P2BPO: Permeable Penalty Barrier-Based Policy Optimization for Safe RL

Sumanta Dey[1], **Pallab Dasgupta**[2], **Soumyajit Dey**[1]

[1]Indian Institute of Technology Kharagpur, India
[2]Synopsys Inc, Sunnyvale, USA
sumanta.dey@iitkgp.ac.in, pallabd@synopsys.com, soumya@cse.iitkgp.ac.in

Abstract

Safe Reinforcement Learning (SRL) algorithms aim to learn a policy that maximizes the reward while satisfying the safety constraints. One of the challenges in SRL is that it is often difficult to balance the two objectives of reward maximization and safety constraint satisfaction. Existing algorithms utilize constraint optimization techniques like penalty-based, barrier penalty-based, and Lagrangian-based dual or primal policy optimization methods. However, they suffer from training oscillations and approximation errors, which impact the overall learning objectives.

This paper proposes the Permeable Penalty Barrier-based Policy Optimization (P2BPO) algorithm that addresses this issue by allowing a small fraction of penalty beyond the penalty barrier, and a parameter is used to control this permeability. In addition, an adaptive penalty parameter is used instead of a constant one, which is initialized with a low value and increased gradually as the agent violates the safety constraints. We have also provided theoretical proof of the proposed method's performance guarantee bound, which ensures that P2BPO can learn a policy satisfying the safety constraints with high probability while achieving a higher expected reward. Furthermore, we compare P2BPO with other SRL algorithms on various SRL tasks and demonstrate that it achieves better rewards while adhering to the constraints.

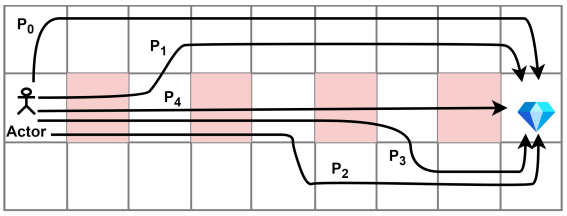
Introduction

Reinforcement Learning (RL) (Sutton and Barto 1998), a nature-inspired reward-based learning method, has shown potential for complex strategy-based tasks like playing Chess, Go (Silver et al. 2018), Atari games (Mnih et al. 2013) etc. Considering these, industry leaders in many safety-critical domains, like industrial robots and autonomous vehicles, are planning or already using RL-based controllers. In conventional RL algorithms, the RL agents are trained only to learn the policies that maximize cumulative reward. However, for safety-critical environments, the learning agent must take action while satisfying safety constraints. Such approaches often discard policies with good rewards due to potential safety violations leading to undesirable behaviors. For example, an autonomous vehicle must be cautious and slow in slippery road conditions (rain, snow) to avoid accidents, even if it takes a longer time to reach the destination.

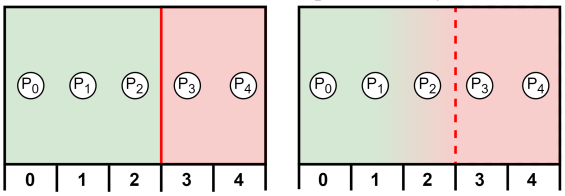
Constrained environments modeled as Markov Decision Processes (MDPs) (Sutton and Barto 1998) with constraints are generally called Constrained Markov Decision Processes (CMDPs) (Altman 1999). The RL agent must learn a policy that maximizes the cumulative reward while satisfying the constraints for these environments. There can be different types of constraint configurations: binary or instantaneous constraints and cumulative constraints. Binary constraints are evaluated on current state valuations. Whereas for the latter one, the constraint values (generally termed as costs) are accumulated over the episodes, and their violation depends on whether the cumulative cost is within a defined limit or not. The cumulative sum of constraints (costs) could be discounted or averaged over a few episodes.

Various methods have been proposed in recent years to train RL agents in CMDP and learn the maximum rewarded policies under given constraints. One such prior work is Constrained policy optimization (CPO) (Achiam et al. 2017), which follows the Trust Region Policy Optimization (TRPO) (Schulman et al. 2015) algorithm. CPO solves the CMDP objective function by approximating the non-convex constrained objective function to a quadratic optimization problem within some trust regions. Therefore, like TRPO, CPO improves the policy monotonically. However, CPO demands longer computation time to calculate the inverse Fisher information matrix and the second-order derivatives. In addition, CPO requires additional interactions for feasible recovery if the primal problem or the objective function is infeasible under certain initial policies. CPO also does not handle mean valued constraints and is hard to implement with multiple constraints.

Some other approaches (Achiam and Amodei 2019), (Stooke, Achiam, and Abbeel 2020) use Lagrangian multiplier-based constraints optimization techniques to solve CMDP, where the CMDP objective is first converted into an unconstrained objective function by adding the constraints functions multiplied by the corresponding Lagrange multipliers value with the original objective. The Lagrange multipliers are also updated during training based on the constraint satisfaction and dissatisfaction. This method ensures the constraints will be satisfied when the policy converges. However, Lagrange multipliers-based methods' per-



(a) Grid World: Red blocks represent safety hazards.



(b) General Penalty Barrier (c) Permeable Penalty Barrier

Figure 1: Small Grid World. X-axis buckets in (b) and (c) represent the safety violation counts (costs), and the red lines represent the cost limit.

formance greatly depends on the Lagrange multiplier's initial value and update rate and also suffers from inherent oscillation and cost overshoot (Wah et al. 1997).

Penalty-based methods (Liu, Ding, and Liu 2019), (Zhang et al. 2022) are another clan of popular constrained optimization techniques used to solve CMDPs. The penalty-based methods are the simplest, where the penalty function returns a huge penalty factor on constraint violations, creates an equivalent primal problem, and subsequently helps to achieve probable sub-optimal solutions (Zhang, Vuong, and Ross 2020b). However, as the penalty factor grows, the approximation inaccuracy also grows, presenting a difficult trade-off.

