Heterogeneous Multi-Robot Cooperation With Asynchronous Multi-Agent Reinforcement Learning

Han Zhang, Xiaohui Zhang, Zhao Feng, and Xiaohui Xiao, *Member, IEEE*

***Abstract*—Multi-robot systems (MRSs) are becoming increasingly important in various domains. However, effective communication and coordination among multiple robots remain significant challenges. In this letter, we introduce a novel architecture for multi-robot decision-making and control based on multi-agent reinforcement learning (MARL). Our architecture can accommodate heterogeneous robots operating asynchronously in different scenarios. We propose an improved practical Q-value mixing network (Qrainbow), which builds on value-decomposition networks and applies the multi-head attention mixer of Qatten and effective components from Rainbow, such as double network, dueling network, and prioritized experience replay. To migrate the algorithm to MRS, we fuse macro-action into Qrainbow and make a slight change to the process of calculating the loss function, enabling Qrainbow to work in asynchronous scenarios. We evaluate our architecture in both the benchmark environment for MARL and a multi-robot environment with varying layouts. In terms of convergence speed and final result, Qrainbow outperforms other state-of-the-art MARL algorithms. Additionally, our architecture achieves superior performance in reducing time costs and avoiding collisions between robots in homogeneous and heterogeneous multi-robot cooperation tasks.**

***Index Terms*—Multi-robot systems, reinforcement learning, heterogeneous robots, asynchronous execution.**

I. Introduction

Multi-robot systems (MRSs) are groups of fixed or mobile robots that cooperate to perform certain tasks in a distributed manner. They have applications in various domains, such as foraging [1], inspection [2], simultaneous localization and mapping [3], object transportation [4], [5], and search and rescue [6]. The employment of heterogeneous robots within MRS is essential for accomplishing a broad spectrum of tasks and capabilities. With their varied abilities and attributes, heterogeneous robots can collaborate to complete complex tasks that would be challenging or unattainable for a single type of robot. Different types of robots can offset each other's weaknesses and constraints. Furthermore, in real-world MRS, robots are typically given navigation points that have varying time costs for execution. As a result, asynchrony is a significant issue that must be addressed in algorithm design.

Multi-agent reinforcement learning (MARL) is concerned with how multiple agents interact with the environment in order to achieve a common goal. Research in this area began in the last century [7]. Recently, deep neural networks have been applied to reinforcement learning methods such as DQN [8], DDPG [9], and PPO [10], with many works proposing extensions to multi-agent settings. A common approach is centralized training and decentralized execution (CTDE) [11], which addresses challenges such as non-stationarity, communication, and coordination [12]. Classical applications of the actor-critic algorithm on MARL include Multi-agent DDPG [11] and Multi-agent PPO [13]. Each agent has a centralized critic that estimates its value function, and a decentralized actor to learn its policy. Another method, Value-Decomposition Networks (VDN) [14], is based on Q-learning and decomposes the global Q-function into individual Q-functions, with extensions such as QMIX [15] and Qatten [16] allowing for inter-agent communication through different mixing networks. MARL can accommodate heterogeneity in robots with varying action spaces, observation spaces, and behavior patterns by training distinct networks with different input and output dimensions using the same network architecture.

Actions are modeled as primitive operations and executed synchronously across agents in most current MARL methods. While the application of Q-learning to box pushing [17] and soccer [18] has been successful in some earlier studies, this assumption may not be realistic for MRS, where robots often have different action selection and completion times. A synchronous algorithmic architecture, which makes MRS wait for every robot to be ready before taking new actions, can be impractical and inefficient. A macro-action-based architecture [19] is a possible solution, which allows asynchronous action selection and termination, as well as representing high-level robot controllers naturally. The MacDec-POMDP [19], [20] adapts the options architecture for partially observable multi-agent domains and uses macro-actions to convert from synchronous to asynchronous execution to accommodate the asynchrony of MRS [21]. Several studies have investigated multi-robot tasks, including exploration [22], cooperative manipulation [23], and navigation [24].

