PIPO: Policy Optimization with Permutation-Invariant Graph Network for Decentralized Multi-Agent Navigation

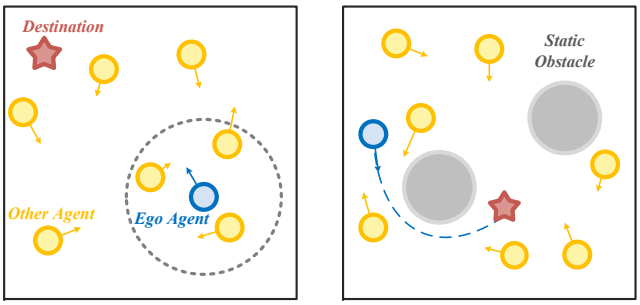
Ruiqi Zhang[1], Guang Chen[1,2*], Jing Hou[1], and Zhijun Li[3]

***Abstract*— Guaranteeing the safety and efficiency of navigation in complex scenarios is a challenging task for large-scale multi-agent systems (MAS). Most existing centralized reinforcement learning methods invalidate when the number of agents changes, and high latency and computational consumption hinder the application of leading decentralized methods. In this paper, we propose a novel algorithm for the navigation policy optimization of decentralized MAS with permutation-invariant graph networks named PIPO. We first set a guide-policy to conduct the navigation and skip the inefficient exploration in the initial epochs. Then we discuss the permutation invariant property in decentralized multi-agent systems and introduce the graph convolution network to generate identical output under the shuffled observations. Due to the decentralized training and execution, our approach can be deployed to large-scale systems and readily generalize to an arbitrary number of agents. We provide a series of experiments to demonstrate that our PIPO significantly outperforms the baselines of multi-agent reinforcement learning algorithms and other leading methods in variant scenarios.**

I. Introduction

Multi-agent navigation is a promising issue and has wide real-world applications such as cargo transportation, traffic scheduling, and autonomous driving [1]. However, safe navigation from the initial location to the destination is always challenging due to the high density of agents, partial observation, high-dimension state space, nonlinear dynamics, etc. Moreover, in a multi-agent system (MAS), we take the overall system efficiency and security as the main criterion rather than consider the completion of single-agent tasks as the primary. In other words, for each individual, a reasonable policy is required in some cases to make altruistic behavior in the shared environment to maximize total returns, rather than just greedily obtaining private rewards.

Although the emergence of Deep Learning (DL) and Reinforcement Learning (RL) simplify the large-scale control systems, few studies consider both the safety and efficiency of the navigation for high-density MAS, which is important to the applications in the real world. Besides, there are many urgent problems with existing methods. For instance, the value estimation based on the global states or observation set in classical multi-agent RL [2], [3] undermines the scalability of their algorithms. When the number of system agents changes, the network usually needs to be retrained, which increases the computational consumption. Concurrently, researchers point out the exponential dependence on the number of agents for centralized multi-agent RL [4]. Furthermore, RL methods commonly process their observations with multi-layer perceptron (MLPs), which causes the policy to over-fit in specific scenarios and outputs variant actions under the same observation with different representations. Besides, the convergence to a sub-optimal equilibrium state for decentralized methods always hampers the exploration of agents [5].



(a) Particle Navigation in MPEs (b) Obstacle Avoiding

Fig. 1. The task of multi-agent navigation with partial observations in decentralized multi-agent particle environments (MPEs). Sub-figure (a) shows that each agent is distributed with a random target. Only when other agents move into the field-of-view can the ego agent gain their states in decentralized MPEs. Sub-figure (b) demonstrates that the ego agent needs to avoid other moving agents while rounding the static obstacles.

To solve the existing problems, we propose a permutation-invariant graph-based Policy Optimization (PIPO) algorithm for safe multi-agent navigation. First, we design a branched network structure based on graph convolutional blocks [6] for proximal policy optimization (PPO) [7] algorithm with the Lagrangian constraints. Then we leverage a guide policy based on the improved artificial potential field method to accelerate the convergence of networks, which can practically avoid the efficient stochastic exploration in the initial epochs and provide reliable guidance for the high-density MAS. The extensive experimental results show that PIPO can stably outperform previous leading methods. In general, our contribution is three-fold:

- We propose a novel PIPO algorithm for high-density multi-agent navigation tasks by learning the residual part beyond the guidance of improved artificial potential field. The guide policy effectively accelerates the convergence and skip the inefficient sampling during initial exploration.
- We present the branched permutation-invariant network to solve the sensitivity to the input of classic MLPs-based methods. By segmentation and shuffling the observation from adjacent agents, the representation ability

*Correspondence to Guang Chen, Email: guangchen@tongji.edu.cn
[1]Tongji University, Shanghai 200092, China
[2]Technical University of Munich, Munich 80333, Germany
[3]University of Science and Technology of China, Hefei 230026, China

and robustness of networks is enhanced.
- We provide extensive experiments to illustrate that our method outperforms former multi-agent reinforcement learning algorithms and other leading methods, which can be scaled to an arbitrary number of agents.

