Feasible Constraint Policy Optimization for Safe Reinforcement Learning

Luoyang Sun*
Institute of Automation, CAS
School of Artificial Intelligence, UCAS
Beijing, China
sunluoyang2022@ia.ac.cn

Jiwen Jiang*
Institute of Automation, CAS
Beijing, China
Nanjing Artificial Intelligence Research of IA
Nanjing, China
University of Chinese Academy of Sciences, Nanjing
Nanjing, China
jiangjiwen2023@ia.ac.cn

Ning Yang†
Institute of Automation, CAS
School of Artificial Intelligence, UCAS
Beijing, China
ning.yang@ia.ac.cn

Rasul Tutunov
Huawei R&D
London, United Kingdom
rasul.tutunov@huawei.com

Haifeng Zhang†
Institute of Automation, CAS
School of Artificial Intelligence, UCAS
Beijing, China
haifeng.zhang@ia.ac.cn

Jun Wang
University College London
London, United Kingdom
jun.wang@cs.ucl.ac.uk

ABSTRACT

Safe reinforcement learning (RL) ensures that policies satisfy explicit constraints in safety-critical applications. However, existing primal-dual methods suffer from training instability. Trust region-based approaches often produce infeasible policies during training due to initialization and approximation errors. We introduce Feasible Constraint Policy Optimization (FCPO), which seamlessly combines penalty and trust region methods to address policy feasibility while ensuring stability and performance. FCPO efficiently decomposes optimization problems with the Alternating Direction Multiplier Method (ADMM), enabling efficient optimization through the utilization of first-order degree information. Comprehensive experiments showcase FCPO's consistent superiority, outperforming the baselines in both performance and constraint satisfaction across the majority of tasks.

KEYWORDS

Reinforcement Learning, Constrained Markov Decision Processes, Constrained optimization

ACM Reference Format:
Luoyang Sun, Jiwen Jiang, Ning Yang, Rasul Tutunov, Haifeng Zhang, and Jun Wang. 2026. Feasible Constraint Policy Optimization for Safe Reinforcement Learning. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/VKKT4028

*Equal contribution.
†Corresponding to Ning Yang ⟨ning.yang@ia.ac.cn⟩ and Haifeng Zhang ⟨haifeng.zhang@ia.ac.cn⟩

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/VKKT4028

1 INTRODUCTION

The combination of deep learning [11] and reinforcement learning (RL) [22] has enabled significant breakthroughs in diverse domains such as playing Atari games [13, 26], Go [18, 19], StarCraft [23], robotics [8, 16, 17, 20] and recommendations [2]. Nevertheless, the exploration nature of conventional RL methods has hindered their seamless application to real-world challenges [4]. This limitation becomes apparent when addressing critical concerns, like safeguarding robots from damage and ensuring human safety during their practical deployment. As a result, the emerging research avenue of safe RL has gained prominence, driven by the practical requirements.

A common approach to address these safety concerns is to model problems as Constrained Markov Decision Processes (CMDPs) [3], which convert safety requirements into explicit policy constraints. This formulation introduces a challenge: agents cannot freely explore their environment, making the learning problem more difficult than standard RL. Over the past few years, many solutions have been proposed. Based on the primal-dual method, [24] exploited Lagrangian functions to recast constrained conundrums into unconstrained optimization objectives. These frameworks leverage classical RL methodologies to find the optimal policy. While computationally efficient and conceptually straightforward, they suffer from convergence oscillations and instability issues.

Building on the foundations of Trust Region Policy Optimization (TRPO) [16], algorithms such as Constrained Policy Optimization (CPO) [1] and First Order Constrained Optimization in Policy Space (FOCOPS) [32] retained the advantages of optimization stability and high-performance outcomes. However, it is crucial to note that the optimization problem constructed by this method is solvable only when an intersection exists between the trust and the feasible region, as illustrated in Figure 1.

On the contrary, the penalty function method offers a more succinct approach, converting constrained optimization problems

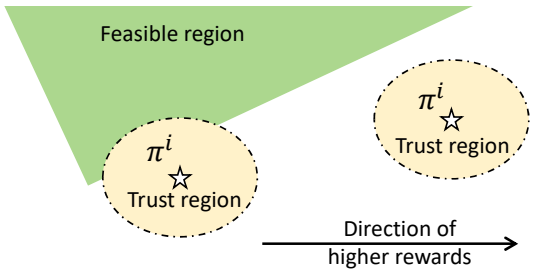

Figure 1: Constrained local policy search

into unconstrained ones, exemplified by IPO [12] and P3O [31]. However, the use of penalty functions can significantly affect the quality and stability of the solution. Previous work comes with its own set of advantages and disadvantages. Our constrained policy optimization algorithm achieves three key properties: (1) stable convergence, (2) performance guarantees, and (3) policy feasibility throughout training."

To bridge the gap, this paper seamlessly integrates the penalty function method and the trust region method, introducing Feasible Constraint Policy Optimization (FCPO) to address policy feasibility limitations while ensuring performance and stability. We employ precise penalty functions to broaden the feasible domain of optimization problems, liberating ourselves from the constraints inherent in the initial feasible domain. By utilizing the objective function as a metric for the policy, we introduce a policy iteration algorithm reminiscent of TRPO, ensuring monotonic convergence. Through a reasonable approximation, we put forth the FCPO algorithm, employing the Alternating Direction Method of Multipliers (ADMM) [5] algorithm to decompose the policy optimization problem into several straightforward subproblems that can be optimized efficiently. In summary, this paper makes several significant contributions:

(1) *Theoretical Insights:* We present a sequential policy optimization method characterized by its ability to monotonically enhance constraint violations outside the feasible domain while simultaneously improving algorithm performance when constraints are met.
(2) *FCPO Algorithm:* Our proposed FCPO successfully addresses the constraints associated with initial policy feasibility while ensuring the stable convergence of trust region methods. To alleviate optimization complexity, we introduce the use of ADMM decomposition and advocate for first-order degree updates.
(3) *Superior Performance:* Extensive experimentation demonstrates the advantages of the FCPO algorithm. Demonstrating stable convergence, it consistently delivers robust performance across diverse scenarios, showcasing its reliability and effectiveness in optimization tasks.

2 RELATED WORK AND PRELIMINARY

2.1 Related Work

The primal-dual approach stands as a widely employed method in solving constrained optimization problems. This approach has given rise to influential algorithms [6, 24]. These algorithms tackle constrained optimization by translating them into unconstrained counterparts through Lagrangian functions. By manipulating the Lagrange multiplier, they ensure the constraints are satisfied. Yet, the primal-dual methods are susceptible to variations in Lagrange multipliers, rendering optimization unstable. To address this concern, [21] drew inspiration from control mechanisms and devised CPPOPID. This method uses a PID controller [27] to update Lagrange multipliers resulting in it converging quickly and stably. Notably, conventional dual methods necessitate separate Lagrange multiplier learning and cannot guarantee constraint satisfaction throughout training.

