}}(a_t|s_t) + \frac{1}{\eta^k}\mathsf{KL}[\pi_{\theta^k_{(i)}}(\cdot|s_t), \pi_{\theta^k}(\cdot|s_t)]$$

8: Value loss:

$$\mathcal{L}_V = \sum_t \delta^0_{(i)}(s_t, a_t)^2 + \delta^1_{(i)}(s_t, a_t)^2$$

9: Regularization loss:

$$\mathcal{L}_\Omega = \sum_t \mathsf{KL}[\pi_{\theta^k_{(i)}}(\cdot|s_t)||\mathcal{N}(0, I)]$$

10: Return $\alpha_\pi\mathcal{L}_\pi + \alpha_V\mathcal{L}_V + \alpha_\Omega\mathcal{L}_\Omega$

returns from across training and uses them to compute constraint violations.

MetaL Meta-gradients for the Lagrange learning rate (MetaL; Calian et al., 2020) extends RC-D4PG by using meta-gradients (Xu et al., 2018) to meta-learn the learning rate for the Lagrange multiplier η_μ with the goal of stabilizing optimization. Specifically, they define a set of inner losses for the actor, critic, and Lagrange multiplier corresponding to the standard components of Lagrangian optimization for CMDPs while also defining an outer loss $L_{outer} = L_{critic}(\theta^{k+1}_q(\eta^k_\mu), \mu^{k+1}(\eta^k_\mu))$. That is, the outer loss is the standard critic loss evaluated using the updated critic parameters and Lagrange multipliers from the inner loop (which are themselves functions of the current Lagrange multiplier learning rate). The Lagrange multiplier learning rate is then updated using meta-gradients on this loss.

G. Further Experimental Details

Performance Measures In simple cases like the paradoxical CMDP, we know the optimal solution exactly and can compute, e.g., the L_2 distance between the iterates produced by the algorithm (d^k_π, μ^k) and the SP $(d^\star_\pi, \mu^\star)$. In more

Algorithm 6 ReLOAD-MDPI

Require: CMDP $\mathcal{M}_C$, step sizes $\{\eta_\pi, \eta_\mu\} > 0$
Initialize π^1, μ^1, π^0, μ^0
for $k = 1, \ldots, K$ **do**
  $\{q_n\}_{n=0}^N \leftarrow \texttt{PolicyEval}(\mathcal{M}_C, \pi^k)$
  $q_\mu^k \leftarrow -q_0^k + \sum_{n=1}^N \mu_n^k q_n^k$ (mixed q-values)
  $v_{1:N}^k \leftarrow [\langle q_1^k, \pi^k \rangle, \ldots, \langle q_N^k, \pi^k \rangle]^\top$
  Update policy with OMWU:

$$\pi^{k+1} = \frac{\pi^k \exp\left((2q_\mu^k - q_\mu^{k-1})/\eta_\pi\right)}{\left\langle \pi^k \exp\left((2q_\mu^k - q_\mu^{k-1})/\eta_\pi\right), \mathbf{1} \right\rangle}$$

  Update Lagrange multiplier with projected OGA:

$$\mu^{k+1} = \max\{\mu^k + \eta_\mu(2v_{1:N}^k - v_{1:N}^{k-1} - \theta),\ 0\}$$

end for
return π^K, μ^K

complex settings such as the RWRL suite and control suite, previous work, e.g., Calian et al. (2020) and Huang et al. (2022) has used the *penalized reward* $R_{\text{penalized}}$:

$$R_{\text{penalized}} = R_0 - \sum_{n=1}^{N} \max\{R_n - \theta_n, 0\}, \tag{38}$$

which is the average task return R_0 minus the *constraint overshoot*—how much the average constraint returns violate their associated constraint thresholds. (No bonus is given for being further beneath the threshold than necessary.) However, this metric ignores the effect of the Lagrange multiplier, which in Lagrangian optimization acts on a coefficient on the constraint term, e.g., $\mathcal{L} = v_0 + \mu(v_1 - \theta_1)$ for a single constraint. We therefore prefer the *weighted reward* R_{weighted} given by

$$R_{\text{weighted}} = R_0 - \sum_{n=1}^{N} \mu_n^\star \max\{R_n - \theta_n, 0\}, \tag{39}$$

where $\mu_\star^n$ is the optimal Lagrange multiplier. In our case, for large-scale environments we use a sigmoid to bound the Lagrange multiplier as described in Appendix F. Therefore, because the agent is maximizing the weighted value/advantage $(1 - \sigma(\mu))A^0 + \sigma(\mu)A^1$, we divide by $1 - \sigma(\mu)$ to give

$$R_{\text{weighted}} = R_0 - \sum_{n=1}^{N} \hat{\mu}_n^\star \max\{R_n - \theta_n, 0\} \quad \text{where} \quad \hat{\mu}_n^\star \coloneqq \frac{\sigma(\mu_n^\star)}{1 - \sigma(\mu_n^\star)}. \tag{40}$$

This approach is also used by Stooke et al. (2020) and Zahavy et al. (2022). To estimate the optimal Lagrange multiplier(s), we ran μ- agents for 8 random seeds on each task and averaged the Lagrange multipliers, as they enjoy guarantees for average-iterate convergence.