Manuscript received 25 July 2023; accepted 13 October 2023. Date of publication 30 October 2023; date of current version 21 November 2023. This letter was recommended for publication by Associate Editor Nare Karapetyan and Editor Aniket Bera upon evaluation of the reviewers' comments. This work was supported in part by the National Key R&B Program of China under Grant 2018YFB2100903 and in part by the Knowledge Innovation Program of Wuhan-Shuguang Project under Grant 2023010201020252. *(Corresponding authors: Zhao Feng; Xiaohui Xiao.)*

The authors are with the School of Power and Mechanical Engineering, Wuhan University, Wuhan 430072, China (e-mail: sagirizhanghan@whu.edu.cn; zhangxh@whu.edu.cn; fengzhao@whu.edu.cn; xhxiao@whu.edu.cn).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3328448, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3328448

In this letter, we present a novel architecture for multi-robot decision-making and control based on multi-agent reinforcement learning. Our architecture can handle heterogeneous robots that operate asynchronously in different scenarios. To achieve this, we propose an enhanced MARL algorithm, which builds on value-decomposition networks [14] and applies the multi-head attention mixer of Qatten. Rainbow [25] is an algorithm that combines several improvements to DQN into a single learner, which incorporates six effective extensions to the DQN algorithm: double Q-learning [26], prioritized experience replay [27], dueling networks [28], multi-step learning [29], distributional reinforcement learning [30], and noisy linear layers [31] for exploration. We evaluate these extensions in our algorithm and select those with evident effects, resulting in our proposed algorithm, Qrainbow. Additionally, Qrainbow uses macro-action joint experience replay trajectories (Mac-JERTs) [21], to extend the algorithm to the asynchronous setting. The main components of our architecture are as follows. While training, we first use fully convolutional networks to extract features from the global state map and the local observation maps of each robot. Then, we use these features to generate Q-value maps that indicate the best macro-actions for each robot to execute. Finally, we use Qrainbow to compute the global Q-value from the state feature, observation features and Q-value maps. While deploying on MRS, we use the CNN-based executive policy trained above that takes the local observation as input and outputs the optimal macro-action for each robot. The macro-actions are then separated to several micro-actions and implemented with low-level control.

Overall, we propose an approach that enhances the performance of value-decomposition networks in terms of convergence speed and final outcome. Furthermore, we extend Q-learning-based MARL to MRS, addressing the problem of balancing task allocation and conflict resolution among heterogeneous robots and the asynchronous operation of robot teams. We conduct experiments in both the benchmark environment for MARL and a multi-robot environment with varying layouts. The experimental results confirm the effectiveness of our proposed architecture, demonstrating that our architecture leads to reduced time costs and fewer collisions between robots while completing cooperative tasks.

II. Problem Formulation

We model our task as a MacDec-POMDP with macro-actions and shared rewards. The execution of actions is separated into two parts: first, a macro-action is generated by the CNN-based executive policy; then, it is separated to several micro-actions and implemented with low-level control. This control aims to reach the specified location via the shortest path.

A MacDec-POMDP is defined by $\langle D, M, \zeta, T', Z, R^c\rangle$. $D = \langle I, S, A, \Omega, T, O, R, \gamma\rangle$ represents the Dec-POMDP definition, where I is a set of n agents, S is the global state space, and A is the joint micro-action space. Ω is the joint local observation space, and $T(s, \vec{a}, s') = P(s' \mid s, \vec{a})$ represents the state transition function when a global state $s \in S$ transitions to a new state s' after taking a joint action $\vec{a}$. $O(\vec{o}, \vec{a}, s') = P(\vec{o} \mid \vec{a}, s')$ represents the observation probability function when a joint micro-action $\vec{a}$ is taken and the global state arrives in s' to receive a joint local observation $\vec{o} \in \Omega$. $R : S \times A \to \mathbb{R}$ is the reward function generating a joint reward $\vec{r}$, and $\gamma \in [0, 1]$ is the discount factor.

M represents the joint macro-action space, where each macro-action $m = \langle I_m, \beta_m, \pi_m\rangle$ consists of the initial set I_m, the stochastic termination condition β_m, and the low-level policy to achieve the macro-action π_m. ζ is the joint macro-observation space. The transformed transition probability, considering the stochastic termination of a macro-action, is $T'(s', t_c, s, \vec{m}) = P(s', t_c \mid s, \vec{m})$, where t_c represents the length of time sequence between two terminated macro-actions. $Z(\vec{z}, \vec{m}, s') = P(\vec{z} \mid \vec{m}, s')$ denotes the joint macro-observation likelihood model when a joint macro-action $\vec{z}$ is taken and the global state arrives in s' to receive a joint macro-observation $\vec{z} \in Z$.

R^c is the reward function generating a macro joint reward $\vec{r_c}$: $R^c(\vec{\tau}, \vec{m}, t_c) = \sum_{t=t_{\vec{m}}}^{t_{\vec{m}}+t_c-1} \gamma^{t-t_{\vec{m}}} r_t$, where $\vec{\tau}$ is a joint macro-action-observation history, $t_{\vec{m}}$ is the start time of joint macro-action $\vec{m}$, and $t_{\vec{m}} + t_c - 1$ is the end time of joint macro-action $\vec{m}$ when any agent finishes its macro-action.

