

DISTRIBUTED ALGORITHM FOR MULTI-OBJECTIVE MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors

Paper under double-blind review

ABSTRACT

Multi-objective reinforcement learning (MORL) aims to optimize multiple conflicting objectives for a single agent, where finding Pareto-optimal solutions is NP-hard and existing algorithms are often centralized with high computational complexity, limiting their practical applicability. Multi-objective multi-agent reinforcement learning (MOMARL) extends MORL to multiple agents, which not only increases computational complexity exponentially due to the global state-action space, but also introduces communication challenges, as agents cannot continuously communicate with a central coordinator in large-scale scenarios. This necessitates distributed algorithm, where each agent relies only on the information of its neighbors within a limited range rather than depending on the global scale. To address these challenges, we propose a distributed MOMARL algorithm in which each agent leverages only the state of its κ -hop neighbors and locally adjusts the weights of multiple objectives through a consensus protocol. We introduce an approximated policy gradient that reduces the dependency on global actions and a linear function approximation that limits the state space to local neighborhoods. Each agent i 's computational complexity is thus reduced from $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$ with global state-action space in centralized algorithms to $\mathcal{O}(|\mathcal{S}_{\mathcal{N}_i^\kappa}||\mathcal{A}_i|)$ with κ -neighborhood state and local action space. We prove that the algorithm converges to a Pareto-stationary solution at a rate of $\mathcal{O}(1/T)$ and demonstrate in simulations for robot path planning that our approach achieves higher multi-objective values than state-of-the-art method.

1 INTRODUCTION

As real-world tasks grow increasingly complex, many scenarios naturally involve multiple conflicting objectives, motivating the study of multi-objective reinforcement learning (MORL). For instance, in robotic path planning (Zhang et al., 2016), a robotic agent may aim to simultaneously minimize path length, avoid collisions, and maximize information collection.

Different from the rapid development of traditional reinforcement learning (RL) (Grondman et al, 2012; Zhang et al, 2021), research on MORL (Ge et al., 2022; Stamenkovic et al, 2022) remains in its infancy due to the inherent conflicts among multiple objectives. Unlike scalar-reward RL, in MORL the improvement of one objective may degrade others, making standard policy optimization insufficient. A common approach to tackle MORL is to assign fixed weights to objectives and reduce the problem to a single-objective RL (Blondin & Hale, 2020); however, this requires prior knowledge of objective importance and may fail to explore the full Pareto front. To address this limitation, a more rigorous metric is Pareto optimality, where no objective can be improved without degrading others.

However, for non-convex MORL problems, finding exact Pareto-optimal solutions is NP-hard. Consequently, practical algorithms aim for ϵ -Pareto stationary solutions (Sener & Koltun, 2018), which provide a necessary condition for approximate Pareto optimality. On the algorithmic side, for MORL problem with continuous action spaces, (Chen et al., 2021) proposed an actor-critic MORL algorithm based on deterministic policy gradients (Silver et al., 2014) to directly optimize multiple objectives. For MORL with discrete action spaces, (Zhou et al., 2024) introduced a unified multi-objective actor-critic framework applicable to both discounted and average-reward settings, where stochastic policy parameters are updated via a multi-gradient descent approach (Désidéri, 2012), ensuring convergence toward ϵ -Pareto stationary solutions.

The aforementioned methods are all directed towards addressing the MORL problem in a centralized setting or for a single agent. However, practical applications of MORL problems often involve multi-agents. For instance, teams of robots need to decide themselves how to explore distinct regions by simultaneously minimizing energy consumption and travel time. In comparison to the MORL problem with single-agent, the multi-objective multi-agent problem (MOMARL) (Rădulescu, 2020) is more intricate as it encompasses not only potential conflicts among different objectives but also

interactions between the distributed agents with limited communication. An intuitive approach to the MOMARL problem is to consider it as a MORL problem with a single agent, where the state and action are represented by the joint states and joint actions of all agents, respectively. However, as the number of agents increases, the size of their joint state-action space grows exponentially, and in large-scale scenarios, agents cannot communicate with a central coordinator continuously. These characteristic renders the current algorithms used for solving MORL problems with a single agent (Chen et al., 2021; Zhou et al., 2024) unsuitable for large-scale scenarios with multi-agents. Consequently, the MOMARL problem poses new challenges to the design of scalable algorithms and their theoretical analysis.

In this paper, we aim to address the following problem: *How to develop a fully distributed algorithm for the MOMARL problem and ensure its convergence to Pareto-stationary of the multi-objective function?* The contributions of this paper are described as follows.

1. We first propose a novel approximated policy gradient for each agent i , which reduces the global action $\mathbf{a}$ required by centralized algorithms to the agent’s local action a_i . Furthermore, to reduce the dimensionality of state information, we employ a linear function approximation to restrict agent i ’s state to the neighborhood state $s_{\mathcal{N}_i^\kappa}$, encompassing only its κ -hop neighbors. The per-agent computational complexity is thus reduced from $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$ in centralized algorithms to $\mathcal{O}(|\mathcal{S}_{\mathcal{N}_i^\kappa}||\mathcal{A}_i|)$.
2. We propose a novel distributed algorithm in which each agent only uses the policy gradient information from its immediate neighbors, while collaboratively adjusting the weights of multiple objectives via a consensus protocol. This design enables the agents to perform cooperative optimization toward a Pareto-stationary solution, without requiring access to the policy gradient information of agents beyond the direct neighbors.
3. We prove that the proposed distributed algorithm, despite relying only on local neighborhood information, achieves convergence to an ϵ -Pareto-stationary solution at a rate of $\mathcal{O}(1/T)$, matching the convergence speed of centralized algorithms. Moreover, we run simulations in a robot path planning environment and show our algorithm converges to greater multi-objective values as compared to the extension of the latest MORL algorithm (Zhou et al., 2024), and performs close to the central optimum with much shorter running time.

For the sake of convenience, some key functions in this paper are presented in Table 1.

Table 1: Symbols and functions.

