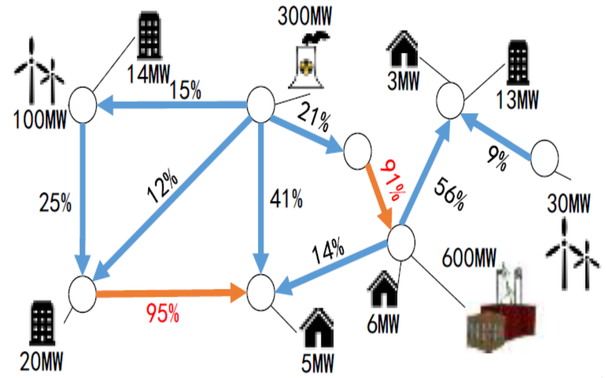
Multi-Agent First Order Constrained Optimization in Policy Space

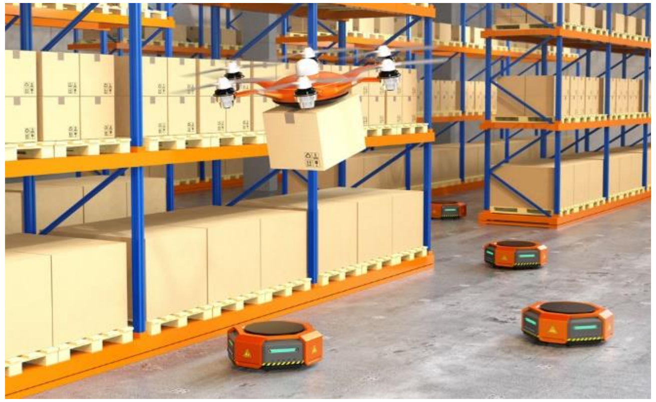
Youpeng Zhao, Yaodong Yang, Zhenbo Lu, Wengang Zhou, Houqiang Li

NeurIPS 2023

Background

- MARL has wide applications in many real-life scenarios.



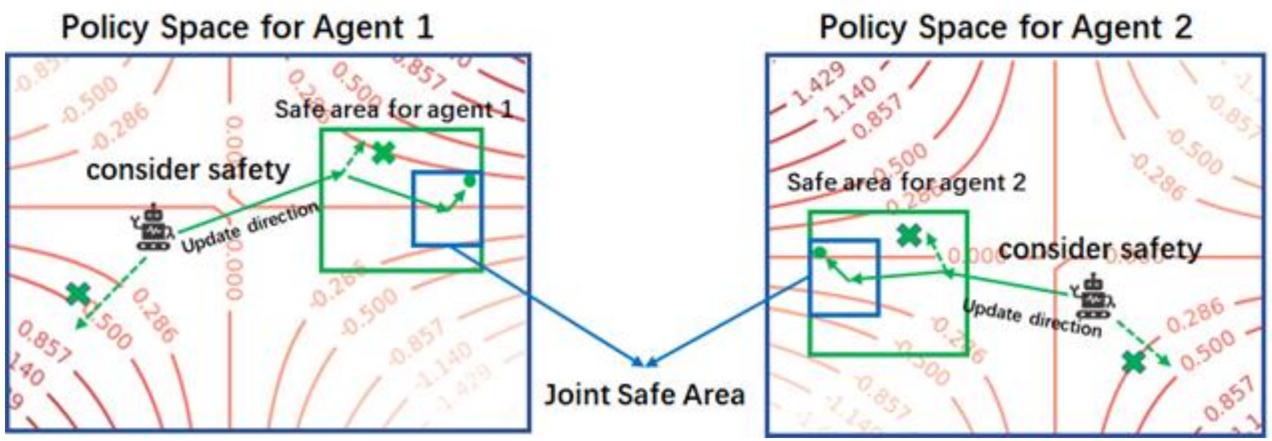


- However, most MARL algorithms prioritize policy optimization solely for reward maximization, while disregarding potential negative or harmful consequences resulting from the agents' behaviors.

- In this work, we focus on designing algorithms that learn policies which adhere to safety constraints.

