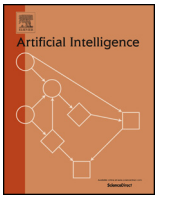
Artificial Intelligence 319 (2023) 103905

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Safe multi-agent reinforcement learning for multi-robot control ☆

Shangding Gu [d,a,1,2], Jakub Grudzien Kuba [b,1], Yuanpei Chen [d], Yali Du [c], Long Yang [d], Alois Knoll [a], Yaodong Yang [d,*]

[a] Department of Computer Science, Technical University of Munich, Germany
[b] Department of Statistics, University of Oxford, UK
[c] Department of Informatics, King's College London, UK
[d] Institute for Artificial Intelligence, Peking University, China

ARTICLE INFO

Article history:
Received 15 April 2022
Received in revised form 19 January 2023
Accepted 12 March 2023
Available online 16 March 2023

Dataset link: https://sites.google.com/view/aij-safe-marl/

Keywords:
Constrained Markov game
Constrained policy optimisation
Safe multi-agent benchmarks
Safe multi-robot control

ABSTRACT

A challenging problem in robotics is how to control multiple robots cooperatively and safely in real-world applications. Yet, developing multi-robot control methods from the perspective of safe multi-agent reinforcement learning (MARL) has merely been studied. To fill this gap, in this study, we investigate safe MARL for multi-robot control on cooperative tasks, in which each individual robot has to not only meet its own safety constraints while maximising their reward, but also consider those of others to guarantee safe team behaviours. Firstly, we formulate the safe MARL problem as a constrained Markov game and employ policy optimisation to solve it theoretically. The proposed algorithm guarantees monotonic improvement in reward and satisfaction of safety constraints at every iteration. Secondly, as approximations to the theoretical solution, we propose two safe multi-agent policy gradient methods: *Multi-Agent Constrained Policy Optimisation* (MACPO) and *MAPPO-Lagrangian*. Thirdly, we develop the first three safe MARL benchmarks—Safe Multi-Agent MuJoCo (Safe MAMuJoCo), Safe Multi-Agent Robosuite (Safe MARobosuite) and Safe Multi-Agent Isaac Gym (Safe MAIG) to expand the toolkit of MARL and robot control research communities. Finally, experimental results on the three safe MARL benchmarks indicate that our methods can achieve state-of-the-art performance in the balance between improving reward and satisfying safety constraints compared with strong baselines. Demos and code are available at the link (https://sites.google.com/view/aij-safe-marl/).[2]

1. Introduction

In the past several decades, traditional control methods have played an important role in multi-robot control, e.g. the Lyapunov based method [34], backstepping method [10], and feedback linearisation method [18] have been widely adopted in the field. However, these model-based control methods may be hard to generalise to complex tasks with unknown

☆ This paper is part of the Special Issue: "Risk-aware Autonomous Systems: Theory and Practice".

* Corresponding author.
E-mail address: yaodong.yang@pku.edu.cn (Y. Yang).

[1] These authors contributed equally to this work.

[2] Work done as a research intern at Institute for AI, Peking University.

https://doi.org/10.1016/j.artint.2023.103905

dynamics [6,9,23]. On the contrary, in recent years, reinforcement learning (RL) has been successfully applied in many tasks, such as transportation schedule [11], traffic signal control [15], the game of Go [45], autonomous driving [30,20], finance [1], recommender systems [52] and robotics [9]. The robustness of RL to variety of tasks comes from its experience-driven learning that, while relying on a learner's ability to interact with an environment, does not have to possess all of the environment's underlying information. As such, an RL agent formulates a problem of solving a task via maximisation of the reward signal by taking adequate actions [46]. In combination with expressive deep neural networks that model an agent's policy, these methods have made remarkable advancements in the past decade [41,43,27,47].

However, in the context of practical applications, most RL algorithms share a major limitation–they optimise the agents' policies purely for the sake of reward maximisation, completely ignoring safety considerations. For example, in robot control scenarios, it is essential that a robot does not visit certain states, or must not take certain actions, which can be thought of as *unsafe* either for itself or for elements of its environment [32,2,48]. While the potential danger, e.g., collisions, widely exist in the multi-robot tasks, the difficulty of ascertaining safety is exacerbated in applying multi-agent RL (MARL). Due to the existence of multiple agents, MARL is faced with the difficulty of nonstationarity in training. Furthermore, for practical considerations, tackling safety in MARL is also challenging because each individual agent has to not only consider its own safety constraints, which already may conflict its reward maximisation, but also consider the safety constraints of others so that their joint behaviours are guaranteed to be safe. Currently, there are very few solutions that offer effective learning algorithms for safe multi-robot control problems. In this study, we address the problem with theoretical analysis, practical algorithms and three benchmarks.

The partial work of the study is shared on an open platform [21]. In this study, we provide a more comprehensive investigation and solution for safe MARL. Firstly, we investigate the safe multi-robot control problem from the safe MARL perspective. We develop both theoretical and practical algorithms that solve it. Secondly, we develop sophisticated environments for benchmarking safe MARL algorithms: Safe MAMuJoCo, Safe MARobosuite and Safe MAIG. Finally, we provide experiments to evaluate the effectiveness of our proposed methods, Besides, we also perform parameter studies to show sensitivity of the proposed algorithms to safety-related hyper-parameter changes. To the best of our knowledge, MACPO and MAPPO-Lagrangian are the first safety-aware model-free MARL algorithms and that work effectively in the challenging tasks with safety constraints.

2. Related work

Safety is a long-standing pursuit in the development of robotic systems [16]. In safe reinforcement learning, [19], *Constrained Markov Decision Processes* (CMDPs) [4] are commonly employed to describe the safe control problem. At each step of decision making, the environment emits both rewards and costs of the decision, and agents need to maximise reward while avoid violating constraints on costs. Below, we review recent advances on safe RL for both single agent and multi-agent domains.

To tackle the learning problem in CMDPs, [2] introduced *Constrained Policy Optimisation* (CPO) which, like TRPO [41], updates the agent's policy under the trust region constraint to maximise a surrogate return, meanwhile obeying surrogate cost constraints. However, similar to TRPO, the computation of CPO may be expensive, because the Fisher information matrix still needs to be computed. An alternative solution is to apply primal-dual methods, giving rise to methods like TRPO-Lagrangian and PPO-Lagrangian [39]. Although these methods achieve impressive performance in terms of safety, their performance in terms of reward is poor [39]. Another class of algorithms that solves CMDPs is by [13,14]; these algorithms leverage the theoretical property of the Lyapunov functions and propose safe value iteration and policy gradient procedures. In contrast to CPO, [13,14] can work with off-policy methods; they also can be trained end-to-end with no need for line search that appears in CPO.

Safe multi-agent learning is an emerging research domain. Despite its importance [44], there are few sound solutions that work with MARL in a model-free setting. For example, CMIX [28] extends QMIX [38] by amending the reward function to take peak constraint violations into account, yet this approach does not come with safety guarantees during training. Furthermore, the majority of methods are designed specifically for learning robotics tasks. For example, the technique of barrier certificates [7,5,37] or model predictive shielding [51] from control theory are used to model safety. These methods, however, are derived from the traditional robotics point of view; unlike model-free MARL, they either are supervised learning based approaches, or require specific assumptions on the state space and environment dynamics. Moreover, due to the lack of a benchmark suite for safe MARL algorithms, the generalisation ability of those methods is unclear.

The most related work to ours is Safe Dec-PG [29], in which a primal-dual framework is adopted to find the saddle point between maximising reward and minimising cost. In particular, they proposed a decentralised policy gradient descent-ascent method through a consensus network. However, reaching a consensus equivalently imposes an extra constraint of parameter sharing among neighbouring agents, which could yield suboptimal solutions [24]. Furthermore, multi-agent policy gradient methods can suffer from high variance with growing state and action space sizes [25]. In contrast, our methods employ trust region optimisation and do not assume the parameter sharing.

HATRPO, introduced in [24], is the first multi-agent trust region method that enjoys theoretically-justified monotonic improvement guarantee. Its key idea is to make agents follow a sequential policy update scheme so that the expected joint advantage will always be positive, thus increasing reward. In this work, we show how to further develop this theory and derive a protocol which, in addition to the monotonic improvement, also guarantees to satisfy the safety constraints

at every iteration during learning. The resulting algorithm (Algorithm 1) successfully attains theoretical guarantees of both monotonic improvement in reward and satisfaction of safety constraints.

3. Problem formulation: constrained Markov game

We formulate the safe MARL problem as a *constrained Markov game* $\langle \mathcal{N}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \mathrm{p}, \rho^0, \gamma, R, \boldsymbol{C}, \boldsymbol{c} \rangle$. Here, $\mathcal{N} = \{1, \dots, n\}$ is the set of agents, $\mathcal{S}$ is the state space, $\boldsymbol{\mathcal{A}} = \prod_{i=1}^{n} \mathcal{A}^i$ is the product of the agents' action spaces, known as the joint action space, $\mathrm{p} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to \mathbb{R}$ is the probabilistic transition function, ρ^0 is the initial state distribution, $\gamma \in [0, 1)$ is the discount factor, $R : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ is the joint reward function, $\boldsymbol{C} = \{C_j^i\}_{1 \le j \le m^i}^{i \in \mathcal{N}}$ is the set of sets of cost functions (every agent i has m^i cost functions) of the form $C_j^i : \mathcal{S} \times \mathcal{A}^i \to \mathbb{R}$, and finally the set of corresponding cost-constraining values is given by $\boldsymbol{c} = \{c_j^i\}_{1 \le j \le m^i}^{i \in \mathcal{N}}$. At time step t, the agents are in a state s_t, and every agent i takes an action a_t^i according to its policy $\pi^i(\mathrm{a}^i|\mathrm{s}_t)$. Together with other agents' actions, it gives a joint action $\mathbf{a}_t = (\mathrm{a}_t^1, \dots, \mathrm{a}_t^n)$ and the joint policy $\boldsymbol{\pi}(\mathbf{a}|\mathrm{s}) = \prod_{i=1}^{n} \pi^i(\mathrm{a}^i|\mathrm{s})$. The agents receive the reward $R(\mathrm{s}_t, \mathbf{a}_t)$, meanwhile each agent i pays the costs $C_j^i(\mathrm{s}_t, \mathrm{a}_t^i)$, $\forall j = 1, \dots, m^i$. The environment then transits to a new state $\mathrm{s}_{t+1} \sim \mathrm{p}(\cdot|\mathrm{s}_t, \mathbf{a}_t)$. In this paper, we consider a *fully-cooperative* setting where all agents share the same reward function, aiming to maximise the expected total reward of

$$J(\boldsymbol{\pi}) \triangleq \mathbb{E}_{\mathrm{s}_0 \sim \rho^0, \mathbf{a}_{0:\infty} \sim \boldsymbol{\pi}, \mathrm{s}_{1:\infty} \sim \mathrm{p}} \Big[\sum_{t=0}^{\infty} \gamma^t R(\mathrm{s}_t, \mathbf{a}_t) \Big],$$

meanwhile trying to satisfy every agent i's safety constraints, written as

$$J_j^i(\boldsymbol{\pi}) \triangleq \mathbb{E}_{\mathrm{s}_0 \sim \rho^0, \mathbf{a}_{0:\infty} \sim \boldsymbol{\pi}, \mathrm{s}_{1:\infty} \sim \mathrm{p}} \Big[\sum_{t=0}^{\infty} \gamma^t C_j^i(\mathrm{s}_t, \mathrm{a}_t^i) \Big] \le c_j^i, \quad \forall j = 1, \dots, m^i. \tag{1}$$

We define the state-action value and the state-value functions in terms of reward as

$$Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \triangleq \mathbb{E}_{\mathrm{s}_{1:\infty} \sim \mathrm{p}, \mathbf{a}_{1:\infty} \sim \boldsymbol{\pi}} \Big[\sum_{t=0}^{\infty} \gamma^t R(\mathrm{s}_t, \mathbf{a}_t) \big| \mathrm{s}_0 = s, \mathbf{a}_0 = \boldsymbol{a} \Big],$$
$$V_{\boldsymbol{\pi}}(s) \triangleq \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}} \big[Q_{\boldsymbol{\pi}}(s, \mathbf{a}) \big].$$

The joint policies $\boldsymbol{\pi}$ that satisfy the Inequality (1) are referred to as **feasible**. Notably, in the above formulation, although the action a_t^i of agent i does not directly influence the costs $\{C_j^k(\mathrm{s}_t, \mathrm{a}_t^k)\}_{j=1}^{m^k}$ of other agents $k \neq i$, the action a_t^i will implicitly influence their total costs due to the dependence on the next state s_{t+1}. We believe that this formulation realistically describes multi-agent interactions in the real-world; an action of an agent has an instantaneous effect on the system only locally, but the rest of agents may suffer from its consequences at later stages. For example, we consider a car that crosses on the red light: although other cars may not be at risk of riding into pedestrians immediately, the induced traffic may cause hazards soon later. For the j^{th} cost function of agent i, we define the j^{th} state-action cost value function and the state cost value function as

$$Q_{j,\boldsymbol{\pi}}^i(s, a^i) \triangleq \mathbb{E}_{\mathbf{a}^{-i} \sim \boldsymbol{\pi}^{-i}, \mathrm{s}_{1:\infty} \sim \mathrm{p}, \mathbf{a}_{1:\infty} \sim \boldsymbol{\pi}} \Big[\sum_{t=0}^{\infty} \gamma^t C_j^i(\mathrm{s}_t, \mathrm{a}_t^i) \;\big|\; \mathrm{s}_0 = s, \mathrm{a}_0^i = a^i \Big],$$
$$V_{j,\boldsymbol{\pi}}^i(s) \triangleq \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}, \mathrm{s}_{1:\infty} \sim \mathrm{p}, \mathbf{a}_{1:\infty} \sim \boldsymbol{\pi}} \Big[\sum_{t=0}^{\infty} \gamma^t C_j^i(\mathrm{s}_t, \mathrm{a}_t^i) \;\big|\; \mathrm{s}_0 = s \Big].$$

Notably, the cost value functions $Q_{j,\boldsymbol{\pi}}^i$ and $V_{j,\boldsymbol{\pi}}^i$, although similar to traditional $Q_{\boldsymbol{\pi}}$ and $V_{\boldsymbol{\pi}}$, involve extra indices i and j; the superscript i denotes an agent, and the subscript j denotes its j^{th} cost.

Throughout this work, we pay close attention to the contribution to performance from different subsets of agents, therefore, we introduce the following notations. We denote an arbitrary subset $\{i_1, \dots, i_h\}$ of agents as $i_{1:h}$; we write $-i_{1:h}$ to refer to its complement. Given the agent subset $i_{1:h}$, we define the *multi-agent state-action value function*:

$$Q_{\boldsymbol{\pi}}^{i_{1:h}}(s, \boldsymbol{a}^{i_{1:h}}) \triangleq \mathbb{E}_{\mathbf{a}^{-i_{1:h}} \sim \boldsymbol{\pi}^{-i_{1:h}}} \Big[Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}^{i_{1:h}}, \mathbf{a}^{-i_{1:h}}) \Big].$$

On top of it, for disjoint sets $j_{1:k}$ and $i_{1:h}$,[3] the *multi-agent advantage function*[4] is defined as follows,

[3] Where $i_{1:0}$ denotes the empty set, $A^{i_1}(s, a^{i_{1:0}}, a^{i_1}) = A^{i_1}(s, a^{i_1})$, $j_{1:k}, i_{1:h}$ denotes a union of two sets of agents.

[4] We would like to highlight that these multi-agent functions of $Q_{\boldsymbol{\pi}}^{i_{1:h}}$ and $A_{\boldsymbol{\pi}}^{i_{1:h}}$, although involve agents in superscripts, describe values *w.r.t.* the reward rather than costs since they do not involve cost subscripts.

$$A_{\boldsymbol{\pi}}^{i_{1:h}}\left(s, \boldsymbol{a}^{j_{1:k}}, \boldsymbol{a}^{i_{1:h}}\right) \triangleq Q_{\boldsymbol{\pi}}^{j_{1:k}, i_{1:h}}\left(s, \boldsymbol{a}^{j_{1:k}}, \boldsymbol{a}^{i_{1:h}}\right) - Q_{\boldsymbol{\pi}}^{j_{1:k}}\left(s, \boldsymbol{a}^{j_{1:k}}\right). \tag{2}$$

An interesting fact about the above multi-agent advantage function is that any advantage $A_{\boldsymbol{\pi}}^{i_{1:h}}$ can be written as a sum of sequentially-unfolding multi-agent advantages of individual agents, and the advantage function is computed with GAE [42].

