Hierarchical and Stable Multiagent Reinforcement Learning for Cooperative Navigation Control

Yue Jin, *Graduate Student Member, IEEE*, Shuangqing Wei, *Senior Member, IEEE*, Jian Yuan, *Member, IEEE*, and Xudong Zhang, *Member, IEEE*

***Abstract*— We solve an important and challenging cooperative navigation control problem, Multiagent Navigation to Unassigned Multiple targets (MNUM) in unknown environments with minimal time and without collision. Conventional methods are based on multiagent path planning that requires building an environment map and expensive real-time path planning computations. In this article, we formulate MNUM as a stochastic game and devise a novel multiagent deep reinforcement learning (MADRL) algorithm to learn an end-to-end solution, which directly maps raw sensor data to control signals. Once learned, the policy can be deployed onto each agent, and thereby, the expensive online planning computations can be offloaded. However, to solve MNUM, traditional MADRL suffers from large policy solution space and nonstationary environment when agents make decisions independently and concurrently. Accordingly, we propose a hierarchical and stable MADRL algorithm. The hierarchical learning part introduces a two-layer policy model to reduce the solution space and uses an interlaced learning paradigm to learn two coupled policies. In the stable learning part, we propose to learn an extended action-value function that implicitly incorporates estimations of other agents' actions, based on which the environment's nonstationarity caused by other agents' changing policies can be alleviated. Extensive experiments demonstrate that our method can converge in a fast way and generate more efficient cooperative navigation policies than comparable methods.**

***Index Terms*— Cooperative navigation, hierarchical policy learning, multiagent deep reinforcement learning (MADRL).**

I. Introduction

MULTIAGENT cooperative navigation is an important function for an intelligent robot team to complete cooperative tasks, which has drawn a lot of attention in recent years. In this article, we solve the problem of Multiagent Navigation to Unassigned Multiple targets (MNUM), where agents need to arrive at the same number of unassigned targets autonomously without collision and using minimal makespan (the time spent by the agent who is the last one to arrive at a target). Some applications include multirobot formation control [1]–[3], multirobot target search [4], and autonomous mobile service robots [5], where target positions are known and each robot can reach any target position, but they are required to minimize their makespan.

Conventional methods are planning-based two-step methods. These methods depend on an environment map to assign targets and plan collision-free paths for agents and then compute motion values for agents to move along the paths. Typical planning methods of solving MNUM mainly include centralized planning and distributed planning. In centralized planning [6]–[9], targets are assigned either before or simultaneously with path planning. To avoid collisions or blocks between paths, a path planning priority of each agent is computed. In unknown environments where doing such a global and one-time planning is impossible, dynamic path planning methods are employed to do periodical replanning based on updated environment map and some assumptions about the unknown part of environment [10]. However, in many practical applications, running a centralized planning algorithm in real time is time-consuming and unacceptable, especially when the number of agents is large [11].

Distributed planning-based methods [3], [12]–[14] are generally more computationally efficient. They dynamically adjust target assignment based on communications between the agents. For instance, in [13], a heuristic method is applied to determine whether to exchange target selections between two agents when they are within a communication distance. Simultaneously, to avoid collisions, local planning methods are employed on each single agent. Because of lacking global target assignment and path planning, distributed planning-based methods usually lack optimality. Besides, they are not applicable to communication-constrained cases.

In this article, we leverage multiagent deep reinforcement learning (MADRL) to learn an end-to-end solution to MNUM, which directly maps raw sensor data to control signals instead of using planning-based methods. Specifically, the training phase is conducted in random environments, during which a cooperative policy can be learned by agents. Once learned, the policy is deployed onto each agent to complete MNUM in unknown environments without the need of time-consuming planning and exchanging information about target selections.

Despite its great potentials, the MADRL-based method is hindered by some crucial challenges, including large policy solution space and nonstationary environment. Specifically,

Manuscript received January 15, 2020; revised September 3, 2020 and March 14, 2021; accepted June 6, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant U20B2060. *(Corresponding author: Shuangqing Wei.)*

Yue Jin, Jian Yuan, and Xudong Zhang are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: jiny23@126.com; jyuan@tsinghua.edu.cn; zhangxd@mail.tsinghua.edu.cn).

Shuangqing Wei is with the School of Electrical Engineering and Computer Science, Louisiana State University, Baton Rouge, LA 70803 USA (e-mail: swei@lsu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3089834.

Digital Object Identifier 10.1109/TNNLS.2021.3089834

large policy solution space is caused by the combination of multiple agents' policies. In MNUM, an implementation of an agent's policy generates a trajectory from the agent's initial location to its final one when an episode ends. Reinforcement learning (RL) can improve a policy toward gaining more cumulative rewards in a trajectory. In MNUM, each agent's reward is related to other agents' policies. Therefore, agents need to try sufficient possible combinations of their policies so that each agent can know how to get more rewards. To tackle this challenge, previous works [15], [16] propose to decompose cumulative rewards and disentangle the combined policy solution space as separate policy solution spaces for each agent. However, they are not feasible for tasks involving complex dependence among policies. Other works [17]–[19] apply hierarchical RL methods [20] to multiagent cases. Specifically, high-level policies are cooperatively learned to select subgoals and low-level policies are independently learned to complete primitive actions related to the subgoals. However, for the tasks that require close cooperation among agents, extracting independent low-level policies becomes infeasible.

The second challenge, i.e., nonstationary environment, is caused by agents' concurrent learning and independent decision-making when they have to treat other agents as part of the environment. In MNUM, from the perspective of a certain agent, the environment state (observation) includes other agents' locations. Therefore, state transition is correlated with all agents' policies. Because agents' policies change during learning and agents make decisions independently, from the perspective of each agent, states' transition statistics are nonstationary. To address this issue, some works [15], [16], [21] directly assume that other agents' policy functions or value functions are available during learning, and therefore, each agent can leverage them to learn its policy stably. These works adopt a centralized learning framework, which may cause high computational overhead and low robustness. Other works [22], [23] propose to infer other agents' policies by supervised learning or Bayesian inference. However, when policy functions are constructed with deep neural networks, estimating them causes heavy computational overhead. Foerster *et al.* [24] proposed to replace other agents' policies with low-dimensional fingerprints that can record policy changes to some extent. However, it is difficult to get a tradeoff between low computational overhead and policy information completeness [25].

To address the challenges mentioned above, we propose a novel hierarchical and stable MADRL (HIST-MADRL) framework. Specifically, inspired by the idea of hierarchical policy learning [26], we propose to learn a hierarchical navigation policy, where a high-level policy selects a target dynamically and a low-level policy decides a steering direction, rather than directly learning a steering policy without target selection [22], [27], [28]. In this way, each agent can only move within a direction range pointing to its selected target. Moreover, when no obstacle is observed in the target direction, agents will not try to move toward other directions except the direction toward the target. As a result, policies that generate trajectories deviating from target directions will not be tried by agents. Thus, policy solution space can be reduced considerably.

~~To address the nonstationary environment issue, we propose a stable MADRL method that goes beyond treating all agents as part of the environment. Specifically, each agent takes the estimation of other agents' actions into consideration to evaluate its policy, which alleviates the nonstationarity caused by unknown changes in others' policies. To this end, we leverage other agents' states at adjacent time steps to estimate their actions. To guide policy learning with the action estimation, we extend the traditional action-value function as a composite function, which incorporates the estimation function, and derive the Bellman (optimality) equation for the extended action-value function that does not include nonstationary components. Therefore, policies can be learned stably.~~

Based on the general stable MADRL method, we propose two stable learning algorithms, stable multiagent deep Q-learning (SMADQN) and stable multiagent deep deterministic policy gradient (SMADDPG) algorithms, to learn the high- and low-level policies. Because the high-level policy selects a target, its action space is discrete. This policy is learned based on SMADQN. Meanwhile, as the action space of the low-level policy is a continuous steering angle space, this policy is learned based on SMADDPG. Moreover, because the two policies are coupled, we propose an interlaced learning framework to learn them. Simulation results demonstrate the efficiency and superiority of our HIST-MADRL method in terms of both convergence speed and agents' cooperation performance compared with comparable and relevant methods.

The main contributions of this article are as follows.

1) We model MNUM as a stochastic game (SG), including a reward function designed based on the MNUM's objective.
2) We propose a general stable MADRL method, where each agent can learn its policy stably with the guidance of an extended action-value function incorporating implicit estimations of others' actions. As there is no need to know others' policy functions or to estimate them, agents can learn policies independently with low computational complexity.
3) We propose a novel hierarchical and stable MADRL framework to solve MNUM based on the general stable MADRL method and an interlaced learning framework. The policy solution space can be reduced compared with single-policy learning method.

The remainder of this article is organized as follows. Section II introduces background knowledge about Markov decision process (MDP), SG, and DRL frameworks briefly. In Section III, we model MNUM as an SG. Our proposed HIST-MADRL framework for MNUM is elaborated in Section IV. Simulation results and discussions are in Section V. Section VI draws a conclusion and presents some future works.

II. Background

In this section, we first introduce basics about MDP and SG. Then, we give a brief introduction to DRL framework and two typical MADRL algorithms.

A. MDPs and SGs

MDP is a model that describes a decision process of an agent in partly random environments. At each time step, the process is in a certain state $s \in S$ and an agent executes an action $a \in A$, where S and A denote the state space and action space, respectively. Then, the process moves into the next state s' and gives the agent a reward feedback $r(s,a)$. s' obeys a state transition probability function $p(s'|s,a)$ that satisfies the Markov property. A problem in an MDP is to find a policy that specifies the action of an agent at each time step and maximizes the expectation of a long-term cumulative discounted reward $E[\sum_{t=0}^{T}\gamma^t r(s^t,a^t)]$, where T is the time horizon of the MDP, $0 \leq \gamma \leq 1$ is a discount factor, and s^t and a^t denote the state and action at time step t, respectively. The policy could be either deterministic or stochastic. The deterministic policy, which can be denoted as $\pi(s) : S \to A$, maps states to actions. The stochastic policy, which can be denoted as $\pi(a|s) : S \times A \to [0,1]$, gives an action probability in each state. For simplicity, we use π to denote both kinds of policies.

An SG that is also referred as a Markov game extends an MDP to the setting containing multiple agents. An SG with N agents involves a set of states s, joint action $(a_1,\ldots,a_N)$, transition probability function $p(s'|s,a_1,\ldots,a_N)$, and each agent's reward function $r_i(s,a_1,\ldots,a_N)$. At each time step, each agent executes an action according to its policy π_i. A problem of SG is to find the optimal policy π_i^* for each agent so that $\forall\pi_i, R_i(s^t,\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_N^*) \geq R_i(s^t,\pi_1^*,\ldots,\pi_i,\ldots,\pi_N^*)$, where $R_i(s^t,\pi_1,\ldots,\pi_N) = E[\sum_{\tau=0}^{T}\gamma^\tau r_i(s^{t+\tau},a_1^{t+\tau},\ldots,a_i^{t+\tau},\ldots,a_N^{t+\tau})]$ denotes the expectation of a cumulative discounted reward of agent A_i.

For the sake of conciseness, in the following, we use r^t and r_i^t to denote the value of the reward gained by the whole system and agent A_i at time step t, respectively.

