ACPO: A Policy Optimization Algorithm for Average MDPs with Constraints

Akhil Agnihotri [1] **Rahul Jain** [1] **Haipeng Luo** [1]

Abstract

Reinforcement Learning (RL) for constrained MDPs (CMDPs) is an increasingly important problem for various applications. Often, the average criterion is more suitable than the discounted criterion. Yet, RL for average-CMDPs (ACMDPs) remains a challenging problem. Algorithms designed for discounted constrained RL problems often do not perform well for the average CMDP setting. In this paper, we introduce a new policy optimization with function approximation algorithm for constrained MDPs with the average criterion. The Average-Constrained Policy Optimization (ACPO) algorithm is inspired by trust region-based policy optimization algorithms. We develop basic sensitivity theory for average CMDPs, and then use the corresponding bounds in the design of the algorithm. We provide theoretical guarantees on its performance, and through extensive experimental work in various challenging OpenAI Gym environments, show its superior empirical performance when compared to other state-of-the-art algorithms adapted for the ACMDPs.

1. Introduction

Over the last decade, we have seen an enormous impact of RL techniques on a variety of problems, from mastering complex games like Go (Silver et al., 2017) and StarCraft (Vinyals et al., 2019) to robotic control (Levine et al., 2016; Akkaya et al., 2019; Aractingi et al., 2023). Many of these have used RL-policy optimization algorithms such as Schulman et al. (2017) for discounted MDPs (DMDPs). These have come in handy even in generative AI, e.g., training large language models (LLMs) (Achiam et al., 2023). However, applications often need satisfaction of some *constraints*, e.g., physical safety of mobile robots (Hu et al., 2022), safe language, image or multi-modal output generation. Furthermore, the *average criterion* when long-term rewards and safety are of consideration is more suitable. Using discounted cost formulations (as a proxy for safety) incentivizes policy optimization algorithms to search for policies that are short-term safe but not long-term because of future-discounting.

The planning problem for MDPs with constraints is often formulated as a Constrained MDP (CMDP) model (Manne, 1960; Hordijk & Kallenberg, 1979; Altman, 1999). Unfortunately, CMDP models do not satisfy Bellman's principle of optimality, and hence dynamic programming (DP)-style algorithms cannot be developed for the setting. Instead, an alternative approach called the convex analytic approach (Borkar, 1988; Altman, 1999) is used by way of introducing occupation measures that leads to optimization formulations. This can be done for both discounted (DCMDPs) and average-criterion (ACMDPs) constrained MDPs.

Theory and algorithms for RL deal with settings when the MDP model is unknown. While DP-inspired RL algorithms such as DQN, when combined with deep learning architectures for function approximation work remarkably effectively (Mnih et al., 2015), policy optimization algorithms such as TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017) have proven even more effective in solving high dimensional problems. Since the discounted criterion is sometimes not suitable, policy optimization algorithms such as ATRPO (Zhang et al., 2021; Wan et al., 2021; Liao et al., 2022) have been developed for AMDPs. Furthermore, as already mentioned, certain RL applications have multiple objectives, one of which is to be optimized and the rest constrained. Thus, the Constrained Policy Optimization (CPO) algorithm (Achiam et al., 2017) was introduced for infinite-horizon DCMDP problems. Unfortunately, as motivated above, not all such applications fit the discounted-criterion formulation: there are settings, for example where there may be safety requirements when the average-CMDP model is a better fit. No scalable RL algorithms are currently available for such settings.

We note that the RL problem is usually harder than the corresponding planning problem; average-MDPs are more challenging than discounted MDPs; and constrained MDPs are more challenging than unconstrained ones. In this paper, we

[1]University of Southern California, Los Angeles, CA, USA. RJ is also affiliated with Google DeepMind. Correspondence to: Akhil Agnihotri <agnihotri.akhil@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

present the first practical algorithm for policy optimization-based RL algorithm for average-constrained MDPs. We propose ACPO, a policy optimization algorithm for an average-CMDP with deep learning for function approximation. Our approach is motivated by theoretical guarantees that bound the difference between the average long-run rewards or costs of different policies. It draws inspiration from CPO (Achiam et al., 2017) (see also Tessler et al. (2019) for DCMDPs), which uses a policy improvement theorem for the discounted setting based on the trust region methods of Schulman et al. (2015). Unfortunately, this result trivializes for the average setting and hence can't be used. Instead, we derive a new bound that depends on the worst-case level of "mixture" of the irreducible Markov chain associated with a policy. Our proposed algorithm, ACPO is based on these theoretical developments. For experimental evaluation, we use several OpenAI Gym environments from Todorov et al. (2012), train large neural network policies and demonstrate the effectiveness and superior performance of the ACPO algorithm as compared to others.

Main Contributions and Novelty.

Algorithmic: We introduce the first practical policy optimization-based RL algorithm for average-constrained MDPs with new and tight performance bounds and violation guarantees. The algorithm draws inspiration from CPO (for discounted-CMDPs) and ATRPO (for average-MDPs) but is not a straightforward extension of either. One may posit that setting the discount factor $\gamma = 1$ in CPO for the discounted setting may suffice but that does not perform well on average-CMDPs even with a large discount factor. Further, constraint violation and policy degradation bounds of CPO do not hold in the average setting and hence we develop novel bounds (in Corollary 3.5). In fact, the advantage function estimation routine in our algorithm (line 4 and 6 in Algorithm 1) is also different from that in CPO, since the discounted-setting procedure cannot be used for the average setting (see Appendix A.3): We first approximate the average-reward bias and then use a one-step TD backup to estimate the action-bias function. Furthermore, policy optimization algorithms for the average case (Zhang & Ross, 2020; Wan et al., 2021; Liao et al., 2022) cannot incorporate constraints. We enable this by introducing sublevel sets of cost constraints. We also introduce an approximate but novel line search procedure that improves the empirical performance of our algorithm, an idea that may help improve performance of other policy optimization algorithms such as PPO.

Technical: Since ACPO is a trust region method, one can expect some overlap in analysis techniques with other similar algorithms. Nevertheless, our analysis has several novel elements: Lemma 3.3, where we use eigenvalues of the transition matrix to relate total variation of stationary distributions with that of the policies, and in Lemma A.6, we use the sublevel sets of constraints and projection inequality of Bregman divergence. Furthermore, several important results from CPO and ATRPO papers cannot be applied to the analysis of our algorithm.

Empirical: We evaluate the empirical performance of ACPO in the OpenAI Gym (Mujoco) environments, a standard benchmark. We find that ACPO outperforms all state-of-the-art Deep RL algorithms such as CPO in (Achiam et al., 2017), PCPO in (Yang et al., 2020), PPO in (Schulman et al., 2017), BVF-PPO in (Satija et al., 2020) and ATRPO in (Zhang et al., 2021). We use a large discount factor if the algorithm is not for the average setting, and a Lagrangian objective if it is not for the constrained setting, and in some cases both.

Significance: ACPO is the first practical trust region-style policy optimization algorithm for ACMDPs with excellent empirical performance. ACMDPs are important models because they allow incorporation of long term safety constraints, which are important not only in the context of safe robot learning and control, but also safety-constrained RLHF fine-tuning and inference for LLMs (Moskovitz et al., 2023) and Diffusion models as well. In the absence of suitable policy optimization algorithms for ACMDPs, researchers have resorted to using adaptations of PPO, etc.

Related Work. Learning constraint-satisfaction policies has been explored in the Deep RL literature in (Agnihotri et al., 2019; Garcia & Fernandez, 2015). This can either be (1) through expert annotations and demonstrations as in (Rajeswaran et al., 2017; Gao et al., 2018) or, (2) by exploration with constraint satisfaction as in (Achiam et al., 2017; Tessler et al., 2019). While the former approach is not scalable since it requires human interventions, current state-of-the-art algorithms for the latter are not applicable to the average reward setting.

Previous work on RL with the average reward criterion has mostly attempted to extend stochastic approximation schemes for the tabular setting, such as Q-learning in (Abounadi et al., 2001; Wan et al., 2021), to the non-tabular setting with function approximation in (Wei et al., 2021; Zhang & Ross, 2020). (Chen et al., 2022) deals with online learning in a constrained MDP setting, but their aim is regret minimization or exploration, both in tabular settings. We are inspired by the work of (Zhang et al., 2021) to develop techniques required to derive the policy degradation and constraint violation bounds in Section 3.

The more recent works of (Bhatnagar & Lakshmanan, 2012) and (Calvo-Fullana et al., 2023) also fail to address our problem setting as the former test on a 2x4 queueing network with maximum state space of 128, while the latter test on a grid of size 10x10 (maximum states of 100). In addition to that, the way they incorporate constraints during training is just via a Lagrangian formulation. In our paper we show

that simply doing this (in the case of PPO and ATRPO for example) leads to much inferior performance to ACPO, which can outperform current state-of-the-art algorithms in state spaces of upto 10^{96}.

2. Preliminaries

A Markov decision process (MDP) is a tuple, (S, A, r, P, μ), where S is the set of states, A is the set of actions, $r : S \times A \times S \to \mathbb{R}$ is the reward function, $P : S \times A \times S \to [0, 1]$ is the transition probability function such that $P(s'|s, a)$ is the probability of transitioning to state s' from state s by taking action a, and $\mu : S \to [0, 1]$ is the initial state distribution. A stationary policy $\pi : S \to \Delta(A)$ is a mapping from states to probability distributions over the actions, with $\pi(a|s)$ denoting the probability of selecting action a in state s, and $\Delta(A)$ is the probability simplex over the action space A. We denote the set of all stationary policies by Π. For the average setting, we will make the standard assumption that the MDP is *ergodic* and is *unichain*.

In reinforcement learning, we aim to select a policy π which maximizes a performance measure, $J(\pi)$, which, for continuous control tasks is either the discounted reward criterion or the average reward approach. Below, we briefly discuss both formulations.

2.1. Discounted criterion

For a given discount factor $\gamma \in (0, 1)$, the discounted reward objective is defined as

$$\begin{aligned} J_\gamma(\pi) &:= \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t, s_{t+1})\right] \\ &= \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{\substack{s\sim d_{\pi,\gamma}\\ a\sim\pi\\ s'\sim P(\cdot|s,a)}}[r(s, a, s')] \end{aligned}$$

where τ refers to a sample trajectory of $(s_0, a_0, s_1, \cdots)$ generated when following a policy π, that is, $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim P(\cdot|s_t, a_t)$; $d_{\pi,\gamma}$ is the *discounted occupation measure* that is defined by $d_{\pi,\gamma}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathop{P}_{\tau\sim\pi}(s_t = s)$, which essentially refers to the discounted fraction of time spent in state s while following policy π.