Challenges

- Developing safe policies for multi-agent systems poses daunting challenges.



- Two problems:
  - ➢ The environment may suffer from non-stationarity due to simultaneously learning agents.
  - ➢ Ensuring safety in MARL is highly intricate.

Safe Multi-Agent RL Formulation

- We model the problem with Constrained Markov Decision Process, which can be described by a tuple $< \mathcal{N}, \mathcal{S}, \mathbf{A}, p, \rho^0, \gamma, R, \boldsymbol{C}, c >$.
  - ➢ $\mathcal{N}$ means the number of agents.
  - ➢ $\mathcal{S}$ and $\mathbf{A}$ denotes the state and action space of agents.
  - ➢ $p: \mathcal{S} \times \mathbf{A} \times \mathcal{S} \rightarrow R$ represents the probabilistic transition function.
  - ➢ ρ^0 is the initial state distribution and γ is the discounted factor.
  - ➢ R means the team reward while $\boldsymbol{C}$ is the set of cost functions and c denotes the corresponding cost-constraining values.
- The objective of safe MARL problem is to maximize team reward while satisfying safety constraints.

Related Work

- A recent remarkable work called MACPO addresses safe MARL problem by developing the multi-agent trust region learning based on CPO, which also motivates our work.
- Although providing theoretical guarantees of both monotonic improvement in reward and compliance with cost constraints, this method involves solving an optimization problem using very complex computation, which also introduces nonnegligible approximation errors.

Method

- We propose a new algorithm to help multi-agent systems learn policies while ensuring safety constraints. It can deduced that for agent i_h and the index of its cost function j, given the joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}$ and updated policies of previous agent sets $\pi_{\theta_{k+1}}^{i_{1:h-1}}$, the new policy is obtained by solving the following problem:
  - $$\max_{\pi_\theta^{i_h}} E_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}, a^{i_{1:h-1}}\sim\pi_{\theta_{k+1}}^{i_{1:h-1}}, a^{i_h}\sim\pi_h^{i}} \left[A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}^{i_h}\left(s, a^{i_{1:h-1}}, a^{i_h}\right)\right],$$
  - $$J_j^{i_h}\left(\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}\right) + E_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}, a^{i_h}\sim\pi^{i_h}} \left[A_{j,\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}^{i_h}\left(s, a^{i_h}\right)\right] \le c_j^{i_h}, \forall j \in 1, \cdots, m^{i_h} \text{ and } \overline{\mathrm{D}}_{\mathrm{KL}}\left(\pi_\theta^{i_h}, \pi_{\theta_k}^{i_h}\right) \le \delta.$$
- We solve the problem using a two-step approach:
  - We first find the optimal policy update which may be in nonparameterized policy space.
  - Then we need to project the optimal policy back into parameterized policy space, which allows for evaluation and sampling.

Method

- Finding the optimal policy update:

  - For agent i_h, we define $b_j^{i_h} = c_j^{i_h} - J_j^{i_h}(\boldsymbol{\pi_{\theta_k}})$, the optimal policy can be represented using

$$\pi^{i_h *}(a|s) = \frac{\pi_{\theta_k}^{i_h}(a|s)}{Z_{\lambda_j,\nu_j}(s)} exp\{\frac{1}{\lambda_j}\left(\eta_{\boldsymbol{\pi_{\theta_k}}}(s, a^{i_h}) - \nu_j A_{j,\boldsymbol{\pi_{\theta_k}}}^{i_h}(s, a^{i_h})\right)\} ,$$

  - $\eta_{\boldsymbol{\pi_{\theta_k}}}(s, a^{i_h}) = E_{a^{i_{1:h-1}} \sim \pi_{\theta_{k+1}}^{i_{1:h-1}}}\left[A_{\pi_{\theta_k}}^{i_h}(s, a^{i_{1:h-1}}, a^{i_h})\right]$

  - $Z_{\lambda_j,\nu_j}(s)$ is the partition function that ensures the policy to be a valid probability distribution.