B. Deep Q-Learning, Actor–Critic, and Deep Deterministic Policy Gradient

The optimal policy of an MDP with unknown transition probability and reward function can be solved by RL. To evaluate a policy with a discounted long-term rewards, two value functions are defined in RL. The first one is termed state-value function, which measures the expected cumulative discounted rewards when starting in s and following π thereafter, which is defined as:

$$V^\pi(s) = \mathbb{E}_{a^t,s^{t+1},\ldots}\left[\sum_{\tau=0}^{T}\gamma^\tau r^{t+\tau+1}\middle|s^t=s,\pi\right]. \quad (1)$$

Another value function is termed action-value function (Q-function), which is defined as

$$Q^\pi(s,a) = \mathbb{E}_{s^{t+1},a^{t+1},\ldots}\left[\sum_{\tau=0}^{T}\gamma^\tau r^{t+\tau+1}\middle|s^t=s,a^t=a,\pi\right]. \quad (2)$$

It measures the expected total discounted rewards of taking action a in state s and following π thereafter. The Bellman equation for the Q-function is

$$Q^\pi(s^t,a^t) = \sum_{s^{t+1}} p(s^{t+1}|s^t,a^t) \times\left[r^{t+1} + \gamma\sum_{a^{t+1}}\pi(a^{t+1}|s^{t+1})Q^\pi(s^{t+1},a^{t+1})\right]. \quad (3)$$

The optimal policies can achieve the optimal state-value function $V^*(s) = \max_\pi V^\pi(s)$ and action-value function $Q^*(s,a) = \max_\pi Q^\pi(s,a)$ [29]. The Bellman optimality equation for the optimal Q-function is

$$Q^*(s^t,a^t) = \sum_{s^{t+1}} p(s^{t+1}|s^t,a^t)\left[r^{t+1} + \gamma\max_{a^{t+1}}Q^*(s^{t+1},a^{t+1})\right]. \quad (4)$$

A typical DRL method is deep Q-learning (DQN) [30], which is a kind of value-based DRL method. It learns a neural network to approximate the optimal Q-function by minimizing the squared temporal difference error (TD error) [29]. The loss function is defined as

$$L(\theta^Q) = \mathbb{E}\left[\left(r^{t+1} + \gamma\max_{a^{t+1}}Q(s^{t+1},a^{t+1}|\theta^{Q'}) - Q(s^t,a^t|\theta^Q)\right)^2\right] \quad (5)$$

where θ^Q denotes neural network's weights, $\theta^{Q'}$ denotes target network's weights [31], and θ^Q is updated with stochastic gradient descent method. Experience replay is used in DQN as an sample-efficient technique [30]. $\theta^{Q'}$ periodically duplicates θ^Q. Then, it solves the optimal policy as

$$\pi(s) = \arg\max_a Q^*(s,a). \quad (6)$$

This method can solve the optimal policy for the MDPs with discrete action space.

Different from value-based methods, policy-based methods [32] directly optimize a policy according to the gradient of the value function. A policy-based method can learn a continuous policy function parameterized with θ^μ and defined as $a \sim \pi(\cdot|s,\theta^\mu)$ for a stochastic policy or $a = \mu(s|\theta^\mu)$ for a deterministic policy. Taking stochastic policy as an example, the policy gradient is

$$\nabla_{\theta^\mu}V^\pi(s_0) = \mathbb{E}_{s,a}\left[\nabla_{\theta^\mu}\log\pi(a|s,\theta^\mu)Q^\pi(s,a)\right] \quad (7)$$

where s_0 is the initial state.

The actor–critic method [33] integrates value-based methods and policy-based methods. It learns both a Q-function (critic) and a policy function (actor) simultaneously. Based on (3), the critic is learned by minimizing a corresponding squared TD error. The loss function is

$$L(\theta^Q) = \mathbb{E}\left[\left(r^{t+1} + \gamma Q(s^{t+1},a^{t+1}|\theta^Q) - Q(s^t,a^t|\theta^Q)\right)^2\right]. \quad (8)$$

Deep deterministic policy gradient (DDPG) [34] is a kind of actor–critic method, which can solve the MDP with continuous action space and deterministic policy. Its critic is learned to minimize the following loss function:

$$L(\theta^{Q}) = \mathbb{E}\Big[\Big(r^{t+1} + \gamma\, Q\Big(s^{t+1}, \mu\Big(s^{t+1}\big|\theta^{\mu'}\Big)\Big|\theta^{Q'}\Big) - Q\big(s^{t}, a^{t}\big|\theta^{Q}\big)\Big)^{2}\Big] \quad (9)$$

where $\theta^{\mu'}$ and $\theta^{Q'}$ denote target networks. Its actor is optimized with the policy gradient derived as

$$\nabla_{\theta^{\mu}} V^{\pi}(s_0) = \mathbb{E}_{s}\big[\nabla_{\theta^{\mu}}\mu(s|\theta^{\mu})\nabla_{a} Q\big(s, a|\theta^{Q}\big)|_{a=\mu(s|\theta^{\mu})}\big]. \quad (10)$$

DDPG also adopts experience replay for sample efficiency.

C. Independent DQN and Independent DDPG

DQN can be extended to MADRL by independent learning method [35] where each agent treats others as part of the environment. We name this method Ind-DQN. It defines an action-value function for A_i to evaluate its policy

$$Q_i^{\pi_i}(s, a) = \mathbb{E}_{s^{t+1}, a_i^{t+1}, \ldots}\left[\sum_{\tau=0}^{T}\gamma^{\tau} r_i^{t+\tau+1}\big|s^{t} = s, a_i^{t} = a, \pi_i\right] \quad (11)$$

where s^t denotes fully observable state. The Bellman optimality equation for A_i's optimal Q-function is given as

$$Q_i^{\pi_i^{*}}\big(s^{t}, a_i^{t}\big) = \sum_{s^{t+1}} p\big(s^{t+1}\big|s^{t}, a_i^{t}\big)\left[r_i^{t+1} + \gamma \max_{a_i^{t+1}} Q_i^{\pi_i^{*}}\big(s^{t+1}, a_i^{t+1}\big)\right]. \quad (12)$$

$Q_i^{\pi_i^{*}}(s^{t}, a_i^{t})$ is approximated by a neural network with parameters θ_i^{Q}. The loss function is given by

$$L\Big(\theta_i^{Q}\Big) = \mathbb{E}\left[\left(r_i^{t+1} + \gamma \max_{a_i^{t+1}} Q_i\Big(s^{t+1}, a_i^{t+1}\big|\theta_i^{Q'}\Big) - Q_i\Big(s^{t}, a_i^{t}\big|\theta_i^{Q}\Big)\right)^{2}\right]. \quad (13)$$

Similarly, DDPG can also be extended to MADRL, named independent DDPG (Ind-DDPG). The Bellman equation for A_i's Q-function is

$$Q_i^{\pi_i}\big(s^{t}, a_i^{t}\big) = \sum_{s^{t+1}} p\big(s^{t+1}\big|s^{t}, a_i^{t}\big)\big[r_i^{t+1} + \gamma\, Q_i^{\pi_i}\big(s^{t+1}, \mu_i\big(s^{t+1}\big)\big)\big]. \quad (14)$$

A_i's critic parameterized with θ_i^{Q} is learned by minimizing the loss function

$$L\Big(\theta_i^{Q}\Big) = \mathbb{E}\Big[\Big(r_i^{t+1} + \gamma\, Q_i\Big(s^{t+1}, \mu_i\Big(s^{t+1}\big|\theta_i^{\mu'}\Big)\big|\theta_i^{Q'}\Big) - Q_i\Big(s^{t}, a_i^{t}\big|\theta_i^{Q}\Big)\Big)^{2}\Big]. \quad (15)$$

The policy gradient is

$$\nabla_{\theta_i^{\mu}} V_i^{\pi}(s_0) = \mathbb{E}_{s}\left[\nabla_{\theta_i^{\mu}}\mu_i\big(s\big|\theta_i^{\mu}\big)\nabla_{a_i} Q_i\Big(s, a_i\big|\theta_i^{Q}\Big)\big|_{a_i=\mu_i\left(s\middle|\theta_i^{\mu}\right)}\right]. \quad (16)$$

TABLE I

NOTATIONS FREQUENTLY USED IN THIS ARTICLE

A_i	The i-th agent
s	SG's state
s_{-i}	State of agents other than A_i
a_i	A_i's action
a_{-i}	The joint action of agents other than A_i
$\mathbf{o}_{ip}$	Positions of targets and other agents measured by A_i
$\mathbf{o}_{id}$	Ranging results of the detection beams of A_i
$\mathbf{o}_{i}$	Overall observations of A_i. $\mathbf{o}_i = [\mathbf{o}_{ip}, \mathbf{o}_{id}]$
$\mathbf{o}_{i,ag_j}$	Relative position of the j-th agent measured by A_i
$\mathbf{o}_{i,ag_{-i}}$	Relative position of all the other agents measured by A_i
$d_{i,k}$	Ranging result of the k-th detection beam
d_e	Effective detection range
d_s	Safety distance

III. MNUM From SG's Perspective

In this section, we model MNUM as an SG with N agents. The detailed definitions are as follows.

For each agent, its observation of the state is defined as two parts composed of the global observation and the local observation. The global observation, denoted as $\mathbf{o}_{\text{ip}}^{t} = [\mathbf{o}_{i,tar_1}^{t}, \ldots, \mathbf{o}_{i,tar_n}^{t}, \ldots, \mathbf{o}_{i,tar_N}^{t}, \mathbf{o}_{i,\text{ag}_1}^{t}, \ldots, \mathbf{o}_{i,\text{ag}_{j\neq i}}^{t}, \ldots, \mathbf{o}_{i,\text{ag}_N}^{t}]$, includes the relative position coordinates of all targets and other agents. These positions can be known by leveraging positioning systems, such as GPS or indoor positioning systems [36]. The local observation, denoted as $\mathbf{o}_{\text{id}}^{t} = [d_{i,1}^{t}, d_{i,2}^{t}, \ldots, d_{i,K}^{t}]$, includes distances between the agent and its surrounding obstacles in K directions. The distances can be measured by using range finders. An illustration is shown in Fig. 1(b). For notation convenience, A_i's overall observation at time step t is denoted as $\mathbf{o}_i^{t} = [\mathbf{o}_{\text{ip}}^{t}, \mathbf{o}_{\text{id}}^{t}]$. Agents' joint observation is denoted as $\mathbf{o}_{1:N}^{t} = [\mathbf{o}_1^{t}, \ldots, \mathbf{o}_N^{t}]$.

The action of an agent stands for its motion decision. For simplification, we assume that the speed value is constant, and therefore, the action represents a steering angle in a range $[-\Phi, \Phi]$. The policy of an agent is a mapping function from its observation to action. We denote the policy set, including all agents' policies as $\pi_{1:N} = \{\pi_1, \ldots, \pi_N\}$.

