Robust Lagrangian and Adversarial Policy Gradient for Robust Constrained Markov Decision Processes

1st David M. Bossens
IHPC
Agency for Science, Technology and Research
Singapore
CFAR
Agency for Science, Technology and Research
Singapore
bossensdm@cfar.a-star.edu.sg

***Abstract*—Robustness and safety constraints are key requirements for AI systems. The robust constrained Markov decision process is a recent task-modelling framework that incorporates behavioural constraints and robustness to reinforcement learning systems. Earlier work proposed the robust constrained policy gradient (RCPG) algorithm, which robustifies either the value or the constraint and updates the worst-case distribution through constrained optimisation on a sorted value list. Highlighting potential downsides of RCPG such as not robustifying the full constrained objective and the lack of incremental learning, this paper introduces algorithms to robustify the Lagrangian and to learn incrementally using gradient descent over an adversarial policy. A theoretical analysis derives the Lagrangian policy gradient for the policy optimisation and the Lagrangian adversarial policy gradient for the adversary optimisation. Empirical experiments injecting perturbations in inventory management and safe navigation tasks demonstrate the benefit of these modifications, and combining both modifications yields the best overall performance.**

***Index Terms*—robust artificial intelligence, safe reinforcement learning, policy gradient, constrained Markov decision processes**

I. Introduction

Reinforcement learning (RL) is the standard framework for interactively learning in a complex environment. By maximising a long-term utility function, traditional RL does not take into account various behavioural constraints that would be desirable for a policy (e.g. to ensure safety or to follow legal and moral norms). Moreover, RL systems typically assume that the agent can learn directly in the true transition dynamics model. This is often not the case: for instance, in applications such as robotic control and recommendation, one may want to learn from a simulated environment rather than the true environment as this will be more safe.

Due to allowing to learn policies that satisfy long-term behavioural constraints, constrained Markov decision processes (CMDPs) [1] have become the de facto standard for safe reinforcement learning [2]. CMDPs formulate a constraint-cost function in addition to the reward function and formalise long-term behavioural constraints based on a threshold of the expected cumulative constraint-cost.

Tied to safety is the concept of robustness, which is the ability to retain performance even when the true environment changes or differs from the training environment. Robustness is represented in RL by robust Markov decision processes (RMDPs) [3], [4], which conceptualise robustness in terms of the uncertainty over the transition dynamics model of the MDP. While there are alternatives for robust RL such as domain randomisation [5] and meta-learning [6], these solutions are less theoretically sound.

The recently proposed framework of robust CMDPs (RCMDPs) [7] optimises the CMDP with the worst-case dynamics model in the uncertainty set, effectively combining RMDPs and CMDPs. To optimise RCMDP policies, the Robust Constrained Policy Gradient (RCPG) algorithm [7] combines a policy gradient algorithm with a Lagrangian relaxation for constraints and a worst-case dynamics computation for robustness. RCPG regularly recomputes the worst-case dynamics by sorting the list of values from each state and then performing constrained minimisation of the value subject to the norm constraints (i.e. the maximal distance to the expected, or "nominal", dynamics). The algorithm may not be optimal for learning robust policies since a) the algorithm does not consider the combined objective of rewards and constraint-costs; b) immediately presenting worst-case dynamics may prevent learning important and representative patterns; and c) due to repeatedly performing constrained optimisation over a sorted value list, the transition distributions of all state-action pairs can be subject to large changes whenever a state has a changed value estimate.

To mitigate these three problems, this paper proposes two algorithms. First, mitigating problem a), a variant of RCPG is introduced, called RCPG with Robust Lagrangian, which computes the worst-case over the Lagrangian, which combines the expected cumulative reward with the expected cumulative constraint-cost into a single objective. Second, to mitigate all three problems, an algorithm called Adversarial RCPG is proposed, which uses an adversary to minimise the Lagrangian of the current RL policy subject to the constraints of the

This work has been supported by the UKRI Trustworthy Autonomous Systems Hub, EP/V00784X/1, and was part of the Safety and Desirability Criteria for AI-controlled Aerial Drones on Construction Sites project.

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00219

uncertainty set (e.g. the L1 distance to the nominal model). Adversarial RCPG thereby addresses the above-mentioned limitations of RCPG by using a worst-case Lagrangian objective and by incrementally updating an adversary that represents the dynamics directly – rather than updating the dynamics indirectly and abruptly based on a sorted value list.

II. Preliminaries

The RCMDP framework is defined by a tuple $(\mathcal{S}, \mathcal{A}, r, c, d, \gamma, P_*, \mathcal{P})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is the reward function, c is the constraint-cost function, d is the budget of expected cumulative constraint-cost, γ is the discount factor, P_* is the unknown true transition dynamics model, and $\mathcal{P}$ is the uncertainty set which includes many candidate transition dynamics models. The value of executing a policy from a given state $s \in \mathcal{S}$ given a particular transition model $P \in \mathcal{P}$ is given by the expected discounted cumulative reward,

$$V_{\pi,P}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) | s_0 = s, a_t \sim \pi(s_t), s_{t+1} \sim P_{s_t,a_t}\right]. \tag{1}$$

Analogously, the expected discounted cumulative constraint-cost is denoted by

$$C_{\pi,P}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t) | s_0 = s, a_t \sim \pi(s_t), s_{t+1} \sim P_{s_t,a_t}\right]. \tag{2}$$

Denoting $P^+ := \min_{P \in \mathcal{P}} V_{\pi,P}(s)$, the objective within the RCMDP framework is given by

$$\max_{\pi} V_{\pi,P^+}(s) \quad \text{s.t.} \quad C_{\pi,P^+}(s) \leq d\,. \tag{3}$$

Note that instead of the worst-case over the value, P^+ may also be defined as the worst-case over the constraint-cost.

III. Related work

RCMDPs are a recent field of endeavour with few directly related works. Below section summarises the directly related works as well as works combining CMDPs with other notions of robustness.

A. RCMDP related works

Russel, Benosman, and Van Baar [7], [8] formulate RCMDPs as defined in Sec. II, effectively combining distributional robustness with constrained Markov decision processes. They propose the Robust Constrained Policy Gradient (RCPG), which is a policy gradient algorithm that uses L1-norm uncertainty sets and Lagrange relaxation [9]. While Lagrange relaxation is common in CMDP works (e.g. [10], [11]), the algorithm additionally provides worst-case distributional robustness over the L1-norm uncertainty set. The worst-case dynamics distribution for a given state-action pair is computed by first formulating a list of sorted values or constraint-costs and then applying linear programming or a special-purpose algorithm to re-assign probabilities subject to the norm constraints [12]. Lyapunov-based reward shaping has also been shown to yield convergence to a local optimum, although its benefits have not been demonstrated in practice [8]. Adversarial RCPG modifies the RCPG algorithm by replacing the worst-case value computation with an adversarial training scheme. In this scheme, the adversary provides transition dynamics that minimise the Lagrangian of the policy's RCMDP objective. The training is incremental, starting from the nominal model and gradually changing to less representative and more difficult CMDPs. These features help provide learning progress as well as robustness to the full constrained objective.

Other works related to RCMDPs are tailored to somewhat different purposes. Explicit Explore, Exploit, or Escape (E4) [13] provides a framework for safe exploration in RCMDPs. The approach distinguishes between known states, where the CMDP model and therefore the value function is approximately correct, and unknown states, where a worst-case assumption is taken on the transitions and the constraint-cost. The approach yields near-optimal policies for the underlying CMDP in polynomial time while satisfying the constraint-cost budget at all times. While the approach has solid theoretical support, maintaining safety throughout exploration is not always the primary concern and comes at significant training costs. The present paper is mainly interested in the final policy being safe rather than in safe exploration, and assumes an uncertainty set is available at the start of learning. The R3C objective [14] combines the worst-case value and the worst-case constraint over distinct simulators with different parametrisations being run for one step from the current state. Avoiding to explicitly compute the transition dynamics matrix makes it applicable to large scale domains such as control problems. However, the number of simulators must be limited (e.g. 4 distinct transition dynamics), thereby reducing the worst-case robustness and the scope of robustness. The present paper focuses on L1 uncertainty sets derived from state-action trajectories; such sets include a much wider range of dynamics and do not require significant prior knowledge of the environment (e.g. the internal parameters of a simulator).

