Gradient-Adaptive Pareto Optimization for Constrained Reinforcement Learning

Zixian Zhou[*,1,2], **Mengda Huang**[*,1], **Feiyang Pan**[†,3], **Jia He**[3], **Xiang Ao**[†,1,2,4], **Dandan Tu**[3], **Qing He**[1,2]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Huawei EI Innovation Lab
[4] Institute of Intelligent Computing Technology, Suzhou, CAS
{zhouzixian21s, huangmengda19s}@ict.ac.cn

Abstract

Constrained Reinforcement Learning (CRL) burgeons broad interest in recent years, which pursues maximizing long-term returns while constraining costs. Although CRL can be cast as a multi-objective optimization problem, it is still facing the key challenge that gradient-based Pareto optimization methods tend to stick to known Pareto-optimal solutions even when they yield poor returns (e.g., the safest self-driving car that never moves) or violate the constraints (e.g., the record-breaking racer that crashes the car). In this paper, we propose Gradient-adaptive Constrained Policy Optimization (GCPO for short), a novel Pareto optimization method for CRL with two adaptive gradient recalibration techniques. First, to find Pareto-optimal solutions with balanced performance over all targets, we propose *gradient rebalancing* which forces the agent to improve more on under-optimized objectives at every policy iteration. Second, to guarantee that the cost constraints are satisfied, we propose *gradient perturbation* that can temporarily sacrifice the returns for costs. Experiments on the *SafetyGym* benchmarks show that our method consistently outperforms previous CRL methods in reward while satisfying the constraints.

1 Introduction

Due to the close relationship with real-world applications, Constrained Reinforcement Learning (CRL) has burgeoned broad interests in recent years (Achiam et al. 2017; Ding et al. 2020; Wachi and Sui 2020; Satija, Amortila, and Pineau 2020). Unlike unconstrained reinforcement learning which only aims at maximizing cumulative returns, CRL pursues rewards while satisfying specific constraints. For example, in the scenario of auto-pilot, the well-trained agent should arrive at the destination and meets the safety constraints simultaneously (Kong, Zhang, and Xu 2021).

CRL problems are usually formulated as Constrained Markov Decision Processes (CMDPs) (Altman 1999), which incorporate rewards and constraints into the same framework. Aside from returning a scalar reward after each action like conventional MDPs, CMDPs send back one or multiple cost signals independent of reward. Constraints are expressed explicitly in CMDPs by limiting the expected sum of each cost in the corresponding region.

Essentially, the purpose of CRL is to maximize rewards while controlling costs, which would be naturally associated with Multi-objective optimization. In recent years, Pareto approaches (Sener and Koltun 2018; Lin et al. 2019), which find a steep gradient that benefits all objectives, have been generally leveraged to multi-objective optimization. The ultimate goal of Pareto approaches is to find a Pareto-optimal (Pareto 1897) solution, in which no objective can be advanced without harming any other objectives.

Adhering to this idea, the CRL problem can be transformed into a search for a specific Pareto-optimal policy, which should also remain in the feasible region restricted by constraints. However, algorithms for CRL seldom consider experience from Pareto optimization area because existing Pareto approaches perform poorly in practical CRL problems (Tessler, Mankowitz, and Mannor 2018; Ray, Achiam, and Amodei 2019). Although existing Pareto approaches are able to find Pareto-optimal policy effectively, they not only overlook the constraints, but also yield imbalanced performance in terms of rewards and costs. As shown in Fig.1(b), when the gradients of two objective functions disagree, traditional Pareto approaches will synthesize an updating vector which is biased to the shorter gradient. Eventually, policy updated in such biased direction will be either too risky to be aware of the constraint or too conservative to interact with the environment. A more detailed theoretical explanation is presented in Section 4.

To tackle these challenges, we propose a practical algorithm for CRL named GCPO (abbr. for Gradient-adaptive Constrained Policy Optimization). Beyond previous Pareto methods, GCPO employs two gradient recalibration techniques. First, the *gradient rebalancing* mechanism redefines the length of gradients of each objective and ensures that the agent focuses more on the underdeveloped objective, leading to more balanced development. Second, to achieve feasible solutions, we use *gradient perturbation* to force the agent to control the costs when the constraints are not satisfied. Experimental re-

[*]These authors contributed equally.

[†]Correspondence to Feiyang Pan (pfy824@gmail.com) and Xiang Ao (aoxiang@ict.ac.cn).

sults on the widely used benchmark *SafetyGym* (Ray, Achiam, and Amodei 2019) demonstrate the superiority of the proposed method. The contributions are summarized as follows:

- We theoretically analyze the obstacles to applying existing gradient-based Pareto methods to the CRL framework.
- We design a practical algorithm for CRL named GCPO with two novel gradient recalibration techniques.
- We conduct extensive experiments on the *SafetyGym* environment, the results of which demonstrate the superiority of GCPO compared to the commonly used baselines.

2 Related Work

Constrained Reinforcement Learning

Constrained Reinforcement Learning is a generalized RL with additional constraints in the environment. Conventionally, CRL is formulated as CMDP (Altman 1999), in which the environment returns both a reward and non-negative costs state-wise. CRL is broadly leveraged in several real-world applications, such as networks (Hou and Zhao 2017), smart grids (Gao et al. 2020), and robotics (Dalal et al. 2018). Among all CRL scenarios, safety is the most common constraint. Safety CRL (Sui et al. 2015; Wachi et al. 2018) has more strict demand in constraints, which also raises the problem of safe exploration(Moldovan and Abbeel 2012).