Lemma 1 *(Multi-agent advantage decomposition, [25]). For any state $s \in \mathcal{S}$, subset of agents $i_{1:h} \subseteq \mathcal{N}$, and joint action $\boldsymbol{a}^{i_{1:h}}$, the following identity holds*

$$A_{\boldsymbol{\pi}}^{i_{1:h}}\left(s, \boldsymbol{a}^{i_{1:h}}\right) = \sum_{j=1}^{h} A_{\boldsymbol{\pi}}^{i_j}\left(s, \boldsymbol{a}^{i_{1:j-1}}, a^{i_j}\right).$$

Proof. We write the multi-agent advantage as in its definition, and expand it in a telescoping sum, where $A_{\boldsymbol{\pi}}^{i_{1:h}}\left(s, \boldsymbol{a}^{i_{1:h}}\right)$ is the advantage of agents $i_{1:h}$ taking $\boldsymbol{a}^{1:h}$.

$$\begin{aligned} A_{\boldsymbol{\pi}}^{i_{1:h}}\left(s, \boldsymbol{a}^{i_{1:h}}\right) &= Q_{\boldsymbol{\pi}}^{i_{1:h}}\left(s, \boldsymbol{a}^{i_{1:h}}\right) - V_{\boldsymbol{\pi}}(s) \\ &= \sum_{j=1}^{h}\left[Q_{\boldsymbol{\pi}}^{i_{1:j}}\left(s, \boldsymbol{a}^{i_{1:j}}\right) - Q_{\boldsymbol{\pi}}^{i_{1:j-1}}\left(s, \boldsymbol{a}^{i_{1:j-1}}\right)\right] \\ &= \sum_{j=1}^{h} A_{\boldsymbol{\pi}}^{i_j}\left(s, \boldsymbol{a}^{i_{1:j-1}}, a^{i_j}\right). \quad \square \end{aligned}$$

4. Multi-agent constrained policy optimisation

In this section, we first present a theoretically-justified safe multi-agent policy iteration procedure that leverages multi-agent trust region learning and constrained policy optimisation to solve constrained Markov games. Based on this, we propose two practical deep MARL algorithms, enabling optimising neural-network based policies that satisfy safety constraints. Throughout this work, we refer the symbols $\boldsymbol{\pi}$ and $\bar{\boldsymbol{\pi}}$ to be the "current" and the "new" joint policies, respectively.

4.1. Multi-agent trust region learning with constraints

[24] introduced the first multi-agent trust region method—HATRPO—that enjoys theoretically-justified monotonic improvement guarantee. Specifically, it relies on the multi-agent advantage decomposition in Lemma 1, and the "surrogate" return that is given as follows.

Definition 1. Let $\boldsymbol{\pi}$ be a joint policy, $\bar{\boldsymbol{\pi}}^{i_{1:h-1}}$ be some other joint policy of agents $i_{1:h-1}$, and $\hat{\pi}^{i_h}$ be a policy of agent i_h. Then we define

$$L_{\boldsymbol{\pi}}^{i_{1:h}}\left(\bar{\boldsymbol{\pi}}^{i_{1:h-1}}, \hat{\pi}^{i_h}\right) \triangleq \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}^{i_{1:h-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:h-1}}, \mathrm{a}^{i_h}\sim\hat{\pi}^{i_h}}\left[A_{\boldsymbol{\pi}}^{i_h}\left(\mathrm{s}, \mathbf{a}^{i_{1:h-1}}, \mathrm{a}^{i_h}\right)\right].$$

For any $\bar{\boldsymbol{\pi}}^{i_{1:h-1}}$, we have

$$\begin{aligned} L_{\boldsymbol{\pi}}^{i_{1:h}}\left(\bar{\boldsymbol{\pi}}^{i_{1:h-1}}, \pi^{i_h}\right) &= \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}^{i_{1:h-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:h-1}}, \mathrm{a}^{i_h}\sim\pi^{i_h}}\left[A_{\boldsymbol{\pi}}^{i_h}\left(\mathrm{s}, \mathbf{a}^{i_{1:h-1}}, \mathrm{a}^{i_h}\right)\right] \\ &= \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}^{i_{1:h-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:h-1}}}\left[\mathbb{E}_{\mathrm{a}^{i_h}\sim\pi^{i_h}}\left[A_{\boldsymbol{\pi}}^{i_h}\left(\mathrm{s}, \mathbf{a}^{i_{1:h-1}}, \mathrm{a}^{i_h}\right)\right]\right] = 0. \end{aligned} \tag{3}$$

Building on Lemma 1 and Definition 1, we can observe,

$$L_{\boldsymbol{\pi}}^{i_{1:h}}(\bar{\boldsymbol{\pi}}^{i_{1:h-1}}, \bar{\pi}^{i_h}) = \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}^{i_{1:h-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:h-1}}, \mathrm{a}^{i_h}\sim\hat{\pi}^{i_h}}\left[\sum_{j=1}^{h} A_{\boldsymbol{\pi}}^{i_j}\left(s, \boldsymbol{a}^{i_{1:j-1}}, a^{i_j}\right)\right]. \tag{4}$$

With the above definition, we see that Lemma 1 allows for decomposing the joint surrogate return $L_{\boldsymbol{\pi}}(\bar{\boldsymbol{\pi}}) \triangleq \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}\sim\bar{\boldsymbol{\pi}}}[A_{\boldsymbol{\pi}}(\mathrm{s}, \mathbf{a})]$ into a sum over surrogates of $L_{\boldsymbol{\pi}}^{i_{1:h}}(\bar{\boldsymbol{\pi}}^{i_{1:h-1}}, \bar{\pi}^{i_h})$, for $h = 1, \ldots, n$. This can be used to justify that if agents, with a joint policy $\boldsymbol{\pi}$, update their policies by following a *sequential update scheme*, that is, if each agent in the subset $i_{1:h}$ sequentially solves the following optimisation problem:

$$\bar{\pi}^{i_h} = \arg\max_{\hat{\pi}^{i_h}} L^{i_{1:h}}_{\boldsymbol{\pi}}\left(\bar{\boldsymbol{\pi}}^{i_{1:h-1}}, \hat{\pi}^{i_h}\right) - \nu D^{\max}_{\mathrm{KL}}\left(\pi^{i_h}, \hat{\pi}^{i_h}\right),$$
$$\text{where } \nu = \frac{4\gamma \max_{s,\boldsymbol{a}} |A_{\boldsymbol{\pi}}(s,\boldsymbol{a})|}{(1-\gamma)^2}, \tag{5}$$
$$\text{and } D^{\max}_{\mathrm{KL}}(\pi^{i_h}, \hat{\pi}^{i_h}) \triangleq \max_s D_{\mathrm{KL}}(\pi^{i_h}(\cdot|s), \hat{\pi}^{i_h}(\cdot|s)),$$

then the resulting joint policy $\bar{\boldsymbol{\pi}}$ will surely improve the expected return, i.e., $J(\bar{\boldsymbol{\pi}}) \geq J(\boldsymbol{\pi})$ (see the proof in [24, Lemma 2]). We know that due to the penalty term $D^{\max}_{\mathrm{KL}}(\pi^{i_h}, \hat{\pi}^{i_h})$, the new policy $\bar{\pi}^{i_h}$ will stay close (*w.r.t.* max-KL distance) to π^{i_h}.

For the safety constraints, we can extend Definition 1 to incorporate the "surrogate" cost, thus allowing us to study the cost functions in addition to the return.

Definition 2. Let $\boldsymbol{\pi}$ be a joint policy, and $\bar{\pi}^i$ be some other policy of agent i. Then, for any of its costs of index $j \in \{1, \ldots, m^i\}$, we define

$$L^i_{j,\boldsymbol{\pi}}\left(\bar{\pi}^i\right) = \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathrm{a}^i\sim\bar{\pi}^i}\left[A^i_{j,\boldsymbol{\pi}}\left(\mathrm{s}, \mathrm{a}^i\right)\right].$$

By generalising the result about the surrogate return in Equation (1), we can derive how the expected costs change when the agents update their policies. Specifically, we provide the following lemma.

Lemma 2. *Let $\boldsymbol{\pi}$ and $\bar{\boldsymbol{\pi}}$ be joint policies. Let $i \in \mathcal{N}$ be an agent, and $j \in \{1, \ldots, m^i\}$ be an index of one of its costs. The following inequality holds*

$$J^i_j(\bar{\boldsymbol{\pi}}) \leq J^i_j(\boldsymbol{\pi}) + L^i_{j,\boldsymbol{\pi}}\left(\bar{\pi}^i\right) + \nu^i_j \sum_{h=1}^{n} D^{max}_{KL}\left(\pi^h, \bar{\pi}^h\right),$$

where $\nu^i_j = \frac{4\gamma \max_{s,a^i} |A^i_{j,\boldsymbol{\pi}}(s,a^i)|}{(1-\gamma)^2}$.

Proof. From the upper-bound version of Theorem 1 of [41] applied to joint policies $\boldsymbol{\pi}$ and $\bar{\boldsymbol{\pi}}$, we conclude that

$$J^i_j(\bar{\boldsymbol{\pi}}) \leq J^i_j(\boldsymbol{\pi}) + \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}\sim\bar{\boldsymbol{\pi}}}\left[A^i_{j,\boldsymbol{\pi}}(\mathrm{s}, \mathrm{a}^i)\right] + \frac{4\alpha^2\gamma \max_{s,a^i} |A^i_{j,\boldsymbol{\pi}}(s,a^i)|}{(1-\gamma)^2},$$

where $\alpha = D^{\max}_{\mathrm{TV}}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}) = \max_s D_{\mathrm{TV}}(\boldsymbol{\pi}(\cdot|s), \bar{\boldsymbol{\pi}}(\cdot|s))$.

Using Pinsker's inequality $D_{\mathrm{TV}}(p,q)^2 \leq D_{\mathrm{KL}}(p,q)/2$ [26,36,41], we obtain

$$J^i_j(\bar{\boldsymbol{\pi}}) \leq J^i_j(\boldsymbol{\pi}) + \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}\sim\bar{\boldsymbol{\pi}}}\left[A^i_{j,\boldsymbol{\pi}}(\mathrm{s}, \mathrm{a}^i)\right] + \frac{2\gamma \max_{s,a^i} |A^i_{j,\boldsymbol{\pi}}(s,a^i)|}{(1-\gamma)^2} D^{\max}_{\mathrm{KL}}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}),$$
$$\text{where } D^{\max}_{\mathrm{KL}}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}) = \max_s D_{\mathrm{KL}}(\boldsymbol{\pi}(\cdot|s), \bar{\boldsymbol{\pi}}(\cdot|s)).$$

Notice now that we have $\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}\sim\bar{\boldsymbol{\pi}}}\left[A^i_{j,\boldsymbol{\pi}}(\mathrm{s}, \mathrm{a}^i)\right] = \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathrm{a}^i\sim\bar{\pi}^i}\left[A^i_{j,\boldsymbol{\pi}}(\mathrm{s}, \mathrm{a}^i)\right]$ as the action of agents other that i do not change the value of the variable inside of the expectation. Furthermore

$$D^{\max}_{\mathrm{KL}}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}) = \max_s D_{\mathrm{KL}}(\boldsymbol{\pi}(\cdot|s), \bar{\boldsymbol{\pi}}(\cdot|s)) = \max_s \left(\sum_{l=1}^{n} D_{\mathrm{KL}}\left(\pi^l(\cdot|s), \bar{\pi}^l(\cdot|s)\right)\right)$$
$$\leq \sum_{l=1}^{n} \max_s D_{\mathrm{KL}}\left(\pi^l(\cdot|s), \bar{\pi}^l(\cdot|s)\right) = \sum_{l=1}^{n} D^{\max}_{\mathrm{KL}}\left(\pi^l, \bar{\pi}^l\right). \tag{6}$$

Setting $\nu^i_j = \frac{2\gamma \max_{s,a^i} |A^i_{j,\boldsymbol{\pi}}(s,a^i)|}{(1-\gamma)^2}$, we finally obtain

$$J^i_j(\bar{\boldsymbol{\pi}}) \leq J^i_j(\boldsymbol{\pi}) + L^i_{j,\boldsymbol{\pi}}\left(\bar{\pi}^i\right) + \nu^i_j \sum_{l=1}^{n} D^{\max}_{\mathrm{KL}}\left(\pi^l, \bar{\pi}^l\right). \quad \square$$

The above lemma suggests that, as long as the distances between the policies π^h and $\bar{\pi}^h$, $\forall h \in \mathcal{N}$, are sufficiently small, then the change in the j^{th} cost of agent i, i.e., $J^i_j(\bar{\boldsymbol{\pi}}) - J^i_j(\boldsymbol{\pi})$, is controlled by the surrogate $L^i_{j,\boldsymbol{\pi}}(\bar{\pi}^i)$. Importantly, this surrogate is independent of other agents' new policies. Hence, when the changes in policies of all agents are sufficiently small, each agent i can learn a better policy $\bar{\pi}^i$ by only considering its own surrogate return and surrogate costs. To summarise, we provide Algorithm 1 that guarantees both safety constraints satisfaction and monotonic performance improvement.

Algorithm 1: Safe multi-agent policy iteration with monotonic improvement property.

1: Initialise a joint policy $\boldsymbol{\pi}_0 = (\pi^1_0, \dots, \pi^n_0)$. // initial unsafe policies are allowed, see experiments.
2: **for** $k = 0, 1, \dots$ **do**
3: Compute the advantage functions $A_{\boldsymbol{\pi}_k}(s, \boldsymbol{a})$ and $A^i_{j,\boldsymbol{\pi}_k}(s, a^i)$, for all state-(joint)action pairs $(s, \boldsymbol{a})$, agents i, and constraints $j \in \{1, \dots, m^i\}$.
4: Compute $\nu = \frac{2\gamma \max_{s,\boldsymbol{a}} |A_{\boldsymbol{\pi}_k}(s,\boldsymbol{a})|}{(1-\gamma)^2}$, and $\nu^i_j = \frac{2\gamma \max_{s,a^i} |A^i_{j,\boldsymbol{\pi}_k}(s,a^i)|}{(1-\gamma)^2}$, $\forall i \in \mathcal{N}$, $j = 1, \dots, m^i$.
5: Draw a permutation $i_{1:n}$ of agents arbitrarily.
6: **for** $h = 1 : n$ **do**
7: Compute the radius of the KL-constraint δ^{i_h} // *see Equation (8) for the setup of* δ^{i_h}.
8: Make an update $\pi^{i_h}_{k+1} = \arg\max_{\pi^{i_h} \in \overline{\Pi}^{i_h}} \left[L^{i_{1:h}}_{\boldsymbol{\pi}_k}\left(\pi^{i_{1:h-1}}_{k+1}, \pi^{i_h}\right) - \nu D^{\max}_{\text{KL}}\left(\pi^{i_h}_k, \pi^{i_h}\right)\right]$,
where $\overline{\Pi}^{i_h}$ is a subset of safe policies of agent i_h, given by

$$\overline{\Pi}^{i_h} = \Big\{ \pi^{i_h} \in \Pi^{i_h} \mid D^{\max}_{\text{KL}}(\pi^{i_h}_k, \pi^{i_h}) \le \delta^{i_h}, \text{ and } J^{i_h}_j(\boldsymbol{\pi}_k) + L^{i_h}_{j,\boldsymbol{\pi}_k}(\pi^{i_h}) + \nu^{i_h}_j D^{\max}_{\text{KL}}(\pi^{i_h}_k, \pi^{i_h}) \le c^{i_h}_j - \sum_{l=1}^{h-1} \nu^{i_l}_j D^{\max}_{\text{KL}}(\pi^{i_l}_k, \pi^{i_l}), \forall j = 1, \dots, m^{i_h} \Big\}. \tag{7}$$

9: **end for**
10: **end for**

In this algorithm, in addition to sequentially maximising agents' surrogate returns, the agents must assure that their surrogate costs stay below the corresponding safety thresholds. Meanwhile, they have to constrain their policy search to small local neighbourhoods (*w.r.t.* max-KL distance). As such, Algorithm 1 demonstrates two desirable properties: reward performance improvement and satisfaction of safety constraints, as stated by Theorem 1.