B. Other approaches to robustifying CMDPs

Other approaches propose techniques other than worst-case optimisation to robustify CMDPs. Some works assume that transition dynamics are known but the reward and constraint functions are not. For instance, Zheng et al. (2020) [15] have previously used a robust version of LP in the context of UCRL, which estimates an upper confidence bound on the cost and the reward. Another approach is to use the conditional value at risk (CVaR); for instance, PG-CVaR and AC-CVaR [16] consider a CVaR of the value function (in a non-constrained approach) and later approaches use the CVaR for defining a cost critic for the CMDP [17], [18]. Other works have also explored the use of Lyapunov stability to ensure safe exploration within CMDPs [19]. Concepts of stability and safe exploration are complementary to RCMDPs, and indeed have been investigated in theory but not in practice [8], [13]. Compared to these exemplary approaches, the RCMDP

framework focuses on the uncertainty in dynamics models, making it particularly useful when transition dynamics are estimated from observational data. Other approaches define robustness guarantees based on an available baseline policy. For instance, SPIBB considers safe policy improvement across the uncertainty set in the sense of guaranteeing at least the performance of a baseline policy [20]; as shown in Satija et al. [21], this approach can be cast in a CMDP framework.

IV. Adversarial RCPG

Adversarial RCPG modifies RCPG by using a function approximator for the worst-case dynamics and by combining the values and constraints into a single objective. This is achieved by updating, by policy gradient, an adversary π_{adv} as the dynamics that minimise the Lagrangian that the policy π is maximising. Before explaining the details of Adversarial RCPG, this section first presents the original RCPG algorithm.

A. Robust-Constrained Policy Gradient

RCPG finds the saddle point of the Lagrangian for a given budget d. Denoting π_θ as the policy parametrised by θ, λ as the Lagrangian multiplier, and P^+ as the worst-case transition dynamics, the objective is given by

$$\min_{\lambda \geq 0} \max_{\pi_\theta} L(\lambda, \pi_\theta; P^+) = V_{\pi_\theta, P^+}(s) - \lambda \left(C_{\pi_\theta, P^+}(s) - d \right) . \tag{4}$$

To optimise the above objective, sampling of limited-step trajectories is repeated for a large number of independent iterations starting from a random initial state $s \sim P_0$. Based on the large number of trajectories collected, one then performs gradient descent in λ and gradient ascent in θ.

a) Estimating the worst-case distribution: RCPG is based on L1 uncertainty sets of the type $\mathcal{P}_{s,a} = \{P \in \Delta^S : ||P - \hat{P}_{s,a}||_1 \leq \alpha\}$. Computing the worst-case distribution, also known as the "inner problem", is equivalent to the constrained optimisation problem

$$P^+ = \arg\min_P V^+ = P^\intercal \hat{V} \quad \text{s.t.} \quad ||P - \hat{P}_{s,a}||_1 \leq \alpha \quad \mathbf{1}^\intercal P = 1 . \tag{5}$$

The solution to the inner problem, P^+, is the distribution that minimises the value subject to the norm constraints. RCPG solves the innner problem based on linear programming or related constrained optimisation algorithms (e.g. Petrik et al. [12] present a special purpose algorithm that solves the problem with $O(S \log S)$ time complexity). These algorithms require a tabular approximation or (preferably) a critic network of the quantity to robustify (the expected cumulative reward or constraint-cost).

b) Learning problems of RCPG: The RCPG algorithm has several features in its learning that could be improved. First, the RCPG objective provides robustness to either the worst-case value or the worst-case constraint-cost but not the desired Lagrangian objective combining both. Second, RCPG training results in only training on the worst case, which may be too challenging at the start of training; a more gradual training with an incrementally improving adversary could be beneficial. Third, if the critic estimates the worst-case state erroneously then the updated worst-case transition dynamics will sample this state at an excessively high rate as the next state for all the state-action pairs. This results in many abrupt changes in the distributions, again making an incremental learning process difficult.

B. RCPG with Robust Lagrangian

A relatively straightforward way to solve the first learning problem of RCPG is to directly robustify the Lagrangian Eq. 4. RCPG with Robust Lagrangian replaces Eq. 5 with

$$P^+ = \arg\min_P L^+ = P^\intercal \left(\hat{V} - \lambda \hat{C} \right) \quad \text{s.t.} \quad ||P - \hat{P}_{s,a}||_1 \leq \alpha \quad \mathbf{1}^\intercal P = 1 . \tag{6}$$

As shown in Theorem 1, the same RCPG algorithm can be used for optimising Eq. 4 when the worst-case distribution P^+ is defined based on Eq. 6.

C. Adversarial RCPG

While RCPG with Robust Lagrangian provides a suitable objective, it does not address the other learning problems of RCPG. The Adversarial RCPG algorithm is proposed to mitigate all three learning problems. It learns an adversarial policy $\pi_{\text{adv}} : \mathcal{S} \times \mathcal{A} \to \Delta^S$ which approximates the robust Lagrangian dynamics model P^+ from Eq. 4 to robustify the full constrained objective. The adversarial policy is learned by gradient descent together with the policy: the adversary starts from the nominal distribution $\hat{P}$ and incrementally updates the dynamics to be more challenging using gradient descent. The resulting algorithm is shown in Algorithm 1).

As in Lagrangian RCPG, Adversarial RCPG optimises the Lagrangian in Eq. 4 robustly based on the worst-case distribution defined in Eq. 6. However, Adversarial RCPG replaces P^+ with an adversarial policy π_{adv} which minimises the policy's objective subject to the norm constraints:

$$\pi_{\text{adv}} = \arg\min_{\pi_{\text{adv}}} L(\lambda, \pi_\theta; \pi_{\text{adv}}) \quad \text{s.t.} \quad ||\pi_{\text{adv}}(s,a) - \hat{P}_{s,a}||_1 \leq \alpha(s,a) \forall (s,a) \in \mathcal{S} \times \mathcal{A} . \tag{7}$$

This leads to a constrained optimisation problem where π_{adv} is the solution to a different Lagrangian $L_{\text{adv}}(\lambda_{\text{adv}}, \pi_{\text{adv}})$, namely

$$\max_{\lambda_{\text{adv}} \geq 0} \min_{\pi_{\text{adv}}} L_{\text{adv}}(\lambda_{\text{adv}}, \pi_{\text{adv}}) = L(\lambda, \pi_\theta; \pi_{\text{adv}}) + \sum_{s,a} \lambda_{\text{adv}} \left(||\pi_{\text{adv}}(s,a) - \hat{P}_{s,a}||_1 - \alpha(s,a) \right) , \tag{8}$$

where the multiplier is the same for all state-action pairs to make the optimisation scalable to large state-action spaces.

The distribution of states depends crucially on the adversary, and this distribution should be within an L1-norm of $\alpha(s,a)$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$. Minimising the L1 norm based on samples obtained from π_{adv} itself may lead to a system hack, in the sense that the adversarial policy may learn to sample

states which have minimal L1 norm rather than minimising the L1 norm independent of the state-action pairs observed. Therefore, to compute the gradient for the L1 norm (l.20) for the adversarial policy update, a new random batch with random state-action samples is used for each gradient such that overall the norm is being reduced regardless of the state-action pairs encountered (see Algorithm 2). By contrast, the term ΔP (l.22) is used for updating the multiplier only, so there is no risk for such system hacks; therefore, it is based directly based on the samples from the adversary, which gives more impact of frequently observed state-action pairs on the multiplier. This approach empirically satisfies the norm constraints.

