DeCOM: Decomposed Policy for Constrained Cooperative Multi-Agent Reinforcement Learning

Zhaoxing Yang, Haiming Jin*, Rong Ding, Haoyi You, Guiyun Fan, Xinbing Wang, Chenghu Zhou

Shanghai Jiao Tong University, Shanghai, China
{yiannis, jinhaiming, dingrong, yuri-you, fgy726, xwang8}@sjtu.edu.cn, zhouchsjtu@gmail.com

Abstract

In recent years, multi-agent reinforcement learning (MARL) has presented impressive performance in various applications. However, physical limitations, budget restrictions, and many other factors usually impose *constraints* on a multi-agent system (MAS), which cannot be handled by traditional MARL frameworks. Specifically, this paper focuses on constrained MAS where agents work *cooperatively* to maximize the expected team-average return under various constraints on expected team-average costs, and develops a *constrained cooperative MARL* framework, named DeCOM, for such MAS. In particular, DeCOM decomposes the policy of each agent into two modules, which empowers information sharing among agents to achieve better cooperation. In addition, with such modularization, the training algorithm of DeCOM separates the original constrained optimization into an unconstrained optimization on reward and a constraints satisfaction problem on costs. DeCOM then iteratively solves these problems in a computationally efficient manner, which makes DeCOM highly scalable. We also provide theoretical guarantees on the convergence of DeCOM's policy update algorithm. Finally, we conduct extensive experiments to show the effectiveness of DeCOM with various types of costs in both moderate-scale and large-scale (with 500 agents) environments that originate from real-world applications.

Introduction

Recent years have seen great success of *multi-agent reinforcement learning (MARL)* in unconstrained *multi-agent systems (MAS)*, such as video games (Hughes et al. 2018; Jaques et al. 2019; Baker et al. 2020), and many others. However, in practice, a MAS usually works under various constraints introduced by physical limitations, budget restrictions, as well as requirements on certain performance metrics. For example, to avoid collisions, robot swarms have to keep distances from obstacles and between each other above a threshold (Luo, Sun, and Kapoor 2020). As another example, the fairness of power consumption among sensors has to be maintained above a certain level for the sustainability of distributed sensor networks (Xu, Zhong, and Wang 2020).

In practice, many constrained MAS are *cooperative* in nature, where agents cooperatively maximize the team-average return under constraints on certain types of team-average costs. Such cooperation exists in the above robot swarms and sensor networks, as well as other real-world scenarios, such as managing a fleet of ridesharing vehicles (Lin et al. 2018; Wang et al. 2022) where the unfairness among drivers' incomes has to be upper bounded for sufficient driver satisfaction. Inevitably, such joint requirement of cooperation and constraint satisfaction calls for new decision-making mechanisms. Therefore, in this paper, we aim to *develop a MARL framework specifically tailored for constrained cooperative MAS*.

One intuitive solution is to directly extend existing single-agent constrained reinforcement learning (Achiam et al. 2017; Tessler, Mankowitz, and Mannor 2019; Chow et al. 2019; Le, Voloshin, and Yue 2019) to our multi-agent setting, by utilizing a centralized controller to compute the joint actions of all agents. However, it is hard to scale such approach to MAS with a large number of agents. Instead, we achieve scalability by adopting the *centralized training decentralized execution* framework (Foerster et al. 2016a), where each agent is equipped with a local policy that makes decisions without the coordination from any central controller.

However, it is usually challenging for decentralized decision making to achieve cooperation. To address this challenge, we propose a novel constrained cooperative MARL framework, named DeCOM[1], which facilitates agent cooperation by appropriate information sharing among them. Specifically, DeCOM decomposes an agent's local policy into a *base policy* and a *perturbation policy*, where the former outputs the agent's base action and shares it with other agents, and the latter aggregates other agents' base actions to compute a perturbation. DeCOM then combines the base action and the perturbation to obtain the agent's final action. Such base action sharing mechanism provides an agent timely and necessary information about others, which helps the agent better regulate its actions for cooperation.

Furthermore, in our constrained MAS setting, agents' optimal policies correspond to the optimal solution of a constrained optimization which is intractable to solve directly. DeCOM addresses this issue by training the base policy to optimize the return and the perturbation policy to decrease the

*Corresponding author.

[1]The name DeCOM comes from Decomposed policy for Constrained cOoperative MARL.

constraints violation, respectively. Such learning framework essentially decomposes the original constrained optimization into an *unconstrained optimization* and *constraints satisfaction* problem, which is computationally efficient, easy-to-implement, and end-to-end.

Contributions. The contributions of this paper are threefold. First, to the best of our knowledge, this paper is the first that develops a constrained cooperative MARL framework. Our proposed framework DeCOM and its end-to-end training algorithm are both scalable and computationally efficient. Second, we give theoretical results which show that DeCOM's policy update algorithm is guaranteed to converge within limited number of steps under only mild assumptions. Third, in addition to three toy environments, we also conduct experiments in a large-scale environment with 500 agents based on a real-world dataset with roughly 1 million ride-hailing orders from Nov. 1 to 30, 2016, in Chengdu, China. Furthermore, the various types of costs considered in the experiments, including unsafety, unfairness, and operational costs show the potentially wide applications of DeCOM.

Constrained Cooperative Markov Game

We consider *constrained cooperative Markov game (CCMG)* in this paper, which is defined by a tuple $([N], \mathcal{S}, \{\mathcal{O}_i\}_{i=1}^N, \{\mathcal{A}_i\}_{i=1}^N, T, p, p_0, \gamma, \{r_i\}_{i=1}^N, \{c_i^1\}_{i=1}^N, \cdots, \{c_i^M\}_{i=1}^N, \{D_j\}_{j=1}^M)$. In a CCMG, $[N] = \{1, \cdots, N\}$ denotes the set of agents, and $\mathcal{S}$ denotes the global state space. Each agent i has an observation space $\mathcal{O}_i$ and action space[2] $\mathcal{A}_i$. At each global state $s \in \mathcal{S}$, each agent i only has limited observation $o_i = T(s, i) \in \mathcal{O}_i$, where $T(s, i)$ maps the global state s to agent i's observation. At each time step, each agent i chooses an action a_i from its action space $\mathcal{A}_i$. Given the joint action $\boldsymbol{a} = [a_1, \cdots, a_N]$ and the current global state s, the CCMG transits to the next global state s' with probability $p(s'|s, \boldsymbol{a})$, and each agent i receives an immediate reward $r_i(s, \boldsymbol{a})$ and M types of immediate costs, denoted as $c_i^1(s, \boldsymbol{a}), c_i^2(s, \boldsymbol{a}), \cdots, c_i^M(s, \boldsymbol{a})$. Furthermore, p_0 denotes the initial global state distribution, and constant $\gamma \in (0, 1]$ denotes the discount factor.

Each agent i selects its actions based on a local policy $\pi_i : \mathcal{O}_i \mapsto \Omega(\mathcal{A}_i)$, where $\Omega(\mathcal{A}_i)$ denotes all possible distributions over space $\mathcal{A}_i$. We denote $\boldsymbol{\pi} = [\pi_1, \cdots, \pi_N] \in \Psi$ as the joint policy of N agents, where Ψ denotes the set of all possible joint policies. Each agent i's expected long-term discounted return $J_i^R(\boldsymbol{\pi})$ and expected long-term discounted cost $J_i^{C_j}(\boldsymbol{\pi})$ for each type $j \in [M] = \{1, \cdots, M\}$ are defined in Eq. (1) and (2), respectively.