2.2. Average criterion

The average-reward objective is given by:

$$\begin{aligned} J(\pi) &:= \lim_{N\to\infty}\frac{1}{N}\mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{N-1} r(s_t, a_t, s_{t+1})\right] \\ &= \mathop{\mathbb{E}}_{\substack{s\sim d_\pi\\ a\sim\pi(\cdot|s)\\ s'\sim P(\cdot|s,a)}}[r(s, a, s')], \end{aligned} \tag{1}$$

where $d_\pi(s) := \lim_{N\to\infty}\frac{1}{N}\sum_{t=0}^{N-1} P_{\tau\sim\pi}(s_t = s)$ is the *stationary state distribution* under policy π. The limits in $J(\pi)$ and $d_\pi(s)$ are guaranteed to exist under our ergodic assumption. Since the MDP is aperiodic, it can also be shown that $d_\pi(s) = \lim_{t\to\infty} P_{\tau\sim\pi}(s_t = s)$. Since we have $\lim_{\gamma\to 1} d_{\pi,\gamma}(s) \to d_\pi(s), \forall s$, it can be shown that $\lim_{\gamma\to 1}(1-\gamma)J_\gamma(\pi) = J(\pi)$.

In the average setting, we seek to keep the estimate of the state value function unbiased and hence, introduce the *average-reward bias function* as

$$\bar{V}^\pi(s) := \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}(r(s_t, a_t, s_{t+1}) - J(\pi)) \,\middle|\, s_0 = s\right]$$

and the *average-reward action-bias function* as

$$\bar{Q}^\pi(s, a) := \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}(r(s_t, a_t, s_{t+1}) - J(\pi)) \,\middle|\, \begin{matrix} s_0 = s,\\ a_0 = a\end{matrix}\right].$$

Finally, define the *average-reward advantage function* as $\bar{A}^\pi(s, a) := \bar{Q}^\pi(s, a) - \bar{V}^\pi(s)$.

2.3. Constrained MDPs

A constrained Markov decision process (CMDP) is an MDP augmented with constraints that restrict the set of allowable policies for that MDP. Specifically, we augment the MDP with a set C of auxiliary cost functions, $C_1, \cdots, C_m$ (with each function $C_i : S \times A \times S \to \mathbb{R}$ mapping transition tuples to costs, just like the reward function), and bounds $l_1, \cdots, l_m$. Similar to the value functions being defined for the average reward criterion, we define the average cost objective with respect to the cost function C_i as

$$\begin{aligned} J_{C_i}(\pi) &:= \lim_{N\to\infty}\frac{1}{N}\mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{N-1} C_i(s_t, a_t, s_{t+1})\right] \\ &= \mathop{\mathbb{E}}_{\substack{s\sim d_\pi\\ a\sim\pi\\ s'\sim P(\cdot|s,a)}}[C_i(s, a, s')]. \end{aligned} \tag{2}$$

where J_{C_i} will be referred to as the *average cost* for constraint C_i. The set of feasible stationary policies for a CMDP then is given by $\Pi_C := \{\pi \in \Pi : J_{C_i}(\pi) \leqslant l_i, \ \forall\, i \in \{1, \cdots, M\}\}$. The goal is to find a policy $\pi^\star$ such that $\pi^\star \in \arg\max_{\pi\in\Pi_C} J(\pi)$.

However, finding an exact $\pi^\star$ is infeasible for large-scale problems. Instead, we aim to derive an iterative policy improvement algorithm that given a current policy, improves upon it by approximately maximizing the increase in the reward, while not violating the constraints by too much and not being too different from the current policy.

Lastly, analogous to $\bar{V}^\pi$, $\bar{Q}^\pi$, and $\bar{A}^\pi$, we define similar quantities for the cost functions $C_i(\cdot)$, and denote them by $\bar{V}^\pi_{C_i}$, $\bar{Q}^\pi_{C_i}$, and $\bar{A}^\pi_{C_i}$.

2.4. Policy Improvement for discounted CMDPs

In many on-policy constrained RL problems, we improve policies iteratively by maximizing a predefined function within a local region of the current best policy as in (Tessler et al., 2019; Achiam et al., 2017; Yang et al., 2020; Song et al., 2020). (Achiam et al., 2017) derived a policy improvement bound for the discounted CMDP setting as:

$$J_\gamma(\pi_{k+1}) - J_\gamma(\pi_k) \geqslant \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi_{k+1}}} \left[A_\gamma^{\pi_k}(s,a) - \frac{2\gamma\epsilon^{\pi_{k+1}}}{1-\gamma} D_{TV}(\pi_{k+1}||\pi_k)[s] \right], \tag{3}$$

where $A_\gamma^{\pi_k}$ is the discounted version of the advantage function, $\epsilon^{\pi_{k+1}} := \max_s |\mathbb{E}_{a\sim\pi_{k+1}}[A_\gamma^{\pi_k}(s,a)]|$, and $D_{TV}(\pi_{k+1}||\pi_k)[s] = (1/2)\sum_a |\pi_{k+1}(a|s) - \pi_k(a|s)|$ is the total variational divergence between π_{k+1} and π_k at s. These results laid the foundations for on-policy constrained RL algorithms as in(Wu et al., 2017; Vuong et al., 2019).

However, Equation (3) does not generalize to the average setting ($\gamma \to 1$) (see Appendix A.1). In the next section, we will derive a policy improvement bound for the average case and present an algorithm based on trust region methods, which will generate almost-monotonically improving iterative policies. Proofs of theorems and lemmas, if not already given, are available in Appendix A.

3. ACPO: The Average-Constrained Policy Optimization Algorithm

In this section, we present the main results of our work. For conciseness, we denote by $d_\pi \in \mathbb{R}^{|S|}$ the column vector whose components are $d_\pi(s)$ and $P_\pi \in \mathbb{R}^{|S|\times|S|}$ to be the state transition probability matrix under policy π.

3.1. Policy Improvement for the Average-CMDP

Let π' be the policy obtained via some update rule from the current policy π. Analogous to the discounted setting of a CMDP, we would like to characterize the performance difference $J(\pi') - J(\pi)$ by an expression which depends on π and some divergence metric between the two policies.

Lemma 3.1. *(Zhang & Ross, 2020) Under the unichain assumption of the underlying Markov chain, for any stochastic policies π and π':*

$$J(\pi') - J(\pi) = \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi'} \\ a\sim\pi'}} \left[\bar{A}^\pi(s,a) \right]. \tag{4}$$

Note that this difference depends on the stationary state distribution obtained from the *new* policy, $d_{\pi'}$. This is computationally impractical as we do not have access to this $d_{\pi'}$. Fortunately, by use of the following lemma we can show that if d_π and $d_{\pi'}$ are "close" with respect to some metric, we can approximate Eq. (4) using samples from d_π.

Lemma 3.2. *Under the unichain assumption, for any stochastic policies π and π' we have:*

$$\left| J(\pi') - J(\pi) - \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\pi'}} \left[\bar{A}^\pi(s,a) \right] \right| \leqslant 2\epsilon D_{TV}(d_{\pi'} \parallel d_\pi) \tag{5}$$

, where $\epsilon = \max_s \left| \mathop{\mathbb{E}}_{a\sim\pi'} [\bar{A}^\pi(s,a)] \right|$. See Appendix A.1 for proof. Lemma 3.2 implies $J(\pi') \approx J(\pi) + \mathbb{E}[\bar{A}^\pi(s,a)]$ when d_π and $d_{\pi'}$ are "close". Now that we have established this approximation, we need to study the relation of how the actual change in policies affects their corresponding stationary state distributions. For this, we turn to standard analysis of the underlying Markov chain of the CMDP.

Under the ergodic assumption, we have that P_π is irreducible and hence its eigenvalues $\{\lambda_{\pi,i}\}_{i=1}^{|S|}$ are such that $\lambda_{\pi,1} = 1$ and $\lambda_{\pi,i\neq 1} < 1$. For our analysis, we define $\sigma^\pi = \max_{i\neq 1} (1-\lambda_{\pi,i})^{-1/2}$, and from (Levene & Loizou, 2002) and (Doyle, 2009), we connect $\{\lambda_{\pi,i}\}_{i=1}^{|S|}$ to the sensitivity of the stationary distributions to changes in the policy using the result below.

Lemma 3.3. *Under the ergodic assumption, the divergence between the stationary distributions d_π and $d_{\pi'}$ is upper bounded as:*

$$D_{TV}(d_{\pi'} \parallel d_\pi) \leqslant \sigma^\star \mathop{\mathbb{E}}_{s\sim d_\pi} [D_{TV}(\pi' \parallel \pi)[s]], \tag{6}$$

, where $\sigma^\star = \max_\pi \sigma^\pi$. See Appendix A.1 for proof. This bound is tighter and easier to compute than the one given by (Zhang et al., 2021), which replaces $\sigma^\star$ by $\kappa^\star = \max_\pi \kappa^\pi$, where κ^π is known as *Kemeny's constant* from (Kemeny & Snell, 1960). It is interpreted as the expected number of steps to get to any goal state, where the expectation is taken with respect to the stationary-distribution of those states.

Combining the bounds in Lemma 3.2 and Lemma 3.3 gives us the following result:

Proposition 3.4. *Under the ergodic assumption, the following bounds hold for any stochastic policies π and π':*

$$L_\pi^-(\pi') \leqslant J(\pi') - J(\pi) \leqslant L_\pi^+(\pi') \tag{7}$$

where

$$L_\pi^\pm(\pi') = \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\pi'}} \left[\bar{A}^\pi(s,a) \right] \pm 2\nu \mathop{\mathbb{E}}_{s\sim d_\pi} [D_{TV}(\pi' \parallel \pi)[s]]$$

and $\nu = \sigma^\star \max_s \left| \mathop{\mathbb{E}}_{a\sim\pi'} [\bar{A}^\pi(s,a)] \right|$.

It is interesting to compare the inequalities of Equation (7) to Equation (4). The term $\mathbb{E}[\bar{A}^{\pi}(s,a)]$ in Prop. 3.4 is somewhat of a *surrogate* approximation to $J(\pi') - J(\pi)$, in the sense that it uses d_π instead of $d_{\pi'}$. As discussed before, we do not have access to $d_{\pi'}$ since the trajectories of the new policy are not available unless the policy itself is updated. This surrogate is a first order approximation to $J(\pi') - J(\pi)$ in the parameters of π' in a neighborhood around π as in (Kakade & Langford, 2002). Hence, Eq. (7) can be viewed as bounding the worst-case approximation error.

Extending this discussion to the cost function of our CMDP, similar expressions follow immediately.

Corollary 3.5. *For any policies π', π, and any cost function C_i, the following bound holds:*

$$M_\pi^-(\pi') \leqslant J_{C_i}(\pi') - J_{C_i}(\pi) \leqslant M_\pi^+(\pi') \tag{8}$$

where

$$M_\pi^\pm(\pi') = \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\pi'}} \left[\bar{A}^{\pi}_{C_i}(s,a)\right] \pm 2\nu_{C_i} \mathop{\mathbb{E}}_{s\sim d_\pi} [D_{TV}(\pi' \parallel \pi)[s]]$$

and $\nu_{C_i} = \sigma^\star \max_s \left| \mathop{\mathbb{E}}_{a\sim\pi'} [\bar{A}^{\pi}_{C_i}(s,a)] \right|$.