Algorithm 1 Offline optimisation with Adversarial RCPG

1: **procedure** OFFLINE-OPTIMISATION
2: $B = \mathcal{S} \times \mathcal{A}$
3: $\theta_{\text{adv}} \leftarrow \arg\min_{\theta_{\text{adv}}} \text{MAE}\left(\pi_{\text{adv}}(B) - \hat{P}(B)\right)$
4: **for** $i = 1, 2, \ldots, I$ **do** ▷ Independent iterations
5: Random initial state $s \sim P_0$.
6: **for** $t = 0, 1, \ldots, T-1$ **do** ▷ Simulate trajectory
7: **if** s is terminal **then**
8: **break**
9: Transition $(s, a \sim \pi(a|s), r(s,a), s' \sim \pi_{\text{adv}}(s'|s,a))$.
10: Policy gradient $\nabla_t \leftarrow \nabla_\theta \log(\pi_\theta(a|s))$.
11: Adversary gradient $\nabla_t^{\text{adv}} \leftarrow \nabla_{\theta_{\text{adv}}} \log(\pi_{\text{adv}})$.
12: $s \leftarrow s'$.
13: $T_{\text{stop}} \leftarrow t$.
14: **for** $t = T_{\text{stop}} - 1, T_{\text{stop}} - 2, \ldots, 0$ **do**
15: ▷ Policy update
16: $\mathbf{V}_t \leftarrow V_t - \lambda C_t$.
17: $\theta \leftarrow \theta + \eta_1(k) * \mathbf{V}_t \nabla_t$
18: $\lambda \leftarrow \lambda + \eta_2(k) * (C - d)$
19: ▷ Adversary update
20: $\nabla P \leftarrow$ `compute-nominal-deviation-grad()`
21: $\theta_{\text{adv}} \leftarrow \theta_{\text{adv}} - \eta_1(k) * \left(\mathbf{V}_{\text{next}} \nabla_t^{\text{adv}} + \lambda_{\text{adv}} \nabla P\right)$
22: $\Delta P \leftarrow ||\pi_{\text{adv}}(s_t, a_t) - \hat{P}_{s_t, a_t}||_1$.
23: $\lambda_{\text{adv}} \leftarrow \lambda_{\text{adv}} + \eta_2(k) * (\Delta P - \alpha(s_t, a_t))$
24: **return** π_θ

Algorithm 2 Computing gradient for deviation from nominal.

1: **procedure** COMPUTE-NOMINAL-DEVIATION-GRAD(parameters θ_{adv})
2: ▷ Set batch with small number of samples (N_{samp})
3: $B \leftarrow$ `random`$(\mathcal{S} \times \mathcal{A}, N_{\text{samp}})$
4: **for** $i \in 1, \ldots, B$ **do**
5: $\alpha[i] \leftarrow \alpha(s[i], a[i])$.
6: nom$[i] \leftarrow \hat{P}(s[i], a[i])$
7: $y[i] \leftarrow \pi_{\text{adv}}(: |s[i], a[i])$.
8: dev$[i] = \max(0, ||y[i] - \text{nom}[i]||_1 - \alpha[i])$
9: **return** $\nabla_{\theta_{\text{adv}}}$`mean`(dev)

D. Uncertainty Set and Uncertainty Budget

Uncertainty sets can be constructed in many ways. Methods of choice include sets based on Hoeffding's inequality and Bayesian methods [22]. The experiments select Hoeffding L1 uncertainty sets which have state-action dependent transitions according to

$$\mathcal{P}_{s,a} = \{P : ||P - \hat{P}_{s,a}|| < \alpha(s,a)\}, \tag{9}$$

where $\hat{P}$ is the nominal model, and $\alpha(s,a)$ is the uncertainty budget for state-action pair (s,a). The uncertainty budget is set according to $\alpha(s,a) = \sqrt{\frac{2}{n(s,a)} \ln\left(\frac{2^S SA}{\delta}\right)}$, where $1-\delta$ is the confidence and $n(s,a)$ is the number of visitations of (s,a). If P^* is the true transition dynamics model, then the Hoeffding set ensures that $P^*_{s,a} \in \mathcal{P}_{s,a}$ with probability at least $1 - \frac{\delta}{SA}$ and by union bound, that $P^* \in \mathcal{P}$ with probability at least $1 - \delta$ [22].

V. Lagrangian Policy Gradient Theorems

To prove that the desired objectives are indeed being optimised by Adversarial RCPG, two Lagrangian policy gradient theorems are derived. The first theorem shows that the Lagrangian of the policy can indeed be maximised using a simple policy gradient. The second theorem shows that the chosen adversary indeed follows the gradient steps to *minimise* the Lagrangian of the policy.

A. Deriving the Robust Lagrangian Policy Gradient

The key realisation for optimising the Lagrangian is that in CMDPs, both the value and the constraint-cost are expected cumulative quantities with the same discounting factor. This allows reusing existing results by reformulating the Lagrangian in terms of rewards and constraint-costs.

Theorem 1. ***Lagrangian policy gradient theorem.*** *Let $\pi : \mathcal{S} \to \Delta^A$ be a stochastic policy, let P be the transition dynamics, let s_0 be the starting state, and for any state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$ define $\mathbf{Q}_\pi(s,a) = Q_\pi(s,a) - \lambda C_\pi(s,a)$ and $\mathbf{V}_\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[\mathbf{Q}_\pi(s,a)]$. Then it follows that*

$$\nabla_\theta \mathbf{V}_\pi(s_0) \propto \mathbb{E}_{\pi,P}\left[\mathbf{Q}_\pi(s_t, a_t) \nabla_\theta \log(\pi(a_t|s_t))\right]. \tag{10}$$

Proof: The proof (see Appendix A) reformulates the CMDP as a Lagrangian MDP [23] and then makes analogous steps to the proof of the policy gradient theorem [24].

As a consequence of this theorem, updating with steps according to $\mathbf{Q}_\pi(s_t,a_t)\nabla_\theta \log(\pi(a_t|s_t))$ will follow the gradient of the Lagrangian $L = V_\pi(s_0) - \lambda(C_\pi(s_0) - d)$ since the term λd is a constant. Since P is arbitrarily chosen, this also holds for P^+ as defined in Eq. 6.

B. Deriving the Lagrangian Adversarial Policy Gradient

Theorem 2. ***Lagrangian adversarial policy gradient theorem.*** *Let $\pi_{adv} : \mathcal{S} \times \mathcal{A} \to \Delta^S$ be the adversary replacing the transition dynamics of the CMDP, let s_0 be the starting state, let T be the horizon of the decision process, and for any state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$ define $\mathbf{Q}_\pi(s,a) = Q_\pi(s,a) - \lambda C_\pi(s,a)$ and $\mathbf{V}_\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)}[\mathbf{Q}_\pi(s,a)]$. Then it follows that*

$$\nabla_{\theta_{adv}} \mathbf{V}_\pi(s_0) = \sum_{k=0}^{T-1} \mathbb{E}\left[\mathbf{V}_\pi(s_{k+1}) \nabla_{\theta_{adv}} \log(\pi_{adv}(s_{k+1}|s_k, a_k))\right]. \tag{11}$$

Proof: The proof uses a formalism similar to the previous proof but this time expands the gradient with respect to θ_{adv}. The full proof is given in Appendix B.

This theorem implies that applying consecutive updates of $\mathbf{V}_\pi(s_{t+1})\nabla_{\theta_{\text{adv}}} \log\left(\pi_{\text{adv}}(s_{t+1}|s_t, a_t)\right)$ for $t = 0, \ldots, T-1$ will move π_{adv} along the gradient of the objective.