Apart from the primal-dual paradigm, the projection approach finds prevalence in constrained optimization. [28, 29] split optimization into two phases. Firstly, it exclusively refined the objective function within the trust region, securing the optimal policy. Subsequently, this optimal policy underwent projection into the feasible region. However, the policies derived from such methodologies often display excessive conservatism, not aligning with practical utility. [30] counteracted this by integrating a baseline policy with outstanding pre-training performance. This policy improvement involves projecting it closer to the baseline policy to ensure performance standards. One drawback of this approach is its sensitivity to the performance of the baseline algorithm, which can impact overall algorithm performance.

In addition to the aforementioned intricate methodologies, the penalty function method emerges as a straightforward yet effective optimization technique. By directly crafting a penalty function, a minute value prevails within the feasible region while swiftly surging toward infinity as the edge of the region approaches. This category includes notable algorithms such as Interior-point Policy Optimization (IPO) [12] employing a logarithmic penalty function, and Penalized Proximal Policy Optimization (P3O) [31] incorporating exact penalty functions. However, the challenge lies in delineating the penalty coefficient magnitude and narrowing the gap with the optimal solution.

Beyond the traditional optimization models outlined earlier, the local policy search approach further extends the unconstrained RL method into constrained optimization [15, 16]. CPO [1] ingeniously involved upper and lower bounds from TRPO to transform the primary problem into a local policy search problem. Subsequently, the second-order approximation problem is iteratively solved to secure the policy. While CPO exhibits robust convergence in practical scenarios, the computational costs are substantial due to the information matrix calculations. FOCOPS [32] introduced enhancements over CPO by attaining the approximate optimal solution from the original problem, substantially curtailing computational overhead. It's important to note that this method primarily focuses on analyzing the intersection between feasible region and trust region.

While our method shares a similarity with P3O in utilizing exact penalty functions for constructing optimization problems, a distinctive feature sets it apart. Our approach is designed to establish a finite upper bound on the optimal solution of the original problem. This not only provides a metric for gauging the superiority or inferiority of policies beyond the feasible region but also imbues our

algorithm with the capacity for monotonic improvement. In comparison to both FCPO and CPO leveraging confidence domain methods for optimization, our method stands out due to enhancements in the policy iteration optimization process. This improvement ensures the construction of optimization problems featuring feasible solutions across the entire policy definition domain. Furthermore, it facilitates a continuous reduction in the disparity with the optimal policy through iterative refinement.

2.2 Preliminary

A CMDP [3] is formally defined as a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, C, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space , $\mathcal{A}$ is the action space, $\mathcal{P}(s'|s,a) : \mathcal{S}\times\mathcal{S}\times\mathcal{A} \to [0,1]$ is the transition probability function which represents the probability of state transition from s to s' after applying action a, $r : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is the reward function, $C = \{(c_k, b_k)\}_{k=1}^{K}$ is the constraint set (where $c_k : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is the k-th cost function, and b_k is the k-th limits), $\gamma \in (0,1)$ is the discount factor, and $\rho_0 : \mathcal{S} \to [0,1]$ is the initial state distribution.

In the context of CMDP, a policy π is a probability distribution defined on $\mathcal{S}\times\mathcal{A}$, where $\pi(a|s)$ denotes the probability of selecting action a at state s. Π denotes the set encompassing all possible policies. Additionally, a stationary parameterized policy $\pi_{\boldsymbol{\theta}}$ is a probability distribution defined on $\mathcal{S}\times\mathcal{A}$, with $\pi_{\boldsymbol{\theta}}(a|s)$ denotes the probability of playing a at state s, and $\Pi_{\boldsymbol{\theta}} = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathbb{R}^p\}$ denotes the set of all parameterized policies. We denote the probability of visiting state s after t time steps from the initial state s_0 by executing policy π as $\mathcal{P}_\pi(s_t = s|s_0)$. Let $d_\pi^{s_0}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathcal{P}_\pi(s_t = s|s_0)$ be the stationary state distribution of the Markov chain starting at state s_0. We define $d_\pi^{\rho_0}(s) = \mathbb{E}_{s_0\sim\rho_0}[d_\pi^{s_0}(s)]$ as the discounted state visitation distribution on initial distribution ρ_0.

We define $\tau = \{s_t, a_t, r(s_t,a_t), c_k(s_t,a_t)\}_{t=0}^{\infty} \sim \pi$ as the trajectory distribution generated by π, where $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim \mathcal{P}(\cdot|s_t,a_t)$. We express the state value function of the reward as $V_\pi(s) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)|s_0 = s]$, the state-action value function of the reward as $Q_\pi(s,a) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)|s_0 = s, a_0 = a]$ and the advantage function of the reward is $A_\pi(s,a) := Q_\pi(s,a) - V_\pi(s)$. Similarly, we can define the state value function of the k-th cost as $V_\pi^{c_k}(s) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t c_k(s_t,a_t)|s_0 = s]$, define the state-action value function of the cost k as $Q_\pi^{c_k}(s,a) := \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t c_k(s_t,a_t)|s_0 = s, a_0 = a]$ and define the advantage function of the cost k as $A_\pi^{c_k}(s,a) := Q_\pi^{c_k}(s,a) - V_\pi^{c_k}(s)$. The expected discount k-th cost return is defined as $J_k(\pi) := \mathbb{E}_{s\sim\rho_0}[V_\pi^{c_k}(s)]$ and the expected discount reward return is defined as $J(\pi) := \mathbb{E}_{s\sim\rho_0}[V_\pi(s)]$. The feasible policy set Π^C is defined as: $\Pi^C := \bigcap_{k=1}^{K}\{J_k(\pi) \le b_k\}$. The goal of safe RL is to search the optimal policy $\pi^* = \arg\max_{\pi\in\Pi^C} J(\pi)$.

Solving the CMDP problem directly is challenging and inefficient. Typically, policy iteration methods are employed for solving CMDP problems. We formulate the i-th iteration as an unbiased constrained policy iteration problem:

$$\min_{\pi\in\Pi} \; -J(\pi^i) - \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d_{\pi^i}^{\rho_0}\\ a\sim\pi}}[A_{\pi^i}(s,a)] \tag{1a}$$

$$\text{s.t. } J_k(\pi^i) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d_{\pi^i}^{\rho_0}\\ a\sim\pi}}[A_{\pi^i}^{c_k}(s,a)] - b_k \le 0, k = 1,\dots,K \tag{1b}$$

where π^i is the policy from the last iteration.