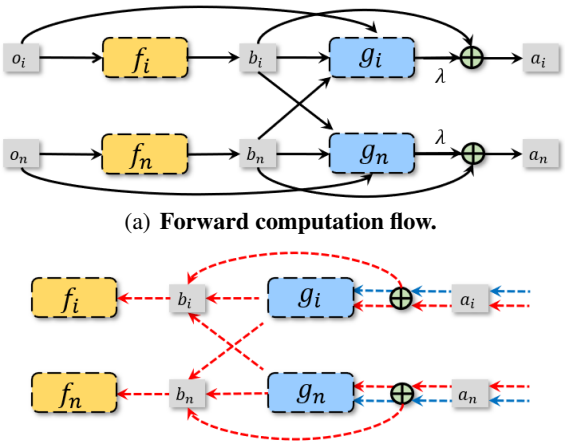
$$J_i^R(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}, p, p_0}\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, \boldsymbol{a_t})\right]. \tag{1}$$

$$J_i^{C_j}(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}, p, p_0}\left[\sum_{t=0}^{\infty} \gamma^t c_i^j(s_t, \boldsymbol{a_t})\right]. \tag{2}$$

We consider the CCMG where agents work cooperatively to maximize the expected team-average return[3] $\frac{1}{N}\sum_{i\in[N]} J_i^R(\boldsymbol{\pi})$, and to ensure that the expected team-average cost $\frac{1}{N}\sum_{i\in[N]} J_i^{C_j}(\boldsymbol{\pi})$ for each type $j \in [M]$ is upper bounded by D_j. Thus, the optimal joint policy to our CCMG is the optimal solution to the following Problem (3).

$$\begin{aligned} \max_{\boldsymbol{\pi}\in\Psi} \; & J^R(\boldsymbol{\pi}) = \frac{1}{N}\sum_{i\in[N]} J_i^R(\boldsymbol{\pi}) \\ \text{s.t. } & J^{C_j}(\boldsymbol{\pi}) = \frac{1}{N}\sum_{i\in[N]} J_i^{C_j}(\boldsymbol{\pi}) \le D_j, \quad \forall j \in [M]. \end{aligned} \tag{3}$$

To obtain the optimal policy of our CCMG is exactly the objective of this paper. However, as we consider the practical scenario where the state transition kernel is unknown *a priori*, the optimal policy could not be obtained by directly solving Problem (3). Thus, we take the approach of learning such policy via a novel framework of MARL, which will be elaborated in the following Section . Moreover, the thresholds $D_j, \forall j \in [M]$, may not be set by experts in practice, and thus, the feasibility of Problem (3) is not guaranteed. In such case, we aim to maximize the return, while minimizing the constraint violation.



(a) **Forward computation flow.**

(b) **Backward gradient flow.**

Figure 1: DeCOM framework for neighboring agents i and n, where the red dashed arrows in Fig. 1(b) indicate the gradient to maximize the expected team-average return and the blue ones indicate the gradient to minimize the constraints violation.

DeCOM Framework

As shown by Problem (3), in addition to cooperatively maximizing the expected team-average return, agents have to satisfy the constraint on each type of expected team-average cost. Such additional dimension of cooperation makes it more imperative that agents share timely and necessary information with others, so that agents could better regulate their actions based on their understandings about other agents. Thus, we propose a novel MARL framework, named DeCOM, which

[2] We consider continuous action space in this paper.

[3] Team-average reward has been widely considered in prior works (e.g., (Qu et al. 2019; Zhang et al. 2018)).

enables communication among agents by decomposing the policy of each agent into a base policy and a perturbation policy, as shown in Fig. 1(a).

More specifically, at each time step, each agent i's base policy f_i receives a local observation o_i from the environment, and outputs a base action $b_i \in \mathcal{A}_i$ which is shared with its neighbors. We define the neighbor set[4] of each agent i as the set of agents that are able to communicate with it, and denote it as $\mathcal{N}_i$. Then, each agent i's perturbation policy g_i takes as inputs its observation o_i, its own base action b_i, as well as the base actions of its neighbors $\boldsymbol{b}_{\mathcal{N}_i} = [b_j]_{j\in\mathcal{N}_i}$, and outputs a perturbation $g_i(o_i, b_i, \boldsymbol{b}_{\mathcal{N}_i})$. A scaled perturbation is then added to b_i to obtain the final action a_i. Thus,

$$a_i \;=\; b_i \;+\; \lambda\, g_i(o_i, b_i, \boldsymbol{b}_{\mathcal{N}_i}),\ \text{with}\ \ b_i \sim f_i(\cdot|o_i), \qquad (4)$$

where λ controls the magnitude of the perturbation, and $f_i(\cdot|o_i)$ is a probability distribution over the action space $\mathcal{A}_i$. Note that f_i could also be a deterministic function, which is in fact a special case of a stochastic one. In contrast, DeCOM fixes g_i as deterministic for strong representation power, as shown in the proof of Proposition 1 given in Appendix A.1.

Fig. 1(b) also shows the gradient flows in the training procedure. In DeCOM, f_i is updated by gradient ascent over $J^R(\boldsymbol{\pi})$, and thus f_i is in charge of improving the expected team-average return. In contrast, g_i receives the gradient to minimize constraints violation, making g_i undertake the duty of perturbing the base action to satisfy constraints. Such modularization essentially divides Problem (3) into an unconstrained optimization problem and a constraints satisfaction problem. This "divide and conquer" method not only enables simple end-to-end training, but also avoids the heavy computation to solve complex constrained optimization problems (Wagener, Boots, and Cheng 2021) which is inevitable in previous solution methods for constrained Markov decision process (Achiam et al. 2017; Satija, Amortila, and Pineau 2020; Yu et al. 2019; Yang et al. 2020). Moreover, as shown in Fig. 1(b), DeCOM incorporates the gradients from $\mathcal{N}_i$ to update f_i, since gradient sharing among agents could facilitate cooperation as shown by recent studies (Foerster et al. 2016b; Jiang and Lu 2018).

We next show that DeCOM's decomposed policy structure dose not reduce the representation power in Proposition 1, whose proof is given in Appendix A.1.

Proposition 1. *Let Ψ_{DeCOM} contain all possible joint policies representable by DeCOM, and $\boldsymbol{\pi}^* \in \Psi$ be the optimal solution to Problem (3). Then, for the optimal joint policy $\boldsymbol{\pi}^+ \in \Psi_{\text{DeCOM}}$, we have $J^R(\boldsymbol{\pi}^+) = J^R(\boldsymbol{\pi}^*)$ and $J^{C_j}(\boldsymbol{\pi}^+) = J^{C_j}(\boldsymbol{\pi}^*), \forall j \in [M]$.*

Essentially, Proposition 1 states that the optimal joint policy under DeCOM yields the same expected term-average return and costs as that of the CCMG. Such result further validates our choice of decomposing the policy as in DeCOM. In this paper, we adopt the practical approach of realizing each f_i and g_i by neural networks, and denote the parameters for f_i and g_i as θ_i and ϕ_i, respectively. To further simplify notation, we let $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_N]$, $\boldsymbol{\phi} = [\phi_1, \cdots, \phi_N]$, and treat both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ as vectors. We next represent agents' joint base policy as $\boldsymbol{f} = [f_1, \cdots, f_N]$, and joint perturbation policy as $\boldsymbol{g} = [g_1, \cdots, g_N]$. Thus, under the DeCOM framework, the return and costs satisfy that $J^R(\boldsymbol{\pi}) = J^R(\boldsymbol{f}, \boldsymbol{g})$ and $J^{C_j}(\boldsymbol{\pi}) = J^{C_j}(\boldsymbol{f}, \boldsymbol{g}), \forall j \in [M]$.

[4]The neighbor set can be decided by physical proximity or other factors, depending on the actual scenario.