Our work proposes a smooth function for the penalty-based constraint optimization method. For example, consider the Grid World shown in Figure 1a. In this grid world, the agent aims to collect the treasure (diamond) as quickly as possible. However, the red blocks pose safety hazards, and the agent gets a cost of one each time the agent visits a red block. We show five different paths (P_0, P_1, P_2, P_3, and P_4) the agent can follow to collect the diamond. The subscripts of the paths represent the accumulated costs for the respective path. Considering the maximum allowable cost limit as '2', P_0, P_1, and P_2 are safe paths, whereas P_3 and P_4 are unsafe. Therefore, with the conventional penalty function (Figure 1b), paths P_3 and P_4 are penalized (in the Figure colored in red) due to violation of safety boundary (> 2). In contrast, in our proposed approach, we consider a *permeable penalty barrier* (dashed red line in Figure 1c) that allows a small fraction of the penalty to pass to the other side of the barrier. However, the amount gets smaller as it goes deep into the safe region. Therefore, we show in Figure 1c with a gradient of red that the "Path P_2" also gets a small fraction of the penalty near the penalty barrier. In comparison, P_1 and P_0 do not receive any as they reside deep inside the safe region. This allows us to mitigate the issue of oscillation across the penalty barrier during training. We define the permeability parameter that controls the aggressiveness of the penalty boundary. We consider the adaptive penalty parameter initially set to a low value and increased gradually as the agent violates the safety constraints. This helps mitigate the approximation error arising from a large fixed penalty-based method. Our overall algorithm is easy to implement, and the hyperparameters are easy to tune. In summary, our contributions are as follows:

- We propose P2BPO, a permeable penalty-based first-order constraint optimization SRL algorithm for CMDPs with an easy-to-tune parameter to control the permeability.
- We provide a theoretical performance bound of our P2BPO algorithm.
- We conduct various experiments to compare P2BPO with the existing state-of-the-art penalty-based, Lagrangian-based, and quadratic optimization-based SRL algorithms.

Preliminaries

Markov Decision Process (MDP) (van Otterlo and Wiering 2012) can be defined as a tuple, $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mu >$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions. The symbols $\mathcal{R}$, $\mathcal{P}$, and μ stand for the reward function, the transition probability function, and the starting state distribution, respectively. A policy $\pi \in \Pi$ is a mapping from state to the probability distribution over actions, which can also be written as $\pi(\mathcal{S}) \rightarrow \mathcal{P}(\mathcal{A})$, where Π represents the set of all policies. The probability of selecting the action a in state s for a given policy π is denoted using $\pi(a|s)$.

The primary objective in Reinforcement Learning(RL) is to select a policy $\pi \in \Pi$ that maximizes the total accumulated discounted reward or $\mathcal{J}(\pi)$ over the infinite horizon. Therefore,

$$\mathcal{J}(\pi) = \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}) \right] \tag{1}$$

Here $\gamma \in [0, 1))$ is the discount factor, $\tau \sim \pi$ represents the trajectory sampled using the policy π. Now considering $\mathcal{R}(\tau)$ as the discounted return of the trajectory τ, we define the on-policy value function $\mathcal{V}^\pi(s)$, action-value function $\mathcal{Q}^\pi(s, a)$, and the advantage function $A^\pi(s, a)$ as follows:

$$\begin{aligned} \mathcal{V}^\pi(s) &= \mathop{\mathbb{E}}_{\tau \sim \pi} [\mathcal{R}(\tau) | s_0 = s] \\ \mathcal{Q}^\pi(s, a) &= \mathop{\mathbb{E}}_{\tau \sim \pi} [\mathcal{R}(\tau) | s_0 = s, a_0 = a] \\ A^\pi(s, a) &= \mathcal{Q}^\pi(s, a) - \mathcal{V}^\pi(s) \end{aligned}$$

Now, considering the discounted future state distribution, $d^\pi = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathcal{P}(s_t = s | \pi)$. Now as mentioned in (Kakade and Langford 2002), the difference in performance between two policies π' and π is

$$\mathcal{J}(\pi') - \mathcal{J}(\pi) = \frac{1}{1 - \gamma} \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'(s)}} [A^\pi(s, a)] \tag{2}$$

Constrained Markov Decision Process (CMDP) can be defined as a tuple, $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mu, \mathcal{C} >$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{P}$, μ denotes the standard MDP tuple and $\mathcal{C}$ defines the set of cost functions. Similar to the reward function, each $\mathcal{C}_i \in \mathcal{C}$ is also a function that returns cost ($\in \mathbb{R}$) for each transition tuple $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$.

$$\mathcal{C}_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$$

Considering $\mathcal{J}_{\mathcal{C}_i}(\pi)$ as the total accumulated discounted cost for the policy π,

$$\mathcal{J}_{\mathcal{C}_i}(\pi) = \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{C}_i(s_t, a_t, s_{t+1}) \right]$$

For each constraint, the corresponding cost function $\mathcal{C}_i$ is bounded by an upper limit b_i that denotes the maximum limit of dissatisfaction of the respective constraint. The constraint, therefore, limits the set of feasible or allowable policies $\Pi_{\mathcal{C}} \subseteq \Pi$ for a CMDP.

$$\Pi_{\mathcal{C}} = \{\pi \in \Pi \ : \ \forall i \ \mathcal{J}_{\mathcal{C}_i}(\pi) \leq b_i\}$$

Therefore, the updated RL objective for a CMDP is

$$\pi^* = \arg\max_{\pi \in \Pi_{\mathcal{C}}} \mathcal{J}(\pi) \tag{3}$$

Proximal Policy Optimization (PPO) (Schulman et al. 2017) is a Policy Gradient (Sutton et al. 1999) based method for finding an optimal policy for an MDP problem. PPO defines the objective function as

$$\max_{\theta} \mathcal{L}^{CLIP}(\theta) = \mathbb{E}_t \left[min(r_t(\theta)A_t, r_t^{clip}(\theta)A_t) \right] \tag{4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio between the old policy ($\pi_{\theta_{old}}$) and the new policy (π_θ) and $r_t^{clip}(\theta) = clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$, a clipped $r_t(\theta)$ value between $[1-\epsilon, 1+\epsilon]$.

