Learning Graph-Enhanced Commander-Executor for Multi-Agent Navigation

Xinyi Yang
Tsinghua University
Beijing, China
yang-xy20@mails.tsinghua.edu.cn

Shiyu Huang
4Paradigm Inc.
Beijing, China
huangsy1314@163.com

Yiwen Sun
Fudan University
Shanghai, China
ywsun22@m.fudan.edu.cn

Yuxiang Yang
Tsinghua University
Beijing, China
yuxiang-18@mails.tsinghua.edu.cn

Chao Yu
Tsinghua University
Beijing, China
yc19@mails.tsinghua.edu.cn

Wei-Wei Tu
4Paradigm Inc.
Beijing, China
tuweiwei@4paradigm.com

Huazhong Yang
Tsinghua University
Beijing, China
yanghz@tsinghua.edu.cn

Yu Wang
Tsinghua University
Beijing, China
yu-wang@tsinghua.edu.cn

Abstract

This paper investigates the multi-agent navigation problem, which requires multiple agents to reach the target goals in a limited time. Multi-agent reinforcement learning (MARL) has shown promising results for solving this issue. However, it is inefficient for MARL to directly explore the (nearly) optimal policy in the large search space, which is exacerbated as the agent number increases (e.g., 10+ agents) or the environment is more complex (e.g., 3D simulator). Goal-conditioned hierarchical reinforcement learning (HRL) provides a promising direction to tackle this challenge by introducing a hierarchical structure to decompose the search space, where the low-level policy predicts primitive actions in the guidance of the goals derived from the high-level policy. In this paper, we propose *Multi-Agent Graph-Enhanced Commander-EXecutor* (MAGE-X), a graph-based goal-conditioned hierarchical method for multi-agent navigation tasks. MAGE-X comprises a high-level *Goal Commander* and a low-level *Action Executor*. The *Goal Commander* predicts the probability distribution of the goals and leverages them to assign the most appropriate final target to each agent. The *Action Executor* utilizes graph neural networks (GNN) to construct a subgraph for each agent that only contains its crucial partners to improve cooperation. Additionally, the Goal Encoder in the *Action Executor* captures the relationship between the agent and the designated goal to encourage the agent to reach the final target. The results show that MAGE-X outperforms the state-of-the-art MARL baselines with a 100% success rate with only 3 million training steps in multi-agent particle environments (MPE) with 50 agents, and at least a 12% higher success rate and 2× higher data efficiency in a more complicated quadrotor 3D navigation task.

Keywords

Multi-agent Reinforcement Learning; Goal-conditioned Reinforcement Learning; Multi-agent Navigation; Graph Neural Network

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

ACM Reference Format:
Xinyi Yang, Shiyu Huang, Yiwen Sun, Yuxiang Yang, Chao Yu, Wei-Wei Tu, Huazhong Yang, and Yu Wang. 2023. Learning Graph-Enhanced Commander-Executor for Multi-Agent Navigation. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 Introduction

Navigation is a typical task in the intelligent agent system, applied in a wide range of applications, such as autonomous driving [1, 7], logistics and transportation [6, 22], and search and rescue for disasters [2, 16]. In this paper, we consider a multi-agent navigation task where multiple agents simultaneously move to the target goals in a cooperative fashion. Multi-agent reinforcement learning (MARL) has attracted significant attention due to its powerful expressiveness in multi-agent navigation tasks [12, 25, 38, 41].

The common approach to searching for the near-optimal solution in MARL [37, 40] is to directly train a policy that produces environmental actions for agents. However, learning the strategy directly from large search space results in low data efficiency, which is more severe as the number of agents or the complexity of the environment increases. Therefore, the existing methods in navigation tasks target simple scenarios with few agents. Goal-conditioned HRL [5, 8, 14, 19, 20, 26, 42] has been recognized as an effective paradigm to address this problem, comprising a high-level policy that breaks the original task into a series of subgoals and a low-level policy that aims at the arrival of these subgoals. Recent works [4, 10] in goal-conditioned HRL mainly focus on developing high-level policies for providing agents with appropriate subgoals. However, designating the subgoals may confuse agents on which target goals they should reach.

Graph neural networks (GNN) have been widely applied in multi-agent cooperative tasks due to their effective learning of agents' graph representations. The literature on GNN [11, 21, 28] in MARL constructs a graph whose nodes represent agents' information to model the interaction among them and encourage agents to cooperate. To enhance the cooperation among agents, we can leverage

GNN in the low-level policy in HRL to capture the relationship of agents and express preferences for different teammates.

To improve data efficiency and cooperation, we propose *Multi-Agent Graph-Enhanced Commander-EXecutor* (MAGE-X), a graph-based goal-conditioned hierarchical framework in multi-agent navigation tasks. MAGE-X consists of two components, the high-level **Goal Commander** and the low-level **Action Executor**. For the target-goal assignment, the **Goal Commander** infers the probability distribution of target goals to assign the most appropriate target goal for each agent instead of the subgoal. As a result, the multi-agent navigation is converted to multiple single-agent navigation tasks in the multi-agent environment, in which each agent is required to reach the designated goal while avoiding collisions. In the **Action Executor**, we take advantage of GNN to perceive the relationship among agents and produce a subgraph for each agent to decide to whom they should pay attention. After that, the State Extractor receives the correlation between the agent and its target goal from the Goal Encoder and the agent's embedded feature in the subgraph to empower the team representation with strong goal guidance, promoting agents to reach target goals. The suggested scheme is challenged against MARL baselines in multi-agent particle environments (MPE) [23] with a massive number of agents and a more complicated quadrotor navigation task [29]. The experimental results demonstrate that MAGE-X outperforms the MARL baselines, achieving a 100% success rate with only 3 million training steps in MPE with 50 agents. Furthermore, MAGE-X attains at least a 12% higher success rate and 2× higher data efficiency in the quadrotor navigation task. We would suggest to visit https://sites.google.com/view/mage-x23 for more information.

Our contributions can be summarized as follows:

- We introduce a graph-based goal-conditioned framework in multi-agent navigation tasks, *Multi-Agent Graph-enhanced Commander-Executor* (MAGE-X), to solve the problem of data efficiency and cooperation in large search space.
- We develop a high-level Goal Commander, which utilizes the probability distribution of goals to allocate each agent to the most appropriate target goal.
- We propose the low-level Action Executor, which adopts GNN to improve the coordination, and the Goal Encoder and the State Extractor to encourage agents to complete the task.
- MAGE-X converges much faster and substantially outperforms MARL algorithms in multi-agent particle environments (MPE) [23] with a massive number of agents and a quadrotor navigation task [29].

