Accelerated Policy Gradient for s-rectangular Robust MDPs with Large State Spaces

Ziyi Chen[1] **Heng Huang[1]**

[1]Department of Computer Science, University of Maryland

UNIVERSITY OF MARYLAND

Problem Formulation: Robust MDP

- Environmental state s_t.
- Agent takes action $a_t \sim \text{policy } \pi(\cdot \mid s_t)$.
- Environment transitions to the next state $s_{t+1} \sim \text{transition kernel } p(\cdot \mid s_t, a_t)$.
- Agent receives cost $c_t = c(s_t, a_t, s_{t+1})$.

Problem Formulation: Robust MDP

- Environmental state s_t.
- Agent takes action $a_t \sim \text{policy } \pi(\cdot \mid s_t)$.
- Environment transitions to the next state $s_{t+1} \sim \text{transition kernel } p(\cdot \mid s_t, a_t)$.
- Agent receives cost $c_t = c(s_t, a_t, s_{t+1})$.

- Transition Kernel $p \in \text{ambiguity set } \mathscr{P}$, which is convex, compact and **s-rectangular**:

$$\mathscr{P} = \times_{s \in \mathcal{S}} \mathscr{P}_s, \text{ where } \mathscr{P}_s = \{p(\cdot \mid s, \cdot)\}$$

Problem Formulation: Robust MDP

- Environmental state s_t.
- Agent takes action $a_t \sim \text{policy } \pi(\cdot \mid s_t)$.
- Environment transitions to the next state $s_{t+1} \sim \text{transition kernel } p(\cdot \mid s_t, a_t)$.
- Agent receives cost $c_t = c(s_t, a_t, s_{t+1})$.

- Transition Kernel $p \in \text{ambiguity set } \mathscr{P}$, which is convex, compact and **s-rectangular**:

$$\mathscr{P} = \times_{s \in \mathcal{S}} \mathscr{P}_s, \text{ where } \mathscr{P}_s = \{p(\cdot \mid s, \cdot)\}$$

- Objective of **Robust** MDP:

$$\min_{\pi \in \Pi} \max_{p \in \mathscr{P}} J_\rho(\pi, p) := \mathbb{E}_{\pi,p}\left[\sum_{t=0}^{\infty} \gamma^t c_t \middle| s_0 \sim \rho\right]$$

Policy Gradient Methods are Practical

❖ 3 Methods for Robust MDP:

- Value iteration
- Policy iteration
- **Policy gradient**

❖ **Policy Gradient Methods** are **practical** with advantages:

- Simple implementation
- Many successes in real world
- Scalable to large state and action spaces

Our Contributions:

❖ Our Contributions: Improve Policy Gradient for s-rectangular robust MDP

- **Acceleration** in deterministic setting: $\mathcal{O}(\epsilon^{-4})(\text{SOTA}) \searrow \mathcal{O}(\epsilon^{-3}\ln\epsilon^{-1})$(Our Algorithm 1)

- Extend to **Stochastic** setting for the first time (Algorithm 1–>2).

- Extend to **Stochastic**+**Large State Space** for the first time (Algorithm 2–>3).

- Our Algorithms 1-3 are also the first policy gradient methods to solve **Entropy Regularized Robust MDP**.

Entropy Regularized Robust MDP

❖ **Entropy Regularized Robust MDP** with coefficient $\tau \geq 0$:

$$\min_{\pi\in\Pi}\max_{p\in\mathscr{P}} J_{\rho,\tau}(\pi, p) := \mathbb{E}_{\pi,p}\left[\sum_{t=0}^{\infty}\gamma^t[c_t + \tau \ln \pi(a_t \mid s_t)]\,\middle|\, s_0 \sim \rho\right]$$

⇒ Special case: **Robust MDP ($\tau = 0$)**

$$\min_{\pi\in\Pi}\left\{\Phi_\rho(\pi) \stackrel{\text{def}}{=} \max_{p\in\mathscr{P}} J_{\rho,0}(\pi, p)\right\}$$

Entropy Regularized Robust MDP

❖ **Entropy Regularized Robust MDP** with coefficient $\tau \geq 0$:

$$\min_{\pi \in \Pi} \max_{p \in \mathscr{P}} J_{\rho,\tau}(\pi, p) := \mathbb{E}_{\pi,p}\left[\sum_{t=0}^{\infty} \gamma^t [c_t + \tau \ln \pi(a_t | s_t)] \middle| s_0 \sim \rho\right]$$

⇒ Special case: **Robust MDP ($\tau = 0$)**

$$\min_{\pi \in \Pi} \left\{ \Phi_\rho(\pi) \overset{\text{def}}{=} \max_{p \in \mathscr{P}} J_{\rho,0}(\pi, p) \right\}$$