Reward is the environment feedback on the state transition after agents' actions. Solving MNUM under sparse reward that only gives a positive value when agents complete a task successfully is a tough issue, because if there is no reward observed, the policy gradient would be zero and no policy improvement would be achieved. In order to circumvent this problem, we propose a properly defined nonsparse reward function by leveraging prior knowledge of the objective of MNUM. For doing this, we first formulate the objective function of MNUM, which is given as

$$\begin{aligned} &\min_{\pi_{1:N}} \mathbb{E}_{\mathbf{o}_{1:N}^{0}, \ldots, \mathbf{o}_{1:N}^{T}}\big[T\big|\mathbf{o}_{1:N}^{0}, \pi_{1:N}\big] \\ &\text{s.t. } \mathbf{o}_{1:N}^{T} \in \mathbf{O}_{\text{success}} \\ &\quad \big\|\mathbf{o}_{i,\text{ag}_j}^{t}\big\|_2 \geq d_s \quad \forall i, j \neq i \quad \forall t \in [0, T] \\ &\quad d_{i,k}^{t} \geq d_s \quad \forall i, k \quad \forall t \in [0, T] \end{aligned} \quad (17)$$

where T denotes the makespan of cooperative navigation and $\mathbf{O}_{\text{success}}$ denotes a set of joint observations when agents complete the task successfully. Success means that agents arrive at different targets without conflict. d_s denotes a safety distance. The physical meaning of the constraints is that to accomplish a cooperative navigation task, agents should arrive at different targets and each agent keeps a safe distance from both other agents and obstacles during the navigation process.

According to the minimization item and three constrains in the objective function (17), we design a reward function composed of four items, which is given as

$$r_i^t = r_{i,\text{step}}^t + r_{i,\text{trans}}^t + r_{i,\text{coll}_1}^t + r_{i,\text{coll}_2}^t. \tag{18}$$

The first two reward items are designed as nonsparse. Specifically, $r_{i,\text{step}}^t = -C_1$ is a negative constant reward given to each agent at each time step until all agents arrive at all targets. $r_{i,\text{trans}}^t$ denotes a reward item related to the first constraint in (17). To get a nonsparse $r_{i,\text{trans}}^t$, we decompose the distance between agents' observations at final time step T and observations at successful state (i.e., $\mathbf{O}_{\text{success}}$) by distance difference and thereby design $r_{i,\text{trans}}^t$ using the distance difference. To be specific, the distance of agents' observations at final time step T and observations $\mathbf{O}_{\text{success}}$ equals the sum of an initial distance and a cumulative differential distance over each two successive time steps, which is given by

$$\varphi\left(\mathbf{o}_{1:N}^T, \mathbf{O}_{\text{success}}\right) = \varphi\left(\mathbf{o}_{1:N}^0, \mathbf{O}_{\text{success}}\right) + \sum_{t=1}^{T}\left(\varphi\left(\mathbf{o}_{1:N}^t, \mathbf{O}_{\text{success}}\right) - \varphi\left(\mathbf{o}_{1:N}^{t-1}, \mathbf{O}_{\text{success}}\right)\right) \tag{19}$$

where $\varphi(\mathbf{o}_{1:N}^0, \mathbf{O}_{\text{success}})$ denotes the initial distance and $\varphi(\mathbf{o}_{1:N}^t, \mathbf{O}_{\text{success}}), t \in [1, T]$ denotes the distance between agents' observations at timesetp t and observations $\mathbf{O}_{\text{success}}$. Because the initial distance is irrelevant to the policy, we use the differential distance to define a per-step transition reward as $r_{i,\text{trans}}^t = -(\varphi(\mathbf{o}_{1:N}^t, \mathbf{O}_{\text{success}}) - \varphi(\mathbf{o}_{1:N}^{t-1}, \mathbf{O}_{\text{success}}))$, which is a positive value if agents draw near to the success state and otherwise is a negative value. Defining the distance function is intractable. Thus, we leverage prior knowledge regarding whether agents draw near to the success state to quantify the transition reward. Specifically, if agents head to different targets, they get closer to the success state generally, and we define $r_{i,\text{trans}}^t = C_2$ as a positive constant in this case. An exceptional case is that agents swap targets frequently but do not get closer to the success states. To discourage this case, C_2 is smaller than C_1, and thereby, the step reward item still penalizes agents. The effect of a bigger C_2 will be demonstrated in Section VI. If more than one agent chooses the same target, they will be farther from the success state. Meanwhile, the more agents choose the same target, the farther they deviate from the success state. Therefore, if A_i chooses the same target with other m agents, $r_{i,\text{trans}}^t = -m \cdot C_3$, where C_3 a positive constant. Note that, rewards are only used to guide policy learning. Once learned, the policy does not rely on rewards. In our work, policies are executed in a decentralized manner.

The last two reward items in (18) are used to penalize the violation of the last two constraints in (17), which is given by

$$\begin{aligned} r_{i,\text{coll}_1}^t &= -C_4 \sum_{j=1, j\neq i}^{N} u\left(d_s - \left\|\mathbf{o}_{i,\text{ag}_j}^t\right\|_2\right) \\ r_{i,\text{coll}_2}^t &= -C_5 \sum_{k=1}^{K} u\left(d_s - d_{i,k}^t\right) \end{aligned} \tag{20}$$

where C_4 and C_5 are positive constants and $u(\cdot)$ denotes the step function. Specifically, in r_{i,coll_1}^t, the value of the step function is 1 if A_i is within the safety distance from A_j; otherwise, it is 0. In r_{i,coll_2}^t, the step function value is 1 if A_i is within the safety distance from the obstacle in the kth observing direction. Thus, the two reward items are used to penalize collisions between an agent and obstacles and other agents. A summary of the reward items is as follows, where we use N_{coll_1} and N_{coll_2} to denote the two summation items in (20)

$r_{i,\text{step}}$:	$-C_1$	r_{i,coll_1}:	$-C_4\ N_{\text{coll}_1}$
$r_{i,\text{trans}}$:	C_2, if agents head to different targets $-mC_3$, otherwise	r_{i,coll_2}:	$-C_5\ N_{\text{coll}_2}$

For convenience, notations frequently used in this article are listed in Table I.

IV. HIST-MADRL FRAMEWORK FOR MNUM

In this section, we address the two aforementioned challenges of solving MNUM with MADRL. We first model a hierarchical policy for MNUM, based on which each agent can only move within a certain range in a target direction, and therefore, the policy solution space can be reduced. Then, we propose a general stable MADRL method to stabilize the concurrent learning of agents. Finally, based on the hierarchical policy model and stable MADRL method, we propose an interlaced learning framework to learn a two-layer policy.

A. Hierarchical Policy Model

Traditional MADRL methods generally model a single policy for each agent, which maps observations to actions. However, for MNUM where targets are not preallocated and the environment contains various obstacles, the policy solution space is so large that solving a single policy is difficult. Actually, a complicated MNUM can be divided into a two-layer problem, where the first layer is a dynamic target selection problem and the second layer is a collision avoidance problem. Based on this idea, we model each agent's policy as a hierarchical policy, which consists of a high-level policy for dynamic target selection and a low-level policy for collision avoidance. According to this policy, a hierarchical control scheme is designed as in Fig. 1(a). At each time step, each agent first selects a target based on its observation of the global state and then rotates the center of its field of view to the selected target and observes the local state. If no obstacle is observed in the direction within the sensing range, the agent will move straight toward the target. Otherwise, the low-level policy is activated to output an angle, and the agent will turn to this angle and move forward.

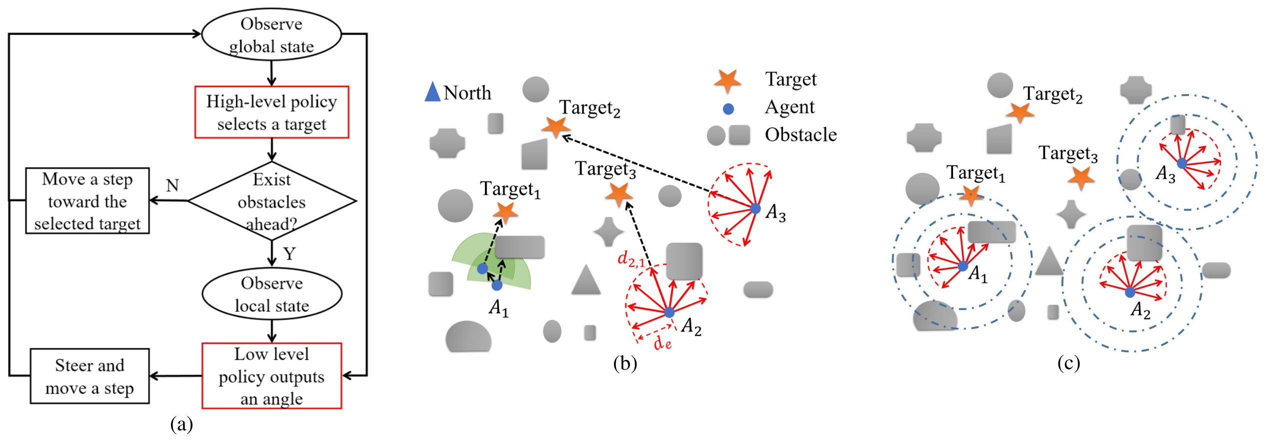


Fig. 1. (a) Hierarchical control scheme. (b) and (c) Comparison of agent's motion range when the agent follows a hierarchical policy and a single policy, respectively. (b) Our hierarchical policy, where each black dashed arrow points to a target selected by the high-level policy. For A_1, we show its two sequential states. In the first state, an obstacle is observed. The black solid arrow shows its moving direction. The green region shows its motion range. (c) Single policy, where circular dotted lines show the boundary of agents' motion range at several sequential time steps.

Based on the hierarchical policy model, each agent only moves within a direction range pointing to its selected target. Besides, when no obstacle is observed in the direction of the selected target, agent will not move to other directions except the direction toward the target, as shown in Fig. 1(b). Thus, trajectories will all draw near to targets and policies that generate trajectories deviating from target directions will not be tried by agents. Namely, policy solution space is reduced.

B. Stabilizing MADRL

As mentioned above, when agents learn concurrently and make decisions independently, from the perspective of each agent, changes of other agents' policies cause nonstationary environment state transitions, and therefore, conventional DRL methods are ineffective. For instance, a direct way to apply DRL to multiagent cases is each agent learning independently and treating others as part of the environment, such as Ind-DQN and Ind-DDPG. In this way, an agent's Q-function can be evaluated based on (12) or (14). Because state transition $p(s^{t+1}|s^t, a_i^t)$ can be decomposed as

$$p\left(s^{t+1}\middle|s^t, a_i^t\right) = \sum_{a_{-i}^t} p\left(a_{-i}^t\middle|s^t\right) p\left(s^{t+1}\middle|s^t, a_i^t, a_{-i}^t\right) \tag{21}$$

where $p(a_{-i}^t|s^t) = \prod_{j\neq i} p(a_j^t|s^t)$ and a_{-i}^t denotes the joint actions of agents except A_i. During learning, changes of agents' policies may cause $p(s^{t+1}|s^t, a_i^t)$ to be nonstationary. Thus, when Q-function is learned iteratively using (13) or (15), the nonstationary component may cause ineffective results.

In fact, because experience replay technique is used in Ind-DQN and Ind-DDPG, this problem will be more severe. Specifically, experience replay uses data sampled randomly from an experience buffer to calculate a sampled gradient of the loss function. Due to time-varying policies of other agents, samples in the experience buffer are generated from diverse distributions corresponding to the past policies. The diversity of sample statistics exacerbates nonstationarity of learning.