2 Related Work

2.1 Navigation

Navigation has been widely investigated in recent years, where RL has shown its ability to solve various applications [17, 24, 39, 43]. For example, Rao [31] presents a model-embedded actor-critic architecture for the multi-goal visual navigation task. Furthermore, Zhu [43] introduces a target-driven actor-critic model to achieve greater adaptability and flexibility for the target-driven visual navigation task.

As for multi-agent navigation tasks in MARL [12, 25, 38, 41], the difficulties lie in data efficiency and cooperation in large space spaces, which is exacerbated as agent number increases and the environment becomes more complicated. EPS [25] is introduced to enhance exploration efficiency and improve sample efficiency in multi-robot mapless navigation, which uses the evolutionary population periodically generated from robots' policies to search for different and novel states. Furthermore, Xia [38] proposes an inference-based hierarchical reinforcement learning framework (IHRL) to address the multi-agent cooperative navigation problem via the interplay of high-level inference and low-level actions. The proposed MAGE-X specializes in navigation tasks and outperforms the MARL baselines with high sample efficiency.

2.2 Goal-conditioned HRL

Goal-conditioned hierarchical reinforcement Learning (HRL) has shown its capability in a wide range of tasks [3, 27, 30] with a hierarchy consisting of high-level and low-level policies. The high-level policy generates intermediate subgoals every global timestep, which is regarded as a goal guidance of the low-level policy.

Kreidieh [18] addresses the challenges of the interactions between high-level and low-level agents by introducing inter-level cooperation. This inter-level cooperation is given by modifying the high-level policy's objective function and subsequent gradients. Another representative is LGA [4], where the subgoal assignment is parameterized as latent variables to be trained. LGA directly provides primitive actions for agents depending on latent variables to accomplish multi-agent tasks. MASER [10] automatically produces subgoals for agents from the experience replay buffer relying on both individual and total Q-values. Besides, it adopts the individual intrinsic reward for each agent to reach the assigned subgoals and maximize the joint action value. However, the generation of subgoals may be inefficient since the correspondence between subgoals and the final goals is implicit. In this paper, MAGE-X utilizes the goal-conditioned hierarchical framework in multi-agent navigation tasks to improve the sample efficiency, where the high-level policy deals with target-goal assignment and the low-level policy deals with the action execution for each agent.

2.3 Graph Neural Networks

Graph neural networks (GNN) [34] are broadly used due to their effective learning of graph representations and the ability to capture the relationships of different graphs. Recently, several works have applied GNN in multi-agent systems to model agents' interactions. HAMA [33] proposes a hierarchical graph attention network that captures the underlying relationships at the agent-level and the group-level, enhancing generalization and scalability. MAGIC [28] is a novel graph communication protocol that implies the topology of agents' interactions, helping agents decide when to communicate and with whom to communicate. DICG [21] leverages dynamic coordination graphs to infer joint actions and values implicitly. GCS [32] learns coordinated behaviors by factorizing the joint team policy into a graph generator and a graph-based coordinated policy. The Action Executor of MAGE-X benefits from GNN to capture the team representation and introduces a Goal Encoder and a State Extractor to strengthen the expression of target goals.

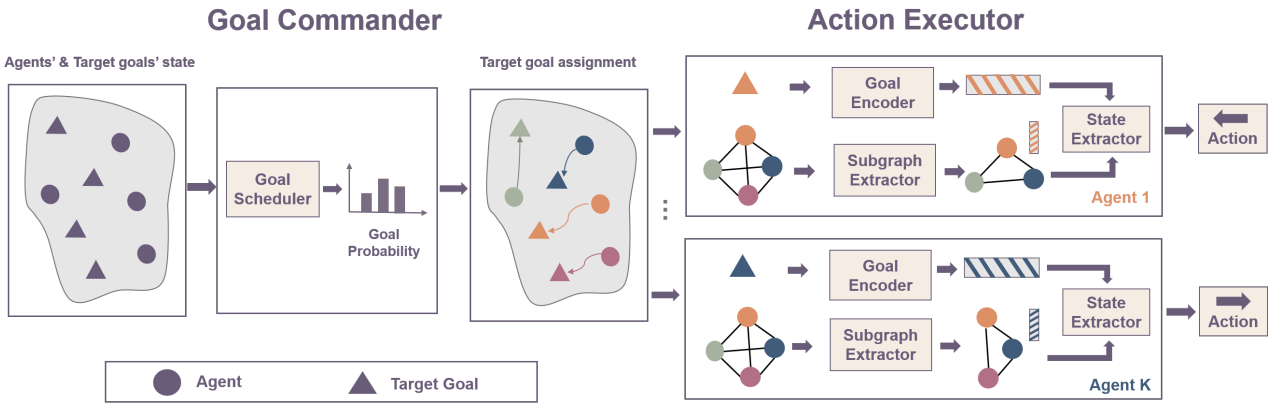


Figure 1: Overview of *Multi-Agent Graph-enhanced Commander-Executor* (MAGE-X).

3 Preliminary

3.1 Dec-MDPs

In this paper, we consider a variant of MDP to solve the decentralized control problem in stochastic environments called Decentralized Markov Decision Processes (Dec-MDPs).

Here, the multi-agent Dec-MDPs problem is formulated as:

$$< \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, R, G, \mathcal{O}, \gamma >, \tag{1}$$

where $\mathcal{N} \equiv \{1, ..., n\}$ is a set of $N = |\mathcal{N}|$ agents. Note that $\mathcal{S}$ is a set of global states in the assumption that $\mathcal{S}$ is jointly observable. $\mathcal{A}$ is the action space of each agent, and $\mathbf{A} \equiv \mathcal{A}^N$ is the joint action space. $\mathcal{O}$ is the observation space. R represents the reward function, and $R(s, \mathbf{a}, s')$ is the reward obtained from the transition of joint actions $\mathbf{a} \in \mathbf{A}$ from the state $s \in \mathcal{S}$ to the state $s' \in \mathcal{S}$. $\gamma \in [0, 1)$ is the discount factor. $\mathcal{T}(s, \mathbf{a}, s') : \mathcal{S} \times \mathbf{A} \times \mathcal{S} \mapsto [0, 1]$ is the dynamics function denoting the transition probability. G is the observation function, and $G(s, \mathbf{a}, s', o_i)$ is the probability of agents $i \in \mathcal{N}$ seeing observation $o_i \in \mathcal{O}$. Each agent has a policy $\pi^i(a_t^i|o_{1:t}^i)$ to produce action a_t^i from observations o_t^i at step t. And agents need to maximize the expected discounted return $\mathbb{E}\sum_{l=0}^{\inf} \gamma^l r_{t+l}$, where $r_t = R(s_t, \mathbf{a}_t, s_{t+1})$ is the joint reward at step t.