Mainstreams of the CRL literature are (i) Lagrangian methods (Borkar 2005; Tessler, Mankowitz, and Mannor 2018; Stooke, Achiam, and Abbeel 2020); (ii) Trust Region methods (Achiam et al. 2017; Yang et al. 2019). Besides, model-based CRL methods (Chow et al. 2017; Berkenkamp et al. 2017; Wachi and Sui 2020) are worthy of being mentioned, which guarantee agents to explore in states which are traceable from known states with low costs. Notably, Chow et al. (2018, 2019) utilizes Lyapunov functions to analyze the stability of dynamical systems as the constraints of safety;Tessler, Mankowitz, and Mannor (2018) introduces a penalty term into Lagrangian function for infeasibility, thus making infeasible solutions sub-optimal. However, most model-based CRL algorithms are restricted to discrete-action domains due to their value-based modeling methods of environments.

Pareto Optimization

Besides the approaches focusing on gradients, some other Pareto approaches attempt to solve the Pareto Optimization problem by searching the Pareto frontier directly (Yang, Sun, and Narasimhan 2019; Xu et al. 2020). Since this scheme is much slower in complicated and continuous environments, we mainly focus on gradient-based Pareto optimization methods in this paper.

At present, Pareto optimizing (Fliege and Svaiter 2000; Désidéri 2012) provides a novel and time-economical way to solve the Multi-Objective Optimization Problem (MOOP) by updating in the gradient descent direction that benefits all objectives. Such direction is determined by a linear combination of gradients of each objective, whose weights, called Pareto weights, are computed alongside the training process. The Pareto weights have empirical formula only when the number of the objectives is less than four. Under such circumstances, importing Pareto approaches would bring only $\mathbf{O}(1)$ additional time complexity to the original algorithm. Sener and Koltun (2018) first adapted the Pareto optimizer in Désidéri (2012) to deep learning by designing an approximate solver of Pareto weights. Similarly, Lin et al. (2019) improved Fliege and Svaiter (2000) in order to comply with multi-objective optimization problems with preferred vector.

In Appendix B, we first re-elaborate the algorithms proposed in Fliege and Svaiter (2000) and Désidéri (2012) under the CRL framework. Furthermore, we provide concise proof for their effectiveness in finding Pareto direction and show that they are fundamentally identical in CRL problems.

3 Preliminary

Constrained Markov Decision Process

Markov Decision Process A normal Markov Decision Process (Sutton and Barto 1998) can be described as a quadruple (S, A, P, R). Precisely, S denotes the state set; A denotes the action set; P is the distribution returning the probability of transiting to s' assuming we take action a in s, denoted as $P(s'|s,a)$; $R : S \times A \times S \to \mathbb{R}$ is the reward function, which delivers reward r as soon as the transition $s \to s'$ is accomplished. We make decisions for choosing actions by a policy $\pi : S \to \Delta_A$, which returns a distribution over A. In this work, we parameterize our policy π_ω by a neural network with parameters $\omega \in \mathbb{R}^k$.

In an MDP, we take an action $a \sim \pi$ from initial state $s_0 \sim \rho_0(s_0)$ iteratively, and transit to a new state according to P, yielding a finite or infinite trajectory $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \dots) \sim \pi$. Given a policy π, we are able to evaluate the goodness of a state or action by state-value function $V(s)$, action-value function $Q(s,a)$, and advantage-value function $A(s,a)$:

$$Q_R^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1},a_{t+1},\dots}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l}\right],$$
$$V_R^\pi(s_t) = \mathbb{E}_{a_t}\left[Q_R^\pi(s_t,a_t)\right],\ A_R^\pi(s,a) = Q_R^\pi(s,a) - V_R^\pi(s). \tag{1}$$

In Eq (1), $\gamma \in [0,1]$ is the discount factor, which weighs the future reward and instant reward. The ultimate goal of reinforcement learning is to discover an optimal policy π^* for MDP by solving:

$$\arg\max_\pi \mathbb{E}_{s_0\sim\rho_0,\tau\sim\pi}\left[V_R^\pi(s_0)\right]. \tag{2}$$

Constrained MDP In this paper, we concentrate on CMDP with only one kind of cost, which is consistent with the settings in Achiam et al. (2017); Tessler, Mankowitz, and Mannor (2018); Yang et al. (2019). The main difference between CMDP and MDP is that CMDP has an extra cost function $C : S \times A \times S \to \mathbb{R}$. Therefore, the feedback of the environment in one transition is a vector $(r, c) \in \mathbb{R}^2$, where $c \in \mathbb{R}_+$ is the value of cost. Similarly, we have value functions for cost: $V_C^\pi(s_t), Q_C^\pi(s_t,a_t), A_C^\pi(s,a)$ by switching the reward r to the cost c in Eq (1).

Formally, the policy optimization problem in CMDP is

$$\begin{aligned} \max_\pi\ & J_R(\pi) = \mathbb{E}_{s_0\sim\rho_0,\tau\sim\pi}\left[V_R^\pi(s_0)\right], \\ \text{s.t.}\ & J_C(\pi) = -\mathbb{E}_{s_0\sim\rho_0,\tau\sim\pi}\left[V_C^\pi(s_0)\right] \geq \zeta, \end{aligned} \tag{3}$$

where $\zeta < 0$ denotes the predefined constraint threshold. In Eq (3), $J_R(\pi)$ and $J_C(\pi)$ represent the objective functions of π with respect to reward and cost, respectively. Specifically, here we let $J_C(\pi)$ negative for readability.

Pareto-optimal in CRL

To understand what is a Pareto-optimal policy, we need a rule to compare which policy is better.

Definition 3.1 (Dominate)**.** For two policies π and π', we say that π **dominates** π', denoted as $\pi \succ \pi'$ if and only if $J_R(\pi) \geq J_R(\pi'), J_C(\pi) \geq J_C(\pi')$ and at least one inequation strictly holds.

By Definition 3.1, we know that a policy π is better than π' when π is not worse than π' over reward and cost, and outperforms π' on at least one objective.