VI. Results

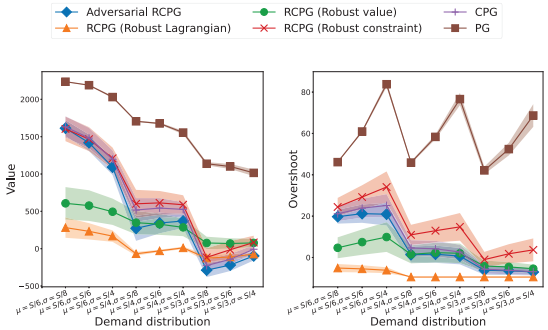
Having introduced Adversarial RCPG and RCPG, the experimental validation below compares their performance on the cumulative reward and constraint-cost in perturbed environments. The experiments are set up in three consecutive phases. In the **model estimation phase**, a random uniform policy is run on a dynamics model P_{data}, which represents a centroid of the test dynamics models. The result of phase 1 is a nominal model $\hat{P}$ and (if applicable) the Hoeffding L1 uncertainty set $\mathcal{P}$. In the **policy training phase**, policies are trained across 5,000 episodes based on either P_{data} (non-robust algorithms) or $\mathcal{P}$. In the **policy test phase**, the trained policy is tested by taking greedy actions on a set of test dynamics that are perturbations of P_{data}. To evaluate the training and test performance, the value and constraint-cost are evaluated without discounts, and the resulting budget d is corrected accordingly by a factor $T/(\sum_{i=0}^{T-1} \gamma^i)$, where T is the maximal episode length.

The algorithms evaluated are the following: 1) the **Adversarial RCPG** algorithm implementing the Lagrangian adversary as described in Algorithm 1 and supported by Theorem 1–2; 2) **RCPG (Robust Lagrangian)**, the variant of RCPG formulated in Section IV-B to formulate the worst-case dynamics as the model that minimises the Lagrangian, as supported by Theorem 1, which can be seen as an ablation that removes the adversary but keeps the Lagrangian objective; 3) **RCPG (Robust value)** [7], which formulates robustness in terms of the dynamics with worst-case value; 4) **RCPG (Robust constraint)**, which formulates robustness in terms of the dynamics with worst-case constraint-cost [7], [13]; 5) **CPG**, which uses the nominal transition dynamics instead of the worst-case transition dynamics, as an ablation without robustness; and 6) **PG**, a further ablation condition with no constraints, which corresponds to REINFORCE [24].

To demonstrate a range of applications, experiments include an inventory management domain and two safe navigation tasks, each with a variety of test cases. As a quick overview of the test results, Tab. I shows that the Adversarial RCPG is always among the top two performing algorithms on the penalised return, a performance metric for CMDPs. The reader may also refer to Appendix C, D, and E of the supplementary information for additional details and figures of the experiments. The source code used for the experiments is available at https://github.com/bossdm/RCMDP.

A. Inventory Management

The first domain is the inventory management problem [25], which has been the test bed of the RCPG algorithm [7]. The task of the agent is to purchase items to make optimal profits selling the items, balancing supply with demand in the process. The state is the current inventory while the action is the purchased number of items from the supplier. States are integers in $\{0, 1, \ldots, S-1\}$, where S is the number of states. Initially the inventory is empty, corresponding to initial state $s_0 = 0$, and a full inventory contains $S-1$ items. The constraint is that the number of purchased items, $a \in \mathcal{A}$, should not exceed the purchasing limit. Oscillating behaviours shown by RCPG algorithms (see Appendix E) are attributed to large abrupt changes in the estimated worst-case distribution. The policy test phase consists of 9 different parameters μ and σ for the demand distribution, resulting in changed transition dynamics. The penalised return scores in Tab. I demonstrates that RCPG (Robust Lagrangian) has the highest penalised return. This indicates that robustifying the Lagrangian is beneficial but also that the abrupt changes in the estimated worst-case distribution do not appear to hamper RCPG's test performance; because any state is reachable from any other state and the demand is *iid*, the algorithm is less sensitive to excessive sampling of a single state and shifting transition dynamics. Fig. 1 demonstrates the value, which is proportional to the profit, and the overshoot across the different demand distributions.



(a) Test value (b) Test overshoot

Figure 1. Test performance metrics of the algorithms on the test set of perturbed transition dynamics in Inventory Management. For each of 20 training runs, each parameter setting is run 50 times and the plot displays the mean and standard error over runs. The parameter manipulated is the mean, μ, and standard deviation, σ, of the demand distribution.

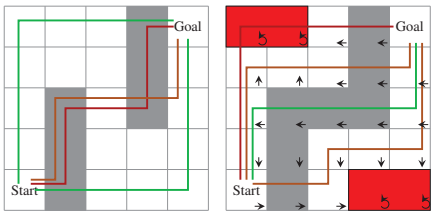
B. Safe Navigation

The second domain and third domain are safe navigation tasks in a 5-by-5 square grid world, formulated specifically to highlight the advantages of agents that satisfy constraints robustly (see Fig. 2).

a) Safe Navigation 1: In Safe Navigation 1 (see Fig. 2a), the grid contains 6 grey cells that incur constraint-cost of 1.0 and agents should satisfy a budget of $d = 3.0$. Agents stay in the grid world for $T = 200$ time steps if the goal is not found. Test 1A manipulates P_{success}, the probability with which the agent successfully moves to the intended location. Test 1B fixes $P_{\text{success}} = 0.80$ while manipulating N_ϵ, the number of state-action pairs perturbed by setting $s' = s + \epsilon(s, a)$ upon successful action. As shown in Tab. I, Adversarial RCPG outperforms all other algorithms in both tests of Safe Navigation 1. In line with the hypothesis that

Table I
COMPARISON OF ALGORITHMS ON ALL TESTS. TO PROVIDE A SINGLE STATISTIC, THE PENALISED RETURN [14] IS DEFINED AS $R_{\text{PEN}} = V(s_0) - \lambda \max(0, (C(s_0) - d))$. THE EVALUATION WEIGHT IS SET AS $\lambda = 500$, WHICH IS EQUAL TO THE MAXIMAL LAGRANGIAN MULTIPLIER DURING THE CONSTRAINED OPTIMISATION. BOLD HIGHLIGHTS THE TOP TWO SCORES WHILE UNDERLINE INDICATES THE HIGHEST SCORE.

	Adversarial RCPG	RCPG (Robust Lagrangian)	RCPG (Robust value)	RCPG (Robust constraint)	CPG	PG
Inventory Management	$\mathbf{-3058.6 \pm 1341.6}$	$\underline{\mathbf{31.8 \pm 26.3}}$	-3843.1 ± 1567.9	-7812.0 ± 2148.4	-3977.0 ± 1243.6	-28092.3 ± 913.1
Safe Navigation 1A	$\underline{\mathbf{-76.7 \pm 20.2}}$	-190.5 ± 9.2	-4825.8 ± 4537.9	-200.0 ± 0.0	$\mathbf{-133.6 \pm 20.2}$	-9443.7 ± 6307.1
Safe Navigation 1B	$\underline{\mathbf{-71.9 \pm 18.9}}$	-273.9 ± 81.7	-4751.3 ± 3965.5	-735.3 ± 312.2	$\mathbf{-123.5 \pm 18.7}$	-8686.0 ± 5718.3
Safe Navigation 2A	$\mathbf{-48.1 \pm 9.7}$	-316.8 ± 282.0	-275.7 ± 224.0	$\underline{\mathbf{-30.6 \pm 8.1}}$	-259.1 ± 218.3	-512.4 ± 299.0
Safe Navigation 2B	$\mathbf{-1437.2 \pm 107.4}$	-1451.0 ± 221.9	-1825.0 ± 391.8	$\underline{\mathbf{-1259.3 \pm 101.2}}$	-1681.6 ± 421.2	-2395.5 ± 546.5