Permeable Penalty Barrier-based Policy Optimization

As opposed to general penalty-based methods where the penalty is applied only on constraint violations, with P2BPO [1], we apply a gradual penalty as state trajectories approach constraint violations. Based on Eq. 2, the safe reinforcement learning objective as mentioned in (Achiam et al. 2017) is given by,

$$\begin{aligned}
&\pi_{k+1} = \arg\max_{\pi} \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'(s)}} [A^{\pi_k}(s,a)] \\
&s.t. \ \forall i, \ \mathcal{J}_{c_i}(\pi_k) + \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'(s)}} [A_{c_i}^{\pi_k}(s,a)] \leq b_i \\
&or, \ (\mathcal{J}_{c_i}(\pi_k) - b_i) + \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'(s)}} [A_{c_i}^{\pi_k}(s,a)] \leq 0 \\
&or, \ (1-\gamma)\hat{\mathcal{J}}_{c_i}(\pi_k) + \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'(s)}} [A_{c_i}^{\pi_k}(s,a)] \leq 0
\end{aligned} \tag{5}$$

[1] Code, Full Paper: https://github.com/sumantasunny/P2BPO.git

where, $\hat{\mathcal{J}}_{c_i}(\pi_k) = (\mathcal{J}_{c_i}(\pi_k) - b_i)$ and $A_{c_i}^{\pi_k}(s,a)$ is the cost advantage function. Considering $(1-\gamma)\hat{\mathcal{J}}_{c_i}(\pi_k) + \mathbb{E}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'(s)}} [A_{c_i}^{\pi_k}(s,a)]$ as $\phi_{c_i}^{\pi_k}(\pi)$ and $\mathbb{E}_{\substack{s \sim d^{\pi'} \\ a \sim \pi'(s)}} [A^{\pi_k}(s,a)]$ as $\mathcal{L}_{\mathcal{R}}^{\pi_k}(\pi)$, Eq. 5 can be rewritten as,

$$\begin{aligned}
&\pi_{k+1} = \arg\min_{\pi} -\mathcal{L}_{\mathcal{R}}^{\pi_k}(\pi) \\
&s.t. \ \phi_{c_i}^{\pi_k}(\pi) \leq 0, \ \forall i
\end{aligned} \tag{6}$$

We construct a linear first-order penalty function from Eq. 6 as,

$$\varphi_{c_i}^{\pi_k}(\pi) = \begin{cases} 0, & \text{if } \phi_{c_i}^{\pi_k}(\pi) \leq 0 \\ \chi_{i,k} \times \phi_{c_i}^{\pi_k}(\pi), & otherwise. \end{cases} \tag{7}$$

Here $\chi_{i,k} \geq 0$ is a scalar quantity called the penalty parameter. We update the value of $\chi_{i,k}$ in each iteration based on the current constraint violation $(\mathcal{J}_{c_i}(\pi_k) - b_i)$ following,

$$\chi_{i,k+1} = max\{0, \ [\chi_{i,k} + \eta_\chi.(\mathcal{J}_{c_i}(\pi_k) - b_i)]\}. \tag{8}$$

In the above equation, η_χ refers to the learning rate of the penalty parameters. The penalty function in Eq. 7 is a ReLU-like function. However, the disadvantage of this penalty function is that it is not a smooth/derivable function. Therefore, we consider the Softplus function Eq. 9 as an approximate replacement, a derivable and smooth function.

$$softplus(x, \beta) = \frac{1}{\beta} ln(1 + e^{\beta.x}) \tag{9}$$

The β parameter is used to control the degree of smoothness of the Softplus function at zero. With the value of $\beta \to \infty$, the Softplus function behaves like the ReLU function. We consider β as the permeability parameter, which has a reciprocal relation to permeability. In the later Section, we will discuss how this β value can be used to control the tradeoff between reward vs. cost.

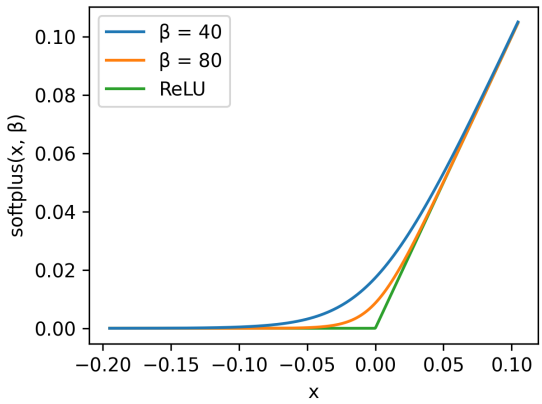


Figure 2: Softplus function with different β value.

Considering the objective function in Eq. 6 and the penalty function in Eq. 7, we construct the unconstrained optimization objective function as:

$$\Phi_C(\pi_{k+1}) : \arg\min_{\pi} \Big[-\mathcal{L}_{\mathcal{R}}^{\pi_k}(\pi) + \sum_{i=0}^{m} softplus(\phi_{c_i}^{\pi_k}(\pi), \beta) \Big] \tag{10}$$

where m denotes the number of safety constraints.

Our work considers the policy as a parametric policy parameterized using θ. We define the parametric representation of $\mathcal{L}_{\mathcal{R}}^{\pi_k}(\pi)$ as $\mathcal{L}_{\mathcal{R}}^{\pi_\theta}(\theta)$ or in short $\mathcal{L}_{\mathcal{R}}(\theta)$. Similarly, we define the parametric representation of $\sum_{i=0}^{m} softplus[\phi_{c_i}^{\pi_k}(\pi)]$ as $\sum_{i=0}^{m} softplus[\phi_{c_i}^{\pi_\theta}(\theta)]$ or in short $\mathcal{L}_{\mathcal{C}}(\theta)$. Therefore, the overall parametric optimization objective of P2BPO method ($\mathcal{L}^{P2BPO}(\theta)$) in Eq. 10 can be written as follows:

$$\mathcal{L}^{P2BPO}(\theta) = -\mathcal{L}_{\mathcal{R}}(\theta) + \mathcal{L}_{\mathcal{C}}(\theta) \tag{11}$$

Performance Guarantee Bound

Theorem 1 *Difference between the optimal values of P2BPO objective of Eq. 11 and SRL objective of Eq. 6 is bounded by* $\frac{m.ln(2)}{\beta}$.