To solve this problem, our insight is that the nonstationary component in (21) is other agents' action distributions in a given state, and therefore, we can define an extended Q-function that incorporates other agents' actions so that the action distributions can be removed. Specifically, we define an extended Q-function $Q_i^{\pi_i}(s, a_{-i}, a)$ that measures A_i's expected total rewards when the current state is s, other agents take actions a_{-i}, A_i takes action a_i and then follows π_i thereafter. Then, we can derive a Bellman optimality equation for the optimal extended Q-function of A_i's policy as

$$\begin{aligned} Q_i^{\pi_i^*}&\left(s^t, a_{-i}^t, a_i^t\right) \\ &= \sum_{s^{t+1}} p\left(s^{t+1}\middle|s^t, a_{-i}^t, a_i^t\right) \\ &\quad \times \left[r_i^{t+1} + \gamma \max_{a_i^{t+1}} Q_i^{\pi_i^*}\left(s^{t+1}, a_{-i}^{t+1}, a_i^{t+1}\right)\right]. \end{aligned} \tag{22}$$

Because agents make decisions independently without sharing decisions, other agents' actions are unknown. We address this issue by using an auxiliary variable to replace other agents' true actions. Generally, other agents' states at two adjacent time steps can partially reveal their actions. Thus, we define the auxiliary variable as

$$\kappa_{-i}^t = f\left(s_{-i}^t, s_{-i}^{t+1}\right) \tag{23}$$

where s_{-i}^t denotes the part of global state related with agents other than A_i and f is a function to be learned to implicitly estimate others' actions using their states before and after their actions. During learning, we can leverage the delay of learning to look one step ahead to receive the next state. Besides, due to the uncertainty of state transition, the auxiliary variable can be estimated with its expectation as

$$\widehat{\kappa_{-i}^t} = \mathbb{E}_{s_{-i}^{t+1}|s_{-i}^t, a_{-i}^t} f\left(s_{-i}^t, s_{-i}^{t+1}\right). \tag{24}$$

Based on the auxiliary variable and the estimated auxiliary variable, we can derive an approximate Bellman optimality equation for the optimal extended Q-functions that correspond to agents' optimal policies. Specifically, we first replace a_{-i}^t in the left-hand side of (22) with $\widehat{\kappa^t_{-i}}$. Second, in the right-hand side of (22), leveraging continuity property,[1] we have $a^{t+1}_{-i} \approx a^t_{-i}$ and thereby use κ^t_{-i} to replace a^{t+1}_{-i}. Because s^{t+1} is given, we do not need to use $\widehat{\kappa^t_{-i}}$ to replace κ^t_{-i}. By replacing other agents' actions, we have an approximation of (22), which is given as

$$Q_i^{\pi_i^*}\left(s^t, \mathbb{E}_{s^{t+1}_{-i}|s^t_{-i},a^t_{-i}} f\left(s^t_{-i}, s^{t+1}_{-i}\right), a^t_i\right) \approx \sum_{s^{t+1}} p\left(s^{t+1}|s^t, a^t_{-i}, a^t_i\right) \times \left[r_i^{t+1} + \gamma \max_{a_i^{t+1}} Q_i^{\pi_i^*}\left(s^{t+1}, f\left(s^t_{-i}, s^{t+1}_{-i}\right), a_i^{t+1}\right)\right]. \quad (25)$$

We assume that given s^t_{-i} and a^t_{-i}, $f(s^t_{-i}, s^{t+1}_{-i})$ does not change much and $Q_i^{\pi_i^*}$ is locally linear so that the expectation operator over f can be moved outside the Q-function. Thus, (25) can be transformed as

$$\mathbb{E}_{s^{t+1}_{-i}|s^t_{-i},a^t_{-i}} Q_i^{\pi_i^*}\left(s^t, f\left(s^t_{-i}, s^{t+1}_{-i}\right), a^t_i\right) \approx \mathbb{E}_{s^{t+1}|s^t,a^t_i,a^t_{-i}}\left[r_i^{t+1} + \gamma \max_{a_i^{t+1}} Q_i^{\pi_i^*}\left(s^{t+1}, f\left(s^t_{-i}, s^{t+1}_{-i}\right), a_i^{t+1}\right)\right] \quad (26)$$

which does not include nonstationary components $p(a^t_{-i}|s^t)$. Therefore, we can learn the extended Q-function based on the above formula in a stationary environment. Note that, the extended Q-function to be learned is a composite function that incorporates function f. Thus, we define a G-function as

$$G_i^{\pi_i^*}\left(s^t, s^t_{-i}, s^{t+1}_{-i}, a^t_i\right) = Q_i^{\pi_i^*}\left(s^t, f\left(s^t_{-i}, s^{t+1}_{-i}\right), a^t_i\right). \quad (27)$$

Accordingly, the approximate Bellman optimality equation for G-function is

$$\mathbb{E}_{s^{t+1}_{-i}|s^t_{-i},a^t_{-i}} G_i^{\pi_i^*}\left(s^t, s^t_{-i}, s^{t+1}_{-i}, a^t_i\right) \approx \mathbb{E}_{s^{t+1}|s^t,a^t_i,a^t_{-i}}\left[r_i^{t+1} + \gamma \max_{a_i^{t+1}} G_i^{\pi_i^*}\left(s^{t+1}, s^t_{-i}, s^{t+1}_{-i}, a_i^{t+1}\right)\right]. \quad (28)$$

A neural network denoted as G_i can be learned to approximate G-function by minimizing the squared TD error. The loss function is given by

$$L = \mathbb{E}\left[\left(r_i^{t+1} + \gamma \max_{a_i^{t+1}} G_i\left(s^{t+1}, s^t_{-i}, s^{t+1}_{-i}, a_i^{t+1}\right) - G_i\left(s^t, s^t_{-i}.s^{t+1}_{-i}, a^t_i\right)\right)^2\right]. \quad (29)$$

[1]We define the continuity property as: $(1/T)\sum_{t=1}^{T} |a^{t+1} - a^t| < \epsilon$, where ϵ is a tiny value related with the problem to be solved. It means that actions generated by the optimal policy are not so different at adjacent time steps on average. The continuity property of the optimal policies exists in many applications such as MNUM where the optimal collision avoidance policy can generate a smooth trajectory in order to reduce time consumption.

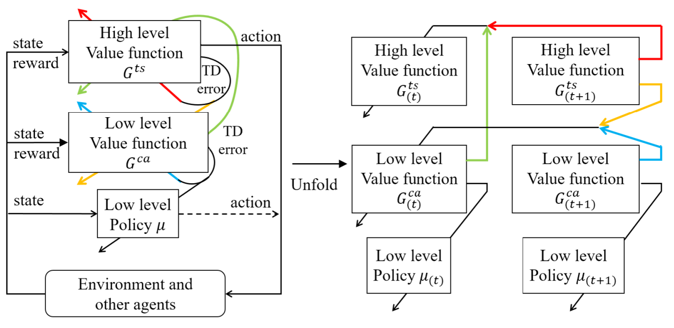

state
reward
High level Value function G^{ts}
action
TD error
state
reward
Low level Value function G^{ca}
TD error
state
Low level Policy μ
action
Environment and other agents
Unfold
High level Value function $G^{ts}_{(t)}$
High level Value function $G^{ts}_{(t+1)}$
Low level Value function $G^{ca}_{(t)}$
Low level Value function $G^{ca}_{(t+1)}$
Low level Policy $\mu_{(t)}$
Low level Policy $\mu_{(t+1)}$

Fig. 2. Framework of HIST-MADRL algorithm. The high-level policy is learned based on SMADQN. The low-level policy is learned based on SMADDPG that has a critic and an actor. Two G-functions are learned in an interlaced way according to whether any obstacle is observed in the selected target direction. The green (red) arrow shows the TD error used to learn G^{ts} when an obstacle is (not) observed in the target direction at the current time step. The blue (yellow) arrow shows the TD error used to learn G^{ca} when an obstacle is (not) observed in the target direction at the next time step.

Similarly, an approximate Bellman expectation equation can be derived as

$$\mathbb{E}_{s^{t+1}_{-i}|s^t_{-i},a^t_{-i}} G_i^{\pi}\left(s^t, s^t_{-i}, s^{t+1}_{-i}, a^t_i\right) \approx \mathbb{E}_{s^{t+1}|s^t,a^t_i,a^t_{-i}}\left[r_i^{t+1} + \gamma\, G_i^{\pi}\left(s^{t+1}, s^t_{-i}, s^{t+1}_{-i}, a_i^{t+1}\right)\right]. \quad (30)$$

To approximate G_i^{π} with a neural network G_i, the loss function is given as

$$L = \mathbb{E}\left[\left(r_i^{t+1} + \gamma\, G_i\left(s^{t+1}, s^t_{-i}, s^{t+1}_{-i}, a_i^{t+1}\right) - G_i\left(s^t, s^t, s^{t+1}_{-i}, a^t_i\right)\right)^2\right]. \quad (31)$$

Based on the above (optimal) G-function learning methods, we design SMADQN and SMADDPG algorithms. During action execution, because future states of other agents are not available, κ^{t-1}_{-i} is used to replace a^t_{-i}. Therefore, in the two algorithms, actions are computed by $a^t_i = \arg\max_{a_i} G^*_i(s^t, s^{t-1}_{-i}, s^t_{-i}, a_i)$ and $a^t_i = \mu_i(s^t, s^{t-1}_{-i}, s^t_{-i})$. Complete algorithms are provided in the Appendix.

C. Interlaced Learning Framework of HIST-MADRL

In this section, we implement SMADQN and SMADDPG to learn the high-level policy and the low-level policy of MNUM, respectively. Because the two policies are executed alternatively, their cumulative reward is dependent on each other. Therefore, we derive the interlaced relationship between their G-functions, based on which an interlaced learning framework is proposed to learn the two policies. The overall algorithm of learning the hierarchical policy for MNUM is named HIST-MADRL.

The overview of HIST-MADRL framework is shown in Fig. 2. Since agents in MNUM are homogeneous, a unified navigation policy is learned for them [37], and therefore, subscripts of G-function and policy function μ are omitted. Specifically, as the high-level policy generates actions for target selection, its action space is discrete. Based on SMADQN, we define an extended Q-function as G^{ts} for this policy. Target selection action is generated by

$a_{i,\text{ts}}^t = \arg\max_{a_{i,\text{ts}}} G^{\text{ts}}(\boldsymbol{o}_{\text{ip}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t-1}, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, a_{i,\text{ts}})$, where $\boldsymbol{o}_{\text{ip}}^t$ is A_i's global observation of the positions of targets and other agents and it corresponds to the fully observable state s^t in SMADQN, and $\boldsymbol{o}_{i,\text{ag}_{-i}}^{t-1}$ and $\boldsymbol{o}_{i,\text{ag}_{-i}}^t$ denote A_i's observation of other agents' positions at two adjacent time steps and they correspond to s_{-i}^{t-1} and s_{-i}^t in SMADQN, respectively. The low-level policy decides a steering direction, and therefore, its action space is continuous. Based on SMADDPG, we define an extended Q-function G^{ca} as a critic and define μ as an actor. Steering action is generated by $a_{i,\text{ca}}^t = \mu(\boldsymbol{o}_{\text{ip}}^t, \boldsymbol{o}_{\text{id}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t-1}, \boldsymbol{o}_{i,\text{ag}_{-i}}^t)$, where $\mathbf{o}_{\text{id}}^t$ is the local state (detection results of the surroundings) defined in Section III.

