# Reward Constrained Policy Optimization

**Hung Wei**
Department of Computer Science
National Chiao Tung University
hwei1048576.iie08g@nctu.edu.tw

## 1 Introduction

This paper is about a novel multi-timescale approach for constrained policy optimization.

The goal of Reinforcement Learning is to maximize the accumulated reward, it may lead to incorrect learning. And then it may cause unexpected results. While there are some methods such as Proximal policy optimization algorithms (Schulman et al., 2017) or Constrained policy optimization (Achiam et al., 2017) which could solve this issue, it still has some problem should be solved.(e.g. Reward agnostic)

In this paper, they propose the 'Reward Constrained Policy Optimization' (RCPO) algorithm. RCPO incorporates the constraint as a penalty signal into the reward function. This penalty signal guides the policy towards a constraint satisfying solution.

In the beginning, I thought that this issue should be easily solved by reward shaping. However, without figuring out the relationship between reward and behavior, it would cause an unexpected result. Hence, I think that this is such an useful algorithm. This design of the target function is pretty novel.

## 2 Problem Formulation

### 2.1 Markov Decision Processes (MDP)

1) Markov Decision Processes $M$ is defined by the tuple (S,A,R,P,$\mu$,$\gamma$).
   [1] S is the set of states.
   [2] A is the available actions.
   [3] R : S $\times$ A $\times$ S $\longmapsto \mathbb{R}$ is the reward function.
   [4] P : S $\times$ A $\times$ S $\longmapsto$ [0,1] is the transition matrix.
   [5] $\mu$ : S $\longmapsto$ [0,1] is the initial state distribution.
   [6] $\gamma \in$ [0,1) is the discount factor for future reward.
2) A policy $\pi$ : S $\longmapsto \Delta_A$ is a probability distribution over actions.
3) Value function :
   $V_R^\pi(s) = \mathbb{E}^\pi[\Sigma_t \gamma^t r(s_t, a_t)|s_0 = s]$
4) Recursive Bellman equation:

   $V_R^\pi(s) = \mathbb{E}^\pi[r(s,a) + \gamma V_R^\pi(s')|s]$
5) The goal is to maximize the expectation of the reward-to-go, given the initial state distribution $\mu$:

$$\max_{\pi \in \Pi} J_R^\pi, where J_R^\pi = \mathbb{E}_{s\sim\mu}^\pi[\Sigma_{t=0}^\infty \gamma^t r_t] = \Sigma_{s\in S}\mu(s)V_R^\pi(s). \quad (1)$$

Preprint. Under review.

### 2.2 Constrained MDPs (CMDP)

1) Penalty : c(s,a), the penalty on state s assuming action a is taken.
2) Constraint : C($s_t$) = F (c(st,at),..., c($s_N, a_N$)), discounted sum / average sum of the penalties.
3) Threshold : $\alpha \in [0, 1]$, the limitation of Constraint.
4) We denote the expectation over the constraint by :

$$J_C^\pi = \mathbb{E}_{s\sim\mu}^\pi[C(s)]. \tag{2}$$

5) The problem thus becomes :

$$\max_{\pi\in\Pi} J_R^\pi, \quad s.t. \quad J_C^\pi \leq \alpha. \tag{3}$$

### 2.3 Parametrized Policies

1) $\theta$ is the policy.
2) $\pi_\theta$ is a parametrized policy.
3) We make the following assumptions to ensure convergence to a constraint satisfying policy:

**Assumption 2.1** *- The value $V_R^\pi(s)$ is bounded for all policies $\pi \in \Pi$.*

**Assumption 2.2** *- Every local minima of $J_C^{\pi_\theta}$ is a feasible solution.*

Assumption 2.2 is the minimal requirement to ensure convergence. Stricter assumptions, such as convexity, may ensure convergence to the optimal solution; however, in practice constraints are non-convex and such assumptions do not hold.

### 2.4 Constrained Policy Optimization (CPO)

Constrained MDP's are often solved using the Lagrange relaxation technique (Bertesekas,1999). In Lagrange relaxation, the CMDP is converted into an equivalent unconstrained problem.

1) L is the Lagrangian relaxation.
2) $\lambda \geq 0$, the Lagrange multiplier which is a penalty coefficient here.
3) The unconstrained problem :

$$\min_{\lambda\geq 0}\max_{\theta} L(\lambda,\theta) = \min_{\lambda\geq 0}\max_{\theta}[J_R^{\pi_\theta} - \lambda\cdot(J_C^{\pi_\theta} - \alpha)] \tag{4}$$

[3.1]Notice, as $\lambda$ increases, the solution to (4) converges to that of (3).