Symbols	Annotation
$Q^m(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta})$	Global Q -function in the m -th objective under joint policy $\boldsymbol{\pi}_\theta$
$Q_i^m(\mathbf{s}, \mathbf{a}; \boldsymbol{\theta})$	Local Q -function of agent i in the m -th objective under joint policy $\boldsymbol{\pi}_\theta$
$Q_{\text{tru},i}^m(s_{\mathcal{N}_i^\kappa}, a_{\mathcal{N}_i^\kappa}; \boldsymbol{\theta})$	Graph-truncated Q -function of agent i in the m -th objective under joint policy $\boldsymbol{\pi}_\theta$
$\nabla_{\theta_i} J_{\text{tru},i}^m(\boldsymbol{\theta})$	Graph-truncated policy gradient of agent i in the m -th objective under joint policy $\boldsymbol{\pi}_\theta$
$\bar{Q}_i^m(\mathbf{s}, a_i; \boldsymbol{\theta})$	Action-averaged Q -function of agent i in the m -th objective under policy $\boldsymbol{\pi}_\theta$
$\nabla_{\theta_i} J_{\text{app},i}^m(\boldsymbol{\theta})$	Approximated policy gradient of agent i in the m -th objective under joint policy $\boldsymbol{\pi}_\theta$
$\hat{Q}_i^m(s_{\mathcal{N}_i^\kappa}, a_i; w_i^m)$	Linear approximation function of agent i in the m -th objective

2 THE NEW MOMARL PROBLEM FORMULATION AND PRELIMINARIES

2.1 MODEL OF THE MOMARL PROBLEM

The MOMARL problem is described as $(\mathcal{N}, \mathcal{M}, \mathcal{G}(\mathcal{N}, \mathcal{E}), \{\mathcal{S}_i\}_{i \in \mathcal{N}}, \{\mathcal{A}_i\}_{i \in \mathcal{N}}, \{\mathcal{P}_i\}_{i \in \mathcal{N}}, \boldsymbol{\rho}, \{r_i^m\}_{i \in \mathcal{N}, m \in \mathcal{M}}, \gamma)$, where $\mathcal{N} = \{1, \dots, N\}$ and $\mathcal{M} = \{1, \dots, M\}$ represent the agent set and the objective set, respectively. $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ represents the communication network among agents with $\mathcal{E}$ being the set of edges¹. For integer $\kappa \geq 1$, denote $\mathcal{N}_i^\kappa$ as the κ -hop neighborhood of agent i and $\mathcal{N}_{-i}^\kappa = \mathcal{N} \setminus \mathcal{N}_i^\kappa$.

State and action: $\mathcal{S}_i$ and $\mathcal{A}_i$ represent the local state space and the local action space of agent i , respectively. Denote $\mathcal{S} = \prod_{i=1}^N \mathcal{S}_i$ and $\mathcal{A} = \prod_{i=1}^N \mathcal{A}_i$ as the global state space and the global action space, respectively. Denote $\mathbf{s} = (s_1, \dots, s_N) \in \mathcal{S}$ and $\mathbf{a} = (a_1, \dots, a_N) \in \mathcal{A}$ as the global state and the global action of agents, where $s_i \in \mathcal{S}_i$ and $a_i \in \mathcal{A}_i$ represent the local state and local action

¹For the case of time-varying neighbor agents, our algorithm is still applicable if the agent communicates intermittently (or delays communication) with its initial neighbor. In the process of convergence analysis of the algorithm, we just need to introduce an additional error to (22) caused by communication disconnection or delay.

of agent $i \in \mathcal{N}$, respectively. For integer $\kappa \geq 1$, denote $s_{\mathcal{N}_i^\kappa}$ and $a_{\mathcal{N}_i^\kappa}$ as the state and action of agent i 's κ -hop neighbors, respectively. Denote $\mathcal{S}_{\mathcal{N}_i^\kappa} = \prod_{j \in \mathcal{N}_i^\kappa} \mathcal{S}_j$ and $\mathcal{A}_{\mathcal{N}_i^\kappa} = \prod_{j \in \mathcal{N}_i^\kappa} \mathcal{A}_j$ as the state space and the action space of agent i 's κ -hop neighbors, respectively. Moreover, denote $\mathcal{S}_{\mathcal{N}_{-i}^\kappa}$ and $\mathcal{A}_{\mathcal{N}_{-i}^\kappa}$ as the state space and the action space of agents excluding agent i 's κ -hop neighbors, respectively. $a_{\mathcal{N}_{-i}^\kappa} \in \mathcal{S}_{\mathcal{N}_{-i}^\kappa}$.

State transition probability function: $\mathcal{P}_i(s'_i | s_{\mathcal{N}_i^1}, a_i) : \mathcal{S}_{\mathcal{N}_i^1} \times \mathcal{A}_i \times \mathcal{S}_i \rightarrow [0, 1]$ is the state transition probability function of agent i , dependent of its 1-hop neighborhood state and its local action. Denote $\mathcal{P}(s' | s, a) = \prod_{i=1}^N \mathcal{P}_i(s'_i | s_{\mathcal{N}_i^1}, a_i) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ as the global state transition probability function. Note that the definition of the state transition probability function $\prod_{i=1}^N \mathcal{P}_i(s'_i | s_{\mathcal{N}_i^1}, a_i)$ is common in the literature. For example, it applies to the scenario of traffic signal control problem (Chu et al., 2020), where the traffic flow at each intersection is influenced by the traffic flow at its neighboring intersections and its own signal light.

Initial state distribution: ρ is the distribution of the initial state s_0 .

Reward function: $r_i^m(s_i, a_i) : \mathcal{S}_i \times \mathcal{A}_i \rightarrow \mathbb{R}$ is the reward function of agent $i \in \mathcal{N}$ in the objective $m \in \mathcal{M}$. Denote $\mathbf{s}_t = (s_{1,t}, \dots, s_{N,t})$ and $\mathbf{a}_t = (a_{1,t}, \dots, a_{N,t})$ as the global state and the global action at time t , respectively. The reward of agent $i \in \mathcal{N}$ in the objective $m \in \mathcal{M}$ at time t can be represented as $r_{i,t}^m = r_i^m(s_{i,t}, a_{i,t})$.

Discount factor: $\gamma = (\gamma^1, \dots, \gamma^M)^\top \in \mathbb{R}^M$ with $\gamma^m \in (0, 1)$ being the discount factor in the objective $m \in \mathcal{M}$.

Softmax policy: In this paper, we use the parameterized softmax policy $\pi_{\theta_i}(a_i | s_i)$ with parameter $\theta_i \in \mathbb{R}^{|\mathcal{S}_i| \times |\mathcal{A}_i|}$, which is described as

$$\pi_{\theta_i}(a_i | s_i) = \frac{\exp(\theta_{i,s_i,a_i})}{\sum_{a'_i} \exp(\theta_{i,s_i,a'_i})}, \quad (1)$$

where θ_{i,s_i,a_i} represents the element corresponding to (s_i, a_i) in θ_i . Denote $\boldsymbol{\theta} = (\theta_1^\top, \dots, \theta_N^\top)^\top \in \mathbb{R}^{\sum_{i=1}^N |\mathcal{S}_i| \times |\mathcal{A}_i|}$ as the joint policy parameter of agents and $\boldsymbol{\pi}_\theta(\mathbf{a} | \mathbf{s}) = \prod_{i=1}^N \pi_{\theta_i}(a_i | s_i)$ be the joint policy of all agents. Note that the softmax policy is used in RL to ensure the exploration of agents (Zhou et al., 2023; Zhang et al., 2022).