The Paradoxical CMDP Tabular agents were trained for 500 episodes, with each episode lasting 10 time steps and with the agent starting in a uniformly-randomly chosen state. The discount factor γ was 0.9. μ-MDPI-Avg was obtained by maintaining a running average of the iterates of μ-MDPI. DRL agents (μ-MDPO and ReLOAD-MDPO) were trained for 30,000 episodes. The policy was parameterized as an MLP with a single hidden layer with 16 units. States were provided as a one-hot encoding. The agents were trained using RMSProp with an initial learning rate of 6e-4, which decayed linearly to 1e-4 over the course of training. There were 5 inner loop steps to approximate the MD update with a fixed MD step size of 0.25.

Catch The Catch task was as described by Osband et al. (2019) with the addition of the constraint reward and threshold as described in Section 5. μ-MDPO and ReLOAD-MDPO agents were trained for 30,000 episodes with the same hyperparameter settings as for the Paradoxical CMDP, with the exception that the policy network had two hidden layers, each with 32 hidden units.

Domain, Task	Constrained Quantity	Observation Dims	θ
Walker, Walk	Height	24	0.5
Walker, Walk	Velocity	14-23	155
Reacher, Easy	Velocity	4-5	1.02
Quadruped, Walk	Torque	54-77	610
Humanoid, Walk	Height	21	400

Table 1. Experiment settings for oscillating control suite experiments. For multidimensional quantities, the constraint was placed on the L_2 norm.

The RWRL Suite Hyperparameters and training regimes for 0.1-D4PG, RC-D4PG, and MetaL are as reported by Calian et al. (2020). For μ-DMPO and ReLOAD-DMPO, hyperparameters are given in Table 8. All task settings are as reported in Calian et al. (2020).

Oscillating Control Suite Control suite domains and accompanying thresholds are described in Table 1. OGD-IMPALA, DMPO is implemented by augmenting μ-IMPALA, DMPO with the optimistic gradient descent optimizer from the Optax library, whose code is available at: `https://github.com/deepmind/optax/blob/master/optax/_src/alias.py`. PID control is implemented as described in Stooke et al. (2020) with the same values of $K_P = 0.95$ and $K_D = 0.9$ as used in that work. For IMPALA, which does not normally have a trust region component to the policy update, RNTR-IMPALA implemented ReLOAD without a trust region (that is, only using optimistic advantages/value estimates).

H. Further Experimental Results

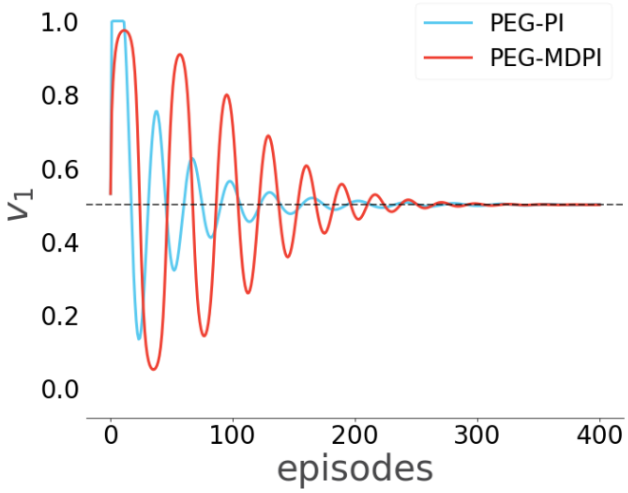

Figure 10. **Tabular PEG-based policy iteration in the paradoxical CMDP.** PEG-PI uses projected optimistic gradient descent instead of MWU for the policy, while PEG-MDPI uses optimistic MWU for the policy update, which here produces the same updates as ReLOAD. Both variants converge in the last-iterate.