In a Dec-POMDP problem, the solution is a collection of decentralized policies denoted as $\vec{\eta} = (\eta^{(1)}, \ldots, \eta^{(n)})$. Each policy $\eta^{(i)}$ generates the subsequent micro-action $a_t^{(i)} = \eta^{(i)}(h_t^{(i)})$, given the individual micro-action-observation history $h_t^{(i)}$ at timestep t. With this definition, the value associated with a specific policy $\vec{\eta}$ is defined as $V^{\vec{\eta}}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \vec{a_t}) | s, \vec{\eta}]$. Therefore, the optimal policy beginning at state s can be formally defined as $\vec{\eta}^*(s) = argmax_{\vec{\eta}} V^{\vec{\eta}}(s)$. However, without additional assumptions, the Dec-POMDP problem, as defined above, is undecidable over continuous spaces.

In order to accommodate asynchronous scenarios, we consider the problem of heterogeneous multi-robot cooperation with asynchronous execution as a MacDec-POMDP problem, dividing the policy into a joint high-level policy, denoted as $\vec{\mu} = (\mu^{(1)}, \ldots, \mu^{(n)})$, and a low-level policy for executing the macro-action, denoted as π. Each time the high-level policy $\mu^{(i)}$ yields a macro-action $m_t^{(i)}$, the corresponding low-level policy π_m generates the path to accomplish $m_t^{(i)}$, which manifests as a sequence of micro-actions $(a_0^{(i)}, \ldots, a_t^{(i)})$. The value associated with $\vec{\mu}$ is defined as $V^{\vec{\mu}}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \vec{a_t}) | s, \pi, \vec{\mu}]$, and the optimal policy is $\vec{\mu}^*(s) = argmax_{\vec{\mu}} V^{\vec{\mu}}(s)$.

III. Methodology

In this section, we design and improve an MARL algorithm based on value-decomposition networks called Qrainbow. We propose a architecture based on it for multi-robot decision-making and control. This architecture, as shown in Fig. 1, extends the synchronous multi-agent method to asynchronous and heterogeneous multi-robot systems.

A. *Qrainbow*

In this section, we propose the improved Q-value decomposition network called Qrainbow. The pseudo-code of Qrainbow is shown in Algorithm 1. It uses the same mixing network with attention machanism as *Qatten*, and redesigns the executive policy networks for each agent to accommodate the multi-robot

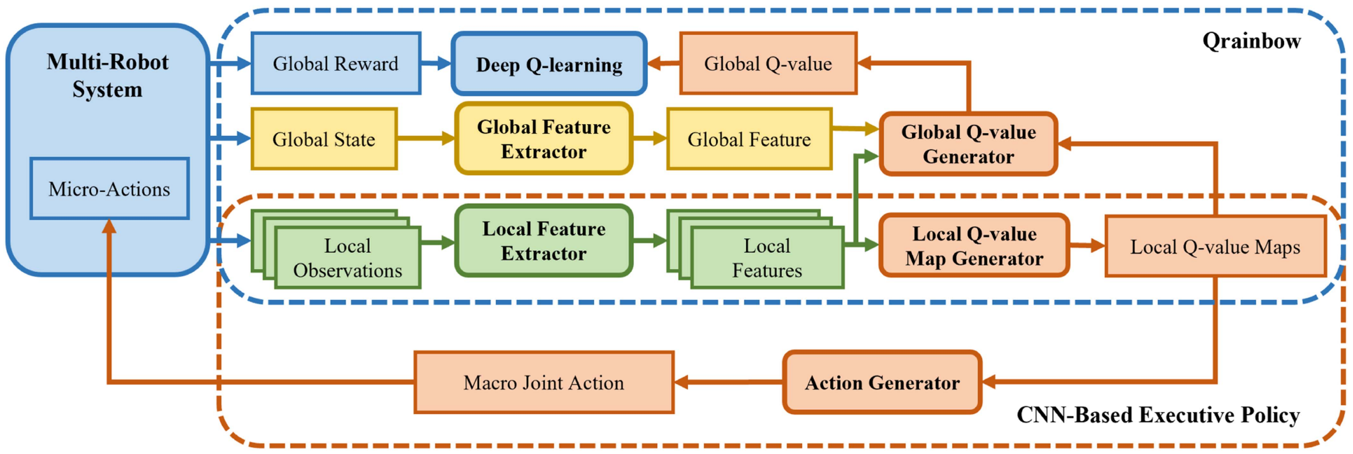

Fig. 1. Overview of our architecture for multi-robot decision-making and control based on multi-agent reinforcement learning. The architecture is composed of two parts: the improved MARL algorithm, Qrainbow, and a CNN-based executive policy. Qrainbow takes both global state and local observations for training, while the CNN-based executive policy takes only local observations to generate joint macro-actions. These macro-actions are then separated into micro-actions and implemented with low-level control.

environment. Moreover, it adds some practical improvements to the executive policy networks, which have been proven effective on DQN, such as *double network*, *duel network* and *prioritized experience replay*. To make Qrainbow work asynchronously, it uses the method of collecting, storing and processing datas in *macro-action joint experience replay trajectories*.