Until now, we have been dealing with bounds given with regards to the TV divergence of the policies. However, in practice, bounds with respect to the KL divergence of policies is more commonly used as in (Schulman et al., 2015; 2016; Ma et al., 2021). From Pinsker's and Jensen's inequalities, we have that

$$\mathop{\mathbb{E}}_{s\sim d_\pi} \left[D_{\text{TV}}(\pi' \parallel \pi)[s]\right] \leqslant \sqrt{\mathop{\mathbb{E}}_{s\sim d_\pi} \left[D_{\text{KL}}(\pi' \| \pi)][s]\right]/2}. \tag{9}$$

We can thus use Eq. (9) in the bounds of Proposition 3.4 and Corollary 3.5 to make policy improvement guarantees, i.e., if we find updates such that $\pi_{k+1} \in \arg\max_\pi L^-_{\pi_k}(\pi)$, then we will have monotonically increasing policies as, at iteration k, $\mathop{\mathbb{E}}_{s\sim d_{\pi_k}, a\sim\pi} [\bar{A}^{\pi_k}(s,a)] = 0$, $\mathop{\mathbb{E}}_{s\sim d_{\pi_k}} [D_{\text{KL}}(\pi \| \pi_k)[s]] = 0$ for $\pi = \pi_k$, implying that $J(\pi_{k+1}) - J(\pi_k) \geqslant 0$. However, this sequence does not guarantee constraint satisfaction at each iteration, so we now turn to trust region methods to incorporate constraints, do policy improvement and provide safety guarantees.

3.2. Trust Region Based Approach

For large or continuous state and action CMDPs, solving for the exact optimal policy is impractical. However, *trust region*-based policy optimization algorithms have proven to be effective for solving such problems as in (Schulman et al., 2015; 2016; 2017; Achiam, 2017). For these approaches, we usually consider some parameterized policy class $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$ for tractibility. In addition, for CMDPs, we also require the policy iterates to be feasible, so instead of optimizing just over Π_Θ, we optimize over $\Pi_\Theta \cap \Pi_C$. However, it is much easier to solve the above problem if we introduce hard constraints, rather than limiting the set to $\Pi_\Theta \cap \Pi_C$. Therefore, we now introduce the ACPO algorithm, which is inspired by the trust region formulations above as the following optimization problem:

$$\begin{aligned} \underset{\pi\in\Pi_\Theta}{\text{maximize}} \quad & \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi_{\theta_k}} \\ a\sim\pi}} [\bar{A}^{\pi_{\theta_k}}(s,a)] \\ \text{s.t.} \quad & J_{C_i}(\pi_{\theta_k}) + \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi_{\theta_k}} \\ a\sim\pi}} [\bar{A}^{\pi_{\theta_k}}_{C_i}(s,a)] \leqslant l_i, \quad \forall\, i \\ & \bar{D}_{\text{KL}}(\pi \parallel \pi_{\theta_k}) \leqslant \delta \end{aligned} \tag{10}$$

where $\bar{D}_{\text{KL}}(\pi \parallel \pi_{\theta_k}) := \mathop{\mathbb{E}}_{s\sim d_{\pi_{\theta_k}}} [D_{\text{KL}}(\pi \| \pi_{\theta_k})[s]]$, $\bar{A}^{\pi_{\theta_k}}(s,a)$ is the average advantage function defined earlier, and $\delta > 0$ is a step size. We use this form of updates as it is an approximation to the lower bound given in Proposition 3.4 and the upper bound given in Corollary 3.5.

In most cases, the trust region threshold for formulations like Eq. (10) are heuristically motivated. We now show that it is quantitatively motivated and comes with a worst case performance degradation and constraint violation. Proof is in Appendix A.2.

Theorem 3.6. *Let $\pi_{\theta_{k+1}}$ be the optimal solution to Eq. (10) for some $\pi_{\theta_k} \in \Pi_\Theta$. Then, we have*

$$J(\pi_{\theta_{k+1}}) - J(\pi_{\theta_k}) \geqslant -\sqrt{2(\delta + V_{max})}\nu^{\pi_{\theta_{k+1}}} \tag{11}$$

$$\text{and} \quad J_{C_i}(\pi_{\theta_{k+1}}) \leqslant l_i + \sqrt{2(\delta + V_{max})}\nu^{\pi_{\theta_{k+1}}}_{C_i} \quad \forall\, i, \tag{12}$$

where $\nu^{\pi_{\theta_{k+1}}} = \sigma^{\pi_{\theta_{k+1}}} \max_s \left| \mathop{\mathbb{E}}_{a\sim\pi_{\theta_{k+1}}} [\bar{A}^{\pi_{\theta_k}}(s,a)] \right|$, $\nu^{\pi_{\theta_{k+1}}}_{C_i} = \sigma^{\pi_{\theta_{k+1}}} \max_{i,s} \left| \mathop{\mathbb{E}}_{a\sim\pi_{\theta_{k+1}}} [\bar{A}^{\pi_{\theta_k}}_{C_i}(s,a)] \right|$, $V_{max} = \max_i \beta_i^2$, *and* $\beta_i = [J_{C_i}(\pi_{\theta_k}) - l_i]_+$.

Remark 3.7. Note that if the constraints are ignored (by setting $V_{max} = 0$), then this bound is tighter than given in (Zhang et al., 2021) for the unconstrained average-reward setting.

However, the update rule of Eq. (10) is difficult to implement in practice as it takes steps that are too small, which degrades convergence. In addition, it requires the exact knowledge of $\bar{A}^{\pi_{\theta_k}}(s,a)$ which is computationally infeasible for large-scale problems. In the next section, we will introduce a specific sampling-based practical algorithm to alleviate these concerns.

4. Practical Implementation of ACPO

In this section, we introduce a practical version of the ACPO algorithm with a principle recovery method. With a small step size δ, we can approximate the reward function and constraints with a first order expansion, and approximate the KL divergence constraint with a second order expansion. This gives us a new optimization problem which can be solved exactly using Lagrangian duality.

4.1. An Implementation of ACPO

Since we are working with a parameterized class, we shall now overload notation to use θ_k as the policy at iteration k, i.e., $\theta_k \equiv \pi_{\theta_k}$. In addition, we use g to denote the gradient of the advantage function objective, a_i to denote the gradient of the advantage function of the cost C_i, H as the Hessian of the KL-divergence. Formally,

$$g := \nabla_\theta \mathop{\mathbb{E}}_{\substack{s\sim d_{\theta_k}\\ a\sim\theta}} [\bar{A}^{\theta_k}(s,a)]\Big|_{\theta=\theta_k},$$

$$a_i := \nabla_\theta \mathop{\mathbb{E}}_{\substack{s\sim d_{\theta_k}\\ a\sim\theta}} [\bar{A}^{\theta_k}_{C_i}(s,a)]\Big|_{\theta=\theta_k},$$

$$H := \nabla^2_\theta \mathop{\mathbb{E}}_{s\sim d_{\theta_k}} \left[D_{\mathrm{KL}}\left(\theta\|\theta_k\right))[s]\right]\Big|_{\theta=\theta_k}.$$

In addition, let $c_i := J_{C_i}(\theta_k) - l_i$. The approximation to the problem in Eq. (10) is:

$$\begin{aligned} \max_{\theta} \quad & g^T(\theta-\theta_k) \\ \text{s.t.} \quad & c_i + a_i^T(\theta-\theta_k) \leqslant 0, \quad \forall\, i \\ \text{and,} \quad & \tfrac{1}{2}(\theta-\theta_k)^T H(\theta-\theta_k) \leqslant \delta. \end{aligned} \tag{13}$$

This is a convex optimization problem in which strong duality holds, and hence it can be solved using a Lagrangian method. The update rule for the dual problem then takes the form

$$\theta_{k+1} = \theta_k + \frac{1}{\lambda^\star} H^{-1}\left(g - A\mu^\star\right). \tag{14}$$

where $\lambda^\star$ and $\mu^\star$ are the Lagrange multipliers satisfying the dual

$$\max_{\substack{\lambda\geqslant 0\\ \mu\succeq 0}} \frac{-1}{2\lambda}\left(g^T H^{-1} g - 2r^T\mu + \mu^T S\mu\right) + \mu^T c - \frac{\lambda\delta}{2}, \tag{15}$$

with $r := g^T H^{-1} A$, $S := A^T H^{-1} A$, $A := [a_1, \cdots, a_m]$, and $c := [c_1, \cdots, c_m]^T$.

4.2. Feasibility and Recovery

The approximation regime described in Eq. (13) requires H to be invertible. For large parametric policies, H is computed using the conjugate gradient method as in (Schulman et al., 2015). However, in practice, using this approximation along with the associated statistical sampling errors, there might be potential violations of the approximate constraints leading to infeasible policies.

Algorithm 1 Average-Constrained Policy Optimization (ACPO)

Input: Initial random policy $\pi_0 \in \Pi_\theta$
for $k = 0, 1, 2, ..., K$ **do**
  Sample a set of trajectories Ω using $\pi_k = \pi_{\theta_k}$
  Find estimates of g, a, H, c using Ω
  if a feasible solution to Equation (13) exists **then**
    Solve dual problem in Equation (15) for $\lambda_k^\star, \mu_k^\star$
    Find policy update π_{k+1} with Equation (14)
  else
    Find recovery policy $\pi_{k+1/2}$ with Equation (16)
    Obtain π_{k+1} by linesearch till approximate constraint satisfaction of Equation (13)
  end if
end for

To rectify this, for the case where we only have one constraint, one can recover a feasible policy by applying a recovery step inspired by the TRPO update on the cost surrogate as:

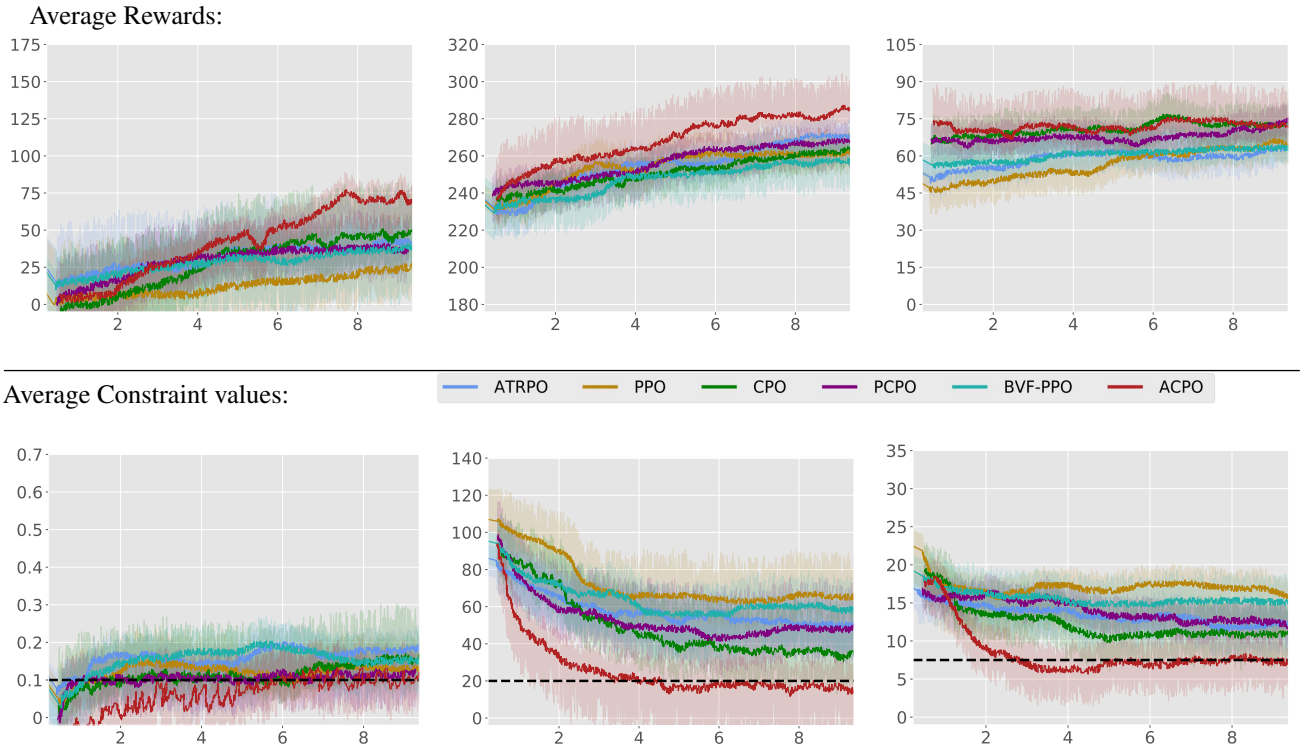
$$\theta_{k+1/2} = \theta_k - \sqrt{2\delta}\left[t\cdot\frac{H^{-1}a}{\sqrt{a^T H^{-1} a}} + (1-t)\cdot\frac{H^{-1}g}{\sqrt{g^T H^{-1} g}}\right] \tag{16}$$