In the MOMARL problem, given a joint policy parameter $\boldsymbol{\theta}$, the m -th objective of all agents is defined as

$$J^m(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{s} \sim \rho} \left[\frac{1}{N} \sum_{t=0}^{\infty} \sum_{i=1}^N (\gamma^m)^t r_{i,t}^m | \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_t \sim \boldsymbol{\pi}_\theta(\cdot | \mathbf{s}_t) \right], \quad (2)$$

which represents the average discounted reward of all agents over all time t . The goal of agents in the MOMARL problem is to find a joint policy parameter $\boldsymbol{\theta}$ to maximize the following composite objective, i.e.,

$$\max_{\boldsymbol{\theta}} \mathbf{J}(\boldsymbol{\theta}) = [J^1(\boldsymbol{\theta}), \dots, J^M(\boldsymbol{\theta})]^\top \in \mathbb{R}^M. \quad (3)$$

In order to address the potential conflicts among the $\mathbf{J}(\boldsymbol{\theta})$ in (3), the notions of Pareto-optimality and ϵ -Pareto-stationarity are introduced as follows.

Definition 1 (Pareto-optimality) A solution $\boldsymbol{\theta}$ dominates solution $\boldsymbol{\theta}'$ if and only if $J^m(\boldsymbol{\theta}) \geq J^m(\boldsymbol{\theta}')$, $\forall m \in \mathcal{M}$ and $\exists m' \in \mathcal{M}$, $J^{m'}(\boldsymbol{\theta}) > J^{m'}(\boldsymbol{\theta}')$. A solution $\boldsymbol{\theta}$ is Pareto-optimal if it is not dominated by any other solution.

Considering that finding Pareto-optimal solutions for non-convex MOMARL problems is NP-hard, it is generally more practical to seek the ϵ -Pareto-stationary solution instead of the Pareto-optimal solution (Kumar et al., 2019).

Definition 2 (ϵ -Pareto-stationarity) Define $\nabla_{\boldsymbol{\theta}} \mathbf{J}(\boldsymbol{\theta})$ as the gradient of $\mathbf{J}(\boldsymbol{\theta})$ respect to $\boldsymbol{\theta}$. A solution $\boldsymbol{\theta}$ is ϵ -Pareto stationary if there exists $\boldsymbol{\lambda} = (\lambda^1, \dots, \lambda^M)^\top \in \mathbb{R}^M$ such that $\min_{\boldsymbol{\lambda} \in \mathbb{R}^M} \|\nabla_{\boldsymbol{\theta}} \mathbf{J}(\boldsymbol{\theta})^\top \boldsymbol{\lambda}\|_2^2 \leq \epsilon$ with $\boldsymbol{\lambda} \geq 0$, $\|\boldsymbol{\lambda}\|_1 = 1$, and $\epsilon > 0$.

Based on Definitions 1-2, it is obvious that the Pareto-stationarity is a necessary condition for a solution to be Pareto-optimal. Specifically, in the context of convex MOMARL problems, the solutions that are Pareto-stationary also qualify as Pareto-optimal. Given the complexity associated with the MOMARL problem, this paper focuses on developing a distributed scalable algorithm to identify and achieve Pareto-stationarity.

2.2 PRELIMINARIES IN THE MOMARL PROBLEM

In the MOMARL problem, for any joint policy parameter θ and $m \in \mathcal{M}$, the global Q -function $Q^m(s, \mathbf{a}; \theta)$ in m -th objective is defined as

$$Q^m(s, \mathbf{a}; \theta) = \mathbb{E}_{\pi_\theta} \left[\frac{1}{N} \sum_{t=0}^{\infty} \sum_{i=1}^N (\gamma^m)^t r_{i,t}^m | s_0 = s, \mathbf{a}_0 = \mathbf{a} \right], \quad (4)$$

which represents the value of the state-action pair $(s, \mathbf{a})$ in m -th objective under joint policy π_θ . In the MOMARL problem, given the joint policy parameter θ , define $d_\rho^{\theta, m}(s)$ as the discounted state visitation distribution, which is represented as

$$d_\rho^{\theta, m}(s) = (1 - \gamma^m) \sum_{t=0}^{\infty} (\gamma^m)^t \Pr^{\pi_\theta}(s_t = s | s_0 \sim \rho), \quad (5)$$

where $\Pr^{\pi_\theta}(s_t = s | s_0 \sim \rho)$ represents the probability of $s_t = s$ at time t under the initial state distribution ρ and the joint policy π_θ .

Recall that the policy gradient theorem (Sutton et al., 2000) is the foundation of algorithm design in RL. Inspired by the theorem, in our MOMARL problem, we also have the following policy gradient lemma.

Lemma 1 *In the MOMARL problem, for any joint policy parameter θ , the gradient of $J^m(\theta)$ in m -th objective with respect to θ is given by:*

$$\nabla_\theta J^m(\theta) = \frac{1}{1 - \gamma^m} \mathbb{E}_{s \sim d_\rho^{\theta, m}, \mathbf{a} \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(\mathbf{a} | s) Q^m(s, \mathbf{a}; \theta)], \forall m \in \mathcal{M}. \quad (6)$$

For the policy gradients involved in Definition 2, Lemma 1 shows that the calculation of the policy gradient $\nabla_\theta J^m(\theta)$ depends on $Q^m(s, \mathbf{a}; \theta)$, which involves global state-action $(s, \mathbf{a})$. Consequently, there are two challenges in applying (6): (i) the computational complexity of handling the global state-action $(s, \mathbf{a})$ in a centralized setting is high; (ii) achieving distributed decision making among multi-agents with limited communication.

3 DISTRIBUTED SCALABLE ACTOR-CRITIC ALGORITHM FOR MOMARL PROBLEM

Before presenting our proposed method, we first revisit the centralized approach for solving MOMARL problems. In the centralized setting, the policy update requires the global state-action pair $(s, \mathbf{a})$. This immediately leads to an exponential growth of the joint state-action space with the number of agents, i.e., of order $\mathcal{O}(|\mathcal{S}| |\mathcal{A}|^N)$. Such computational complexity makes centralized algorithms prohibitive for large-scale systems. Moreover, centralized training implicitly assumes that agents can constantly communicate with a central controller, which is unrealistic in many real-world scenarios (e.g., swarm robotics, sensor networks). To overcome these limitations, the natural way forward is to make the algorithm scalable, meaning that the per-agent computational cost should remain polynomial in the state-action dimension of its neighbors within a limited range rather than depending on the global scale. The most effective way to achieve scalability is to design a distributed algorithm, where (i) each agent i makes decisions independently using only $(s_{\mathcal{N}_i^\kappa}, a_i)$, where $s_{\mathcal{N}_i^\kappa}$ is the state from its κ -hop neighbors and (2) exchanges its local Lagrangian multiplier λ_i with its directly neighbors for balancing multiple objectives. This shift from centralized to distributed design is the key idea underlying our algorithm: by restricting each agent’s decision to local state-action information and coordinating objective trade-offs through a consensus protocol, we preserve scalability while maintaining rigorous convergence guarantees.