II. Related Works

In the previous works, researchers define the multi-agent navigation task from the perspective of artificial intelligence and robotic control. We generally divide them by their settings and methods.

For classic discrete methods, the time, world, and action spaces are discretized into time-steps, grids, and optional actions, respectively. Based on these, agents first plan their paths to the targets and can only occupy the grids, waiting or moving to the adjacent grids at each time-step [8], [9]. However, the discretized methods can hardly be deployed into real scenarios with continuous action space, thus we skip the discussion about them but acknowledge their importance.

For multi-agent control in continuous space, one significant approach is path optimization with trajectory prediction. To guarantee security, researchers contribute their ideas for collision avoidance like the priority-based search with possible swept area in S2M2 [10], or standard gradient optimization and symbolic inference through geometric and algebraic features in [11]. Another solution is setting the control shield like the inequality constraint with Gauss principle in [12] or the control barrier function (CBF) for agents to keep system states in the safety set. In the MDBC [13], authors discuss a scalable CBF-based approach under the limited actuation, which provides a backup policy set and selects a trajectory satisfying the safety constraint. Nevertheless, it is difficult for them to extend to large systems for the high computational consumption and complexity of their algorithms.

Compared with classic control approaches, learning-based methods are always more flexible and extendable. As the extension of CBF-based methods, [14] introduce the decentralized neural network to learn the CBF from demonstration and successfully deploy it to the large scale MAS. Instead of system safety, other leading decentralized algorithms [4], [5], [15] focus on optimization of global optimal, which is established on massive constrained assumptions and can hardly adapt to real-world applications. Currently, another mainstream of this task is multi-agent reinforcement learning (MARL), which collects experience from interaction with the environment and optimizes independent policies for agents. As the classic paradigms, [2] provide a series of challenging multi-agent particle environments (MPEs) and MADDPG algorithm and are improved by [3] through centralized training and decentralized execution. However, no or few safety guarantees and efficiency contributions are involved in these works. In the work of permutation-invariant critic (PIC), [16] firstly proposed a centralized MARL algorithm with a graph neural network, while PIC can hardly guarantee the system safety. Independent policy optimization method extends RL algorithms for single agent to the MAS [17]. Although many works have proved this optimization method can handle the robot navigation complex transportation system [18], independent policy is uneconomic especially for the large MAS with robots in the same type.

III. Preliminaries

In this section, we will give out the crucial description and definition of multi-agent navigation task, and the experimental criterion of this paper.

A. The system and agent model

Multi-agent navigation requires the agents to reach their target rapidly while guaranteeing robotic safety. Each agent is equipped with a LiDAR and visual odometry. In this paper, we adopt the improved multi-agent particle environment (MPE) [2] to simulate the multi-agent system, where the signal of sensors is processed, so we do not discuss how to extract the features from the signal but study how to realize the safe and efficient navigation through it. Compared with the original MPE and former work [19], we complement the pile and square in the maps to present the common obstacles in the structured scenarios. The robots are of the same size and have double integral dynamics. Each agent and its target are randomly initialized and its global position is unknown. They are unconnected until one of them enters the field-of-view of the other one.

B. Constrained Markov Decision Process

We define the whole trajectory as a constrained Markov Decision Process (CMDP) [20] with partial observation. The CMDP can be presented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{C}, \mathcal{R}, \gamma)$, where Ω and $\mathcal{R}$ are respectively the observation and reward set. $\mathcal{O}$, $\mathcal{T}$ are respectively the conditional probability of observation and the state transition function. For the individual in the MAS, $\mathcal{S} \in \mathbb{R}^m$ is the state set for each agent and similarly, $\mathcal{A} \in \mathbb{R}^k$ is the action set. $\mathcal{C}$ denote the set of constraints. $\mathcal{C}_i \in \mathcal{C} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$ is a independent term and define the constraint. The γ_r and γ_c denotes the decay rate of the reward and cost, respectively.

$$o_i(t) = \{o_e(t), \mathcal{N}_1(t), \mathcal{N}_2(t), \cdots, \mathcal{N}_n(t)\} \tag{1}$$

$$\mathcal{J}_R = \mathbb{E}[\sum_{t=0}^{T} \gamma_r^t r_i(t)],\ \mathcal{J}_{\mathcal{C}_i} = \mathbb{E}[\sum_{t=0}^{T} \gamma_c^t \mathcal{C}_i(t)] \leq \epsilon_i \tag{2}$$

For a MAS with N agents $\mathcal{E}_N$, assuming $a_{i,t} \in \mathcal{I}$ is the i-th agent at the time t and $\mathcal{I}$ is the agent set, there are n other agents in the observable range of $a_{i,t}$ as shown in Fig. 1, and the proportion of others in its observation can be represented as $o_n(t) \in \mathbb{R}^{n \times |\mathcal{N}_j(t)|}$, where $\mathcal{N}_j(t)$ is its neighbors information and o_n has the unfixed dimension at different time t for the changing n. Thus, with the ego part $o_e(t) \in \mathbb{R}^e$, the observation of $a_{i,t}$ can be described as follow. The observation $o_i(t)$ has the changeable dimension for the changing n at different time t. For decentralized system, let $u_i(t) \in \mathcal{A}$ and $r_i(t) \in \mathcal{R}$ denote the action and immediate reward in a length-T episode. For a CMDP, the episode return and discounted cumulative cost can be defined as $\mathcal{J}_R$ and $\mathcal{J}_{\mathcal{C}_i}$ in the Equation (2).