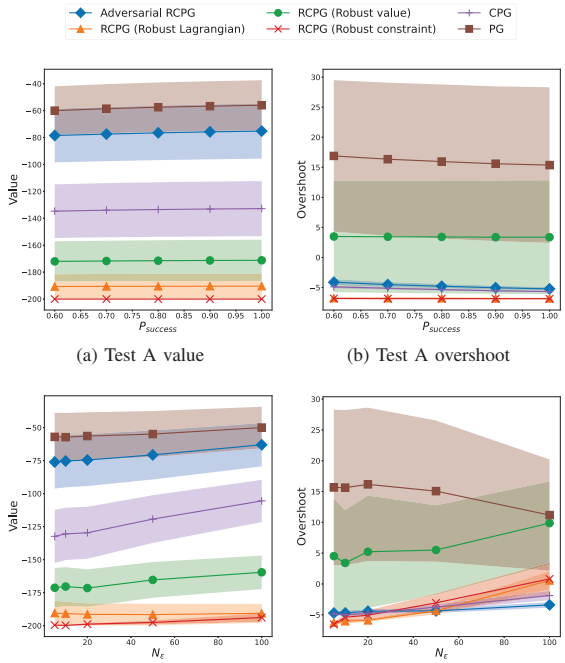


(a) Safe Navigation 1 (b) Safe Navigation 2

Figure 2. Illustration of the safe navigation tasks. In Safe Navigation 1, the constraint is to hit no more than 4 grey cells on average. In Safe Navigation 2, the constraint is to avoid the red cells and only a limited number of grey cells. Unconstrained solutions, constrained, and robust-constrained trajectories are demonstrated in red, orange, and green, respectively. The arrows represent the worst-case transitions for test Safe Navigation 2B.

Adversarial RCPG provides incremental learning, the training value and constraint-overshoot develop much more smoothly in Adversarial RCPG when compared to RCPG variants, which display oscillating and high-variance scores on these metrics (see Appendix E). In the test, the RCPG algorithms do not find paths to the goal location although the RCPG (Robust constraint) and RCPG (Robust Lagrangian) satisfy the constraint. As shown in Fig. 3, Adversarial RCPG combines a high value comparable to PG with a negative overshoot that is not affected even by severe perturbations. Safe Navigation 1 presents a particular challenge for RCPG; this is attributed to learning paths that satisfy the constraint (by avoiding grey cells) but that do not come closer to the goal.

b) Safe Navigation 2: In Safe Navigation 2 (see Fig. 2b), the grid contains 7 grey cells that incur constraint-cost of 0.1, 4 red cells that incur a cost of 1.0, and agents should satisfy a budget of $d = 0.4$. Agents stay in the grid world for $T = 100$ time steps if the goal is not found. Test 2A manipulates P_{success}. Test 2B keeps $P_{\text{success}} = 0.50$ and upon failure, the agent is moved according to worst-case transitions as shown in the arrows of Fig. 2b. As shown in Tab. I, RCPG (Robust constraint) is the top performer followed by Adversarial RCPG in both tests of Safe Navigation 2. The tighter uncertainty set and shorter episode leads to a similarly smooth training for all RCPG variants when compared to Adversarial RCPG (see Appendix E) as it makes starting from the worst-case dynamics less challenging. Avoiding constraint-cost becomes comparably more challenging than eventually



(a) Test A value (b) Test A overshoot

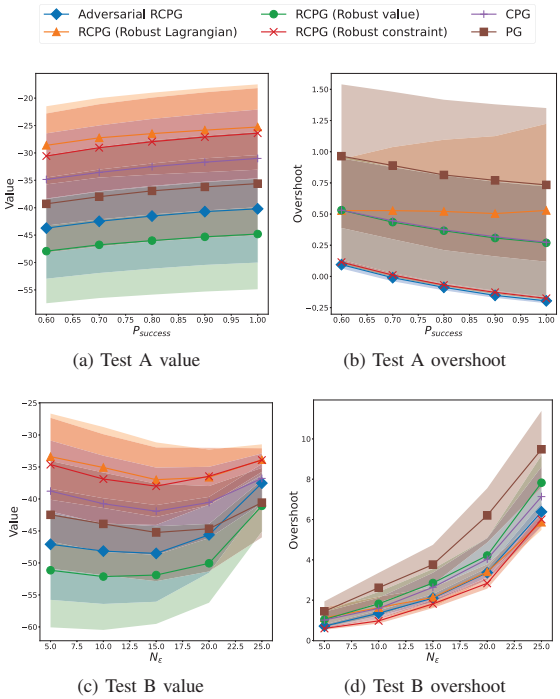
(c) Test B value (d) Test B overshoot

Figure 3. Test performance metrics of the algorithms on the test set of perturbed transition dynamics in Safe Navigation 1. For each of 20 training runs, each parameter setting is run 50 times and the plot displays the mean and standard error over runs. **Test A:** The parameter manipulated is the move probability of the actions. **Test B:** The parameter manipulated is the number of perturbations, i.e. randomly selected state-action pairs that are perturbed with a random offset in $\mathcal{N}(s)$.

finding the goal; therefore it becomes more important to robustify the constraint-cost compared to the value (see highest rank of RCPG with Robust constraint). As shown in Fig. 4, Adversarial RCPG does not have the highest value but achieves a low overshoot comparable to RCPG (Robust constraint); RCPG (Robust Lagrangian) also performs comparably on the overshoot on test B.

VII. CONCLUSION AND FUTURE WORK

Providing robustness as well as constraints into policies is of critical importance for safe reinforcement learning. This paper proposes a robust Lagrangian objective and an adversarial



(a) Test A value (b) Test A overshoot

(c) Test B value (d) Test B overshoot

Figure 4. Test performance metrics of the algorithms on the test set of perturbed transition dynamics in Safe Navigation 2. For each of 20 training runs, each parameter setting is run 50 times and the plot displays the mean and standard error over runs. **Test A:** The parameter manipulated is the move probability of the actions. **Test B:** The parameter manipulated is the number of perturbations, i.e. randomly selected states that are perturbed with a worst-case transition according to the arrows in Fig. 2b.

gradient descent algorithm for a modified robust constrained policy gradient algorithm with stable and incremental learning properties. These modifications are empirically demonstrated to improve reward-based and constraint-based metrics on a wide range of test perturbations. Adversarial policies have been of interest in designing realistic attacks on reinforcement learning policies (e.g. [26], [27]); while these works do not consider uncertainty sets, an interesting avenue for future research is to extend Adversarial RCPG with uncertainty sets that satisfy realism constraints in addition to the current norm constraints.

References

[1] E. Altman, *Constrained Markov decision processes*. Chapman and Hall/CRC, 1998.

[2] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll, "A Review of Safe Reinforcement Learning: Methods, Theory and Applications," *arXiv preprint*, pp. 1–89, 2023.

[3] G. N. Iyengar, "Robust dynamic programming," *Mathematics of Operations Research*, vol. 30, no. 2, pp. 257–280, 2005.

[4] A. Nilim and L. E. Ghaoui, "Robust control of Markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[5] J. Van Baar, A. Sullivan, R. Cordorel, D. Jha, D. Romeres, and D. Nikovski, "Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2019)*, pp. 6001–6007, 2019.

[6] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *Proceedings of the International Conference on Machine Learning (ICML 2017)*, (Sydney, Australia), pp. 1–10, 2017.

[7] R. H. Russel, M. Benosman, and J. Van Baar, "Robust Constrained-MDPs: Soft-Constrained Robust Policy Optimization under Model Uncertainty," in *Advances in Neural Information Processing Systems workshop (NeurIPS 2021)*, 2021.