In Algorithm 1, we compute the size of KL constraint as

$$\delta^{i_h} = \min \left\{ \min_{l \le h-1} \min_{1 \le j \le m^l} \frac{c^{i_l}_j - J^{i_l}_j(\boldsymbol{\pi}_k) - L^{i_l}_{j,\boldsymbol{\pi}_k}(\pi^{i_l}_{k+1}) - \nu^{i_l}_j \sum_{u=1}^{h-1} D^{\max}_{\text{KL}}(\pi^u_k, \pi^u_{k+1})}{\nu^{i_l}_j}, \min_{l \ge h+1} \min_{1 \le j \le m^l} \frac{c^{i_l}_j - J^{i_l}_j(\boldsymbol{\pi}_k) - \nu^{i_l}_j \sum_{u=1}^{h-1} D^{\max}_{\text{KL}}(\pi^u_k, \pi^u_{k+1})}{\nu^{i_l}_j} \right\}. \tag{8}$$

Note that δ^{i_1} (i.e., $h = 1$) is guaranteed to be non-negative if $\boldsymbol{\pi}_k$ satisfies safety constraints; that is because then $c^{i_l}_j \ge J^{i_l}_j(\boldsymbol{\pi}_k)$ for all l and j, and the set $\{l \mid l < h\}$ is empty.

This formula for δ^{i_h}, combined with Lemma 2, assures that the policies π^{i_h} within δ^{i_h} max-KL distance from $\pi^{i_h}_k$ will not violate other agents' safety constraints, as long as the base joint policy $\boldsymbol{\pi}_k$ did not violate them (which assures $\delta^{i_1} \ge 0$). To see this, notice that for every $l = 1, \dots, h-1$, and $j = 1, \dots, m^l$,

$$D^{\max}_{\text{KL}}(\pi^{i_h}_k, \pi^{i_h}) \le \delta^{i_h} \le \frac{c^{i_l}_j - J^{i_l}_j(\boldsymbol{\pi}_k) - L^{i_l}_{j,\boldsymbol{\pi}_k}(\pi^{i_l}_{k+1}) - \nu^{i_l}_j \sum_{u=1}^{h-1} D^{\max}_{\text{KL}}(\pi^u_k, \pi^u_{k+1})}{\nu^{i_l}_j},$$

$$\text{implies } J^{i_l}_j(\boldsymbol{\pi}_k) + L^{i_l}_{j,\boldsymbol{\pi}_k}(\pi^{i_l}_{k+1}) + \nu^{i_l}_j \sum_{u=1}^{h-1} D^{\max}_{\text{KL}}(\pi^u_k, \pi^u_{k+1}) + \nu^{i_l}_j D^{\max}_{\text{KL}}(\pi^{i_h}_k, \pi^{i_h}) \le c^{i_l}_j.$$

By Lemma 2, the left-hand side of the above inequality is an upper bound of $J^{i_l}_j(\boldsymbol{\pi}^{i_{1:h-1}}_{k+1}, \pi^{i_h}, \boldsymbol{\pi}^{-i_{1:h}}_k)$, which implies that the update of agent i_h does not violate the constraint of $J^{i_l}_j$. The fact that the constraints of $J^{i_l}_j$ for $l \ge h+1$ are not violated, i.e.,

$$J^{i_l}_j(\boldsymbol{\pi}_k) + \nu^{i_l}_j \sum_{u=1}^{h-1} D^{\max}_{\text{KL}}(\pi^u_k, \pi^u_{k+1}) + \nu^{i_l}_j D^{\max}_{\text{KL}}(\pi^{i_h}_k, \pi^{i_h}) \le c^{i_l}_j, \tag{9}$$

is analogous.

Theorem 1. *If a sequence of joint policies $(\boldsymbol{\pi}_k)_{k=0}^{\infty}$ is obtained from Algorithm 1, then it has the monotonic improvement property, $J(\boldsymbol{\pi}_{k+1}) \geq J(\boldsymbol{\pi}_k)$, as well as it satisfies the safety constraints, $J_j^i(\boldsymbol{\pi}_k) \leq c_j^i$, for all $k \in \mathbb{N}$, $i \in \mathcal{N}$, and $j \in \{1, \ldots, m^i\}$.*

Proof. Safety constraints are assured to be met if the agents' update size is bounded by Equation (8). It suffices to show the monotonic improvement property. By Theorem 1 from [41], we have

$$J(\boldsymbol{\pi}_{k+1}) \geq J(\boldsymbol{\pi}_k) + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_k}, \mathbf{a}\sim\boldsymbol{\pi}_{k+1}}\left[A_{\boldsymbol{\pi}_k}(s, \mathbf{a})\right] - \nu D_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}_k, \boldsymbol{\pi}_{k+1}),$$

which by Equation (6) is lower-bounded by

$$\geq J(\boldsymbol{\pi}_k) + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_k}, \mathbf{a}\sim\boldsymbol{\pi}_{k+1}}\left[A_{\boldsymbol{\pi}_k}(s, \mathbf{a})\right] - \sum_{h=1}^{n} \nu D_{\mathrm{KL}}^{\max}(\pi_k^{i_h}, \pi_{k+1}^{i_h})$$

which by Lemma 1 equals

$$= J(\boldsymbol{\pi}_k) + \sum_{h=1}^{n} \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_k}, \mathbf{a}^{i_{1:h}}\sim\boldsymbol{\pi}_{k+1}^{i_{1:h}}}\left[A_{\boldsymbol{\pi}_k}^{i_h}(s, \mathbf{a}^{i_{1:h-1}}, a^{i_h})\right] - \sum_{h=1}^{n} \nu D_{\mathrm{KL}}^{\max}(\pi_k^{i_h}, \pi_{k+1}^{i_h})$$

$$= J(\boldsymbol{\pi}_k) + \sum_{h=1}^{n} \left(L_{\boldsymbol{\pi}_k}^{i_{1:h}}(\boldsymbol{\pi}_{k+1}^{i_{1:h-1}}, \pi_{k+1}^{i_h}) - \nu D_{\mathrm{KL}}^{\max}(\pi_k^{i_h}, \pi_{k+1}^{i_h})\right), \tag{10}$$

and as for every h, $\pi_{k+1}^{i_h}$ is the argmax, this is lower-bounded by

$$\geq J(\boldsymbol{\pi}_k) + \sum_{h=1}^{n} \left(L_{\boldsymbol{\pi}_k}^{i_{1:h}}(\boldsymbol{\pi}_{k+1}^{i_{1:h-1}}, \pi_k^{i_h}) - \nu D_{\mathrm{KL}}^{\max}(\pi_k^{i_h}, \pi_k^{i_h})\right),$$

which, as follows from Definition 1 and Equation (3), equals

$$= \mathcal{J}(\boldsymbol{\pi}_{\boldsymbol{k}}) + \sum_{h=1}^{n} 0 = J(\boldsymbol{\pi}_k), \text{ which finishes the proof.} \quad \square$$

Theorem 1 assures that agents following Algorithm 1 will not only explore safe policies; meanwhile, every new policy will be guaranteed to result in performance improvement. These two properties hold under the condition that only restrictive policy updates are made; this is due to the KL-penalty term in every agent's objective (i.e., $\nu D_{\mathrm{KL}}^{\max}(\pi_k^{i_h}, \pi^{i_h})$), as well as the constraints on cost surrogates (i.e., the conditions in $\overline{\Pi}^{i_h}$). In practice, it can be intractable to evaluate $D_{\mathrm{KL}}\big(\pi_k^{i_h}(\cdot|s), \pi^{i_h}(\cdot|s)\big)$ at every state in order to compute $D_{\mathrm{KL}}^{\max}(\pi_k^{i_h}, \pi^{i_h})$. In the following subsections, we describe how to approximate Algorithm 1 in the case of parameterised policies.

4.2. MACPO: multi-agent constrained policy optimisation

In this section, we focus on the practical implementation where large state and action spaces prevent agents from designating policies $\pi^i(\cdot|s)$ for each state separately. To handle this, we parameterise each agent's policy $\pi_{\theta^i}^i$ by a neural network with parameter θ^i. Correspondingly, the joint policies $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ are parametrised by $\boldsymbol{\theta} = (\theta^1, \ldots, \theta^n)$.

At each iteration of Algorithm 1, every agent i_h maximises its surrogate return with a KL-penalty, subject to surrogate cost constraint. Yet, direct computation of the max-KL constraint is intractable in practical settings, as it would require computation of KL-divergence at every single state. Instead, we relax it by adopting a form of expected KL-constraint

$$\overline{D}_{\mathrm{KL}}(\pi_k^{i_h}, \pi^{i_h}) \leq \delta,$$

where $\overline{D}_{\mathrm{KL}}(\pi_k^{i_h}, \pi^{i_h}) \triangleq \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_k}}\big[D_{\mathrm{KL}}(\pi_k^{i_h}(\cdot|s), \pi^{i_h}(\cdot|s))\big]$.

We approximate these expectations by sampling. As a result, the optimisation problem solved by agent i_h can be written as

$$\theta_{k+1}^{i_h} = \arg\max_{\theta^{i_h}} \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}, \mathbf{a}^{i_{1:h-1}}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_{k+1}}^{i_{1:h-1}}, a^{i_h}\sim\pi_{\theta^{i_h}}^{i_h}}\left[A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h}\left(s, \mathbf{a}^{i_{1:h-1}}, a^{i_h}\right)\right]$$

$$\text{s.t.} \quad J_j^{i_h}\left(\boldsymbol{\pi}_{\boldsymbol{\theta}_k}\right) + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}, a^{i_h}\sim\pi_{\theta_k^{i_h}}^{i_h}}\left[A_{j,\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h}\left(s, a^{i_h}\right)\right] \leq c_j^{i_h}, \ \forall j \in \{1, \ldots, m^{i_h}\},$$

$$\text{and } \overline{D}_{\mathrm{KL}}\left(\pi_k^{i_h}, \pi^{i_h}\right) \leq \delta. \tag{11}$$

To solve this problem, we approximate Equation (11) with the Taylor expansion of the optimisation objective and cost constraints up to the first order, and the KL-divergence up to the second order. Consequently, the optimisation problem (11) can be written as

$$\begin{aligned}\theta_{k+1}^{i_h} = \arg\max_{\theta^{i_h}} & \left(\boldsymbol{g}^{i_h}\right)^T \left(\theta^{i_h} - \theta_k^{i_h}\right) \\ \text{s.t. } & d_j^{i_h} + \left(\boldsymbol{b}_j^{i_h}\right)^T \left(\theta^{i_h} - \theta_k^{i_h}\right) \leq 0, \quad j = 1, \ldots, m \\ & \frac{1}{2}\left(\theta^{i_h} - \theta_k^{i_h}\right)^T \boldsymbol{H}^{i_h} \left(\theta^{i_h} - \theta_k^{i_h}\right) \leq \delta,\end{aligned} \tag{12}$$

where $\boldsymbol{g}^{i_h}$ is the gradient of the objective of agent i_h in Equation (11), $d_j^i = J_j^i(\boldsymbol{\pi}_{\boldsymbol{\theta}_k}) - c_j^i$ and $\boldsymbol{H}^{i_h} = \nabla^2_{\theta^{i_h}} \overline{D}_{\mathrm{KL}}(\pi^{i_h}_{\theta_k^{i_h}}, \pi^{i_h})\big|_{\theta^{i_h}=\theta_k^{i_h}}$, is the Hessian of the average KL divergence of agent i_h, and $\boldsymbol{b}_j^{i_h}$ is the gradient of agent of the j^{th} constraint of agent i_h.

Similar to [2] and [12], we can take a primal-dual optimisation approach to solve the linear quadratic optimisation in Equation (12). Specifically, the dual form is given by

$$\max_{\lambda^{i_h} \geq 0, \mathbf{v}^{i_h} \succeq 0} \frac{-1}{2\lambda^{i_h}}\left[(\boldsymbol{g}^{i_h})^T (\boldsymbol{H}^{i_h})^{-1} \boldsymbol{g}^{i_h} - 2(\boldsymbol{r}^{i_h})^T \mathbf{v}^{i_h} + (\mathbf{v}^{i_h})^T \boldsymbol{S}^{i_h}\right] + (\mathbf{v}^{i_h})^T \boldsymbol{c}^{i_h} - \frac{\lambda^{i_h}\delta}{2}, \tag{13}$$

where $\boldsymbol{B}^{i_h} = \left[\boldsymbol{b}_1^{i_h}, \ldots, \boldsymbol{b}_m^{i_h}\right]$, $\boldsymbol{r}^{i_h} \triangleq (\boldsymbol{g}^{i_h})^T (\boldsymbol{H}^{i_h})^{-1} \boldsymbol{B}^{i_h}$, $\boldsymbol{S}^{i_h} \triangleq (\boldsymbol{B}^{i_h})^T (\boldsymbol{H}^{i_h})^{-1} \boldsymbol{B}^{i_h}$.

Given the solution to the dual form in Equation (13), i.e., $\lambda_*^{i_h}$ and $\mathbf{v}_*^{i_h}$, the solution to the primal problem in Equation (12) can thus be written by

$$\theta_*^{i_h} = \theta_k^{i_h} + \frac{1}{\lambda_*^{i_h}}\left(\boldsymbol{H}^{i_h}\right)^{-1}\left(\boldsymbol{g}^{i_h} - \boldsymbol{B}^{i_h}\mathbf{v}_*^{i_h}\right).$$

See Appendix B for derivation. In practice, we use backtracking line search starting at $1/\lambda_*^{i_h}$ to choose the step size of the above update. Furthermore, we note that the optimisation step in Equation (12) is an approximation to the original problem from Equation (11); therefore, it is possible that an infeasible policy $\pi_{\theta_{k+1}^{i_h}}$ will be generated. Fortunately, as the policy optimisation takes place in the trust region of $\pi^{i_h}_{\theta_k^{i_h}}$, the size of update is small, and a feasible policy can be easily recovered. We do so by means of a negative TRPO step applied on the cost of agent i_h:

$$\theta_{k+1}^{i_h} = \theta_k^{i_h} - \alpha^j \sqrt{\frac{2\delta}{{\boldsymbol{b}^{i_h}}^T (\boldsymbol{H}^{i_h})^{-1} \boldsymbol{b}^{i_h}}} \left(\boldsymbol{H}^{i_h}\right)^{-1} \boldsymbol{b}^{i_h}. \tag{14}$$

To put it together, we refer to this algorithm as *MACPO*, and provide its pseudocode in Algorithm 2. The computational complexity of MACPO is $\mathcal{O}(KMHP^3)$, where K denotes the number of steps, M denotes the number of agents, H denotes the number of constraints, P denotes the number of policy parameters.

$$\hat{\boldsymbol{g}}_k^{i_h} = \frac{1}{B}\sum_{b=1}^{B}\sum_{t=1}^{T} \nabla_{\theta_k^{i_h}} \log \pi^{i_h}_{\theta_k^{i_h}}\left(a_t^{i_h} \mid o_t^{i_h}\right) M^{i_{1:h}}(s_t, \boldsymbol{a}_t), \tag{15}$$

$$\hat{\boldsymbol{b}}_j^{i_h} = \frac{1}{B}\sum_{b=1}^{B}\sum_{t=1}^{T} \nabla_{\theta_k^{i_h}} \log \pi^{i_h}_{\theta_k^{i_h}}\left(a_t^{i_h} | o_t^{i_h}\right) \hat{A}_j^{i_h}(s_t, a_t^{i_h}), \tag{16}$$

$$\boldsymbol{x}_k^{i_h} = (\hat{\boldsymbol{H}}_k^{i_h})^{-1}\left(\boldsymbol{g}_k^{i_h} - \hat{\boldsymbol{B}}^{i_h}\mathbf{v}_*^{i_h}\right), \tag{17}$$

$$\theta_{k+1}^{i_h} = \theta_k^{i_h} + \frac{\alpha^j}{\lambda_*^{i_h}}\hat{\boldsymbol{x}}_k^{i_h}, \tag{18}$$

$$M^{i_{1:h+1}}(\mathrm{s}, \mathbf{a}) = \frac{\pi^{i_h}_{\theta_{k+1}^{i_h}}\left(\mathrm{a}^{i_h} \mid \mathrm{o}^{i_h}\right)}{\pi^{i_h}_{\theta_k^{i_h}}\left(\mathrm{a}^{i_h} \mid \mathrm{o}^{i_h}\right)} M^{i_{1:h}}(\mathrm{s}_t, \mathbf{a}_t), \tag{19}$$

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{\mathcal{N}}\frac{1}{T}\sum_{n=1}^{\mathcal{N}}\sum_{t=0}^{T}\left(V_\phi(s_t) - \hat{R}_t\right)^2. \tag{20}$$

Algorithm 2: MACPO.