C. The safety of multi-agent system

For the MAS $\mathcal{E}_N$, the joint state of this system at time t is recorded as $s^{\tilde{j}}(t) = \{s_1(t), s_2(t), \cdots, s_N(t)\} \in \mathcal{S}^{\tilde{j}}$, where $s_i(t) \in \mathcal{S}$ is the state of $a_{i,t}$ and $\mathcal{S}^{\tilde{j}}$ is the joint state set. Let $u_i(t) \in \mathcal{A}$ donates the action and $d_i(s_i(t), u_i(t))$ is the dynamics function of $a_{i,t}$. Consider the state-observation space $\mathcal{X}_i = \mathcal{S}_i \times \Omega_i$, $\mathcal{X}_i$ contains the safe set $\mathcal{X}_i^{\tilde{s}}$ as Equation (3), where $\mathcal{D}(s_i, o_i) \geq \tilde{l}$ describes the safe condition that Euclidean Metric $\mathcal{D}$ is greater than the safety threshold $\tilde{l}$, then the dangerous set is $\mathcal{X}_i^{\tilde{d}} = \mathcal{X}_i \setminus \mathcal{X}_i^{\tilde{s}}$.

$$\mathcal{X}_i^{\tilde{s}} = \left\{(s_i, o_i) \mid \mathcal{D}_i(s_i, o_i) \geq \tilde{l}\right\} \tag{3}$$

$$\mathcal{S}_s^{\tilde{j}} = \left\{s \in \mathcal{S} \mid \forall a_i \in \mathcal{I}, \mathcal{D}_i(s_i) \geq \tilde{l}\right\} \tag{4}$$

For the agent i at time t, it is safe when its state-observation satisfies $\mathcal{D}(s_i(t), o_i(t)) \geq \tilde{l}$. If all agents at time t are safe, then the multi-agent system $s^{\tilde{j}}(t) \in \mathbb{R}^{N \times m}$ is safe. The formulaic description of the safe joint state $\mathcal{S}_s^{\tilde{j}}$ is as Equation (4).

IV. Methodology

In this section, we will introduce the principle and pipeline of our framework. Meanwhile, we defined the crucial functions in the PIPO in this part.

A. Artificial Potential Field in PIPO

The artificial potential field (APF) is an intuitive path planning algorithm and achieves many outstanding results in the autonomous systems [21], [22]. According to Section. III-B, the agent observation is separated into the ego state o_e and its neighbors' information o_n. Hence, with the partial observation, the relative location and speed of the adjacent entities are known for a_i. For each agent, it maintains a distributed occupancy map to represent its field-of-view and calculates its local targets and potential field force independently. Thus, through the observation $o_i(t) \in \Omega$ at time t of a_i, we can calculate the potential field force $\mathbf{F}_f$ as Equation (5), which is consist of the attractive proportion $\mathbf{F}_{att}$ and repulsive proportion $\mathbf{F}_{rep}$. Note that the random sampling or searching methods like A*, RRT and their variants [23]–[25] are also workable as the guide-policy, while they have higher time complexity and uncertainty.

$$\mathbf{F}_f(o_i) = \eta_{att} \underbrace{\mathbf{L}_{\mathcal{G}_i}^2}_{\text{attractive}} + \eta_{rep} \sum_{j=1}^{n} \underbrace{\widehat{\mathbf{L}_j} d_i d_j / (|\mathbf{L}_j| - d_i - d_j)^2}_{\text{repulsive}} \tag{5}$$

$$u_{g,i}(t) = tanh\left[\mathbf{F}_f(o_i(t))\right] \in \mathbb{R}^k \tag{6}$$

In Equation (5), $\mathbf{L}_{\mathcal{G}_i}$ and $\mathbf{L}_j$ are the relative location of a_i's destination $\mathcal{G}_i$ and that of the neighbor a_j respectively, where $\widehat{\mathbf{L}\cdot}$ is the unit vector of $\mathbf{L}\cdot$. The d_j presents the size of the j-th observed agent. Besides, we utilize a couple of coefficients (η_{att}, η_{rep}) to balance the proportion between $\mathbf{F}_{att}$ and $\mathbf{F}_{rep}$. We therefore map the potential force to the normalized control commands through a hyperbolic tangent function as Equation (6).