[8] R. H. Russel, M. Benosman, J. van Baar, and R. Corcodel, "Lyapunov Robust Constrained-MDPs for Sim2Real Transfer Learning," in *Federated and Transfer Learning*, vol. 27, pp. 307–328, 2022.

[9] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 2003.

[10] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the International Conference on Machine Learning (ICML 2017)*, vol. 1, pp. 30–47, 2017.

[11] A. Ray, J. Achiam, and D. Amodei, "Benchmarking Safe Exploration in Deep Reinforcement Learning," *arXiv preprint*, pp. 1–6, 2019.

[12] M. Petrik, "RAAM : The Benefits of Robustness in Approximating Aggregated MDPs in Reinforcement Learning," in *Advances in Neural Information Processing Systems (NeurIPS 2005)*, pp. 1–9, 2005.

[13] D. M. Bossens and N. Bishop, "Explicit Explore, Exploit, or Escape (E^4): near-optimal safety-constrained reinforcement learning in polynomial time," *Machine Learning*, vol. 112, pp. 817–858, 2023.

[14] D. J. Mankowitz, D. A. Calian, R. Jeong, C. Paduraru, N. Heess, S. Dathathri, M. Riedmiller, and T. Mann, "Robust Constrained Reinforcement Learning for Continuous Control with Model Misspecification," *arXiv preprint*, pp. 1–23, 2020.

[15] L. Zheng and L. J. Ratliff, "Constrained Upper Confidence Reinforcement Learning with Known Dynamics," in *Proceedings of the Annual Conference on Learning for Dynamics and Control (L4DC 2020)*, pp. 1–10, 2020.

[16] Y. Chow and M. Ghavamzadeh, "Algorithms for CVaR optimization in MDPs," in *Advances in Neural Information Processing Systems (NeurIPS 2014)*, pp. 3509–3517, 2014.

[17] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. J. Spaan, "WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning," in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI 2021)*, vol. 35, pp. 10639–10646, 2021.

[18] R. Zhang and J. Sjölund, "Risk-sensitive Actor-free Policy via Convex Optimization," in *AISafety and SafeRL Joint Workshop at the International Joint Conference on Artificial Intelligence (IJCAI 2023)*, 2023.

[19] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, "Lyapunov-based Safe Policy Optimization for Continuous Control," in *Proceedings of the Reinforcement Learning for Real Life Workshop in the International Conference on Machine Learning (ICML 2019)*, 2019.

[20] R. Laroche, P. Trichelair, and R. T. Des Combes, "Safe policy improvement with baseline bootstrapping," in *Proceedings of the International Conference on Machine Learning (ICML 2019)*, pp. 6487–6520, 2019.

[21] H. Satija, J. Pineau, P. S. Thomas, and R. Laroche, "Multi-Objective SPIBB: Seldonian Offline Policy Improvement with Safety Constraints in Finite MDPs," in *Advances in Neural Information Processing Systems (NeurIPS 2021)*, pp. 2004–2017, 2021.

[22] R. H. Russel and M. Petrik, "Beyond confidence regions: Tight Bayesian ambiguity sets for robust MDPs," in *Advances in Neural Information Processing Systems (NeurIPS 2019)*, vol. 32, 2019.

[23] M. A. Taleghan and T. G. Dietterich, "Efficient exploration for constrained MDPs," in *AAAI Spring Symposium – Technical Report*, pp. 313–319, 2018.

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, second edition ed., 2017.

[25] Puterman, Martin L, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley New York, 2005.

[26] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial Policies: Attacking Deep Reinforcement Learning," in *Proceedings of the International Conference on Learning Representations (ICLR 2020)*, pp. 1–16, 2020.

[27] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially Robust Policy Learning: Active construction of physically-plausible perturbations," in *IEEE International Conference on Intelligent Robots and Systems (IROS 2017)*, pp. 3932–3939, 2017.

Supplementary Information for *Robust Lagrangian and Adversarial Policy Gradient for Robust Constrained Markov Decision Processes*

David M. Bossens

Appendix A: Proof of Robust Constrained Policy Gradient theorem

Using the notation $\mathbf{r}(s,a) = r(s,a) - \lambda c(s,a)$ to formulate the problem as an MDP, we have

$$\begin{aligned}
\nabla_\theta \mathbf{V}_\pi(s) &= \nabla_\theta \left(\sum_{a\in\mathcal{A}} \pi(a|s) \mathbf{Q}_\pi(s,a) \right) && \text{(definition)} \\
&= \sum_{a\in\mathcal{A}} \mathbf{Q}_\pi(s,a) \nabla_\theta \pi(a|s) + \pi(a|s) \nabla_\theta \mathbf{Q}_\pi(s,a) && \text{(product rule)} \\
&= \sum_{a\in\mathcal{A}} \mathbf{Q}_\pi(s,a) \nabla_\theta \pi(a|s) + \pi(a|s) \nabla_\theta \sum_{s',\mathbf{r}} \mathbb{P}(s',\mathbf{r}|a,s) \left(\mathbf{r}(s,a) + \mathbf{V}_\pi(s') \right) && \text{(bootstrap from next Q)} \\
&= \sum_{a\in\mathcal{A}} \mathbf{Q}_\pi(s,a) \nabla_\theta \pi(a|s) + \pi(a|s) \nabla_\theta \sum_{s'} P(s'|a,s) \nabla_\theta \mathbf{V}_\pi(s') \\
&\quad \text{(noting that } (P(s'|a,s) = \sum_r \mathbb{P}(s',\mathbf{r}|a,s)) \\
&= \sum_{a\in\mathcal{A}} \mathbf{Q}_\pi(s,a) \nabla_\theta \pi(a|s) + \pi(a|s) \sum_{s'} P(s'|s,a) \\
&\quad \left(\sum_{a'\in\mathcal{A}} \nabla_\theta \pi(a'|s') \mathbf{Q}_\pi(s',a') + \pi(a'|s') \sum_{s''} P(s''|s',a') \nabla_\theta \mathbf{V}_\pi(s'') \right) && \text{(unpacking analogously)} \\
&= \sum_{s_{\text{next}}\in\mathcal{S}} \sum_{k=0}^{\infty} \mathbb{P}(s \to s_{\text{next}}|k,\pi) \sum_a \mathbf{Q}_\pi(s_{\text{next}},a) \nabla_\theta \pi(a|s_{\text{next}}) \,. && \text{(repeated unpacking)}
\end{aligned}$$

To demonstrate the objective is satisfied from $t=0$ to $t=\infty$, the proof continues from the initial state s_0. There it is useful to consider the average number of visitations of s in an episode, $n(s) := \sum_{k=0}^{\infty} \mathbb{P}(s_0 \to s|k,\pi)$, and its relation to the on-policy distribution $\mu(s|\pi)$, the fraction of time spent in each state when taking actions from π:

$$\begin{aligned}
\nabla_\theta \mathbf{V}_\pi(s_0) &= \sum_{s\in\mathcal{S}} n(s) \sum_a \mathbf{Q}_\pi(s,a) \nabla_\theta \pi(a|s) \\
&\propto \sum_{s\in\mathcal{S}} \mu(s|\pi) \sum_a \mathbf{Q}_\pi(s,a) \nabla_\theta \pi(a|s) \\
&= \mathbb{E}_{\pi,P} \left[\sum_a \nabla_\theta \mathbf{Q}_\pi(s_t,a) \pi(a|s_t) \right] \\
&= \mathbb{E}_{\pi,P} \left[\sum_a \mathbf{Q}_\pi(s_t,a) \pi(a|s_t) \frac{\nabla_\theta \pi(a|s_t)}{\pi(a|s_t)} \right] \\
&= \mathbb{E}_{\pi,P} \left[\frac{\mathbf{Q}_\pi(s_t,a_t) \nabla_\theta \pi(a_t|s_t)}{\pi(a_t|s_t)} \right] \\
&= \mathbb{E}_{\pi,P} \left[\mathbf{Q}_\pi(s_t,a_t) \nabla_\theta \log\left(\pi(a_t|s_t)\right) \right] \,.
\end{aligned}$$