where $t \in [0, 1]$. Contrasting with the policy recovery update of (Achiam et al., 2017) which only uses the cost advantage function gradient a, we introduce the reward advantage function gradient g as well. This choice is to ensure recovery while simultaneously balancing the "regret" of not choosing the best (in terms of the objective value) policy π_k. In other words, we wish to find a policy $\pi_{k+1/2}$ as close to π_k in terms of their objective function values. We follow up this step with a simple linesearch to find feasible π_{k+1}. Based on this, Algorithm 1 provides a basic outline of ACPO. For more details of the algorithm, see Appendix A.3.

5. Empirical Results

We conducted a series of experiments to evaluate the relative performance of the ACPO algorithm and answer the following questions: (i) Does ACPO learn a sequence of constraint satisfying policies while maximizing the average reward in the long run? (ii) How does ACPO compare with the already existing constraint policy optimization algorithms which are applied with a large discount factor? (iii) What are the factors that affect the performance of ACPO?

We work with the OpenAI Gym environments to train the various learning agent on the following tasks - *Gather*, *Circle*, *Grid*, and *Bottleneck* tasks (see Figure 3 in Appendix A.6.1 for more details on the environments). For our exper-



(a) Ant Gather (b) Bottleneck (c) Grid

Figure 1. The average reward and constraint cost function values vs iterations (in 10^4) learning curves for some algorithm-task pairs. Solid lines in each figure are the empirical means, while the shaded area represents 1 standard deviation, all over 5 runs. The dashed line in constraint plots is the constraint threshold l. ATRPO and PPO are tested with constraints, which are included in their Lagrangian formulation. Additional results are available in Appendix A.6.

iments we only work with a single constraint with policy recovery using Eq. (16) (this is only a computational limitation; ACPO in principle can handle multiple constraints). We compare ACPO with the following baseline algorithms: CPO by (Achiam et al., 2017), ATRPO by (Zhang et al., 2021), PCPO by (Yang et al., 2020) (a close variant of CPO), BVF-PPO by (Satija et al., 2020) and PPO by (Schulman et al., 2017).

Although ATRPO and PPO originally do not incorporate constraints, for fair comparison, we introduce constraints using a Lagrangian. Also, CPO, PCPO and PPO are compared with $\gamma = 0.999$. See Appendix A.5 for more details.

5.1. Evaluation Details and Protocol

For the Gather and Circle tasks we test two distinct agents: a point-mass ($S \subseteq \mathbb{R}^9, A \subseteq \mathbb{R}^2$), and an ant robot ($S \subseteq \mathbb{R}^{32}, A \subseteq \mathbb{R}^8$). The agent in the Bottleneck task in $S \subseteq \mathbb{R}^{71}, A \subseteq \mathbb{R}^{16}$, and for the Grid task is $S \subseteq \mathbb{R}^{96}, A \subseteq \mathbb{R}^4$. We use two hidden layer neural networks to represent Gaussian policies for the tasks. For Gather and Circle, size is (64,32) for both layers, and for Grid and Bottleneck the layer sizes are (16,16) and (50,25). We set the step size δ to 10^{-4}, and for each task, we conduct 5 runs to get the mean and standard deviation for reward objective and cost constraint values during training. We train CPO, PCPO, and PPO with the discounted objective, however, evaluation and comparison with BVF-PPO, ATRPO and ACPO[1] is done using the average reward objective (this is a standard evaluation scheme as in (Schulman et al., 2015; Wu et al., 2017; Vuong et al., 2019)).

For each environment, we train an agent for 10^5 steps, and for every 10^3 steps, we instantiate 10 evaluation trajectories with the current (deterministic) policy. For each of these trajectories, we calculate the trajectory average reward for the next 10^3 steps and finally report the total average-reward as the mean of these 10 trajectories. Learning curves for the algorithms are compiled in Figure 1 (for Point-Circle, Point-Gather, and Ant-Circle see Appendix A.6).

Since there are two objectives (rewards in the objective and costs in the constraints), we show the plots which maximize the reward objective while satisfying the cost constraint. See Appendix A.4 and A.5 for more details.

[1]Code of the ACPO implementation will be made available on GitHub.

5.2. Performance Analysis

From Figure 1, we can see that ACPO is able to improve the reward objective while having approximate constraint satisfaction on all tasks. In particular, ACPO is the only algorithm that best learns almost-constraint-satisfying maximum average-reward policies across all tasks: in a simple Gather environment, ACPO is able to almost exactly track the cost constraint values to within the given threshold l; however, for the high dimensional Grid and Bottleneck environments we have more constraint violations due to complexity of the policy behavior. Regardless, in these environments, ACPO still outperforms all other baselines.

ACPO vs. CPO/PCPO. For the Point-Gather environment (see Figure 4), we see that initially ACPO and CPO/PCPO give relatively similar performance, but eventually ACPO improves over CPO and PCPO by 52.5% and 36.1% on average-rewards respectively. This superior performance does not come with more constraint violation. The Ant-Gather environment particularly brings out the effectiveness of ACPO where it shows 41.1% and 61.5% improvement over CPO and PCPO respectively, while satisfying the constraint. In the high dimensional Bottleneck and Grid environments, ACPO is particularly quick at optimizing for low constraint violations, while improving over PCPO and CPO in terms of average-reward.

ACPO vs Lagrangian ATRPO/PPO. One could suppose to use the state of the art unconstrained policy optimization algorithms with a Lagrangian formulation to solve the average-rewards CMDP problem in consideration, but we see that such an approach, although principled in theory, does not give satisfactory empirical results. This can be particularly seen in the Ant-Circle, Ant-Gather, Bottleneck, and Grid environments, where Lagrangian ATRPO and PPO give the least rewards, while not even satisfying the constraints. If ATRPO and PPO were used with constraints ignored, one would see higher rewards but even worse constraint violations, which are not useful.

ACPO vs BVF-PPO. BVF-PPO is a whole different formulation than the other baselines, as it translates the cumulative cost constraints into state-based constraints, which results in an almost-safe policy improvement method which maximizes returns at every step. However, we see that this approach fails to satisfy the constraints even in the moderately difficult Ant Gather environment, let alone the high dimensional Bottleneck and Grid environments.

5.3. Dependence of the Recovery Regime

In Equation (16) we introduced a hyperparameter t, which provides for an intuitive trade-off as follows: either we purely decrease the constraint violations ($t = 1$), or we decrease the average-reward ($t = 0$), which consequently decreases the constraint violation. The latter formulation is principled in that if we decrease rewards, we are bound to decrease constraints violation due to the nature of the environments. Figure 2 shows the experiments we conducted with varying t. With $t = 1$, we obtain the same recovery scheme as that of (Achiam et al., 2017). Our results show that this scheme does not lead to the best performance, and that $t = 0.75$ and $t = 1$ perform the best across all tasks. See Appendix A.6 for a detailed study.

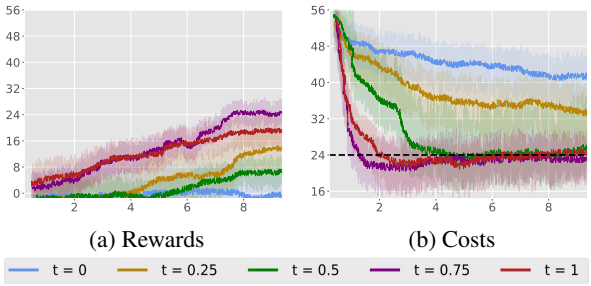


Figure 2. Comparison of performance of ACPO with different values of the hyperparameter t in the Point-Circle environment. X-axis is iterations in 10^4. See Appendix A.6 for more details.

6. Conclusions

In this paper, we studied the problem of learning policies that maximize average-rewards for a given CMDP with average-cost constraints. We showed that the current algorithms with constraint violation bounds for the discounted setting do not generalize to the average setting. We then proposed a new algorithm, the Average-Constrained Policy Optimization (ACPO) that is inspired by the TRPO class of algorithms but based on theoretical sensitivity-type bounds for average-CMDPs we derive, and use in designing the algorithm. Our experimental results on a range of OpenAI Gym environments (including some high dimensional ones) show the effectiveness of ACPO on ACMDP RL problems, as well as its superior empirical performance vis-a-vis some current alternatives. A direction for future work is implementation of ACPO to fully exploit the parallelization potential.

Impact Statement

This paper not only advances the field of RL theory and algorithms but also introduces a practical and scalable algorithm (supported by theory) that is of utility in many fields including LLMs, Diffusion Models, and robotic control. Currently, algorithms such as PPO are adapted for use simply because of lack of alternatives.

References

Abounadi, J., Bertsekas, D., and Borkar, V. S. Learning algorithms for markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698, 2001.

Achiam, J. UC Berkeley CS 285 (Fall 2017), Advanced Policy Gradients, 2017. URL: http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_13_advanced_pg.pdf. Last visited on 2020/05/24.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31. JMLR. org, 2017.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Agnihotri, A., Saraf, P., and Bapnad, K. R. A convolutional neural network approach towards self-driving cars. In *2019 IEEE 16th India Council International Conference (INDICON)*, pp. 1–4, 2019. doi: 10.1109/INDICON47234.2019.9030307.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Altman, E. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Aractingi, M., Léziart, P.-A., Flayols, T., Perez, J., Silander, T., and Souères, P. Controlling the solo12 quadruped robot with deep reinforcement learning. *scientific Reports*, 13(1):11945, 2023.

Bhatnagar, S. and Lakshmanan, K. An Online Actor–Critic Algorithm with Function Approximation for Constrained Markov Decision Processes. https://doi.org/10.1007/s10957-012-9989-5, 2012. [Accessed 08-10-2023].

Borkar, V. S. A convex analytic approach to markov decision processes. *Probability Theory and Related Fields*, 78(4):583–602, 1988.