Algorithm 1: Training Algorithm of DeCOM

1 Initialize $\mathcal{D} \leftarrow \emptyset$; Initialize reward critic Q^{η_0}, cost critics $Q^{\zeta_{j,0}}, \forall j \in [M]$, $\boldsymbol{\theta}_0$, $\boldsymbol{\phi}_0$; Initialize λ, δ;
2 $\eta'_0 \leftarrow \eta_0$, $\zeta'_{j,0} \leftarrow \zeta_{j,0}, \forall j \in [M]$, $\boldsymbol{\theta}'_0 \leftarrow \boldsymbol{\theta}_0$, $\boldsymbol{\phi}'_0 \leftarrow \boldsymbol{\phi}_0$;
3 **foreach** *episode* $k = 0$ *to max-episodes* **do**
4 **foreach** $t = 0$ *to episode-length* **do**
5 Each agent i selects base action b_i based on f_i, and shares it with $\mathcal{N}_i$;
6 Each agent i calculates action $a_i \leftarrow b_i + \lambda\, g_i(o_i, b_i, \boldsymbol{b}_{\mathcal{N}_i})$, and executes a_i;
7 Observe team-average reward r and costs $c^j, \forall j \in [M]$, and next global state s_{t+1};
8 Store experience $(s_t, \boldsymbol{b}, \boldsymbol{a}, r, \{c^j\}_{j=1}^M, s_{t+1})$ into $\mathcal{D}$;
9 Sample a random mini-batch of L transitions $\mathcal{B} = \{(s_l, \boldsymbol{b}_l, \boldsymbol{a}_l, r_l, \{c_l^j\}_{j=1}^M, s'_l)\}_{l=1}^L$ from $\mathcal{D}$;
10 Update reward critic by minimizing Eq. (5), and cost critics by minimizing Eq. (6) ;
11 Update $\boldsymbol{\theta}_k$ and $\boldsymbol{\phi}_k$ to $\boldsymbol{\theta}_{k+1}$ and $\boldsymbol{\phi}_{k+1}$ according to Alg. 2;
12 $\eta'_{k+1} \leftarrow \delta\, \eta_{k+1} + (1-\delta)\, \eta'_k$,;
13 $\zeta'_{j,k+1} \leftarrow \delta\, \zeta_{j,k+1} + (1-\delta)\, \zeta'_{j,k}, \forall j \in [M]$;
14 $\boldsymbol{\theta}'_{k+1} \leftarrow \delta\, \boldsymbol{\theta}_{k+1} + (1-\delta)\, \boldsymbol{\theta}'_k$,;
15 $\boldsymbol{\phi}'_{k+1} \leftarrow \delta\, \boldsymbol{\phi}_{k+1} + (1-\delta)\, \boldsymbol{\phi}'_k$;

Training Algorithm

Algorithm Overview

Our training algorithm of DeCOM follows the actor-critic framework, as shown in Alg. 1. At each episode k, agents interact with the environment and the experiences of such interactions are collected into buffer $\mathcal{D}$ (line 4-8). Then, the algorithm samples a mini-batch from $\mathcal{D}$, and updates the reward and cost critics by minimizing the TD error over the mini-batch (line 9-10). After that, $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ get updated through Alg. 2 (line 11), which will be elaborated in Sec. 15. Finally, Alg. 1 performs soft update for the target networks to stabilize learning (line 12-15). Next, we present our method of updating the critics, and the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ in detail.

Updating Critics

At each episode k of Alg.1, we update the reward critic by minimizing the TD error over the sampled mini-batch of L

Algorithm 2: Policy Update Algorithm

1 **Input:** Sampled mini-batch $\mathcal{B}$; $\boldsymbol{\theta}_k$, $\boldsymbol{\phi}_k$;
2 **Output:** $\boldsymbol{\theta}_{k+1}$, $\boldsymbol{\phi}_{k+1}$;
3 Update $\boldsymbol{\theta}_k$ to $\boldsymbol{\theta}_{k+1}$ by Eq. (7);
`// By Eq.(15) in Appendix A.2, if` $\boldsymbol{f}$ `is deterministic.`
4 $\boldsymbol{\varphi}_0 \leftarrow \boldsymbol{\phi}_k$;
5 **foreach** $w = 0$ *to* W **do**
6 $\quad j^* \leftarrow \arg\max_{j\in[M]} \tilde{\mathcal{L}}_j(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})$;
7 $\quad \boldsymbol{\varphi}_{w+1} \leftarrow \Gamma_\Phi\Big[\boldsymbol{\varphi}_w - \tau \cdot \text{Clip}\big(\nabla_{\boldsymbol{\phi}} \mathcal{L}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})\big)\Big]$;
`// ` Γ_Φ `is the projection operator.`
8 $\boldsymbol{\phi}_{k+1} \leftarrow \boldsymbol{\varphi}_W$;

transitions, given in the following Eq. (5),

$$\mathcal{L}(\eta_k) = \frac{1}{L}\sum_{l=1}^{L}\big(q_l^R - Q^{\eta_k}(s_l, \boldsymbol{a}_l)\big)^2, \tag{5}$$
$$\text{with } q_l^R = r_l + \gamma Q^{\eta'_k}(s'_l, \boldsymbol{a}'_l)|_{\boldsymbol{a}'_l \sim \boldsymbol{\pi}_{\boldsymbol{\theta}'_k, \boldsymbol{\phi}'_k}},$$

where Q^{η_k} is the reward action-value function with parameter η_k, $\boldsymbol{\pi}_{\boldsymbol{\theta}'_k, \boldsymbol{\phi}'_k}$ is the target policy with parameters $\boldsymbol{\theta}'_k$ and $\boldsymbol{\phi}'_k$.

The cost critics are updated in a similar manner with the TD error given in Eq. (6) for each $j \in [M]$,

$$\mathcal{L}(\zeta_{j,k}) = \frac{1}{L}\sum_{l=1}^{L}\big(q_l^{C_j} - Q^{\zeta_{j,k}}(s_l, \boldsymbol{a}_l)\big)^2, \tag{6}$$
$$\text{with } q_l^{C_j} = c_l^j + \gamma Q^{\zeta'_{j,k}}(s'_l, \boldsymbol{a}'_l)|_{\boldsymbol{a}'_l \sim \boldsymbol{\pi}_{\boldsymbol{\theta}'_k, \boldsymbol{\phi}'_k}},$$

where $Q^{\zeta_{j,k}}$ is the action-value function on cost j with parameter $\zeta_{j,k}$.

Updating Policies

Algorithm Overview We present in Alg. 2 the algorithm for updating the policy parameters, which is called on line 11 of Alg. 1. At each episode k of Alg. 1, Alg. 2 takes $\boldsymbol{\theta}_k$ and $\boldsymbol{\phi}_k$ as input, and updates $\boldsymbol{\theta}_k$ to $\boldsymbol{\theta}_{k+1}$ (line 3), and $\boldsymbol{\phi}_k$ to $\boldsymbol{\phi}_{k+1}$ (line 4-8), whose details will be elaborated in Section 8 and 8.

Updating $\boldsymbol{\theta}$ DeCOM updates $\boldsymbol{\theta}$ via policy gradient methods. Next, we present the policy gradient of $J^R(\boldsymbol{f}, \boldsymbol{g})$ under stochastic base policies in Theorem 2.