Qrainbow is trained end-to-end by following loss:

$$\begin{aligned}\mathscr{L}(\theta) &= \mathbb{E}_{\langle \vec{z},\vec{m},\vec{z}',\vec{r_c}\rangle\sim\mathscr{D}}\left[w\left(y - Q_{tot}(\vec{\tau},\vec{m},s;\theta)\right)^2\right]\\ &= \sum_{i=1}^{b}\left[w_i\left(y_i^{tot} - Q_{tot}\left(\vec{\tau},\vec{m},s;\theta\right)\right)^2\right], \qquad (1)\end{aligned}$$

where b is the batch size of transitions sampled from the replay buffer $\mathscr{D}$, w_i is the importance sampling weight to correct the bias while using prioritized replay, y^{tot} is the DQN target, and Q_{tot} is a joint action-value function.

Here, w_i is defined as:

$$\begin{aligned}w_i &= \left(\frac{1}{N}\cdot\frac{1}{P(i)}\right)^{\beta}\\ &= \left(\frac{N\cdot p_i^{\alpha}}{\sum_k p_k^{\alpha}}\right)^{-\beta}, \qquad (2)\end{aligned}$$

where N is the size of the replay buffer, $p_i > 0$ is the priority of transition i, α determines how much prioritization is used, and β determines how much to compensate for the non-uniform probabilities. In the practice, we use the temporal-difference error to represent the priority, and normalize weights by $1/max_i w_i$ for stability. And y^{tot} is defined as:

$$y^{tot} = \vec{r^c} + \gamma Q_{tot}\left(\vec{\tau}', \underset{\vec{m}'}{argmax} Q_{tot}(\vec{\tau}',\vec{m}',s';\theta), s';\theta^-\right), \qquad (3)$$

where θ^- are the parameters of a target network as in DQN. Qrainbow uses the same technique as double DQN to replace $\vec{m}'$ so as to suppress the overestimation of action values. Due to the asynchronous execution, macro-actions that are not terminated will continue running to the next time-step and should be considered when choosing the joint macro-action $\vec{m}'$. Therefore, we record whether the macro-actions are terminated at each time-step while collecting data and calculate y^{tot} as with (3) during training. The consistency between the deterministic greedy decentralized policies $argmax_{m^i} Q_i$ and the deterministic greedy centralized policy based on the optimal joint action-value function $argmax_{\vec{m}} Q_{tot}$ follows the rule as:

$$\underset{\vec{m}}{argmax} Q_{tot}(\vec{\tau},\vec{m},s) = \begin{pmatrix} argmax_{m^1} Q_1(\tau^1,m^1)\\ \vdots\\ argmax_{m^n} Q_n(\tau^n,m^n)\end{pmatrix}. \qquad (4)$$

Here, Q_{tot} which uses attention machanism to generate the joint action-value is defined as:

$$\begin{aligned}Q_{tot}(\vec{\tau},\vec{m},s) &= c(s) + \sum_{h=1}^{H} w_h \sum_{i=1}^{n} \lambda_{i,h} Q_i(\tau^i,m^i)\\ &= \sum_{h,i} w_h \lambda_{i,h} Q_i(\tau^i,m^i), \qquad (5)\end{aligned}$$

where $c(s)$ is a constant that depends on the global state s, w_h are weights for Q-values from different heads when using Multi-Head Attention, with H representing the number of heads, and $\lambda_{i,h}$ is a linear functional of all partial derivatives $\frac{\partial^h Q_{tot}}{\partial Q_{i_1}\cdots\partial Q_{i_h}}$ of order h. In practice, $c(s)$ and w_h are retrieved from neural networks based on the global state s, and $\lambda_{i,h} \propto \exp(e_i^{\top}(z^i) W_{k,h}^{\top} W_{q,h} e_s(s))$ is retrieved based on the global state s and agent i's individual macro-observation z^i, with the individual macro-observation's embedding vector $e_i(z^i)$, the global state's embedding vector $e_s(s)$, the global query transformation matrix $W_{q,h}$, and the individual observation transformation matrix $W_{k,h}$.