In earlier studies [14, 16], the policy was iteratively updated using a fixed policy to collect samples for constructing local optimization problems. Their findings highlighted the effectiveness of incorporating local trust domain constraints during the construction of these problems, leading to improved efficiency and performance. Consequently, in the i-th iteration, we anticipate utilizing the current policy π^i to formulate a local optimization problem, resulting in the updated policy π^{i+1}. Expanding on this foundation, CPO introduces upper and lower bound theorems for the objective function and constraint function, articulated as follows:

$$J(\pi^{i+1}) - J(\pi^i) \ge \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d_{\pi^i}^{\rho_0}\\ a\sim\pi^{i+1}}}[A_{\pi^i}(s,a) - \frac{2\gamma\delta_{\pi^{i+1}}}{1-\gamma}D_{TV}(\pi^{i+1}||\pi^i)[s]] \tag{2a}$$

$$J_k(\pi^{i+1}) - J_k(\pi^i) \le \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d_{\pi^i}^{\rho_0}\\ a\sim\pi^{i+1}}}[A_{\pi^i}^{c_k}(s,a) + \frac{2\gamma\delta_{\pi^{i+1}}^{c_k}}{1-\gamma}D_{TV}(\pi^{i+1}||\pi^i)[s]] \tag{2b}$$

where $D_{TV}(\pi^{i+1}||\pi^i)[s] = \frac{1}{2}\sum_a|\pi^{i+1}(a|s) - \pi^i(a|s)|$ is the total variational divergence between action distributions at s, $\delta_{\pi^{i+1}} = \max|\mathbb{E}_{a\sim\pi^{i+1}}[A_{\pi^i}(s,a)]|$ and $\delta_{\pi^{i+1}}^{c_k} = \max|\mathbb{E}_{a\sim\pi^{i+1}}[A_{\pi^i}^{c_k}(s,a)]|$

3 METHODOLOGY

This section is dedicated to the formulation and development of a constrained policy iteration algorithm that guarantees a monotonic improvement in performance.

3.1 Monotonic Improvement Guarantee for Constraint Policies

In contrast to conventional RL tasks, safe RL requires optimizing the objective function while adhering to policy constraints. Traditional approaches, such as Trust Region Policy Optimization (TRPO), often struggle to ensure a consistent monotonic improvement in policy. To provide a theoretical guarantee for their algorithm, previous works typically assume that the initial solution satisfies constraint conditions, ensuring a consistent enhancement of the objective function while meeting constraints. To broaden the algorithm's applicability to the entire policy space, this paper introduces a metric designed to measure the relative superiority or inferiority of two policies that may not strictly adhere to the constraints.

To address this, we incorporate the exact penalty function method [9], a penalty function method preserving the optimal solution, into problem (1). It is reformulated as follows:

$$\min_{\pi\in\Pi,\xi} \; -J(\pi^i) - \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d_{\pi^i}^{\rho_0}\\ a\sim\pi}}[A_{\pi^i}(s,a)] + \sigma g(\boldsymbol{\xi}) \tag{3a}$$

$$\text{s.t. } J_k(\pi^i) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d_{\pi^i}^{\rho_0}\\ a\sim\pi}}[A_{\pi^i}^{c_k}(s,a)] - b_k + \xi_k = 0, \quad k = 1,\dots,K \tag{3b}$$

where σ is positive penalty parameter, $g(\boldsymbol{\xi}) = \sum_{k=1}\max\{0, -\xi_k\}$ is exact penalty function and $\boldsymbol{\xi} = (\xi_1,\dots,\xi_K)^T$ is relaxation variables vector.

Proposition 1. *Assuming $\boldsymbol{\upsilon}$ is the optimal Lagrange multipliers to the constraints (1a) of problem (1). Provided that the penalty factor σ is a sufficiently large constant ($\sigma \geq ||\boldsymbol{\upsilon}||_\infty$), problem (1) and problem (4) yield the same optimal solution.*

Proof. Please see Appendix B.1 in the supplementary material. □

This section is dedicated to the formulation and development of a constrained policy iteration algorithm that guarantees a monotonic improvement in performance.

3.2 Monotonic Improvement Guarantee for Constraint Policies

In contrast to conventional RL tasks, safe RL requires optimizing the objective function while adhering to policy constraints. Traditional approaches, such as Trust Region Policy Optimization (TRPO), often struggle to ensure a consistent monotonic improvement in policy. To provide a theoretical guarantee for their algorithm, previous works typically assume that the initial solution satisfies constraint conditions, ensuring a consistent enhancement of the objective function while meeting constraints. To broaden the algorithm's applicability to the entire policy space, this paper introduces a metric designed to measure the relative superiority or inferiority of two policies that may not strictly adhere to the constraints.

To address this, we incorporate the exact penalty function method [9], a penalty function method preserving the optimal solution, into problem (1). It is reformulated as follows:

$$\min_{\pi\in\Pi,\xi} \; -J(\pi^i) - \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\rho_0}_{\pi^i}\\ a\sim\pi}}[A_{\pi^i}(s,a)] + \sigma g(\xi) \tag{4a}$$

$$\text{s.t. } J_k(\pi^i) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\rho_0}_{\pi^i}\\ a\sim\pi}}[A^{c_k}_{\pi^i}(s,a)] - b_k + \xi_k = 0, \quad k = 1,\dots,K \tag{4b}$$

where σ is positive penalty parameter, $g(\xi) = \sum_{k=1}\max\{0,-\xi_k\}$ is exact penalty function and $\xi = (\xi_1,\dots,\xi_K)^T$ is relaxation variables vector.

Proposition 2. *Assuming $\boldsymbol{\upsilon}$ is the optimal Lagrange multipliers to the constraints (1a) of problem (1). Provided that the penalty factor σ is a sufficiently large constant ($\sigma \geq ||\boldsymbol{\upsilon}||_\infty$), problem (1) and problem (4) yield the same optimal solution.*

Proof. Please see Appendix B.1 in the supplementary material. □

Proposition 2 establishes the convergence of both the original problem (1) and the penalty problem (4) to the same optimal solution, ensuring the integrity and consistency of the optimization process within the bounds of the penalty factor σ.