Theorem 2. *If each f_i is stochastic (e.g., Gaussian policy), then at each episode k of Alg. 1, the gradient of $J^R(\boldsymbol{f}, \boldsymbol{g})$ w.r.t. $\theta_i, \forall i \in [N]$, is*

$$\nabla_{\theta_i} J^R(\boldsymbol{f}, \boldsymbol{g}) \approx \mathbb{E}_{(s_0, \boldsymbol{b}, \boldsymbol{a})\sim\mathcal{B}}\Big[\nabla_{\theta_i} Q^{\eta_{k+1}}(s_0, \boldsymbol{a}) + \nabla_{\theta_i} \log f_i(b_i|o_i)\; Q^{\eta_{k+1}}(s_0, \boldsymbol{a})\Big]. \tag{7}$$

Note that the term $\nabla_{\theta_i} Q^{\eta_{k+1}}(s_0, \boldsymbol{a})$ in Eq. (7) implicitly shows that agents do share gradients in the training process as discussed in Section . As DeCOM does not restrict $\boldsymbol{f}$ to be stochastic, we also derive the policy gradient under deterministic base policies, whose mathematical expression together with the proofs and more discussions on Theorem 2 is presented in Appendix A.2.

Updating $\boldsymbol{\phi}$ In DeCOM, $\boldsymbol{g}$ perturbs the base action to satisfy constraints, whose parameter $\boldsymbol{\phi}$ solves the following constraints satisfaction problem:

$$\text{Find} \quad \boldsymbol{\phi} \in \Phi, \;\; \text{s.t.} \;\; J^{C_j}(\boldsymbol{f}, \boldsymbol{g}) \le D_j, \;\; \forall j \in [M], \tag{8}$$

where Φ is the space of $\boldsymbol{\phi}$. As an exhaustive search for $\boldsymbol{\phi}$ that solves Problem (8) is intractable, we switch to a learning approach. Given any $\boldsymbol{\theta}$, for each $j \in [M]$, the constraint violation loss is defined as

$$\mathcal{L}_j(\boldsymbol{\phi}; \boldsymbol{\theta}) = \Big(\max\big(0,\; J^{C_j}(\boldsymbol{f}, \boldsymbol{g}) - D_j\big)\Big)^2. \tag{9}$$

Given $\boldsymbol{\theta}_{k+1}$, we empirically approximate the above loss by

$$\tilde{\mathcal{L}}_j(\boldsymbol{\phi}; \boldsymbol{\theta}_{k+1}) = \big(\max(0, \mathop{\mathbb{E}}_{(s_0, \boldsymbol{a})\sim\mathcal{B}}\big[Q^{\zeta_{j,k+1}}(s_0, \boldsymbol{a})\big] - D_j)\big)^2. \tag{10}$$

As realized by the for loop (line 5-7) in Alg. 2, we update $\boldsymbol{\phi}$ with W iterations. This design is motivated by the convergence analysis presented in Section 8, and practical settings of W can be found in Appendix B.4. In each iteration w of Alg. 2, given the current value $\boldsymbol{\varphi}_w$ for the parameter $\boldsymbol{\phi}$, we find the cost $j^* \in [M]$ with the maximum empirical constraint violation loss $\tilde{\mathcal{L}}_j(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})$. Then, $\boldsymbol{\varphi}_w$ is updated to $\boldsymbol{\varphi}_{w+1}$ by projected gradient descent with the clipped version of the gradient $\nabla_{\boldsymbol{\phi}} \tilde{\mathcal{L}}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})$. Thus,

$$\boldsymbol{\varphi}_{w+1} = \Gamma_\Phi\Big[\boldsymbol{\varphi}_w - \tau \cdot \text{Clip}\big(\nabla_{\boldsymbol{\phi}} \tilde{\mathcal{L}}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})\big)\Big], \tag{11}$$

where τ is the learning rate, $\Gamma_\Phi[\boldsymbol{\varphi}]$ projects $\boldsymbol{\varphi}$ into the space Φ, and

$$\text{Clip}\big(\nabla_{\boldsymbol{\phi}} \tilde{\mathcal{L}}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})\big) = \begin{cases} \nabla_{\boldsymbol{\phi}} \tilde{\mathcal{L}}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1}), & \text{if } ||\nabla_{\boldsymbol{\phi}} \tilde{\mathcal{L}}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})|| \le G, \\ \frac{G \cdot \nabla_{\boldsymbol{\phi}} \tilde{\mathcal{L}}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})}{||\nabla_{\boldsymbol{\phi}} \tilde{\mathcal{L}}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})||}, & \text{otherwise,} \end{cases} \tag{12}$$

with G denoting the maximum allowable gradient norm. We adopt the above clipping operation to stabilize learning, which also helps Alg. 2 converge, as shown in Section 8. Furthermore, the estimation of constraint violation only considers the initial time step in Eq. (10) and (11), in practice, we implement a more efficient estimation method which utilize backward value function (Satija, Amortila, and Pineau 2020) to assign the constraint violation to each time step. See Appendix B.4 for more details.

Convergence Analysis Before formally stating Theorem 3 on the convergence of Alg. 2, we introduce two mild assumptions[5], including that the space Φ is compact and convex, and that $\mathcal{L}_j(\boldsymbol{\phi}; \boldsymbol{\theta}_{k+1})$ is L_j-smooth w.r.t. $\boldsymbol{\phi}$, $\forall j \in [M]$, with $L_{\max}$ denoting $\max\{L_1, \cdots, L_M\}$.

Theorem 3. *Let $\boldsymbol{\phi}$ be updated with the exact constraint violation losses given $\boldsymbol{\theta}_{k+1}$. That is, in each iteration w of Alg. 2, $\boldsymbol{\varphi}_{w+1}$ is set as $\Gamma_\Phi\big[\boldsymbol{\varphi}_w - \tau \cdot \text{Clip}(\nabla_{\boldsymbol{\phi}} \mathcal{L}_{j^*}(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1}))\big]$ with $j^* = \arg\max_{j\in[M]} \mathcal{L}_j(\boldsymbol{\varphi}_w; \boldsymbol{\theta}_{k+1})$. Then, for any $\epsilon > 0$ and*

[5]These assumptions are commonly adopted in existing works (e.g., (Tessler, Mankowitz, and Mannor 2019; Jacot, Gabriel, and Hongler 2018)).

$j \in [M]$, if both τL_{max} and τG^2 are sufficiently small, $\boldsymbol{\varphi}_w$ will converge in $H \leq \frac{\min_{\boldsymbol{\phi} \in \mathcal{X}} ||\boldsymbol{\phi}_k - \boldsymbol{\phi}||^2}{2\tau\epsilon}$ steps to the region

$$C_k \leq \mathcal{L}_j(\boldsymbol{\phi}; \boldsymbol{\theta}_{k+1}) \leq C_k + \frac{2\epsilon + \tau G^2}{2F(H)},$$

where the set $\mathcal{X}$ contains all $\boldsymbol{\phi}$ that is the solution to the minimal problem: $C_k = \min_{\boldsymbol{\phi} \in \Phi} \mathcal{L}_{j^*}(\boldsymbol{\phi}; \boldsymbol{\theta}_{k+1})$, and $F(H) = \min\left(1, \frac{G}{||\nabla_{\boldsymbol{\phi}} \mathcal{L}_{j^*}(\boldsymbol{\varphi}_H; \boldsymbol{\theta}_{k+1})||}\right)$.

Theorem 3 states that under mild conditions, Alg. 2 converges within limited iterations. Specifically, if Problem (8) is feasible, which happens when constraint bounds D_j are set appropriately, or the parameterization space Φ is large enough, then there exists $\boldsymbol{\phi} \in \Phi$ such that $\mathcal{L}_j(\boldsymbol{\phi}; \boldsymbol{\theta}_{k+1}) = 0, \forall j \in [M]$ and $C_k = 0$. In this case, $\boldsymbol{\varphi}_w$ will converge to an approximately feasible solution of Problem (8) with a maximum constraint violation of $\sqrt{\frac{2\epsilon + \tau G^2}{2F(H)}}$. Theorem 3 motivates us to set small τ and G, and use sufficient number of iterations to update $\boldsymbol{\varphi}_w$ in practice. See Appendix A.3 and B.4 for the proof of Theorem 3 and detailed hyper-parameter settings, respectively. Note that Theorem 3 only establishes the convergence of Alg. 2, while the convergence of Alg. 1 to the (near-) optimal policy is observed empirically in several applications, as shown in Fig. 3 and Fig. 4 in Sec. 8.