Because the two policies are executed alternatively, we modify the Bellman equation in (28) and (30) and then derive loss functions to learn G-functions of the two policies. As aforementioned hierarchical control scheme, after target selection, $\mathbf{o}_{\text{id}}^t$ is received. According to whether any obstacle is detected in the target direction, agent will go straight or not and then followed by the high-level decision at the next step or the low-level decision at this step. Therefore, for G^{ts}, its Bellman optimality equation consists of two cases corresponding to whether any obstacle is detected in the target direction. Specifically, if no obstacle is observed in the selected target direction, $a_{i,\text{ts}}^t$ is followed by $a_{i,\text{ts}}^{t+1}$ and $a_{i,\text{ca}}^t = 0$. We denote this case as $d_{i,1}^t \geq d_e$, where $d_{i,1}^t$ denotes the ranging result in the selected target direction and d_e is effective detection range. Thus, corresponding to (28), G^{ts} satisfies

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1} | \boldsymbol{o}_{i,\text{ag}_{-i}}^t, a_{-i}^t} G^{\text{ts}}\left(\boldsymbol{o}_{\text{ip}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ts}}^t\right) \\ &\quad \approx \mathbb{E}_{\boldsymbol{o}_{\text{ip}}^{t+1} | \boldsymbol{o}_{\text{ip}}^t, a_i^t, a_{-i}^t}\left[r_i^{t+1} + \gamma \max_{a_{i,\text{ts}}} G^{\text{ts}}\left(\boldsymbol{o}_{\text{ip}}^{t+1}, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ts}}\right)\right] \end{aligned} \tag{32}$$

where $a_i^t = [a_{i,\text{ts}}^t, a_{i,\text{ca}}^t]$, $a_{-i}^t = [a_{-i,\text{ts}}^t, a_{-i,\text{ca}}^t]$. If an obstacle is observed in the direction of the selected target, i.e., $d_{i,1}^t < d_e$, then $a_{i,\text{ts}}^t$ is followed by $a_{i,\text{ca}}^t$ and G^{ts} satisfies

$$\begin{aligned}&G^{\text{ts}}\left(\boldsymbol{o}_{\text{ip}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ts}}^t\right) \\ &\quad \approx \mathbb{E}_{\mathbf{o}_{\text{id}}^t | \boldsymbol{o}_{\text{ip}}^t, a_{i,\text{ts}}^t} G^{\text{ca}}\left(\boldsymbol{o}_i^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}, \mu\left(\boldsymbol{o}_i^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}\right)\right). \end{aligned} \tag{33}$$

Based on this formula, G^{ts} can be learned by minimizing the loss function given by

$$L_{\text{ts}} = \mathbb{E}\left[\left(y_{i,\text{ts}}^t - G^{\text{ts}}\left(\boldsymbol{o}_{\text{ip}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ts}}^t\right)\right)^2\right] \tag{34}$$

where $y_{i,\text{ts}}^t$ is defined based on (32) and (33), which is given as

$$y_{i,\text{ts}}^t = \begin{cases} G^{\text{ca}'}\left(\mathbf{o}_i^t, \mathbf{o}_{i,\text{ag}_{-i}}^t, \mathbf{o}_{i,\text{ag}_{-i}}^{t+1}, \mu'\left(\mathbf{o}_i^t, \mathbf{o}_{i,\text{ag}_{-i}}^t, \mathbf{o}_{i,\text{ag}_{-i}}^{t+1}\right)\right), \\ \quad \text{if } d_{i,1}^t < d_e \\ r^{t+1} + \gamma \max_{a_{i,\text{ts}}} G^{\text{ts}'}\left(\mathbf{o}_{\text{ip}}^{t+1}, \mathbf{o}_{i,\text{ag}_{-i}}^t, \mathbf{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ts}}\right) \\ \quad \text{otherwise} \end{cases} \tag{35}$$

where $G^{\text{ts}'}$, $G^{\text{ca}'}$ and μ' are target networks.

For G^{ca}, as the collision avoidance action $a_{i,\text{ca}}^t$ is followed by the target selection action $a_{i,\text{ts}}^{t+1}$, its Bellman optimality equation can be derived as

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1} | \boldsymbol{o}_{i,\text{ag}_{-i}}^t, a_{-i}^t} G^{\text{ca}}\left(\mathbf{o}_i^t, \mathbf{o}_{i,\text{ag}_{-i}}^t, \mathbf{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ca}}^t\right) \\ &\quad \approx \mathbb{E}_{\boldsymbol{o}_{\text{ip}}^{t+1} | \boldsymbol{o}_{\text{ip}}^t, a_i^t, a_{-i}^t}\left[r^{t+1} + \gamma \max_{a_{i,\text{ts}}} G^{\text{ts}}\left(\mathbf{o}_{\text{ip}}^{t+1}, \mathbf{o}_{i,\text{ag}_{-i}}^t, \mathbf{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ts}}\right)\right]. \end{aligned} \tag{36}$$

Furthermore, if an obstacle is observed at the next time step, we can leverage (33) to replace the maximized G^{ts} on the right-hand side of this equation with the expectation of G^{ca}. In this way, the estimation bias propagation from G^{ts} to G^{ca} can be avoided. Specifically, (36) is transformed as

$$\begin{aligned}&\mathbb{E}\left[G^{\text{ca}}\left(\mathbf{o}_i^t, \mathbf{o}_{i,\text{ag}_{-i}}^t, \mathbf{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ca}}^t\right)\right] \\ &\quad \approx \mathbb{E}\Bigg[r^{t+1} + \gamma\, \mathbb{E}_{\mathbf{o}_{\text{id}}^{t+1} | \boldsymbol{o}_{\text{ip}}^{t+1}, a_{i,\text{ts}}^{t+1}} \\ &\qquad\qquad \left[G^{\text{ca}}\left(\boldsymbol{o}_i^{t+1}, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}, \mu\left(\boldsymbol{o}_i^{t+1}, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}\right)\right)\right]\Bigg] \end{aligned} \tag{37}$$

where $a_{i,\text{ts}}^{t+1} = \arg\max_{a_{i,\text{ts}}} G^{\text{ts}}(\mathbf{o}_{\text{ip}}^{t+1}, \mathbf{o}_{i,\text{ag}_{-i}}^t, \mathbf{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ts}})$. Therefore, G^{ca} can be learned by minimizing the loss function given by

$$L_{\text{ca}} = \mathbb{E}\left[\left(y_{i,\text{ca}}^t - G^{\text{ca}}\left(\boldsymbol{o}_i^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ca}}^t\right)\right)^2\right] \tag{38}$$

where $y_{i,\text{ca}}^t$ is defined as follows based on (36) and (37):

$$\begin{aligned}&y_{i,\text{ca}}^t \\ &= \begin{cases} r^{t+1} + \gamma\, G^{\text{ca}'}\left(\boldsymbol{o}_i^{t+1}, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}, \mu'\left(\boldsymbol{o}_i^{t+1}, \boldsymbol{o}_{i,\text{ag}_{-i}}^t, \boldsymbol{o}_{i,\text{ag}_{-i}}^{t+1}\right)\right) \\ \quad \text{if } d_{i,1}^{t+1} < d_e \\ r^{t+1} + \gamma \max_{a_{i,\text{ts}}} G^{\text{ts}'}\left(\mathbf{o}_{\text{ip}}^{t+1}, \mathbf{o}_{i,\text{ag}_{-i}}^t, \mathbf{o}_{i,\text{ag}_{-i}}^{t+1}, a_{i,\text{ts}}\right) \\ \quad \text{otherwise.} \end{cases}\end{aligned} \tag{39}$$

During learning, we adopt experience replay to enhance sample efficiency. A complete HIST-MADRL algorithm for MNUM is provided in Algorithm 1.

V. Simulation Results

In this section, we evaluate our proposed methods with simulation experiments on the multiagent cooperative navigation tasks. In our experiment, we pretrain the target selection policy using SMADQN in barrier-free environments. In this process, we evaluate the performance of SMADQN. Then, the pretrained target selection policy is used as a warm start, based on which the hierarchical policy is trained with HIST-MADRL in environments with unknown and randomly placed obstacles.

Algorithm 1 Hierarchical and Stable MADRL for MNUM

```
1: Initialize networks $G^{\mathrm{ts}}$, $G^{\mathrm{ca}}$ and $\mu$ with random weights $\theta^{\mathrm{ts}}$, $\theta^{\mathrm{ca}}$, $\theta^{\mu}$
2: Initialize the target networks $G^{\mathrm{ts}'}$, $G^{\mathrm{ca}'}$ and $\mu'$ with weights $\theta^{\mathrm{ts}'} \leftarrow \theta^{\mathrm{ts}}$,
   $\theta^{\mathrm{ca}'} \leftarrow \theta^{\mathrm{ca}}$, $\theta^{\mu'} \leftarrow \theta^{\mu}$
3: Initialize replay buffer $\mathcal{D}^{\mathrm{ts}}$, $\mathcal{D}^{\mathrm{ca}}$
4: for episode= 1 to $Z$ do
5:     Receive initial state $\mathbf{o}_{\mathrm{ip}}^{1}$ for each agent, $\mathbf{o}_{\mathrm{ip}}^{0} \leftarrow \mathbf{o}_{\mathrm{ip}}^{1}$
6:     for $t = 1$ to $T$ do
7:         for agent $i = 1$ to $N$ do
8:             Execute high-level policy to select a target:
               $a_{i,\mathrm{ts}}^{t} = \arg\max_{a_{i,\mathrm{ts}}} G^{\mathrm{ts}}(\boldsymbol{o}_{\mathrm{ip}}^{t}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{t-1}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{t}, a_{i,\mathrm{ts}})$
9:             Compute the direction angle $\rho_{\mathrm{ts}}^{t}$ of target $a_{i,\mathrm{ts}}^{t}$
10:            Rotate $\rho_{\mathrm{ts}}^{t}$, detect and receive $\mathbf{o}_{\mathrm{id}}^{t}$
11:            if $d_{i,1}^{t} > d_e$ then
12:                Move a step toward target $a_{i,\mathrm{ts}}^{t}$
13:            else
14:                Execute low-level policy to compute a steering angle:
                   $a_{i,\mathrm{ca}}^{t} = \mu(\boldsymbol{o}_{i}^{t}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{t-1}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{t})$
15:                Rotate $a_{i,\mathrm{ca}}^{t}$ and move a step
16:            end if
17:        end for
18:        for agent $i = 1$ to $N$ do
19:            Receive $r_{i}^{t}$, $\mathbf{o}_{\mathrm{ip}}^{t+1}$
20:            Execute high-level policy to select a target:
               $a_{i,\mathrm{ts}}^{t+1} = \arg\max_{a_{i,\mathrm{ts}}} G^{\mathrm{ts}}(\boldsymbol{o}_{\mathrm{ip}}^{t+1}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{t}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{t+1}, a_{i,\mathrm{ts}})$
21:            Receive $\mathbf{o}_{\mathrm{id}}^{t+1}$
22:            Store transition $(\mathbf{o}_{\mathrm{ip}}^{t}, a_{i,\mathrm{ts}}^{t}, r_{i}^{t}, \mathbf{o}_{i}^{t+1})$ in $\mathcal{D}^{\mathrm{ts}}$
23:            if $d_{i,1}^{t} \le d_e$ then
24:                Store transition $(\mathbf{o}_{i}^{t}, a_{i,\mathrm{ca}}^{t}, r_{i}^{t}, \mathbf{o}_{i}^{t+1})$ in $\mathcal{D}^{\mathrm{ca}}$
25:            end if
26:        end for
27:        Sample $M$ transitions $\{(\mathbf{o}_{\mathrm{ip}}^{j}, a_{i,\mathrm{ts}}^{j}, r_{i}^{j}, \mathbf{o}_{i}^{j+1})\}_{j=1}^{M}$ from $\mathcal{D}^{\mathrm{ts}}$
28:        for $j = 1$ to $M$ do
29:            Set $y_{i,\mathrm{ts}}^{j}$ according to (35)
30:        end for
31:        Update $\theta^{\mathrm{ts}}$ by minimizing:
           $L_{\mathrm{ts}} = \frac{1}{M}\sum_j (y_{i,\mathrm{ts}}^{j} - G^{\mathrm{ts}}(\boldsymbol{o}_{\mathrm{ip}}^{j}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{j}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{j+1}, a_{i,\mathrm{ts}}^{j}))^2$
32:        Sample $M$ transitions $\{(\mathbf{o}_{i}^{j}, a_{i,\mathrm{ca}}^{j}, r_{i}^{j}, \mathbf{o}_{i}^{j+1})\}_{j=1}^{M}$ from $\mathcal{D}^{\mathrm{ca}}$
33:        for $j = 1$ to $M$ do
34:            Set $y_{i,\mathrm{ca}}^{j}$ according to (39)
35:        end for
36:        Update $\theta^{\mathrm{ca}}$ by minimizing:
           $L_{\mathrm{ca}} = \frac{1}{M}\sum_j (y_{i,\mathrm{ca}}^{j} - G^{\mathrm{ca}}(\boldsymbol{o}_{i}^{j}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{j}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{j+1}, a_{i,\mathrm{ca}}^{j}))^2$,
37:        Update $\theta^{\mu}$ using sampled gradient:
   $\nabla_{\theta^{\mu}} J \approx \frac{1}{M}\sum_j$
   $\nabla_{a_{i,\mathrm{ca}}} G^{\mathrm{ca}}(\boldsymbol{o}_{i}^{j}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{j}, \boldsymbol{o}_{i,\mathrm{ag}_{-i}}^{j+1}, a_{i,\mathrm{ca}})|_{a_{i,\mathrm{ca}}=\mu(\mathbf{o}_{i}^{j}, \mathbf{o}_{i,\mathrm{ag}_{-i}}^{j-1}, \mathbf{o}_{i,\mathrm{ag}_{-i}}^{j})}$
   $\nabla_{\theta^{\mu}} \mu(\mathbf{o}_{i}^{j}, \mathbf{o}_{i,\mathrm{ag}_{-i}}^{j-1}, \mathbf{o}_{i,\mathrm{ag}_{-i}}^{j})$
38:        Update the target networks ($\eta$ is the soft target update rate):
           $\theta^{\mathrm{ts}'} \leftarrow \eta\theta^{\mathrm{ts}} + (1-\eta)\theta^{\mathrm{ts}'}$
           $\theta^{\mathrm{ca}'} \leftarrow \eta\theta^{\mathrm{ca}} + (1-\eta)\theta^{\mathrm{ca}'}$
           $\theta^{\mu'} \leftarrow \eta\theta^{\mu} + (1-\eta)\theta^{\mu'}$
39:    end for
40: end for
```