3.2 Graph Neural Networks

Graph neural networks (GNN) are a special type of neural network capable of dealing with data in the graph structure. The critical ingredient of GNN is pairwise message passing, i.e., graph nodes iteratively update their representations by exchanging information with their neighbors. The general formula is given as below:

$$h_i^{(l)} = \sigma(\sum_{j \in N_i} \frac{1}{\sqrt{d_i d_j}} (h_j^{l-1} W^{(l)})), \tag{2}$$

where h_i is the feature vector of node i. N_i represents a set of neighbouring nodes of i and $W^{(l)}$ is the learnable weights in the layer l. σ is the activation function. d_i is the dimension of feature stored in node i. Equation 2 shows that the feature of node i will be influenced by its neighbors. In this paper, we use graph convolutional networks (GCN) [15] to model the interaction of agents. GCN is a variant of convolution neural networks (CNN) to be applied in the data with graph structure.

4 Methodology

In this section, we introduce the proposed framework, MAGE-X, to improve sample efficiency and the cooperation in multi-agent navigation tasks. The overview of our framework is demonstrated in Fig. 1. MAGE-X comprises two components, the Goal Commander and the Action Executor. The high-level Goal Commander follows the principles of centralized-training-centralized-execution (CTCE). In contrast, the low-level Action Executor is in a decentralized setting with partial observation of N agents, where agent k receives local observation, o_t^k, at step t. Agent k learns the policy, π_k, to produce a distribution over actions at each time step t, $a_t^k \sim \pi_k(a_t^k|o_t^k)$. The process of a navigation task begins with the Goal Scheduler in the Goal Commander receiving the spawn locations of all the agents and the target goals. The probability distribution of the target goals is predicted by the scheduler, and each agent is assigned the most appropriate target goal rather than the subgoal. Therefore, the multi-agent navigation is converted to multiple single-agent navigation tasks in the multi-agent environment, where each agent is required to reach the given goal as quickly as possible while avoiding collisions. Take agent k as an example. The Subgraph Extractor takes in the observation of agent k and other agents and produces the subgraph of agent k only including its crucial neighbors to improve cooperation. The relationship of agent k and its target goal is extracted from the Goal Encoder, which is then sent to the State Extractor combined with the feature of agent k in the subgraph to endow the team representation with strong goal guidance. Finally, agent k takes the preliminary action from the Action Generator to complete the navigation task.

4.1 Goal Commander

Goal assignment is a long-studied maximal matching problem, especially in scenarios with large-scale agents. The Goal Commander builds upon a Goal Scheduler module for target-goal assignment, which produces the probability distribution of target goals. The designed reward is related to the distance cost of the Hungarian algorithm [13], a state-of-the-art classical method to tackle the combinatorial optimization algorithm in graph theory.

Goal Scheduler: This module is made up by a Multi-Layer Perception (MLP) layer, f_{sche}, that takes the positions of all agents,

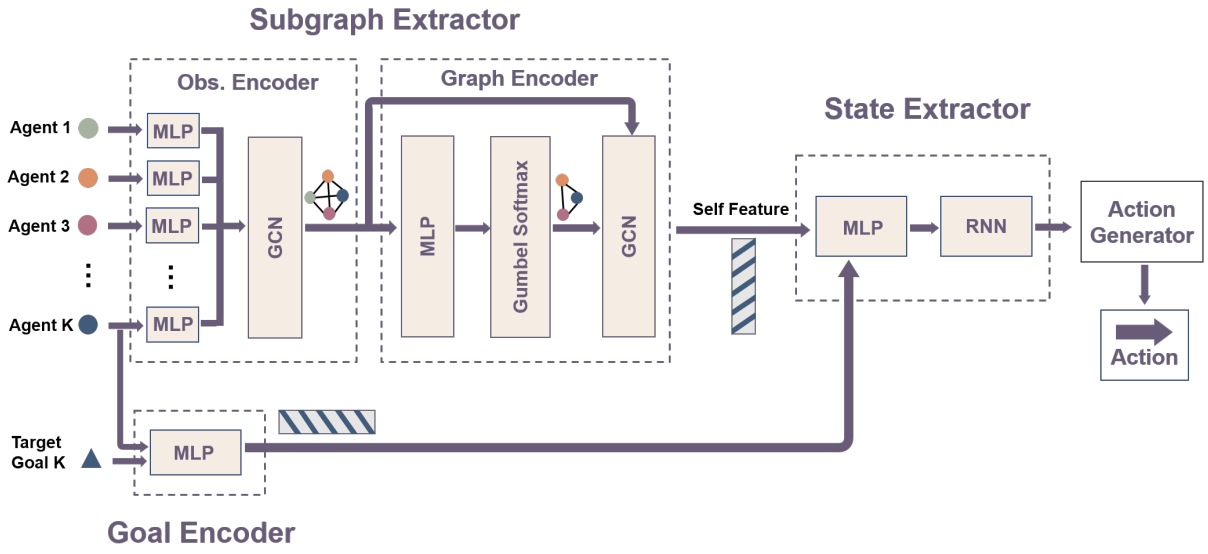


Figure 2: Workflow of Action Executor, including a Subgraph Extractor, a Goal Encoder, a State Extractor and an Action Generator.

P_a, and the positions of target goals, P_g, as input. We obtain the probability of the target goals, P_{goal}, by computing the softmax operator over the output of f_{sche}:

$$P_{goal} = Softmax(f_{sche}(P_a, P_g)). \tag{3}$$

Thereafter, P_{goal} is ranked in decreasing order and the reordered goals 1...N are sequentially assigned to agent 1...N. The reward of the Goal Scheduler, R_c, represents the distance cost of our assignment strategy, C_c, against the distance cost of the Hungarian algorithm, C_h:

$$R_c = 1 - \frac{C_c}{C_h}. \tag{4}$$

4.2 Action Executor

The Action Executor is designed for high cooperation where agents speedily reach the designated goal with little collision in the multi-agent environment. The workflow of the Action Executor is illustrated in Fig. 2. It consists of the Subgraph Extractor, the Goal Encoder, the State Extractor, and the Action Generator. The Subgraph Extractor encodes the observations of all agents and yields a subgraph for agent k only containing the crucial teammates. The Goal Encoder extracts the correlation of agent k and its assigned goal, which is then fed into the State Extractor with the feature of agent k in the subgraph to endow the team representation with target goal guidance. Finally, the Action Generator produces the action for the agent k.