Experiments

Simulation Environments and Costs

To evaluate DeCOM, we construct four simulation environments, namely *CTC-safe* and *CTC-fair* which extend the cooperative treasure collection (CTC) environment (Iqbal and Sha 2019), as well as *constrained directional sensor network (CDSN)* which extends the directional sensor network (DSN) environment (Xu, Zhong, and Wang 2020) and *constrained large-scale fleet management (CLFM)*.

As in CTC, both CTC-safe and CTC-fair have two types of agents, namely *hunters* and *banks*. Hunters collect treasures and store them into banks, and treasures will get re-spawned randomly once collected. The action of an agent is to select a coordinate within a square box where it will reposition at the next time step. Each agent's reward is positively correlated with the amount of treasures stored in banks, and a hunter will be punished, if it collides with another hunter. Both CTC-safe and CTC-fair have 3 hunters and 1 bank in our experiments.

CTC-safe adds 3 randomly initialized unsafe regions into CTC, as shown in Fig. 2(a). Each unsafe region generates one type of cost, and each agent receives 1 for *unsafety* cost j, if it locates in unsafe region j. Each agent in CTC-fair receives an *unfairness* cost which equals to the maximal difference between agents' accumulated traveling distance.

CDSN extends the DSN environment (Xu, Zhong, and Wang 2020) to continuous action space. In CDSN, sensors adjust their directions to capture moving objects, as shown in Fig. 2(c). Each agent receives two immediate reward: individual reward counts the number of objects it captured, shared global reward calculates the ratio of all captured objects. Each agent also receives an operational cost positively related to the angle adjusted. The goal is to maximize the accumulated summed reward, while satisfying the constraint on the average accumulated cost.

CLFM treats vehicles of an online-hailing platform[6] as agents, and focuses on relocating them in a distributed way, so as to maximize the revenue of the whole system under constraints on city-wide demand-supply gap and unfairness among drivers' incomes. More specifically, each agent receives a *demand-supply gap* cost that equals to the KL-divergence between the idle vehicle and order distributions, as well as an *unfairness* cost defined as the squared difference between agents' average accumulated income and its own. CLFM is built with a public city-scale dataset[7] that contains approximate 1 million orders from November 1 to November 30, 2016 in Chengdu, China. In our simulation, we consider 500 vehicles and divide the urban area of Chengdu into 103 equal-size hexagon grids. An agent's action is to choose a weight vector locally, which is multiplied with the feature vector of each candidate grid to obtain a score. Then, the grid where the agent will reposition is sampled based on grids' scores. More details on these settings are in Appendix B.

Algorithms and Neural Network Structures

We implement the following algorithms: (1) **Fixed Penalty (FP)**. FP treats costs as penalties by adding to the reward. Each type of cost is multiplied with an identical weight chosen from the set $\mathcal{W} = \{0, -0.1, -1.0, -100\}$. We let FP-$|\omega|$ denote FP with weight $\omega \in \mathcal{W}$. (2) **Lagrangian (La)**. La extends RCPO (Tessler, Mankowitz, and Mannor 2019) to CCMG by replacing the single-agent reward (costs) with the team-average reward (costs) and single-agent policy with agents' joint policy. (3) **Nocomm DeCOM (DeCOM-N)**. DeCOM-N is a variant of DeCOM where agents do not share base actions. That is, $a_i = b_i + \lambda\ g_i(o_i, b_i),$ with $b_i \sim f_i(\cdot|o_i), \forall i \in [N]$. (4) **Independent DeCOM (DeCOM-I)**. DeCOM-I is a variant of DeCOM, where the input of each g_i is only o_i. That is, $a_i = b_i + \lambda\ g_i(o_i),$ with $b_i \sim f_i(\cdot|o_i), \forall i \in [N]$. (5) **DeCOM-A**. We denote the original DeCOM which retains all communication as DeCOM-A to align with its variants.

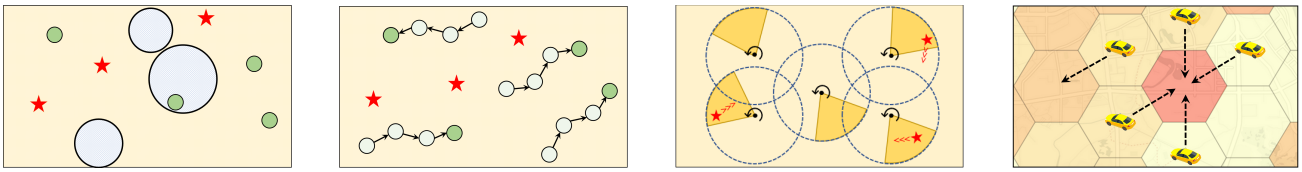
Note that FP-0 is exactly the unconstrained MARL algorithm that aims to maximize the expected team-average return without any constraints. Based on the performance of FP-0, we select the neural network structures as follows: in CTC-safe and CTC-fair, we set $\boldsymbol{f}$ as deterministic and use MADDPG critics (Lowe et al. 2017); in CDSN, we use stochastic $\boldsymbol{f}$ and MADDPG critics; in CLFM, we use stochastic $\boldsymbol{f}$ and Mean-Field critics (Yang et al. 2018). We set λ in CTC-safe, CDSN and CLFM as 1, and 0.01 in CTC-fair. Due to space limit, we put more discussions about chooing λ and detailed training curves in Appendix B.5.

Results Comparison

Table 1 shows the test results of CTC-safe and CTC-fair. In CTC-safe, the constraint bound for each unsafe region is set

[6]Examples include Uber (https://www.uber.com/) and Didi Chuxing (http://www.didichuxing.com/en/).

[7]Data source: DiDi Chuxing GAIA Open Dataset Initiative (https://gaia.didichuxing.com).



(a) CTC-safe. (b) CTC-fair. (c) CDSN. (d) CLFM.

Figure 2: Simulation environments, where stars represent treasures, areas I-III are unsafe regions 1-3 in Fig. 2(a), Fig. 2(b) shows agents' trajectories. Black dots refer to deployed sensors and stars are moving objects to capture in Fig. 2(c). The dashed arrows represent repositioning decisions and darker grids have larger demand-supply gaps in Fig. 2(d).

Algorithms	CTC-safe				CTC-fair	
	Reward	Unsafety 1 (0.6)	Unsafety 2 (0.8)	Unsafety 3 (1.0)	Reward	Unfairness (0)
FP-0	1.67 ± 1.08	0.82 ± 0.20	1.41 ± 0.20	2.02 ± 0.47	4.46 ± 0.54	8.91 ± 1.03
FP-0.1	-1.42 ± 0.86	0.75 ± 0.08	1.37 ± 0.13	2.16 ± 0.24	3.42 ± 0.78	9.96 ± 1.65
FP-1.0	-1.74 ± 0.98	0.82 ± 0.61	1.46 ± 1.05	2.19 ± 1.44	0.00 ± 0.25	9.83 ± 2.10
FP-100	-1.95 ± 0.95	0.61 ± 0.37	0.82 ± 0.65	1.45 ± 0.96	-1.21 ± 0.16	11.99 ± 4.38
La	-1.52 ± 0.24	1.05 ± 0.35	1.47 ± 0.59	2.38 ± 0.69	-1.41 ± 0.08	8.99 ± 3.64
DeCOM-I	**-0.74** ± 0.45	**0.38** ± 0.09	**0.44** ± 0.27	**0.65** ± 0.11	3.37 ± 0.60	**8.05** ± 0.20
DeCOM-N	-1.20 ± 0.06	**0.43** ± 0.28	**0.41** ± 0.26	**0.75** ± 0.26	**4.36** ± 0.65	9.22 ± 0.78
DeCOM-A	-1.24 ± 0.25	**0.43** ± 0.11	**0.45** ± 0.19	**0.64** ± 0.20	3.76 ± 0.84	**8.11** ± 0.08

Table 1: CTC-safe and CTC-fair test results (average ± standard deviation). The constraint thresholds are included in the brackets after the cost name for easy reading. In the reward columns, the highest value is in bold. In the cost columns, the values that are below the threshold or have the minimal constraint violation are in bold. Results of FP-0 are not in bold, since FP-0 is an unconstrianed algorithm.