The fundamental distinction between the two problems lies in their feasible domains. Problem (1) has its feasible domain defined by constraint (1b), while problem (4) extends the feasible solution space to encompass the entire policy space. Given that any policy can be a feasible solution (as proven in Appendix Lemma 4), we leverage the objective function (4a) to gauge the superiority or inferiority of a policy. Furthermore, with the penalty coefficient having a finite value, our objective function acts as a metric for safe RL policies. However, the complexity associated with calculating the state distribution of the policy in problem (4) often presents challenges. In practical scenarios, approximating the distribution by sampling a fixed policy has become a common approach. For convenience, we introduce the following functions:

$$\mathcal{L}^r_{\pi^i}(\pi) := -J(\pi^i) - \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\rho_0}_{\pi^i}\\ a\sim\pi}}[A_{\pi^i}(s,a)]$$

$$\mathcal{L}^{c_k}_{\pi^i}(\pi) := J_k(\pi^i) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\rho_0}_{\pi^i}\\ a\sim\pi}}[A^{c_k}_{\pi^i}(s,a)] - b_k$$

$$D^r_{\pi^i}(\pi) = \frac{2\gamma\delta_{\pi^{i+1}}}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\rho_0}_{\pi^i}\\ a\sim\pi}}[D_{TV}(\pi^{i+1}||\pi^i)[s]]$$

$$D^{c_k}_{\pi^i}(\pi) = \frac{2\gamma\delta^{c_k}_{\pi^{i+1}}}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\rho_0}_{\pi^i}\\ a\sim\pi}}[D_{TV}(\pi^{i+1}||\pi^i)[s]]$$

Leveraging the previously established definition, we formulate the following optimization problem and introduce a constraint policy iteration Algorithm 1 with a guarantee of nonincreasing for function (4a).

$$\min_{\pi\in\Pi,\xi} \quad \mathcal{L}^r_{\pi^i}(\pi) + D^r_{\pi^i}(\pi) + \sigma g(\xi) \tag{5a}$$

$$\text{s.t.} \quad \mathcal{L}^{c_k}_{\pi^i}(\pi) + D^{c_k}_{\pi^i}(\pi) + \xi_k = 0\,, k = 1,\dots,K \tag{5b}$$

Algorithm 1 Policy iteration algorithm guaranteeing nonincreasing objective function

Require: Policy π^0, Penalty parameter σ.
  for $i = 0, 1, \dots$ until convergence **do**
    Collect a set of trajectories $\mathcal{D}$ with policy π^i.
    Compute all advantage values $A_{\pi^i}(s,a),\dots,A^{c_K}_{\pi^i}(s,a)$.
    Construct and solve problem (5) for π^{i+1}.
  end for
Ensure: π^*, ξ^*.

Theorem 3. *There exists a sufficiently large penalty factor σ that ensures the following characteristics of the policy sequence output $\{\pi^i\}_{i=0}^\infty$ generated by Algorithm 1:*

1) The objective function (4a) is guaranteed to be nondecreasing: $-J(\pi^{i+1}) + \sigma g(\xi^{i+1}) \leq -J(\pi^i) + \sigma g(\xi^i)$.

2) When the policy π^i satisfies the constraints (1b), the performance of the policy monotonically improves without violating the constraints.

3) Given any initial policy π^0, the degree of policy constraint violation $g(\xi)$ is guaranteed to be nonincreasing: $g(\xi^{i+1}) \leq g(\xi^i)$.

Proof. Please see Appendix B.2 in the supplementary material. □

Theorem 3 derives the optimization laws that adhere to the objective function and constraints of the algorithm through an in-depth analysis and derivation of the constructed measurement function. In instances where the current policy complies with the constraints, the penalty function ensures the policy's feasibility. During such conditions, the algorithm can monotonically enhance the return

function of the policy while guaranteeing that the output policy remains within the specified constraints. The penalty function serves as a metric for assessing the degree of constraint violation. Conversely, when the current policy fails to meet the constraints, the algorithm systematically improves the constraint violation.

In practical applications, a direct solution to problem (5) may lead to very small step sizes for each policy iteration update. To overcome this challenge, we draw inspiration from the trust region method employed in TRPO. By incorporating the KL divergence between new and old policies as a constraint, we empower the algorithm to achieve more substantial update step sizes. Consequently, we formulate the following optimization problem:

$$\min_{\pi\in\Pi} \mathcal{L}^r_{\pi^i}(\pi) + \sigma g(\boldsymbol{\xi}) \tag{6a}$$

$$\text{s.t. } \mathcal{L}^{c_k}_{\pi^i}(\pi) + \xi_k = 0\,, k = 1,\dots,K \tag{6b}$$

$$\overline{D}_{KL}(\pi||\pi^i) \le \epsilon \tag{6c}$$

where $\bar{D}_{KL}(\pi||\pi^i) = \mathbb{E}_{s\sim d^{\rho_0}_{\pi^i}}[D_{KL}(\pi(\cdot|s)||\pi^i(\cdot|s))]$

As a result of employing penalty function and trust region method, our algorithm demonstrates worst-case constraint violation during the optimization process, as indicated in Proposition 4. Specifically, the relaxation variable converges to 0 with an increasing number of iterations.

Proposition 4. *Suppose π^{i+1}, $\boldsymbol{\xi}^{i,*}$ is the optimal solution of problem (4), the upper bound on the k-th return of π^{i+1} is:*

$$J_k(\pi^{i+1}) \le b_k + \frac{\sqrt{2\epsilon}\gamma\delta^{c_k}_{\pi^{i+1}}}{(1-\gamma)^2} - \xi^{i,*}_k \tag{7}$$

where $\delta^{c_k}_{\pi^{i+1}} = \max|\mathbb{E}_{a\in\pi^{i+1}}[A^{c_k}_{\pi^i}(s,a)]|$.

In contrast to CPO and FOCOPS, the optimization problems they employ only have feasible solutions if the Slater condition is satisfied. Consequently, these algorithms face limitations in optimizing policies that do not meet the constraints. In CPO, the author proposed a recovery method at infeasible case, while FOCOPS did not present a similarly effective solution. Our proposed optimization method offers a unified policy optimization approach, effectively resolving this challenge.

3.3 Finding the Optimal Update Policy

While the exact penalty function method maintains the optimality of the solution, it introduces non-smoothness problems due to the penalty term. Directly applying the Lagrange dual ascending algorithm to solve the problem can compromise the algorithm's stability, particularly when faced with significant fluctuations in the Lagrange multiplier during the training process.

The optimization problem (6) involves numerous variables and intricate constraints, presenting substantial optimization challenges. The ADMM algorithm, rooted in the primal-dual augmented Lagrangian problem, offers a decomposition approach that simplifies complex optimization problems into multiple easily solvable subproblems. By leveraging this method, we break down the original optimization problem into three simpler subproblems.