To reduce the reliance of algorithm on global information, we design a distributed algorithm, where each agent (i) estimates a local policy gradient based only on its own action, (ii) leverages a linear approximation restricted to κ -hop neighborhood states, and (iii) updates multi-objective weights through a consensus protocol to cooperatively approach Pareto-stationary solutions.

3.1 A NOVEL APPROXIMATED POLICY

In contrast to the global Q -function utilized in (6), which relies on global state-action information, we introduce a novel concept—the “action-averaged Q -function” for each agent i . This formulation leverages rewards from agent i ’s κ -hop neighbors to effectively reduce the dependence on the full joint action $\mathbf{a}$ by focusing on the local action a_i , as defined below:

$$\widehat{Q}_i^m(s, a_i; \theta) = \mathbb{E}_{\pi_\theta} \left[\frac{1}{N} \sum_{t=0}^{\infty} (\gamma^m)^t \sum_{j \in \mathcal{N}_i^\kappa} r_j^m(s_{j,t}, a_{j,t}) | s_0 = s, a_{i,0} = a_i \right]. \quad (7)$$

To approximate the true policy gradient $\nabla_{\theta} J^m(\theta)$ in (6), we define $\nabla_{\theta_i} J_{app}^m(\theta)$ as an approximated policy gradient for agent i , derived using the action-averaged Q -function presented in (7), as follows:

$$\nabla_{\theta_i} J_{app,i}^m(\theta) = \frac{1}{1 - \gamma^m} \mathbb{E}_{s \sim d_{\rho}^{\theta,m}, a_i \sim \pi_{\theta_i}} \left[\widehat{Q}_i^m(s, a_i; \theta) \nabla_{\theta_i} \log \pi_{\theta_i}(a_i | s_i) \right]. \quad (8)$$

Unlike the policy gradient in centralized algorithm that requires global action $\mathbf{a}$, (8) only requires the local action a_i of agent i . The approximation error between $\nabla_{\theta_i} J_{app,i}^m(\theta)$ and original $\nabla_{\theta_i} J^m(\theta)$ in (6) can be well bounded for the MOMARL problem in the following theorem.

Theorem 1 *In the MOMARL problem, given a joint policy π_{θ} , for any agent $i \in \mathcal{N}$ and objective $m \in \mathcal{M}$, it holds that*

$$\|\nabla_{\theta_i} J_{app,i}^m(\theta) - \nabla_{\theta_i} J^m(\theta)\|_2 \leq \frac{\sqrt{2}R}{(1 - \gamma^m)^2} (\gamma^m)^{\kappa+1}. \quad (9)$$

The proof of Theorem 1 is provided in Appendix A.2. The policy gradient has been approximated so far by constructing $\widehat{Q}_i^m(s, a_i; \theta)$ in (7) and $\nabla_{\theta_i} J_{app,i}^m(\theta)$ in (8), which reduces the action dimension of each agent i to its local action a_i . However, the expression of $\widehat{Q}_i^m(s, a_i; \theta)$ still requires the global state. Therefore, in the following, we will focus on reducing the dimensionality of agents' state information.

3.2 CRITIC STEP: LINEAR FUNCTION APPROXIMATION

Algorithm 1: Linear function approximation

- 1 **Require:** The number of samples K , the learning-rate η_w^m and $\varepsilon > 0$;
 - 2 **Initialization:** Initialize the ε -exploration policy $\pi_{\theta}^{\varepsilon} = \Pi_{i=1}^N \pi_{\theta_i}^{\varepsilon}$, where $\pi_{\theta_i}^{\varepsilon}(a_i | s_i) = (1 - \varepsilon)\pi_{\theta_i}(a_i | s_i) + \frac{\varepsilon}{|\mathcal{A}_i|}$ for all $i \in \mathcal{N}$. The initial values of the parameters $w_{i,0}^m$ is set as $w_{i,0}^m = \mathbf{0}_{d_i}$ for all $i \in \{1, 2, \dots, N\}$;
 - 3 The agents execute the ε -exploration policy $\pi_{\theta}^{\varepsilon}$ and each agent $i \in \mathcal{N}$ collects a sequence of samples $\{(s_{i,k}, a_{i,k}, r_{i,k}^m)\}_{0 \leq k \leq K}$ in m -the objective;
 - 4 **for** $i = 1, 2, \dots, N$ **do**
 - 5 **For** each objective $m \in \mathcal{M}$, agent $i \in \mathcal{N}$ collects the state information $\{s_j\}_{j \in \mathcal{N}_i^{\kappa}}$ of its κ -hop neighbors and reward $\{r_j^m\}_{j \in \mathcal{N}_i^{\kappa}}$ from its κ -hop neighbors to form a sample set $\{s_{\mathcal{N}_i^{\kappa},k}, a_{i,k}, r_{\mathcal{N}_i^{\kappa},k}^m\}_{0 \leq k \leq K}$;
 - 6 **for** $k = 0, 1, 2, \dots, K - 1$ **do**
 - 7 Each agent $i \in \mathcal{N}$ estimates its local temporal difference error:
$$\delta_{i,k}^m = \phi_i(s_{\mathcal{N}_i^{\kappa},k}, a_{i,k})^{\top} w_{i,k}^m - \frac{1}{N} \sum_{j \in \mathcal{N}_i^{\kappa}} r_{j,k}^m - \gamma^m \phi_i(s_{\mathcal{N}_i^{\kappa},k+1}, a_{i,k+1})^{\top} w_{i,k}^m;$$
 - 8 $w_{i,k+1}^m = w_{i,k}^m - \eta_w^m \delta_{i,k}^m \phi_i(s_{\mathcal{N}_i^{\kappa},k+1}, a_{i,k+1})$;
 - 9 **end**
 - 10 **end**
 - 11 **Output:** $\{w_{i,K}^m\}_{i \in \mathcal{N}, m \in \mathcal{M}}$
-

In this subsection, we use the localized stochastic approximation and propose a linear function in (10) to reduce the dimension of the state-action required by agent $i \in \mathcal{N}$ to $(s_{\mathcal{N}_i^{\kappa}}, a_i)$. Specially, the linear function $\hat{Q}_i^m(s_{\mathcal{N}_i^{\kappa}}, a_i; w_i^m)$ of agent i to approximate $\widehat{Q}_i^m(s, a_i; \theta)$ in (7) is given as

$$\hat{Q}_i^m(s_{\mathcal{N}_i^{\kappa}}, a_i; w_i^m) = \phi_i(s_{\mathcal{N}_i^{\kappa}}, a_i)^{\top} w_i^m, \quad (10)$$

where $\phi_i(s_{\mathcal{N}_i^{\kappa}}, a_i) : \mathcal{S}_{\mathcal{N}_i^{\kappa}} \times \mathcal{A}_i \rightarrow \mathbb{R}^{d_i}$ is the feature vector mapping and $w_i^m \in \mathbb{R}^{d_i}$ is the parameter of agent i in m -th objective. By the definition of $\widehat{Q}_i^m(s, a_i; \theta)$ in (7), the parameter with initial value $w_{i,0}^m$ can be updated by sample sequence $\{s_{\mathcal{N}_i^{\kappa},k}, a_{i,k}, r_{\mathcal{N}_i^{\kappa},k}^m\}_{0 \leq k \leq K}$ as