Proof: By converting the Eq. 6 into the parameterized form with parameter θ

$$\min_{\theta}\{-L_R(\theta)\}, \quad s.t.\ \phi_{c_i}^{\pi_\theta} \leq 0 \tag{12}$$

Now converting Eq. 12 to Lagrangian form, we get:

$$L(\theta, \boldsymbol{\lambda}) = \min_{\theta}\Big[(-L_R(\theta)) + \sum_{i=1}^{m} \lambda_i \phi_{c_i}^{\pi_\theta}\Big] \tag{13}$$

where λ_i is the Lagrange Multiplier for the constraint c_i and $\boldsymbol{\lambda}$ is the set of all Lagrange Multipliers. From this, we get dual function:

$$g(\boldsymbol{\lambda}) = \min_{\theta}\Big[-L_R(\theta) + \sum_{i=1}^{m} \lambda_i \phi_{c_i}^{\pi_\theta}\Big] \tag{14}$$

The parametric objective function of P2BPO from Eq. 10 is:

$$\min_{\theta}\Big[-L_R(\theta) + \sum_{i=1}^{m} softplus(\phi_{c_i}^{\pi_\theta}, \beta)\Big] \tag{15}$$

Now if Eq. 15 is strictly feasible, then there must be an optimal parameter θ^*, such that the following should hold:

$$-\nabla L_R(\theta^*) + \sum_{i=1}^{m} sigmoid(\phi_{c_i}^{\pi_{\theta^*}}, \beta)\nabla\phi_{c_i}^{\pi_{\theta^*}} = 0 \tag{16}$$

as the derivative of the $softplus()$ is $sigmoid()$. The full derivation is shown in Appendix A.1. Now let,

$$\lambda_i^* = sigmoid(\phi_{c_i}^{\pi_{\theta^*}}, \beta) \tag{17}$$

and let $\boldsymbol{\lambda}^* = [\lambda_1^* \cdots \lambda_m^*]^T$. Now, from Eq. 16 and 17,

$$-\nabla L_R(\theta^*) + \sum_{i=1}^{m} \boldsymbol{\lambda}^* \nabla\phi_{c_i}^{\pi_{\theta^*}} = 0 \tag{18}$$

It means that θ^* minimizes the Lagrangian Eq. 14 when $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$. Therefore, we get

$$g(\boldsymbol{\lambda}^*) = -L_R(\theta^*) + \sum_{i=1}^{m} \lambda_i^* \phi_{c_i}^{\pi_{\theta^*}}$$

$$= -L_R(\theta^*) + \sum_{i=1}^{m} sigmoid(\phi_{c_i}^{\pi_{\theta^*}}, \beta)\phi_{c_i}^{\pi_{\theta^*}} \text{ by Eq.17} \tag{19}$$

$$= -L_R(\theta^*) + \sum_{i=1}^{m} (softplus(\phi_{c_i}^{\pi_{\theta^*}}, \beta) - k) \tag{20}$$

by approximating $sigmoid(\phi_{c_i}^{\pi_{\theta^*}}, \beta)\phi_{c_i}^{\pi_{\theta^*}}$ using $softplus(\phi_{c_i}^{\pi_{\theta^*}}, \beta)$ with k being the approximation error.

Lemma 1: *Approximation error (k) for approximating $(x.sigmoid(x, \beta))$ using $softplus(x, \beta)$ is $\leq \frac{ln(2)}{\beta}$.*

Proof: Proof of Lemma 1 is given in Appendix A.2.

R.H.S. of Eq. 20 can be written as:

$$\Big[-L_R(\theta^*) + \sum_{i=1}^{m} softplus(\phi_{c_i}^{\pi_{\theta^*}}, \beta)\Big] - \sum_{i=1}^{m} k$$

$$= \Big(\hat{p}^* - \sum_{i=1}^{m} k\Big) = \Big(\hat{p}^* - mk\Big) \tag{21}$$

where $\hat{p}^*$ is the optimal value in the problem of Eq. 15. Let p^* be the optimal value in the problem Eq. 12. Now, considering the property of duality gap, $p^* \geq g(\boldsymbol{\lambda}^*)$. Hence,

$$p^* \geq \hat{p}^* - mk$$
$$\hat{p}^* - p^* \leq mk$$
$$\hat{p}^* - p^* \leq \frac{m.ln(2)}{\beta} \qquad \text{(From } Lemma\ 1.\text{)} \tag{22}$$

Therefore, if $\beta \to \infty$, then $\frac{m.ln(2)}{\beta} \to 0$ or $\hat{p}^*$ will be equal to p^* (as $m \geq 0$ and $ln(2) \geq 0$) and the $softplus$ function will be more like a ReLU function. However, with a large value of β, the cost may also increase even though we might get higher rewards. We leverage binary search to find a reasonable β value that balances conservativeness in terms of cost and reward maximization.

Practical Implementation

We have implemented our method in the Deep RL setup and developed it considering the (PKU-Alignment 2023) GitHub repository as the base. We consider Proximal Policy Optimization (PPO) (Schulman et al. 2017) as the base. Therefore, we have constructed one actor or the policy network and several critic networks, one for reward and one for each cost function. Like the PPO algorithm, our algorithm also iteratively samples trajectories and updates the policy and critic networks. Algo 1 provides the step-by-step description of our P2BPO algorithm. One of the major advantages of our method over the other penalty methods comes with using the Softplus function as the boundary function. The Softplus function is differentiable. Thus, the loss function of our P2BPO algorithm is also differentiable everywhere and, therefore, can easily be solved using a first-order optimizer like Adam Optimizer (Kingma and Ba 2014).