[3.2]There are two timescales,which makes $\theta$ faster to found by (4), and make $\lambda$ satisfy the constraint more slowly.

[3.3]The goal is to find a saddle point $(\theta^*(\lambda^*), \lambda^*)$ of (4), which is a feasible solution.

**Definition 2.1** *A feasible solution of the CMDP is a solution which satisfies $J_C^\pi \leq \alpha$*

### 2.5 Estimating the gradient

1) Symbol

[[1.]]$\Gamma_\theta$ is a projection operator, which keeps the iterate $\theta_k$ stable by projecting onto a compact and convex set.

[[2.]]$\Gamma_\lambda$ is a projection operator, which projects $\lambda$ in to the range $[0, \lambda_{max}]$.

[[3.]]$\nabla_\lambda L$ is derived from (4).

[[4.]]$\nabla_\theta L$ is derived from (4), and it derived using the log-likelihood trick.

[[5.]]$\eta_1(k)$ is the timescale $\lambda$ updates.

[[6.]]$\eta_2(k)$ is the timescale $\theta$ updates.

$Notice \quad that \quad \eta_2(k) \quad is \quad greater \quad than \quad \eta_1(k).$

2) Update

[[1.]]$\lambda$ :

$$\lambda_{k+1} = \Gamma_\lambda[\lambda_k - \eta_1(k)\nabla_\lambda L(\lambda_k, \theta_k)] \tag{5}$$

[[2.]]$\theta$ :

$$\theta_{k+1} = \Gamma_\theta[\theta_k + \eta_2(k)\nabla_\theta L(\lambda_k, \theta_k)] \tag{6}$$

[[3.]]$\nabla_\lambda L(\lambda_k, \theta_k)$ :

$$\nabla_\lambda L(\lambda, \theta) = -(\mathbb{E}^{\pi_\theta}_{s\sim\mu}[C(s)] - \alpha) \tag{7}$$

[[4.]]$\nabla_\theta L(\lambda_k, \theta_k)$ :

$$\nabla_\theta L(\lambda, \theta) = \nabla_\theta \mathbb{E}^{\pi_\theta}_{s\sim\mu}[\log \pi(s, a; \theta)[R(s) - \lambda \cdot C(s)]] \tag{8}$$

**Assumption 2.3**

$$\Sigma^\infty_{k=0}\eta_1(k) = \Sigma^\infty_{k=0}\eta_2(k) = \infty$$

$$\Sigma^\infty_{k=0}(\eta_1(k)^2 + \eta_2(k)^2) < \infty$$

$$\frac{\eta_1(k)}{\eta_2(k)} \longrightarrow 0$$

**Theorem 2.1** *Under Assumption 3, as well as the standard stability assumption for the iterates and bounded noise (Borkar et al., 2008), the iterates $(\theta_n, \lambda_n)$ converge to a fixed point (a local minima) almost surely.*

**Lemma 2.1** *Under assumptions 2.1 and 2.2, the fixed point of Theorem 2.1 is a feasible solution.*

### 2.6 Reward Constrained Policy Optimization (RCPO)

This paper implement their RCPO algorithm on Actor-Critic, its goal is to tackle general constraints. However, it wouldn't ensured to satisfy the recursive property required to train a critic. To solve this problem, this paper uses an alternative penalty, the discounted penalty. And, it also defines a new penalized reward functions. To make sure that it satisfy constraint, $\lambda$ is still optimized using Monte-Carlo sampling on the original constraint.(Equation 7)

**Definition 2.2** *The value of the discounted (guiding) penalty is defined as:*

$$V^\pi_{C_\gamma}(s) \triangleq \mathbb{E}^\pi[\Sigma^\infty_{t=0}\gamma^t c(s_t, a_t)|s_0 = s] \tag{9}$$

**Definition 2.3** *The penalized reward functions are defined as:*

$$\hat{r}(\lambda, s, a) \triangleq r(s, a) - \lambda c(s, a), \tag{10}$$

$$\hat{V}^\pi(\lambda, s) \triangleq \mathbb{E}^\pi[\Sigma^\infty_{t=0}\gamma^t \hat{r}(\lambda, s_t, a_t)|s_0 = s]$$

$$= \mathbb{E}^\pi[\Sigma^\infty_{t=0}\gamma^t (r(s_t, a_t) - \lambda c(s_t, a_t))|s_0 = s]$$