Algorithms	Reward	Num. of Captured	Coverage Ratio	Operational Cost (20)
FP-0	62.26 ± 4.73	22.23 ± 1.59	0.40 ± 0.03	69.65 ± 2.77
FP-0.1	50.96 ± 5.24	18.19 ± 1.56	0.32 ± 0.04	61.10 ± 7.23
FP-1.0	51.14 ± 7.32	18.24 ± 2.71	0.33 ± 0.05	60.79 ± 6.35
FP-100	47.72 ± 1.82	17.00 ± 0.60	0.30 ± 0.01	63.49 ± 16.56
La	46.79 ± 0.47	16.71 ± 0.29	0.30 ± 0.00	80.82 ± 23.01
DeCOM-I	34.67 ± 25.58	19.15 ± 8.53	0.32 ± 0.15	58.19 ± 15.41
DeCOM-N	51.93 ± 24.05	13.75 ± 9.79	0.21 ± 0.16	**32.88** ± 4.25
DeCOM-A	**58.41** ± 11.21	**21.05** ± 3.75	**0.37** ± 0.10	58.90 ± 26.23

Table 2: CDSN test results (average ± standard deviation).

Algorithms	Revenue	ORR	Demand-Supply Gap (90)	Unfairness (60)
FP-0	18944.83 ± 48.91	0.49 ± 0.07	100.29 ± 1.80	74.57 ± 3.03
FP-0.1	18809.87 ± 41.58	0.48 ± 0.03	103.25 ± 1.15	81.08 ± 3.31
FP-1.0	18815.76 ± 66.79	0.48 ± 0.08	103.10 ± 1.46	78.98 ± 2.13
FP-100	18840.14 ± 55.99	0.49 ± 0.10	100.52 ± 1.94	81.05 ± 3.78
La	18819.07 ± 235.92	0.49 ± 0.25	103.22 ± 4.19	82.54 ± 14.61
DeCOM-I	19004.70 ± 76.88	0.49 ± 0.09	97.33 ± 1.28	70.40 ± 2.48
DeCOM-N	18668.26 ± 362.78	0.48 ± 0.37	105.57 ± 5.06	89.61 ± 20.29
DeCOM-A	**19286.68** ± 78.42	**0.50** ± 0.14	**89.11** ± 3.53	**62.99** ± 1.35

Table 3: CLFM test results (average ± standard deviation).

as 0.6, 0.8 and 1.0. Among all algorithms, FP-0 achieves the highest reward, but violates all constraints. Although DeCOM-A and its variants DeCOM-I and DeCOM-N do not have as high reward as FP-0, but they satisfy all constraints. La has the worst performance on all constraints. As shown in Fig. 3, La's training curve fluctuates heavily and fails to converge finally.

In CTC-fair, the constraint bound for unfairness is 0. Test results in Table 1 show that no algorithms satisfy the constraints. DeCOM-I and DeCOM-A have relatively low vi-

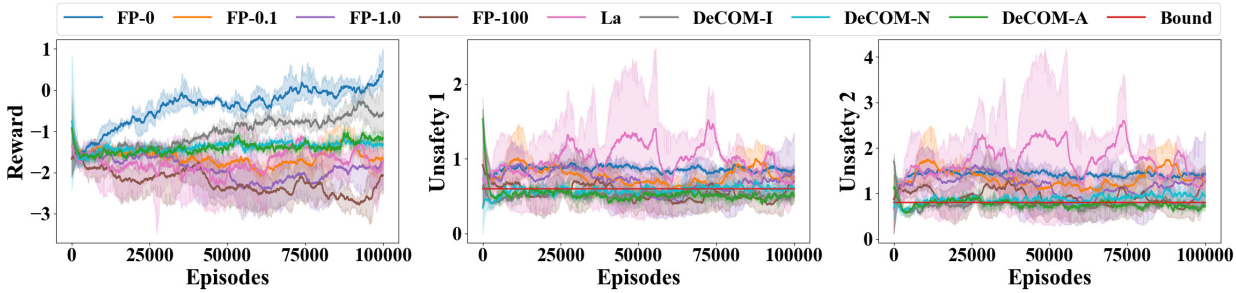


Figure 3: Training curves of CTC-safe.

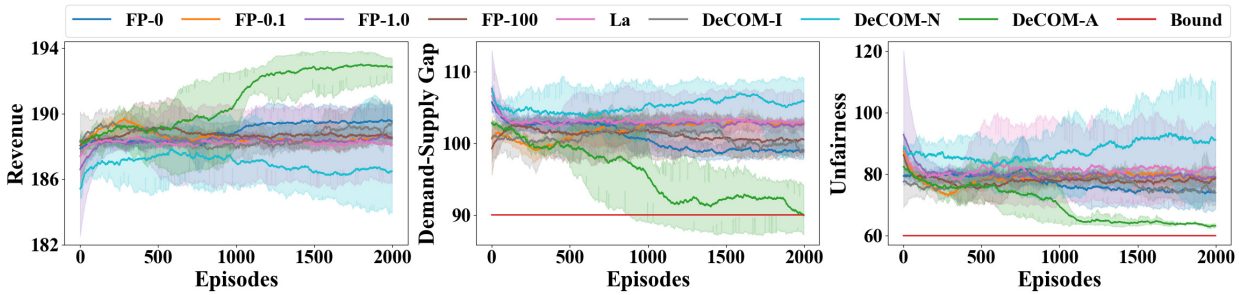


Figure 4: Training curves of CLFM, where revenue curve is scaled down by 100 times.

olation on unfairness, FP-0 and DeCOM-N have relatively high reward. Interestingly, DeCOM-A performs slightly better in CTC-fair comparing to CTC-safe. This phenomenon is highly related to the fact that ensuring fairness typically requires more agent interactions than safety. More specifically, an agent can avoid unsafe regions by its own observation, even without others' information. However, agent communication in DeCOM-A becomes more beneficial to help an agent control its traveling distance for fairness. Even with such promising observation, DeCOM-A still violates the constraint heavily. Achieving a better trade-off between reward and unfairness remains an interesting problem to explore in the future.

Table 2 lists the test results in CDSN. DeCOM-A achieves the highest value in reward, number of captured objects and global coverage ratio except FP-0. The constraint bound on the operational cost is set as 20. DeCOM-N has the lowest constraint violation. Due to space limit, we put more discussions on the results in CDSN in Appendix B.5.