Experiments

We have run various experiments with different environments, learning agents, and constraints to evaluate the following claims empirically.

- P2BPO-based policy will return the maximum cumulative reward while satisfying constraints over the existing state-of-the-art SRL algorithms.

Env/Cost Limit	CPO	FOCOPS	PCPO	PPO-Lag	CPPO-PID	APPO	IPO	P3O	P2BPO
Ant Run	356 ± 21	977 ± 2.6	898 ± 9.7	983 ± 2.5	969 ± 7.4	975 ± 6.5	1175 ± 16	797 ± 18	$\mathbf{992 \pm 0.1}$
(25)	11 ± 1.3	8.7 ± 0.3	5.0 ± 0.2	2.4 ± 0.1	2.3 ± 0.1	5.4 ± 0.3	54 ± 3.0	10 ± 0.8	$\mathbf{4.4 \pm 0.1}$
Humanoid Run	487 ± 4.9	554 ± 6.8	574 ± 7.3	1496 ± 57	4412 ± 57	3942 ± 69	4210 ± 64	2562 ± 90	$\mathbf{4632 \pm 45}$
(25)	0 ± 0.0	0 ± 0.0	0 ± 0.0	0 ± 0.0	0 ± 0.0	0 ± 0.0	0 ± 0.0	0 ± 0.0	$\mathbf{0 \pm 0.0}$
SafeBallCircle	506 ± 4.8	752 ± 4.6	455 ± 6.4	675 ± 5.5	706 ± 7.3	701 ± 7.8	734 ± 7.0	575 ± 3.0	$\mathbf{728 \pm 6.2}$
(25)	28 ± 1.3	34 ± 1.3	55 ± 1.6	17 ± 1.1	25 ± 1.2	25 ± 1.3	32 ± 1.3	22 ± 1.1	$\mathbf{25 \pm 1.4}$
SafeCarCircle	810 ± 7.4	1155 ± 4.9	589 ± 11	1223 ± 7.1	1262 ± 7.6	1224 ± 5.0	1276 ± 4.8	527 ± 10	$\mathbf{1275 \pm 5.5}$
(25)	23 ± 1.9	23 ± 1.3	83 ± 3.1	22 ± 1.4	27 ± 1.7	29 ± 1.4	41 ± 1.7	17 ± 1.6	$\mathbf{25 \pm 1.1}$
PointGoal1	24 ± 0.2	18 ± 0.4	26 ± 0.1	$\mathbf{15 \pm 0.3}$	8.4 ± 0.3	6.4 ± 0.3	22 ± 0.2	6.2 ± 0.3	12 ± 0.4
(25)	36 ± 1.4	28 ± 1.4	51 ± 1.6	$\mathbf{20 \pm 1.1}$	34 ± 4.7	25 ± 3.8	37 ± 2.0	28 ± 4.6	27 ± 2.9
CarGoal1	32 ± 0.3	24 ± 0.5	33 ± 0.1	14 ± 0.5	5.3 ± 0.2	4.3 ± 0.2	24 ± 0.4	10 ± 0.4	$\mathbf{15 \pm 0.5}$
(25)	43 ± 1.9	35 ± 2.3	57 ± 2.3	21 ± 1.8	26 ± 2.9	21 ± 2.8	42 ± 2.0	19 ± 1.4	$\mathbf{22 \pm 1.4}$
PointButton1	8.2 ± 0.5	9.2 ± 0.6	28 ± 0.3	4.2 ± 0.3	0.7 ± 0.1	1.2 ± 0.1	12 ± 0.5	1.1 ± 0.1	$\mathbf{3.8 \pm 0.2}$
(25)	47 ± 3.1	58 ± 3.4	126 ± 3.3	26 ± 2.8	23 ± 2.3	23 ± 3.3	77 ± 3.9	17 ± 1.9	$\mathbf{24 \pm 2.7}$
CarButton1	4.6 ± 0.3	3.6 ± 0.3	21 ± 0.6	0.7 ± 0.2	-1.0 ± 0.1	-0.6 ± 0.1	6.6 ± 0.4	-1.9 ± 0.2	$\mathbf{-0.2 \pm 0.1}$
(25)	76 ± 4.5	69 ± 5.4	275 ± 8.2	41 ± 4.7	39 ± 4.0	32 ± 3.5	143 ± 6.7	23 ± 3.8	$\mathbf{24 \pm 2.6}$

Table 1: Average reward and cost $\pm$ 95% confidence interval over 128 episodes of Baselines and P2BPO for various environments. The first row of each environment represents the rewards, and the second row represents the costs. The best rewarding policies while satisfying the cost limit are highlighted in bold.

Algorithm 1: P2BPO Parametric Policy Update

Input: Initial Parametric Policy Network (π_{θ_0}), Reward Critic Network(V_{Θ_0}) and Cost Critic Networks ($V^{c_i}_{\vartheta_{i,0}}$); Penalty Parameter $\chi_{i,0}$

for *each epoch k* **do**
  Generate and Collect trajectories τ based on π_{θ_k}
  for *each training-batch* **do**
    Update Penalty Param $\chi_{i,k}$ following Eq. 8
    Update Policy (Actor) Network (π_{θ_k})
      $\theta_{k+1} \leftarrow \theta_k - \eta \nabla_\theta \mathcal{L}^{P2BPO}(\theta_k)$
    Update Reward Critic Network (V_{Θ_k})
    Update Cost Critic Networks ($V^{c_i}_{\vartheta_{i,k}}$)
  end
end
Comments:
η: Policy Learning Rate
η_χ: Penalty Parameter Update Rate

- P2BPO algorithm can adapt to the different cost thresholds and precisely maintain the cost thresholds during training.
- Minor changes in the permeability parameter value do not significantly impact the P2BPO algorithm's performance.