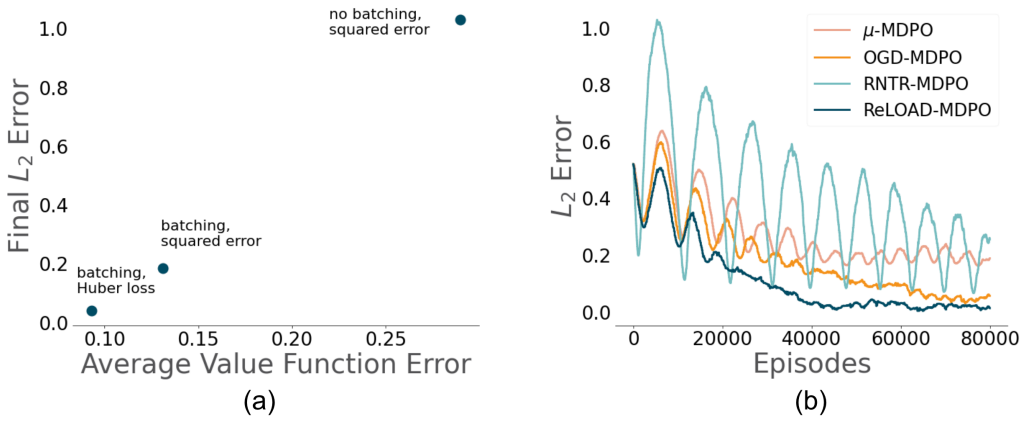

Figure 11. **Policy evaluation and ablations in the paradoxical CMDP.** (a) To demonstrate the importance of accurate value estimation on performance, we plotted the L_2 distance of the final (d_π^K, μ^K) pair obtained by ReLOAD-MDPO from the SP of the paradoxical CMDP in Fig. 3a as a function of the average error in the value estimates over the course of training. We can see that performance gets dramatically worse as the value function error increases, and that adding standard techniques like batching and Huber loss rather than squared error loss improves value estimates sufficiently to allow for strong performance of the overall algorithm. (b) Here, we compare ReLOAD-MDPO against various ablations by measuring the L_2 distance from the SP of the paradoxical CMDP over the course of training. We can see that both μ-MDPO and ReLOAD without a trust region (RNTR-) fail to converge. However, performing OGD directly on the policy parameters, rather than via optimistic value estimates, performs nearly as well as ReLOAD. This is likely because the CMDP only has two states, so using the gradient computed from value estimates obtained from an old batch of data will likely be close to that obtained from the current batch of data, as there's more likely to be significant overlap. If we look to higher-dimensional tasks like control suite (Fig. 8), there is a much larger gap between the direct OGD approach and ReLOAD as the state-action pairs used for value estimation are more likely to differ from one minibatch to another.

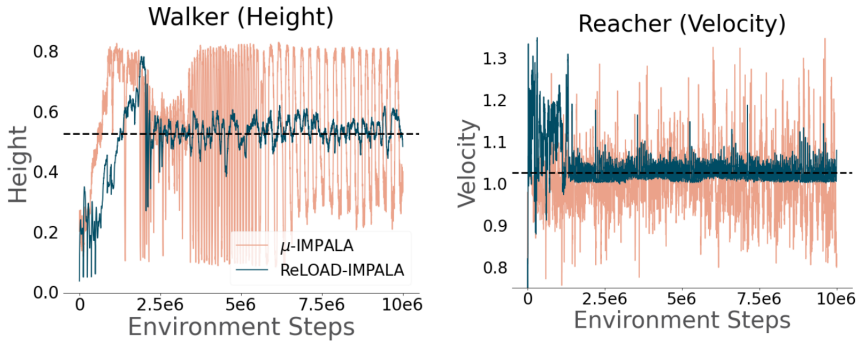

Figure 12. Learning curves for μ-IMPALA and ReLOAD-IMPALA on `Walker, Walk` with a height constraint (left) and `Reacher, Easy` with a velocity constraint (right). The horizontal dotted line indicates the value of the constraint. In each case, ReLOAD-IMPALA significantly dampens oscillations compared to μ-IMPALA.

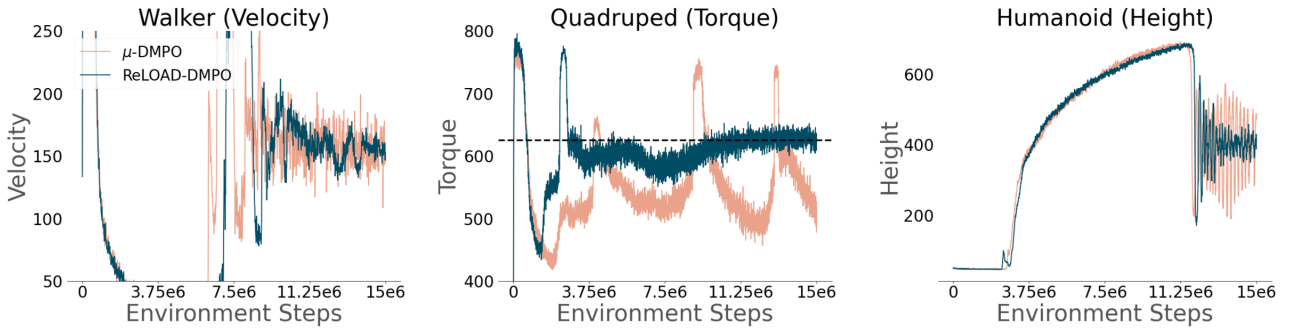

Figure 13. Learning curves for μ-DMPO and ReLOAD-DMPO on `Walker, Walk` with a velocity constraint (left), `Quadruped, Walk` with a torque constraint (center), and `Humanoid, Walk` with a height constraint (right). The horizontal dotted line indicates the value of the constraint. In each case, ReLOAD-DMPO significantly dampens oscillations compared to μ-DMPO.