The environmental reward for each agent, R_e, is the linear combination of the complete bonus, R_b, the distance penalty, R_d, and the collision penalty, R_c:

$$R_e = \alpha R_b + \beta R_d + \gamma R_c, \tag{5}$$

where α, β and γ are the coefficient of R_b, R_d and R_c, respectively.

Subgraph Extractor: The Subgraph Extractor is comprised of the Observation Encoder and the Graph Encoder. In the Observation Encoder, we apply an MLP layer f_o to encode the observations of all agents and GCN to produce a fully connected graph of agents, G_a. This can be formulated as:

$$G_a = GCN\left(\left(f_o(o^1), f_o(o^2), ..f_o(o^N)\right), A_f\right), \tag{6}$$

where A_f is denoted as an adjacent matrix of a fully connected graph. In the Graph Encoder, we compute the gumbel softmax [9] over the feature of G_a updated by an MLP layer, f_g. Thus, we can obtain an adjacent matrix of A_s^k through it:

$$A_s = Gumbel\_Softmax(f_g(G_a)). \tag{7}$$

Afterwards, we use GCN to generate G_s^k:

$$G_s^k = GCN(G_a, A_s^k). \tag{8}$$

The feature of the agent k in G_s^k is then sent to the State Extractor. Note that there are l_g blocks in the Graph Encoder.

Goal Encoder and State Extractor: To capture the correlation between the agent k and its assigned goal, the Goal Encoder comprises an MLP layer f_{goal}, and takes in the target goal's position P_g^k, and the agent's observation o^k. This can be formulated as:

$$E_{goal}^k = f_{goal}(P_g^k, o^k). \tag{9}$$

Receiving the relationship between the agent and its target goal and the embedded feature of agent k in G_s, E_a^k, we leverage the State Extractor to enhance the team representation with target goal guidance and output E_{all}^k. The State Extractor consists of an MLP layer f_{state}, and recurrent neural networks (RNN) in consideration of the high correlation between current and historical states:

$$E_{all}^k = RNN\left(f_{state}\left(concat\left(E_{goal}^k, E_a^k\right)\right)\right). \tag{10}$$

Algorithm 1 Training Procedure of *MAGE-X*

Input : The positions P_a of all agents, the joint observation **o** of all agents , and the positions P_g of target goals.

Output : Final policy π_θ for the Goal Commander and $\pi_{\theta'}$ for the Action Executor.

1: **Initialize:** agents number N, maximal steps in an episode T_e, params in the Goal Commander θ, ϕ, params in the Action Executor θ', ϕ', the Goal Commander buffer D, the Action Executor buffer D'.
2: **while** θ and θ' not converges **do**
3: Reset environment and get P_a, P_g.
4: **Initialize:** the step-count $t \leftarrow 1$ in D, the step-count $t' \leftarrow 1$ in D'.
5: $P_{goal} \leftarrow$ **Goal_Commander**(P_a, P_g).
6: Calculate $\pi_\theta(a_t|\mathbf{o}_t)$ and $V_\phi(\mathbf{o}_t)$.
7: Perform $a_t \sim \pi_\theta(a_t|\mathbf{o}_t)$.
8: **while** $t' < T_e$ and not terminal **do**
9: $E^k_{all} \leftarrow$ **Action Executor (o,** P^k_g**)** for each agent k
10: Calculate $\pi_{\theta'}(a'^k_{t'}|o'^k_{t'})$ and $V_{\phi'}(o'^k_{t'})$ for each agent k.
11: Perform $a'^k_{t'} \sim \pi_{\theta'}(a'^k_{t'}|o'^k_{t'})$ for each agent k.
12: Receive $r'_{t'}$ and $o'^k_{t'+1}$ for each agent k.
13: Store $(o'^k_{t'}, a'^k_{t'}, \pi_{\theta'}, (a'^k_{t'}|o'_{t'}), V_{\phi'}(o'^k_{t'}), r'_{t'}, o'^k_{t'+1})$ in D'.
14: $t' \leftarrow t' + 1$
15: **end while**
16: Receive r_t and o_{t+1}.
17: $t \leftarrow t + 1$
18: Store $(o_t, a_t, \pi_\theta, (\mathbf{a}_t|\mathbf{o}_t), V_\phi(o_t), r_t, o_{t+1})$ in D.
19: Perform update of θ, ϕ, θ' and ϕ'.
20: **end while**

4.3 Multi-agent Commander-Executor Training

MAGE-X, following the goal-conditioned MARL framework, trains two policy networks for the Goal Commander and the Action Executor separately, which are optimized by maximizing the accumulated reward in the entire episode via reinforcement learning. We use Multi-agent Proximal Policy Optimization (MAPPO) [40], a multi-agent variant of Proximal Policy Optimization (PPO) [35], as the policy optimizer.

As shown in Algorithm 1, MAGE-X takes in the positions of agents, the observations of agents, and the positions of landmarks. Then, it produces the final policy π_θ for the Goal Commander and $\pi_{\theta'}$ for the Action Executor. First, we initialize the number of agents along with several training parameters, including maximal steps, parameters in the Goal Commander, parameters in the Action Executor, the Goal Commander reply buffer, D, and the Action Executor reply buffer, D' (Line 1). The Goal Commander only performs one action every episode to assign the goals to agents in the beginning (Line 5 $\sim$ 7) and receives the reward r_t and the subsequent observation $o_{(t+1)}$ at the end of each episode, where t represents the step-count in D (Line 15 and 16). Thereafter, we update θ for the policy network π_θ, and ϕ for the value network V_ϕ, in the Goal Commander. Regarding the Action Executor, it outputs the action $a'^k_{t'}$ for each agent k, and stores a group of data in D' at each timestep t' (Line 9 $\sim$ 13). Similarly, we update θ' for the policy network $\pi_{\theta'}$, and ϕ' for the value network $V_{\phi'}$, in the Action Executor.