A. Experimental Settings

The environment constructed for the pretraining process is a small barrier-free space, 15×15 m^2, for a brief warm-up training. At the beginning of each training episode, N target locations and N departure locations of agents are generated randomly. All coordinate values of the locations are normalized to $[0, 1]$. In the hierarchical policy training process, the size of the environment is set as 36×36 m^2. At the beginning of each training episode, in addition to randomly generated locations of targets and agents, ten obstacles are randomly placed in the environment. Each obstacle is shaped as a circle or a square randomly, whose diameter or side length is generated by a uniform distribution $U(1\ \mathrm{m}, 4\ \mathrm{m})$. During navigation, each agent moves at a constant speed $v = 1$ m/time step with a varied steering angle in the range $[-(\pi/2), (\pi/2)]$ while detecting surrounding obstacles using seven ranging beams in directions that evenly separate the moving range. The effective detection range is set as 4 m. The maximum episode length (time step) is set as twice as the environment's side length. The rewards are instantiated as $r_{i,\mathrm{step}} = -1/L$, $r_{i,\mathrm{coll}_1} = r_{i,\mathrm{coll}_2} = -45/L$, $C_2 = 0.8/L$, and $C_3 = -30/L$, which are normalized by the environment's side length L. The discount factor is $\gamma = 1$.

The extended Q-function of the target selection policy, G^{ts}, is estimated by a neural network with two hidden layers containing 300 and 200 units, respectively. RMSProp optimizer is applied to learn the parameters of this network with a learning rate of 0.01. The capacity of the replay memory is 1500 and the batch size of the SGD is 32. For the collision avoidance policy, the critic G^{ca} and actor μ are constructed as two networks both with two hidden layers containing 100 units in each layer. The actor's output layer contains one unit activated by a tanh function multiplied by $\pi/2$. Adam optimizer is adopted to learn the parameters of the actor and critic. The learning rate is 0.001 and 0.002. The capacity of the replay memory is 3000 and the batch size of SGD is 32. All target networks are updated with update rate $\eta = 0.001$.

B. Performance of SMADQN

We apply SMADQN to pretrain the high-level policy with N agents ($N = 2, 3, 4, 5$, and 6) in barrier-free environments. During this phase, agents only learn to cooperatively select targets in order to arrive at all targets from any departure locations using minimum time. Collisions between agents are not considered during this training phase. We compare our method with Ind-DQN [35] and the fingerprint-based method [24]. Hyperparameters of the two methods are identical to ours. Convergence curves of the average episode reward are shown in the left column of Fig. 3. We can see that our method converges to similar average episode rewards when a different number of agents are involved in tasks. As the number of agents increases, the convergence speed slows down and the average episode reward decreases. We consider that the decrease in convergence speed results from the increase in state space and action space. Besides, when more agents are involved in a task, agents need more time to coordinate with each other and the task success rate decreases, which causes the decrease in average episode reward. By contrast, the performance of Ind-DQN drops drastically as the number of agents increases. When $N > 4$, Ind-DQN gets less average episode reward compared with our method, and the variances of its convergence curves are larger than ours. When $N \le 4$, Ind-DQN can finally achieve an episode reward similar to ours. However, its convergence speed is much slower than ours. Compared with the fingerprint-based method, our method still holds the superiority of faster convergence speed and higher reward. We believe that the fingerprint-based method can alleviate the nonstationarity by using the fingerprints to identify part of the change in other agents' policies, but it cannot distinguish the degree and the direction of their policy change, which limits the performance of this method.

Convergence curves of normalized average maximum navigation time are shown in the right column of Fig. 3. The maximum navigation time refers to the time cost by the agent who is the last one to arrive at a target. If an episode ends with a collision, the maximum navigation time is recorded as the maximum episode length. As we can see, compared with Ind-DQN and the fingerprint-based method, the average maximum navigation time of our method decreases rapidly as learning proceeds. In particular, Ind-DQN can hardly succeed in the training episodes with more than four agents.

Apart from the convergence performance, we also evaluate the performance of the policies after training. Given the number of agents ($N = 2, 3, 4, 5,$ and 6), 1000 tasks with random target locations and random agent departure locations are generated to test the policy. From the results in Table II, we can see that the success rate of our method exceeds 90% in all circumstances and exceeds 95% when $N \leq 5$. By contrast, Ind-DQN and the fingerprint-based method only yield good policies when $N \leq 4$. Specifically, Ind-DQN performs relatively well when $N \leq 4$, but compared with our method, its success rate is lower and average maximum navigation time is longer. When $N > 4$, its success rate drops below 5%. The fingerprint-based method yields as good policies as ours when $N \leq 4$, but its success rate decreases sharply when $N > 4$. As we can see from the results, compared with Ind-DQN, our method achieves at most 88.6% improvement in success rate and 4.9% reduction in average maximum navigation time. Compared with the fingerprint-based method, our method achieves at most 72.9% improvement in success rate and 2% reduction in average maximum navigation time.

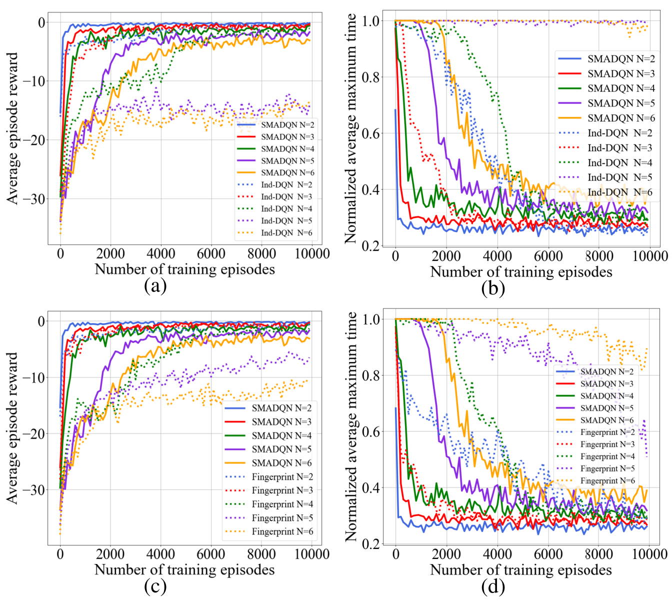


Fig. 3. Convergence curves of average episode reward gained by different methods (the left column) and convergence curves of normalized average maximum navigation time cost by different methods (the right column). (a) and (b) Comparison between SMADQN (ours) and Ind-DQN, respectively. (c) and (d) Comparison between SMADQN (ours) and the fingerprint-based method, respectively.

TABLE II

Test Results of the Policies Learned With SMADQN (Ours), Ind-DQN, and the Fingerprint-Based Method

N	Success rate			Normalized average maximum time		
	SMADQN	Ind-DQN	Finger-print	SMADQN	Ind-DQN	Finger-print
2	**99.9%**	99.6%	99.6%	0.517	0.516	0.521
3	**98.2%**	97.7%	98.3%	0.537	0.541	0.548
4	**97.8%**	94.7%	96.3%	0.560	0.589	0.563
5	**96.1%**	4.0%	66.1%	0.581	0.603	0.585
6	**91.2%**	2.6%	18.3%	0.589	0.613	0.596

TABLE III

Test Results of the Policies Learned With Our Reward, Bigger Transition Reward, and Sparse Transition Reward

N	Success rate			Normalized average maximum time		
	Ours	Bigger $r_{i,trans}$	Sparse $r_{i,trans}$	Ours	Bigger $r_{i,trans}$	Sparse $r_{i,trans}$
2	99.9%	98.4%	95.3%	0.517	0.523	0.551
3	98.2%	90.5%	4.0%	0.537	0.580	0.684
4	97.8%	79.4%	0.0%	0.560	0.617	1.000
5	96.1%	61.0%	0.0%	0.580	0.623	1.000
6	91.2%	60.9%	0.0%	0.589	0.631	1.000

C. Effect of Reward

We investigate the effect of different reward settings on policy learning. Specifically, we modify our reward function in two ways. In the first modified reward setting, we replace the transition reward used in our reward function with a sparse one, which is set as $r^t_{i,\text{trans}} = 15 \cdot I_{\mathbf{O}_{\text{success}}}(\mathbf{o}^T_{1:N})$ with $I_{\mathbf{O}_{\text{success}}}(\mathbf{o}^T_{1:N})$ defined as

$$I_{\mathbf{O}_{\text{success}}}\left(\mathbf{o}^T_{1:N}\right) = \begin{cases} 1, & \text{if } \mathbf{o}^T_{1:N} \in \mathbf{O}_{\text{success}} \\ 0, & \text{otherwise.} \end{cases} \quad (40)$$

We denote this reward setting as "sparse $r_{i,\text{trans}}$." In the second modified reward setting, we set the transition reward given to agents when they select different targets as $C_2 = 50/L$, which is bigger than the absolute value of $r^t_{i,\text{step}}$, i.e., $1/L$. We denote this reward setting as "bigger $r_{i,\text{trans}}$."