Definition 3.2 (Pareto-optimal Policy)**.** We call a policy π_ω **global Pareto-optimal** if and only if $\forall \omega' \in \mathbb{R}^k, \pi_{\omega'} \succ \pi_\omega$ is invalid, i.e. $\forall \omega' \in \mathbb{R}^k, \pi_{\omega'} \not\succ \pi_\omega$. Similarly, we call a policy π_ω **local Pareto-optimal** if and only if there exists a neighborhood $U \subset \mathbb{R}^k$ of ω s.t. $\forall \omega' \in U, \pi_{\omega'} \succ \pi_\omega$ is invalid, i.e. $\forall \omega' \in U, \pi_{\omega'} \not\succ \pi_\omega$.

Without additional specification, we use Pareto-optimal referring to local Pareto optimal hereafter. A local Pareto-optimal policy is guaranteed to be a global Pareto-optimal policy only if $J_R(\pi_\omega)$ and $J_C(\pi_\omega)$ are concave over ω in $\mathbb{R}^k$. However, this assumption is too strict to be valid in most Deep RL setting.

From the angle of optimizing multi-targets, we notice that if π is a Pareto-optimal policy we cannot advance any objective function while maintaining the performance of the other. This leads to the definition of Pareto direction:

Definition 3.3 (Pareto direction)**.** Given a parameterized policy π_ω, if a vector $\mathbf{v} \in \mathbb{R}^k$ is an updating direction of ω which can boost at least one $J(\pi_\omega)$ without harming the other $J(\pi_\omega)$, then $\mathbf{v}$ is a Pareto direction of π_ω. Namely, if $\langle \nabla_\omega J_R(\pi_\omega), \mathbf{v}\rangle \geq 0, \langle \nabla_\omega J_C(\pi_\omega), \mathbf{v}\rangle \geq 0$ and at least one inequation strictly holds, then we call $\mathbf{v}$ a **Pareto direction** of π, where $\langle\, ,\, \rangle$ is the standard inner product.

Pareto-optimal Searching To search for a Pareto-optimal policy, the most straightforward idea is to design an iteration $\omega' = \omega + \eta(\omega)\Delta(\omega)$, where $\Delta(\omega)$ is a Pareto direction of ω, and $\eta(\omega) \in \mathbb{R}_+$ is the step size. With appropriate η, we could ensure $\pi_{\omega'} \succ \pi_\omega$ at each update until π_ω is Pareto-optimal. Intuitively, $\Delta(\omega)$ should be a linear combination of gradients of $J_R(\pi_\omega)$ and $J_C(\pi_\omega)$, i.e. $\Delta(\omega) = \beta_R \nabla_\omega J_R(\pi_\omega) + \beta_C \nabla_\omega J_C(\pi_\omega)$, where β_R and β_C are called **Pareto weights**.

Both Fliege and Svaiter (2000) and Désidéri (2012) are established works for Pareto-optimal searching. Under the problem settings of CRL, the Pareto weights are obtained in Désidéri (2012) by solving the following Quadratic Programming (QP):

$$\begin{aligned} \min_{\beta_R, \beta_C \in \mathbb{R}} \; & \|\beta_R \nabla_\omega J_R(\pi_\omega) + \beta_C \nabla_\omega J_C(\pi_\omega)\|_2^2 \\ s.t. \;\; & \beta_R, \beta_C \geq 0, \;\; \beta_R + \beta_C = 1. \end{aligned} \tag{4}$$

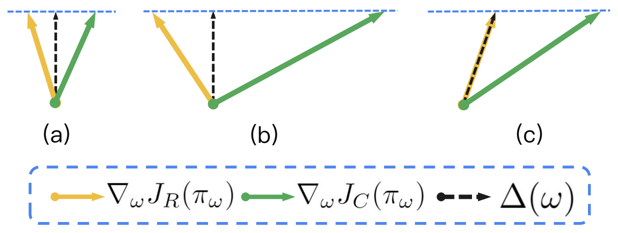
(a)
(b)
(c)
$\nabla_\omega J_R(\pi_\omega)$
$\nabla_\omega J_C(\pi_\omega)$
$\Delta(\omega)$

Figure 1: An illustration of Pareto direction $\Delta(\omega)$.
(a): $\Delta(\omega)$ generalizes both gradients when they have similar directions.
(b): $\Delta(\omega)$ is biased to the shorter gradient when $\nabla_\omega J_R(\pi_\omega)$ and $\nabla_\omega J_C(\pi_\omega)$ disagree in direction.
(c): $\Delta(\omega)$ coincides with one of the gradients when the perpendicular vector does not lie between $\nabla_\omega J_R(\pi_\omega)$ and $\nabla_\omega J_C(\pi_\omega)$

Figure 1 illustrates the geometric relationship among $\nabla_\omega J_R(\pi_\omega)$,$\nabla_\omega J_C(\pi_\omega)$ and $\Delta(\omega)$ obtained by solving Problem (10) in three cases. If we overlap the starting points of $\nabla_\omega J_R(\pi_\omega)$ and $\nabla_\omega J_C(\pi_\omega)$, then $\Delta(\omega)$ should be the vector that starts from the same point and ends on the line segment determined by endpoints of $\nabla_\omega J_R(\pi_\omega)$ and $\nabla_\omega J_C(\pi_\omega)$ as a result of the constraints in Problem (10). Since we minimize the length of $\Delta(\omega)$, $\Delta(\omega)$ is normally the perpendicular vector of $\nabla_\omega J_R(\pi_\omega) - \nabla_\omega J_C(\pi_\omega)$ (See Figure 1 (a) and (b)). However, it coincides with one of the gradient vectors when the perpendicular vector does not lie between $\nabla_\omega J_R(\pi_\omega)$ and $\nabla_\omega J_C(\pi_\omega)$ as shown in Figure 1 (c).

Connections to Prior CRL Works

As aforementioned, the mainstream approaches to CRL problems could be grouped into two genres: (i) Lagrangian methods (Borkar 2005; Tessler, Mankowitz, and Mannor 2018; Stooke, Achiam, and Abbeel 2020); (ii) Trust Region methods (Achiam et al. 2017; Yang et al. 2019). Lagrangian methods combine primal reward-oriented and cost-oriented objectives into one dual min-max problem, converting CMDPs into unconstrained MDPs, while Trust Region methods attempt to determine a trust region in the parameter space, where policies updated inside would not violate the constraints.