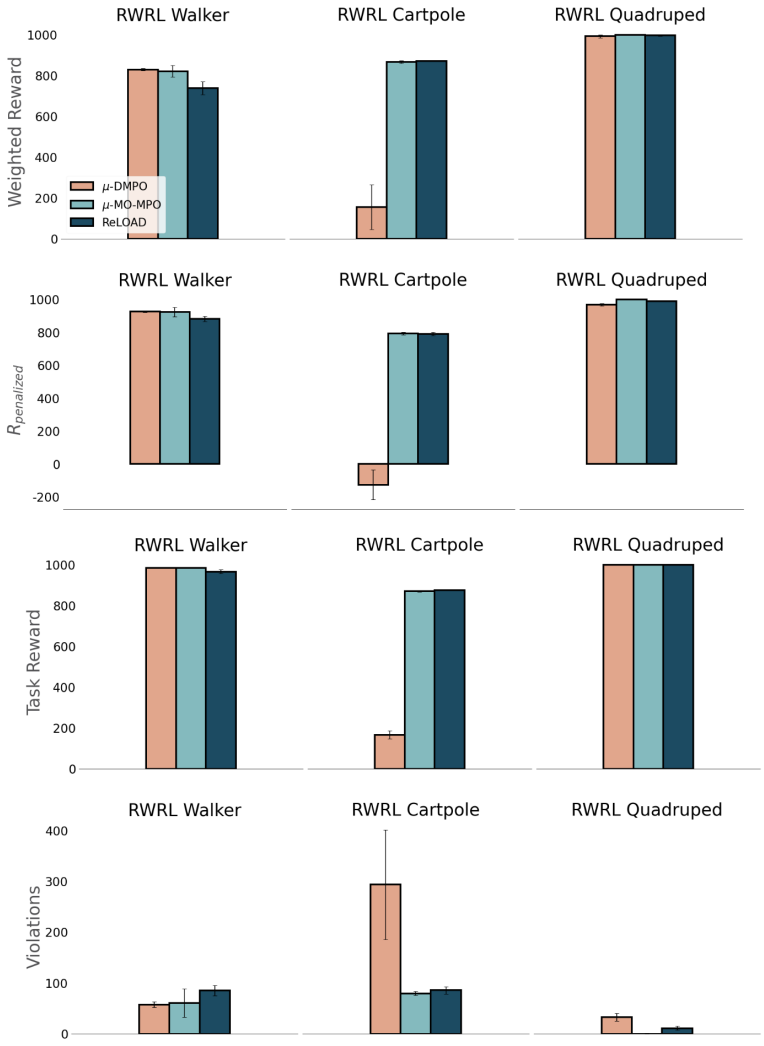

Figure 14. **ReLOAD-DMPO outperforms μ-DMPO and matches the performance of MO-DMPO on the RWRL Suite.** Here, we trained agents on the comparatively easy RWRL settings used by (Huang et al., 2022) over 8 random seeds. Error bars denote one standard error.

Domain	Algorithm	Safety-Coeff/Threshold	Weighted Reward	$\mathbf{R}_{\mathbf{penalized}}$	Task Reward	Constraint Violation
Walker	0.1-D4PG	0.05/0.057	-153.8 ± 12.1	552.8 ± 12.0	979.6 ± 0.3	426.8 ± 12.1
		0.05/0.077	-175.4 ± 20.0	544.3 ± 19.9	979.1 ± 0.5	434.8 ± 20.0
		0.05/0.097	-75.1 ± 20.5	581.8 ± 20.3	978.6 ± 0.4	396.8 ± 20.5
		0.1/0.057	533.9 ± 9.0	811.3 ± 8.7	978.9 ± 0.4	167.6 ± 9.0
		0.1/0.077	630.2 ± 7.9	847.7 ± 7.7	979.1 ± 0.3	131.4 ± 7.9
		0.1/0.097	652.4 ± 5.8	855.8 ± 5.7	978.7 ± 0.3	122.9 ± 5.8
		0.2/0.057	982.0 ± 0.2	982.0 ± 0.2	982.0 ± 0.2	0.0 ± 0.0
		0.2/0.077	981.8 ± 0.2	981.8 ± 0.2	981.8 ± 0.2	0.0 ± 0.0
		0.2/0.097	981.5 ± 0.3	981.5 ± 0.3	981.5 ± 0.3	0.0 ± 0.0
		0.3/0.057	982.7 ± 0.3	982.7 ± 0.3	982.7 ± 0.3	0.0 ± 0.0
		0.3/0.077	981.8 ± 0.3	981.8 ± 0.3	981.8 ± 0.3	0.0 ± 0.0
		0.3/0.097	982.0 ± 0.1	982.0 ± 0.1	982.0 ± 0.1	0.0 ± 0.0
	RC-D4PG	0.05/0.057	268.7 ± 34.6	296.1 ± 28.3	312.7 ± 34.0	16.6 ± 6.4
		0.05/0.077	325.9 ± 44.4	346.6 ± 36.9	359.0 ± 43.8	12.5 ± 7.2
		0.05/0.097	329.1 ± 33.4	334.8 ± 32.8	338.2 ± 33.4	3.4 ± 1.3
		0.1/0.057	914.3 ± 10.9	915.0 ± 10.8	915.5 ± 10.9	0.5 ± 0.4
		0.1/0.077	942.7 ± 1.6	945.4 ± 1.2	947.1 ± 1.4	1.7 ± 0.8
		0.1/0.097	956.1 ± 3.3	959.3 ± 3.3	961.3 ± 3.2	2.0 ± 0.8
		0.2/0.057	980.5 ± 0.3	981.0 ± 0.4	981.4 ± 0.2	0.3 ± 0.2
		0.2/0.077	979.1 ± 1.0	980.7 ± 1.1	981.7 ± 0.3	1.0 ± 1.0
		0.2/0.097	977.7 ± 0.8	980.3 ± 0.9	981.9 ± 0.3	1.6 ± 0.8