$$w_{i,k+1}^m = w_{i,k}^m - \eta_w^m \delta_{i,k}^m \phi_i(s_{\mathcal{N}_i^{\kappa},k+1}, a_{i,k+1}), \quad (11)$$

where K is the number of sample for linear parameter training and $\delta_{i,k}^m$ is the local temporal difference error, which is described as

$$\delta_{i,k}^m = \phi_i(s_{\mathcal{N}_i^{\kappa},k}, a_{i,k})^{\top} w_{i,k}^m - \frac{1}{N} \sum_{j \in \mathcal{N}_i^{\kappa}} r_{j,k}^m - \gamma^m \phi_i(s_{\mathcal{N}_i^{\kappa},k+1}, a_{i,k+1})^{\top} w_{i,k}^m, \quad (12)$$

and η_w^m is the fixed learning rate of parameters w_i^m . The detailed description of linear function approximation is given in Algorithm 1.

3.3 ACTOR STEP: POLICY PARAMETER UPDATE

To estimate the policy gradient and ensure convergence toward a Pareto-stationary solution, we need to dynamically adjust the weights associated with multiple objectives. In centralized algorithms, this requires access to all agents' policy gradients, which is often infeasible and unscalable (Zhou et al., 2024). To overcome this limitation, we design a distributed algorithm in which agents collaboratively adjust the objective weights through a consensus protocol. This design removes the dependence on global policy gradient while still enabling cooperative optimization toward Pareto-stationary solution.

To estimate the approximated policy gradient $\nabla_{\theta_i} J_{app,i}^m(\theta)$ in (8), for joint policy π_{θ_t} , our estimate $g_{i,t}^m(B)$ is calculated iteratively based on the sample sequence $\{(s_{\mathcal{N}_i^c,h}^b, a_{i,h}^b)\}_{0 \leq b \leq B-1, 0 \leq h \leq H-1}$:

$$g_{i,t}^m(b+1) = \frac{b}{b+1} g_{i,t}^m(b) + \frac{1}{b+1} \hat{\nabla}_{\theta_i} J_{app,i}^{m,b}(\theta_t), \quad (13)$$

where $g_{i,t}^m(0) = \mathbf{0}_{|\mathcal{S}_i||\mathcal{A}_i|}$ and

$$\hat{\nabla}_{\theta_i} J_{app,i}^{m,b}(\theta_t) = \sum_{h=0}^{H-1} (\gamma^m)^h \phi_i(s_{\mathcal{N}_i^c,h}^b, a_{i,h}^b)^\top w_i^m(t) \nabla_{\theta_i} \log \pi_{\theta_{i,t}}(a_{i,h}^b | s_{i,h}^b) \quad (14)$$

with $w_i^m(t)$ being the output of Algorithm 1 in t -th iteration of policy parameters. Let $g_{i,t}^m = g_{i,t}^m(B)^\top$ and $\mathbf{g}_t^m = ((g_{1,t}^m)^\top, \dots, (g_{N,t}^m)^\top)^\top \in \mathbb{R}^{\sum_{i=1}^N |\mathcal{S}_i||\mathcal{A}_i|}$. Following Pareto-stationarity in Definition 1, we denote $\lambda_t^* = (\lambda_t^{*1}, \dots, \lambda_t^{*M})^\top \in \mathbb{R}^M$ as solution of the following quadratic programming problem:

$$\begin{aligned} \min_{\lambda_t = (\lambda_t^1, \dots, \lambda_t^M)^\top \in \mathbb{R}^M} J_t^g(\lambda_t) &= \left\| \sum_{m=1}^M \lambda_t^m \mathbf{g}_t^m \right\|_2^2 \\ \text{s.t. } \lambda_t &\geq 0, \|\lambda_t\|_1 = 1. \end{aligned} \quad (15)$$

For the network $\mathcal{G}(\mathcal{N}, \mathcal{E})$ among agents, we define its weight matrix as $W^{\mathcal{G}} = [w_{ij}^{\mathcal{G}}]_{N \times N}$, where each element $w_{ij}^{\mathcal{G}}$ represents the weight of the edge from agent j to agent i , which is defined as

$$w_{ij}^{\mathcal{G}} = \begin{cases} \frac{1}{1 + \max(|\mathcal{N}_i|, |\mathcal{N}_j|)}, & j \in \mathcal{N}_i, \\ 1 - \sum_{l \in \mathcal{N}_i} w_{il}^{\mathcal{G}}, & j = i, \\ 0, & \text{otherwise.} \end{cases} \quad (16)$$

By using the definition of $W^{\mathcal{G}}$ in (16), we solve the problem (15) in a distributed way, which is presented in the following Algorithm 2.

Algorithm 2: Distributed computation to solve problem (15) of objective weight adjustment

```

1 Initialization: Each agent  $i \in \mathcal{N}$  sets  $\lambda_i(0) = \frac{1}{M} \mathbf{1}_M$  and chooses step sizes  $\alpha_k = \frac{2}{k+2}$  for all  $k \geq 0$ ;
2 for  $k = 0, 1, 2, \dots, K_\lambda - 1$  do
3   Each agent  $i$  initializes  $x_{i,t}(k) = \sum_{m=1}^M \lambda_{i,m}(k) g_{i,t}^m$ ;
4   for  $m = 1, 2, \dots, M$  do
5     Each agent  $i$  initializes  $y_{i,t}^m(k, 0) = \langle x_{i,t}(k), g_{i,t}^m \rangle$ ;
6     while  $\exists i \in \mathcal{N}, y_{i,t}^m(l_1) \neq \frac{1}{N} \sum_{i=1}^N y_{i,t}^m(k, 0)$  do
7        $y_{i,t}^m(k, l_1 + 1) = \sum_{j \in \mathcal{N}_i} w_{ij}^{\mathcal{G}} y_{j,t}^m(k, l_1)$ ;
8        $l_1 \leftarrow l_1 + 1$ ;
9     end
10    Each agent  $i$  obtains local output  $y_{i,t}^m(k)$ ;
11  end
12  Each agent obtains  $u_{i,t}^*(k) = \arg \min_m y_{i,t}^m(k)$ ;
13  Each agent updates  $\lambda_i(k+1) = (1 - \alpha_k) \lambda_i(k) + \alpha_k e_{u_{i,t}^*(k)}$ , where  $e_{u_{i,t}^*(k)}$  is an  $M$ -dimensional unit
    vector with the  $u_{i,t}^*(k)$ -th element being 1, and the other elements being 0;
14 end
15 Output:  $\hat{\lambda}_t = \lambda_i(K_\lambda)$  for all  $i \in \mathcal{N}$ ;

```

In Algorithm 2 each agent iteratively computes policy gradients, engages in consensus steps to evaluate the quadratic objective (ref. Lines 5-9), and updates its local weight vector via a Frank-Wolfe update rule (ref. Lines 12-13). Through repeated consensus and update steps, all agents asymptot-

ically agree on the optimal weight vector $\hat{\lambda}_t$, thereby achieving a distributed solution that approximates the centralized Pareto-stationary weighting without requiring global information exchange.