Input: Stepsize α, batch size B, number of: agents n, episodes K, steps per episode T, possible steps in line search L.
Initialise: Actor networks $\{\theta_0^i,\ \forall i \in \mathcal{N}\}$, V-value network $\{\phi_0\}$, V^i-cost networks $\{\phi_{j,0}^i\}_{i=1:n,j=1:m}$, Replay buffer $\mathcal{B}$.
for $k = 0, 1, \ldots, K-1$ **do**
  Collect a set of trajectories by running the joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_k}$.
  Push transitions $\{(o_t^i, a_t^i, o_{t+1}^i, r_t), \forall i \in \mathcal{N}, t \in T\}$ into $\mathcal{B}$.
  Sample a random minibatch of M transitions from $\mathcal{B}$.
  Compute advantage function $\hat{A}(s, \mathbf{a})$ based on V-value network with GAE.
  Compute cost-advantage functions $\hat{A}_j^i(s, a^i)$ based on V^i-cost critics with GAE.
  Draw a random permutation of agents $i_{1:n}$.
  Set $M^{i_1}(s, \mathbf{a}) = \hat{A}(s, \mathbf{a})$.
  for agent $i_h = i_1, \ldots, i_n$ **do**
    Estimate the gradient of the agent's maximisation objective (15).
    for $j = 1, \ldots, m^{i_h}$ **do**
      Estimate the gradient of the agent's j^{th} cost (16).
    end for
    Set $\hat{\boldsymbol{B}}^{i_h} = \left[\hat{\boldsymbol{b}}_1^{i_h}, \ldots, \hat{\boldsymbol{b}}_m^{i_h}\right]$.
    Compute $\hat{\boldsymbol{H}}_k^{i_h}$, the Hessian of the average KL-divergence

$$\frac{1}{BT}\sum_{b=1}^{B}\sum_{t=1}^{T} D_{\text{KL}}\left(\pi_{\theta_k^{i_h}}^{i_h}(\cdot|o_t^{i_h}), \pi_{\theta^{i_h}}^{i_h}(\cdot|o_t^{i_h})\right).$$

    if the problem is feasible **then**
      Solve the dual (13) for $\lambda_*^{i_h}, \mathbf{v}_*^{i_h}$.
      Use the conjugate gradient algorithm to compute the update direction (17).
      Update agent i_h's policy by (18), where $j \in \{0, 1, \ldots, L\}$ is the smallest such j which improves the sample loss, and satisfies the sample constraints, found by the backtracking line search.
    else
      Use equation (14) to recover policy $\theta_{k+1}^{i_h}$ from unfeasible points.
    end if
    Compute (19). //Unless $h = n$.
  end for
  Update V-value network by formula (20).
end for

4.3. MAPPO-Lagrangian

As an alternative to MACPO, one can use Lagrangian multipliers in place of optimisation with linear and quadratic constraints to solve Equation (11). The Lagrangian method is simple to implement, and it does not require the repetitive computation of the Hessian $\boldsymbol{H}^{i_h}$ whose size grows quadratically with the dimension of the parameter vector θ^{i_h}.

Before we proceed, let us briefly recall the optimisation procedure with a Lagrangian multiplier. Suppose that our goal is to maximise a bounded real-valued function $f(x)$ under a constraint $g(x)$; $\max_x f(x)$, s.t. $g(x) \leq 0$. We can introduce a scalar variable λ and reformulate the optimisation by

$$\max_x \min_{\lambda \geq 0} f(x) - \lambda g(x). \tag{21}$$

Suppose that x_+ satisfies $g(x_+) > 0$. This immediately implies that $-\lambda g(x_+) \to -\infty$, as $\lambda \to +\infty$, and so Equation (21) equals $-\infty$ for $x = x_+$. On the other hand, if x_- satisfies $g(x_-) < 0$, we have that $-\lambda g(x_-) \geq 0$, with equality only for $\lambda = 0$. In that case, the optimisation objective's value equals $f(x_-) > -\infty$. Hence, the only candidate solutions to the problem are those x that satisfy the constraint $g(x) \leq 0$, and the objective matches with $f(x)$.

We can employ the above trick to the constrained optimisation problem from Equation (11) by subsuming the cost constraints into the optimisation objective with Lagrangian multipliers. As such, agent i_h computes $\bar{\lambda}_{1:m^{i_h}}^{i_h}$ and $\theta_{k+1}^{i_h}$ to solve the following min-max optimisation problem

$$\max_{\theta^{i_h}} \min_{\lambda_{1:m^{i_h}}^{i_h} \geq 0} \Bigg[\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}, \mathbf{a}^{i_{1:h-1}}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_{k+1}}^{i_{1:h-1}}, a^{i_h}\sim\pi_{\theta^{i_h}}^{i_h}} \left[A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h}\left(s, \mathbf{a}^{i_{1:h-1}}, a^{i_h}\right)\right] - \sum_{u=1}^{m^{i_h}} \lambda_u^{i_h}\left(\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}, a^{i_h}\sim\pi_{\theta^{i_h}}^{i_h}}\left[A_{u,\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h}\left(s, a^{i_h}\right)\right] + d_u^{i_h}\right)\Bigg],$$
$$\text{s.t. } \overline{D}_{\text{KL}}\left(\pi_{\theta_k^{i_h}}^{i_h}, \pi_{\theta^{i_h}}^{i_h}\right) \leq \delta. \tag{22}$$

Although the objective from Equation (22) is affine in the Lagrangian multipliers $\lambda_u^{i_h}$ $(u = 1, \dots, m^{i_h})$, which enables gradient-based optimisation, computing the KL-divergence constraint still complicates the overall process. To handle this, one can further simplify it by adopting the *PPO-clip* objective [43], which replaces the KL-divergence constraint with the *clip* operator, and updates the policy parameter with first-order methods. We do so by defining

$$A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h,(\lambda)}\left(s, \boldsymbol{a}^{i_{1:h-1}}, a^{i_h}\right) \triangleq A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h}\left(s, \boldsymbol{a}^{i_{1:h-1}}, a^{i_h}\right) - \sum_{u=1}^{m^{i_h}} \lambda_u^{i_h}\left(A_{u,\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h}\left(s, a^{i_h}\right) + d_u^{i_h}\right),$$

and rewriting the Equation (22) as

$$\begin{aligned}&\max_{\theta^{i_h}} \min_{\substack{\lambda_{1:m^{i_h}}^{i_h} \geq 0}} \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}, \mathbf{a}^{i_{1:h-1}}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_{k+1}}^{i_{1:h-1}}, \mathrm{a}^{i_h}\sim\pi_{\theta^{i_h}}^{i_h}} \left[A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h,(\lambda)}\left(\mathrm{s}, \mathbf{a}^{i_{1:h-1}}, \mathrm{a}^{i_h}\right)\right],\\ &\text{s.t. } \overline{D}_{\mathrm{KL}}\left(\pi_{\theta_k^{i_h}}^{i_h}, \pi_{\theta^{i_h}}^{i_h}\right) \leq \delta.\end{aligned} \tag{23}$$

The objective in Equation (23) takes a form of an expectation with a KL-divergence constraint on the policy. To approximately adjust the distance between old and new policies, the *clip* operator is leveraged. Finally, the objective takes the form

$$\begin{aligned}\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}, \mathbf{a}^{i_{1:h-1}}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_{k+1}}^{i_{1:h-1}}, \mathrm{a}^{i_h}\sim\pi_{\theta_k^{i_h}}^{i_h}} \Bigg[\min\Bigg(&\frac{\pi_{\theta^{i_h}}^{i_h}(\mathrm{a}^{i_h}|\mathrm{s})}{\pi_{\theta_k^{i_h}}^{i_h}(\mathrm{a}^{i_h}|\mathrm{s})} A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h,(\lambda)}\left(\mathrm{s}, \mathbf{a}^{i_{1:h-1}}, \mathrm{a}^{i_h}\right),\\ &\mathrm{clip}\left(\frac{\pi_{\theta^{i_h}}^{i_h}(\mathrm{a}^{i_h}|\mathrm{s})}{\pi_{\theta_k^{i_h}}^{i_h}(\mathrm{a}^{i_h}|\mathrm{s})}, 1\pm\epsilon\right) A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}^{i_h,(\lambda)}\left(\mathrm{s}, \mathbf{a}^{i_{1:h-1}}, \mathrm{a}^{i_h}\right)\Bigg)\Bigg].\end{aligned} \tag{24}$$

The clip operator replaces the policy ratio with $1-\epsilon$, or $1+\epsilon$, depending on whether its value is below or above the threshold interval. As such, agent i_h can learn within its trust region by updating θ^{i_h} to maximise Equation (24), while the Lagrangian multipliers are updated towards the direction opposite to their gradients of Equation (22), which can be computed analytically. We name this algorithm *MAPPO-Lagrangian*, and give its pseudocode in Algorithm 3. The computational complexity of MAPPO-L is $\mathcal{O}(KMNHP)$, where K denotes the number of steps, M denotes the number of agents, N denotes the number of PPO-Epoch, H denotes the number of constraints, P denotes the number of policy parameters.

$$M^{i_h,(\lambda)}(\mathrm{s}_t, \mathbf{a}_t) = M^{i_h}(\mathrm{s}_t, \mathbf{a}_t) - \sum_{j=1}^{n} \lambda_j^{i_h} \hat{A}_j^{i_h}(\mathrm{s}_t, \mathrm{a}_t^{i_h}), \tag{25}$$

$$\begin{aligned}\Delta_{\theta^{i_h}} = \nabla_{\theta^{i_h}} \frac{1}{B} \sum_{b=1}^{B} \sum_{t=0}^{T} \min\Bigg(&\frac{\pi_{\theta^{i_h}}^{i_h}\left(a_t^{i_h} \mid o_t^{i_h}\right)}{\pi_{\theta_k^{i_h}}^{i_h}\left(a_t^{i_h} \mid o_t^{i_h}\right)} M^{i_h,(\lambda)}(s_t, \boldsymbol{a}_t),\\ &\mathrm{clip}\left(\frac{\pi_{\theta^{i_h}}^{i_h}\left(a_t^{i_h} \mid o_t^{i_h}\right)}{\pi_{\theta_k^{i_h}}^{i_h}\left(a_t^{i_h} \mid o_t^{i_h}\right)}, 1\pm\epsilon\right) M^{i_h,(\lambda)}(s_t, \boldsymbol{a}_t)\Bigg),\end{aligned} \tag{26}$$

$$d_j^{i_h} = \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} \hat{V}_j^{i_h}(s_t) - c_j^{i_h}, \tag{27}$$

$$\Delta\lambda_j^{i_h} = \frac{-1}{B} \sum_{b=1}^{B} \left(d_j^{i_h}(1-\gamma) + \sum_{t=0}^{T} \frac{\pi_{\theta^{i_h}}^{i_h}(a_t^{i_h}|o_t^{i_h})}{\pi_{\theta_k^{i_h}}^{i_h}(a_t^{i_h}|o_t^{i_h})} \hat{A}_j^{i_h}(s_t, a_t^{i_h})\right), \tag{28}$$

$$\lambda_j^{i_h} \leftarrow \left(\lambda_j^{i_h} - \alpha_\lambda \Delta\lambda_j^{i_h}\right)_+, \tag{29}$$

$$M^{i_{h+1}}(\mathrm{s}, \mathbf{a}) = \frac{\pi_{\theta_{k+1}^{i_h}}^{i_h}\left(\mathrm{a}^{i_h} \mid \mathrm{o}^{i_h}\right)}{\pi_{\theta_k^{i_h}}^{i_h}\left(\mathrm{a}^{i_h} \mid \mathrm{o}^{i_h}\right)} M^{i_h}(\mathrm{s}, \mathbf{a}), \tag{30}$$

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t\right)^2. \tag{31}$$

Algorithm 3: MAPPO-Lagrangian.

Input: Stepsizes $\alpha_\theta, \alpha_\lambda$, batch size B, number of: agents n, episodes K, steps per episode T, discount factor γ.
Initialise: Actor networks $\{\theta_0^i,\ \forall i \in \mathcal{N}\}$, V-value network $\{\phi_0\}$,
V-cost networks $\{\phi_{j,0}^i\}_{1\leq j\leq m^i}^{i\in\mathcal{N}}$, Replay buffer $\mathcal{B}$.
for $k = 0, 1, \ldots, K-1$ **do**
  Collect a set of trajectories by running the joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_k}$.
  Push transitions $\{(o_t^i, a_t^i, o_{t+1}^i, r_t), \forall i \in \mathcal{N}, t \in T\}$ into $\mathcal{B}$.
  Sample a random minibatch of B transitions from $\mathcal{B}$.
  Compute advantage function $\hat{A}(\mathrm{s}, \mathbf{a})$ based on V-value network with GAE.
  Compute cost advantage functions $\hat{A}_j^i(\mathrm{s}, \mathrm{a}^i)$ for all agents and costs,
  based on V-cost networks with GAE.
  Draw a random permutation of agents $i_{1:n}$.
  Set $M^{i_1}(\mathrm{s}, \mathbf{a}) = \hat{A}(\mathrm{s}, \mathbf{a})$.
  for agent $i_h = i_1, \ldots, i_n$ **do**
    Initialise a policy parameter $\theta^{i_h} = \theta_k^{i_h}$,
    and Lagrangian multipliers $\lambda_j^{i_h} = 0$, $\forall j = 1, \ldots, m^{i_h}$.
    Compute the Lagrangian in Equation (25).
    for $e = 1, \ldots, e_{\text{PPO}}$ **do**
      Compute the gradient $\Delta_{\theta^{i_h}}$ of the Lagrangian PPO-Clip objective with Equation (26).
      Update temporarily the actor parameters

$$\theta^{i_h} \leftarrow \theta^{i_h} + \alpha_\theta \Delta_{\theta^{i_h}}.$$

      for $j = 1, \ldots, m^{i_h}$ **do**
        Approximate the constraint violation via Equation (27).
        Differentiate the constraint via Equation (28).
      end for
      for $j = 1, \ldots, m^{i_h}$ **do**
        Update temporarily the Lagrangian multiplier as in Equation (29).
      end for
    end for
    Update the actor parameter $\theta_{k+1}^{i_h} = \theta^{i_h}$.
    Compute Equation (30). //Unless $h = n$.
  end for
  Update the critic via Equation (31).
end for

5. Experiments

In this section, first, we introduce three safe MARL benchmarks. Second, we provide and analyse results of experiments conducted in these environments. Lastly, for completeness, we give the details of settings for the experiments at the end of the section.

5.1. Safe Multi-Agent MuJoCo

Although MARL researchers have long had a variety of environments to test different algorithms, such as StarCraftII [40] and Multi-Agent MuJoCo [35], no public safe MARL benchmark has been proposed; this impedes researchers from evaluating and benchmarking safety-aware multi-agent learning methods. As a key contribution of this paper, we introduce three Safe MARL benchmarks, Safe MAMuJoCo, Safe MARobosuite and Safe MAIG. We use these environments to evaluate the performance of our methods, in terms of reward and safety, against strong baselines.

5.1.1. Safe Multi-Agent MuJoCo: the environment

For visualisation see Fig. 1. Safe MAMuJoCo is an extension of MAMuJoCo [35]. In particular, the background environment, agents, physics simulator, and the reward function are preserved. However, as opposed to its predecessor, Safe MAMuJoCo environments come with obstacles, like walls or pitfalls. Specifically, Furthermore, with the increasing risk of an agent stumbling upon an obstacle, the environment emits cost [8]. To test our algorithms on the Ant and ManyAgent Ant tasks in terms of reward and safety performance, and see how safe the robots walk cooperatively in static obstacle environments. The Safe ManyAgent Ant, Humanoid, and Ant tasks are proposed, the Safe HalfCheetah and Hopper tasks are developed in dynamic obstacle environments. According to the scheme from [50], we characterise the cost functions for each task below.