  - λ_j *and* ν_j are solutions to an optimization problem:

$$\min_{\lambda_j,\nu_j \geq 0} \lambda_j \delta + \nu_j b_j^{i_h} + \lambda_j E_{s \sim \rho_{\boldsymbol{\pi_{\theta_k}}}, a^{i_h} \sim \pi^{i_h *}}\left[log Z_{\lambda_j,\nu_j}(s)\right]$$

Method

- Approximating the Optimal Update Policy :
  - Minimizing the loss function $L(\theta) = E_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}}\left[D_{KL}\left(\pi_{\theta}^{i_h}\middle\|\pi^{i_h*}\right)(s)\right]$ to obtain the parameterized policy which is closest to the optimal update policy.
  - We propose that first-order methods can be adopted in this process.

$$\nabla_{\theta}L(\theta) = E_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}}\left[\nabla_{\theta}D_{KL}\left(\pi_{\theta}^{i_h}\middle\|\pi^{i_h*}\right)(s)\right]$$

$$=\nabla_{\theta}D_{KL}(\pi_{\theta}^{i_h}||\pi_{\theta_k}^{i_h}) - \frac{1}{\lambda_j}E_{a\sim\pi_{\theta_k}^{i_h}}\left[\frac{\nabla_{\theta}\pi_{\theta}^{i_h}(a|s)}{\pi_{\theta_k}^{i_h}(a|s)}\left(\eta_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}(s,a^{i_h}) - \nu_j A_{j,\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}^{i_h}(s,a^{i_h})\right)\right]$$

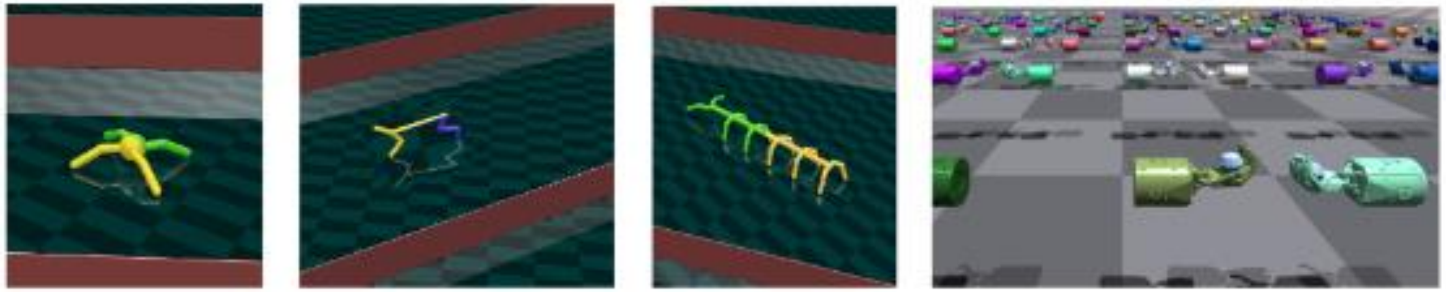
Method

- Overall Implementation
  - Solve λ_j and ν_j
    - λ_j is similar to temperature term and we set it as a fixed value.
    - ν_j can be obtained by $\frac{\partial L(\pi^{i_h *},\lambda_j,\nu_j)}{\partial \nu_j} = b_j^{i_h} - E_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}, a^{i_h}\sim\pi^{i_h *}(a|s)}\left[A_{j,\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}}^{i_h}(s,a^{i_h})\right]$