		0.3/0.057	978.4 ± 0.5	980.5 ± 0.5	981.8 ± 0.3	1.3 ± 0.4
		0.3/0.077	978.6 ± 0.9	980.5 ± 0.7	981.6 ± 0.3	1.1 ± 0.8
		0.3/0.097	979.6 ± 0.6	980.9 ± 0.6	981.7 ± 0.4	0.8 ± 0.5
	MetaL	0.05/0.057	-168.5 ± 10.5	540.0 ± 9.6	968.0 ± 1.5	428.0 ± 10.4
		0.05/0.077	-138.8 ± 13.7	551.7 ± 13.5	968.9 ± 1.4	417.1 ± 13.6
		0.05/0.097	-131.1 ± 13.0	553.9 ± 12.2	967.8 ± 1.0	413.8 ± 13.0
		0.1/0.057	663.4 ± 19.8	853.7 ± 19.8	968.6 ± 1.2	114.9 ± 19.8
		0.1/0.077	627.4 ± 17.8	841.7 ± 17.1	971.2 ± 1.1	129.5 ± 17.8
		0.1/0.097	766.9 ± 19.8	893.3 ± 19.7	969.7 ± 0.7	76.4 ± 19.7
		0.2/0.057	981.3 ± 0.5	982.0 ± 0.6	982.5 ± 0.2	0.4 ± 0.4
		0.2/0.077	980.2 ± 0.3	981.5 ± 0.3	982.3 ± 0.1	0.8 ± 0.3
		0.2/0.097	978.6 ± 0.8	980.7 ± 0.7	981.9 ± 0.3	1.3 ± 0.7
		0.3/0.057	977.7 ± 0.9	980.6 ± 1.0	982.3 ± 0.4	1.7 ± 0.8
		0.3/0.077	978.4 ± 0.5	980.6 ± 0.6	981.9 ± 0.2	1.3 ± 0.5
		0.3/0.097	979.6 ± 0.5	980.9 ± 0.5	981.6 ± 0.2	0.8 ± 0.5
	ReLOAD	0.05/0.057	367.0 ± 114.8	626.9 ± 113.4	783.8 ± 96.9	157.0 ± 9.6
		0.05/0.077	423.3 ± 80.1	716.7 ± 43.5	893.9 ± 26.1	177.2 ± 24.0
		0.05/0.097	315.8 ± 43.1	677.7 ± 28.9	896.4 ± 22.1	218.6 ± 10.3
		0.1/0.057	754.1 ± 39.0	875.4 ± 23.7	948.7 ± 15.8	73.3 ± 10.1
		0.1/0.077	717.9 ± 26.8	877.1 ± 11.2	973.3 ± 4.8	96.2 ± 9.9
		0.1/0.097	746.9 ± 28.4	889.3 ± 12.0	975.3 ± 4.7	86.0 ± 10.4
		0.2/0.057	907.7 ± 6.9	950.6 ± 5.0	976.6 ± 4.1	26.0 ± 1.9
		0.2/0.077	881.5 ± 9.7	940.7 ± 5.9	976.4 ± 4.3	35.7 ± 3.1
		0.2/0.097	882.7 ± 7.8	942.1 ± 5.2	977.9 ± 4.0	35.9 ± 2.4
		0.3/0.057	938.3 ± 5.8	963.0 ± 5.0	977.9 ± 4.3	14.9 ± 1.2
		0.3/0.077	930.4 ± 5.6	960.5 ± 4.6	978.8 ± 4.0	18.2 ± 1.3
		0.3/0.097	925.8 ± 5.8	958.3 ± 4.7	977.9 ± 4.0	19.6 ± 1.5