$$= V^\pi_R(s) - \lambda V^\pi_{C_\gamma}(s) \tag{11}$$

**Algorithm 1:** Template for an RCPO implementation

**Input:** penalty c(·), constraint C(·), threshold $\alpha$, learning rates $\eta_1(k) < \eta_2(k) < \eta_3(k)$

Initialize actor parameters $\theta = \theta_0$, critic parameters $v = v_0$, Lagrange multipliers and $\lambda = 0$;
**for** *k = 0,1,...* **do**
  Initialize state $s_0 \sim \mu$;
  **for** *t = 0,1,...,T-1* **do**
    Sample action $a_t \sim \pi$, observe next state $s_{t+1}$, reward $r_t$ and penalties $c_t$;
    $\hat{R}_t = r_t - \lambda_k c_t + \gamma \hat{V}(\lambda, s_t; v_k)$
    **Critic update:** $v_{k+1} \leftarrow v_k - \eta_3(k)[\delta(\hat{R}_t - \hat{V}(\lambda, s_t; v_k))^2 / \delta_{v_k}]$
    **Actor update:** $\theta_{k+1} \leftarrow \Gamma_\theta[\theta_k + \eta_2(k)\nabla_\theta \hat{V}(\lambda, s)]$
  **end**
  **Lagrange multiplier update:** $\lambda_{k+1} \leftarrow \Gamma_\lambda[\lambda_k + \eta_1(k)(J_C^{\pi_\theta} - \alpha)]$
**end**
**return** *policy parameters* $\theta$

**Theorem 2.2** *Denote by* $\Theta = \{\ \theta\ :\ J_C^{\pi_\theta} \leq \alpha\ \}$*the set of feasible solutions and the set of local-minimas of* $J_{C_\gamma}^{\pi_\theta}$ *as* $\Theta_\gamma$*. Assuming that* $\Theta_\gamma \subseteq \Theta$ *then the 'Reward Constrained Policy Optimization' (RCPO) algorithm converges almost surely to a fixed point* $(\theta^*(\lambda^*, v^*), v^*(\lambda^*), \lambda^*)$ *which is a feasible solution (e.g.*$\theta^* \in \Theta$*).*

# 3 Theoretical Analysis

## 3.1 Theorem 2.1

The proof of convergence to a local saddle point of the Lagrangian (4) contains the following main steps:

**1. Convergence of $\theta$-recursion:** We utilize the fact that owing to projection, the $\theta$ parameter is stable. We show that the $\theta$-recursion tracks an ODE in the asymptotic limit, for any given value of $\lambda$ on the slowest timescale.

**2. Convergence of $\lambda$-recursion:** This step is similar to earlier analysis for constrained MDPs. In particular, we show that $\lambda$-recursion in (4) converges and the overall convergence of $(\theta_k, \lambda_k)$ is to a local saddle point $(\theta^*, (\lambda^*, \lambda^*))$ of $L(\lambda, \theta)$.

**Step 1:** Showing that the $\theta$-recursion converge. We can assume that the value of $\lambda$ is constant(learning rate of $\lambda$ is small). The following ODE governs the evolution of $\theta$:

$$\dot{\theta}_t = \Gamma_\theta(\nabla_\theta L(\lambda, \theta_t)) \tag{12}$$

As $\lambda$ is considered constant, the process over $\theta$ is:

$$\theta_{k+1} = \Gamma_\theta[\theta_k + \eta_2(k)\nabla_\theta L(\lambda, \theta_k)]$$

$$\theta_{k+1} = \Gamma_\theta[\theta_k + \eta_2(k)\nabla_\theta \mathbb{E}_{s\sim\mu}^{\pi_\theta}[\log \pi(s, a; \theta)[R(s) - \lambda \cdot C(s)]]]$$

Finally, using the standard stochastic approximation arguments from (Borkar et al. (2008)) concludes step 1.
**Step 2:** Showing that the $\lambda$-recursion converge. The process governing the evolution of $\lambda$:

$$\lambda_{k+1} = \Gamma_\lambda[\lambda_k - \eta_1(k)\nabla_\lambda L(\lambda_k, \theta(\lambda_k)]$$

$$= \Gamma_\lambda[\lambda_k + \eta_1(k)(\mathbb{E}_{s\sim\mu}^{\pi_{\theta(\lambda_k)}}[C(s)] - \alpha)]$$

Where $\theta(\lambda_k)$ is the limiting point of the $\theta$ -recursion corresponding to $\lambda$.
As shown in Borkar et al. (2008) chapter 6, $(\lambda_n, \theta_n)$ converges to the internally chain transitive invariant sets . Thus, $(\lambda_n, \theta_n) \to \{(\lambda(\theta), \theta) : \theta \in \mathbb{R}^k\}$almost surely.
Finally, as seen in Theorem 2 of Chapter 2 of (Borkar et al. (2008)),$\theta_n \to \theta^*$ a.s. then $\lambda_n \to \lambda(\theta^*)$ a.s. which completes the proof.