The formulation of the original dual augmented Lagrangian problem is detailed below, with its construction principles provided in Appendix:

$$\max_{\eta^i\ge 0,\boldsymbol{\lambda}^i} \min_{\pi\in\Pi,\boldsymbol{\xi}^i} L(\pi,\boldsymbol{\lambda}^i,\eta^i,\boldsymbol{\xi}^i;\sigma,\rho) \tag{8}$$

$$\begin{aligned} &L(\pi,\boldsymbol{\lambda}^i,\eta^i,\boldsymbol{\xi}^i;\sigma,\rho)\\ &\triangleq \mathcal{L}^r_{\pi^i}(\pi) + \eta^i(\overline{D}_{KL}(\pi||\pi^i)-\epsilon) + \sigma g(\boldsymbol{\xi}^i)\\ &\quad + \sum_{k=1}^K \lambda^i_k(\mathcal{L}^{c_k}_{\pi^i}(\pi)+\xi^i_k) + \sum_{k=1}^K \frac{\rho}{2}||\mathcal{L}^{c_k}_{\pi^i}(\pi)+\xi^i_k||^2 \end{aligned} \tag{9}$$

where ρ is the proximal factor, $\boldsymbol{\lambda}^i = (\lambda^i_1,\dots,\lambda^i_K)^T$ and η^i are Lagrange multipliers.

For brevity, let us introduce the notation $f^i(\pi) := \mathcal{L}^r_{\pi^i}(\pi) + \max_{\eta^i\ge 0}\eta^i(\overline{D}_{KL}(\pi||\pi^i)-\epsilon)$. Then, we can process by applying the ADMM [5] algorithm to solve the reformulated problem efficiently, see Appendix for the ADMM algorithm details. Then we proceed with the following steps in the j-th iteration.

- Obtain an optimal policy based on the prior estimation $\boldsymbol{\lambda}^{i,j}$ and $\boldsymbol{\xi}^{i,j}$ by solving the following optimization problem:

$$\pi^{i,j+1} = \arg\min_{\pi\in\Pi} f^i(\pi) + \frac{\rho}{2}\sum_{k=1}^K ||\mathcal{L}^{c_k}_{\pi^i}(\pi) + \xi^{i,j}_k + \frac{\lambda^{i,j}_k}{\rho}||^2_2, \tag{10a}$$

- Update $\boldsymbol{\xi}$ by:

$$\xi^{i,j+1} = \arg\min_{\boldsymbol{\xi}} \sigma g(\boldsymbol{\xi}) + \frac{\rho}{2}\sum_{k=1}^K ||\mathcal{L}^{c_k}_{\pi^i}(\pi^{i,j+1}) + \xi_k + \frac{\lambda^{i,j}_k}{\rho}||^2_2, \tag{10b}$$

- Update $\boldsymbol{\lambda}$ by:

$$\lambda^{i,j+1}_k = \lambda^{i,j}_k + \rho(\mathcal{L}^{c_k}_{\pi^i}(\pi^{i,j+1}) + \xi^{i,j+1}_k), k = 1,\dots,K. \tag{10c}$$

It is worth noting that, based on the prior estimation $\boldsymbol{\lambda}^{i,j}$ and $\boldsymbol{\xi}^{i,j}$, we have proved the Theorem 2 in appendix. This allow us to obtain the closed form of the subsequent policy. Through observing the results of Theorem 2, we discover that solving the problem (10a) optimally is akin to addressing an unconstrained problem by maximizing a weighted average of the reward and cost. We identify the constraint coefficient as a composition of error accumulation terms $\lambda^{i,j}_k$, error terms $\rho(\mathcal{L}^{c_k}_{\pi^i}(\pi^i) + \xi^{i,j}_k)$, and differential terms $\rho(\mathcal{L}^{c_k}_{\pi^i}(\pi) - \mathcal{L}^{c_k}_{\pi^i}(\pi^i))$. This insight reveals that our algorithm implicitly incorporates a PID controller for learning Lagrange multipliers, enhancing learning stability.

3.4 Practical Implementation

While solving the problem (10a), allowing π to reside within the policy space Π may result in a policy that does not necessarily belong to the parameterized policy space $\Pi_{\boldsymbol{\theta}}$. Consequently, evaluating or sampling from π may no longer be feasible. To address this issue, we replace the policy π with the parameterized policy $\pi_{\boldsymbol{\theta}}$.

To bolster computational efficiency and simplify complexity, our proposed algorithm in this paper embraces a first-order approach. Given that Theorem 2 establishes closed-form solution for the optimal policy, our algorithm can facilitate the utilization of FOCOPS for policy optimization.

While considering the complexity of FOCOPS, we opt to utilize the Proximal Policy Optimization (PPO) Algorithm [17] to optimize the policy, which incorporates advantage function clipping to ensure adherence to the KL divergence constraint. The policy loss function of the FCPO algorithm is expressed as equation (11). For a

comprehensive understanding of FCPO algorithm, please refer to Algorithm 2. Details can be found in Appendix.

$$Loss(\boldsymbol{\theta}) = -\mathbb{E}_{\substack{s\sim d^{\rho_0}_{\pi_{\boldsymbol{\theta}^i}} \\ a\sim\pi_{\boldsymbol{\theta}^i}}}\left[\frac{\min\{r^i(\boldsymbol{\theta})A_{\pi_{\boldsymbol{\theta}^i}}(s,a), \text{clip}(r^i(\boldsymbol{\theta}),\varepsilon)A_{\pi_{\boldsymbol{\theta}^i}}(s,a)\}}{1-\gamma}\right] + \frac{\rho}{2}\sum_{k=1}^{K}\left\|\frac{\max\left\{r^i(\boldsymbol{\theta})A_{\pi_{\boldsymbol{\theta}^i}}(s,a), \text{clip}(r^i(\boldsymbol{\theta}),\varepsilon)A_{\pi_{\boldsymbol{\theta}^i}}(s,a)\right\}}{1-\gamma} + \xi^{i,j}_k + \frac{\lambda^{i,j}_k}{\rho}\right\|^2_2, \tag{11}$$

where $r^i(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}^i}(a|s)}$ is the importance sampling ratio and $\text{clip}(x,\varepsilon) = \max(\min(x, 1+\varepsilon), 1-\varepsilon)$ is clip function.