ManyAgent ant task 1.0 & ant task 1.0

The width of the corridor set by two walls is 9 m (ManyAgent Ant with 2 agents (2x3), 3 agents (3x2), 6 agents (6x1))/10 m (Ant with 2 (2x4, 2x4d), 4 (4x2), 8 (8x1) agents). The environment emits the cost of 1 for an agent, if the distance between the robot and the wall is less than 1.8 m, or when the robot topples over, see Fig. 2 (a) and (b). This can be described as

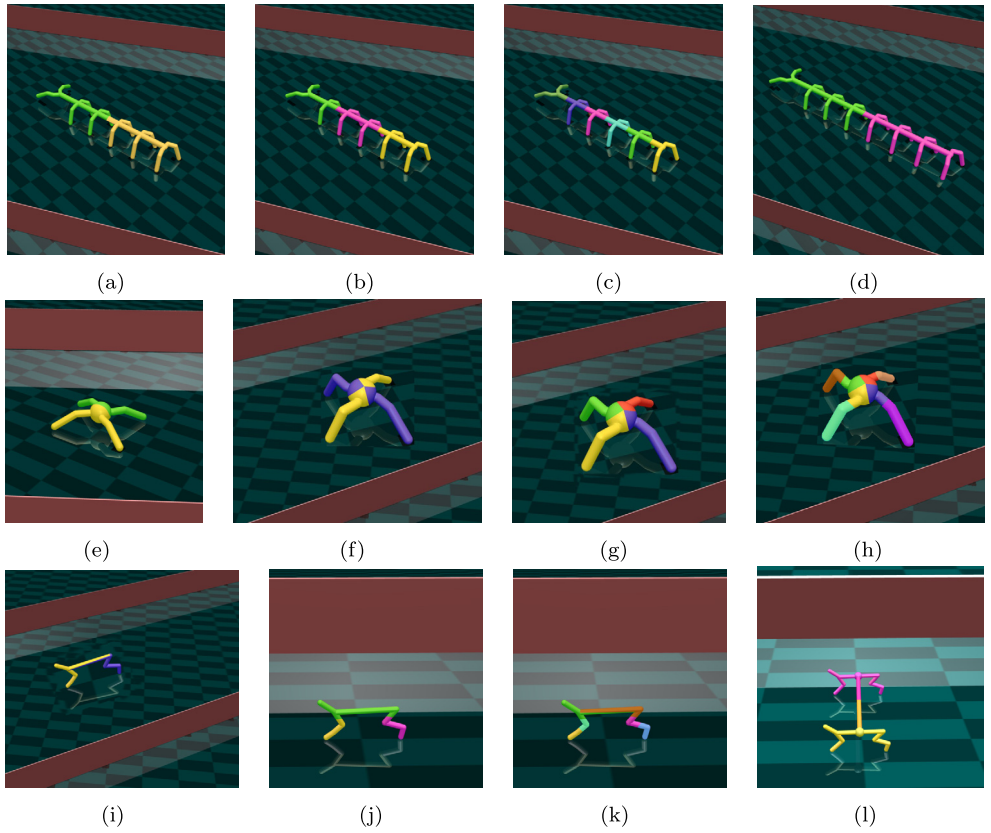


Fig. 1. Example tasks in Safe Multi-Agent MuJoCo. (a)-(d): Safe ManyAgent Ant, (e)-(h): Safe Ant, (i)-(l): Safe HalfCheetah. Body parts of different colours are controlled by different agents. Agents jointly learn to manipulate the robot, while avoiding crashing into unsafe red areas. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

$$c_t = \begin{cases} 0, & \text{for } 0.2 \leq \boldsymbol{z}_{\text{torso},t+1} \leq 1.0 \text{ and } z_{rot} > -0.7 \\ & \quad \text{and } \left\| \boldsymbol{x}_{\text{torso},t+1} - \boldsymbol{x}_{\text{wall}} \right\|_2 \geq 1.8, \\ 1, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{z}_{\text{torso},t+1}$ is the robot's torso's z-coordinate, z_{rot} is the robot's rotation's z-coordinate, and $\boldsymbol{x}_{\text{torso},t+1}$ is the robot's torso's x-coordinate, at time $t+1$; $\boldsymbol{x}_{\text{wall}}$ is the x-coordinate of the wall. A similar cost scheme is implemented in *ManyAgent Ant Tasks 2.1 & 2.2* and in *Ant Tasks 2.1 & 2.2*.

HalfCheetah task 1.0 & couple HalfCheetah task 1.0

In these tasks, the HalfCheetah agents move inside a corridor (which constraints their movement, but does not induce costs). Together with them, there are pitfalls moving inside the corridor. If an agent finds itself too close to a bomb, the distance between an agent and a bomb is less than 9 m, a cost of 1 will be emitted, see Fig. 2 (c) and (d).

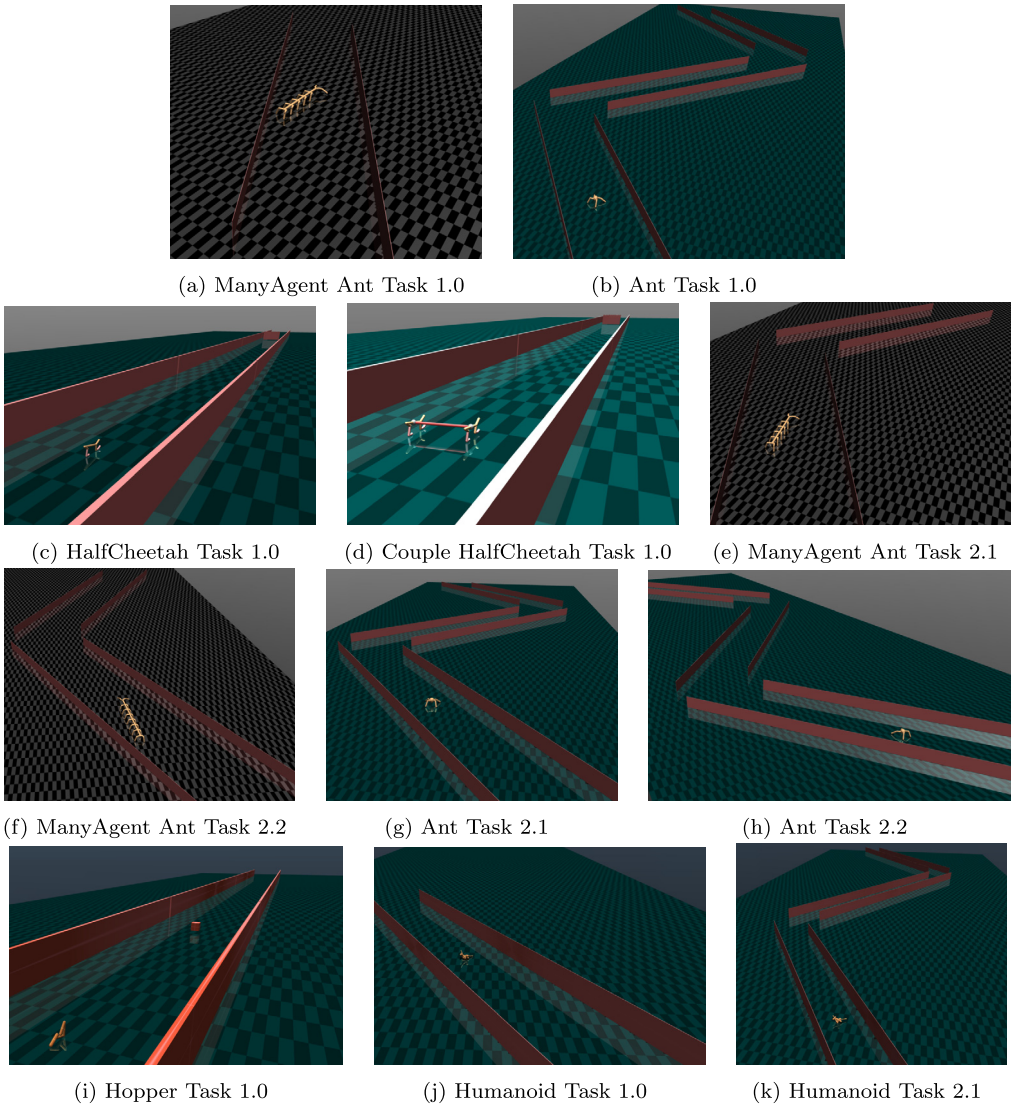
$$c_t = \begin{cases} 0, & \text{for } \left\| \boldsymbol{y}_{\text{torso},t+1} - \boldsymbol{y}_{\text{obstacle}} \right\|_2 \geq 9, \\ 1, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{y}_{\text{torso},t+1}$ is the y-coordinate of the robot's torso, and $\boldsymbol{y}_{\text{obstacle}}$ is the y-coordinate of the moving obstacle.

Hopper 1.0

The Hopper (3x1) robot is in a corridor, and there is a moving obstacle in the corridor (which can be regarded as a pitfall). When the robot encounters the moving obstacle, it will suffer a cost,

$$c_t = \begin{cases} 0, & \text{for } \left\| \boldsymbol{y}_{\text{torso},t+1} - \boldsymbol{y}_{\text{obstacle}} \right\|_2 \geq 1, \\ 1, & \text{otherwise.} \end{cases}$$



(a) ManyAgent Ant Task 1.0 (b) Ant Task 1.0

(c) HalfCheetah Task 1.0 (d) Couple HalfCheetah Task 1.0 (e) ManyAgent Ant Task 2.1

(f) ManyAgent Ant Task 2.2 (g) Ant Task 2.1 (h) Ant Task 2.2

(i) Hopper Task 1.0 (j) Humanoid Task 1.0 (k) Humanoid Task 2.1

Fig. 2. Specific tasks in Safe Multi-Agent MuJoCo environment. (a): ManyAgent Ant Task 1.0: ManyAgent Ant 3x2 one folding line walls (corridor width is 9 m). (b): Ant Task 1.0: Ant 4x2 with three folding Jagged (30°) line walls. (c): HalfCheetah Task 1.0: HalfCheetah 2x3 with moving obstacle, (d): Couple HalfCheetah Task 1.0: Couple HalfCheetah 1P1 with moving obstacle, (e): ManyAgent Ant Task 2.1: Many-Agent Ant 3x2 with two folding line walls (corridor width is 12 m). (f): ManyAgent Ant Task 2.2: Many-Agent Ant 4x2 with two folding line walls (corridor width is 12 m). (g): Ant Task 2.1: Ant 4x2 with three folding Jagged (40°) line walls (corridor width is 10 m). (h): Ant Task 2.2: Ant 4x2 with three folding Jagged (40°) line walls (corridor width is 8 m). (i): Hopper Task 1.0: Hopper 3x1 with moving obstacle. (j): Humanoid task 1.0: Humanoid 17x1 with one folding line walls (corridor width is 7 m). (k): Humanoid task 1.0: Humanoid 17x1 with three folding line walls (corridor width is 9 m).

Humanoid task 1.0 & Humanoid task 2.1

The Humanoid (17x1) robot run in a corridor, the width of the corridor set by two walls is 7 m (Humanoid task 1.0)/9 m (Humanoid task 2.1). The environment emits the cost of 1 for an agent, if the distance between the robot and the wall is less than 1.8 m, or when the robot topples over, this can be described as

$$c_t = \begin{cases} 0, & \text{for } 1.0 \leq \boldsymbol{z}_{\text{torso},t+1} \leq 2.0 \text{ and } z_{rot} > -0.7 \\ & \quad \text{and } \left\| \boldsymbol{x}_{\text{torso},t+1} - \boldsymbol{x}_{\text{wall}} \right\|_2 \geq 1.8, \\ 1, & \text{otherwise.} \end{cases}$$

5.1.2. Safe Multi-Agent MuJoCo: experiments

In this section, we use Safe MAMuJoCo to examine if the MACPO/MAPPO-Lagrangian agents can satisfy their safety constraints and cooperatively learn to achieve high rewards, compared to existing MARL algorithms. Notably, our proposed methods adopt two different approaches for achieving safety. MACPO reaches safety via *hard* constraints and backtracking

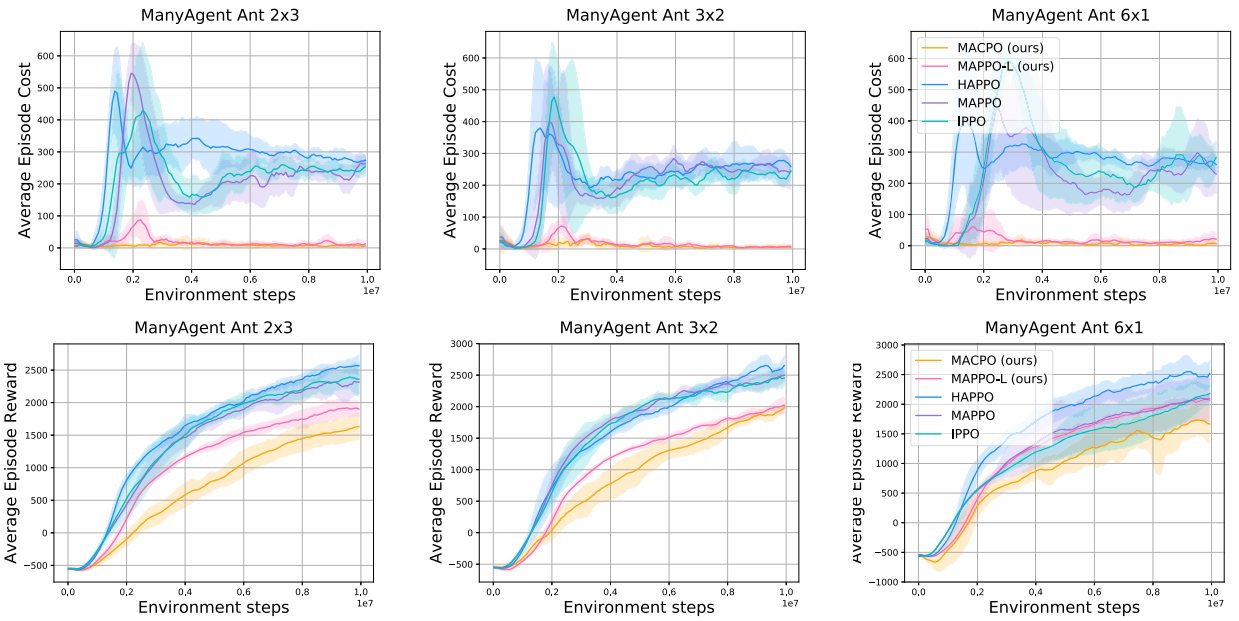


Fig. 3. Performance comparisons on Safe ManyAgent Ant task **1.0** in terms of cost and reward. The safety constraint value is 1. Each subfigure represents a different robot, within each subfigure, three task setups in terms of multi-agent control are considered, and for each task we plot the cost curves (the lower the better) in the upper row and the reward curves (the higher the better) in the bottom row.

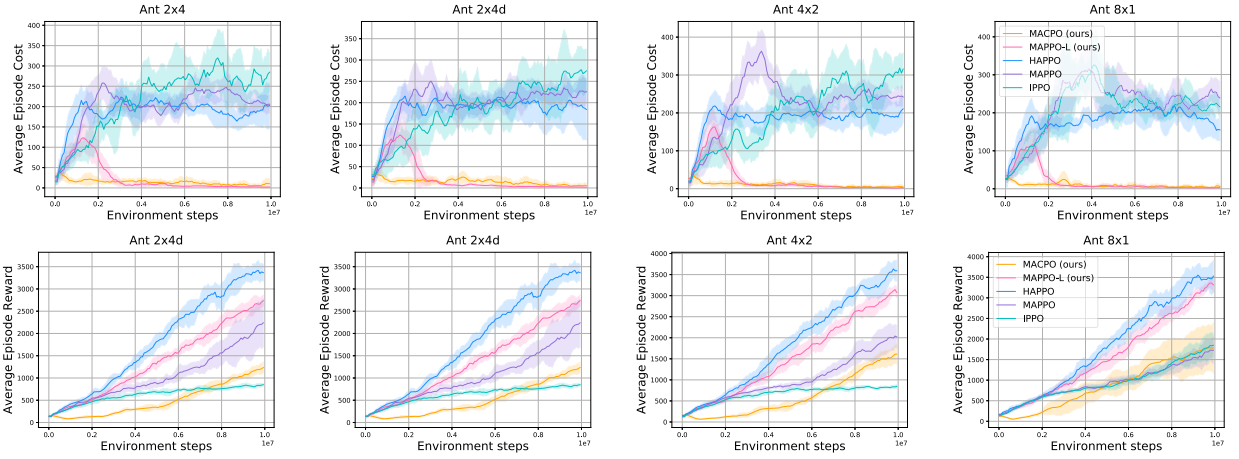


Fig. 4. Performance comparisons on Safe Ant task **1.0** in terms of cost and reward. The safety constraint values are: 0.2 for 2-agent and 4-agent Ants, and 1 for 8-agent Ants. Each subfigure represents a different robot, within each subfigure, four task setups in terms of multi-agent control are considered, and for each task we plot the cost curves (the lower the better) in the upper row and the reward curves (the higher the better) in the bottom row.

line search, while MAPPO-Lagrangian maintains a rather *soft* safety awareness by performing gradient descent-ascent on the PPO-clip objective.

Figs. 3–8 show cost and reward performance comparisons between MACPO, MAPPO-Lagrangian, MAPPO [49], IPPO [17], and HAPPO [24] algorithms on challenging Safe MAMuJoCo tasks. The experiments reveal that both MACPO and MAPPO-Lagrangian quickly learn to satisfy safety constraints, and keep their explorations within the feasible policy space. This stands in contrast to IPPO, MAPPO, and HAPPO which largely violate the constraints thus being unsafe. Furthermore, our algorithms achieve comparable reward scores; both methods are often better than IPPO. In general, the performance (in terms of reward) of MAPPO-Lagrangian is better than of MACPO; moreover, MAPPO-Lagrangian outperforms the unconstrained MAPPO on challenging *Ant* tasks (Fig. 4). We note that on none of the tasks the reward of HAPPO was exceeded, though the algorithm is unsafe.