5 Experiments

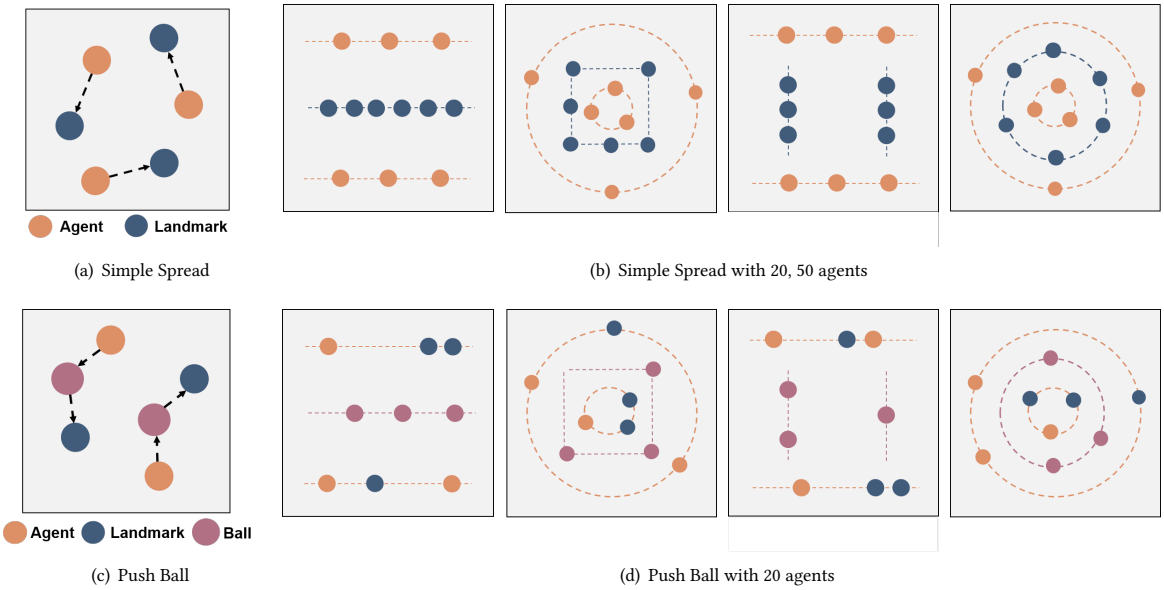
5.1 Task Setup

To evaluate the effectiveness of our algorithm in large search space, we consider MPE [23] with a massive number of agents and pybullet-gym-drones [29] in 3D space as the experimental environments, as shown in Fig. 3 and Fig. 4. We select three typical tasks from these two environments: Simple Spread, Push Ball, and Drones. Then, we conduct experiments with $N \in \{5, 20, 50\}$ agents in Simple Spread and $N \in \{5, 20\}$ agents in Push Ball in MPE. Drone is a more complicated quadrotor navigation task in gym-pybullet-drones, which adopts aerodynamic models of quadrotors to narrow the gap between the simulation and the real world. Therefore, quadrotors in Drone require stronger cooperation to avoid the crash. We conduct experiments with 2 and 4 quadrotors in this 3D simulator.

5.1.1 Simple Spread We utilize Simple Spread environment in MPE [23], a classical 2D navigation task. The episode starts with the initialization of N agents and N landmarks. When all the agents reach the landmarks, this task is 100% successful. The consequence of the collision between agents is that they will bounce off each other, which is detrimental to navigation efficiency. The available discrete actions of agents include *Up*, *Down*, *Left*, and *Right*. The task's difficulty increases with a larger agent number and map space. The experiment is conducted on 5, 20, and 50 agents. In the 5-agent setting, the spawn locations of agents and landmarks are random on the map with a size of 4. In the setting of $N \in \{20, 50\}$, the spawn locations of agents and landmarks are randomly initialized in four challenging maps with the size of 36 and 100, respectively, as shown in Fig 3(b). The horizons of steps in $N = 5$, 20, 50 are 60, 100, and 120, respectively.

5.1.2 Push Ball Push Ball is a more complicated two-stage navigation task in MPE [23], requiring each agent to find the ball first, then push it into the landmark. At the beginning of each episode, N agents, N balls, and N landmarks are initialized in the environment. The success rate of the task is the number of agents reaching the landmarks with the balls to the total number of all agents. The available discrete actions of agents include *Up*, *Down*, *Left*, and *Right*. We conduct the experiments in the setting of 5 and 20 agents. In the scenarios of $N = 5$, the spawn locations of agents, balls, and landmarks are randomly distributed on the map of size 16. For $N = 20$, their initialization follows four challenging space arrangements on the map with a size of 144, as displayed in Fig 3(d). The horizons of the steps in $N = 5$ and 20 are 100 and 200, respectively.

5.1.3 Drone We further adopt Drone in pybullet-gym-drones [29], a 3D simulator for flying quadrotors based on pybullet. Drone models the dynamics of quadrotors and controls them by adjusting the torque. Unlike MPE, the collision between quadrotors will lead to a crash and the failure of the task. In our navigation task, the spawn locations of N quadrotors and N landmark are random in the coordinates x and y, but fixed in the coordinate z, with $x \in (-1.5, 1.5)$, $y \in (-1.5, 1.5)$ and $z = 1.5$. The task's objective is that quadrotors are required to reach all the landmarks on a limited time budget. The action space of each quadrotor is the direction of



(a) Simple Spread

(b) Simple Spread with 20, 50 agents

(c) Push Ball

(d) Push Ball with 20 agents

Figure 3: Experimental environments of Simple Spread and Push Ball in MPE. We remark that (b) represents simplified demonstration of 4 challenging task modes in Simple Spread, where the spawn locations of agents and landmarks are randomly distributed in the orange lines and the blue lines, respectively. (d) expresses simplified demonstration of 4 typical task modes in Push Ball, where the spawn locations of agents and landmarks are randomly distributed in the orange lines, and the spawn locations of balls are randomly distributed in the pink lines.

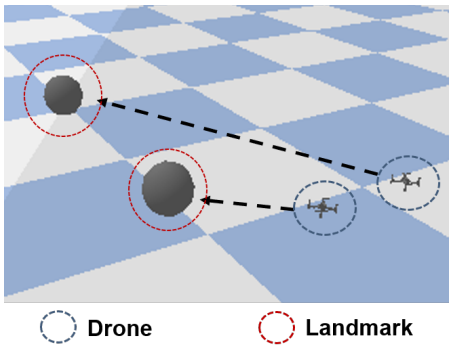


Figure 4: Experiment environments of Drone.

the acceleration in x, y, z coordinate, containing *Forward*, *Backward*, and *Stop* in each coordinate. We consider the experiments with 2 and 4 quadrotors. In our setting, the frequency at which the physics engine steps is 120 HZ; the max speed is 2 m/s, and the acceleration is 5 m/s^2. In every low-level step, we utilize the predicted acceleration direction to calculate each quadrotor's target velocity, and then the quadrotor executes the action every four physics engine steps. The horizon of the steps in $N = 2$ and 4 is 120.

5.2 Implementation Details

Each RL training is performed over 3 random seeds for a fair comparison. Each evaluation score is expressed in the format of "mean (standard deviation)", which is averaged over a total of 300 testing episodes, i.e., 100 episodes per random seed. In addition, we use the success rate, the number of landmarks reached by the agents to the total number of landmarks, to express the performance of each algorithm. In the Drone, we additionally consider the collision rate, the number of crashed quadrotors to the total number of quadrotors, to express the capability of cooperation. Our experimental platform involves a 128-core CPU, 256GB RAM, and an NVIDIA GeForce RTX 3090Ti with 24GB VRAM.