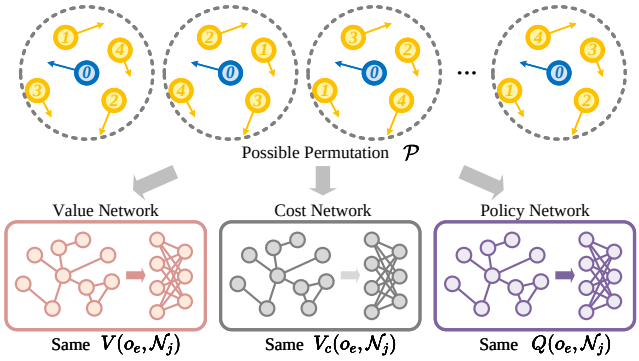


Fig. 2. Permutation-Invariant property in decentralized agent observation. Networks for both value $V(s \mid \theta_v)$ and policy $\pi(s \mid \theta_\pi)$ take different possible $\mathcal{P}_x$ for adjacent agents as input and are supposed to output the same estimation and action.

B. Constrained Proximal Policy Optimization

For multi-agent navigation, each agent is required to reach the target while avoiding colliding with other agents and obstacles, and our goal is to maximize the system efficiency while guaranteeing safety. It means that the system is a goal-based and both cooperative and competitive environment. The reward of PIPO is defined as Equation (7). The agent obtains a positive reward for reaching its goal while is punished when collides with other agents.

$$r = \begin{cases} +1 & \textit{if reach the target} \\ -1 & \textit{if collide} \end{cases} \tag{7}$$

$$c_j^i = \left[(|\mathbf{L}_j| - d_i)/(\hat{d} - d_i)\right]^2, \ s.t. \ |\mathbf{L}_j| \leq d_i + \hat{d} \tag{8}$$

We set our constraints for from j-th neighbor to the i-th agent as Equation (8), in which $\hat{d}$ is a general range and $\mathbf{L}_j$ is the relative location of the j-th adjacent agent. Compared to the discrete constraint, the agent obtains a smoother state cost with early warning.

Proximal Policy Optimization (PPO) [7] is a classic RL algorithm for continuous control. Generally, the optimized $\pi(s \mid \theta_\pi)$, which is the actor network parameterized by θ_π, provides the action $u_i(t)$ based on the observation $s_i(t)$ for the agent $a_{i,t}$. Concurrently, the critic network $V(s \mid \theta_v)$ and the cost network $C(s \mid \theta_c)$ estimates the reward and the cost of state $s_i(t)$, where θ presents the trainable parameters. These networks are trained with the mean square error $L_V(v_e, v_{truth})$ and $L_C(c_e, c_{truth})$, which presents the error between the true value and the estimation one.

$$\hat{A}_Q^t = -V(s_t) + r_t + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T) \tag{9}$$

$$\hat{A}_C^t = -C(s_t) + c_t + \cdots + \gamma^{T-t+1} c_{T-1} + \gamma^{T-t} C(s_T) \tag{10}$$

For original PPO, researchers introduce the clipped policy divergence and generative advantage estimation (GAE) [26] to constrain the update and reduce the error of the advantage estimation, respectively. The $\hat{A}$ is the GAE function defined as Equation (9). Similarly, the advantage of cost is defined as Equation (10).

$$L_Q(\theta_\pi) = \hat{\mathbb{E}}\left[min(p\hat{A}), \ clip\left(p_{\theta_\pi}, 1-\epsilon, 1+\epsilon\right)\right] \tag{11}$$

$$L_C(\theta_{\pi_c}) = \hat{\mathbb{E}}\left[(p\hat{A}_c)\right] \tag{12}$$

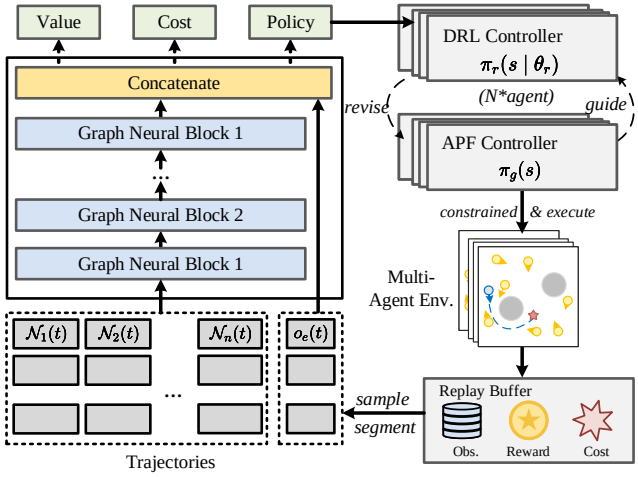


Fig. 3. The framework of PIPO. The global policy π_s generate actions to interact with environment and consists of the APF-based guide policy π_g and the residual policy π_r. The replay buffer stores decentralized trajectories with only the actions from residual policy π_r and independently sampling to update the value and policy networks.

$$\hat{\theta_\pi} = argmin\ [-L_Q(\theta_\pi) + \lambda L_C(\theta_{\pi_c})] \tag{13}$$

As the original definition in PPO, the loss of state-action (Q network) can be presented as Equation (11). The probability ratio $p = \pi(u|s,\theta_\pi)/\pi(u|s,\theta_\pi^{'})$ estimates the divergence of the former policy with $\theta_\pi^{'}$ and the updated θ_π. The expectation of the minimum product of advantage estimation and the clipped divergence equals to the Q-loss. Similarly, the cumulative cost loss is defined as Equation (12) and we can transfer our target and constraints into a convex optimization with Lagrangian relaxation. As shown in Equation (13), the λ is an adaptive Lagrangian multiplier. The $\hat{\theta_\pi}$ is our target and the optimal inner parameters of the policy.