Table 2. Mean performance for `RWRL-Walker` averaged over 8 random seeds for each of the 12 constraint settings in the RWRL suite, where lower values of the safety coefficient and threshold indicate more challenging tasks. $\pm$ values denote one standard error.

Domain	Algorithm	Safety-Coeff/Threshold	Weighted Reward	$\mathbf{R}_{\mathbf{penalized}}$	Task Reward	Constraint Violation
Cartpole	0.1-D4PG	0.05/0.07	833.8 ± 1.3	352.6 ± 1.2	851.7 ± 0.2	499.1 ± 1.3
		0.05/0.09	834.3 ± 1.5	372.6 ± 1.5	851.5 ± 0.1	478.9 ± 1.5
		0.05/0.115	835.0 ± 0.9	399.7 ± 0.8	851.2 ± 0.1	451.5 ± 0.9
		0.1/0.07	844.9 ± 0.7	568.2 ± 0.7	855.2 ± 0.1	287.0 ± 0.7
		0.1/0.09	845.5 ± 1.0	588.3 ± 1.0	855.1 ± 0.2	266.8 ± 1.0
		0.1/0.115	846.4 ± 0.9	612.9 ± 0.9	855.1 ± 0.2	242.2 ± 0.9
		0.2/0.07	850.7 ± 4.8	763.1 ± 6.3	853.9 ± 2.4	90.8 ± 4.2
		0.2/0.09	854.7 ± 1.3	791.3 ± 1.5	857.0 ± 0.5	65.7 ± 1.2
		0.2/0.115	852.0 ± 4.4	807.6 ± 5.8	853.6 ± 2.4	46.0 ± 3.7
		0.3/0.07	857.5 ± 0.3	825.5 ± 0.3	858.7 ± 0.1	33.3 ± 0.3
		0.3/0.09	858.4 ± 0.6	845.4 ± 0.5	858.9 ± 0.1	13.5 ± 0.5
		0.3/0.115	856.7 ± 1.9	856.6 ± 2.0	856.7 ± 1.9	0.1 ± 0.1
	RC-D4PG	0.05/0.07	278.6 ± 114.7	158.7 ± 21.7	283.1 ± 88.2	124.4 ± 73.3
		0.05/0.09	260.4 ± 84.4	144.8 ± 43.9	264.7 ± 66.4	119.9 ± 52.2
		0.05/0.115	422.8 ± 105.5	269.1 ± 23.0	428.5 ± 83.5	159.4 ± 64.4
		0.1/0.07	262.6 ± 48.7	180.4 ± 49.6	265.7 ± 43.1	85.3 ± 22.6
		0.1/0.09	534.4 ± 104.0	296.2 ± 93.7	543.3 ± 81.9	247.1 ± 64.1
		0.1/0.115	710.0 ± 45.1	537.5 ± 35.5	716.4 ± 38.8	178.9 ± 23.0
		0.2/0.07	310.9 ± 90.6	298.3 ± 86.4	311.4 ± 90.3	13.1 ± 7.5
		0.2/0.09	770.0 ± 18.0	760.6 ± 16.4	770.3 ± 17.7	9.7 ± 3.0
		0.2/0.115	847.7 ± 2.4	847.4 ± 2.4	847.7 ± 2.4	0.3 ± 0.1
		0.3/0.07	845.2 ± 3.4	844.7 ± 3.6	845.3 ± 3.4	0.6 ± 0.3
		0.3/0.09	857.3 ± 0.2	857.1 ± 0.1	857.3 ± 0.1	0.2 ± 0.1
		0.3/0.115	859.2 ± 0.3	858.7 ± 0.2	859.2 ± 0.1	0.4 ± 0.3
	MetaL	0.05/0.07	832.9 ± 1.1	348.2 ± 1.1	850.9 ± 0.2	502.7 ± 1.1
		0.05/0.09	829.5 ± 2.3	365.1 ± 2.5	846.8 ± 2.1	481.6 ± 1.0
		0.05/0.115	834.4 ± 1.7	391.9 ± 1.6	850.9 ± 0.2	458.9 ± 1.7
		0.1/0.07	843.4 ± 1.0	580.3 ± 0.9	853.2 ± 0.2	272.9 ± 1.0
		0.1/0.09	844.3 ± 2.3	599.3 ± 2.3	853.4 ± 0.2	254.1 ± 2.3
		0.1/0.115	845.5 ± 1.2	623.7 ± 1.3	853.7 ± 0.1	230.0 ± 1.2
		0.2/0.07	851.1 ± 2.6	787.3 ± 3.3	853.5 ± 2.0	66.2 ± 1.6
		0.2/0.09	850.8 ± 3.8	809.3 ± 4.2	852.3 ± 2.1	43.1 ± 3.1
		0.2/0.115	855.1 ± 2.1	831.8 ± 1.9	855.9 ± 0.2	24.1 ± 2.1
		0.3/0.07	856.1 ± 1.9	839.9 ± 1.5	856.7 ± 0.4	16.8 ± 1.9
		0.3/0.09	852.8 ± 3.2	848.5 ± 4.1	852.9 ± 2.7	4.4 ± 1.6
		0.3/0.115	857.3 ± 2.0	856.9 ± 2.1	857.3 ± 2.0	0.4 ± 0.2
	ReLOAD	0.05/0.07	875.8 ± 2.6	798.3 ± 9.4	878.7 ± 2.4	80.4 ± 7.4
		0.05/0.09	871.5 ± 3.0	781.0 ± 11.2	874.8 ± 2.8	93.8 ± 8.7
		0.05/0.115	869.1 ± 3.0	769.0 ± 11.3	872.8 ± 2.7	103.9 ± 8.9
		0.1/0.07	875.8 ± 2.6	801.9 ± 9.0	878.5 ± 2.4	76.6 ± 6.9
		0.1/0.09	869.5 ± 2.9	780.7 ± 9.9	872.8 ± 2.7	92.1 ± 7.6
		0.1/0.115	871.7 ± 3.0	786.6 ± 10.1	874.8 ± 2.8	88.2 ± 7.7
		0.2/0.07	869.9 ± 2.9	795.1 ± 8.1	872.7 ± 2.7	77.5 ± 6.0
		0.2/0.09	871.9 ± 3.0	798.7 ± 8.9	874.7 ± 2.8	76.0 ± 6.5
		0.2/0.115	869.8 ± 2.9	789.2 ± 8.6	872.7 ± 2.7	83.5 ± 6.4
		0.3/0.07	870.1 ± 2.8	802.9 ± 7.5	872.6 ± 2.7	69.7 ± 5.5
		0.3/0.09	867.8 ± 2.6	797.1 ± 7.3	870.5 ± 2.5	73.4 ± 5.5
		0.3/0.115	867.6 ± 2.6	793.3 ± 7.3	870.4 ± 2.4	77.1 ± 5.5

Table 3. Mean performance for `RWRL-Cartpole` averaged over 8 random seeds for each of the 12 constraint settings in the RWRL suite, where lower values of the safety coefficient and threshold indicate more challenging tasks. $\pm$ values denote one standard error.