Calvo-Fullana, M., Paternain, S., Chamon, L. F., and Ribeiro, A. State augmented constrained reinforcement learning: Overcoming the limitations of learning with rewards. *IEEE Transactions on Automatic Control*, 2023.

Chen, L., Jain, R., and Luo, H. Learning infinite-horizon average-reward Markov decision process with constraints. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 3246–3270. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/chen22i.html.

Cho, G. E. and Meyer, C. D. Comparison of perturbation bounds for the stationary distribution of a markov chain. *Linear Algebra and its Applications*, 335(1-3):137–150, 2001.

Doyle, P. G. The kemeny constant of a markov chain. *arXiv preprint arXiv:0909.2636*, 2009.

Gao, Y., Xu, H., Lin, J., Yu, F., Levine, S., and Darrell, T. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

Garcia, J. and Fernandez, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Hordijk, A. and Kallenberg, L. Linear programming and markov decision chains. *Management Science*, 25(4):352–362, 1979.

Hu, H., Liu, Z., Chitlangia, S., Agnihotri, A., and Zhao, D. Investigating the impact of multi-lidar placement on object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2550–2559, June 2022.

Hunter, J. J. Stationary distributions and mean first passage times of perturbed markov chains. *Linear Algebra and its Applications*, 410:217–243, 2005.

Hunter, J. J. *Mathematical techniques of applied probability: Discrete time models: Basic theory*, volume 1. Academic Press, 2014.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, volume 2, pp. 267–274, 2002.

Kemeny, J. and Snell, I. *Finite Markov Chains*. Van Nostrand, New Jersey, 1960.

Levene, M. and Loizou, G. Kemeny's constant and the random surfer. *The American mathematical monthly*, 109(8):741–745, 2002.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Liao, P., Qi, Z., Wan, R., Klasnja, P., and Murphy, S. A. Batch policy learning in average reward markov decision processes. *The Annals of Statistics*, 50(6):3364–3387, 2022.

Ma, X., Tang, X., Xia, L., Yang, J., and Zhao, Q. Average-reward reinforcement learning with trust region methods. *arXiv preprint arXiv:2106.03442*, 2021.

Manne, A. S. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Moskovitz, T., Singh, A. K., Strouse, D., Sandholm, T., Salakhutdinov, R., Dragan, A. D., and McAleer, S. Confronting reward model overoptimization with constrained rlhf. *arXiv preprint arXiv:2310.04373*, 2023.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Satija, H., Amortila, P., and Pineau, J. Constrained Markov decision processes via backward value functions. 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. 8502–8511. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/satija20a.html.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *International Conference on Learning Representations (ICLR)*, 2016.

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

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., et al. V-mpo: on-policy maximum a posteriori policy optimization for discrete and continuous control. *International Conference on Learning Representations*, 2020.

Tessler, C., Mankowitz, D. J., and Mannor, S. Reward constrained policy optimization. *International Conference on Learning Representation (ICLR)*, 2019.

Tibshirani, R. J. Dykstra's algorithm, admm, and coordinate descent: Connections, insights, and extensions. *Advances in Neural Information Processing Systems*, 30, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Vinitsky, E., Kreidieh, A., Le Flem, L., Kheterpal, N., Jang, K., Wu, F., Liaw, R., Liang, E., and Bayen, A. M. Benchmarks for reinforcement learning in mixed-autonomy traffic. In *Proceedings of Conference on Robot Learning*, pp. 399–409, 2018.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Vuong, Q., Zhang, Y., and Ross, K. W. Supervised policy update for deep reinforcement learning. In *International Conference on Learning Representation (ICLR)*, 2019.

Wan, Y., Naik, A., and Sutton, R. S. Learning and planning in average-reward markov decision processes. In *International Conference on Machine Learning*, pp. 10653–10662. PMLR, 2021.

Wei, C.-Y., Jahromi, M. J., Luo, H., and Jain, R. Learning infinite-horizon average-reward mdps with linear function approximation. In *International Conference on Artificial Intelligence and Statistics*, pp. 3007–3015. PMLR, 2021.

Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems (NIPS)*, pp. 5285–5294, 2017.

Yang, T.-Y., Rosca, J., Narasimhan, K., and Ramadge, P. J. Projection-based constrained policy optimization. In *International Conference on Learning Representation (ICLR)*, 2020.

Zhang, S., Wan, Y., Sutton, R. S., and Whiteson, S. Average-reward off-policy policy evaluation with function approximation. *arXiv preprint arXiv:2101.02808*, 2021.

Zhang, Y. and Ross, K. Average reward reinforcement learning with monotonic policy improvement. *Preprint*, 2020.

A. Appendix

A.1. Proofs

Lemma A.1 (Trivialization of Discounted Criterion Bounds)**.** *Consider the policy performance bound of (Achiam et al., 2017), which says that for any two stationary policies π and π':*

$$J_\gamma(\pi') - J_\gamma(\pi) \geqslant \frac{1}{1-\gamma}\left[\mathop{\mathbb{E}}_{\substack{s\sim d_{\pi,\gamma}\\ a\sim\pi'}}[A^\pi_\gamma(s,a)] - \frac{2\gamma\epsilon^\gamma}{1-\gamma}\mathop{\mathbb{E}}_{s\sim d_{\pi,\gamma}} D_{TV}(\pi' \parallel \pi)[s]\right] \tag{17}$$

where $\epsilon^\gamma = \max_s \left|\mathop{\mathbb{E}}_{a\sim\pi'}[A^\pi_\gamma(s,a)]\right|$. Then, the right hand side times $1-\gamma$ goes to negative infinity as $\gamma \to 1$.

Proof. Since $d_{\pi,\gamma}$ approaches the stationary distribution d_π as $\gamma \to 1$, we can multiply the right hand side of (17) by $(1-\gamma)$ and take the limit which gives us:

$$\begin{aligned}
&\lim_{\gamma\to 1}\left(\mathop{\mathbb{E}}_{\substack{s\sim d_{\pi,\gamma}\\ a\sim\pi'}}[A^\pi_\gamma(s,a)] \pm \frac{2\gamma\epsilon^\gamma}{1-\gamma}\mathop{\mathbb{E}}_{s\sim d_{\pi,\gamma}} D_{\text{TV}}(\pi' \parallel \pi)[s]\right)\\
&= \mathop{\mathbb{E}}_{\substack{s\sim d_\pi\\ a\sim\pi'}}[\bar{A}^\pi(s,a)] - 2\epsilon \mathop{\mathbb{E}}_{s\sim d_\pi}\left[D_{\text{TV}}(\pi' \parallel \pi)[s]\right]\cdot \lim_{\gamma\to 1}\frac{\gamma}{1-\gamma}\\
&= -\infty
\end{aligned}$$

Here $\epsilon = \max_s \left|\mathop{\mathbb{E}}_{a\sim\pi'}[\bar{A}^\pi(s,a)]\right|$. The first equality is a standard result of $\lim_{\gamma\to 1} A^\pi_\gamma(s,a) = \bar{A}^\pi(s,a)$. □

Lemma A.2. *(Zhang & Ross, 2020) Under the unichain assumption of the underlying Markov chain, for any stochastic policies π and π':*

$$J(\pi') - J(\pi) = \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi'}\\ a\sim\pi'}}\left[\bar{A}^\pi(s,a)\right]. \tag{4}$$

Proof. We directly expand the right-hand side using the definition of the advantage function and Bellman equation, which gives us:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\substack{s\sim d_{\pi'}\\ a\sim\pi'}}\left[\bar{A}^\pi(s,a)\right] &= \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi'}\\ a\sim\pi'}}\left[\bar{Q}^\pi(s,a) - \bar{V}^\pi(s)\right]\\
&= \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi'}\\ a\sim\pi'\\ s'\sim P(\cdot|s,a)}}\left[r(s,a,s') - J(\pi) + \bar{V}^\pi(s') - \bar{V}^\pi(s)\right]\\
&= J(\pi') - J(\pi) + \underbrace{\mathop{\mathbb{E}}_{\substack{s\sim d_{\pi'}\\ a\sim\pi'\\ s'\sim P(\cdot|s,a)}}[\bar{V}^\pi(s')] - \mathop{\mathbb{E}}_{s\sim d_{\pi'}}[\bar{V}^\pi(s)]}_{A}
\end{aligned}$$

Analyzing A, since $d_{\pi'}(s)$ is a stationary distribution:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\substack{s\sim d_{\pi'}\\ a\sim\pi'\\ s'\sim P(\cdot|s,a)}}[\bar{V}^\pi(s')] &= \sum_s d_{\pi'}(s)\sum_a \pi'(a|s)\sum_{s'} P(s'|s,a)\bar{V}^\pi(s')\\
&= \sum_s d_{\pi'}(s)\sum_{s'} P_{\pi'}(s'|s)\bar{V}^\pi(s') = \sum_{s'} d_{\pi'}(s')\bar{V}^\pi(s')
\end{aligned}$$

Therefore, $A = \sum_{s'} d_{\pi'}(s')\bar{V}^\pi(s') - \mathop{\mathbb{E}}_{s\sim d_{\pi'}}[\bar{V}^\pi(s)] = 0$. Hence, proved. □

Lemma A.3. *Under the unichain assumption, for any stochastic policies π and π' we have:*

$$\left| J(\pi') - J(\pi) - \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\pi'}} \left[\bar{A}^\pi(s,a)\right] \right| \leqslant 2\epsilon D_{TV}(d_{\pi'} \parallel d_\pi) \tag{5}$$

Proof.

$$\begin{aligned}
\left| J(\pi') - J(\pi) - \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\pi'}} \left[\bar{A}^\pi(s,a)\right] \right| &= \left| \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi'} \\ a\sim\pi'}} \left[\bar{A}^\pi(s,a)\right] - \mathop{\mathbb{E}}_{\substack{s\sim d_\pi \\ a\sim\pi'}} \left[\bar{A}^\pi(s,a)\right] \right| && \text{(from Lemma 3.1)} \\
&= \left| \sum_s \mathop{\mathbb{E}}_{a\sim\pi'} \left[\bar{A}^\pi(s,a)\right] (d_{\pi'}(s) - d_\pi(s)) \right| \\
&\leqslant \sum_s \left| \mathop{\mathbb{E}}_{a\sim\pi'} \left[\bar{A}^\pi(s,a)\right] (d_{\pi'}(s) - d_\pi(s)) \right| \\
&\leqslant \max_s \left| \mathop{\mathbb{E}}_{a\sim\pi'} \left[\bar{A}^\pi(s,a)\right] \right| \|d_{\pi'} - d_\pi\|_1 && \text{(Holder's inequality)} \\
&= 2\epsilon D_{\text{TV}}(d_{\pi'} \parallel d_\pi)
\end{aligned}$$

□

Lemma A.4. *Under the ergodic assumption, the divergence between the stationary distributions d_π and $d_{\pi'}$ is upper bounded as:*

$$D_{TV}(d_{\pi'} \parallel d_\pi) \leqslant \sigma^\star \mathop{\mathbb{E}}_{s\sim d_\pi} [D_{TV}(\pi' \parallel \pi)[s]], \tag{6}$$