### 3.2 Lemma 2.1

The proof is obtained by a simple extension to that of Theorem 2.1. Assumption 2 states that any local minima $\pi_\theta$ of 2 satisfies the constraints, e.g. $J_C^{\pi_\theta} \leq \alpha$; additionally, Lee et al.(2017) show that first order methods such as gradient descent, converge almost surely to a local minima (avoiding saddle points and local maxima). Hence for $\lambda_{max} = \infty$(unbounded Lagrange multiplier), the process converges to a fixed point $(\theta^*(\lambda^*), \lambda^*)$ which is a feasible solution.

### 3.3 Theorem 2.2

(Prashanth and Ghavamzadeh (2016)) had proved a two-timescale stochastic approximation scheme. However, we are considering a three-timescale state.
The full process (which is mentioned in algorithm) is described as follows:
$v_{k+1} \leftarrow v_k - \eta_3(k)[\delta(\hat{R}_t - \hat{V}(\lambda, s_t; v_k))^2/\delta_{v_k}]$
$\theta_{k+1} \leftarrow \Gamma_\theta[\theta_k + \eta_2(k)\nabla_\theta \hat{V}(\lambda, s)]$
$\lambda_{k+1} \leftarrow \Gamma_\lambda[\lambda_k + \eta_1(k)(J_C^{\pi_\theta} - \alpha)]$

**Step 1:** The value $v_k$ runs on the fastest timescale, hence it observes $\theta$ and $\lambda$ as static. As the TD operator is a contraction we conclude that $v_k \to v(\lambda, \theta)$.

**Step 2:** For the policy recursion $\theta_k$, due to the timescale differences, we can assume that the critic $v$ has converged and that $\lambda$ is static. Thus as seen in the proof of Theorem 2.1 , $\theta_k$ converges to the fixed point $\theta(\lambda, v)$.

**Step 3:**As shown previously (and in Prashanth and Ghavamzadeh (2016)), $(\lambda_n, \theta_n, v_n) \to (\lambda(\theta^*), \theta^*, v(\theta^*)$a.s.

**Assumption 3.1** - $\Theta_\gamma \subseteq \Theta$

But there might not work depending on assumption 2.2 and assumption 3.1.

**1.Assumption 2.2 does not hold:**We can only treat the constraint as a regularizing term for the policy in which $\lambda_{max}$ defines the maximal regularization allowed.

**2.Assumption 3.1 does not hold:**A Monte-Carlo method may be used to approximate the gradients. However, this does not enjoy the benefits of reducing variance and smaller samples.

## 4 Conclusion

This paper propose the 'Reward Constrained Policy Optimization' (RCPO) algorithm. It is a constrained policy optimization.

In this paper, they test the RCPO algorithm in various domains. Most tasks are in the Mujoco simulator. And then they compare their behavior with reward shaping. In the mujoco experiments results, we can see that RCPO can get the highest reward with torque constraints in most environment. However, there is something catching my eye. Reward shaping exhibits a superior performance in Walker2d-v2. After comparing those reward curved graph, I consider that while RCPO didn't get good rewards in Walker2d-v2, it still have some irreplaceable advantages. First, reward shaping sometimes have good performance, but it attribute to adjusting $\lambda$ by programmer.On the contrary, RCPO can learn its lagrange multiplier by itself. Second, RCPO tends to satisfy the constraint before getting higher reward.

To sum up, I speak highly of this algorithm. As it mentioned in paper, it could be combined with PPO,A2C... to solve more complex constraints. In the future, it even can combine CPO to pursue better work performance.

Up to now, there are some works having a good performance based on this paper. Such as Reinforcement learning with convex constraints(Miryoosefi et al., 2019), they try to do some research with convex constraints. Constrained Deep Reinforcement Learning for Energy Sustainable Multi-UAV based Random Access IoT Networks with NOMA (Khairy et al., 2020), which is about the application in Access IoT Networks.

## References


Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization, 2017.

Prashanth L. A. and Mohammad Ghavamzadeh. Variance-constrained actor-critic algorithms for discounted and average reward mdps, 2014.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills, 2018.

Aviv Tamar and Shie Mannor. Variance adjusted actor critic algorithms, 2013.

Achiam et al. [2017] A. and Ghavamzadeh [2014] Schulman et al. [2017] Peng et al. [2018] Tamar and Mannor [2013]

Vivek S Borkar et al. Stochastic approximation.Cambridge Books, 2008.