Domain	Algorithm	Safety-Coeff/Threshold	Weighted Reward	$\mathbf{R}_{\mathbf{penalized}}$	Task Reward	Constraint Violation
Quadruped	0.1-D4PG	0.05/0.545	109.5 ± 0.0	546.6 ± 0.0	999.6 ± 0.0	453.0 ± 0.0
		0.05/0.645	305.9 ± 0.0	646.5 ± 0.0	999.5 ± 0.0	353.0 ± 0.0
		0.05/0.745	502.4 ± 0.0	746.5 ± 0.0	999.5 ± 0.0	253.0 ± 0.0
		0.1/0.545	523.7 ± 0.0	547.5 ± 0.0	999.5 ± 0.0	452.0 ± 0.0
		0.1/0.645	629.0 ± 0.0	647.5 ± 0.0	999.5 ± 0.0	352.0 ± 0.0
		0.1/0.745	734.2 ± 0.0	747.5 ± 0.0	999.5 ± 0.0	252.0 ± 0.0
		0.2/0.545	988.7 ± 56.4	942.9 ± 56.3	999.3 ± 0.0	56.4 ± 56.4
		0.2/0.645	999.2 ± 0.0	999.2 ± 0.0	999.2 ± 0.0	0.0 ± 0.0
		0.2/0.745	999.3 ± 0.0	999.3 ± 0.0	999.3 ± 0.0	0.0 ± 0.0
		0.3/0.545	999.3 ± 0.0	999.3 ± 0.0	999.3 ± 0.0	0.0 ± 0.0
		0.3/0.645	999.3 ± 0.0	999.3 ± 0.0	999.3 ± 0.0	0.0 ± 0.0
		0.3/0.745	999.2 ± 0.0	999.2 ± 0.0	999.2 ± 0.0	0.0 ± 0.0
	RC-D4PG	0.05/0.545	-157.5 ± 71.1	279.5 ± 71.1	732.5 ± 71.1	453.0 ± 0.0
		0.05/0.645	28.1 ± 60.0	368.4 ± 60.0	721.2 ± 60.0	352.8 ± 0.2
		0.05/0.745	298.6 ± 49.0	542.7 ± 49.0	795.7 ± 49.0	253.0 ± 0.0
		0.1/0.545	454.9 ± 116.0	469.8 ± 158.4	751.8 ± 81.5	282.1 ± 82.6
		0.1/0.645	948.0 ± 43.8	950.4 ± 44.7	995.7 ± 1.2	45.3 ± 43.8
		0.1/0.745	676.6 ± 56.6	686.6 ± 73.0	875.6 ± 38.7	189.0 ± 41.2
		0.2/0.545	999.1 ± 0.0	999.1 ± 0.0	999.1 ± 0.0	0.0 ± 0.0
		0.2/0.645	998.7 ± 1.4	996.5 ± 1.4	999.1 ± 0.1	2.6 ± 1.4
		0.2/0.745	997.9 ± 4.1	992.8 ± 4.1	999.1 ± 0.1	6.3 ± 4.1
		0.3/0.545	998.8 ± 2.6	996.6 ± 2.6	999.2 ± 0.0	2.6 ± 2.6
		0.3/0.645	999.2 ± 0.2	999.0 ± 0.2	999.2 ± 0.0	0.2 ± 0.2
		0.3/0.745	999.1 ± 0.0	999.1 ± 0.0	999.1 ± 0.0	0.0 ± 0.0
	MetaL	0.05/0.545	108.6 ± 0.1	545.6 ± 0.1	998.6 ± 0.1	452.9 ± 0.1
		0.05/0.645	305.3 ± 0.1	645.8 ± 0.1	998.7 ± 0.0	352.9 ± 0.1
		0.05/0.745	501.6 ± 0.0	745.7 ± 0.0	998.7 ± 0.0	253.0 ± 0.0
		0.1/0.545	523.0 ± 0.3	546.8 ± 0.3	998.4 ± 0.1	451.6 ± 0.3
		0.1/0.645	674.4 ± 44.0	690.7 ± 44.1	998.7 ± 0.1	308.0 ± 44.0
		0.1/0.745	766.7 ± 31.4	778.3 ± 31.5	998.4 ± 0.1	220.1 ± 31.4
		0.2/0.545	998.3 ± 4.6	994.2 ± 4.6	999.2 ± 0.0	4.9 ± 4.6
		0.2/0.645	999.0 ± 0.7	998.4 ± 0.7	999.1 ± 0.0	0.7 ± 0.7
		0.2/0.745	999.2 ± 0.0	999.2 ± 0.0	999.2 ± 0.0	0.0 ± 0.0
		0.3/0.545	999.2 ± 0.3	998.9 ± 0.3	999.2 ± 0.0	0.3 ± 0.3
		0.3/0.645	999.2 ± 0.2	999.0 ± 0.2	999.2 ± 0.0	0.2 ± 0.2
		0.3/0.745	998.7 ± 1.8	995.7 ± 1.8	999.2 ± 0.0	3.5 ± 1.8
	ReLOAD	0.05/0.545	651.8 ± 27.6	657.7 ± 16.6	663.8 ± 4.7	6.1 ± 25.4
		0.05/0.645	657.7 ± 3.4	659.3 ± 3.4	661.0 ± 0.9	1.7 ± 3.2
		0.05/0.745	650.0 ± 19.2	654.7 ± 18.7	659.5 ± 1.9	4.9 ± 19.1
		0.1/0.545	996.4 ± 1.0	996.4 ± 1.0	997.1 ± 0.3	0.7 ± 0.9
		0.1/0.645	993.4 ± 3.7	993.7 ± 3.5	998.4 ± 0.5	4.7 ± 3.6
		0.1/0.745	982.3 ± 7.3	983.2 ± 7.2	999.1 ± 0.1	16.0 ± 7.3
		0.2/0.545	998.0 ± 2.6	992.6 ± 2.6	999.3 ± 0.0	6.7 ± 2.6
		0.2/0.645	997.4 ± 3.3	988.9 ± 3.3	999.3 ± 0.0	10.4 ± 3.3
		0.2/0.745	996.2 ± 4.6	982.8 ± 4.6	999.3 ± 0.0	16.5 ± 4.6
		0.3/0.545	998.4 ± 2.6	992.5 ± 2.6	999.3 ± 0.0	6.8 ± 2.6
		0.3/0.645	998.0 ± 3.1	989.7 ± 3.2	999.3 ± 0.1	9.6 ± 3.1
		0.3/0.745	997.9 ± 3.7	988.8 ± 3.7	999.3 ± 0.0	10.6 ± 3.7

Table 4. Mean performance for `RWRL-Quadruped` averaged over 8 random seeds for each of the 12 constraint settings in the RWRL suite, where lower values of the safety coefficient and threshold indicate more challenging tasks. $\pm$ values denote one standard error.