With regard to individual Q-value functions Q_i of agent i, we separate it into the state value and the advantage:

$$Q_i(\tau^i,m^i) = V(\tau^i) + A(\tau^i,m^i) - mean_{\bar{m}^i\in M^i} A(\tau^i,\bar{m}^i)$$

Algorithm 1: Qrainbow.

```
$Initialize$ the parameters of all networks $\theta$
$Set$ the learning rate $\alpha$ and the replay buffer $\mathscr{D} = \{\}$
$Set$ the learning start step $step_{start}$
$step = 0$, the parameters of target networks $\theta^- = \theta$
while $step < step_{max}$ do
  $t = 0$, $s_0 =$ initial state
  while $s_t \neq terminal$ and $t <$ episode limit do
    for all agents $i = 1$ to $N$ do
      if agent $i$ replans macro-action then
        $j \leftarrow$ agent $i$'s $j$-th macro-actions
        $s_j \leftarrow global\ state$
        $\vec{z_j} \leftarrow joint\ observation$
        $\tau^i_j + = [z^i_j, m^i_{j-1}]$
        $\epsilon =$ epsilon-schedule(step)
        $m^i_j = \begin{cases} argmax_{m^i_j} Q^i(\tau^i_j, m^i_j) \sim 1-\epsilon \\ randint(1, |M^i|) \sim \epsilon \end{cases}$
        $\vec{r^c_j} \leftarrow joint\ reward$
        $s_{j+1} \leftarrow next\ global\ state$
        $\vec{z_{j+1}} \leftarrow next\ joint\ observation$
        $\vec{m_j}^u \leftarrow joint\ macro\text{-}action\ undone$
        $\vec{m_j} = m^i_j + \vec{m_j}^u$
        $p_j \leftarrow maximalpriority$
        $\mathscr{D} + = [s_j, \vec{z_j}, \vec{m_j}, \vec{r^c_j}, s_{j+1}, \vec{z_{j+1}}, \vec{m_j}^u, p_j]$
      end if
      Execute micro-actions $a^i_t \sim m^i_j$
    end for
    $t = t + 1$, $step = step + 1$
  end while
  if $step >= step_{start}$ then
    $Sample$ prioritized batch $b \sim P(b) = p^\alpha_b / \sum_k p^\alpha_k$
    $w_b = (N \cdot P(b))^{-\beta} / max_k w_k$
    $Calculate$ $Q_{tot} \sim \theta$ with (5)
    $Calculate$ target $Q_{tot} \sim \theta^-$ using $\vec{m_b}^u$ with (3)
    $Calculate$ TD-error $\Delta Q_{tot} = y^{tot} - Q_{tot}$
    $Update$ transition priority $p_b \leftarrow |\Delta Q_{tot}|$
    $Calculate$ $\mathscr{L}(\theta) = w_b(\Delta Q_{tot})^2$
    $\Delta\theta = \nabla_\theta \mathscr{L}(\theta)$
    $\theta = \theta - \alpha\Delta\theta$
  end if
  if update-interval steps have passed then
    $\theta^- = \theta$
  end if
end while
```

$$= V(\tau^i) + A(\tau^i, m^i) - \frac{1}{|M^i|}\sum_{\bar{m}^i} A(\tau^i, \bar{m}^i), \quad (6)$$

where M^i is the macro-action space of agent i, and $|M^i|$ is the size of its macro-action space.

B. Overall Architecture

The architecture for multi-robot decision-making and control follows the CTDE rule, facilitating both global and local decision-making to achieve optimal performance in an MRS. During training, global state and local observations are collected and processed by their respective feature extractors to extract relevant information for decision-making. Local Q-value maps, generated from local features, are input into the global Q-value generator along with local and global features. The global Q-value generator produces a global Q-value representing the expected reward for the chosen macro joint action from local Q-value maps. The global feature extractor, local feature extractor, and local Q-value map generator comprise CNN layers that process data hierarchically, extracting increasingly complex features. This enables the extraction of high-level features for decision-making. With the global Q-value and global reward output from the MRS environment, networks are trained through temporal-difference learning. This method updates network parameters based on the difference between predicted and actual rewards. In execution, the CNN-based executive policy takes only local observations as input and generates a macro joint action through the action generator using the argmax of local Q-value maps. The macro joint action represents the optimal action for all robots based on their local observations. It is then separated into micro-actions for each robot and implemented with low-level control to proceed to specified locations along the shortest path. This approach enables each robot to make decisions based on its local observations while coordinating with other robots.

IV. Experiments

In this section, we evaluate the improvements of Qrainbow in the StarCraft Multi-Agent Challenge (SMAC) environment, a commonly used benchmark for evaluating state-of-the-art MARL approaches. We train multiple agents with different combinations in four scenarios to accomplish decentralized micromanagement tasks and compare the winning rate with Qatten and QMIX during the process and at the end. We also evaluate our architecture for multi-robot decision-making and control in a multi-robot environment where decentralized multi-robot teams execute *foraging* and *search and rescue* tasks. We train heterogeneous and homogeneous multi-robot combinations in 16 scenarios and compare the average time cost and average robot collisions with the baseline in [32].