We have also considered that all the algorithm-specific hyperparameters will be the same across all the environments. At the same time, the environment-specific hyperparameters will also be constant across all the algorithms.

Baselines: We consider various current state-of-the-art baselines from three different categories that employ different constraints optimization techniques:

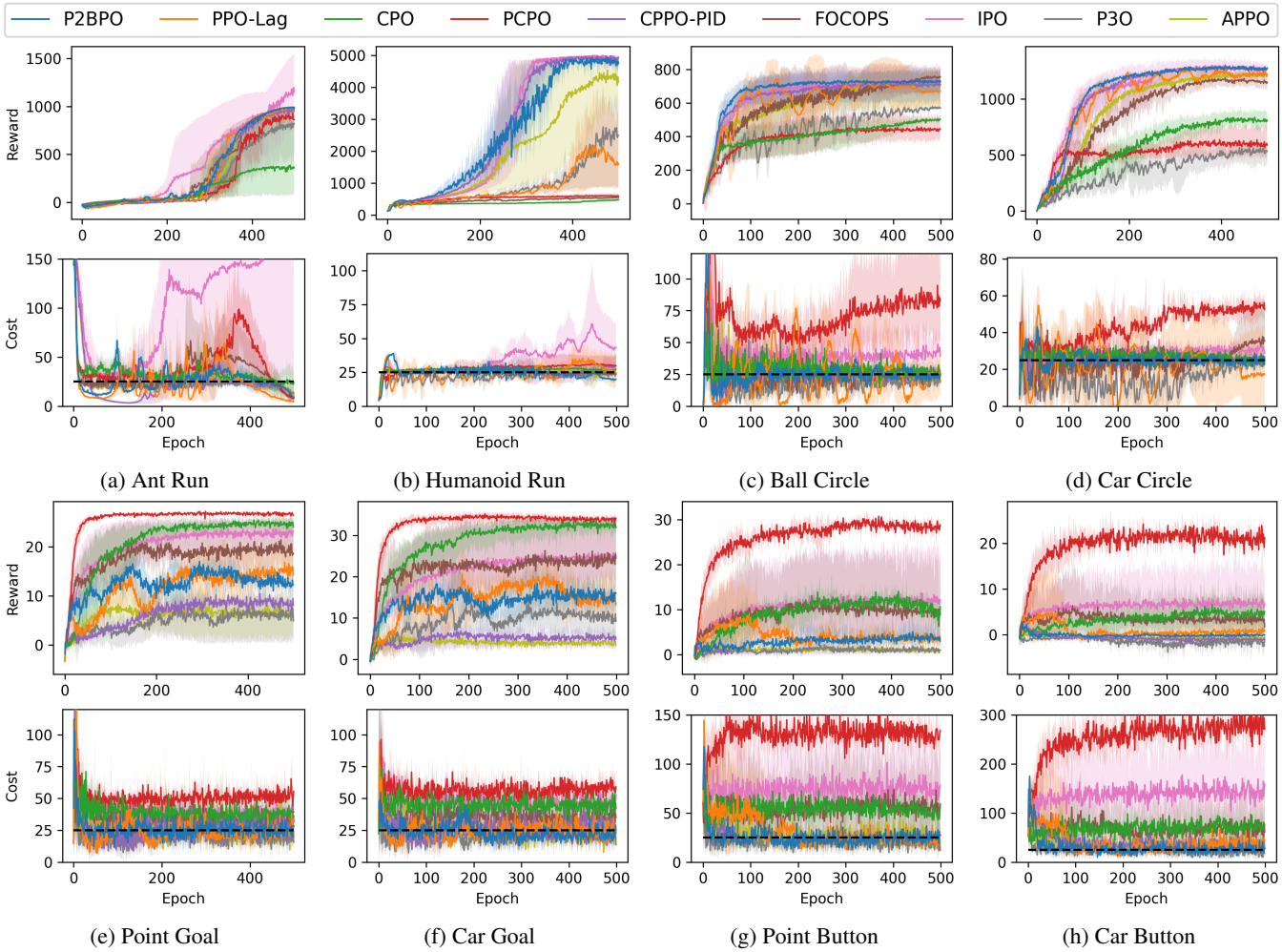
- **Convex Optimization based methods:** CPO (Achiam et al. 2017), FOCOPS(Zhang, Vuong, and Ross 2020a), PCPO(Yang et al. 2020).
- **Lagrangian multiplier based methods:** PPO-Lag(Achiam and Amodei 2019), CPPO-PID(Stooke, Achiam, and Abbeel 2020), APPO (Dai et al. 2023)
- **Penalty based methods:** IPO (Liu, Ding, and Liu 2019) and P3O(Zhang et al. 2022).

Considered Environments: To compare our method with the baselines, we considered four different environments from the three most popular safe RL benchmark environments: Safety Gym (Ray, Achiam, and Amodei 2019), Bullet Safety Gym(Gronauer 2022), and Safe MuJoCo(Zhang, Vuong, and Ross 2020a). All these four environments are categorized into four specific tasks. For each task, we train two different kinds of agents. So, we have a total of eight different environment configurations. The details of the environments and hyperparameters are given in Appendix B.

Results Discussion

As discussed earlier, we have trained our P2BPO and other baseline-based RL agents in eight different environments across four different tasks. We run for 500 epochs for each experiment, with 30,000 steps in each epoch over three different seeds. In Figure 3, we have plotted the epoch vs. reward and cost graphs. We can observe that, except for the initial few epochs, our P2BPO-based algorithm followed the defined cost limit. In comparison, baselines like IPO, CPO, and PCPO struggle to contain the cost within the defined limit in many environments. On the other hand, the cost graphs of the PPO-Lagrangian-based method look like the sinusoidal graph.