After computing $\hat{\lambda}_t$ by Algorithm 2, we update the weight λ_t as

$$\lambda_t = (1 - \eta_{\lambda,t})\lambda_{t-1} + \eta_{\lambda,t}\hat{\lambda}_t, \quad (17)$$

where $\eta_{\lambda,t}$ is the learning rate of λ_t . Denote $g_t = \sum_{m=1}^M \lambda_t^m g_t^m$, the update of θ_{t+1} is presented as

$$\theta_{t+1} = \theta_t + \eta_{\theta,t} g_t, \quad (18)$$

where $\eta_{\theta,t}$ is the learning rate of policy parameter. In the NMARL problem, the agents can use θ_t to execute the actions based on (1).

3.4 OVERALL DISTRIBUTED ALGORITHM FOR MOMARL PROBLEM

Based on the distributed designs in the previous three subsections, we propose a distributed MOMARL algorithm, which is given in Algorithm 3.

Algorithm 3: Distributed algorithm for MOMARL problem

- 1 **Require:** The non-negative integers T, B, H , the learning-rates $\{\eta_{\lambda,t}\}_{t \in \{1, \dots, T\}}$ and $\{\eta_{\theta,t}\}_{t \in \{1, \dots, T\}}$;
 - 2 **Initialization:** Initialize $\lambda_0 = \frac{1}{M} \mathbf{1}_M \in \mathbb{R}^M$, the policy parameter $\theta_{i,1} \in \mathbb{R}^{|S_i| \times |A_i|}$ to follow Gaussian distribution for all $i \in \{1, 2, \dots, N\}$;
 - 3 **for** $t = 1, 2, \dots, T$ **do**
 - 4 Initial policy gradient estimation $g_{i,t}^m(0) = \mathbf{0}_{|S_i| \times |A_i|}$ for all $i \in \mathcal{N}$;
 - 5 **Critic step:** All agents use Algorithm 1 and output the weight vectors $\{w_i^m(t)\}_{i \in \mathcal{N}}$;
 - 6 **Actor step:**
 - 7 **for** $b = 0, 1, 2, \dots, B - 1$ **do**
 - 8 All agents execute the joint policy π_{θ_t} in $H - 1$ horizon;
 - 9 Each agent $i \in \mathcal{N}$ collects a sequence of samples, which includes the state information $\{s_j\}_{j \in \mathcal{N}_i^\kappa}$ from its κ -hop neighbors and its local action information a_i , i.e., $\{(s_{\mathcal{N}_i^\kappa, h}^b, a_{i,h}^b)\}_{0 \leq h \leq H-1}$;
 - 10 Each agent i estimates the local policy gradient in m -th objective according to (13);
 - 11 **end**
 - 12 All agents calculate $g_{i,t}^m = g_{i,t}^m(B)$ by (13) and achieve $g_t^m = ((g_{1,t}^m)^\top, \dots, (g_{N,t}^m)^\top)^\top$ for all $m \in [M]$;
 - 13 Compute $\hat{\lambda}_t$ by Algorithm 2 as the approximation solution to problem (15);
 - 14 Update the weight λ_t according to (17);
 - 15 Update the policy parameter θ_{t+1} according to (18);
 - 16 **end**
 - 17 **Output:** $\pi_{\theta_{\hat{T}}}$ with $\hat{T}$ chosen uniformly from $\{1, \dots, T\}$
-

Algorithm 3 incorporates linear function approximation in Algorithm 1 and distributed consensus-based adjustment of multiplier objective weights in Algorithm 2 into a unified framework. The primary advantages are as follows: (1) each agent relies solely on state information from its κ -hop neighbors, thereby avoiding the exponential expansion of the centralized joint state-action space (ref. Line 9); (2) Each agent in Line 13 only uses the policy gradient estimations of its direct neighbors, which eliminates the requirement for a central coordinator while supporting collaborative multi-objective optimization.

4 PARETO-STATIONARY CONVERGENCE OF ALGORITHM 3

Before the convergence analysis of the algorithm, some assumptions are introduced in the following.

Assumption 1 In the MOMARL problem, for any joint policy parameter θ and objective $m \in \mathcal{M}$, $\xi_{\rho}^{\theta,m}(s, a)$ satisfies that

$$\inf_{\theta} \min_{(s,a) \in \mathcal{S} \times \mathcal{A}} \xi_{\rho}^{\theta,m}(s, a) > 0. \quad (19)$$

Assumption 2 In the MOMARL problem, for any agent $i \in \mathcal{N}$ and objective $m \in \mathcal{M}$, there exists constant $R > 1$ such that the instantaneous reward $r_{i,t}^m$ at time $t \geq 0$ satisfies $|r_{i,t}^m| \leq R$.

Assumption 3 In the MOMARL problem, the network $\mathcal{G}(\mathcal{N}, \mathcal{E})$ among agents is connected graph.

Assumption 1 ensures that for any joint policy π_{θ} , $(s, a) \in \mathcal{S} \times \mathcal{A}$ is visited with a non-zero probability, Assumption 2 provides an upper bound on the reward, and Assumption 3 is provided for designing distributed algorithm. Assumptions 1-2 are standard prerequisite for the convergence

analysis of RL algorithms (e.g. (Zhou et al., 2023; Zhang et al., 2022)) and Assumption 3 is commonly in (Olfati-Saber & Murray, 2004).

Our process to prove the Pareto-stationary convergence of Algorithm 3 is as follows: (i) We start establish the smoothness of objective function (i.e., 2); (ii) We from the definition of Pareto-stationarity in Definition 2 and analyze the error between the true gradient $\nabla_{\theta_i} J^m(\theta_t)$ and the calculated gradient $g_{i,t}^m$ in (13) (i.e., Lemma 3); (iii) We control λ_t by setting the step size $\eta_{\theta,t}$ to ensure that Algorithm 3 converges to Pareto-stationary solution in Theorem 2.

Lemma 2 *In the MOMARL problem, define $L_J = \max_{m \in \mathcal{M}} \frac{6N}{(1-\gamma^m)^3}$. For any objective $m \in \mathcal{M}$, the objective $J^m(\theta)$ is L_J -smooth.*

The detailed proof of Lemma 2 can be found in Appendix A.3. Lemma 2 establishes that each individual objective in the MOMARL problem is L_J -smooth with respect to the joint policy parameters. This smoothness property implies that the gradient of $J^m(\theta)$ does not change abruptly when the policy parameters are slightly perturbed, which is crucial for analyzing the stability and convergence of gradient-based algorithms. In particular, L_J -smoothness allows us to control the error propagation when using approximate gradients or local updates in a distributed setting, and it forms a key technical result for deriving convergence rates toward Pareto-stationary solutions.