A. Experiments in SMAC

We evaluate Qrainbow without the asynchronous component in the SMAC environment, where agents act synchronously. SMAC comprises StarCraft II scenarios that assess the coordination and problem-solving abilities of independent agents in complex tasks. In each scenario, two armies confront each other with varying numbers, locations, and types of units. The scenarios also vary in the presence of impassable terrain or high ground. At each time step, agents receive local observations within their field of view, including a circle with a radius equal to the line of sight on the map around each cell. Due to line-of-sight constraints, the environment is partially observable from each agent's perspective. Agents can only observe other agents if they are within their line of sight and alive. A global state contains information about all units on the map, which agents can access

TABLE I
MAPS IN THE SMAC ENVIRONMENT

Name	Ally Units	Enemy Units
2s_vs_1sc	2 Stalkers	1 Spine Crawler
3s5z	3 Stalkers	5 Zealots
3s_vs_5z	3 Stalkers & 5 Zealots	3 Stalkers & 5 Zealots
5m_vs_6m	5 Marines	6 Marines

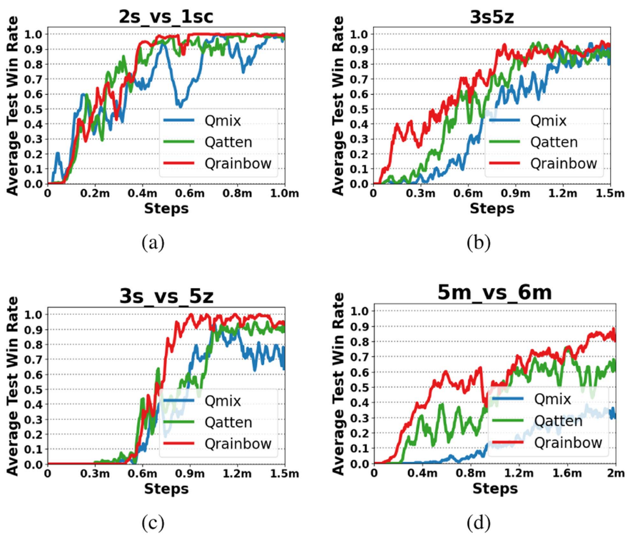

Fig. 2. Average win rate on the easy (a), (b) and hard (c), (d) scenarios.

TABLE II
AVERAGE PERFORMANCE OF THE TEST WIN RATE

Scenario	Qrainbow	Qatten	QMIX
2s_vs_1sc	**100**	**100**	**100**
3s5z	**93**	88	91
3s_vs_5z	**93**	89	64
5m_vs_6m	**80**	61	30

during training. The state vector includes the coordinates of all agents relative to the map center and cell features in the observation. The global state is also augmented by all agents' previous actions. In all combat scenarios, the primary objective is to maximize the win rate. SMAC provides a default reward signal based on health damage received and inflicted by agents, with additional rewards for eliminating enemy units or winning battles.

Table I presents the scenarios selected in SMAC, including two easy scenarios (2s_vs_1sc and 3s5z) and two hard scenarios (3s_vs_5z and 5m_vs_6 m), which concludes homogeneous and heterogeneous, symmetric and asymmetric tasks. In these four scenarios, we implement Qrainbow, Qatten, and QMIX, and evaluate their performance during the process and at the end. For each algorithm in a given scenario, we conduct five experiments and calculate the average result. The time steps of experiments vary across different scenarios due to differences in difficulty. Fig. 2 shows the average win rate during the process on easy and hard scenarios, and Table II presents the average win rate at the end. The result shows that Qrainbow, Qatten and QMIX

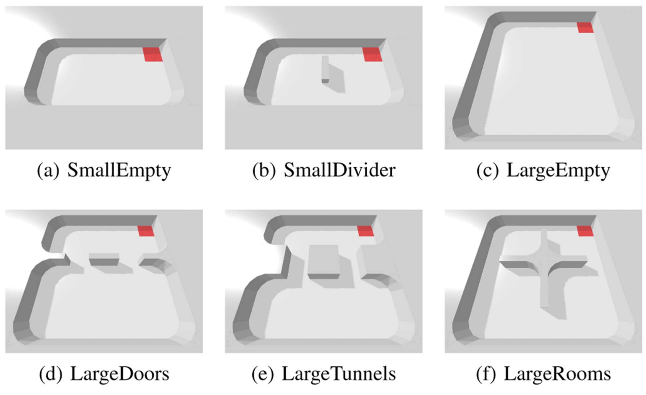

Fig. 3. Six layouts in the multi-robot environment.