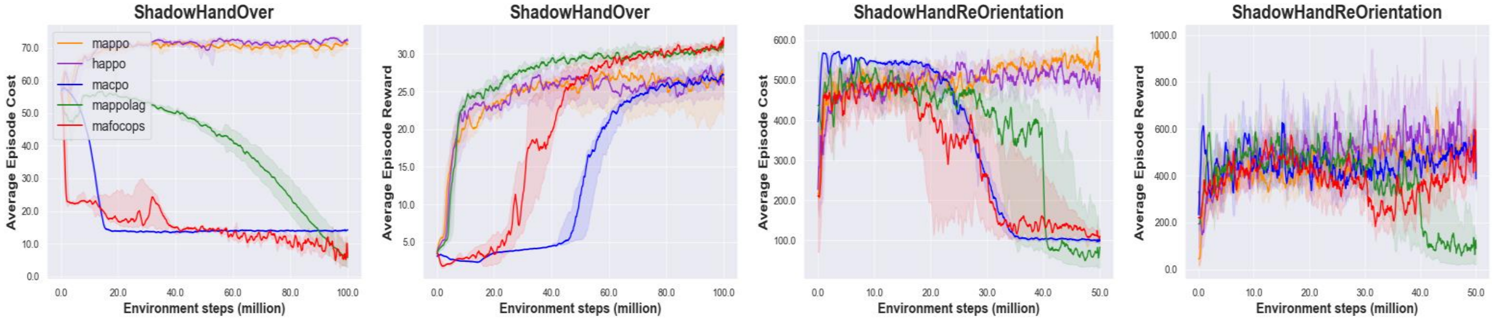
- Algorithm Outline
  - For every iteration, start with joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_{\boldsymbol{k}}}$ and generate trajectories using it.
  - Estimate the C-returns and advantage functions.
  - Making use of the collected data to obtain ν_j.
  - Update value and policy networks using the derived equations.