C. Graph Convolutional Policy Optimization

As discussed in Section. III-B, the input to decentralized agents is $o_i \in \mathbb{R}^{E+n\times|\mathcal{N}_j|}$. They are concatenated in MPEs and imposed implicitly permutation of the adjacent agents. However, the outputs of networks are supposed to be same after shuffling the agent sequence, which is the permutation-invariant property and can be presented as formulaic description. In Equation (14) (15), $\mathcal{P}_x, \mathcal{P}_y \in \mathcal{P}$ present different possible permutations for $o_n \in \mathbb{R}^{n\times|\mathcal{N}_j|}$ in set $\mathcal{P}$.

$$\pi\left(o_e, \mathcal{P}_x(o_n) \mid \theta_\pi\right) = \pi\left(o_e, \mathcal{P}_y(o_n) \mid \theta_\pi\right) \tag{14}$$

$$V\left(o_e, \mathcal{P}_x(o_n) \mid \theta_V\right) = V\left(o_e, \mathcal{P}_y(o_n) \mid \theta_V\right) \tag{15}$$

According to the above discussion, we propose a graph convolutional network (GCN) as shown in Figure 2. We take each segment of adjacent agents $\mathcal{N}_j$ as the nodes for GCNs. Then GCNs seek new representations for each node and latent connections among them. Importantly, the GCNs establish the mapping from $o_n \in \mathbb{R}^{n\times|\mathcal{N}_j|}$ to $\mathcal{V}_G \in \mathbb{R}^{n\times|\mathcal{V}_j|}$, where $\mathcal{V}_j$ is the output feature of GCN for each neighbor. Then the state of ego agent is concatenated and convoluted at the dimension of GCN latent feature. After that, fully connected layers are leveraged to achieve the permutation-invariant at the output. Noting that concatenate layer is $\mathcal{L}_c$ and graph neural blocks are $\left(\mathcal{L}_g^1, \mathcal{L}_g^2, \cdots, \mathcal{L}_g^p\right)$, we record the formal description of output before the value and policy layers $\mathcal{V}_c$ as follow.

$$\mathcal{V}_c(o_i) = \mathcal{L}_c \odot \left\{o_e + \mathcal{L}_g^p \odot \cdots \odot \mathcal{L}_g^2 \odot \mathcal{L}_g^1(o_n) \mid \sigma\right\} \tag{16}$$

Significantly, for different $\mathcal{N}_j$ in o_n, the trainable parameters in each $\mathcal{L}_G$ are shared but independent for policy, value and cost networks, which can effectively accelerate the train process and curtail the models. The σ is the ReLU activation of the layers in GCNs and $\odot$ transfers the latent features.

Algorithm 1 Pseudo-code of PIPO

Input: the ego state o_e, observed adjacent entities o_n

Initialize the N-agent sytem $\mathcal{E}_N$, network parameters θ_π, θ_V and θ_C, replay buffer $\mathcal{B}$, the number of episodes M, the episode length T
for m in range M **do**
  Initialize the poses and targets for $\mathcal{E}_N$
  for t in range T **do**
    Get the observation $o_i = o_e \cup o_n$ for $\mathcal{E}_N$
    Get the guide-action u_g for $\mathcal{E}_N$ as Equation (6)
    Segment and shuffle o_n randomly
    Get the u_r, v_e, c_e from networks for $\mathcal{E}_N$
    env.step (clip $(u_g + u_r)$), get reward r and cost c
    Store $\{o_i, u_r, c, r, v_e, c_e\}$ in buffer $\mathcal{B}$
  end for
  Calculate the loss L_π, L_V and L_C
  Update parameters θ_π, θ_V and θ_C
end for

Output: policy $\pi(o|\theta_\pi)$

D. Learning Residuals beyond Guidance

Residual policy learning (RPL) is a efficient technique for RL-based robot control [27]. It deploy a guide policy at the start of training, which can guide the agent explore with feasible actions and generate better trajectories for training. In classic RL, the global policy $\pi_s(o|\theta_\pi) \rightarrow u_s$ is learned from zero in previous DRL methods, while we divide it into the residual-policy $\pi_r(o|\theta_\pi) \rightarrow u_r$ with trainable parameters θ_π and the guide-policy $\pi_g(o) \rightarrow u_g$. We can express their relationship as $\pi_s(o|\theta_\pi) = \pi_r(o|\theta_\pi) + \pi_g(o)$. Importantly, the gradient satisfies $\bigtriangledown_{\theta_\pi}\pi_s(o|\theta_\pi) = \bigtriangledown_{\theta_\pi}\pi_r(o|\theta_\pi)$. It mean that the gradient of π_s does not depend on π_g so the optimal π_s is learnable through policy gradient method even π_g is not differentiable. So we say instead of learning from zero, RPL learns the residual part beyond the guide policy.