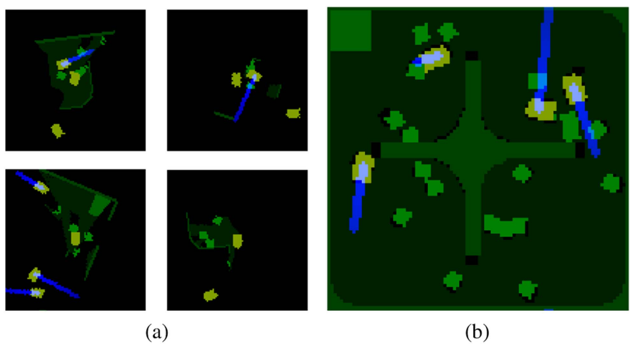

Fig. 4. Visualization of local observations (a) and the global state (b). The observation maps represent the field of view of the robots, the communication between them, and their intended movements through path planning. The global state map gathers all local observations and removes any mist.

perform similarly well in easy scenarios, but Qrainbow can converge to the optimal solution faster. In hard scenarios, QMIX performs significantly worse because of its simple structure on the mixer network. Qrainbow has better performance and faster convergence than Qatten, which proves the efficiency of our improvements.

B. *Experiments in the Multi-Robot Environment*

We evaluate our architecture for multi-robot decision-making and control in the multi-robot environment proposed in [32]. As shown in Fig. 3, it includes six layouts with different designs of the border and obstacles. Robots move and execute specific actions to transport objects in the environment to the red region, which serves as a receptacle. In *search and rescue* scenarios, objects are 'marked' by rescue robots and do not need to be transported to the receptacle. Maps of local observations and the global state is shown in Fig. 4. Observation maps are generated for each robot using a simulated forward-facing RGB-D camera, which is transformed into an overhead image. The observation map of a robot includes: 1) the surrounding environment within its field of view, 2) the observed states of other robots, 3), 4) the distances of the optimal path planning from the robot to a specific pixel location on the map and from that pixel location to the receptacle, and 5) the robot's intended action until its

TABLE III
TIME COST PERFORMANCE

Robots	Environment	Qrainbow	baseline
4L	SmallEmpty	**12.10** ± **0.52**	15.18 ± 0.84
	SmallDivider	**16.25** ± **0.35**	18.11 ± 1.04
	LargeEmpty	**24.42** ± **0.51**	26.55 ± 0.89
	LargeDoors	**39.98** ± **0.45**	47.01 ± 2.96
	LargeTunnels	**47.46** ± **0.75**	55.96 ± 2.57
	LargeRooms	**37.16** ± **0.99**	42.16 ± 1.32
4P	SmallEmpty	**19.87** ± **0.45**	24.72 ± 1.62
	SmallDivider	**32.91** ± **1.45**	38.75 ± 2.98
	LargeEmpty	46.43 ± 1.26	**45.00** ± **1.71**
4R	SmallEmpty	**2.79** ± **0.11**	4.08 ± 0.34
	LargeEmpty	**5.86** ± **0.16**	6.48 ± 0.47
2L + 2P	LargeEmpty	**33.68** ± **0.83**	36.96 ± 0.90
	LargeDoors	**64.41** ± **1.22**	70.27 ± 2.09
	LargeRooms	**45.48** ± **1.13**	54.12 ± 1.87
2L + 2T	LargeEmpty	**28.58** ± **0.83**	38.20 ± 0.92
	LargeDoors	**56.09** ± **1.13**	63.75 ± 2.22

L = lifting, P = pushing, R = rescue, T = throwing

TABLE IV
ROBOT COLLISION PERFORMANCE

Robots	Environment	Qrainbow	baseline
4L	SmallEmpty	**12.10** ± **2.02**	12.35 ± 1.79
	SmallDivider	**2.85** ± **0.55**	9.45 ± 2.03
	LargeEmpty	**17.35** ± **1.67**	26.10 ± 2.97
	LargeDoors	**28.70** ± **1.98**	73.20 ± 7.39
	LargeTunnels	**23.30** ± **1.59**	37.75 ± 4.13
	LargeRooms	**23.85** ± **2.93**	42.60 ± 2.77
4P	SmallEmpty	**2.10** ± **0.36**	5.95 ± 0.81
	SmallDivider	**8.40** ± **1.24**	33.90 ± 4.39
	LargeEmpty	**5.30** ± **0.49**	11.35 ± 1.06
4R	SmallEmpty	**0.90** ± **0.23**	2.55 ± 0.77
	LargeEmpty	**1.70** ± **0.38**	4.05 ± 1.60
2L + 2P	LargeEmpty	**12.50** ± **1.43**	19.75 ± 3.51
	LargeDoors	**18.60** ± **1.75**	33.15 ± 3.30
	LargeRooms	**25.95** ± **2.94**	42.55 ± 6.89
2L + 2T	LargeEmpty	**11.35** ± **1.46**	15.70 ± 1.93
	LargeDoors	**37.20** ± **3.06**	45.00 ± 6.68