Proof. This proof takes ideas from Markov chain perturbation theory in (Cho & Meyer, 2001; Hunter, 2005; Zhang & Ross, 2020). Firstly we state a standard result with $P^\star_\pi = \mathbf{1} d_\pi^T$

$$(d_{\pi'} - d_\pi)^T (I - P_{\pi'} + P^\star_{\pi'}) = d_{\pi'}^T - d_\pi^T - d_{\pi'}^T + d_\pi^T P_{\pi'} = d_\pi^T P_{\pi'} - d_\pi^T = d_\pi^T (P_{\pi'} - P_\pi).$$

Denoting the fundamental matrix of the Markov chain $Z^{\pi'} = (I - P_{\pi'} + P^\star_{\pi'})^{-1}$ and the mean first passage time matrix $M^{\pi'} = (I - Z^{\pi'} + E Z^{\pi'}_{\text{dg}}) D^{\pi'}$, and right multiplying the above by $(Z^{\pi'})^{-1}$ we have,

$$\begin{aligned}
d_{\pi'}^T - d_\pi^T = d_\pi^T (P_{\pi'} - P_\pi)(I - M^{\pi'}(D^{\pi'})^{-1}) \;\Rightarrow\; d_{\pi'} - d_\pi &= (I - M^{\pi'}(D^{\pi'})^{-1})^T (P_{\pi'}^T - P_\pi^T) d_\pi && (18) \\
\text{i.e.} \qquad \|d_{\pi'} - d_\pi\|_1 &\leqslant \left\| (I - M^{\pi'}(D^{\pi'})^{-1})^T (P_{\pi'}^T - P_\pi^T) d_\pi \right\|_1 && \text{(submultiplicative property)} \\
\|d_{\pi'} - d_\pi\|_1 &\leqslant \underbrace{\left\| (I - M^{\pi'}(D^{\pi'})^{-1}) \right\|_\infty}_{T_1} \underbrace{\left\| (P_{\pi'}^T - P_\pi^T) d_\pi \right\|_1}_{T_2} && \text{(Holder's inequality)}
\end{aligned}$$

We know that $\kappa^\pi = \text{Tr}(Z^\pi)$ and from (Hunter, 2014), we can write T_1 using the eigenvalues $\{\lambda_{\pi,i}\}_{i=1}^{|S|}$ of the underlying P_π as

$$T_1 \leqslant \frac{1}{|S|} \sum_{i=2}^{|S|} \frac{1}{(1-\lambda_{\pi,i})^{1/2}} \leqslant \max_i (1-\lambda_{\pi,i})^{-1/2} = \sigma^\pi \leqslant \max_\pi \sigma^\pi = \sigma^\star.$$

For T_2, we refer to the result by (Zhang & Ross, 2020), and provide the proof for completeness below.

$$
\begin{aligned}
T_2 &= \sum_{s'} \left| \sum_{s} \left(\sum_{a} P(s'|s,a)\pi'(a|s) - P(s'|s,a)\pi(a|s) \right) d_\pi(s) \right| \\
&\leqslant \sum_{s',s} \left| \sum_{a} P(s'|s,a)(\pi'(a|s) - \pi(a|s)) \right| d_\pi(s) \\
&\leqslant \sum_{s,s',a} P(s'|s,a) \left| \pi'(a|s) - \pi(a|s) \right| d_\pi(s) \\
&\leqslant \sum_{s,a} \left| \pi'(a|s) - \pi(a|s) \right| d_\pi(s) = 2 \mathop{\mathbb{E}}_{s\sim d_\pi} [D_{\text{TV}}(\pi' \parallel \pi)[s]]
\end{aligned}
$$

Combining these inequalities of T_1 and T_2, we get the desired result. □

A.2. Performance and Constraint Bounds of Trust Region Approach

Consider the trust region formulation in Equation (10). To prove the policy performance bound when the current policy is infeasible (i.e., constraint-violating), we prove the KL divergence between π_k and π_{k+1} for the KL divergence projection, along with other lemmas. We then prove our main theorem for the worst-case performance degradation.

Lemma A.5. *For a closed convex constraint set, if we have a constraint satisfying policy π_k and the KL divergence $\mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{KL} \left(\pi_{k+1/2} \| \pi_k \right) [s] \right]$ of the 'Improve' step is upper bounded by step size δ, then after KL divergence projection of the 'Project' step we have*

$$
\mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{KL} \left(\pi_{k+1} \| \pi_k \right) [s] \right] \leqslant \delta.
$$

Proof. We make use of the fact that Bregman divergence (hence, KL divergence) projection onto the constraint set ($\in \mathbb{R}^d$, $d \in \mathbb{N}$) exists and is unique. Since π_k is safe, we have π_k in the constraint set, and π_{k+1} is the projection of $\pi_{k+\frac{1}{2}}$ onto the constraint set. Using the projection inequality, we have

$$
\begin{aligned}
\mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{\text{KL}} \left(\pi_k \| \pi_{k+1} \right) [s] \right] + \mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{\text{KL}} \left(\pi_{k+1} \middle\| \pi_{k+\frac{1}{2}} \right) [s] \right] &\leqslant \mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{\text{KL}} \left(\pi_k \middle\| \pi_{k+\frac{1}{2}} \right) [s] \right] \\
\Rightarrow \mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{\text{KL}} \left(\pi_k \| \pi_{k+1} \right) [s] \right] &\leqslant \mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{\text{KL}} \left(\pi_k \middle\| \pi_{k+\frac{1}{2}} \right) [s] \right] \leqslant \delta. \qquad (D_{\text{KL}} (\cdot \| \cdot) \geqslant 0)
\end{aligned}
$$

Since KL divergence is asymptotically symmetric when updating the policy within a local neighbourhood ($\delta << 1$), we have

$$
\mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{\text{KL}} \left(\pi_{k+1} \| \pi_k \right) [s] \right] \leqslant \mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{\text{KL}} \left(\pi_{k+\frac{1}{2}} \middle\| \pi_k \right) [s] \right] \leqslant \delta.
$$

□

Lemma A.6. *For a closed convex constraint set, if we have a constraint violating policy π_k and the KL divergence $\mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{KL} \left(\pi_{k+1/2} \| \pi_k \right) [s] \right]$ of the first step is upper bounded by step size δ, then after KL divergence projection of the second step we have*

$$
\mathop{\mathbb{E}}_{s\sim d_{\pi_k}} \left[D_{KL} \left(\pi_{k+1} \| \pi_k \right) [s] \right] \leqslant \delta + V_{max},
$$

where $V_{max} = \max_i \alpha_i \beta_i^2$, $\beta_i = [J_{C_i}(\pi_k) - l_i]_+$, $\alpha_i = \frac{1}{2a_i^T H^{-1} a_i}$, with a_i as the gradient of the cost advantage function corresponding to constraint C_i, and H as the Hessian of the KL divergence constraint. [2].

Proof. Let the sublevel set of cost constraint function for the current infeasible policy π_k be given as:

$$
L_{\pi_k} = \{\pi \mid J_{C_i}(\pi) + \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi_k} \\ a\sim\pi}} [\bar{A}_{C_i}^{\pi_k}(s,a)] \leqslant J_{C_i}(\pi_k) \ \ \forall\ i\}.
$$

[2]For any $x \in \mathbb{R}$, $[x]_+ := \max(0, x)$

This implies that the current policy π_k lies in L_{π_k}. The constraint set onto which $\pi_{k+\frac{1}{2}}$ is projected onto is given by: $\{\pi \mid J_{C_i}(\pi_k) + \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi_k}\\ a\sim\pi}}[\bar{A}^{\pi_k}_{C_i}(s,a)] \leqslant l_i \;\; \forall\, i\}$. Let π^L_{k+1} be the projection of $\pi_{k+\frac{1}{2}}$ onto L_{π_k}.

Note that the Bregman inequality of Lemma A.5 holds for any convex set in $\mathbb{R}^d, d \in \mathbb{N}$. This implies $\mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[D_{\text{KL}}\left(\pi^L_{k+1}\|\pi_k\right)[s]\right] \leqslant \delta$ since π_k and π^L_{k+1} are both in L_{π_k}, which is also convex since the constraint functions are convex. Using the Three-point Lemma [3], for polices π_k, π_{k+1}, and π^L_{k+1}, with $\varphi(\mathbf{x}) := \sum_i x_i \log x_i$, we have

$$
\begin{aligned}
\delta \geqslant \mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[D_{\text{KL}}\left(\pi^L_{k+1}\|\pi_k\right))[s]\right] &= \mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[D_{\text{KL}}\left(\pi_{k+1}\|\pi_k\right)[s]\right] \\
&\quad - \mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[D_{\text{KL}}\left(\pi_{k+1}\|\pi^L_{k+1}\right)[s]\right] \\
&\quad + \mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[(\nabla\varphi(\pi_k) - \nabla\varphi(\pi^L_{k+1}))^T(\pi_{k+1} - \pi^L_{k+1})[s]\right] \\
\Rightarrow \mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[D_{\text{KL}}\left(\pi_{k+1}\|\pi_k\right)[s]\right] &\leqslant \delta + \underbrace{\mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[D_{\text{KL}}\left(\pi_{k+1}\|\pi^L_{k+1}\right)[s]\right]}_{T_1} \\
&\quad - \underbrace{\mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[(\nabla\varphi(\pi_k) - \nabla\varphi(\pi^L_{k+1}))^T(\pi_{k+1} - \pi^L_{k+1})[s]\right]}_{T_2}.
\end{aligned} \tag{19}
$$

If the constraint violations of the current policy π_k are small, i.e., $J_{C_i}(\pi_k) - l_i = b_i$ is small for all i, then T_1 can be approximated by a second order expansion. We analyze T_1 for any constraint C_i and then bound it over all the constraints. As before we overload the notation with $\pi_k = \pi_{\theta_k} = \theta_k$ to write. For any constraint C_i, we can write T^i_1 as the expected KL divergence if projection was onto the constraint set of C_i i.e.

$$
\begin{aligned}
T^i_1 \approx \frac{1}{2}(\pi_{k+1} - \pi^L_{k+1})^T H(\pi_{k+1} - \pi^L_{k+1}) &= \frac{1}{2}\Big(\frac{\beta_i}{a_i^T H^{-1} a_i} H^{-1} a_i\Big)^T H \Big(\frac{\beta_i}{a_i^T H^{-1} a_i} H^{-1} a_i\Big) \\
&= \frac{\beta_i^2}{2 a_i^T H^{-1} a_i} = \alpha_i \beta_i^2,
\end{aligned}
$$

where the second equality is a result of the trust region guarantee (see (Schulman et al., 2015) for more details). Finally we invoke the projection result from (Achiam, 2017) which uses Dykstra's Alternating Projection algorithm from (Tibshirani, 2017) to bound this projection, i.e., $T_1 \leqslant \max_i T^i_1 \approx \max_i \alpha_i \beta_i^2$.