□

APPENDIX B: PROOF OF ROBUST CONSTRAINED ADVERSARIAL POLICY GRADIENT THEOREM

First note that the gradient of $\mathbf{V}_\pi(s_t)$ of a state s_t at time t is given by

$$
\begin{aligned}
\nabla_{\theta_{\text{adv}}}\mathbf{V}_\pi(s_t) &= \nabla_{\theta_{\text{adv}}}\left(\sum_a \pi(a|s_t)\mathbf{Q}_\pi(s_t,a)\right) && \text{(definition)}\\
&= \sum_a \pi(a|s_t)\nabla_{\theta_{\text{adv}}}\mathbf{Q}_\pi(s_t,a) && (\pi \text{ independent of } \pi_{\text{adv}})\\
&= \sum_a \pi(a|s_t)\nabla_{\theta_{\text{adv}}}\left(\mathbb{P}(s',\mathbf{r}|s_t,a)\left(\sum_{\mathbf{r},s'}\mathbf{r}(s_t,a)+\mathbf{V}_\pi(s')\right)\right) && \text{(expand the Q-value)}\\
&= \sum_a \pi(a|s_t)\nabla_{\theta_{\text{adv}}}\left(\sum_{s'}\mathbb{P}(s'|s_t,a)\mathbf{V}_\pi(s')\right) && \text{(reward distribution independent of } \pi_{\text{adv}})\\
&= \sum_a \pi(a|s_t)\nabla_{\theta_{\text{adv}}}\left(\sum_{s'}\pi_{\text{adv}}(s'|s_t,a)\mathbf{V}_\pi(s')\right) && \text{(use } \pi_{\text{adv}} \text{ to generate the next state)}\\
&= \sum_a \pi(a|s_t)\left(\sum_{s'}\mathbf{V}_\pi(s')\nabla_{\theta_{\text{adv}}}\pi_{\text{adv}}(s'|s_t,a)+\pi_{\text{adv}}(s'|s_t,a)\nabla_{\theta_{\text{adv}}}\mathbf{V}_\pi(s')\right) && \text{(product rule)}\\
&= \sum_a \pi(a|s_t)\left(\sum_{s'}\mathbf{V}_\pi(s')\pi_{\text{adv}}(s'|s_t,a)\frac{\nabla_{\theta_{\text{adv}}}\pi_{\text{adv}}(s'|s_t,a)}{\pi_{\text{adv}}(s'|s_t,a)}+\nabla_{\theta_{\text{adv}}}\mathbf{V}_\pi(s')\right)\\
&\quad \text{(divide and multiply by } \pi_{\text{adv}})\\
&= \mathbb{E}_{\pi,\pi_{\text{adv}}}\left[\mathbf{V}_\pi(s_{t+1})\frac{\nabla_{\theta_{\text{adv}}}\pi_{\text{adv}}(s_{t+1}|s_t,a_t)}{\pi_{\text{adv}}(s_{t+1}|s_t,a_t)}+\nabla_{\theta_{\text{adv}}}\mathbf{V}_\pi(s_{t+1})\right] && (\text{expectation over } \pi \text{ and } \pi_{\text{adv}})\\
&= \mathbb{E}_{\pi,\pi_{\text{adv}}}\left[\mathbf{V}_\pi(s_{t+1})\nabla_{\theta_{\text{adv}}}\log\left(\pi_{\text{adv}}(s_{t+1}|s_t,a_t)\right)+\nabla_{\theta_{\text{adv}}}\mathbf{V}_\pi(s_{t+1})\right]. && \text{(derivative of logarithm)}
\end{aligned}
$$

Therefore, expanding this sum across all times $t = 0, \ldots, T-1$, were T is the horizon of the decision process, the expression for $t = 0$ is given by

$$
\nabla_{\theta_{\text{adv}}}\mathbf{V}_\pi(s_0) = \sum_{k=0}^{T-1}\mathbb{E}_{\pi,\pi_{\text{adv}}}\left[\mathbf{V}_\pi(s_{k+1})\nabla_{\theta_{\text{adv}}}\log\left(\pi_{\text{adv}}(s_{t+1}|s_t,a_t)\right)\right].
$$

□

Table I
PARAMETER SETTINGS OF THE EXPERIMENTS

Parameter	**Setting**
Discount	0.99
Entropy regularisation for π	5.0
Architecture for π and π_{adv}	100 hidden RELU units, softmax output
Learning rates for θ,λ,θ_{adv}, and λ_{adv}	0.001, 0.0001, 0.001, and 0.0001, multiplier $\frac{1}{1+n//500}$ for episode n
Initialisation of λ and λ_{adv}	both 50 for Inventory Management, both 1 for Safe Navigation 1 & 2
Critic	learning rate 0.001, 100 hidden RELU units, linear output, Adam optimisation of MSE, batch is episode
Uncertainty set	Hoeffding-based L1, 1 pseudocount, 90% confidence interval

APPENDIX C: EXPERIMENT DETAILS

Inventory Management

For each item, the purchasing cost is 2.49, the selling price is 3.99, and the holding cost is 0.03. The reward $r(s,a)$ is the expected revenue minus the ordering costs and the holding costs. The demand distribution is Gaussian with mean μ and standard deviation σ. Each episode consists of $T = 100$ steps and the discount is set to $\gamma = 0.99$. The constraint-cost is $c(s,a) = \max(0, a - L(s)$, where the purchasing limit is set to $L(s) = \mu + \sigma$ for $s \leq 2$ and $L(s) = \mu$ for $s > 2$. The constraint-cost budget is set to $d = 6.0 \approx \sum_{t=0}^{T-1} \gamma^t 0.1$ which allows the action to exceed the purchasing limit on average roughly one item every 10 time steps. The constraint-cost function is not adjusted for tests; that is, the original μ and σ are used in its computation rather than the perturbed parameters.

Safe Navigation

The objective is to move from start, $s_0 = (0,0)$ to goal, $(4,4)$, as quickly as possible while avoiding areas that incur constraint-costs. The agent observes its (x,y)-coordinate and outputs an action going one step left, one step right, one step up, or one step down. The episode is terminated if either the agent arrives at the goal square or if more than T time steps have passed. Instead of using the full state space as next states in the uncertainty sets, the probability vectors $\mathcal{P}_{s,a}$ consider for the next state only the 5 states in the Von Neumann neighbourhood $\mathcal{N}(s)$ with Manhattan distance of at most 1 from s; this requires setting $\alpha(s,a) = \sqrt{\frac{2}{n(s,a)} \ln\left(\frac{2^{S'} SA}{\delta}\right)}$, replacing S by $S' = 5$ in the set of outcomes.