Proposition 1: π is (ϵ, τ)-Nash equilibrium to **Entropy Regularized Robust MDP**, i.e.,

$$J_{\rho,\tau}(\pi, p) - \min_{\pi' \in \Pi} J_{\rho,\tau}(\pi', p) \leq \epsilon, \quad \max_{p' \in \mathscr{P}} J_{\rho,\tau}(\pi, p') - J_{\rho,\tau}(\pi, p) \leq \epsilon$$

⇒ π is also $[\epsilon + \mathcal{O}(\tau)]$-optimal robust policy to **Robust MDP**, i.e.,

$$\Phi_\rho(\pi) \leq \min_{\pi' \in \Pi} \Phi_\rho(\pi') + \epsilon + \mathcal{O}(\tau)$$

Algorithm 1: Accelerated Policy Gradient (Deterministic)

- Strong Duality holds for **Entropy Regularized** Robust MDP:

$$\min_{\pi \in \Pi} \max_{p \in \mathscr{P}} J_{\rho,\tau}(\pi, p) \Leftrightarrow \max_{p \in \mathscr{P}} \left[F_{\rho,\tau}(p) \overset{\text{def}}{=} \min_{\pi \in \Pi} J_{\rho,\tau}(\pi, p) \right]$$

Algorithm 1: Accelerated Policy Gradient (Deterministic)

- Strong Duality holds for **Entropy Regularized** Robust MDP:

$$\min_{\pi \in \Pi} \max_{p \in \mathscr{P}} J_{\rho,\tau}(\pi, p) \Leftrightarrow \max_{p \in \mathscr{P}} \left[F_{\rho,\tau}(p) \stackrel{\text{def}}{=} \min_{\pi \in \Pi} J_{\rho,\tau}(\pi, p) \right]$$

- Become Smooth: $\nabla F_{\rho,\tau}(p) = \nabla_2 J_{\rho,\tau}(\pi_p, p)$ where $\pi_p := \arg\min_\pi J_{\rho,\tau}(\pi, p)$ (unique).

Algorithm 1: Accelerated Policy Gradient (Deterministic)

- Strong Duality holds for **Entropy Regularized** Robust MDP:

$$\min_{\pi\in\Pi}\max_{p\in\mathscr{P}} J_{\rho,\tau}(\pi,p) \Leftrightarrow \max_{p\in\mathscr{P}}\left[F_{\rho,\tau}(p) \stackrel{\text{def}}{=} \min_{\pi\in\Pi} J_{\rho,\tau}(\pi,p)\right]$$

- Become Smooth: $\nabla F_{\rho,\tau}(p) = \nabla_2 J_{\rho,\tau}(\pi_p, p)$ where $\pi_p := \arg\min_\pi J_{\rho,\tau}(\pi,p)$ (unique).

$\Downarrow$

- Outer loop: $p_{t+1} = \mathbf{proj}_{\mathscr{P}}\left(p_t + \beta\hat{\nabla}_p J_{\rho,\tau}(\pi_t, p_t)\right)$, where $\pi_t \approx \pi_{p_t} := \arg\min_\pi J_{\rho,\tau}(\pi, p_t)$.

Algorithm 1: Accelerated Policy Gradient (Deterministic)

- Strong Duality holds for **Entropy Regularized** Robust MDP:

$$\min_{\pi\in\Pi}\max_{p\in\mathscr{P}} J_{\rho,\tau}(\pi,p) \Leftrightarrow \max_{p\in\mathscr{P}}\left[F_{\rho,\tau}(p) \stackrel{\text{def}}{=} \min_{\pi\in\Pi} J_{\rho,\tau}(\pi,p)\right]$$

- Become Smooth: $\nabla F_{\rho,\tau}(p) = \nabla_2 J_{\rho,\tau}(\pi_p, p)$ where $\pi_p := \arg\min_\pi J_{\rho,\tau}(\pi,p)$ (unique).

$\Downarrow$

- Outer loop: $p_{t+1} = \textbf{proj}_{\mathscr{P}}\left(p_t + \beta\hat{\nabla}_p J_{\rho,\tau}(\pi_t, p_t)\right)$, where $\pi_t \approx \pi_{p_t} := \arg\min_\pi J_{\rho,\tau}(\pi, p_t)$.