We give out the pseudo-code of PIPO in Algorithm 1. Different from former methods, the observation of each agent is divided into the ego part and the segments of adjacent agents as the Equation (1). Then it is randomly shuffled and stored in the replay buffer. This is a crucial technique of our algorithm and we will illustrate its contribution in the experimental section. The graph convolutional blocks (GCBs) extract the latent permutation-invariant descriptor from shuffled segments of the adjacent agents and concatenate it with the ego state information. In PIPO, there are

three parallel networks for outputting the estimation of value, cost and residual policy independently and they all uses this architecture while GCBs are connected to output layers with different dimensions. For decentralized policy optimization, the shared replay buffer only store the residual policy and π_r and gives out the independent trajectories of different agents to updated the policy to update the network. Due to the high parallel process in PIPO and lower parameters of GCBs than classic MLPs, our approach can be trained efficiently and realizes high real-time performance.

V. Experiments and Results

In this section, we illustrate the content of our experiments, including different particle navigation tasks in improved MPEs, the baselines for comparison and the ablation studies.

A. Baselines

To illustrate the advantages of our method, we compared it with leading centralized training and decentralized executing (CTDE) MARL and decentralized methods as follows.

- *MADDPG-safe:* multi-agent deep deterministic policy gradient (MADDPG) is a classic CTDE reinforcement learning method and has been proved to be workable on different robot control tasks [2].
- *MAPPO-safe:* multi-agent proximal policy optimization (MAPPO) is the revised version of PPO for multi-agent system and the state-of-the-art (SOTA) CTDE reinforcement learning method [28].
- *PIC-safe:* it is a robust multi-agent control algorithm because it enhances the representation of policy through permutation-invariant critic (PIC) [16]. However, different from our method, only the critic network of it possesses this property and it is a centralized method.

Due to regular MARL methods can hardly handle the sparse reward function used in our method, we set their reward as the continous one in the original MPEs with the collision penalty and mark them with *"-safe"*.

B. Experimental Settings

At the start of each episode, the locations and targets of all agents are randomly initialized. The distances among all initial locations and targets are large enough to guarantee the beginning and final states of the system are safe. We also improve the diversity of navigation scenarios in MPEs to validate the algorithms. For structured scenarios like factories and rooms, we add blocks and piles as static obstacles. Besides navigation on the plane, there are 4 and 9 static blocks, and piles are arranged as the medium and difficult environments, respectively. For CTDE baselines, we retrain them in these scenarios with increasing agents (8, 16, 32) for 3 Million time-steps with the 100 maximum episode length. For our decentralized method, we first train the models with 2 million time-steps and scale them to 16 and 32 agents with additional 500k million-steps training. These settings reduce the feasible range of navigation and make security more difficult to guarantee. All methods are trained for 5 trials with independent seeds. We also validate our trained models on over 500 agents to illustrate our method is scalable. All Experiments are completed on the server with two *Intel XEON 8160* CPUs and an *Nvidia RTX 3070* GPU.

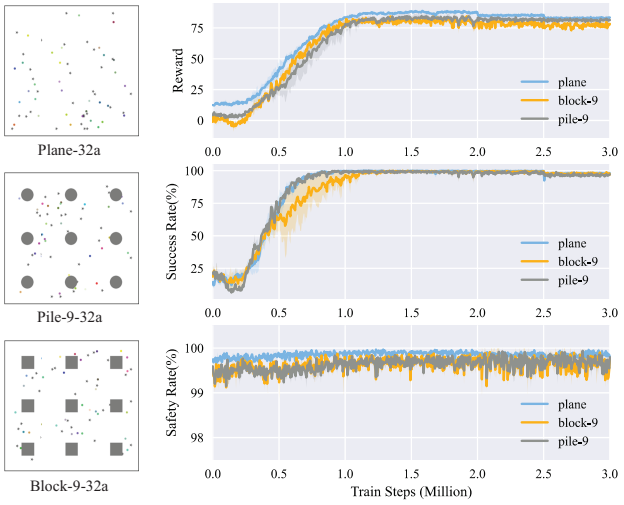


Fig. 4. The visualization of training processes and scenarios. We visualize the training curves of reward, success rate and safety rate. The models are first trained for 2 million steps in the 8-agent scenarios, and extended to 16-agent and 32-agent environments for additional two 500K steps.

The metrics of our tasks include the system safety, efficiency and success rate in one episode. Meanwhile, for the agents reaching their targets, we calculate their proportion of the time consumption t_a to the episode length T, and use the $1 - t_a/T$ as the efficiency metric. To emphasize the gap of safety rate, we calculate the transferred safety by $20 * (\eta - 0.95)$, where η is the ture safety rate and always larger than 0.95. Thus, we can scale all above metrics into [0, 1] and the larger value indicates the better performance.