Prior CRL algorithms fundamentally aim at finding Pareto-optimal policy (See proofs in Appendix C), which unifies our work and existing works in methodology. Nevertheless, not every Pareto-optimal policy makes sense in a CMDP. What we truly need is a Pareto-feasible policy:

Definition 3.4 (Pareto-feasible Policy)**.** We call a policy π_ω **Pareto-feasible** if π_ω is Pareto-optimal while satisfying the constraint of cost.

Figure 2 (a), (b), and (c) illustrate how prior CRL algorithms search for Pareto-optimal policy. In Figure 2 (a), we scalarize (r, c) per transition with random or preset weights linearly. With a diverse selection of weights, the final policy may be different. Under such conditions, the searching route of linear scalarization could only reach a Pareto-feasible policy with certain weights (route 2). As shown in Figure 2 (b),

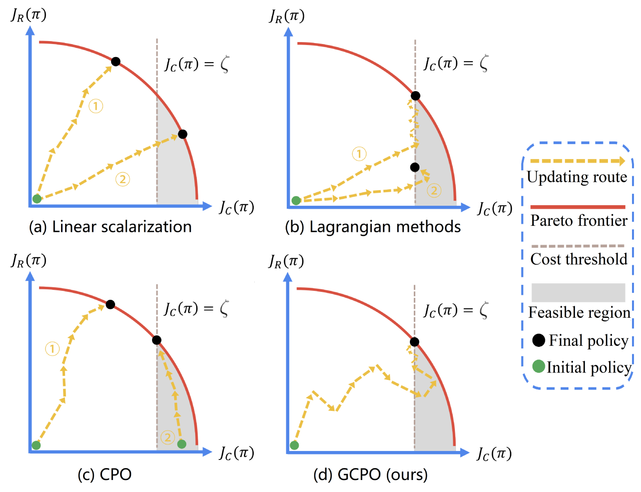

Figure 2: An illustration of how various methods find Pareto-optimal policies. (a) Linear scalarization; (b) Lagrangian methods; (c) CPO; (d) GCPO.

Lagrangian methods ensure that $J_C(\pi)$ is stable near the threshold but they may fail to find the Pareto-optimal policy because they can be too conservative to explore so that they stick to a feasible solution and fail to reach Pareto-optimal(route 2). For CPO (Achiam et al. 2017) in Figure 2 (c), it is able to maintain the policy within the feasible region while increasing $J_R(\pi)$. Yet, it may violate the constraint when the initial policy is too far from the feasible region (route 2). Results in experiments and theoretical proofs in Appendix C could corroborate that the interpretation above is not heuristic. Figure 2 (d) illustrates that our method (GCPO) can eventually find a Pareto-feasible policy, although not every step is a Pareto direction.

4 Methodology

In this section, we first introduce the challenges of applying Pareto methods in CRL and theoretically explain what triggers these disadvantages. Then we present a detailed statement about two gradient recalibration mechanisms used in our algorithm (GCPO) and how they effectively overcome the disadvantages.

Challenges of Pareto-optimal Searching in CRL

Although the existing Pareto methods can find Pareto-optimal policies effectively, we found two main disadvantages from practice when applying them in CRL.

First, existing Pareto-optimal algorithms improve all objectives with a consistent extent (See Theorem 4.2), leading to an **imbalanced performance** of $J_R(\pi_\omega)$ and $J_C(\pi_\omega)$. As shown in Figure 1 (b), when the gradients of the objective functions of reward and cost disagree, $\Delta(\omega)$ determined by Problem (10) is biased to the shorter gradient (i.e.$\nabla_\omega J_R(\pi_\omega)$) in direction. This fact implies that updating in such direction would focus on reward, which is already the better-optimized objective compared to cost. In extreme cases, if Pareto-optimal searching approaches or reaches a Pareto-optimal policy which is not feasible, it has no chance to escape and fails to find a Pareto-feasible policy. Generally, policy updated by this biased gradient will be either too risky to be aware of the constraint of cost or too conservative to interact with the environment, which eventually generates an imbalanced performance in rewards and costs.

Second, existing Pareto-optimal approaches **overemphasize the simultaneous growth** of two objectives. Although optimizing both objectives is significant in MOOP, it prevents the agent from satisfying the constraint in CRL. In fact, the agent in CRL needs to sacrifice $J_R(\pi_\omega)$ for the improvement of $J_C(\pi_\omega)$ when the constraint is violated, which is not possible in traditional Pareto methods.

Gradient Recalibration Mechanisms

Gradient Rebalancing This is the corresponding improvement for the imbalanced development. To tackle this issue, we should modify $\Delta(\omega)$ to achieve a balanced improvement in reward and cost at each iteration of policy parameters. Motivated by recent works about gradient normalization (Chen et al. 2018; Mahapatra and Rajan 2020), we reform Problem (10) as:

$$\begin{aligned} \min_{\beta_R,\beta_C\in\mathbb{R}} \; & ||\beta_R\nabla^N_\omega J_R(\pi_\omega) + \beta_C\nabla^N_\omega J_C(\pi_\omega)||_2^2, \\ s.t. \; & \beta_R, \beta_C \geq 0, \; \beta_R + \beta_C = 1, \end{aligned} \tag{5}$$

where $\nabla^N_\omega J_R(\pi_\omega)$ and $\nabla^N_\omega J_C(\pi_\omega)$ are rebalanced gradients defined as:

$$\begin{aligned} \nabla^N_\omega J_R(\pi_\omega) &= \frac{\nabla_\omega J_R(\pi_\omega)}{||\nabla_\omega J_R(\pi_\omega)||_2^2}, \\ \nabla^N_\omega J_C(\pi_\omega) &= \frac{\nabla_\omega J_C(\pi_\omega)}{||\nabla_\omega J_C(\pi_\omega)||_2^2}. \end{aligned} \tag{6}$$