- Inner loop: Get $\pi_t \approx \pi_{p_t}$ via natural policy gradient method:

$$\pi_{t,k+1}(\cdot\,|\,s) \propto \pi_{t,k}(\cdot\,|\,s)\exp\left[-\frac{\eta\hat{Q}_{t,k}(s,\cdot)}{1-\gamma}\right]$$

Algorithm 1: Accelerated Policy Gradient (Deterministic)

- Strong Duality holds for **Entropy Regularized** Robust MDP:

$$\min_{\pi\in\Pi}\max_{p\in\mathscr{P}} J_{\rho,\tau}(\pi,p) \Leftrightarrow \max_{p\in\mathscr{P}}\left[F_{\rho,\tau}(p) \overset{\text{def}}{=} \min_{\pi\in\Pi} J_{\rho,\tau}(\pi,p)\right]$$

- Become Smooth: $\nabla F_{\rho,\tau}(p) = \nabla_2 J_{\rho,\tau}(\pi_p, p)$ where $\pi_p := \arg\min_\pi J_{\rho,\tau}(\pi,p)$ (unique).

$\Downarrow$

- Outer loop: $p_{t+1} = \textbf{proj}_{\mathscr{P}}\left(p_t + \beta\hat{\nabla}_p J_{\rho,\tau}(\pi_t, p_t)\right)$, where $\pi_t \approx \pi_{p_t} := \arg\min_\pi J_{\rho,\tau}(\pi, p_t)$.

- Inner loop: Get $\pi_t \approx \pi_{p_t}$ via natural policy gradient method:

$$\pi_{t,k+1}(\cdot\,|\,s) \propto \pi_{t,k}(\cdot\,|\,s)\exp\left[-\frac{\eta\hat{Q}_{t,k}(s,\cdot)}{1-\gamma}\right]$$

- Iteration complexity (deterministic: can access exact $Q(s,a)$ and ∇J):

$\mathscr{O}(\epsilon^{-4})(\text{SOTA}) \searrow \mathscr{O}(\epsilon^{-3}\ln\epsilon^{-1})$(Our Algorithm 1)

Algorithm 2 (Extend Algorithm 1 to Stochastic Setting)

- Estimate $Q_\tau(\pi, p)$ via temporal difference (TD):

$$q_{n+1}(s_n, a_n) = q_n(s_n, a_n) + \alpha\left[c(s_n, a_n, s_n') + \tau \ln \pi(a_n \,|\, s_n) + \gamma q_n(s_n', a_n') - q_n(s_n, a_n)\right]$$

$$\Rightarrow \text{output} : \overline{q}_{T_1} \overset{\text{def}}{=} \frac{1}{T_1}\sum_{n=1}^{T_1} q_n \approx Q_\tau(\pi, p)$$

Algorithm 2 (Extend Algorithm 1 to Stochastic Setting)

- Estimate $Q_\tau(\pi, p)$ via temporal difference (TD):

$$q_{n+1}(s_n, a_n) = q_n(s_n, a_n) + \alpha\left[c(s_n, a_n, s_n') + \tau \ln \pi(a_n \,|\, s_n) + \gamma q_n(s_n', a_n') - q_n(s_n, a_n)\right]$$

$$\Rightarrow \text{output}: \bar{q}_{T_1} \stackrel{\text{def}}{=} \frac{1}{T_1}\sum_{n=1}^{T_1} q_n \approx Q_\tau(\pi, p)$$

- Estimate gradient $\nabla_p J_{\rho,\tau}$ via sample average:

$$\hat{\nabla}_p J_{\rho,\tau}(\pi, p)(s, a, s') = \frac{1}{N(1-\gamma)}\sum_{i=1}^{N} \pi(a \,|\, s)\mathbb{1}\{s_{i,H_i} = s\}\left[c(s, a, s') + \tau \ln \pi(a \,|\, s) + \gamma \sum_{a'} \pi(a' \,|\, s')\bar{q}_{T_1}(s', a')\right]$$

Algorithm 2 (Extend Algorithm 1 to Stochastic Setting)

- Estimate $Q_\tau(\pi, p)$ via temporal difference (TD):

$$q_{n+1}(s_n, a_n) = q_n(s_n, a_n) + \alpha\left[c(s_n, a_n, s_n') + \tau \ln \pi(a_n \mid s_n) + \gamma q_n(s_n', a_n') - q_n(s_n, a_n)\right]$$

$$\Rightarrow \text{output} : \bar{q}_{T_1} \overset{\text{def}}{=} \frac{1}{T_1} \sum_{n=1}^{T_1} q_n \approx Q_\tau(\pi, p)$$

- Estimate gradient $\nabla_p J_{\rho,\tau}$ via sample average:

$$\hat{\nabla}_p J_{\rho,\tau}(\pi, p)(s, a, s') = \frac{1}{N(1-\gamma)} \sum_{i=1}^{N} \pi(a \mid s) \mathbb{1}\{s_{i,H_i} = s\} \left[c(s, a, s') + \tau \ln \pi(a \mid s) + \gamma \sum_{a'} \pi(a' \mid s') \bar{q}_{T_1}(s', a')\right]$$

- Sample complexity: $\mathcal{O}[\epsilon^{-7} \ln(\epsilon^{-1})]$.