5.3 Baselines

To demonstrate the effectiveness of our methods, we challenge it against four MARL baseline approaches.

- MAPPO [40]: This is the first and the most direct approach for applying PPO in MARL. It equips all agents with one shared set of parameters and updates the shared policy through agents' aggregated trajectories.
- Multi-Agent-Transformer (MAT) [37]: This is an encoder-decoder architecture that leverages the multi-agent advantage decomposition theorem to transform the joint policy search problem into a sequential decision-making process.
- DICG [21]: This graph-based method comprises a module that infers the dynamic coordination graph structure and a GNN module that implicitly reasons about the joint actions or value based on the former module's output.
- Maser [10]: This is a goal-conditioned MARL method that automatically generates subgoals for multiple agents from the experience replay buffer by considering both the individual Q-value and the total Q-value.

5.4 Main Results

We present and analyze the experiment results of MAGE-X and MARL baselines in three competitive environments.

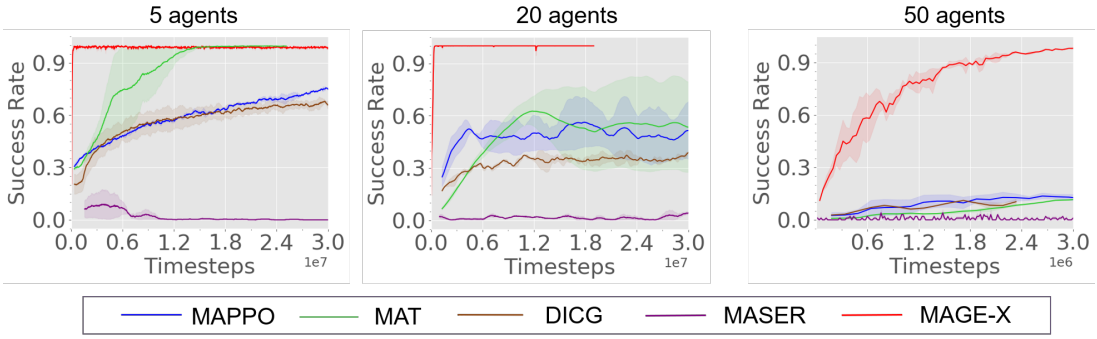


Figure 5: Comparison between MAGE-X and other baselines in Simple Spread with $N = 5, 20, 50$.

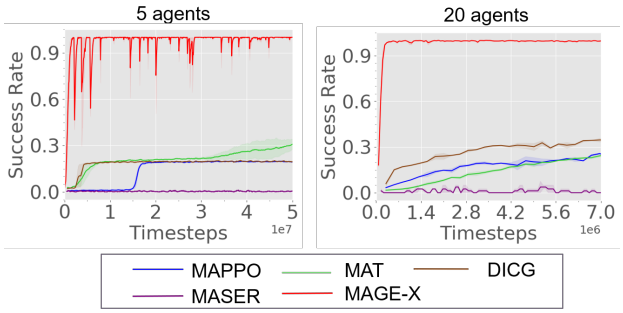


Figure 6: Comparison between MAGE-X and other baselines in Push Ball with $N = 5, 20$.

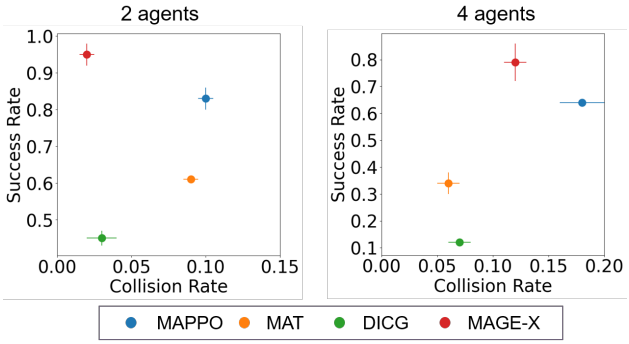


Figure 7: Evaluation performance of the success rate and the collision rate between MAGE-X and baselines in Drone. The node in the upper left corner of the figure represents higher performance.

5.4.1 Simple Spread We demonstrate the training curves in Fig. 5 and the evaluation performance in Table 1. The results suggest that MAGE-X performs the best, especially in scenarios with 50 agents, where it converges to the near-optimal solution quickly with a success rate of 100%, attaining training efficiency more than 10× higher than other competitors. The results show that our algorithm performs the best, especially in the scenario with 50 agents. This graph-enhanced hierarchical framework helps agents quickly handle the problem of reaching targets with high cooperation. MAT excels in other MARL baselines and can achieve a success rate of 100% in the 5-agent setting, indicating that MAT agents profit from the action information of their teammates. However, as the number of agents increases, the performance of MAT degrades significantly.

Table 1: Evaluation performance of the success rate between MAGE-X and baselines in Simple Spread.

	MAPPO	MAT	DICG	MASER	MAGE-X
5 agents	0.82(0.03)	1.00(0.01)	0.71(0.05)	0.06(0.12)	**1.00**(0.01)
20 agents	0.50(0.31)	0.56(0.26)	0.41(0.04)	0.02(0.23)	**1.00**(0.01)
50 agents	0.13(0.02)	0.11(0.01)	0.12(0.02)	0.01(0.05)	**1.00**(0.01)

Table 2: Evaluation performance of the success rate between MAGE-X and baselines in Push Ball.

	MAPPO	MAT	DICG	MASER	MAGE-X
5 agents	0.20(0.01)	0.32(0.04)	0.20(0.01)	0.01(0.01)	**1.00**(0.01)
20 agents	0.26(0.01)	0.25(0.02)	0.36(0.01)	0.01(0.01)	**1.00**(0.01)

Table 3: Evaluation performance of the success rate and the collision rate between MAGE-X and baselines in Drones.

	2 agents		4 agents	
	Suc. Rate	Colli. Rate	Suc. Rate	Colli. Rate
MAPPO	0.83(0.03)	0.10(0.01)	0.64 (0.01)	0.18 (0.02)
MAT	0.61(0.01)	0.09 (0.01)	0.34(0.04)	**0.06**(0.01)
DICG	0.45(0.02)	0.03(0.02)	0.12(0.01)	0.07(0.01)
MAGE-X	**0.95**(0.03)	**0.02**(0.01)	**0.79**(0.07)	0.12 (0.01)

DiCG is on-par with MAPPO, which indicates that directly utilizing the graph-based method can't contribute to better cooperation and efficiency. Although MASER is also a goal-conditioned HRL method, it is the worst among all the competitors. It implies that the value-based method, MASER, fails to discover effective cooperation strategies within limited training steps since it infers subgoals from the experience replay buffer and lacks a strong connection between agents and target goals.