Define ε_{critic} below as the linear approximation error in Algorithm 1:

$$\varepsilon_{critic} = \sup_{m \in \mathcal{M}} \sup_{\theta} \sup_{i \in \mathcal{N}} \mathbb{E} \left[\sup_{s, a_i} \left| \hat{Q}_i(s_{\mathcal{N}_i^c}, a_i; w_{i,K}^m) - \widehat{Q}_i^m(s, a_i; \theta) \right|^2 \right]. \quad (20)$$

Based on ε_{critic} , we further define the gradient approximation error in Section 3.3 as

$$\varepsilon_{actor}^m = \underbrace{\frac{8R^2}{(1-\gamma^m)^4} (\gamma^m)^{2\kappa+2}}_{\text{Truncation error}} + \underbrace{\frac{32}{(1-\gamma^m)^2 B}}_{\text{Sampling error}} + \underbrace{\frac{8(\gamma^m)^{2H}}{(1-\gamma^m)^4}}_{\text{Linear approximation error}} + \underbrace{\frac{8\varepsilon_{critic}}{(1-\gamma^m)^2}}_{\text{Linear approximation error}}. \quad (21)$$

Lemma 3 *In Algorithm 3, for joint policy parameter θ_t , any agent $i \in \mathcal{N}$, and objective $m \in \mathcal{M}$, we have*

$$\mathbb{E}[\|\nabla_{\theta_i} J^m(\theta_t) - g_{i,t}^m\|_2^2] \leq \varepsilon_{actor}^m.$$

The proof of the Lemma 3 is given in Appendix A.4. Lemma 3 provides an upper bound on the error between the true policy gradient and the estimated gradient $g_{i,t}^m$ used by each agent in Algorithm 3. This result quantifies the approximation error induced by the use of a critic or a finite-horizon estimator in computing policy gradients. Importantly, the bound ε_{actor}^m captures the combined effects of truncation, sampling, and linear function approximation errors. Based on Lemma 3, the convergence of Algorithm 2 is established in the following theorem.

Proposition 1 *In Algorithm 2, for any iteration t , we define $G_t = (g_t^1, \dots, g_t^m) \in \mathbb{R}^{M \times \sum_{i=1}^N |S_i| |A_i|}$ and have the following result:*

- (i) *In Line 10 of Algorithm 2, $y_{i,t}^m(k) = \frac{1}{N} \sum_{i=1}^N \langle \sum_{m=1}^M \lambda_{i,m}(k) g_{i,t}^m, g_{i,t}^m \rangle$;*
- (ii) *In Problem 15, $J_t^g(\cdot)$ is L_t^g -Lipschitz continuous with $L_t^g = 2\sigma_{max}^2(G_t)$, where $\sigma_{max}(G_t)$ is the largest singular value of G_t ;*
- (iii) *$|J_t^g(\hat{\lambda}_t) - J_t^g(\lambda_t^*)| \leq \frac{4L_t^g}{K_\lambda + 1}$.*

The proof of Proposition 1 is presented in Appendix A.5. Proposition 1 guarantees that, despite the distributed nature of the update and the use of only local gradient information from neighbors, the agents can collectively achieve a close approximation to the globally optimal weight allocation for combining multiple objectives. This property is fundamental for ensuring the cooperative convergence of the distributed algorithm toward a Pareto-stationary solution. Based on Proposition 1, the Pareto-stationary convergence of Algorithm 3 is presented in the following theorem.

Theorem 2 *In Algorithm 3, let $\eta_{\theta,t} = \frac{1}{3L_J}$, and $\eta_{\lambda,t} = \frac{1}{(t+1)^2}$. Our policy parameter sequences $\{\theta_t\}_{t=1}^T$ generated by Algorithm 3 satisfies:*

$$\begin{aligned} \frac{1}{T} \sum_{t=1}^T \mathbb{E}[\|\nabla_{\theta} J(\theta_t)^\top \hat{\lambda}_t\|_2^2] &\leq \frac{216N}{(1-\|\gamma\|_\infty)^3 T} \left(1 + \sum_{t=1}^T \eta_{\lambda,t} \right) + 5 \max_{m \in \mathcal{M}} \varepsilon_{actor}^m \\ &\quad + \frac{8}{K_\lambda + 1} \left(\max_{m \in \mathcal{M}} (\varepsilon_{actor}^m)^2 + \max_{m \in \mathcal{M}} \frac{2R^2}{(1-\gamma^m)^4} \right). \end{aligned} \quad (22)$$

The proof of Theorem 2 can be found in Appendix A.6. Theorem 2 shows that Algorithm 3 can converge to an approximate Pareto-stationary solution at a rate of $\mathcal{O}(1/T)$ with some approximate error terms. These errors are not significant, as we can control the upper bound of their upper bounds by setting the feature vector in the linear approximation, sample batch B , sample size H for policy gradient approximation, and the K_λ for the calculation of objective weights.

5 SIMULATION EXPERIMENTS

In this section, we employ a path planning for multiple robotics to travel environment analogous to the one described in (Zhou et al., 2023). While (Zhou et al., 2023) examines a road network comprising 8 nodes, our simulation focuses on a larger network containing 18 nodes. Specifically, the path planning problems of N robots (i.e., agents) on a typical acyclic path network in Fig. 1, where the “blue” nodes in $\{b_1, b_2, \dots, b_5\}$ represent the set of starting nodes for agents. “purple” node, “orange” node, and “green” node in the right-hand-side represent the destination nodes of objective 1, objective 2, and objective 3, respectively. In the path planning problem, the local state space $\mathcal{S}_i$ of agent $i \in \mathcal{N}$ is defined as $\mathcal{S}_i = \{b_1, b_2, b_3, b_4, b_5, c_1, c_2, c_3, c_4, c_5, d_1, d_2, d_3, d_4, d_5, e_1, e_2, e_3\}$, and the local action space of agent $i \in \mathcal{N}$ is defined as $\mathcal{A}_i = \{0, 1, 2, 3\}$ with 3 being the maximum out degree of nodes in path network

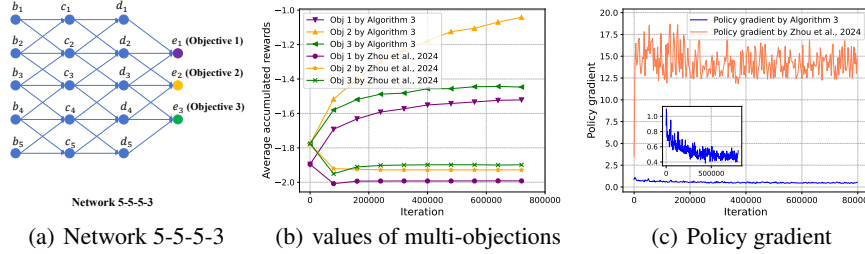


Figure 1: (a) Acyclic network. (b,c) The evolution of the objective performance $J(\theta_t)$ and the norm of policy gradient $\|g_t\|_2$ of the policy sequence generated by Algorithm 3, respectively.