And since δ is small, we have $\nabla\varphi(\pi_k) - \nabla\varphi(\pi^L_{k+1}) \approx 0$ given s. Thus, $T_2 \approx 0$. Combining all of the above, we have $\mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[D_{\text{KL}}\left(\pi_{k+1}\|\pi_k\right)[s]\right] \leqslant \delta + V_{max}$. □

Theorem 3.6. *Let $\pi_{\theta_{k+1}}$ be the optimal solution to Eq. (10) for some $\pi_{\theta_k} \in \Pi_\Theta$. Then, we have*

$$
J(\pi_{\theta_{k+1}}) - J(\pi_{\theta_k}) \geqslant -\sqrt{2(\delta + V_{max})}\nu^{\pi_{\theta_{k+1}}} \tag{11}
$$

$$
\textit{and} \;\; J_{C_i}(\pi_{\theta_{k+1}}) \leqslant l_i + \sqrt{2(\delta + V_{max})}\nu^{\pi_{\theta_{k+1}}}_{C_i} \;\; \forall\, i, \tag{12}
$$

where $\nu^{\pi_{\theta_{k+1}}} = \sigma^{\pi_{\theta_{k+1}}} \max_s \Big|\mathop{\mathbb{E}}_{a\sim\pi_{\theta_{k+1}}}[\bar{A}^{\pi_{\theta_k}}(s,a)]\Big|$, $\nu^{\pi_{\theta_{k+1}}}_{C_i} = \sigma^{\pi_{\theta_{k+1}}} \max_{i,s} \Big|\mathop{\mathbb{E}}_{a\sim\pi_{\theta_{k+1}}}[\bar{A}^{\pi_{\theta_k}}_{C_i}(s,a)]\Big|$, $V_{max} = \max_i \beta_i^2$, *and* $\beta_i = [J_{C_i}(\pi_{\theta_k}) - l_i]_+$.

Proof. Since $\bar{D}_{\text{KL}}(\pi_{\theta_k} \parallel \pi_{\theta_k}) = 0$, π_{θ_k} is feasible. The objective value is 0 for $\pi_\theta = \pi_{\theta_k}$. The bound follows from Equation (7) and Equation (9) where the average KL i.e. $\mathop{\mathbb{E}}_{s\sim d_{\pi_k}}\left[D_{\text{KL}}\left(\pi_{k+1}\|\pi_k\right)[s]\right]$ is bounded by $\delta + V_{max}$ from Lemma A.6.

Similar to Corollary 3.5, expressions for the auxiliary cost constraints also follow immediately as the second result.

[3] For any ϕ, the Bregman divergence identity: $D_\phi(x,y) + D_\phi(y,z) = D_\phi(x,z) + < \nabla\phi(z) - \nabla\phi(y), x - y >$

Remark A.7. Remark If we look at proof as given by (Zhang & Ross, 2020) in Section 5 of their paper, with the distinction now that δ is replaced by $\delta + V_{max}$, we have the same result. Our worse bound is due to the constrained nature of our setting, which is intuitive in the sense that for the sake of satisfying constraints, we undergo a worse worst-case performance degradation.

□

A.3. Approximate ACPO

A.3.1. Policy Recovery Routine

As described in Section 4.2, we need a recovery routine in case the updated policy $\pi_{k+1/2}$ is not approximate constraint satisfying. In this case, the optimization problem is inspired from a simple trust region approach by (Schulman et al., 2015). Since we only deal with one constraint in the practical implementation of ACPO, the recovery rule is obtained by solving the following problem:

$$\begin{array}{ll} \min_\theta & c + a^T(\theta - \theta_k) \\ \text{s.t.} & \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leqslant \delta. \end{array}$$

Let $x = \theta - \theta_k$, then the dual function $L(x, \lambda)$ is given by: $L(x, \lambda) = c + a^T x + \lambda\left(\frac{1}{2}x^T H x - \delta\right)$. Now,

$$\frac{L}{\partial x} = a + \lambda(Hx) = 0 \Longrightarrow x = -\frac{1}{\lambda}H^{-1}a.$$

x obtained above should satisfy the trust-region constraint:

$$\begin{aligned} \frac{1}{2}\left(-\frac{1}{\lambda}H^{-1}a\right)^T H\left(-\frac{1}{\lambda}H^{-1}a\right) &\leqslant \delta \\ \Longrightarrow \quad \frac{1}{2}\cdot\frac{1}{\lambda^2}\cdot a^T H^{-1}a &\leqslant \delta \\ \Longrightarrow \sqrt{\frac{a^T H^{-1}a}{2\delta}} &\leqslant \lambda. \end{aligned}$$

Therefore, the update rule in case of infeasibility takes the form $\theta = \theta_k - \sqrt{\frac{2\delta}{a^T H^{-1}a}}H^{-1}a$. We augment this rule with the gradient of the reward advantage function as well, so the final recovery is

$$\theta_{k+1/2} = \theta_k - \sqrt{2\delta}\left[t\cdot\frac{H^{-1}a}{\sqrt{a^T H^{-1}a}} + (1-t)\cdot\frac{H^{-1}g}{\sqrt{g^T H^{-1}g}}\right] \quad ; \quad t \in [0,1]$$

A.3.2. Line Search

Because of approximation error, the proposed update may not satisfy the constraints in Eq. (10). Constraint satisfaction is enforced via line search, so the final update is

$$\theta_{k+1} = \theta_k + s^j\left(\theta_{k+1/2} - \theta_k\right),$$

where $s \in (0,1)$ is the backtracking coefficient and $j \in \{0, ..., L\}$ is the smallest integer for which π_{k+1} satisfies the constraints in Equation 10. Here, L is a finite backtracking budget; if no proposed policy satisfies the constraints after L backtracking steps, no update occurs.

A.4. Practical ACPO

As explained in Section 4, we use the below problem formulation, which uses first-order Taylor approximation on the objective and second-order approximation on the KL constraint [4] around θ_k, given small δ:

$$\begin{aligned} \max_{\theta} \quad & g^T(\theta - \theta_k) \\ \text{s.t.} \quad & c_i + a_i^T(\theta - \theta_k) \leqslant 0, \;\; \forall\, i \qquad ; \qquad \tfrac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leqslant \delta. \end{aligned} \tag{20}$$

where

$$g := \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi_{\theta_k}} \\ a\sim\pi_{\theta_k}}} \left[\nabla_\theta \log \pi_\theta(a|s)|_{\theta=\theta_k} \bar{A}^{\pi_{\theta_k}}(s,a)\right] \qquad ; \qquad c_i := J_{C_i}(\theta_k) - l_i \;\; \forall\, i$$

$$a_i := \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi_{\theta_k}} \\ a\sim\pi_{\theta_k}}} \left[\nabla_\theta \log \pi_\theta(a|s)|_{\theta=\theta_k} \bar{A}_{C_i}^{\pi_{\theta_k}}(s,a)\right] \qquad ; \qquad H := \mathop{\mathbb{E}}_{\substack{s\sim d_{\pi_{\theta_k}} \\ a\sim\pi_{\theta_k}}} \left[\nabla_\theta \log \pi_\theta(a|s)|_{\theta=\theta_k} \nabla_\theta \log \pi_\theta(a|s)|_{\theta=\theta_k}^T\right]$$

Similar to the work of (Achiam et al., 2017), g, a_i, and H can be approximated using samples drawn from the policy π_{θ_k}. The Hessian H is identical to the Hessian H used by (Achiam et al., 2017) and (Zhang & Ross, 2020). However, the definitons g and a_i's are different since they include the average reward advantage functions, $\bar{A}^{\pi_{\theta_k}}(s,a) = \bar{Q}^{\pi_{\theta_k}}(s,a) - \bar{V}^{\pi_{\theta_k}}(s)$.

Since rewards and cost advantage functions can be approximated independently, we use the framework of (Zhang & Ross, 2020) to do so. We describe the process of estimation of rewards advantage function, and the same procedure can be used for the cost advantage functions as well. Specifically, first approximate the average-reward bias $\bar{V}^{\pi_{\theta_k}}(s)$ and then use a one-step TD backup to estimate the action-bias function. Concretely, using the average reward Bellman equation gives

$$\bar{A}^{\pi_{\theta_k}}(s,a) = r(s,a) - J(\pi_{\theta_k}) + \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)} \left[\bar{V}^{\pi_{\theta_k}}(s')\right] - \bar{V}^{\pi_{\theta_k}}(s) \tag{21}$$

This expression involves the average-reward bias $\bar{V}^{\pi_{\theta_k}}(s)$, which we can approximated using the standard critic network $\bar{V}_{\phi_k}(s)$. However, in practice we use the average-reward version of the Generalized Advantage Estimator (GAE) from (Schulman et al., 2016), similar to (Zhang & Ross, 2020). Hence, we refer the reader to that paper for detailed explanation, but provide an overview below for completeness.

Let $\widehat{J}_\pi = \frac{1}{N}\sum_{t=1}^{N} r_t$ denote the estimated average reward. The Monte Carlo target for the average reward value function is $\bar{V}_t^{\text{target}} = \sum_{t'=t}^{N}(r_t - \widehat{J}_\pi)$ and the bootstrapped target is $\bar{V}_t^{\text{target}} = r_t - \widehat{J}_\pi + \bar{V}_\phi^\pi(s_{t+1})$.

The Monte Carlo and Bootstrap estimators for the average reward advantage function are:

$$\widehat{A}_{\text{MC}}^\pi(s_t,a_t) = \sum_{t'=t}^{N}(r_t - \widehat{J}_\pi) - \bar{V}_\phi^\pi(s_t) \qquad ; \qquad \widehat{A}_{\text{BS}}^\pi(s_t,a_t) = r_{i,t} - \widehat{J}_\pi + \bar{V}_\phi^\pi(s_{t+1}) - \bar{V}_\phi^\pi(s_t)$$

We can similarly extend the GAE to the average reward setting:

$$\widehat{A}_{\text{GAE}}(s_t,a_t) = \sum_{t'=t}^{N} \lambda^{t'-t}\delta_{t'} \qquad , \qquad \delta_{t'} = r_{t'} - \widehat{J}_\pi + \bar{V}_\phi^\pi(s_{t'+1}) - \bar{V}_\phi^\pi(s_{t'}). \tag{22}$$

and set the target for the value function to $\bar{V}_t^{\text{target}} = r_t - \widehat{J}_\pi + \bar{V}_\phi^\pi(s_{t+1}) + \sum_{t'=t+1}^{N} \lambda^{t'-t}\delta_{t'}$.

A.5. Experimental Details

For detailed explanation of Point-Circle, Point-Gather, Ant-Circle, and Ant-Gather tasks, please refer to (Achiam et al., 2017). For detailed explanation of Bottleneck and Grid tasks, please refer to (Vinitsky et al., 2018). For the simulations in the Gather and Circle tasks, we use neural network baselines with the same architecture and activation functions as the policy networks. For the simulations in the Grid and Bottleneck tasks, we use linear baselines. For all experiments we use

[4] The gradient and first-order Taylor approximation of $\bar{D}_{\text{KL}}(\pi_\theta \parallel \pi_{\theta_k})$ at $\theta = \theta_k$ is zero.

Gaussian neural policies whose outputs are the mean vectors and variances are separate parameters to be learned. Seeds used for generating evaluation trajectories are different from those used for training.

For comparison of different algorithms, we make use of CPO, PCPO, ATRPO, and PPO implementations taken from https://github.com/rll/rllab and https://github.com/openai/safety-starter-agents. Even the hyperparameters are selected so as to showcase the best performance of other algorithms for fair comparison. The choice of the hyperparameters given below is inspired by the original papers since we wanted to understand the performance of the average reward case.