We conduct experiments to train the high-level policy with the two reward settings in pretraining environments. For both reward settings, training tasks are generated in the same way as our pretraining process. Policies are both trained using SMADQN. After training 10 000 episodes, we test the two learned policies with 1000 randomly generated tasks. Results are listed in Table III. We can see that compared with our method, the success rate of the policies learned with bigger transition reward drops a lot as the number of agents increases. Besides, the average maximum navigation time of the bigger transition reward is longer than ours. We believe that the result implies that guided by bigger transition reward, agents learn to change targets frequently to get more reward despite costing more time. For the sparse transition reward, it performs well only when $N = 2$. When $N = 3$, its success rate drops drastically. When $N \geq 4$, it cannot succeed even one time. The result verifies that the sparse transition reward cannot

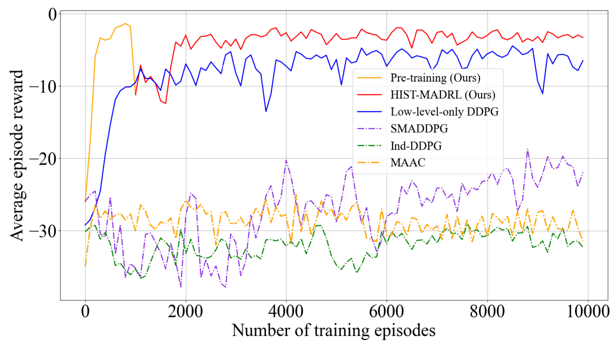


Fig. 4. Convergence curves of different methods of solving MNUM involving three agents in environments with obstacles.

TABLE IV
TEST RESULTS OF THE POLICIES LEARNED WITH HIST-MADRL (OURS) AND LOW-LEVEL-ONLY DDPG

N	Success rate		Normalized average maximum time	
	HIST-MADRL	Low-level-only DDPG	HIST-MADRL	Low-level-only DDPG
2	99.2%	95.4%	0.587	0.791
3	96.1%	94.5%	0.613	0.792
4	94.8%	92.3%	0.632	0.820
5	91.2%	87.7%	0.709	0.853
6	87.9%	77.5%	0.753	0.960

give efficient guidance for policy learning, especially when the number of agents increases and the task becomes harder.

D. Performance of HIST-MADRL

To validate the effectiveness of our proposed HIST-MADRL method, we apply it to learn policies for MNUM in environments with obstacles. To illustrate the superiority of HIST-MADRL over single-policy learning method, we compare it with three single-policy learning methods where the policy only decides a steering angle rather than selects a target and then decides to steer. The first one is Ind-DDPG, in which we extend DDPG [34] to multiagent DRL with independent learning method. The second one is SMADDPG, which is our proposed stable learning algorithm for learning policies with continuous action space in a multiagent setting. The third one is MAAC [28]. It adopts an attention mechanism to learn a critic that selects and incorporates other agents' information. In addition, to investigate the effectiveness of HIST-MADRL in optimizing both layers of the policy, we compare it with a half-trained hierarchical policy learning method, where only the low-level policy is learned. Specifically, the high-level policy adopts a centralized target assignment policy to randomly assign a target for each agent at the beginning of an episode and targets are not reassigned in the navigation process. The low-level policy is learned with DDPG to avoid collisions and does not distinguish whether an obstacle is observed in the direction of the target. We name this method low-level-only DDPG.

Fig. 4 shows the convergence curves of the average episode reward gained by different methods in tasks involving three targets. As we can see, after pretraining the high-level policy for a short time, HIST-MADRL converges after 1000 training episodes. In particular, due to the penalty for collisions and

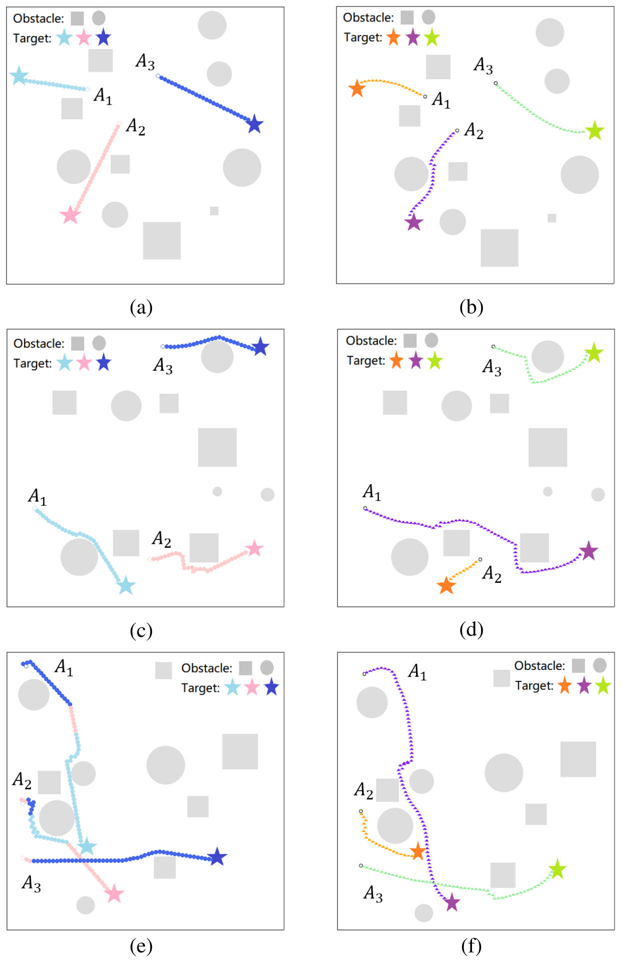


Fig. 5. Navigation trajectories generated by (a), (c), and (e) HIST-MADRL and (b), (d), and (f) low-level-only DDPG.

time cost by collision avoidance in the HIST-MADRL training phase, the average episode reward drops after the pretraining phase. By contrast, low-level-only DDPG nearly converges after 2000 episodes, which is slower than HIST-MADRL but similar to HIST-MADRL plus the pretraining phase. In addition, it gets less average episode reward than HIST-MADRL, which indicates the effectiveness of HIST-MADRL in learning both hierarchies of the policy. For the three single-policy learning methods, they all get fairly low reward. Specifically, Ind-DDPG yields the worst performance. We believe that the ineffectiveness of Ind-DDPG results from both the nonstationary environment and the large policy solution space. By contrast, SMADDPG outperforms Ind-DDPG, which validates the effectiveness of our stable DRL method. Nevertheless, the convergence speed of SMADDPG is much lower than the two hierarchical policy learning methods, which indicates the effectiveness of our hierarchical policy model in accelerating learning. MAAC also fails to learn a good policy. This is because MAAC does not deal with the difficulty caused by large policy solution space. Although agents share information about their observations and actions, the policy solution space is not reduced in MAAC, which hinders efficient learning.

To evaluate the policy performance of our method, we test the learned policies with randomly generated tasks. Given the

Algorithm 2 SMADQN

```
1: for agent $i = 1$ to $N$ do
2:     Initialize networks $G_i(s, s_{-i}, s'_{-i}, a_i|\theta_i)$ with random weights $\theta_i$
3:     Initialize target networks $G_i^{tar}(s', s_{-i}, s'_{-i}, a_i|\theta_i^{tar})$ with
       $\theta_i^{tar} \leftarrow \theta_i$
4:     Initialize replay buffer $\mathcal{D}_i$
5: end for
6: for episode $= 1$ to $Z$ do
7:     Receive initial state $s^1$ for each agent, $s^0 \leftarrow s^1$
8:     for $t = 1$ to $T$ do
9:         for agent $i = 1$ to $N$ do
10:            Execute action $a_i^t = \arg\max_{a_i} G_i(s^t, s_{-i}^{t-1}, s_{-i}^t, a_i|\theta_i)$
11:        end for
12:        for agent $i = 1$ to $N$ do
13:            Receive $r_i^{t+1}$, $s^{t+1}$
14:            Store transition $(s^t, a_i^t, r_i^{t+1}, s^{t+1})$ in $\mathcal{D}_i$
15:            Sample $M$ transitions $\{(s^j, a_i^j, r_i^{j+1}, s^{j+1})\}_{j=1}^M$ from$\mathcal{D}_i$
16:            for $j = 1$ to $M$ do
17:                Set:
               $y_i^j = r_i^{j+1} + \gamma \max_{a_i} G_i^{tar}(s^{j+1}, s_{-i}^j, s_{-i}^{j+1}, a_i|\theta_i^{tar})$
18:            end for
19:            Update $\theta_i$ using the sampled gradient:
               $\nabla_{\theta_i} L(\theta_i) \approx \frac{1}{M}\sum_j\Big[(y_i^j - G_i(s^j, s_{-i}^j, s_{-i}^{j+1}, a_i^j|\theta_i))$
                   $\nabla_{\theta_i} G_i(s^j, s_{-i}^j, s_{-i}^{j+1}, a_i^j|\theta_i)\Big]$
20:            Update the target networks ($\eta$ is the soft target update rate):
                   $\theta_i^{tar} \leftarrow \eta\theta_i + (1-\eta)\theta_i^{tar}$
21:        end for
22:    end for
23: end for
```

number of agents ($N = 2, 3, 4, 5, 6$), 1000 test tasks are generated. In each task, the size of each obstacle and the locations of the obstacles, targets, and agents are randomly generated. Success rate and normalized average maximum time are listed in Table IV. It is remarkable that our method achieves a high success rate of more than 90% when $N < 6$. As a comparison, we test low-level-only DDPG in the same way due to its comparable performance with ours which is shown in Fig. 4. As we can see from the result in Table IV, HIST-MADRL has the potential to achieve 10.4% improvement in success rate and 21.6% reduction in average maximum time. We believe that the superiority of our method comes from the capability of selecting targets dynamically and cooperatively. Therefore, agents can cooperatively select targets to reduce the maximum navigation time and change targets with others to avoid being trapped by obstacles or moving into a collision-prone area, which increases the success rate.