In Network 5-5-5-3, consider agent i at node b_2 for illustration, the action “0” means that the agent i remains stationary at the current node for one time step, while “1”, “2”, and “3” indicate that the agent i follows the edge (b_2, c_1) , edge (b_2, c_2) , and edge (b_2, c_3) , respectively. It should be noted that the agent i remains at the current node even when the action selected by it exceeds the out degree of the current node. For example, if agent $j \in \mathcal{N}$ selects the action “3” at node b_1 , then it will remain stationary at b_1 for one time step.

The reward settings of each agent include: (i) the time run cost -0.5 at each step; (ii) the collision penalty when the agents share a path to move; (iii) the additional rewards for achieving different objectives. Specifically, when an agent reaches objective 1, objective 2, and objective 3, it receives the additional rewards of 0.5, 1.5, and 1, respectively. In this path planning problem, the agents want to efficiently reach the destinations while avoiding collision. The objective of agents is to find a joint policy parameter θ to maximizes (3).

Our robot path planning problem includes 10 agents, whose initial positions are set to $\{b_1, b_2, b_3, b_4, b_5, b_1, b_2, b_3, b_4, b_5\}$. In this simulation, both the proposed Algorithm 3 and the latest MORL centralized algorithm from (Zhou et al., 2024) are tested under different random seeds. The discounted average cumulative reward $\{J^m(\theta_t)\}_{m \in \{1,2,3\}}$ of the policy sequence generated by Algorithm 3 and the centralized algorithm are depicted in Fig. 1(b), where Algorithm 3 outperforms the centralized algorithm on performance of each objective. The main reason is that the centralized algorithm applies linear approximation to the global state-action space, causing large errors, whereas our distributed algorithm only approximates local Q -functions, leading to higher accuracy.

Moreover, the norm of policy gradients (i.e., $\|g_t\|_2$) generated by Algorithm 3 and the centralized algorithm are showed in Fig. 1(c). The norm of the policy gradient in Algorithm 3 exhibits a fast convergence trend towards to 0. However, the policy gradient in the centralized algorithm does not converge but fluctuates over many iterations from 0 due to the excessively large state-action dimension, resulting in a substantial approximation error in the linear approximation.

6 CONCLUSIONS

We propose a distributed algorithm for MOMARL and prove its convergence to a close-to-Pareto-stationary point. Each agent only requires state-action information $(s_{\mathcal{N}_i^c}, a_i)$, ensuring scalability. This framework itself is a significant contribution and may inspire other scalable RL methods in networked systems

REFERENCES

- Rick Zhang and Marco Pavone. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research*, 35(1-3): 186-203, 2016.
- Marina Weck, Eric Blake Jackson, Markus Sihvonen, and Ingrid Pappel. Building smart living environments for ageing societies: Decision support for cross-border e-services between Estonia and Finland. *Technology in Society*, 71: 102066, 2022.
- Jiayi Xu, Mario Di Nardo, and Shi Yin. Improved swarm intelligence-based logistics distribution optimizer: decision support for multimodal transportation of cross-border e-commerce. *Mathematics*, 12(5): 763, 2024.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
- Ivo Grondman, Lucian Buşoniu, Gabriel A. D. Lopes, and Robert Babuška. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, 42(6): 1291-1307, 2012.
- Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, 321-384, 2021.
- Yingqiang Ge, Xiaoting Zhao, Lucia Yu, Saurabh Paul, Diane Hu, Chu-Cheng Hsieh, and Yongfeng Zhang. Toward pareto efficient fairness-utility tradeoff in recommendation through reinforcement learning. In *Proceedings of the ACM international conference on web search and data mining*, ppages 316-324, 2022.
- Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. Choosing the best of both worlds: Diverse and novel recommendations through multi-objective reinforcement learning. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 957-965, 2022.
- Maude J. Blondin and Matthew Hale. An algorithm for multi-objective multi-agent optimization. In *Proceedings of the IEEE American Control Conference*, pages 1489-1494, 2020.
- Haibo Yang, Zhuqing Liu, Jia Liu, Chaosheng Dong, Michinari Momma. Federated multi-objective learning. In *Advances in Neural Information Processing Systems*, 36, 2024.
- Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, 31, 2018.
- Xu Chen, Yali Du, Long Xia, and Jun Wang. Reinforcement recommendation with user multi-aspect preference. In *Proceedings of the Web Conference*, pages 425-435, 2021.
- David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the International Conference on Machine Learning*, pages 387-395, 2014.
- Tianchen Zhou, FNU Hairi, Haibo Yang, Jia Liu, Tian Tong, Fan Yang, Michinari Momma, and Yan Gao. Finite-time convergence and sample complexity of actor-critic multi-objective reinforcement learning. *arXiv preprint*, arXiv: 2405.03082, 2024.
- Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathématique*, 350(5-6): 313-318, 2012.
- Roxana Rădulescu, Patrick Mannion, Diederik M. Roijers, and Ann Nowé. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems*, 34(1): 10, 2020.
- Yiheng Lin, Guannan Qu, Longbo Huang, and Adam Wierman. Multi-agent reinforcement learning in stochastic networked systems. In *Advances in Neural Information Processing Systems*, pages 7825-7837, 2021.
- Zhaoyi Zhou, Zaiwei Chen, Yiheng Lin, and Adam Wierman. Convergence rates for localized actor-critic in networked markov potential games. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 2563-2573, 2023.

- Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3): 1086-1095, 2020.
- Pengcheng Dai, Wenwu Yu, He Wang, and Jiahui Jiang. Applications in traffic signal control: a distributed policy gradient decomposition algorithm. *IEEE Transactions on Industrial Informatics*, 20(2): 2762-2775, 2024.
- Runyu Zhang, Jincheng Mei, Bo Dai, Dale Schuurmans, and Na Li. On the global convergence rates of decentralized softmax gradient play in markov potential games. In *Advances in Neural Information Processing Systems*, pages 1923-1935, 2022.
- Harshat Kumar, Alec Koppel, and Alejandro Ribeiro. On the sample complexity of actor-critic for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.
- Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057-1063, 2000.
- Guannan Qu, Adam Wierman, and Na Li. Scalable reinforcement learning of localized policies for multi-agent networked systems. In *Proceedings of the Conference on Learning for Dynamics and Control*, pages 256-266, 2020.
- Guannan Qu, Yiheng Lin, Adam Wierman, and Na Li. Scalable Multi-Agent Reinforcement Learning for Networked Systems with Average Reward. In *Advances in Neural Information Processing Systems*, pages 2074-2086, 2020.
- John Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5): 674-690, 1997.
- R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520-1533, Sep. 2004.