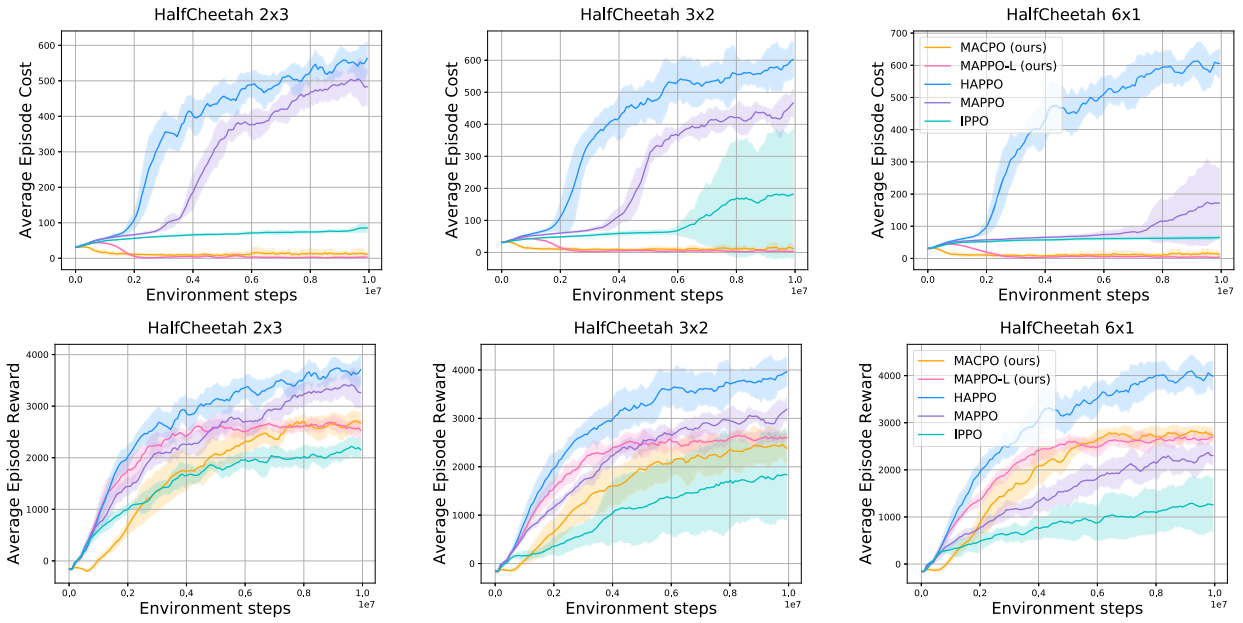


Fig. 5. Performance comparisons on tasks of Safe HalfCheetah task **1.0** in terms of cost and reward. The safety constraint value is 5. Each subfigure represents a different robot, within each subfigure, three task setups in terms of multi-agent control are considered, and for each task we plot the cost curves (the lower the better) in the upper row and the reward curves (the higher the better) in the bottom row.

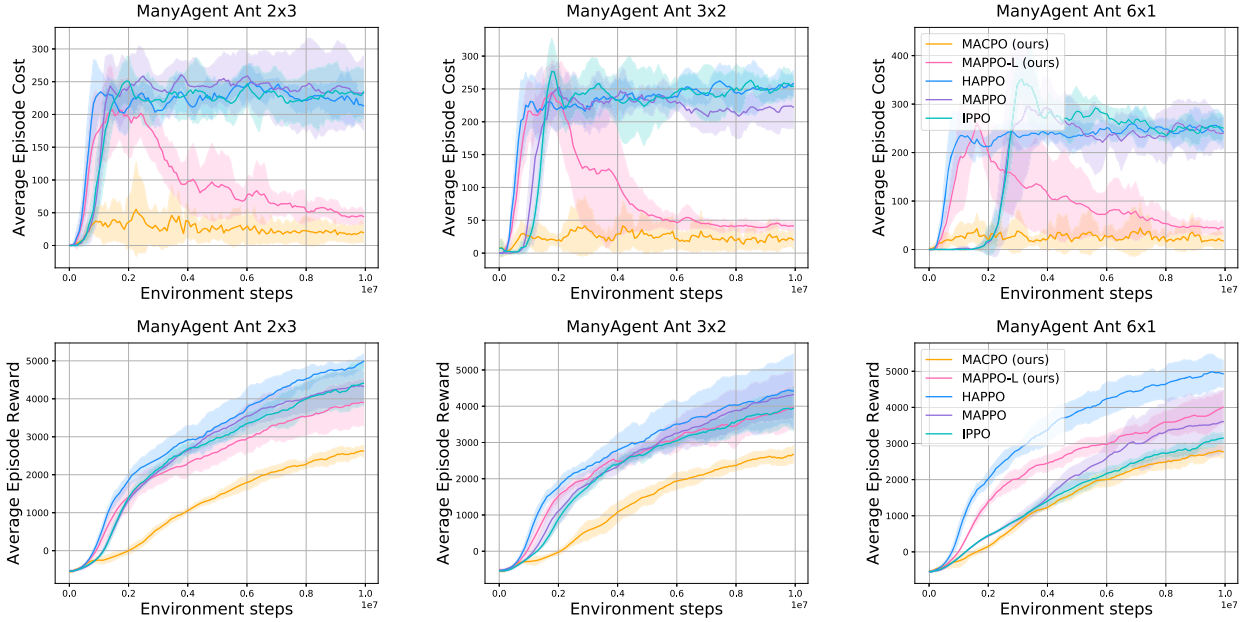


Fig. 6. Performance comparisons on Safe ManyAgent Ant task **2.1** in terms of cost and reward. The safety constraint value is set to 10. Each subfigure represents a different robot, within each subfigure, three task setups in terms of multi-agent control are considered, and for each task we plot the cost curves (the lower the better) in the upper row and the reward curves (the higher the better) in the bottom row.

5.2. Safe Multi-Agent Robosuite

In this subsection we describe the Safe Multi-Agent Robosuite environment and provide the results of our experiments conducted on it.

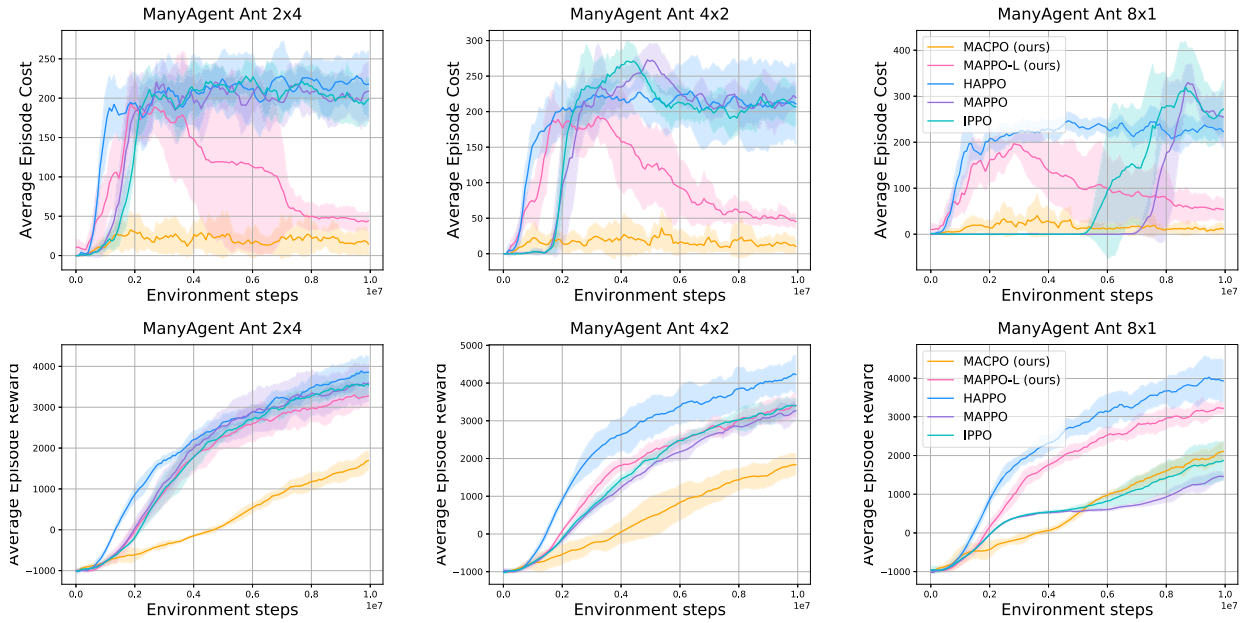


Fig. 7. Performance comparisons on Safe ManyAgent Ant task **2.2** in terms of cost and reward. The safety constraint value is set to 10. Each subfigure represents a different robot, within each subfigure, three task setups in terms of multi-agent control are considered, and for each task we plot the cost curves (the lower the better) in the upper row and the reward curves (the higher the better) in the bottom row.

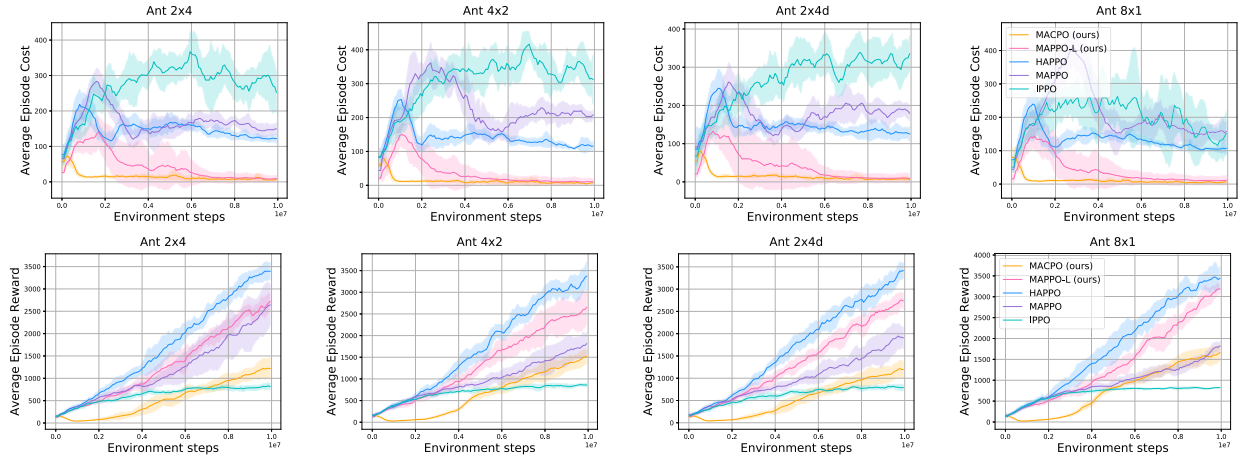


Fig. 8. Performance comparisons on Safe Ant task **2.1** in terms of cost and reward. The safety constraint value is 1. Each subfigure represents a different robot, within each subfigure, four task setups in terms of multi-agent control are considered, and for each task we plot the cost curves (the lower the better) in the upper row and the reward curves (the higher the better) in the bottom row.

5.2.1. Safe Multi-Agent Robosuite: the environment

To evaluate the effectiveness of our algorithms on robotic arms, and see how safe the multiple robots are during carrying out a task with safety constraints. We developed the benchmarks of Safe Multi-Agent Robosuite. Safe Multi-Agent Robosuite is an extension of Robosuite [53]. We split the control over a robot across multiple controllers of its joints—one or more per controller. For example, the *Lift* task comes with 3 variants: 2 four-dimensional agents (2x4 Lift), 4 two-dimensional agents (4x2 Lift), and 8 one-dimensional agents (8x1 Lift).

Safe MARobosuite tasks are fully cooperative, partially observable, continuous, and safety-aware. Its multi-agency makes it a compatible framework for training modular robots which are built of multiple, robust parts [3]. We adopt the reward setting from Robosuite and present it below.

Lift & stack

The task is to grasp and lift a blue object from a table up to at least 0.1 m of height. At the same time, to assure safety, we require that the robotic gripper avoids physical contact with the table while ensuring joint actions' smoothness, i.e.,

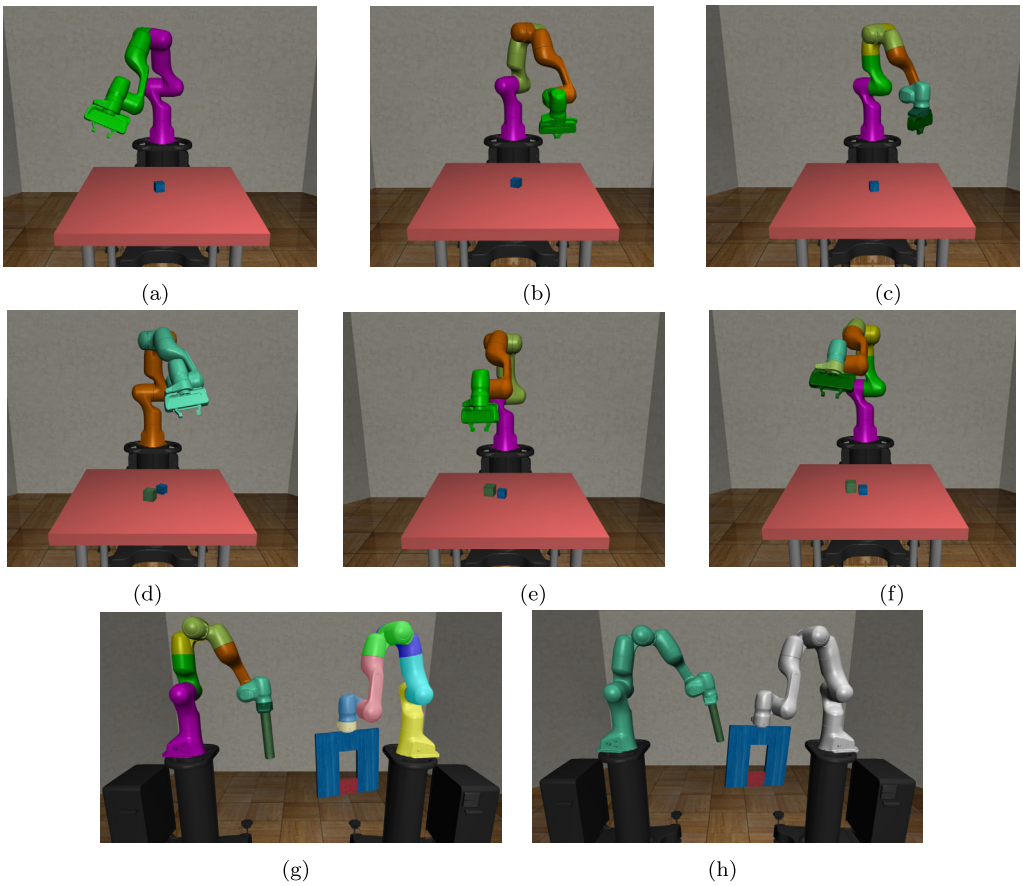


Fig. 9. Example tasks in Safe MARobosuite Environment. (a): Safe 2x4-Lift, (b): Safe 4x2-Lift, (c): Safe 8x1-Lift, (d): Safe 2x4-Stack, (e): Safe 4x2-Stack, (f): Safe 8x1-Stack, (g): Safe 14x1-TwoArmPegInHole, (h): Safe 2x7-TwoArmPegInHole. Body parts of different colours of robots are controlled by different agents. Agents jointly learn to manipulate the robot, while avoiding crashing into unsafe red areas.

keeps at least 0.02 m (without smoothness constraints, see Equation (32)) or 0.001 m (with smoothness constraints, see Equation (33) and Equation (34)) distance from it. For visualisation, see Fig. 9,

$$c_t = \begin{cases} 1.0, & \text{for} \quad |\boldsymbol{p}_{\text{eef},t+1} - \boldsymbol{z}_{\text{obstacle}}| \leq 0.02, \\ 0, & \text{otherwise}, \end{cases} \tag{32}$$

$$c_t = \begin{cases} 1.0, & \text{for} \quad |\boldsymbol{p}_{\text{eef},t+1} - \boldsymbol{z}_{\text{obstacle}}| \leq 0.001, \\ 0, & \text{otherwise}, \end{cases} \tag{33}$$

$$c'_t = \begin{cases} 1.0, & \text{for} \quad |\boldsymbol{p}_{\text{i},t+1} - \boldsymbol{p}_{\text{i},t}| > 0.99, \ i \in [0, 7], \\ 0, & \text{otherwise}, \end{cases} \tag{34}$$

where $\boldsymbol{p}_{\text{eef},t+1}$ is the z-coordinate of the robot's end gripper, and $\boldsymbol{z}_{\text{obstacle}}$ is the z-coordinate of the obstacle; $\boldsymbol{p}_{\text{i},t+1}$ is the action value of the agent i at time step $t+1$.