Algorithm 2 FCPO Outline

Require: Policy network $\pi_{\boldsymbol{\theta}^0}$, Value networks $V_{\boldsymbol{\phi}^0}, V^{c_1}_{\boldsymbol{\phi}^0_1}, \ldots, V^{c_K}_{\boldsymbol{\phi}^0_K}$.
while stopping criteria not meet **do**
  Generate trajectories $\tau \sim \pi_{\boldsymbol{\theta}^i}$.
  Estimate returns and advantage functions.
  for each iteration **do**
    for each minibatch **do**
      Update value networks by minimizing MSE of $V_{\boldsymbol{\phi}}, V^{target}_{\boldsymbol{\phi}}, \ldots, V^{c_K}_{\boldsymbol{\phi}_K}, V^{c_K,target}_{\boldsymbol{\phi}_K}$.
      Update policy network minimizing problem (11).
    end for
    if $\overline{D}_{KL}(\pi_{\boldsymbol{\theta}}||\pi_{\boldsymbol{\theta}^i}) > \epsilon$ **then**
      Break
    end if
    Update $\boldsymbol{\xi}^i$ by minimizing problem (10b).
    Update $\boldsymbol{\lambda}^i$ using equation (10c).
  end for
end while

4 EXPERIMENTS

4.1 Tasks

Safety-Gymnasium [10] integrated and developed a Gym environment by amalgamating environments like Bullet-Safety-Gym [7] and MuJoCo [25], thereby offering a comprehensive set of RL tasks focused on safety. In Safety-Gymnasium, we select experimental scenarios from the two supported categories of environments. Each task within these scenarios offers various levels of difficulty and we typically choose the medium difficulty level. Further details about these experiments are outlined in Appendix, and you can access the code in the **Supplementary Material**.

4.2 Results

Baseline Algorithms To assess the efficacy of our proposed feasible region relaxation method in enhancing the performance of local policy search algorithms, we conducted experiments using CPO [1] and FOCOPS [32] as our baseline algorithms. CPO represents the foundational trust region approach for constrained optimization, employing second-order optimization to ensure monotonic improvement within the feasible region. FOCOPS extends CPO by approximating the optimal solution to the original constrained problem, substantially reducing computational overhead through first-order optimization while maintaining theoretical guarantees. Furthermore, to evaluate the impact of optimization stability achieved through the ADMM algorithm, we introduced CPPOPID [21] to our baseline. CPPOPID incorporates a PID controller mechanism for Lagrange multiplier updates, aiming to achieve faster convergence and improved stability compared to traditional primal-dual methods. Additionally, to investigate the distinction between directly employing penalty functions for constraint handling versus our integrated approach, we included the P3O [31] algorithm as another baseline in our comparative analysis. P3O utilizes exact penalty functions similar to FCPO but lacks the trust region constraints and ADMM decomposition that characterize our method.

Comparison to baselines Figures 2 and 3 illustrate that, in the majority of tasks, FCPO outperforms other baseline algorithms in terms of rewards while adhering to cost constraints. The performance advantage is particularly pronounced in environments requiring nuanced exploration-exploitation balance. Moreover, FCPO exhibits smoother convergence in its training curves and lower variance across random seeds, indicating superior optimization stability. The shaded regions in these figures, representing $\pm 1\sigma$ standard deviation across 5 independent training runs, are consistently narrower for FCPO compared to baselines, demonstrating reproducible performance that is less sensitive to initialization and stochastic variations during training.

Compared to trust region baselines, FOCOPS demonstrates unstable convergence with significant cost fluctuations and variance in *PointCircle* and *CarCircle*. Cost curves reveal multiple violation spikes, with some episodes exceeding cost values of 100, occurring when FOCOPS's aggressive first-order updates overshoot the feasible region. The high variance (standard deviation 2-3× larger than FCPO) indicates sensitivity to initialization and sampling stochasticity. In contrast, CPO shows stable convergence and strong constraint satisfaction, rarely violating cost thresholds after initial exploration. However, this conservative behavior limits reward performance. For example, CPO achieves only 39.49 reward in *SafetySwimmer* versus FCPO's 147.43, a 3.7× performance gap. Additionally, CPO's second-order Fisher information matrix calculations incur 22-32% longer wall-clock time, limiting scalability to high-dimensional problems.

CPPOPID presents an interesting case study in applying control theory to safe RL. While the PID controller mechanism for Lagrange multiplier updates provides rapid initial constraint satisfaction in simple environments like *SafetyAnt*, the algorithm exhibits oscillatory behavior in more complex tasks. Analysis of CPPOPID's Lagrange multiplier trajectories reveals that the proportional, integral, and derivative gains, tuned for stability in conventional control systems, can produce over-correction in the context of neural policy optimization where the "plant" (the policy-environment system) is highly nonlinear and non-stationary. This manifests as alternating periods of constraint satisfaction and violation, visible in the cost curves of 2 and 3 as higher frequency oscillations compared to other methods. In *SafetyHumanoid*, CPPOPID achieves reasonable mean

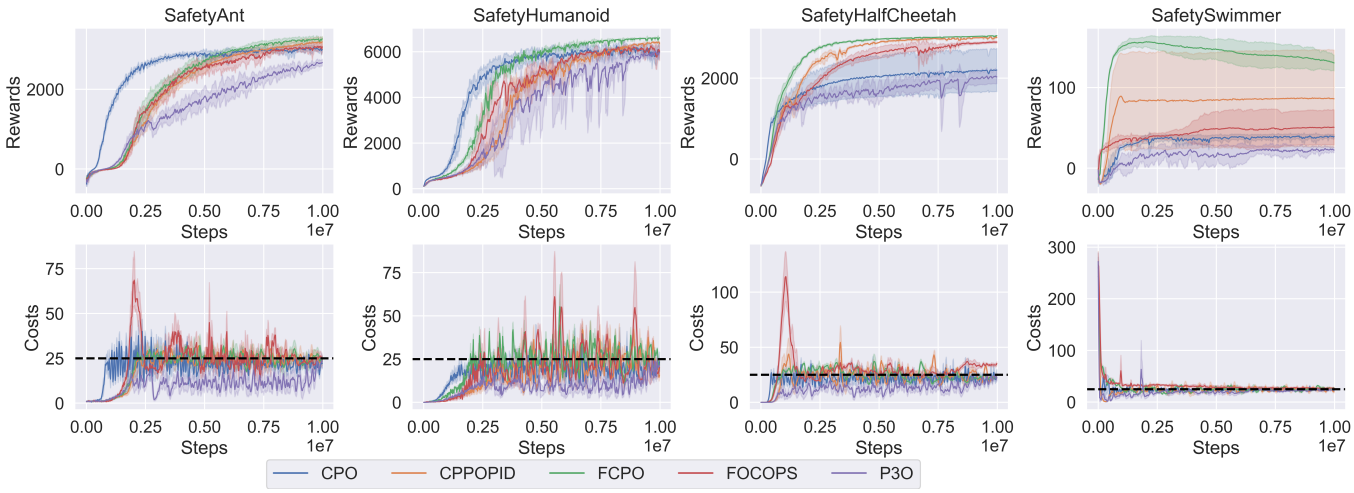

Figure 2: Training curves of Safe Velocity tasks. The rewards and costs are obtained from 10 million interaction steps, with dashed black lines indicating the target cost value set at 25. The shaded region represents the error bars, which are based on the standard deviation (i.e., $\pm 1\sigma$).