In Eq (5), the length of rebalanced gradient vectors becomes the reciprocal of the original length.With solution $\beta^*_N = (\beta^N_R, \beta^N_C) \in \mathbb{R}^2_+$, we can find a better updating direction comparing to the Pareto direction derived from Problem (10):

Lemma 4.1. *If π_ω is not Pareto-optimal, then $\Delta_N(\omega) := \beta^N_R\nabla^N_\omega J_R(\pi_\omega) + \beta^N_C\nabla^N_\omega J_C(\pi_\omega)$ is a Pareto direction of π_ω.*

Theorem 4.2. *Suppose the iteration paradigm is $\omega' = \omega + \eta(\omega)\Delta(\omega)$ and $\eta(\omega) \to 0$. Then:*
(i) *if we use $\Delta(\omega)$ derived from Problem (10), the improvements in reward and cost are consistent. Specifically, when* $\beta^*_R \in (0,1)$, $\dfrac{J_R(\pi_{\omega'}) - J_R(\pi_\omega)}{J_C(\pi_{\omega'}) - J_C(\pi_\omega)} \to 1.$
(ii) *if we use $\Delta_N(\omega)$ derived from Problem (5), the improvements in reward and cost are proportional to the square length of the corresponding gradient. Specifically, when* $\beta^N_R \in (0,1)$, $\dfrac{J_R(\pi_{\omega'}) - J_R(\pi_\omega)}{J_C(\pi_{\omega'}) - J_C(\pi_\omega)} \to \dfrac{||\nabla_\omega J_R(\pi_\omega)||_2^2}{||\nabla_\omega J_C(\pi_\omega)||_2^2}.$

The proofs of Lemma 4.1 and Theorem 4.2 are provided in Appendix D.

Remark. *Theorem 4.2 has two statements: (i) $\Delta(\omega)$ derived from Problem (10) would lead to imbalanced development. Since the scales of $\nabla_\omega J_R(\pi_\omega)$ and $\nabla_\omega J_C(\pi_\omega)$ are different in most realistic scenes, the ratio equal to 1 represents imbalance. Moreover, if one objective is near-optimal, then this* *ratio will restrict the optimizing space of the other objective. (ii) $\Delta_N(\omega)$ derived from Problem (5) would focus more on the objective with longer gradient vector, which implies this objective is farther from the optima. Meanwhile, $\Delta(\omega)$ could update parameters effectively even when one of the objectives is near-optimal.*

According to Theorem 4.2, updating in the direction of $\Delta_N(\omega)$ enables the agent to focus more on the objective which is underdeveloped.

Gradient Perturbation This is the corresponding technique that forces the agent to sacrifice rewards to satisfy the constraint. A naive idea is only optimizing the objective of costs when the constraint is violated. But this is both inefficient and not realistic because: (i) focusing only on costs may drive the agent to be too conservative to explore; (ii) sometimes optimizing rewards and costs are not competitive, thus absolutely ignoring rewards is unwise.

We find that manipulating Pareto weights can achieve the goal of sacrificing rewards when necessary. If current Pareto direction is unable to improve the performance of cost to a desirable extent, we manage to perturb the updating gradient $\Delta_N(\omega)$ by raising β_C^N. Motivated by PPO (Schulman et al. 2017), we design a mechanism to control Pareto weight: given a clipping threshold $t \in [0, 1]$, the original Pareto weight β_R^N will be clipped to $min(t, \beta_R^N)$. Since $\beta_R^N + \beta_C^N = 1$, clipping β_R^N spontaneously brings an increment in β_C^N.

To choose t with efficiency, we devise a heuristic mechanism which keeps the step size $\eta(\omega)$ as a constant and changes t according to the performance of rewards and costs. If the improvement of cost in one iteration is not good enough, we decrease t by a certain quantity to make a stricter threshold of clipping. Similarly, if the performance of reward has not been improved within certain epochs, we increase t by an identical quantity. We introduce how GCPO trains an agent and modifies t in a pseudo-code, which can be found in Appendix E.

With fixed $\eta(\omega) \to 0$, we can deduce a lower bound for the growth of $J_C(\pi)$:

Theorem 4.3 (The lower bound of $J_C(\pi)$ improvement)**.** *Given the parameter updating paradigm $\omega' = \omega + \eta(\omega)\Delta_N(\omega)$ and a clipping threshold t, we have a lower bound for $J_C(\pi'_\omega) - J_C(\pi_\omega)$:*

$$J_C(\pi_{\omega'}) - J_C(\pi_\omega) \geq \eta(\omega)[t||\nabla_\omega J_C(\pi_\omega)||_2^2 \langle \nabla_\omega^N J_R(\pi_\omega) - \nabla_\omega^N J_C(\pi_\omega), \nabla_\omega^N J_C(\pi_\omega)\rangle + 1] - CD_{\mathrm{KL}}^{\max}(\pi_{\omega'}, \pi_\omega), \tag{7}$$

where $C = 4\epsilon\gamma/(1-\gamma)^2$, $\epsilon = max_s|\mathbb{E}_{a\sim\pi_{\omega'}}[A^{\pi_\omega}(s,a)]|$. And this bound is strictly positive related to t.

Remark. *Theorem 4.3 mainly explains that the gradient perturbation mechanism ensures enough Pareto weight on the underdeveloped objective of cost when β_R^N violates the clipping threshold. Moreover, although $\Delta_N(\omega)$ after perturbation may not be a Pareto direction, it ensures a lower bound for the improvement on $J_C(\pi)$, which is vital to satisfy the constraint of cost.*

The proof of Theorem 4.3 is provided in Appendix D. We also prove that the lower bound is tighter than the situation without clipping. Specifically, this theorem still holds if we swap J_C with J_R.