We have tested each trained policy for 128 episodes. In Table 1, we present the mean reward and cost along with each SRL algorithm's 95% confidence interval for the discussed environments. We highlighted (bold) the best-performing algorithms, which means the algorithm that collected the highest reward while also satisfying the cost criteria. Our P2BPO performs the best in terms of average reward while satisfying the cost in seven out of eight environments. In the case of the "PointGoal1" environment, P2BPO



(a) Ant Run (b) Humanoid Run (c) Ball Circle (d) Car Circle

(e) Point Goal (f) Car Goal (g) Point Button (h) Car Button

Figure 3: Training time Epoch(x-axis) vs. Reward and Cost (y-axis) for various SRL environments. The black dotted line in the Cost graphs represents the defined cost limit.

marginally misses(+2) the cost boundary. Our P2BPO algorithm collects an average of 7% better reward than PPO-Lagrangian, PCPO, FOCOPS, and IPO while an average of 115% better reward than the rest considering the environments where the baselines satisfy the cost limit.

Cost Limit Compliance: In Figure 4, we show the performance of P2BPO w.r.t Reward and Cost for SafeCarCircle environment over the cost limits of 25, 50, and 100. From the figures, we can observe that the P2BPO-based algorithm obeys the cost limit. It can also be noticed that as the cost limit increases, the average episodic reward also increases.

Sensitivity to Permeability Parameter (β): As discussed in Section , with the lesser value of Permeability Parameter (β), P2BPO algorithm becomes more and more conservative w.r.t. cost, and therefore the algorithm performance w.r.t. reward is also impacted. However, as the β value is a hyperparameter and manually set, the high impact on algorithm performance due to a slight change in the β value is not desirable. In Figure 5, we have plotted the reward and cost graph of the P2BPO algorithm for the Humanoid Run environment for β values 1, 10, 15, and 100. With $\beta = 1$, it can be observed that the cost graph algorithm is like a sinusoidal wave or more like a Lagrangian method; the algorithm behaves conservatively in terms of cost, and the rewards performance is also bad compared to other β values. In contrast, with $\beta = 100$ (softplus function behaves more like ReLU), the cost graph of the algorithm fluctuates very much near the cost line, like a penalty-based method. However, the cost graph for the $\beta = 10$ is balanced compared to the latter two, and $\beta = 10$ also achieves a better reward graph. Now, for $\beta = 10$, with a small change, P2BPO performs quite the same in terms of cost and reward.

Related Works

This section will discuss some of the relevant prior literature.

Penalty based methods: The penalty-based method is a class of first-order methods that uses penalty functions to satisfy the constraints. In IPO (Liu, Ding, and Liu 2019), the authors use the logarithmic barrier function as a penalty function to limit the set of feasible policies. However, this

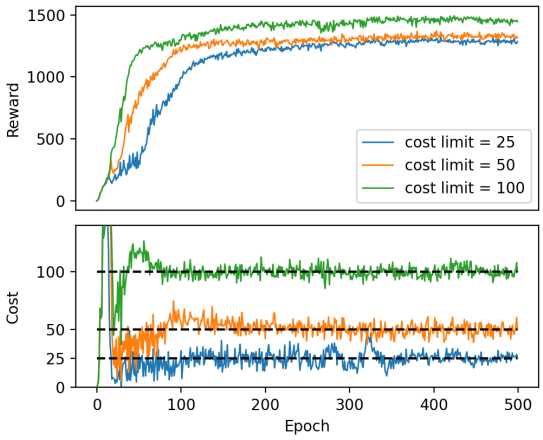


Figure 4: Epoch vs. Reward and Cost graph for SafeCarCircle environment over different cost limits.

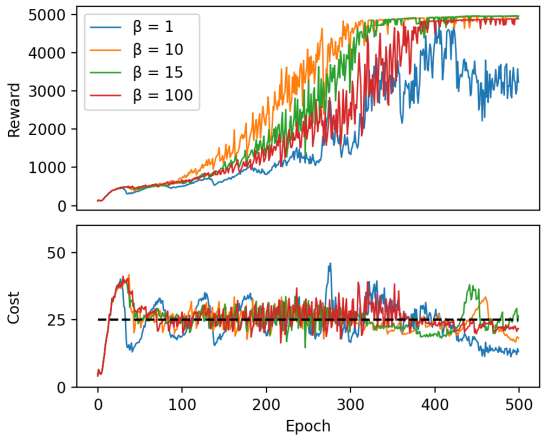


Figure 5: Epoch vs. Reward and Cost graph for Humanoid Run environment over different Permeability Parameter(β)

method assumes a feasible policy upon initialization, which is unrealistic and needs further recovery. Additionally, when the penalty term approaches infinity, the solution might not perfectly match the primary problem. Another penalty-based method, P3O (Zhang et al. 2022), uses the exact penalty method. This method requires a large penalty parameter. The authors also have provided a theoretical lower bound of penalty parameters, larger than the optimal dual parameters of the primal problem, to ensure exactness (Freund 2004). However, this may result in more substantial estimation errors.

Convex Optimization-based methods: Constrained Policy Optimization (CPO) (Achiam et al. 2017) is a Trust Region Policy Optimization (TRPO) (Schulman et al. 2015) based method that searches for a safe and feasible policy in the trust region. CPO solves the quadratic optimization problem with appropriate approximations to ensure monotonic performance improvement while ensuring constraint satisfaction. Another CPO-like method can be found in Projection-based CPO (PCPO) (Yang et al. 2020). In PCPO, the policy is updated in two stages. In the first stage, it performs a TRPO-like policy update, and then in the second stage, the policy is updated to project it back into the constraint set. The key issues with those local policy search-based methods include approximate mistakes, the inversion of large Hessian matrices, and their limited scalability to numerous constraints. To address these, FOCOPS (Zhang, Vuong, and Ross 2020a) first solves the constrained policy optimization in the non-parametric space. In the second stage, it calculates the first-order gradients of the l2-loss function within the parameter space. However, with the growing number of restrictions, FOCOPS has more auxiliary variables to learn than our first-order optimization objective.