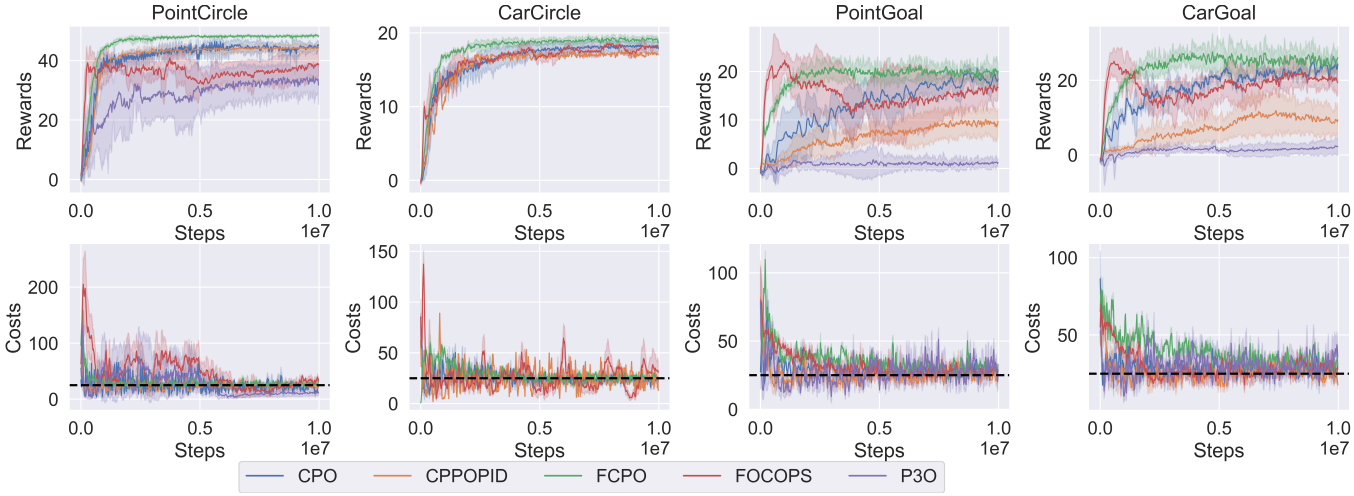

Figure 3: Training curves of Safety Navigation tasks. The rewards and costs are obtained from 10 million interaction steps, with dashed black lines indicating the target cost value set at 25. The shaded region represents the error bars, which are based on the standard deviation (i.e., $\pm 1\sigma$).

performance (6407.81 reward) but with dramatically higher variance (±50.27 cost std) compared to FCPO (±7.76 cost std), indicating less predictable behavior.

Despite both FCPO and P3O employing exact penalty functions for constraint violations, P3O consistently lags behind other algorithms in terms of reward performance, particularly evident in Goal environments where it achieves near-zero rewards. This substantial performance gap reveals a fundamental limitation of naïve penalty function approaches: without the stabilizing structure of trust region constraints and ADMM decomposition, the penalty coefficient becomes difficult to tune. When the penalty is too small, constraints are violated; when too large, the algorithm becomes excessively conservative, effectively treating the constraint boundary as a hard barrier that should never be approached. This prevents exploration of the high reward regions that often exist near constraint boundaries. Our analysis of P3O's policy distributions shows that learned policies exhibit significantly lower action variance compared to FCPO in *SafetySwimmer*, indicating that P3O converges to overly cautious, low entropy policies that sacrifice performance for an abundance of caution.

Generalization Analysis To assess performance robustness, we trained models with a fixed random seed and conducted evaluation trials using 10 different seeds. During evaluation, we use deterministic action selection and disable exploration noise. Table

Table 1: Performance evaluation of different algorithms across various tasks. Data are presented as mean ± standard deviation (±1σ). Bold values indicate the optimal algorithm selected using the following criteria: (1) among algorithms satisfying the safety constraint (mean cost < 25), select the one with highest reward; (2) if no algorithm satisfies the constraint, select the one with lowest mean cost.

Environment		CPO	FOCOPS	CPPOPID	P3O	**FCPO**
PointCircle	Reward	87.03 ± 7.67	64.19 ± 5.22	86.60 ± 2.09	56.06 ± 1.70	**96.18 ± 2.58**
	Cost(<25)	15.02 ± 59.23	84.12 ± 93.12	46.91 ± 34.26	0.97 ± 6.20	13.70 ± 23.15
CarCircle	Reward	37.32 ± 1.76	37.87 ± 2.65	35.58 ± 1.40	37.32 ± 1.76	**38.69 ± 2.50**
	Cost(<25)	64.67 ± 51.10	92.09 ± 54.28	35.03 ± 35.16	64.67 ± 51.10	16.11 ± 31.04
PointGoal	Reward	**19.98 ± 4.21**	16.72 ± 9.07	9.44 ± 5.74	0.82 ± 2.52	21.28 ± 4.28
	Cost(<25)	26.45 ± 26.08	29.28 ± 42.61	28.62 ± 52.08	27.94 ± 95.18	29.80 ± 25.97
CarGoal	Reward	22.50 ± 7.54	19.76 ± 9.12	**12.15 ± 6.93**	−0.10 ± 1.90	23.95 ± 6.42
	Cost(<25)	30.33 ± 31.96	30.83 ± 42.16	24.62 ± 36.53	15.74 ± 49.27	31.36 ± 31.40
SafetyAnt	Reward	3030.91 ± 197.32	3035.05 ± 514.98	3270.54 ± 297.76	2790.09 ± 29.87	**3336.27 ± 11.30**
	Cost(<25)	14.32 ± 5.72	13.87 ± 6.18	29.97 ± 23.19	7.14 ± 4.39	17.12 ± 7.97
SafetyHalfCheetah	Reward	1844.11 ± 19.27	2875.40 ± 19.14	3002.13 ± 145.33	1898.66 ± 21.45	**3053.01 ± 5.51**
	Cost(<25)	20.96 ± 5.79	0.50 ± 0.77	2.18 ± 1.92	27.83 ± 6.88	5.14 ± 2.22
SafetySwimmer	Reward	39.49 ± 1.48	44.63 ± 0.95	41.26 ± 2.87	21.13 ± 11.61	**147.43 ± 1.87**
	Cost(<25)	26.67 ± 1.48	25.19 ± 1.59	23.16 ± 7.29	42.92 ± 45.81	19.38 ± 5.67
SafetyHumanoid	Reward	6388.03 ± 5.64	5934.05 ± 7.38	6407.81 ± 8.53	5985.88 ± 15.53	**6620.16 ± 4.69**
	Cost(<25)	0.16 ± 0.31	20.44 ± 19.76	36.50 ± 50.27	285.78 ± 63.10	5.92 ± 7.76