We use settings which are common in all open-source implementations of the algorithms, such as normalizing the states by the running mean and standard deviation before being fed into the neural network and similarly normalizing the advantage values (for both rewards and constraints) by their batch means and standard deviations before being used for policy updates. Table 1 summarizes the hyperparameters below.

Table 1. Hyperparameter Setup

Hyperparameter	PPO/ATRPO	CPO/PCPO/ACPO
No. of hidden layers	2	2
Activation	tanh	tanh
Initial log std	-0.5	-1
Batch size	2500	2500
GAE parameter (reward)	0.95	0.95
GAE parameter (cost)	N/A	0.95
Trust region step size δ	10^{-4}	10^{-4}
Learning rate for policy	2×10^{-4}	2×10^{-4}
Learning rate for reward critic net	2×10^{-4}	2×10^{-4}
Learning rate for cost critic net	N/A	2×10^{-4}
Backtracking coeff.	0.75	0.75
Max backtracking iterations	10	10
Max conjugate gradient iterations	10	10
Recovery regime parameter t	N/A	0.75

For the Lagrangian formulation of ATRPO and PPO, note that the original papers do not provide any blueprint for formulating the Lagrangian, and even CPO and PCPO use *unconstrained* TRPO for benchmarking. However, we feel that this is unfair to these algorithms as they can possibly perform better with a Lagrangian formulation in an average-reward CMDP setting. To this extent, we introduced a Lagrangian parameter $\ell \in [0, 1]$ that balances the rewards and constraints in the final objective function. More specifically, Equation (13) for a single constraint now becomes

$$\max_{\theta} \quad (1-\ell)g^T(\theta-\theta_k) - \ell\big[\big(c_1 + a_1^T(\theta-\theta_k)\big) + \big(\tfrac{1}{2}(\theta-\theta_k)^T H(\theta-\theta_k) - \delta\big)\big]. \tag{23}$$

Note. The authors of the ATRPO and PPO do not suggest any principled approach for finding an optimal ℓ. Hence, the choice of the Lagrangian parameter ℓ is completely empirical and is selected such that these algorithms achieve maximum rewards while satisfying the constraints. Also see in Figure 1, for Ant-Gather, Bottleneck, and Grid environments, where the constraints cannot be satisfied for *any* value of ℓ, we include the results for a specific value of ℓ for illustrative purposes, as detailed in Table 2.

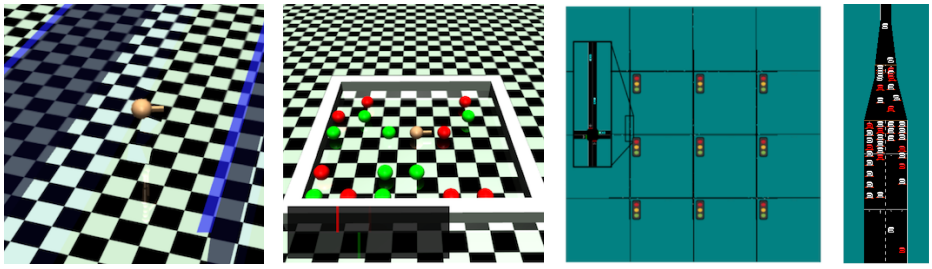
Table 2. Lagrangian parameter ℓ for ATRPO and PPO

Algorithm	Point-Gather	Ant-Circle	Ant-Gather	Bottleneck	Grid
ATRPO	0.50	0.60	0.45	0.50	0.45
PPO	0.55	0.50	0.50	0.50	0.60

A.6. Experimental Addendum

A.6.1. Environments

All environments tested on are illustrated in Figure 3, along with a detailed description of each.



(a) Circle (b) Gather (c) Grid (d) Bottleneck

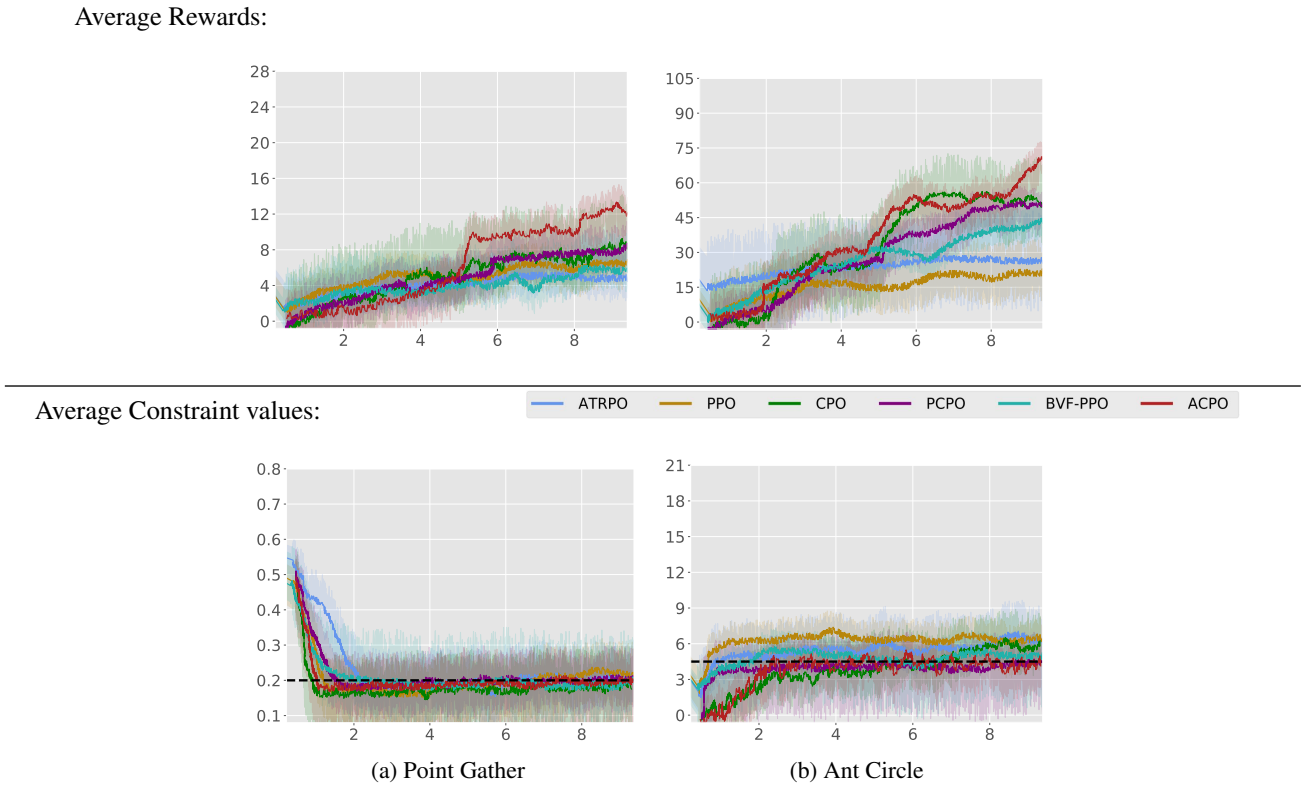
Figure 3. The Circle, Gather, Grid, and Bottleneck tasks. (a) Circle: The agent is rewarded for moving in a specified circle but is penalized if the diameter of the circle is larger than some value as in (Achiam et al., 2017). (b) Gather: The agent is rewarded for collecting the green balls while penalized to gather red balls as in (Achiam et al., 2017). (c) Grid: The agent controls traffic lights in a 3x3 road network and is rewarded for high traffic throughput but is constrained to let lights be red for at most 5 consecutive seconds as in (Vinitsky et al., 2018). (d) Botteneck: The agent controls vehicles (red) in a merging traffic situation and is rewarded for maximizing the number of vehicles that pass through but is constrained to ensure that white vehicles (not controlled by agent) have "low" speed for no more than 10 seconds as in (Vinitsky et al., 2018).

A.6.2. Learning Curves

Due to space restrictions, we present the learning curves for the remaining environments in Figure 4.

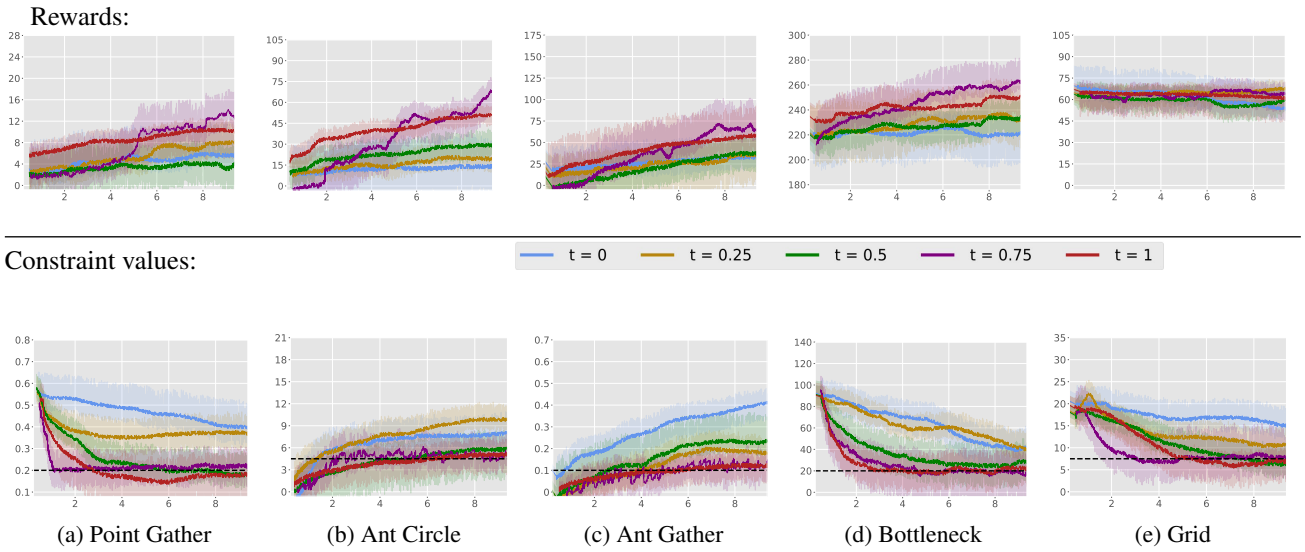
A.6.3. Recovery Regime Revisited

In Subsection 5.3, we studied the effect of the hyperparameter t for only one task. Figure 5 shows the performance of ACPO with different values of t in various environments.



(a) Point Gather (b) Ant Circle

Figure 4. The average reward and constraint cost function values vs iterations (in 10^4) learning curves for some algorithm-task pairs. Solid lines in each figure are the empirical means, while the shaded area represents 1 standard deviation, all over 5 runs. The dashed line in constraint plots is the constraint threshold l. ATRPO and PPO are tested with constraints, which are included in their Lagrangian formulation.



(a) Point Gather (b) Ant Circle (c) Ant Gather (d) Bottleneck (e) Grid

Figure 5. Comparison of performance of ACPO with different values of the hyperparameter t in various environment. X-axis is iterations in 10^4.