Practical Implementation

Our method is adaptable and can be applied with any policy-gradient-based RL algorithm. In this paper, we use Actor-critic-based PPO (Schulman et al. 2015) as the base model of GCPO. To estimate value functions for both reward and cost, we have two critic networks to approach $Q_R^\pi(s_t, a_t)$ and $Q_C^\pi(s_t, a_t)$ separately. Following PPO framework, $\nabla_\omega J_R(\pi_\omega)$ and $\nabla_\omega J_C(\pi_\omega)$ are determined as:

$$\begin{aligned} \nabla_\omega J_R(\pi_\omega) &= \frac{\partial \sum_{s,a\sim\tau}[A_R^{\pi_{\text{old}}}(s,a)]}{\partial\omega}, \\ \nabla_\omega J_C(\pi_\omega) &= \frac{\partial \sum_{s,a\sim\tau}[-A_C^{\pi_{\text{old}}}(s,a)]}{\partial\omega}. \end{aligned} \tag{8}$$

5 Experiments

To validate our proposed algorithm, we conduct experiments on *SafetyGym* (Ray, Achiam, and Amodei 2019), a CRL benchmark. We first provide an overview of the tasks and baselines involved in the experiments, and then we demonstrate the significance of GCPO by responding to the following questions:

- **Q1:** How does GCPO perform compared to the baselines?
- **Q2:** How does GCPO adapt to tasks of different levels and cost thresholds?
- **Q3:** How do the two gradient recalibration mechanisms affect the performance of GCPO respectively?

Environmental Setup

Experiments are conducted in three tasks under the environment *SafetyGym* :

- *Goal*: In this task, the agent wins rewards by reaching the destinations (green cylinders) and gains cost for passing traps (blue circles) and hitting vases (cyan cubes).
- *Button*: In this task, the agent wins rewards by pushing a stationary button (orange balls) and gains cost for passing traps (blue circles) and hitting obstacles (purple cubes) with a fixed moving trajectory.
- *Push*: In this task, the agent wins rewards by pushing a crossing workpiece (a yellow column) to a specific destination (green cylinders) and gains cost for passing traps (blue circles) and hitting towers (Blue cylinders).

For each task, there are two difficulty levels: level-2 environments have more cost-consuming items than level-1 environments, making it harder to satisfy the constraint. Each environment is simulated in Mujoco (Todorov, Erez, and Tassa 2012) with a *point*-based agent.

Involved Baselines

We compare GCPO with baselines from three domains: (i) traditional policy-based RL methods (TRPO (Schulman et al. 2015), PPO (Schulman et al. 2017));

Model	Goal-Lvl 1		Goal-Lvl 2		Button-Lvl 1		Button-Lvl 2		Push-Lvl 1		Push-Lvl 2	
	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret
TRPO	56.14	25.12	213.17	23.58	138.36	25.83	172.61	25.59	47.11	7.90	75.16	4.69
PPO	58.09	24.99	198.65	22.27	150.12	26.01	188.17	23.93	67.12	3.41	71.25	2.21
TRPO$_L$	25.47	17.03	25.27	5.49	32.59	8.18	22.51✓	3.73	26.31	4.19	23.38✓	1.30
PPO$_L$	14.21 ✓	13.58	31.66	1.15	23.01✓	4.84	17.99✓	2.38	40.20	2.15	17.27✓	1.52
CPO	42.52	23.21	60.11	14.37	80.25	18.25	74.43	16.78	38.94	7.21	29.37	1.84
MGDA	40.15	25.61	60.91	7.31	54.62	6.02	11.20✓	1.99	16.93✓	0.80	29.77	0.89
GCPO	23.12 ✓	**21.86**	20.84 ✓	**8.39**	22.67 ✓	**9.38**	23.83 ✓	**6.44**	17.17 ✓	**2.39**	20.92 ✓	**2.98**

Table 1: Results of GCPO and other baselines with a constraint of cumulative cost < 25. The cumulative costs below the threshold are followed by a check mark.

Model	Goal-Lvl 1		Goal-Lvl 2		Button-Lvl 1		Button-Lvl 2		Push-Lvl 1		Push-Lvl 2	
	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret
TRPO$_L$	28.01✓	24.57	50.15	8.72	48.97✓	12.90	69.21	7.27	44.72✓	**6.13**	54.41	2.09
PPO$_L$	26.12✓	25.11	63.02	4.73	73.15	10.65	62.02	4.14	33.62✓	3.67	41.74✓	1.62
GCPO	40.68✓	**25.29**	48.41✓	**16.38**	46.57✓	**14.66**	48.81✓	**13.89**	44.71✓	4.15	43.03✓	**2.52**

Table 2: Results of GCPO and other baselines with a constraint of cumulative cost < 50.

Model	Goal-Lvl 1		Goal-Lvl 2		Button-Lvl 1		Button-Lvl 2		Push-Lvl 1		Push-Lvl 2	
	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret	Cost	Ret
GCPO-R	30.80	13.21	50.57	1.51	35.29	5.94	17.21✓	2.18	8.89✓	0.75	17.75✓	1.61
GCPO-P	45.22	17.89	42.49	2.23	52.05	3.90	54.74	3.50	31.11	1.36	28.77	2.25
GCPO	23.12✓	**21.86**	20.84✓	**8.39**	22.67✓	**9.38**	23.83✓	**6.44**	17.17✓	**2.39**	20.92✓	**2.98**

Table 3: Ablation study with a constraint of cumulative cost < 25.

(ii) Constrained RL methods (TRPO-Lagrangian, PPO-Lagrangian, CPO (Achiam et al. 2017)); (iii) Pareto approach (MGDA (Désidéri 2012)). We consider both mean reward and mean cost as two metrics to evaluate the models.