Table 3 shows the test results in CLFM, where the important metric *Order Response Rate (ORR)* that measures the ratio of served orders is also given. The constraint bound for the demand-supply gap and unfairness cost is set as 90 and 60, respectively. As shown in Table 3, DeCOM-A satisfies both constraints. Specifically, DeCOM-A has the lowest demand-supply gap, which makes it reasonable for DeCOM-A to achieve the highest revenue and ORR. Meanwhile, DeCOM-A has the lowest constraint violation on unfairness. DeCOM-A's better performance versus other baselines comes from its communication mechanism, which essentially provides an agent the repositioning intentions of its neighbors. As we define an agent's neighbors as those in its neighboring grids, such information could help an agent decide to reposition to grids with less vehicles and more orders.

One may argue that DeCOM violates the constraints in some environments, such as CLFM in Fig. 4. However, we want to emphasize that, though the constraints are violated, DeCOM obtains the highest return with the smallest constraint violations among all algorithms.

Related Works

Multi-Agent Reinforcement Learning. MARL is widely used to solve Markov games (Littman 1994), which can be categorized into competitive settings (Foerster et al. 2018a; Xie et al. 2020), cooperative settings (Lin et al. 2019; Böhmer, Kurin, and Whiteson 2020), and a mixture of them (Lowe et al. 2017; Bai et al. 2021). As aforementioned, we focus on the cooperative setting in this paper. A series of recent MARL works for such settings, ranging from VDN (Sunehag et al. 2017) to QMIX (Rashid et al. 2018, 2020), adopt value-based methods that learn each agent's individual Q function to represent the global Q function by different mixing networks. Although these methods achieve good results for discrete action Markov games (Samvelyan et al. 2019), they are generally not applied in our continuous action setting. DeCOM lies in the line of policy-based MARL methods, including MADDPG (Lowe et al. 2017), mean-field based method (Yang et al. 2018), COMA (Foerster et al. 2018b), MAAC (Iqbal and Sha 2019), FACMAC (Peng et al. 2021), and DOP (Wang et al. 2021). However, these methods are designed to solve unconstrained Markov games, which are thus not applicable in our constrained setting.

A family of MARL frameworks exploit *communication* (Foerster et al. 2016a; Kim et al. 2019; Ding, Huang, and

Lu 2020; Wang et al. 2020) by sharing either (encoded) observations or latent variables among agents. Conversely, the messages in DeCOM are agents' base actions, which have a clear and explicit meaning. Besides, DeCOM allows agents to concurrently receive other agents' base actions and the gradient backflows at the current step, which brings more timeliness to help agents make better decisions.

Constrained Reinforcement Learning. Various deep constrained reinforcement learning frameworks are proposed to solve constrained MDPs (CMDPs) (Altman 1999). They either convert a CMDP into an unconstrained min-max problem by introducing Lagrangian multipliers (Tessler, Mankowitz, and Mannor 2019; Le, Voloshin, and Yue 2019; Paternain et al. 2019; Calian et al. 2020; Chow et al. 2015; Bharadhwaj et al. 2021), or seek to obtain the optimal policy by directly solving constrained optimization problems (Achiam et al. 2017; Yang et al. 2020, 2021; Yu et al. 2019; Wen and Topcu 2018; Satija, Amortila, and Pineau 2020; Chow et al. 2018). However, it is hard to scale these single-agent methods to our multi-agent setting due to computational inefficiency, especially those rely on solving constrained optimiztion problems (Wagener, Boots, and Cheng 2021).

We also note that our DeCOM aims to solve CCMG, which involves constraints defined over long-term returns instead of each state (Wagener, Boots, and Cheng 2021; Sheebaelhamd et al. 2021; Thomas, Luo, and Ma 2021). Furthermore, rather than analyzing linear or finite MDPs (Ding et al. 2020; Efroni, Mannor, and Pirotta 2020; Liu et al. 2021; Xu, Liang, and Lan 2021), DeCOM aims at high-dimensional real-world applications, which typically involve continuous controls and complex non-linear dynamics (Kumar et al. 2020).

Similar to DeCOM, one line of prior works (Luo, Sun, and Kapoor 2020; Qin et al. 2021; Lu et al. 2021) developed constrained MARL frameworks in other settings. In particular, (Luo, Sun, and Kapoor 2020; Qin et al. 2021) focus on designing model-based control method to avoid collisions, which is not applicable to our scenario with an unknown state transition kernel; (Lu et al. 2021) studies the scenario where each agent has a local safety constraint, whereas DeCOM is designed for cooperative settings with constraints on team-average costs. Moreover, different from the aforementioned works (Luo, Sun, and Kapoor 2020; Qin et al. 2021; Lu et al. 2021) that focus on specific applications, DeCOM could be applied in a wider range of applications that rely on intensive team cooperation, such as fleet management (Lin et al. 2018; Ding et al. 2021), order dispatch (Li et al. 2019; Sun et al. 2022), multi-agent patrolling (Nguyen, Kumar, and Lau 2018) and directional sensor networks (Xu, Zhong, and Wang 2020).

Conclusions

In this paper, we propose a novel constrained cooperative MARL framework, named DeCOM, which facilitates agent cooperation by empowering information sharing among agents. By iteratively solving the unconstrained optimization problem on reward and the constrains satisfaction problem on costs, DeCOM learns policies in a scalable, efficient, and easy-to-implement manner. Experiment results in four real-world applications validate DeCOM's effectiveness.

Acknowledgements

This work was supported in part by NSF China (No. U21A20519, U20A20181, 61902244, 62202298), and in part by Fellowship of China Postdoctoral Science Foundation (No. 22Z020702116).

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *ICML*, 22–31. PMLR.

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

Bai, Y.; Jin, C.; Wang, H.; and Xiong, C. 2021. Sample-Efficient Learning of Stackelberg Equilibria in General-Sum Games. In *NIPS*.

Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G.; McGrew, B.; and Mordatch, I. 2020. Emergent Tool Use From Multi-Agent Autocurricula. arXiv:1909.07528.

Bharadhwaj, H.; Kumar, A.; Rhinehart, N.; Levine, S.; Shkurti, F.; and Garg, A. 2021. Conservative Safety Critics for Exploration. In *ICLR*.

Böhmer, W.; Kurin, V.; and Whiteson, S. 2020. Deep coordination graphs. In *ICML*, 980–991. PMLR.

Calian, D. A.; Mankowitz, D. J.; Zahavy, T.; Xu, Z.; Oh, J.; Levine, N.; and Mann, T. 2020. Balancing Constraints and Rewards with Meta-Gradient D4PG. In *ICLR*.

Chow, Y.; Nachum, O.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2018. A Lyapunov-Based Approach to Safe Reinforcement Learning. In *NIPS*.

Chow, Y.; Nachum, O.; Faust, A.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2019. Lyapunov-based Safe Policy Optimization for Continuous Control. arXiv:1901.10031.

Chow, Y.; Tamar, A.; Mannor, S.; and Pavone, M. 2015. Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. In *NIPS*.

Ding, D.; Zhang, K.; Basar, T.; and Jovanovic, M. R. 2020. Natural Policy Gradient Primal-Dual Method for Constrained Markov Decision Processes. In *NIPS*, volume 33, 8378–8390.

Ding, R.; Yang, Z.; Wei, Y.; Jin, H.; and Wang, X. 2021. Multi-Agent Reinforcement Learning for Urban Crowd Sensing with For-Hire Vehicles. In *INFOCOM*, 1–10.

Ding, Z.; Huang, T.; and Lu, Z. 2020. Learning Individually Inferred Communication for Multi-Agent Cooperation. In *NIPS*, volume 33, 22069–22079.

Efroni, Y.; Mannor, S.; and Pirotta, M. 2020. Exploration-Exploitation in Constrained MDPs. arXiv:2003.02189.

Foerster, J.; Assael, I. A.; de Freitas, N.; and Whiteson, S. 2016a. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *NIPS*.

Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016b. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *NIPS*.