Lagrangian based methods: Lagrangian multiplier-based constraint optimizations are popular first-order constraint optimization methods and are well-known for simple implementations. In (Achiam and Amodei 2019), the authors provide implementation details of PPO-Lagrangian and TRPO-Lagrangian-based algorithms for Safety Gym environments. CPPO-PID (Stooke, Achiam, and Abbeel 2020) proposes another novel Lagrangian-based SRL algorithm, which uses the PID control-based method for Lagrangian multiplier update. At the same time, it helps to dampen cost oscillation but is hard to implement and tune due to its sensitivity to three PID-based hyperparameters. APPO (Dai et al. 2023) proposes another PPO-Lagrangian-based method with an additional quadratic term in the objective function, which helps to dampen initial cost peaks. Other than these, various Primal-Dual Optimization (PDO)-based (Le, Voloshin, and Yue 2019; Liang, Que, and Modiano 2018; Chow et al. 2017; Paternain et al. 2019; Tessler, Mankowitz, and Mannor 2018) algorithms are proposed for SRL. These methods use Lagrangian duality to satisfy the constraints while learning the safe policies in various SRL environments. One common issue of these methods is they are sensitive to the Lagrangian multiplier's initial values and learning rate.

P2BPO is a penalty-based optimization method that extends proximal policy with an adaptive penalty. The novelty is in ensuring the smoothness of the penalty incurred. Various other methods in the Safe Reinforcement Learning domain can be found in survey paper (García and Fernández 2015) and most recently (Gu et al. 2022).

Conclusion

This paper proposes a novel permeable penalty barrier-based approach (P2BPO) for CMDP-based SRL scenarios to learn policies that maximize reward while obeying the defined cost limit. We also discussed performance guarantee bounds for our proposed approach. Further, we empirically demonstrated that P2BPO outperforms existing techniques by a significant margin regarding rewards (while satisfying the cost limit). Our proposed easy-to-implement and easy-to-tune (hyperparameters) method helps to mitigate the trade-off between performance (reward) and safety (cost) for other real-world SRL applications. A possible future aspect of this work is to investigate the possibilities of using an adaptive permeability parameter (β) instead of the current fixed-valued one.

References

Achiam, J.; and Amodei, D. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning.

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained Policy Optimization. In *International Conference on Machine Learning*.

Altman, E. 1999. Constrained Markov Decision Processes.

Chow, Y.; Ghavamzadeh, M.; Janson, L.; and Pavone, M. 2017. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *J. Mach. Learn. Res.*, 18(1): 6070–6120.

Dai, J.; Ji, J.; Yang, L.; Zheng, Q.; and Pan, G. 2023. Augmented Proximal Policy Optimization for Safe Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Freund, R. M. 2004. Penalty and barrier methods for constrained optimization. Lecture Notes, Massachusetts Institute of Technology.

García, J.; and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16: 1437–1480.

Gronauer, S. 2022. Bullet-Safety-Gym: A Framework for Constrained Reinforcement Learning. Technical report, mediaTUM.

Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; Yang, Y.; and Knoll, A. 2022. A Review of Safe Reinforcement Learning: Methods, Theory and Applications. *ArXiv*, abs/2205.10330.

Kakade, S. M.; and Langford, J. 2002. Approximately Optimal Approximate Reinforcement Learning. In *International Conference on Machine Learning*.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Le, H. M.; Voloshin, C.; and Yue, Y. 2019. Batch Policy Learning under Constraints. *ArXiv*, abs/1903.08738.

Liang, Q.; Que, F.; and Modiano, E. H. 2018. Accelerated Primal-Dual Policy Optimization for Safe Reinforcement Learning. *ArXiv*, abs/1802.06480.

Liu, Y.; Ding, J.; and Liu, X. 2019. IPO: Interior-point Policy Optimization under Constraints. In *AAAI Conference on Artificial Intelligence*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *ArXiv*, abs/1312.5602.

Paternain, S.; Chamon, L.; Calvo-Fullana, M.; and Ribeiro, A. 2019. Constrained Reinforcement Learning Has Zero Duality Gap. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

PKU-Alignment. 2023. Safe Policy Optimization (SafePO). https://github.com/PKU-Alignment/Safe-Policy-Optimization.git. Accessed: 2023-06-01.

Ray, A.; Achiam, J.; and Amodei, D. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M. I.; and Moritz, P. 2015. Trust Region Policy Optimization. In *International Conference on Machine Learning*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *ArXiv*, abs/1707.06347.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T. P.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362: 1140 – 1144.

Stooke, A.; Achiam, J.; and Abbeel, P. 2020. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. *ArXiv*, abs/2007.03964.

Sutton, R. S.; and Barto, A. G. 1998. Reinforcement Learning: An Introduction. *IEEE Trans. Neural Networks*, 9: 1054–1054.

Sutton, R. S.; McAllester, D. A.; Singh, S.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2018. Reward Constrained Policy Optimization. *ArXiv*, abs/1805.11074.

van Otterlo, M.; and Wiering, M. A. 2012. Markov Decision Processes: Concepts and Algorithms.

Wah, B.; Wang, T.; Shang, Y.; and Wu, Z. 1997. Improving the performance of weighted Lagrange-multiplier methods for nonlinear constrained optimization. In *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, 224–231.

Yang, T.-Y.; Rosca, J.; Narasimhan, K.; and Ramadge, P. J. 2020. Projection-Based Constrained Policy Optimization. In *International Conference on Learning Representations*.

Zhang, L.; Shen, L.; Yang, L.; Chen, S.-Y.; Yuan, B.; Wang, X.; and Tao, D. 2022. Penalized Proximal Policy Optimization for Safe Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*.

Zhang, Y.; Vuong, Q.; and Ross, K. 2020a. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33: 15338–15349.

Zhang, Y.; Vuong, Q. H.; and Ross, K. W. 2020b. First Order Optimization in Policy Space for Constrained Deep Reinforcement Learning. *ArXiv*, abs/2002.06506.