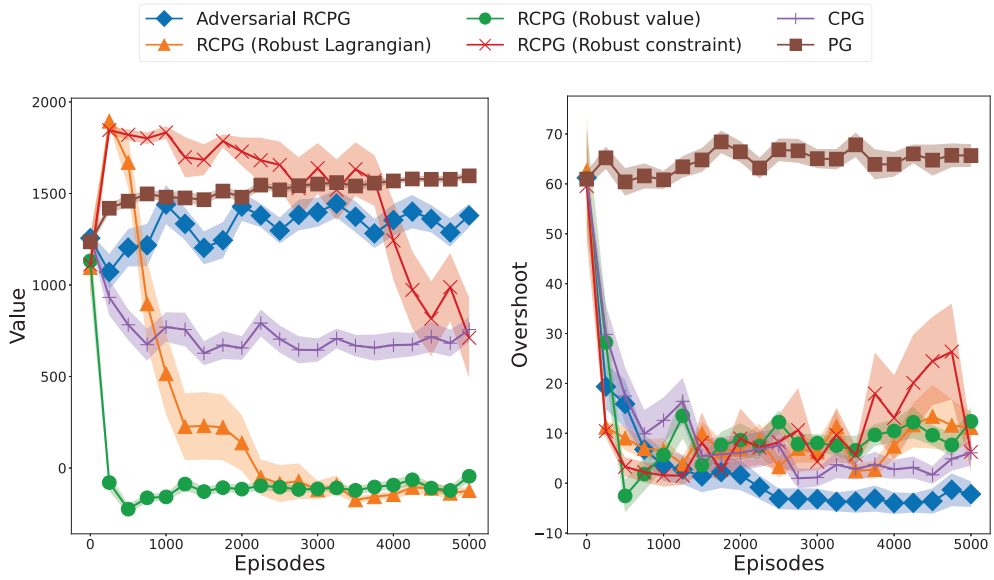
APPENDIX D: TRAINING HYPERPARAMETERS

A. Hyperparameters

Hyperparameters are set according to Table I. The discount is common at 0.99 and the architecture was chosen such that it is large enough for both domains. The entropy regularisation is higher than usual training procedures because of the Lagrangian yielding larger numbers in the objective. Learning rates were tuned in $\{0.10, 0.01, 0.001\}$ for policy parameters (θ and θ_{adv}) and in $\{0.01, 0.001, 0.0001\}$ for Lagrangian multipliers (λ and λ_{adv}); the setting shown in the table is the best setting for Inventory Management and Safe Navigation domains and this loosely corresponds to the two time scale stochastic approximation criteria [1]. The critic was fixed to 0.001 for both domains as this is a reliable setting for the Adam optimiser. For Inventory Management, it is possible to satisfy the constraint from the initial stages of learning so the initial Lagrangian multiplier λ is set to 50. For Safe Navigation domains, the initial λ is set to 1 since it is not immediately possible to satisfy the constraints without learning viable paths to goal. To encourage stochasticity in case of limited samples, each state-action pair (s,a) is initialised with a pseudo-count $n(s,a) \leftarrow 1$, representing the uniform distribution as a weak prior belief. The error probability is $\delta = 0.10$ for a 90% confidence interval.

APPENDIX E: TRAINING AND MODEL ESTIMATION

In Inventory Management, the model estimation phase is based on 100 episodes with $\mu = S/4$ and $\sigma = S/6$, yielding uncertainty sets with budget α ranging in $[0.3, 0.9]$ across the state-action space. The widely varying values and overshoots during training (see Figure 1) reflect in part a different training environment. Fig. 1 shows the performance in the policy training phase.



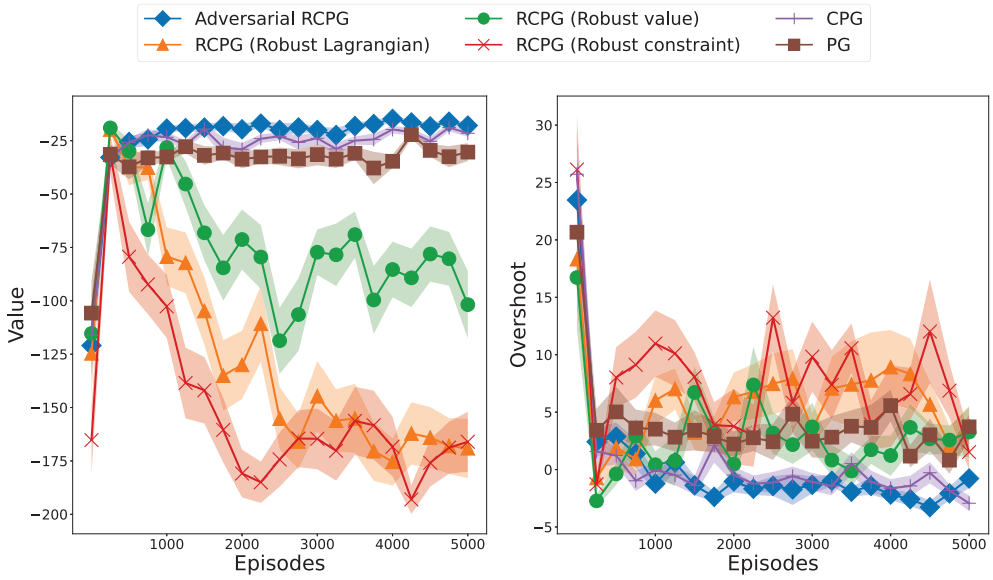
(a) Training value

(b) Training overshoot

Figure 1. Training performance metrics of the algorithms over 5,000 episodes on Inventory Management. Note that the training performance corresponds to the performance on the simulated transition dynamics, which is defined differently for the different algorithms.

In Safe Navigation 1, the model estimation phase is based on 100 episodes with $P_{\text{success}} = 0.80$, which results in the uncertainty budget α ranging in $[0.25, 0.7]$ across the state-action space. Fig. 2 shows the performance in the policy training phase.

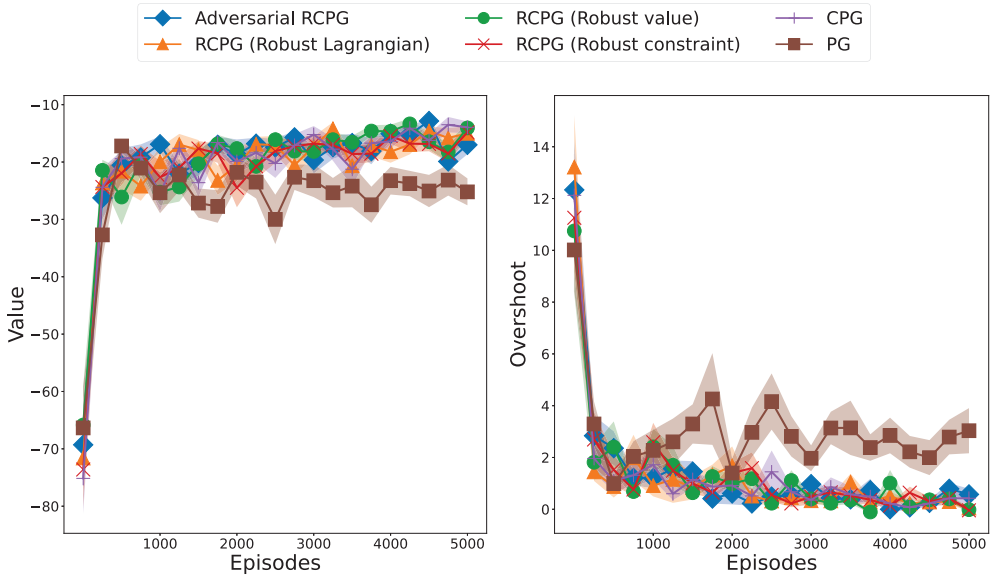
In Safe Navigation 2, the model estimation phase is based on 10,000 episodes with $P_{\text{success}} = 1.0$. The resulting uncertainty set has a smaller uncertainty budget, with α ranging in $[0.03, 0.085]$ across the state-action space. Fig. 3 shows the performance in the policy training phase.



(a) Training value

(b) Training overshoot

Figure 2. Training performance metrics of the algorithms over 5,000 episodes on Safe Navigation 1. Note that the training performance corresponds to the performance on the simulated transition dynamics, which is defined differently for the different algorithms.



(a) Training value

(b) Training overshoot

Figure 3. Training performance metrics of the algorithms over 5,000 episodes on Safe Navigation 2. Note that the training performance corresponds to the performance on the simulated transition dynamics, which is defined differently for the different algorithms.

REFERENCES

[1] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*. Springer, second ed., 2022.