C. Experimental Results

We first denote three scenarios on the plane with different obstacles and agent scales. As shown in Figure 4, the environment with 32 agents and 9 piles is named as *pile-9-32a*. Similarly, the plane scenario with 32 agents is named as *plane-32a*. We also visualize the training process in Figure 4. These curves illustrate our PIPO is scalable to different system scales and possesses high train efficiency, which can converge rapidly in about 1 million time-steps. With the increasing agents, the reduction of PIPO in efficiency, success rate and safety is controllable.

Performance Comparison: In Figure 5, our PIPO show much lower sensitivity to the agent scale. In the small-scale environments, the CTDE and decentralized approaches show comparable performances. However, with the increasing number of agents, the gaps in safety rates and success rates raise rapidly. The large agent scale increases the dimension of concatenated state and the estimation from the value network can hardly converge. Thus, they struggle to learn the feasible policies with over 16 agents and the agents always stay still or move randomly, which reduces their success rate but slightly improves the safety rate. Although the representation of critic in PIC is enhanced by GCNs and it outperforms the

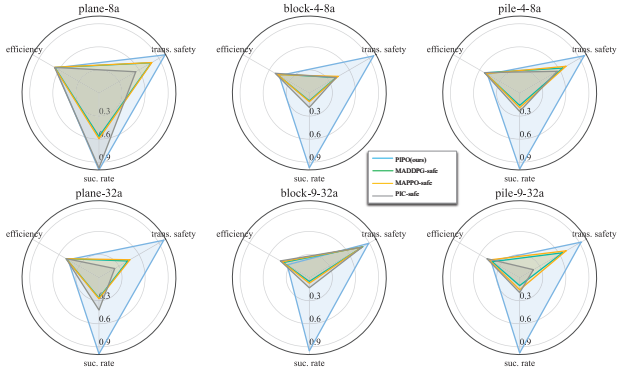


Fig. 5. The radar maps demonstrate the performance of baselines after 3 million time-steps of training. (1) *Efficiency* is positively related to the cumulative rewards and calculated by $1 - t_a/T$, where t_a is the reaching time and T is the maximum episode length. (2) *Success rate* indicates the proportion of the agents reaching their targets in finite time-steps. (3) *Transferred safety* enlarges the safety gaps of baselines and is calculated by $20 * (\eta - 0.95)$, where η is the safety rate.

MADDPG and MAPPO, it requires much more steps to train than our method and can hardly guarantee the safety.

Contrarily, PIPO shows the robustness on safety and success rate in different scenarios and system scales. Different from PIC, the representations of cost, policy, and state value are reinforced by GCNs and random shuffling equally, and it requires no other improvement in the size and depth when the system scale increases. Meanwhile, the square obstacle (blocks) give more challenges to agent navigation because they cover more areas and are tougher to avoid than the piles. According to our definition in Equation (3), the difficulty of maintaining a safe joint state is exponentially related to the scale of agents. Besides, the adjacent agents can be regarded as moving obstacles so the safety is generally reduced with the increasing number of agents. But for PIPO, this tendency is controllable and it still keeps over 99.5% safety rate in the 32-agent environment, whereas other methods can hardly reach the 98%. Besides, to guarantee safety, the agents of PIPO in the large system decelerate more frequently to avoid collision with other agents, and their efficiency and success rate are therefore always slightly reduced.

Scalability: We validate the models trained within 32-agent scenarios in much larger systems with (64, 128, 256, 512) agents to illustrate the scalability of PIPO. As shown in Figure 6, when the models are scaled, they show the same tendency we mentioned before. With the increasing system scale, the safety rates maintain over 99.80% with acceptable reduction and the success rates keep over 95%. The agents can hardly guarantee to reach their targets perfectly because of their conservative policy. According to the demonstration in the simulator, the agent stops when other agents come close to it and therefore spends more time than the steps in one episode to reach its target. These results illustrate our method requires no additional retraining and is general and scalable to different system scales, which improves the practical contribution of PIPO.

Ablation Studies: We illustrate the contribution of each crucial module in PIPO through ablation experiments. For

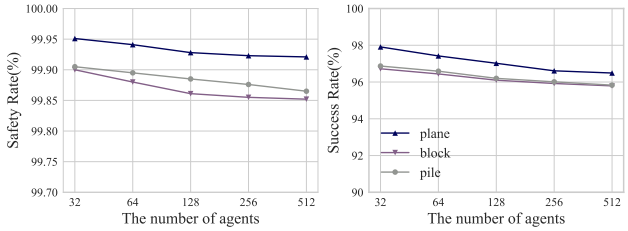


Fig. 6. The scalability validation of the models trained in 32-agent systems. We double the number of agents to 512. The safety rate and success rate are slightly reduced but still controllable. Meanwhile, the performance in pile scenarios is better than that in block scenarios.

residual policy learning, we validate the performance of guide-policy and residual-policy independently. Besides, we replace the graph convolution networks with the typical multi-layer perceptron to validate the improvement coming from GCNs. Meanwhile, we remove the random shuffling and train the models with the fixed permutation to demonstrate the contribution of this technique. We conduct the ablation studies in the easiest scenario *plane-8a* and the most challenging scenarios.