5.4.2 Push Ball As shown in Fig. 6 and Table 2, we conduct the experiments with $N \in \{20, 50\}$ agents in Push Ball. The difficulty of Push Ball lies in that it requires a two-stage goal assignment, i.e., the agents first get the designated ball and then reach the target landmark with the ball. Nevertheless, MAGE-X still achieves a 100% success rate with few training timsteps. In contrast, except for MAT, whose success rate slightly increases, other baselines obtain suboptimal policies with a success rate of 20% in the 5-agent setting. In

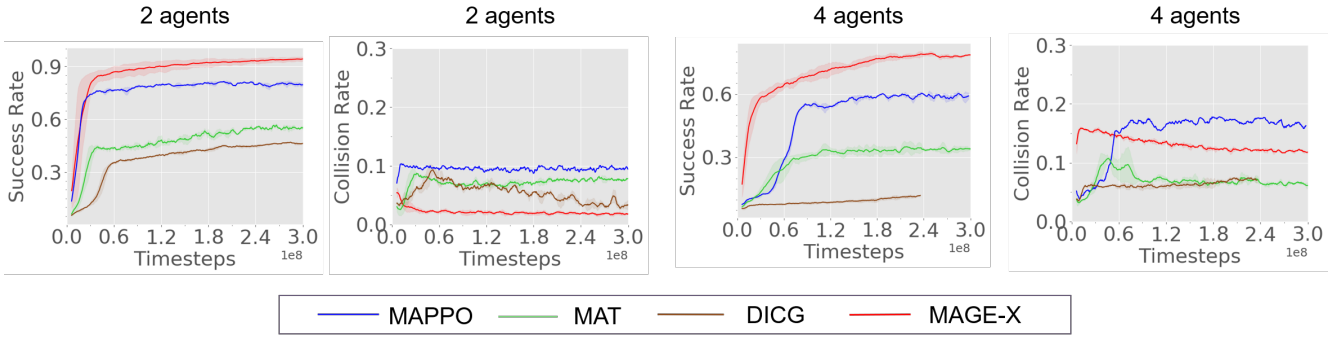


Figure 8: Comparison between MAGE-X and other baselines in Drone.

Table 4: Evaluation performance of Success Rate between MAGE-X and its variants in Simple Spread.

	MAGE-X w. RG	MAGE-X-Atten	MAGE-X-MLP	MAGE-X
Suc. Rate	0.83(0.01)	0.93(0.01)	0.50(0.08)	**1.00**(0.01)

the scenario with 20 agents, all baselines fail to reach the goals. The results express that the high-level Goal Commander in MAGE-X has the potential to simultaneously tackle multiple goal assignment problems, for it breaks this Mstage goal assignment into M independent tasks, only requiring the information of assigned targets.

5.4.3 Drones We report the training performance in Fig. 8 and the evaluation results in Fig. 7 and Table 3. The experiment shows that MAGE-X is superior to other competitors with a success rate of 95% in $N = 2$ and 79% in $N = 4$ in the physically realistic 3D environment, Drone. Unlike MPE, the best competitor in Drone among baselines is MAPPO rather than MAT, indicating that MAPPO has better competence in complex tasks with few agents. Specifically, MAGE-X outperforms MAPPO, with 12% higher and 15% higher success rates in $N = 2$ and 4, respectively. Furthermore, MAGE-X manifests high coordination with a low collision rate of 0.02% in $N = 2$ and 0.12% in $N = 4$, demonstrating that MAGE-X succeeds in assigning target goals to agents to coordinate and capture the interaction among agents. Although MAPPO performs best among baselines, its collision rate is remarkably high. We speculate that MAPPO agent pursues high individual capability instead of cooperation. Furthermore, the sample efficiency of MAGE-X is 2× higher than that of MAPPO in the 4-agent setting.

5.5 Ablation Study

To illustrate the effectiveness of each component of MAGE-X, we consider 3 variants of our method in Simple Spread with 50 agents:

- **MAGE-X w. RG**: We substitute the Goal Scheduler in the Goal Commander with random sampling without replacement to assign each agent a random target goal.
- **MAGE-X-Atten**: We remove GCN in the Obs. Encoder and replace the Graph Encoder in the Action Executor with the attention module [36] to extract the relationship of agents. The concatenation of $(f_o(o_1), f_o(o_2), ..f_o(o_N))$ is fed into the attention module.
- **MAGE-X-MLP**: We consider the MLP layer as the alternative to the Action Executor to capture the correlation between agents and goals. The MLP layer takes the concatenation of $(f_o(o_1), f_o(o_2), ..f_o(o_N), P_g)$ as input.

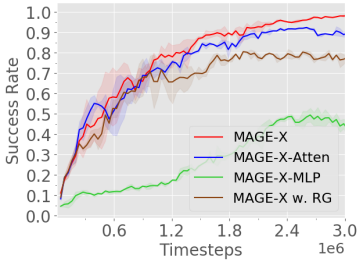


Figure 9: Ablation study on MAGE-X in Simple Spread with 50 agents.

Fig. 9 and Table 4 summarize the performance of MAGE-X and its variants on training and evaluation, respectively. MAGE-X excels in data efficiency and final performance with a 100% success rate. *MAGE-X-MLP* degrades most, implying that the MLP layer is incapable of distinguishing the correlation of agents and target goals from the given information. *MAGE-X w. RG* lacks an appropriate Goal scheduler in the high-level Goal Commander, which may lead to agents being assigned distant goals. Therefore, MAGE-X w. RG reveals a worse performance with an 83% success rate. *MAGE-X-Atten* is slightly inferior to MAGE-X with a 7% lower success rate. We hypothesize that *MAGE-X-Atten* provides each agent with the attention weights of all the neighbors, where needless teammates may influence agents. On the contrary, MAGE-X agent only concentrates on crucial neighbors in the subgraph with useful information.

6 Conclusion and Future Work

In this paper, we propose a goal-conditioned MARL framework, *Multi-Agent Commander-Executor* (MAGE-X), to improve data efficiency and cooperation in multi-agent navigation tasks, especially in scenarios with large space spaces (e.g., a massive number of agents or complex 3D simulator). MAGE-X consists of a high-level Goal Commander and a low-level Action Executor, where the Commander allocates target goals to agents via the probability distribution of goals and the Executor leverages GNN and a Goal Encoder to capture team representation with strong goal guidance. Thorough experiments demonstrate that MAGE-X achieves higher sample efficiency and better performances than all the state-of-the-art MARL baselines in multi-agent particle environments (MPE) with large-scale agents and a more complicated quadrotor navigation task. Currently, MAGE-X mainly focuses on multi-agent navigation tasks, and we will try to apply MAGE-X to other multi-agent tasks beyond navigation in the future.