Algorithm 3 (Extend Algorithm 2 to Large State Spaces)

- Linear Q function approximation: $Q_\tau(\pi_{t,k}, p_t; s, a) \approx \phi(s, a)^\top w_{t,k}$ via TD:

$$w_{n+1} = w_n + \alpha\phi(s_n, a_n)[c(s_n, a_n, s_n') + \tau \ln \pi(a_n \mid s_n) + \gamma\phi(s_n', a_n')^\top w_n - \phi(s_n, a_n)^\top w_n]$$

$$\Rightarrow \text{output} : \overline{w}_{T_1} \stackrel{\text{def}}{=} \frac{1}{T_1}\sum_{n=1}^{T_1} w_n$$

Algorithm 3 (Extend Algorithm 2 to Large State Spaces)

- Linear Q function approximation: $Q_\tau(\pi_{t,k}, p_t; s, a) \approx \phi(s,a)^\top w_{t,k}$ via TD:

$$w_{n+1} = w_n + \alpha\phi(s_n, a_n)[c(s_n, a_n, s_n') + \tau \ln \pi(a_n \mid s_n) + \gamma\phi(s_n', a_n')^\top w_n - \phi(s_n, a_n)^\top w_n]$$

$$\Rightarrow \text{output} : \overline{w}_{T_1} \overset{\text{def}}{=} \frac{1}{T_1}\sum_{n=1}^{T_1} w_n$$

- Linear transition kernel approximation: $p_{\xi_t}(s' \mid s, a) = \psi(s, a, s')^\top \xi_t$

$$\hat{\nabla}_\xi J_{\rho,\tau}(\pi_t, p_{\xi_t}) = \frac{1}{N(1-\gamma)}\sum_{i=1}^{N} \frac{\psi(s_{i,H_i}, a_{i,H_i}, s_{i,H_i+1})}{p_\xi(s_{i,H_i+1} \mid s_{i,H_i}, a_{i,H_i})}\left[c(s_{i,H_i}, a_{i,H_i}, s_{i,H_i+1}) + \tau \ln \pi_t(a_{i,H_i} \mid s_{i,H_i}) + \gamma\phi(s_{i,H_i+1}, a_{i,H_i+1})^\top \overline{w}_{T_1}\right]$$

$$\xi_{t+1} = \text{proj}_\Xi\left(\xi_t + \beta\,\hat{\nabla}_\xi J_{\rho,\tau}(\pi_t, p_{\xi_t})\right)$$

Algorithm 3 (Extend Algorithm 2 to Large State Spaces)

- Linear Q function approximation: $Q_\tau(\pi_{t,k}, p_t; s, a) \approx \phi(s,a)^\top w_{t,k}$ via TD:

$$w_{n+1} = w_n + \alpha\phi(s_n, a_n)[c(s_n, a_n, s_n') + \tau \ln \pi(a_n \mid s_n) + \gamma\phi(s_n', a_n')^\top w_n - \phi(s_n, a_n)^\top w_n]$$

$$\Rightarrow \text{output} : \overline{w}_{T_1} \overset{\text{def}}{=} \frac{1}{T_1}\sum_{n=1}^{T_1} w_n$$

- Linear transition kernel approximation: $p_{\xi_t}(s' \mid s, a) = \psi(s, a, s')^\top \xi_t$

$$\hat{\nabla}_\xi J_{\rho,\tau}(\pi_t, p_{\xi_t}) = \frac{1}{N(1-\gamma)}\sum_{i=1}^{N} \frac{\psi(s_{i,H_i}, a_{i,H_i}, s_{i,H_i+1})}{p_\xi(s_{i,H_i+1} \mid s_{i,H_i}, a_{i,H_i})}\big[c(s_{i,H_i}, a_{i,H_i}, s_{i,H_i+1}) + \tau \ln \pi_t(a_{i,H_i} \mid s_{i,H_i}) + \gamma\phi(s_{i,H_i+1}, a_{i,H_i+1})^\top \overline{w}_{T_1}\big]$$

$$\xi_{t+1} = \text{proj}_\Xi\big(\xi_t + \beta\hat{\nabla}_\xi J_{\rho,\tau}(\pi_t, p_{\xi_t})\big)$$

- Sample complexity: $\mathcal{O}[\epsilon^{-7}\ln(\epsilon^{-1})]$.

Thank You