TwoArmPegInHole

The agents learn to insert a peg into a hole of a rim. Additionally, the agents are required to keep the peg at least 0.11 m distance from a red part of the rim (see Equation (35) for cost and Fig. 9 for visualisation).

$$c_t = \begin{cases} 1.0, & \text{for} \quad \boldsymbol{p}_{\text{pd},t+1} \leq -0.11, \\ 0, & \text{otherwise}, \end{cases} \tag{35}$$

where $\boldsymbol{p}_{\text{pd},t+1}$ is the parallel distance between the robot's peg and robot's hole centre.

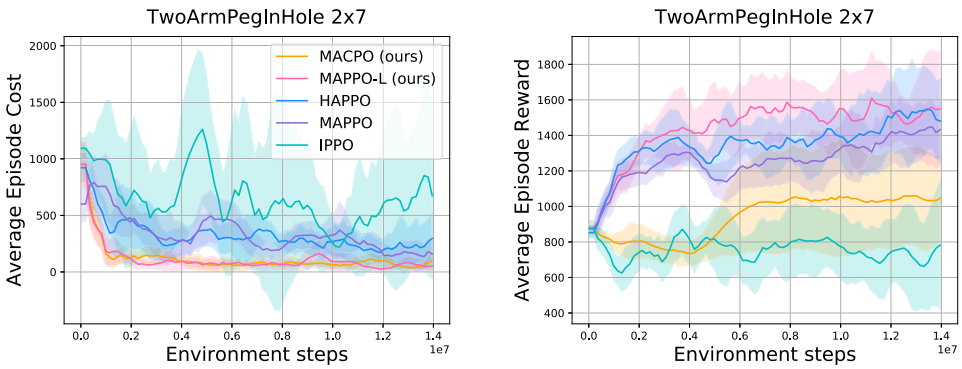


Fig. 10. Performance comparisons on Safe TwoArmPegInHole 2x7 task, with the safety constraint set to 30.

5.2.2. Safe Multi-Agent Robosuite: experiments

We study the performance of our methods in Safe TwoArmPegInHole in comparison to the aforementioned baselines. Notably, from Fig. 10 we learn that IPPO completely fails to increase reward in TwoArmPegInHole 2x7, which suggests that the environment's difficulty is higher than that of Safe MAMuJoCo.

5.3. Safe Multi-Agent Isaac Gym

Safe Multi-Agent Isaac Gym (Safe MAIG) is developed on top of Isaac Gym [31]—a high performance, GPU-based platform for robotics tasks that utilises the Nvidia PhysX [33] engine. Again, we employ this environment to compare our methods to other MARL algorithms.

5.3.1. Safe Multi-Agent Isaac Gym: the environment

To examine our algorithms on hands' tasks regarding reward and safety performance, we proposed the benchmark of Safe Multi-Agent Isaac Gym. Specifically, we describe our tasks: ShadowHandOver (2x6, 6x2, 4x3, 12x1), ShadowHandCatchUnderarm (2x6, 6x2, 12x1), ShadowHandReOrientation (2x6, 6x2, 12x1), ShadowHandTwoCatchUnderarm (2x6, 6x2, 4x3, 12x1), ShadowHandCatchAbreast (2x6, 6x2, 4x3, 12x1), and ShadowHandOver2Underarm (2x6, 6x2, 4x3, 12x1), in detail and visualise them in Fig. 11.

ShadowHandOver (2x6, 6x2, 4x3, 12x1)

The environment involves two hands at fixed positions. The first hand with an object must find a way to hand the item over to the second hand, and the second hand must learn how to grasp the item that is from the first hand, while one finger on the first hand has safety constraints (see Fig. 11 (a)) over the range of motion of one of the fingers. Formally, the cost is given by

$$c_t = \begin{cases} 1.0, & \text{for} \quad |\boldsymbol{F}_{\text{a4},t+1}| \geq 0.1, \\ 0, & \text{otherwise}, \end{cases}$$

where $\boldsymbol{F}_{\text{a4},t+1}$ is the first hand's fourth fingers's motion degree.

ShadowHandCatchUnderarm (2x6, 6x2, 4x3, 12x1)

This task is an extension of ShadowHandOver in which the hands can move freely, i.e., they can translate/rotate their centres of masses within some region (see Fig. 11 (b)). The safety constraints are as in the previous task.

ShadowHandReOrientation (2x6, 6x2, 4x3, 12x1)

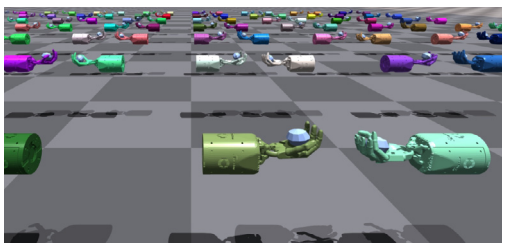
In each of two hands there are two items. The goal of the agents is to rotate the two items around each other (see Fig. 11 (c)). The safety constraints remain the same.

ShadowHandTwoCatchUnderarm (2x6, 6x2, 4x3, 12x1)

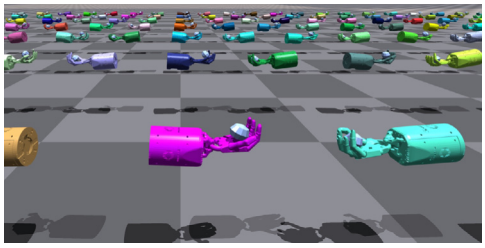
In each of two hands there is a item, and the item in the right hand will be handed to the left hand, and the left hand will hold two items well. The safety constraints are as in the previous task (see Fig. 11 (d)).

ShadowHandCatchAbreast (2x6, 6x2, 4x3, 12x1)

This task is similar to ShadowHandCatchUnderarm, nonetheless, the two hands are placed side by side instead of in a position that is identical to ShadowHandCatchUnderarm (see Fig. 11 (e)). The safety constraints remain the same.



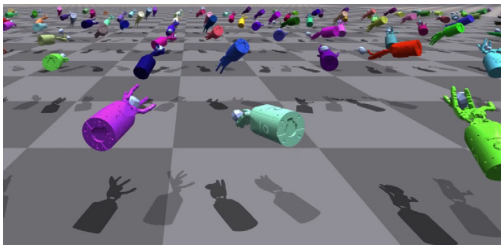
(a) ShadowHandOver 2x6



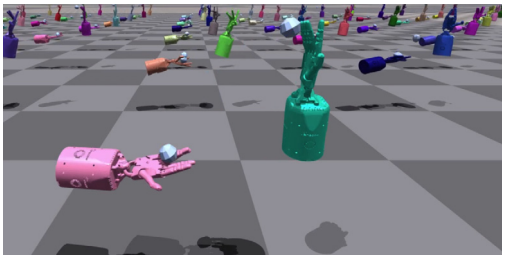
(b) ShadowHandCatchUnderarm 2x6

(c) ShadowHandReOrientation 4x3

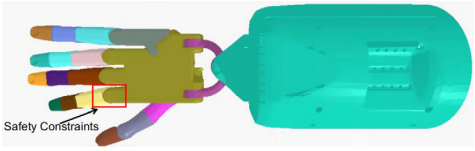
(d) ShadowHandTwoCatchUnderarm 2x3



(e) ShadowHandCatchAbreast 2x6



(f) ShadowHandOver2Underarm 2x6



(g) A hand showing safety constraints

Fig. 11. Safe multi-agent Isaac Gym environments. Body parts of robots are controlled by different agents in each pair of hands. Agents jointly learn to manipulate the robot, while avoiding violating the safety constraints.

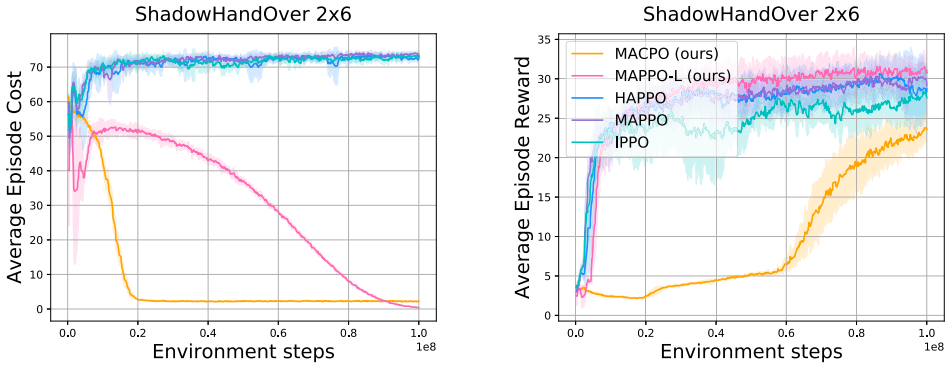


Fig. 12. ShadowHandOver 2x6 task: performance comparisons Safe ShadowHandOver 2x6 task in terms of cost and reward, the safety constraint value is set to 2.5. Our algorithms are the only ones that learn the safety constraints, while achieving satisfying performance in terms of the reward.

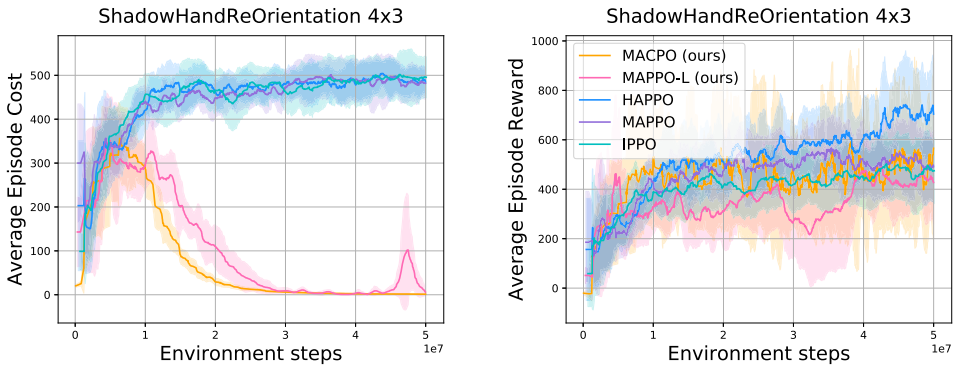


Fig. 13. ShadowHandReOrientation 4x3 task: performance comparisons on Safe ShadowHandReOrientation 4x3 task in terms of cost and reward. The safety constraint value is set to 0.1.

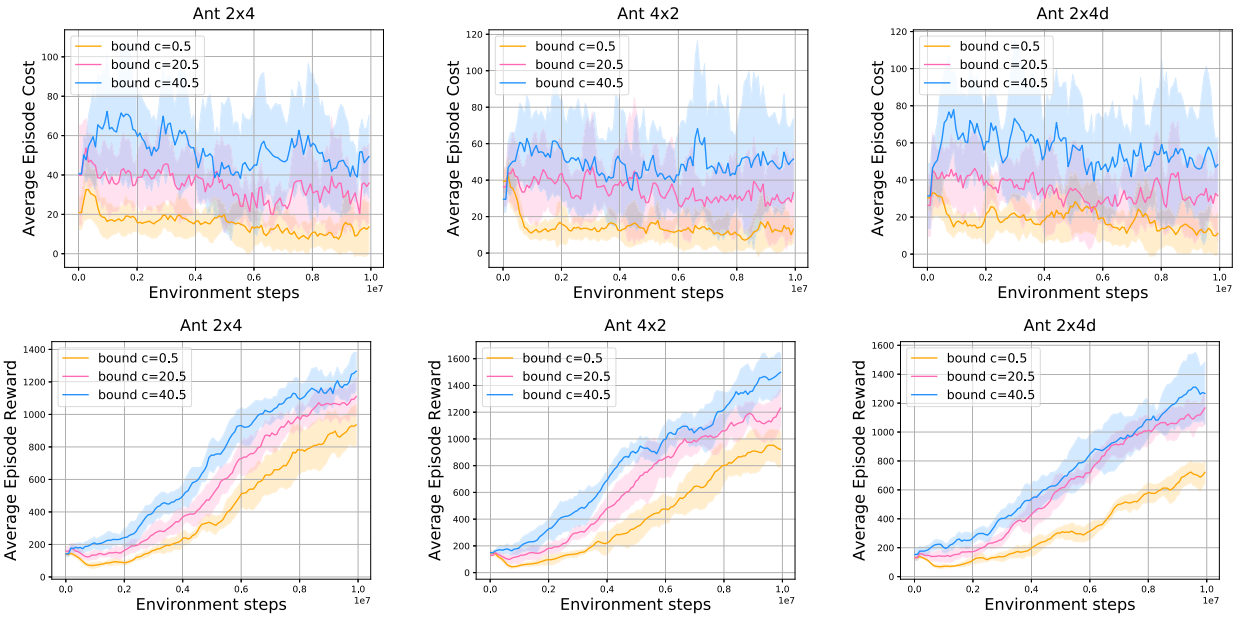


Fig. 14. Ant 2x4, 4x2, 2x4d tasks **2.2**, MACPO with different safety bound in terms of costs and rewards.

ShadowHandOver2Underarm (2x6, 6x2, 4x3, 12x1)

Two hands are present in the environment in free postures, where they will grip the object cooperatively. The first hand holding the object must figure out a way to transfer it to the second hand, and the second hand must learn how to grip the object that came from the first hand. The object is thrown from the vertical hand to the palm-up hand in this scenario. The environment consists of the half ShadowHandCatchUnderarm and the half ShadowHandCatchOverarm (see Fig. 11 (f)). The safety constraints remain the same.

5.3.2. Safe Multi-Agent Isaac Gym: experiments

Again, the study reveals that the performance of our methods in terms of reward is competitive with the SOTA MARL methods, meanwhile being safe. Furthermore, as Figs. 12-13 show, MACPO learns to meet the safety constraints faster than MAPPO-Lagrangian. The latter algorithm, however, performs better in terms of reward.

Moreover, in Fig. 13 left, there is a peak at the end of the MAPPO-L curve. Since MAPPO-L is a soft constraint algorithm, when a safety violation happens, there is a delay to ensure safety, unlike MACPO, which can ensure hard constraints to achieve safety; MAPPO-L is also sensitive to the learning rate. Although there are some disadvantages to the type of MAPPO-Lagrangian methods, currently, it is a popular and efficient algorithm in the safe RL community [22].

5.4. Sensitivity analysis for safety-bound-parameters tuning

We analyse the sensitivity of MACPO to changes in the size of safety bound (*bound c*) on Fig. 14. We learn that with that the algorithm's *effectiveness* does not change under different safety levels. However, with the stricter safety constraints, the reward performance of MACPO decreases.

6. Conclusion

In this paper, we have made a step towards solving multi-robot control problems with safety constraints via safe multi-agent reinforcement learning. We introduced the first general formulation of the safe MARL problem and analysed it theoretically. Central to our findings is a safe multi-agent policy iteration procedure that attains theoretically-justified monotonic improvement and constraints satisfaction guarantee at every iteration. As further outcomes to our analysis, we proposed two practical algorithms: MACPO and MAPPO-Lagrangian. To demonstrate their effectiveness, as well as to increase the range of testbeds for Safe MARL methods, we introduced three new benchmark suites of Safe MAMuJoCo, Safe MARobosuite and Safe MAIG. We used them to compare our methods against strong MARL baselines, over which they showed a significant advantage in terms of safety, and comparable performance in terms of the reward. We believe that our findings will contribute to the rise and applicability of many safe MARL methods for multi-robot control.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was partially supported by the Beijing Municipal Science & Technology Commission No. Z221100003422004, European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 945539 (Human Brain Project SGA3). We would like to thank Munning Wen, Ruiqing Chen, Ziyan Wang, and Hengrui Zhang for their help to run some of experiments, we also thank Zheng Tian and Jun Wang for their useful project discussions and valuable suggestions. We are deeply grateful to all reviewers for their constructive comments.

Appendix A. Details of settings for experiments

In this section, we provide the details of settings for our experiments (See Tables A.1–A.7.). The code is available at: https://sites.google.com/view/aij-safe-marl/.