Foerster, J. N.; Chen, R. Y.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; and Mordatch, I. 2018a. Learning with opponent-learning awareness. In *AAMAS*, 122–130.

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018b. Counterfactual Multi-Agent Policy Gradients. In *AAAI*.

Hughes, E.; Leibo, J. Z.; Phillips, M.; Tuyls, K.; Dueñez Guzman, E.; García Castañeda, A.; Dunning, I.; Zhu, T.; McKee, K.; Koster, R.; Roff, H.; and Graepel, T. 2018. Inequity aversion improves cooperation in intertemporal social dilemmas. In *NIPS*.

Iqbal, S.; and Sha, F. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *ICML*, 2961–2970. PMLR.

Jacot, A.; Gabriel, F.; and Hongler, C. 2018. Neural tangent kernel: Convergence and generalization in neural networks. In *NIPS*.

Jaques, N.; Lazaridou, A.; Hughes, E.; Gulcehre, C.; Ortega, P.; Strouse, D.; Leibo, J. Z.; and De Freitas, N. 2019. Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. In *ICML*, 3040–3049. PMLR.

Jiang, J.; and Lu, Z. 2018. Learning Attentional Communication for Multi-Agent Cooperation. In *NIPS*.

Kim, D.; Moon, S.; Hostallero, D.; Kang, W. J.; Lee, T.; Son, K.; and Yi, Y. 2019. Learning to Schedule Communication in Multi-agent Reinforcement Learning. In *ICLR*.

Kumar, S.; Kumar, A.; Levine, S.; and Finn, C. 2020. One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL. In *NIPS*.

Le, H.; Voloshin, C.; and Yue, Y. 2019. Batch Policy Learning under Constraints. In *ICML*, 3703–3712. PMLR.

Li, M.; Qin, Z.; Jiao, Y.; Yang, Y.; Wang, J.; Wang, C.; Wu, G.; and Ye, J. 2019. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *WWW*, 983–994.

Lin, K.; Zhao, R.; Xu, Z.; and Zhou, J. 2018. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *KDD*, 1774–1783.

Lin, Y.; Zhang, K.; Yang, Z.; Wang, Z.; Başar, T.; Sandhu, R.; and Liu, J. 2019. A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning. In *CDC*, 5562–5567. IEEE.

Littman, M. L. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *ICML*, 157–163.

Liu, T.; Zhou, R.; Kalathil, D.; Kumar, P.; and Tian, C. 2021. Learning Policies with Zero or Bounded Constraint Violation for Constrained MDPs. In *NIPS*.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *NIPS*.

Lu, S.; Zhang, K.; Chen, T.; Başar, T.; and Horesh, L. 2021. Decentralized Policy Gradient Descent Ascent for Safe Multi-Agent Reinforcement Learning. In *AAAI*, volume 35, 8767–8775.

Luo, W.; Sun, W.; and Kapoor, A. 2020. Multi-Robot Collision Avoidance under Uncertainty with Probabilistic Safety Barrier Certificates. In *NIPS*.

Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2018. Credit Assignment for Collective Multiagent RL with Global Rewards. In *NIPS*.

Paternain, S.; Chamon, L. F.; Calvo-Fullana, M.; and Ribeiro, A. 2019. Constrained reinforcement learning has zero duality gap. In *NIPS*, volume 32.

Peng, B.; Rashid, T.; Schroeder de Witt, C.; Kamienny, P.-A.; Torr, P.; Böhmer, W.; and Whiteson, S. 2021. Facmac: Factored multi-agent centralised policy gradients. In *NIPS*.

Qin, Z.; Zhang, K.; Chen, Y.; Chen, J.; and Fan, C. 2021. Learning Safe Multi-Agent Control with Decentralized Neural Barrier Certificates. In *ICLR*.

Qu, C.; Mannor, S.; Xu, H.; Qi, Y.; Song, L.; and Xiong, J. 2019. Value Propagation for Decentralized Networked Deep Multi-agent Reinforcement Learning. In *NIPS*.

Rashid, T.; Farquhar, G.; Peng, B.; and Whiteson, S. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *NIPS*.

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*.

Samvelyan, M.; Rashid, T.; de Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. arXiv:1902.04043.

Satija, H.; Amortila, P.; and Pineau, J. 2020. Constrained Markov Decision Processes via Backward Value Functions. In *ICML*, 8502–8511. PMLR.

Sheebaelhamd, Z.; Zisis, K.; Nisioti, A.; Gkouletsos, D.; Pavllo, D.; and Kohler, J. 2021. Safe Deep Reinforcement Learning for Multi-Agent Systems with Continuous Action Spaces. arXiv:2108.03952.

Sun, J.; Jin, H.; Yang, Z.; Su, L.; and Wang, X. 2022. Optimizing Long-Term Efficiency and Fairness in Ride-Hailing via Joint Order Dispatching and Driver Repositioning. In *KDD*, 3950–3960.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2017. Value-decomposition networks for cooperative multi-agent learning. In *AAMAS*.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2019. Reward constrained policy optimization. In *ICLR*.

Thomas, G.; Luo, Y.; and Ma, T. 2021. Safe Reinforcement Learning by Imagining the Near Future. In *NIPS*.

Wagener, N.; Boots, B.; and Cheng, C.-A. 2021. Safe Reinforcement Learning Using Advantage-Based Intervention. In *ICML*, 10630–10640. PMLR.

Wang, E.; Ding, R.; Yang, Z.; Jin, H.; Miao, C.; Su, L.; Zhang, F.; Qiao, C.; and Wang, X. 2022. Joint Charging and Relocation Recommendation for E-Taxi Drivers via Multi-Agent Mean Field Hierarchical Reinforcement Learning. *TMC*, 21: 1274–1290.

Wang, R.; He, X.; Yu, R.; Qiu, W.; An, B.; and Rabinovich, Z. 2020. Learning efficient multi-agent communication: An information bottleneck approach. In *ICML*, 9908–9918. PMLR.

Wang, Y.; Han, B.; Wang, T.; Dong, H.; and Zhang, C. 2021. DOP: Off-Policy Multi-Agent Decomposed Policy Gradients. In *ICLR*.

Wen, M.; and Topcu, U. 2018. Constrained Cross-Entropy Method for Safe Reinforcement Learning. In *NIPS*.

Xie, Q.; Chen, Y.; Wang, Z.; and Yang, Z. 2020. Learning zero-sum simultaneous-move markov games using function approximation and correlated equilibrium. In *COLT*, 3674–3682. PMLR.

Xu, J.; Zhong, F.; and Wang, Y. 2020. Learning Multi-Agent Coordination for Enhancing Target Coverage in Directional Sensor Networks. In *NIPS*.

Xu, T.; Liang, Y.; and Lan, G. 2021. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *ICML*, 11480–11491. PMLR.

Yang, T.-Y.; Rosca, J.; Narasimhan, K.; and Ramadge, P. J. 2020. Projection-based constrained policy optimization. In *ICLR*.

Yang, T.-Y.; Rosca, J.; Narasimhan, K.; and Ramadge, P. J. 2021. Accelerating Safe Reinforcement Learning with Constraint-mismatched Baseline Policies. In *ICML*, 11795–11807. PMLR.

Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; and Wang, J. 2018. Mean field multi-agent reinforcement learning. In *ICML*, 5571–5580. PMLR.

Yu, M.; Yang, Z.; Kolar, M.; and Wang, Z. 2019. Convergent policy optimization for safe reinforcement learning. In *NIPS*, volume 32.

Zhang, K.; Yang, Z.; Liu, H.; Zhang, T.; and Basar, T. 2018. Fully decentralized multi-agent reinforcement learning with networked agents. In *ICML*, 5872–5881. PMLR.