σ		5	10	20	40
PointCircle	Reward	47.58	48.46	47.97	47.80
	Cost(<25)	20.56	23.73	20.56	22.73
CarCircle	Reward	17.98	17.90	17.89	17.97
	Cost(<25)	23.92	23.60	22.21	22.74
PointGoal	Reward	20.76	20.31	19.73	18.91
	Cost(<25)	30.92	33.54	29.31	27.27
CarGoal	Reward	22.35	22.82	23.98	22.19
	Cost(<25)	29.53	29.33	29.88	29.72
SafetyAnt	Reward	3146	3192	3145	3146
	Cost(<25)	22.99	23.52	21.36	22.99
SafetyHalfCheetah	Reward	3019	3019	3022	3021
	Cost(<25)	24.77	23.96	24.36	24.49
SafetySwimmer	Reward	125	127	126	125
	Cost(<25)	24.83	24.54	23.74	24.22
SafetyHumanoid	Reward	6497	6452	6496	6539
	Cost(<25)	23.80	24.04	24.08	25.78

Table 2: Sensitivity analysis shows performance of FCPO exhibits significant insensitivity to this hyperparameter.

1 summarizes performance across algorithms. FCPO satisfies constraints (cost < 25) in 6 of 8 environments and achieves optimal or near-optimal rewards except in Goal environments, where sparse rewards and procedurally generated obstacles create generalization difficulties for all algorithms. The gap between training and evaluation performance (e.g., *PointCircle*: 96.18 evaluation vs. around 47 during training) occurs because deterministic policies consistently select high probability actions, while stochastic training policies explore more broadly. FCPO exhibits low variance across evaluation seeds. In *SafetyHumanoid*, FCPO achieves 6620.16 ± 4.69 reward, indicating consistent high performance despite the high dimensionality (376-dim observation, 17-dim action) and morphological instability. Comparing to baselines: P3O shows highest cost variance (±95.18 in *PointGoal*, ±63.10 in *SafetyHumanoid*), indicating brittle policies that occasionally violate constraints on out-of-distribution states; FOCOPS shows moderate performance with higher variance (±514.98 in *SafetyAnt*); CPO shows most consistent constraint satisfaction but moderate rewards.

Sensitivity Analysis The penalty factor σ plays a crucial role in FCPO algorithms, directly influencing the trade-off between reward optimization and constraint satisfaction through the exact penalty function $g(\xi) = \sum_k \max\{0, -\xi_k\}$. We systematically vary $\sigma \in \{5, 10, 20, 40\}$ and measure final performance across all eight environments. Table 2 shows FCPO exhibits low sensitivity to σ. The coefficient of variation in reward across these σ values is remarkably low across all environments. This robustness arises from FCPO's algorithmic structure: the exact penalty function ensures sufficiently large σ values achieve the same optimal solution (Proposition 2), while ADMM adaptively adjusts the effective penalty through evolving Lagrange multipliers λ^i. When σ is large, the penalty dominates early in training but multipliers grow slowly; when σ is moderate, multipliers grow larger to compensate, achieving similar equilibrium.

5 CONCLUSION

In this paper, we present a novel constrained policy iteration algorithm with monotonic convergence guarantees. Using ADMM decomposition, our algorithm simplifies constrained policy optimization into three tractable subproblems, ensuring practical efficiency. The algorithm combines the benefits of trust region methods and penalty functions, extending policy optimization to the entire policy space rather than requiring initial feasibility.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grants 62301559.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International conference on machine learning*. PMLR, 22–31.
[2] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–38.
[3] Eitan Altman. 1999. *Constrained Markov decision processes*. Vol. 7. CRC press.
[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).
[5] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning* 3, 1 (2011), 1–122.
[6] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research* 18, 1 (2017), 6070–6120.
[7] Sven Gronauer. 2022. Bullet-safety-gym: A framework for constrained reinforcement learning. (2022).
[8] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.
[9] S P Han and Olvi L Mangasarian. 1979. Exact penalty functions in nonlinear programming. *Mathematical programming* 17 (1979), 251–269.
[10] Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. 2023. Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=WZmlxIuIGR
[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.
[12] Yongshuai Liu, Jiaxin Ding, and Xin Liu. 2020. IPO: Interior-point policy optimization under constraints. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 4940–4947.
[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[14] Jan Peters and Stefan Schaal. 2008. Reinforcement learning of motor skills with policy gradients. *Neural networks* 21, 4 (2008), 682–697.
[15] Matteo Pirotta, Marcello Restelli, Alessio Pecorino, and Daniele Calandriello. 2013. Safe policy iteration. In *International Conference on Machine Learning*. PMLR, 307–315.
[16] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.
[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[18] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.
[19] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.
[20] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. 2022. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review* (2022), 1–46.
[21] Adam Stooke, Joshua Achiam, and Pieter Abbeel. 2020. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*. PMLR, 9133–9143.
[22] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[23] AlphaStar Team. 2019. AlphaStar: Mastering the real-time strategy game StarCraft II. *DeepMind blog* 24 (2019).
[24] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. 2018. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074* (2018).
[25] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.
[26] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.
[27] MJ Willis. 1999. Proportional-integral-derivative control. *Dept. of Chemical and Process Engineering University of Newcastle* 6 (1999).
[28] Long Yang, Jiaming Ji, Juntao Dai, Yu Zhang, Pengfei Li, and Gang Pan. 2022. Cup: A conservative update policy algorithm for safe reinforcement learning. *arXiv preprint arXiv:2202.07565* (2022).
[29] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. 2020. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152* (2020).
[30] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. 2021. Accelerating safe reinforcement learning with constraint-mismatched baseline policies. In *International Conference on Machine Learning*. PMLR, 11795–11807.
[31] Linrui Zhang, Li Shen, Long Yang, Shixiang Chen, Bo Yuan, Xueqian Wang, and Dacheng Tao. 2022. Penalized proximal policy optimization for safe reinforcement learning. *arXiv preprint arXiv:2205.11814* (2022).
[32] Yiming Zhang, Quan Vuong, and Keith Ross. 2020. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems* 33 (2020), 15338–15349.