ACKNOWLEDGMENT

This research was supported by National Natural Science Foundation of China (No.62203257, U19B2019, M-0248), Tsinghua University Initiative Scientific Research Program, Tsinghua-Meituan Joint Institute for Digital Life, Beijing National Research Center for Information Science, Technology (BNRist) and Beijing Innovation Center for Future Chips.

References

[1] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. 2017. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles* 2, 3 (2017), 194–220.

[2] Daniele Calisi, Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. 2005. Autonomous navigation and exploration in a rescue environment. In *IEEE International Safety, Security and Rescue Rototics, Workshop, 2005.* IEEE, 54–59.

[3] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. 2021. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*. PMLR, 1430–1440.

[4] Rui Chen, Peide Huang, and Laixi Shi. 2021. Latent Goal Allocation for Multi-Agent Goal-Conditioned Self-Supervised Imitation Learning. In *Advances in Neural Information Processing Systems*.

[5] Peter Dayan and Geoffrey E Hinton. 1992. Feudal reinforcement learning. *Advances in neural information processing systems* 5 (1992).

[6] Kaivuan Gao, Jing Xin, Han Cheng, Ding Liu, and Jiang Li. 2018. Multi-mobile robot autonomous navigation system for intelligent logistics. In *2018 Chinese Automation Congress (CAC)*. IEEE, 2603–2609.

[7] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. 2020. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics* 37, 3 (2020), 362–386.

[8] Christopher Hoang, Sungryull Sohn, Jongwook Choi, Wilka Carvalho, and Honglak Lee. 2021. Successor Feature Landmarks for Long-Horizon Goal-Conditioned Reinforcement Learning. *Advances in Neural Information Processing Systems* 34 (2021), 26963–26975.

[9] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[10] Jeewon Jeon, Woojun Kim, Whiyoung Jung, and Youngchul Sung. 2022. MASER: Multi-Agent Reinforcement Learning with Subgoals Generated from Experience Replay Buffer. In *International Conference on Machine Learning*. PMLR, 10041–10052.

[11] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2018. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202* (2018).

[12] Yue Jin, Yaodong Zhang, Jian Yuan, and Xudong Zhang. 2019. Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2897–2901.

[13] Roy Jonker and Ton Volgenant. 1986. Improving the Hungarian assignment algorithm. *Operations Research Letters* 5, 4 (1986), 171–175.

[14] Junsu Kim, Younggyo Seo, and Jinwoo Shin. 2021. Landmark-guided subgoal generation in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 28336–28349.

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Alexander Kleiner, Johann Prediger, and Bernhard Nebel. 2006. RFID technology-based exploration and SLAM for search and rescue. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4054–4059.

[17] Songsang Koh, Bo Zhou, Hui Fang, Po Yang, Zaili Yang, Qiang Yang, Lin Guan, and Zhigang Ji. 2020. Real-time deep reinforcement learning based vehicle navigation. *Applied Soft Computing* 96 (2020), 106694.

[18] Abdul Rahman Kreidieh, Glen Berseth, Brandon Trabucco, Samyak Parajuli, Sergey Levine, and Alexandre M Bayen. 2019. Inter-level cooperation in hierarchical reinforcement learning. *arXiv preprint arXiv:1912.02368* (2019).

[19] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

[20] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. 2017. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948* (2017).

[21] Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. 2020. Deep implicit coordination graphs for multi-agent reinforcement learning. *arXiv preprint arXiv:2006.11438* (2020).

[22] Shaotang Liu and Lixin Hu. 2009. Application of beidou navigation satellite system in logistics and transportation. In *Logistics: The Emerging Frontiers of Transportation and Development in China*. 1789–1794.

[23] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[24] Enrico Marchesini and Alessandro Farinelli. 2020. Discrete deep reinforcement learning for mapless navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10688–10694.

[25] Enrico Marchesini and Alessandro Farinelli. 2022. Enhancing deep reinforcement learning approaches for multi-robot navigation via single-robot evolutionary policy search. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 5525–5531.

[26] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems* 31 (2018).

[27] Suraj Nair and Chelsea Finn. 2019. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *arXiv preprint arXiv:1909.05829* (2019).

[28] Yaru Niu, Rohan R Paleja, and Matthew C Gombolay. 2021. Multi-Agent Graph-Attention Communication and Teaming.. In *AAMAS*. 964–973.

[29] Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P Schoellig. 2021. Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 7512–7519.

[30] Karl Pertsch, Oleh Rybkin, Frederik Ebert, Shenghao Zhou, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. 2020. Long-horizon visual planning with goal-conditioned hierarchical predictors. *Advances in Neural Information Processing Systems* 33 (2020), 17321–17333.

[31] Zhenhuan Rao, Yuechen Wu, Zifei Yang, Wei Zhang, Shijian Lu, Weizhi Lu, and ZhengJun Zha. 2021. Visual navigation with multiple goals based on deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems* 32, 12 (2021), 5445–5455.

[32] Jingqing Ruan, Yali Du, Xuantang Xiong, Dengpeng Xing, Xiyun Li, Linghui Meng, Haifeng Zhang, Jun Wang, and Bo Xu. 2022. GCS: Graph-Based Coordination Strategy for Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2201.06257* (2022).

[33] Heechang Ryu, Hayong Shin, and Jinkyoo Park. 2020. Multi-agent actor-critic with hierarchical graph attention network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7236–7243.

[34] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[37] Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. 2022. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. *arXiv preprint arXiv:2205.14953* (2022).

[38] Lijun Xia, Chao Yu, and Zifan Wu. 2021. Inference-based Hierarchical Reinforcement Learning for Cooperative Multi-agent Navigation. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 57–64.

[39] Xinyi Yang, Chao Yu, Jiaxuan Gao, Yu Wang, and Huazhong Yang. 2022. SAVE: Spatial-Attention Visual Exploration. In *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, 1356–1360.

[40] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2021. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. *arXiv preprint arXiv:2103.01955* (2021).

[41] Chao Yu, Xinyi Yang, Jiaxuan Gao, Huazhong Yang, Yu Wang, and Yi Wu. 2022. Learning efficient multi-agent cooperative visual exploration. In *European Conference on Computer Vision*. Springer, 497–515.

[42] Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. 2020. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 21579–21590.

[43] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3357–3364.