To investigate the navigation behavior of the learned policies, three typical tasks involving three targets and three agents are selected to demonstrate agents' navigation trajectories. The results are shown in Fig. 5. For comparison, the results of Low-level-only DDPG are shown in the right column. For visualizing the results of target selection at each time step, the trajectory of each step is painted with the same color as that of the selected target. From the trajectories in the first task shown in the top row, we can see that agents trained by HIST-MADRL can select different targets throughout the navigation process. In addition, in contrast to the curving trajectories generated by low-level-only DDPG, HIST-MADRL can go straight toward a target when no obstacles hinder ahead. This is because our method uses a high-level policy to select a target before steering and combines the empirical knowledge of going ahead when being not blocked. In the second task, we can see that our method can successfully avoid the obstacles during navigation. In addition, compared with the result of randomly target assignment in low-level-only DDPG, in our method, A_2 did not select the light blue target that is the closest to it, but left it to A_1. The longest trajectory length reflects the maximum navigation time. Comparing the longest trajectory of the two methods, we can find that our method enables agents to cooperatively select targets so that they can complete the task with less time. This cooperation capability emerges prominently in the last task. In this task, the closest targets to three agents at the beginning are all the pink one. At the initial several time steps, A_2 and A_3 both head to the pink target. As they move forward and avoid obstacles on their ways, their target selections change dynamically and cooperatively. Finally, all agents successfully arrive at different targets. According to the longest trajectory length, we can see that our method costs less time than low-level-only DDPG.

Algorithm 3 SMADDPG

```
1: for agent $i = 1$ to $N$ do
2:     Initialize networks $G_i(s, s_{-i}, s'_{-i}, a_i|\theta_i^G)$, $\mu_i(s, s_{-i}, s'_{-i}|\theta_i^\mu)$
        with random weights $\theta_i^G$, $\theta_i^\mu$
3:     Initialize target networks $G_i^{tar}$, $\mu_i^{tar}$ with weights
        $\theta_i^{G'} \leftarrow \theta_i^G$, $\theta_i^{\mu'} \leftarrow \theta_i^\mu$
4:     Initialize replay buffer $\mathcal{D}_i$
5: end for
6: for episode $= 1$ to $Z$ do
7:     Receive initial state $s^1$ for each agent, $s^0 \leftarrow s^1$
8:     for $t = 1$ to $T$ do
9:         for agent $i = 1$ to $N$ do
10:            Execute action $a_i^t = \mu_i(s^t, s_{-i}^{t-1}, s_{-i}^t|\theta_i^\mu)$
11:        end for
12:        for agent $i = 1$ to $N$ do
13:            Receive $r_i^{t+1}$, $s^{t+1}$
14:            Store transition $(s^t, a_i^t, r_i^{t+1}, s^{t+1})$ in $\mathcal{D}_i$
15:            Sample $M$ transitions $\{(s^j, a_i^j, r_i^{j+1}, s^{j+1})\}_{j=1}^M$ from$\mathcal{D}_i$
16:            for $j = 1$ to $M$ do
17:                Set: $y_i^j = r_i^{j+1} +$
               $\gamma G_i^{tar}(s^{j+1}, s_{-i}^j, s_{-i}^{j+1}, \mu(s^{j+1}, s_{-i}^j, s_{-i}^{j+1}|\theta_i^{\mu'})|\theta_i^{G'})$
18:            end for
19:            Update $\theta_i^G$ using the sampled gradient:
               $\nabla_{\theta_i^G} L(\theta_i^G) \approx \frac{1}{M}\sum_j\Big[(y_i^j - G_i(s^j, s_{-i}^j, s_{-i}^{j+1}, a_i^j|\theta_i^G))$
                   $\nabla_{\theta_i^G} G_i(s^j, s_{-i}^j, s_{-i}^{j+1}, a_i^j|\theta_i^G)\Big]$
20:            Update $\theta_i^\mu$ using the sampled gradient:
                $\nabla_{\theta_i^\mu} J \approx \frac{1}{M}\sum_j$
                $\nabla_{a_i} G_i(s^j, s_{-i}^j, s_{-i}^{j+1}, a_i|\theta_i^G)|_{a_i=\mu_i(s^j, s_{-i}^{j-1}, s_{-i}^j|\theta_i^\mu)}$
                $\nabla_{\theta_i^\mu} \mu(s^j, s_{-i}^{j-1}, s_{-i}^j|\theta_i^\mu)$
21:            Update the target networks ($\eta$ is the soft target update rate):
                   $\theta_i^{G'} \leftarrow \eta\theta_i^G + (1-\eta)\theta_i^{G'}$
                   $\theta_i^{\mu'} \leftarrow \eta\theta_i^\mu + (1-\eta)\theta_i^{\mu'}$
22:        end for
23:    end for
24: end for
```

VI. Conclusion

In this work, we solve a cooperative navigation control problem, MNUM, where multiagent needs to navigate to unassigned multiple targets in environments containing various unknown obstacles. We model MNUM as an SG and design a reward function based on its objective function. A novel hierarchical and stable MADRL method is proposed to solve the problem, which learns a high-level policy for target selection and a low-level policy for collision avoidance in an interlaced manner. Based on the hierarchical policy model, our method reduces the policy solution space and accelerates the learning process. To stabilize learning, an auxiliary variable incorporating the information of other agents' actions is introduced as guidance for policy learning. Extensive experimental results demonstrate the efficiency and superiority of our method in terms of both convergence speed and agents' cooperation performance. There are also some issues worthy of investigation in depth. For instance, the effect of observation errors on policy learning and the tolerance of our method to these errors need to be studied. Besides, we will further investigate the applicability of our proposed SMADQN, SMADDPG, and HIST-MADRL algorithms in other multiagent control problems.

Appendix

See Algorithms 2 and 3.

References

[1] S. Kloder and S. Hutchinson, "Path planning for permutation-invariant multirobot formations," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 650–665, Aug. 2006.

[2] C. Ze-Su, Z. Jie, and C. Jian, "Formation control and obstacle avoidance for multiple robots subject to wheel-slip," *Int. J. Adv. Robot. Syst.*, vol. 9, no. 5, p. 188, Nov. 2012.

[3] R. Fitch, A. Alempijevic, and T. Peynot, "Global reconfiguration of a team of networked mobile robots among obstacles," *J. Intell. Auton. Syst.*, vol. 302, pp. 639–656, 2016.

[4] X. Cao, D. Zhu, and S. X. Yang, "Multi-AUV target search based on bioinspired neurodynamics model in 3-D underwater environments," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 11, pp. 2364–2374, Nov. 2016.

[5] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "CoBots: Robust symbiotic autonomous mobile service robots," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 4423–4429.

[6] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *J. Auton. Robot.*, vol. 37, no. 4, pp. 401–415, Dec. 2014.

[7] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic Foundations of Robotics X*. Berlin, Germany: Springer, 2013, pp. 157–173.

[8] H. Ma and S. Koenig, "Optimal target assignment and path finding for teams of agents," 2016, *arXiv:1612.05693*. [Online]. Available: http://arxiv.org/abs/1612.05693

[9] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proc. Assoc. Adv. Artif. Intell.*, 2019, pp. 7643–7650.

[10] K. Vedder and J. Biswas, "X*: Anytime multiagent path planning with bounded search," in *Proc. Int. Conf. Auton. Agent. Multi Agent Syst.*, 2019, pp. 2247–2249.

[11] G. Sartoretti *et al.*, "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019.

[12] M. M. Zavlanos and G. J. Pappas, "Dynamic assignment in distributed motion planning with local coordination," *IEEE Trans. Robot.*, vol. 24, no. 1, pp. 232–242, Feb. 2008.

[13] D. Panagou, M. Turpin, and V. Kumar, "Decentralized goal assignment and safe trajectory generation in multirobot networks via multiple Lyapunov functions," *IEEE Trans. Autom. Control*, vol. 65, no. 8, pp. 3365–3380, Aug. 2020.

[14] J. Yu, S.-J. Chung, and P. G. Voulgaris, "Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies," *IEEE Trans. Autom. Control*, vol. 60, no. 2, pp. 327–341, Feb. 2015.

[15] T. Rashid, M. Samvelyan, C. S. D. Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.

[16] P. Sunehag *et al.*, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. Int. Conf. Auton. Agent. Multi Agent Syst.*, 2018, pp. 2085–2087.

[17] H. Tang *et al.*, "Hierarchical deep multiagent reinforcement learning with temporal abstraction," 2018, *arXiv:1809.09332*. [Online]. Available: http://arxiv.org/abs/1809.09332

[18] S. Kumar *et al.*, "Federated control with hierarchical multi-agent deep reinforcement learning," in *Proc. Hierarchical Reinforcement Learn. Workshop Neural Inf. Process. Syst.*, 2017, pp. 1–6.

[19] M. Ghavamzadeh, S. Mahadevan, and R. Makar, "Hierarchical multi-agent reinforcement learning," *Auton. Agents Multi-Agent Syst.*, vol. 13, no. 2, pp. 197–229, Apr. 2006.

[20] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, Aug. 1999.

[21] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. Assoc. Adv. Artif. Intell.*, 2018, pp. 2974–2982.

[22] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[23] G. Tesauro, "Extending Q-learning to general adaptive multi-agent systems," in *Proc. Neural Inf. Process. Syst*, 2004, pp. 871–878.

[24] J. Foerster *et al.*, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1146–1155.

[25] Y. Jin, S. Wei, J. Yuan, X. Zhang, and C. Wang, "Stabilizing multi-agent deep reinforcement learning by implicitly estimating other agents' behaviors," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 3547–3551.

[26] Y. Jin, Y. Zhang, J. Yuan, and X. Zhang, "Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 2897–2901.

[27] J. Roy, P. Barde, F. G. Harvey, D. Nowrouzezahrai, and C. Pal, "Promoting coordination through policy regularization in multi-agent deep reinforcement learning," 2019, *arXiv:1908.02269*. [Online]. Available: http://arxiv.org/abs/1908.02269

[28] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.

[29] R. S. Sutton, A. G. Barto, and F. Bach, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, 1998.

[30] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[31] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[32] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[33] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[34] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971

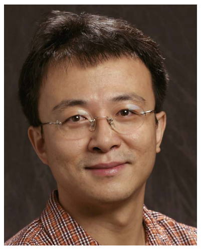
[35] A. Tampuu *et al.*, "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, pp. 1–15, Apr. 2017.

[36] X. Guo, N. Ansari, F. Hu, Y. Shao, N. R. Elikplim, and L. Li, "A survey on fusion-based indoor positioning," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 566–594, 1st Quart., 2020.

[37] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Auton. Agent. Multi Agent Syst.*, 2017, pp. 66–83.

Yue Jin (Graduate Student Member, IEEE) received the B.E. degree in information and communication engineering from Harbin Engineering University, Harbin, China, in 2016. She is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, Tsinghua University, Beijing, China.

Her research interests include signal processing, machine learning, and multiagent systems.



Shuangqing Wei (Senior Member, IEEE) received the B.E. and M.E. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree from the University of Massachusetts, Amherst, MA, USA, in 2003.

He is currently a Tenured Professor with the Division of Electrical and Computer Engineering, School of Electrical Engineering and Computer Science, Louisiana State University (LSU), Baton Rouge, LA, USA, where he also holds the Michel B. Voorhies Distinguished Professorship of Electrical Engineering. His current research interests include information theory, communication theory, and high-dim statistical inference in complex systems and networks, as well as the development of theoretical understanding about deep neural networks.

Jian Yuan (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1998.

He is currently a Professor with the Department of Electronic Engineering, Tsinghua University, Beijing, China. His main research interest is in complex dynamics of networked systems.

Xudong Zhang (Member, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China, in1997.

He is currently a Professor with the Department of Electronic Engineering, Tsinghua University. He has authored or coauthored more than 150 articles and three books in the field of signal processing and machine learning. His research interests include statistical signal processing, machine learning theory, and multimedia signal processing.