Domain (Constraint)	Algorithm	Weighted Reward	Task Reward	Constraint Violation
Walker (Height)	μ-IMPALA	354.5 ± 27.7	360.6 ± 32.4	5.5 ± 4.6
	OGD-IMPALA	487.5 ± 2.8	510.1 ± 15.1	20.4 ± 8.3
	$\mu^\star$-IMPALA	356.7 ± 32.8	527.5 ± 65.0	153.6 ± 29.0
	PID-IMPALA	408.1 ± 12.7	462.6 ± 27.4	49.0 ± 19.8
	RNTR-IMPALA	386.6 ± 22.7	399.0 ± 28.2	11.1 ± 6.5
	ReLOAD-IMPALA	549.0 ± 2.0	592.2 ± 16.4	38.8 ± 10.8
Reacher (Velocity)	μ-IMPALA	436.9 ± 31.7	450.1 ± 68.4	117.3 ± 26.0
	OGD-IMPALA	22.4 ± 10.9	44.6 ± 13.1	197.7 ± 22.5
	$\mu^\star$-IMPALA	893.6 ± 8.5	913.6 ± 18.0	178.1 ± 11.6
	PID-IMPALA	428.5 ± 25.6	439.5 ± 72.0	97.6 ± 22.2
	RNTR-IMPALA	841.7 ± 102.8	850.0 ± 103.0	73.7 ± 11.6
	ReLOAD-IMPALA	938.1 ± 2.9	961.8 ± 11.7	209.7 ± 14.1

Table 5. Oscillating control suite results for IMPALA-based agents, averaged over 8 random seeds. $\pm$ values indicate one standard error.

Domain (Constraint)	Algorithm	Weighted Reward	Task Reward	Constraint Violation
Walker (Velocity)	μ-DMPO	195.9 ± 57.6	441.0 ± 57.5	150.9 ± 2.1
	OGD-DMPO	108.2 ± 10.0	357.5 ± 10.0	153.5 ± 1.1
	$\mu^\star$-DMPO	118.1 ± 15.9	932.7 ± 3.8	501.5 ± 15.4
	PID-DMPO	300.1 ± 18.6	544.2 ± 18.5	150.3 ± 1.6
	ReLOAD-DMPO	321.0 ± 49.3	569.4 ± 49.2	152.9 ± 2.0
Quadruped (Torque)	μ-DMPO	-254.8 ± 23.5	567.1 ± 22.1	558.3 ± 8.0
	OGD-DMPO	-220.1 ± 20.8	664.6 ± 20.7	601.0 ± 1.3
	$\mu^\star$-DMPO	-328.4 ± 36.1	200.4 ± 33.7	359.2 ± 13.0
	PID-DMPO	-232.2 ± 21.1	655.4 ± 20.7	602.9 ± 4.2
	ReLOAD-DMPO	-137.0 ± 55.7	768.4 ± 51.7	615.0 ± 20.6
Humanoid (Height)	μ-DMPO	92.5 ± 76.8	685.8 ± 59.4	576.7 ± 48.7
	OGD-DMPO	1.9 ± 64.3	606.1 ± 49.6	587.2 ± 40.9
	$\mu^\star$-DMPO	-45.9 ± 0.1	0.0 ± 0.0	44.7 ± 0.1
	PID-DMPO	39.9 ± 67.0	538.3 ± 50.0	484.4 ± 44.6
	ReLOAD-DMPO	114.4 ± 44.3	650.1 ± 36.3	520.6 ± 25.2

Table 6. Oscillating control suite results for DMPO-based agents, averaged over 8 random seeds. $\pm$ values indicate one standard error.

Hyperparameter	Value
regularization weight α_Ω	1e-2
value loss weight α_V	0.25
policy loss weight α_π	1.0
discount factor γ	0.99
RMSProp decay	0.99
RMSProp ϵ	1e-4
initial step size	6e-4
final step size	6e-6
max gradient norm	0.2
inner loop steps	5
initial Bregman step size	2.0
final Bregman step size	0.5
initial μ learning rate	1e-1
final μ learning rate	1e-3
network hidden units per layer	256
network depth	3
training duration (environment steps)	10e6

Table 7. Hyperparameter settings for IMPALA experiments.

Hyperparameter	Value
batch size	256
replay size	2e6
optimizer	Adam
regularization weight	1e-2
value loss weight	0.25
discount factor γ	0.99
initial step size	3e-4
final step size	1e-5
initial μ learning rate	1e-1
final μ learning rate	1e-3
network hidden units per layer	256
network depth	3
training duration	15e6

Table 8. Hyperparameter settings for DMPO experiments.