For all methods we conduct 5 runs (1000 episodes each, 10000 steps each episode) with different random seeds and 5×100 episodes of test runs on another 5 random seeds. The threshold for cumulative cost is 25, as recommended in Ray, Achiam, and Amodei (2019). For reproducibility, a detailed statement about the architectures and hyper-parameters is presented in Appendix F.

Comparison Analysis (Q1)

Results of the final tests are listed in Table 1 and learning curves are provided in Figure 3 . To be clear, what we truly need in CRL are feasible policies, therefore any policy that fails to satisfy the constraint is considered inferior to any feasible policy.

Baselines analysis

- Lagrangian methods can find feasible or near-feasible policies in most tasks, but their performances on Lvl 2 tasks are unsatisfactory. This fact shows that Lagrangian methods are too conservative in difficult environments.
- CPO fails to find feasible or even near-feasible policy. This happens because reaching the feasible region in *SafetyGym* is challenging.
- As a deputy of existing Pareto methods, MGDA's final policy not only has extremely imbalanced performance in reward and cost but also fails to meet the constraint except in Button-Lvl 2 and Push-Lvl 1, which coincides with the defects of Pareto methods mentioned in Section 4.

Both Table 1 and Figure 3 confirm that GCPO can find a feasible policy in all tasks while even Lagrangian methods fail to meet the threshold in some Lvl 2 tasks. Moreover, in all tasks, our method outperforms all baselines which can find feasible policies.

In Figure 3, the learning curves illustrate that GCPO's performance on reward improves quickly at the beginning of training and then decreases apparently. The decline in reward indicates the effectiveness of gradient perturbation to sacrifice reward for better performance on cost.

Adaptability Analysis (Q2)

To show the adaptability of GCPO, we compare GCPO with Lagrangian methods under another cost threshold (Table 2).

In Table 2, GCPO loses to TRPO$_L$ in Push-Lvl 1 and then gains its advantage back in Push-Lvl 2. Here we provide an

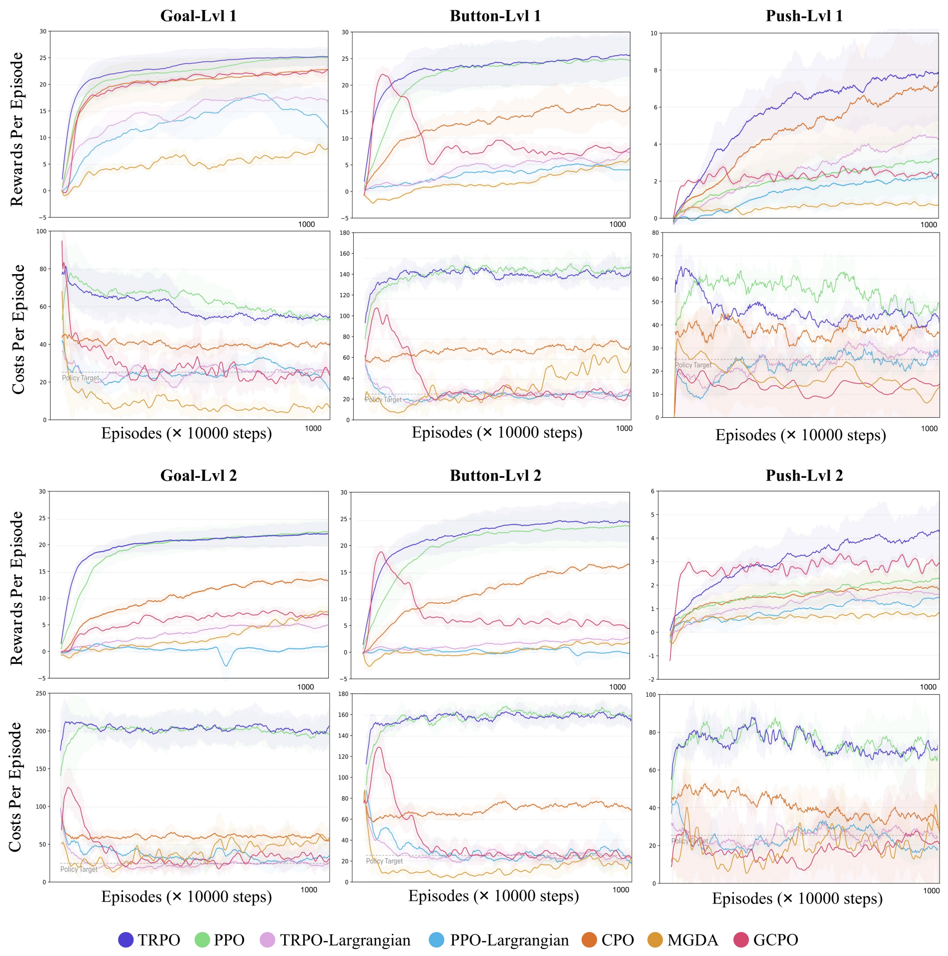

Figure 3: Reward and cost curves in all 6 tasks. The dashed line in cost curves represents the threshold. All lines are averaged over 5 runs and shaded areas indicate one standard deviation.

insight for this interesting phenomenon: the cost threshold in Table 2 is 50, which allows Lagrangian methods to satisfy the constraint more easily and penalizes our recalibration techniques–to sacrifice reward for cost. As the difficulty level upgrades, GCPO performs clearly better than TRPO$_L$, showing that GCPO can handle difficult tasks as well as easy ones. GCPO underperforms TRPO$_L$ in only one task out of all baselines and 6 tasks, which cannot deny its superiority.

In Table 1 and Table 2, GCPO shows an obvious advantage compared to Lagrangian methods in tasks of different levels and thresholds. Moreover, as the threshold goes from 25 to 50, the cumulative cost and reward of GCPO both increase, indicating that GCPO adapts effectively to the relaxed constraint by sacrificing the performance of cost for reward.

Ablation Study (Q3)

We make an ablation study by GCPO-R (without gradient rebalancing) and GCPO-P (without gradient perturbation).