TABLE I
THE RESULTS OF ABLATION EXPERIMENTS

Method	Reward			
	plane-8a	plane-32a	pile-9-32a	block-9-32a
w/o APF	79.95	76.12	74.34	72.87
w/o PPO	48.69	48.67	48.59	48.64
w/o GCB	76.92	74.84	73.18	71.21
w/o shuffling	81.42	78.34	75.41	72.98
PIPO	**86.26**	**84.34**	**81.29**	**77.59**

As shown in Table I, the ablation results demonstrate the improvement of each technique in PIPO. In essence, the different network architectures obtain similar results. Compare with MLPs, the GCNs without guidance require more time-steps for training than MLPs. The performances of zero-start policy optimization and independent APF are both unsatisfactory, while learning residuals beyond the guide-policy effectively improves the cumulative reward, which evidences the complementary property between the guide-policy and residual-policy. Meanwhile, random shuffling does improve the average reward by $+6.92\%$, which illustrates the extracted representation is enhanced by utilizing the permutation-invariant property of observation.

VI. CONCLUSION

In summary, we proposed a decentralized RL algorithm with a branched and permutation-invariant structure based on the graph convolutional network for multi-agent navigation. Our decentralized approach solves the exponential dependence of the system scale and can be generalized to an arbitrary number of agents. Concurrently, the guidance from APF effectively improves the sampling efficiency and avoids the risk behaviors. Our extensive experiments show that our PIPO outperforms the leading baselines and attains state-of-the-art safety performance in different tasks. The stable

exploration process and high safety rate reduce the difficulty of the deployment on the real-world systems and extend the application of PIPO. In our future work, we will discuss the solution for existing problems, including the causes for stationary, the unsafe behaviors in complex scenarios, etc. Moreover, we would like to validate our method in 3D scenarios with more complex robot dynamics and the multi-agent system on a larger scale.

REFERENCES

[1] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

[2] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *NeurIPS*, pp. 6379–6390, 2017.

[3] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *ICML*, pp. 4295–4304, PMLR, 2018.

[4] W. Mao, T. Başar, L. F. Yang, and K. Zhang, "Decentralized cooperative multi-agent reinforcement learning with exploration," *arXiv preprint arXiv:2110.05707*, 2021.

[5] B. Yongacoglu, G. Arslan, and S. Yuksel, "Decentralized learning for optimality in stochastic dynamic teams and games with local control and global state information," *IEEE Transactions on Automatic Control*, 2021.

[6] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proceedings of the 2018 World Wide Web Conference*, pp. 499–508, 2018.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[8] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[9] D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Barták, and N.-F. Zhou, "Robust multi-agent path finding and executing," *Journal of Artificial Intelligence Research*, vol. 67, pp. 549–579, 2020.

[10] J. Chen, J. Li, C. Fan, and B. C. Williams, "Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances," in *AAAI*, pp. 11237–11245, 2021.

[11] C. Mavrogiannis and R. A. Knepper, "Hamiltonian coordination primitives for decentralized multiagent navigation," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1234–1254, 2021.

[12] B. Zhang and H. P. Gavin, "Gauss's principle with inequality constraints for multi-agent navigation and control," *IEEE Transactions on Automatic Control*, 2022.

[13] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2021.

[14] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," in *ICLR*, 2021.

[15] G. Arslan and S. Yüksel, "Decentralized q-learning for stochastic teams and games," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1545–1558, 2017.

[16] I.-J. Liu, R. A. Yeh, and A. G. Schwing, "Pic: permutation invariant critic for multi-agent deep reinforcement learning," in *CoRL*, pp. 590–602, PMLR, 2020.

[17] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. Torr, M. Sun, and S. Whiteson, "Is independent learning all you need in the starcraft multi-agent challenge?," *arXiv preprint arXiv:2011.09533*, 2020.

[18] Z. Peng, Q. Li, K. M. Hui, C. Liu, and B. Zhou, "Learning to simulate self-driven particles system with coordinated policy optimization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10784–10797, 2021.

[19] R. Han, S. Chen, S. Wang, Z. Zhang, R. Gao, Q. Hao, and J. Pan, "Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5896–5903, 2022.

[20] E. Altman, *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.

[21] F. Bounini, D. Gingras, H. Pollart, and D. Gruyer, "Modified artificial potential field method for online path planning applications," in *IV*, pp. 180–185, IEEE, 2017.

[22] Z. Pan, C. Zhang, Y. Xia, H. Xiong, and X. Shao, "An improved artificial potential field method for path planning and formation control of the multi-uav systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2021.

[23] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[24] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[25] W. Xing, A. Song, and L. Zhu, "Real-time robot path planning using rapid visible tree," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11182–11188, IEEE, 2021.

[26] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *ICLR*, 2016.

[27] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[28] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of mappo in cooperative, multi-agent games," *arXiv preprint arXiv:2103.01955*, 2021.