Experiments

- Experiment benchmarks

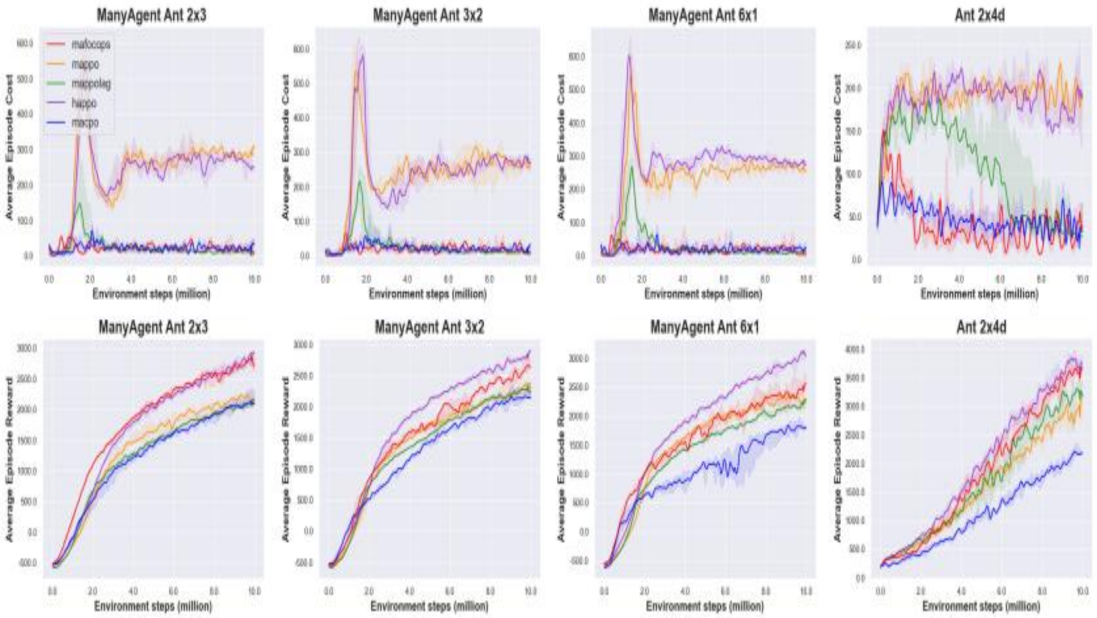


- Performance on Safe MAIG

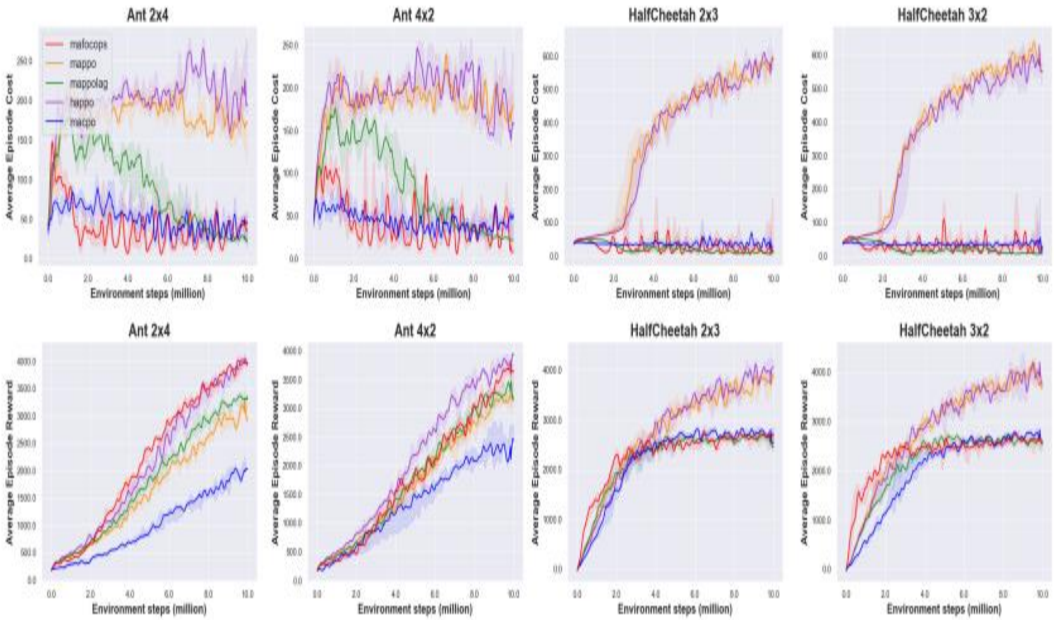


Experiments

- Performance on Safe Multi-agent MuJoCo



(a) ManyAgent Ant 2x3, ManyAgent Ant 3x2, ManyAgent Ant 6x1, Ant 2x4d



(b) Ant 2x4, Ant 4x2, HalfCheetah 2x3, HalfCheetah 3x2

Experiments

- Efficiency Analysis

  ➢ Our algorithm brings apparent improvement in the computational efficiency and memory usage.

Scenarios	Ant Task				HalfCheetah Task		
FPS \ Config	2x4d	2x4	4x2	8x1	2x3	3x2	6x1
MACPO	231	218	130	73	298	192	106
MAFOCOPS	322	270	160	115	340	229	162
Improvement(%)	**39.39**	**23.85**	**23.08**	**57.53**	**14.09**	**19.27**	**52.83**
Scenarios	ManyAgent Ant Task						
FPS \ Config	2x3	3x2	6x1	–	2x4	4x2	8x1
MACPO	244	167	98	–	232	135	73
MAFOCOPS	271	249	149	–	253	193	115
Improvement(%)	**11.07**	**49.10**	**52.04**	–	**9.05**	**42.96**	**57.53**

Scenarios	Ant Task				HalfCheeath Taks		
Memory (MiB) \ Config	2x4d	2x4	4x2	8x1	2x3	3x2	6x1
MACPO	18.85	23.60	31.24	66.25	16.54	30.20	52.08
MAFOCOPS	18.97	21.82	24.23	56.99	19.34	27.26	39.15
Saved Memory	-0.12	**1.77**	**7.01**	**9.26**	-2.80	**2.93**	**12.93**
Scenarios	ManyAgent Ant Task						
Memory (MiB) \ Config	2x3	3x2	6x1	–	2x4	4x2	8x1
MACPO	25.32	32.64	55.27	–	27.31	38.90	65.02
MAFOCOPS	24.45	30.88	44.38	–	24.62	34.73	60.71
Saved Memory	**0.87**	**1.76**	**10.89**	–	**2.69**	**4.17**	**4.31**