As shown in Table 3, GCPO has significant advantages in both reward and cost performance compared to the ablation baselines, indicating that both gradient recalibration techniques are vital and effective for GCPO.

The overall performance of GCPO-R is much better than GCPO-P. We argue that this happens because gradient perturbation puts a hard restriction to Pareto weights, which is more effective in improving the performance of cost.

6 Conclusion

In this paper, we mainly introduce a novel CRL paradigm named GCPO from the perspective of Pareto optimization. We theoretically analyze that the main challenges of applying existing Pareto approaches in CRL are imbalanced performance over rewards and costs and incapability of reaching Pareto-feasible. Therefore we devise two gradient recalibration techniques. Specifically, gradient rebalancing redefines the calculation of Pareto weights and ensures that the agent focuses more on the underdeveloped objective while gradient perturbation aims to meet the constraints by temporarily sacrificing rewards. Experiments on *SafetyGym* validate the superiority of GCPO over the commonly used baselines and demonstrate its adaptability to tasks of different levels and cost thresholds. In addition, the ablation study verifies the necessity of the two gradient recalibration techniques.

Acknowledgments

The research work was supported by the National Key Research and Development Program of China under Grant No. 2022YFC3303302, the National Natural Science Foundation of China under Grant No.61976204, U1811461, U1836206. Xiang Ao is also supported by the Project of Youth Innovation Promotion Association CAS, Beijing Nova Program Z201100006820062.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *International Conference on Machine Learning*, 22–31. PMLR.

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

Armijo, L. 1966. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1): 1–3.

Berkenkamp, F.; Turchetta, M.; Schoellig, A. P.; and Krause, A. 2017. Safe Model-based Reinforcement Learning with Stability Guarantees. In *NIPS*.

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

Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Chen, Z.; Badrinarayanan, V.; Lee, C.-Y.; and Rabinovich, A. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, 794–803. PMLR.

Chow, Y.; Ghavamzadeh, M.; Janson, L.; and Pavone, M. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1): 6070–6120.

Chow, Y.; Nachum, O.; Duéñez-Guzmán, E. A.; and Ghavamzadeh, M. 2018. A Lyapunov-based Approach to Safe Reinforcement Learning. In *NeurIPS*.

Chow, Y.; Nachum, O.; Faust, A.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2019. Lyapunov-based Safe Policy Optimization for Continuous Control. In *International Conference on Learning Representations*.

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

Désidéri, J.-A. 2012. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6): 313–318.

Ding, D.; Zhang, K.; Basar, T.; and Jovanovic, M. R. 2020. Natural Policy Gradient Primal-Dual Method for Constrained Markov Decision Processes. In *NeurIPS*.

Fliege, J.; and Svaiter, B. F. 2000. Steepest descent methods for multicriteria optimization. *Mathematical methods of operations research*, 51(3): 479–494.

Gao, Y.; Wang, W.; Shi, J.; and Yu, N. 2020. Batch-constrained reinforcement learning for dynamic distribution network reconfiguration. *IEEE Transactions on Smart Grid*, 11(6): 5357–5369.

Hou, C.; and Zhao, Q. 2017. Optimization of web service-based control system for balance between network traffic and delay. *IEEE Transactions on Automation Science and Engineering*, 15(3): 1152–1162.

Kakade, S. M. 2001. A natural policy gradient. *Advances in neural information processing systems*, 14.

Kong, Q.; Zhang, L.; and Xu, X. 2021. Constrained Policy Optimization Algorithm for Autonomous Driving via Reinforcement Learning. In *2021 6th International Conference on Image, Vision and Computing (ICIVC)*, 378–383. IEEE.

Lin, X.; Chen, H.; Pei, C.; Sun, F.; Xiao, X.; Sun, H.; Zhang, Y.; Ou, W.; and Jiang, P. 2019. A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, 20–28.

Mahapatra, D.; and Rajan, V. 2020. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In *International Conference on Machine Learning*, 6597–6607. PMLR.

Moldovan, T. M.; and Abbeel, P. 2012. Safe Exploration in Markov Decision Processes. In *ICML*.

Pareto, V. 1897. The new theories of economics. *Journal of political economy*, 5(4): 485–502.

Platt, J. C.; and Barr, A. H. 1987. Constrained differential optimization. In *Proceedings of the 1987 International Conference on Neural Information Processing Systems*, 612–621.

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

Satija, H.; Amortila, P.; and Pineau, J. 2020. Constrained markov decision processes via backward value functions. In *International Conference on Machine Learning*, 8502–8511. PMLR.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Sener, O.; and Koltun, V. 2018. Multi-task learning as multi-objective optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 525–536.

Stooke, A.; Achiam, J.; and Abbeel, P. 2020. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, 9133–9143. PMLR.

Sui, Y.; Gotovos, A.; Burdick, J.; and Krause, A. 2015. Safe exploration for optimization with Gaussian processes. In

International Conference on Machine Learning, 997–1005. PMLR.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT press.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2018. Reward Constrained Policy Optimization. In *International Conference on Learning Representations*.

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

Wachi, A.; and Sui, Y. 2020. Safe reinforcement learning in constrained markov decision processes. In *International Conference on Machine Learning*, 9797–9806. PMLR.

Wachi, A.; Sui, Y.; Yue, Y.; and Ono, M. 2018. Safe exploration and optimization of constrained mdps using gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Xu, J.; Tian, Y.; Ma, P.; Rus, D.; Sueda, S.; and Matusik, W. 2020. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International Conference on Machine Learning*, 10607–10616. PMLR.

Yang, R.; Sun, X.; and Narasimhan, K. 2019. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *arXiv preprint arXiv:1908.08342*.

Yang, T.-Y.; Rosca, J.; Narasimhan, K.; and Ramadge, P. J. 2019. Projection-Based Constrained Policy Optimization. In *International Conference on Learning Representations*.