Table A.1
In the Safe MAMuJoCo, Safe MARobosuite and Safe MAIG domains, common hyperparameters used for MAPPO-Lagrangian, MAPPO, HAPPO, IPPO, and MACPO.

hyperparameters	value	hyperparameters	value
gain	0.01	optimiser	Adam
activation	ReLU	optim eps	1e-5
std x coef	1	max grad norm	10
std y coef	0.5	actor network	mlp

Table A.2
In the Safe MAMuJoCo and Safe MARobosuite domains, common hyperparameters used for MAPPO-Lagrangian, MAPPO, HAPPO, IPPO, and MACPO (episode length: 1000 used for Safe MAMuJoCo, 2000 used for Safe MARobosuite).

hyperparameters	value	hyperparameters	value
critic lr	5e-3	hidden layer	1
gamma	0.99	eval episodes	32
hidden layer dim	64	episode length	[1000, 2000]
num mini-batch	40	batch size	16000
training threads	4	rollout threads	16

Table A.3
In the Safe MAIG domains, common hyperparameters used for MAPPO-Lagrangian, MAPPO, HAPPO, IPPO, and MACPO (m denotes million, *a* indicates the parameter is used for ShadowHandOver task, *b* indicates the parameter is used for ShadowHandReOrientation task).

hyperparameters	value	hyperparameters	value
critic lr	[1e-3^a, 5e-4^b]	hidden layer	2
gamma	0.96	episode length	8
training env number	[512^a, 2048^b]	eval episodes	10000
rollout threads	80	hidden layer dim	512
num mini-batch	1	num env steps	[100 m^a, 50 m^b]

Table A.4
In the Safe MAMuJoCo, Safe MARobosuite and Safe MAIG domains, different hyperparameters used for MAPPO-Lagrangian, MAPPO, HAPPO, IPPO, and MACPO.

Algorithms	MAPPO-L	MAPPO	HAPPO	IPPO	MACPO
ppo epoch	5	5	5	5	/
ppo-clip	0.2	0.2	0.2	0.2	/
fraction	/	/	/	/	0.5

Table A.5
In the Safe MAMuJoCo and Safe MARobosuite domains, different hyperparameters used for MAPPO-Lagrangian, MAPPO, HAPPO, IPPO, and MACPO (kl-threshold: 0.0065 used for Safe MAMuJoCo, 1e-3 used for Safe MARobosuite).

Algorithms	MAPPO-L	MAPPO	HAPPO	IPPO	MACPO
actor lr	9e-5	9e-5	9e-5	9e-5	/
kl-threshold	/	/	/	/	[0.0065, 1e-3]
lambda lagr	0.78	/	/	/	/
lagrangian coef rate	1e-7	/	/	/	/
fraction coef	/	/	/	/	0.27

Table A.6
In the Safe MAIG domains, different hyperparameters used for MAPPO-Lagrangian, MAPPO, HAPPO, IPPO, and MACPO, *a* indicates the parameter is used for ShadowHandOver task, *b* indicates the parameter is used for ShadowHandReOrientation task.

Parameters	actor lr	kl-threshold	fraction coef
MAPPO-L	[3e-5^a, 5e-4^b]	/	/
MAPPO	[3e-5^a, 5e-4^b]	/	/
HAPPO	[3e-5^a, 5e-4^b]	/	/
IPPO	[3e-5^a, 5e-4^b]	/	/
MACPO	/	[67e-4^a, 65e-4^b]	[0.29^a, 0.27^b]

Table A.7
Safety bound in the Safe MAMuJoCo, Safe MARobosuite Safe MAIG domains.

task	value	task	value
Ant(2x4) 1.0	0.2	Ant(4x2) 1.0	0.2
Ant(2x4d) 1.0	0.2	HalfCheetah(2x3) 1.0	5
HalfCheetah(3x2) 1.0	5	HalfCheetah(6x1) 1.0	5
ManyAgent Ant(2x3) 1.0	1	ManyAgent Ant(3x2) 1.0	1
ManyAgent Ant(6x1) 1.0	1	TWoArmPegInHole(2x7) 1.0	30
ManyAgent Ant(2x3) 2.1	10	ManyAgent Ant(3x2) 2.1	10
ManyAgent Ant(6x1) 2.1	10	ManyAgent Ant(2x3) 2.2	10
ManyAgent Ant(3x2) 2.2	10	ManyAgent Ant(6x1) 2.2	10
Ant(2x4) 2.1	1	Ant(4x2) 2.1	1
Ant(2x4d) 2.1	1	Ant(8x1) 2.1	1
ShadowHandOver(2x6)	2.5	ShadowHandReOrientation(4x3)	0.1

Appendix B. Solution to the constrained optimisation problem in MACPO

Theorem 2. *The solution to the following problem*

$$\begin{aligned} \boldsymbol{p}* &= \min_{x} \boldsymbol{g}^T \boldsymbol{x} \\ \text{s.t. } & \boldsymbol{b}^T \boldsymbol{x} + c \leq 0 \\ & \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} \leq \delta, \end{aligned}$$

where $\boldsymbol{g}, \boldsymbol{b}, \boldsymbol{x} \in \mathbb{R}^n, c, \delta \in \mathbb{R}, \delta > 0$, $\boldsymbol{H} \in \mathbb{S}^n$, *and* $\boldsymbol{H} \succ 0$. *When there is at least one strictly feasible point, the optimal point* $\boldsymbol{x}^*$ *satisfies:*

$$\boldsymbol{x}^* = -\frac{1}{\lambda_*} \boldsymbol{H}^{-1} \left(\boldsymbol{g}^T + \nu_* \boldsymbol{b} \right)$$

where λ_* *and* ν_* *are defined by*

$$\begin{aligned} \nu_* &= \left(\frac{\lambda_* c - r}{s} \right)_+ \\ \lambda_* &= \arg\max_{\lambda \geq 0} \begin{cases} f_a(\lambda) \triangleq \frac{1}{2\lambda} \left(\frac{r^2}{s} - q \right) + \frac{\lambda}{2} \left(\frac{c^2}{s} - \delta \right) - \frac{rc}{s} & \text{if } \lambda c - r > 0 \\ f_b(\lambda) \triangleq -\frac{1}{2} \left(\frac{q}{\lambda} + \lambda \delta \right) & \text{otherwise} \end{cases} \end{aligned}$$

where $\boldsymbol{q} = \boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g}$, $\boldsymbol{r} = \boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{b}$, *and* $\boldsymbol{s} = \boldsymbol{b}^T \boldsymbol{H}^{-1} \boldsymbol{b}$.

Furthermore, let $\Lambda_a \triangleq \{\lambda \mid \lambda c - r > 0, \lambda \geq 0\}$, *and* $\Lambda_b \triangleq \{\lambda \mid \lambda c - r \leq 0, \lambda \geq 0\}$. *The value of* λ_* *satisfies*

$$\lambda_* \in \left\{ \lambda_*^a \triangleq \text{Proj}\left(\sqrt{\frac{q - r^2/s}{\delta - c^2/s}}, \Lambda_a \right), \lambda_*^b \triangleq \text{Proj}\left(\sqrt{\frac{q}{\delta}}, \Lambda_b \right) \right\} \tag{B.1}$$

where $\lambda_* = \lambda_*^a$ *if* $f_a\left(\lambda_*^a\right) > f_b\left(\lambda_*^b\right)$ *and* $\lambda_* = \lambda_*$ *otherwise, and* $\text{Proj}(a, \boldsymbol{S})$ *is the projection of a point* x *on to a set* $\boldsymbol{S}$. *Note the projection of a point* $x \in \mathbb{R}$ *onto a convex segment of* $\mathbb{R}$, $[a, b]$, *has value* $\text{Proj}(x, [a, b]) = \max(a, \min(b, x))$.

Proof. See [2] (Appendix 10.2). □

References

[1] Naoki Abe, Prem Melville, Cezar Pendus, Chandan K. Reddy, David L. Jensen, Vince P. Thomas, James J. Bennett, Gary F. Anderson, Brent R. Cooley, Melissa Kowalczyk, et al., Optimizing debt collections using constrained reinforcement learning, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 75–84.

[2] Joshua Achiam, David Held, Aviv Tamar, Pieter Abbeel, Constrained policy optimization, in: International Conference on Machine Learning, PMLR, 2017, pp. 22–31.

[3] Matthias Althoff, Andrea Giusti, Stefan B. Liu, Aaron Pereira, Effortless creation of safe robots from modules through self-programming and self-verification, Sci. Robot. 4 (31) (2019).

[4] Eitan Altman, Constrained Markov Decision Processes, vol. 7, CRC Press, 1999.

[5] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, Paulo Tabuada, Control barrier function based quadratic programs for safety critical systems, IEEE Trans. Autom. Control 62 (8) (2016) 3861–3876.

[6] Thomas Beckers, Leonardo J. Colombo, Sandra Hirche, George J. Pappas, Online learning-based trajectory tracking for underactuated vehicles with uncertain dynamics, IEEE Control Syst. Lett. 6 (2021) 2090–2095.

[7] Urs Borrmann, Li Wang, Aaron D. Ames, Magnus Egerstedt, Control barrier certificates for safe swarm behavior, IFAC-PapersOnLine 48 (27) (2015) 68–73.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, Wojciech Zaremba, Openai gym, 2016.

[9] Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, Angela P. Schoellig, Safe learning in robotics: from learning-based control to safe reinforcement learning, Annu. Rev. Control Robotics Auton. Syst. 5 (2021).

[10] Hao Chen, Jie Qi, Yuqin Dong, Sanguan Zhong, Multi-robot formation control and implementation, in: 2021 40th Chinese Control Conference (CCC), IEEE, 2021, pp. 879–884.

[11] Sandeep Chinchali, Pan Hu, Tianshu Chu, Manu Sharma, Manu Bansal, Rakesh Misra, Marco Pavone, Sachin Katti, Cellular network traffic scheduling with deep reinforcement learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[12] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, Marco Pavone, Risk-constrained reinforcement learning with percentile risk criteria, J. Mach. Learn. Res. 18 (1) (2017) 6070–6120.

[13] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, Mohammad Ghavamzadeh, A Lyapunov-based approach to safe reinforcement learning, Adv. Neural Inf. Process. Syst. 31 (2018).

[14] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, Mohammad Ghavamzadeh, Lyapunov-based safe policy optimization for continuous control, arXiv preprint, arXiv:1901.10031, 2019.

[15] Tianshu Chu, Jie Wang, Lara Codecà, Zhaojian Li, Multi-agent deep reinforcement learning for large-scale traffic signal control, IEEE Trans. Intell. Transp. Syst. 21 (3) (2019) 1086–1095.

[16] Agostino De Santis, Bruno Siciliano, Alessandro De Luca, Antonio Bicchi, An atlas of physical human–robot interaction, Mech. Mach. Theory 43 (3) (2008) 253–270.

[17] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H.S. Torr, Mingfei Sun, Shimon Whiteson, Is independent learning all you need in the starcraft multi-agent challenge?, arXiv preprint, arXiv:2011.09533, 2020.
[18] Rafael Fierro, Aveek Das, John Spletzer, Joel Esposito, Vijay Kumar, James P. Ostrowski, George Pappas, Camillo J. Taylor, Yerang Hur, Rajeev Alur, et al., A framework and architecture for multi-robot coordination, Int. J. Robot. Res. 21 (10–11) (2002) 977–995.
[19] Javier Garcıa, Fernando Fernández, A comprehensive survey on safe reinforcement learning, J. Mach. Learn. Res. 16 (1) (2015) 1437–1480.
[20] Shangding Gu, Guang Chen, Lijun Zhang, Jing Hou, Yingbai Hu, Alois Knoll, Constrained reinforcement learning for vehicle motion planning with topological reachability analysis, Robotics 11 (4) (2022) 81.
[21] Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, Yaodong Yang, Multi-agent constrained policy optimisation, arXiv preprint, arXiv:2110.02793, 2021.
[22] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, Alois Knoll, A review of safe reinforcement learning: methods, theory and applications, arXiv preprint, arXiv:2205.10330, 2022.
[23] Zhong-Sheng Hou, Zhuo Wang, From model-based control to data-driven control: survey, classification and perspective, Inf. Sci. 235 (2013) 3–35.
[24] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, Yaodong Yang, Trust region policy optimisation in multi-agent reinforcement learning, in: International Conference on Learning Representations, 2021.
[25] Jakub Grudzien Kuba, Muning Wen, Yaodong Yang, Linghui Meng, Shangding Gu, Haifeng Zhang, David Henry Mguni, Jun Wang, Settling the variance of multi-agent policy gradients, Adv. Neural Inf. Process. Syst. 34 (2021) 13458–13470.
[26] Tor Lattimore, Csaba Szepesvári, Bandit Algorithms, Cambridge University Press, 2020.
[27] Timothy Paul Lillicrap, Jonathan James Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, Daniel Pieter Wierstra, Continuous control with deep reinforcement learning, September 15 2020. US Patent 10,776,692.
[28] Chenyi Liu, Nan Geng, Vaneet Aggarwal, Tian Lan, Yuan Yang, Mingwei Xu, Cmix: deep multi-agent reinforcement learning with peak and average constraints, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2021, pp. 157–173.
[29] Songtao Lu, Kaiqing Zhang, Tianyi Chen, Tamer Basar, Lior Horesh, Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 8767–8775.
[30] Xiaobai Ma, Jiachen Li, Mykel J. Kochenderfer, David Isele, Kikuo Fujimura, Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 6064–6071.
[31] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al., Isaac gym: high performance gpu based physics simulation for robot learning, in: Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021.
[32] Teodor Mihai Moldovan, Pieter Abbeel, Safe exploration in Markov decision processes, in: Proceedings of the 29th International Conference on International Conference on Machine Learning, 2012, pp. 1451–1458.
[33] NVIDIA, Nvidia physx, https://developer.nvidia.com/physx-sdk, 2020.
[34] Dimitra Panagou, Dušan M. Stipanović, Petros G. Voulgaris, Distributed coordination control for multi-robot networks using Lyapunov-like barrier functions, IEEE Trans. Autom. Control 61 (3) (2015) 617–632.
[35] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, Shimon Whiteson Facmac, Factored multi-agent centralised policy gradients, Adv. Neural Inf. Process. Syst. 34 (2021) 12208–12221.
[36] David Pollard, Asymptopia: an exposition of statistical asymptotic theory, http://www.stat.yale.edu/pollard/Books/Asymptopia, 2000.
[37] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, Chuchu Fan, Learning safe multi-agent control with decentralized neural barrier certificates, in: International Conference on Learning Representations, 2020.
[38] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, Shimon Whiteson, Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 4295–4304.
[39] Alex Ray, Joshua Achiam, Dario Amodei, Benchmarking safe exploration in deep reinforcement learning, arXiv preprint, arXiv:1910.01708, 2019, 7.
[40] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G.J. Rudner, Chia-Man Hung, Philip H.S. Torr, Jakob Foerster, Shimon Whiteson, The starcraft multi-agent challenge, in: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 2186–2188.
[41] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, Philipp Moritz, Trust region policy optimization, in: International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.
[42] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, Pieter Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv preprint, arXiv:1506.02438, 2015.
[43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, Proximal policy optimization algorithms, arXiv preprint, arXiv:1707.06347, 2017.
[44] Shai Shalev-Shwartz, Shaked Shammah, Amnon Shashua, Safe, multi-agent, reinforcement learning for autonomous driving, arXiv preprint, arXiv:1610.03295, 2016.
[45] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[46] Richard S. Sutton, Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[47] Long Yang, Jiaming Ji, Juntao Dai, Linrui Zhang, Binbin Zhou, Pengfei Li, Yaodong Yang, Gang Pan, Constrained update projection approach to safe policy optimization, in: Advances in Neural Information Processing Systems (NeurIPS), 2022.
[48] Long Yang, Yu Zhang, Gang Zheng, Qian Zheng, Pengfei Li, Jianhang Huang, Gang Pan, Policy optimization with stochastic mirror descent, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 8823–8831.
[49] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, Yi Wu, The surprising effectiveness of mappo in cooperative, multi-agent games, arXiv preprint, arXiv:2103.01955, 2021.
[50] Moritz A. Zanger, Karam Daaboul, J. Marius Zöllner, Safe continuous control with constrained model-based policy optimization, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 3512–3519.
[51] Wenbo Zhang, Osbert Bastani, Vijay Kumar, Mamps: safe multi-agent reinforcement learning via model predictive shielding, arXiv preprint, arXiv:1910.12639, 2019.
[52] Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Hui Liu, Jiliang Tang, Dear: deep reinforcement learning for online advertising impression in recommender systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 750–758.
[53] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, robosuite: a modular simulation framework and benchmark for robot learning, arXiv preprint, arXiv:2009.12293, 2020.