L = lifting, P = pushing, R = rescue, T = throwing

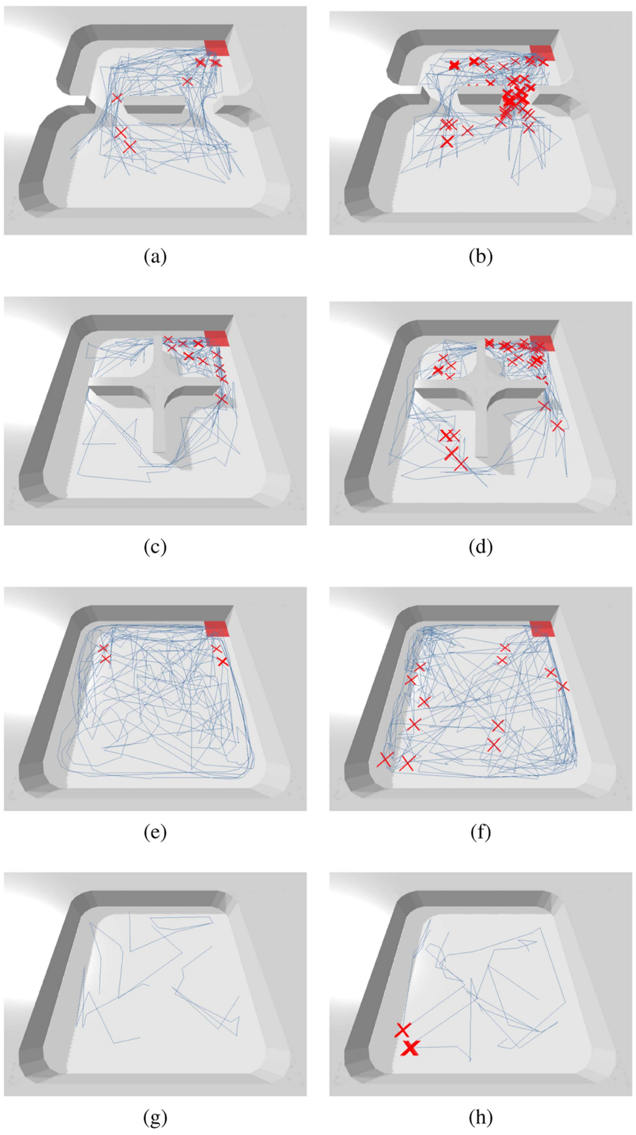

Fig. 5. Comparison of robot collisions using our method (left) and the baseline method (right) in LargeDoors with lifting robots (a), (b), LargeRooms with lifting robots (c), (d), LargeEmpty with pushing robots (e), (f), and LargeEmpty with rescue robots (g), (h). Red crosses indicate robot collisions on the map and blue lines indicate trajectories of robots.

next decision-making. The global state map, which is the same size and dimension as the observation maps, gathers all local observations and removes any mist.

The action space is spatially aligned with the observation map and varies in dimension depending on the robot type. All robots have the ability to move, enabling them to travel to specific locations within their environment. Lifting, throwing, and rescue robots have an additional dimension for executing specific actions with their end effectors, such as lifting objects, throwing objects behind, and marking objects as 'rescued'. During decentralized execution, each robot generates its own macro-action using its individual Q-network, a process that is inherently asynchronous. During centralized training, we track whether the macro-actions terminate at each time-step. This data collection simulates asynchronous execution while formulating the joint action-value. When the local Q-value map generator processes the observation and generates a Q-value map, the robot and its end effector will choose and execute the macro-action with the highest Q-value. This corresponds to the robot moving directly to the position associated with the selected pixel on the map. The chosen macro-action is subsequently divided into several micro-actions, which are executed using low-level control via a trajectory planning algorithm such as SPFA or A*.

In each scenario, we evaluate a trained policy by calculating the average performance across 20 episodes and record the mean and standard deviation of time costs and robot collisions. Our method is contrasted with the baseline method that employs independent Q-learning with spatial intention maps [32]. Table III presents the comparison of time costs in 16 scenarios. The results indicate that our method consistently meets task requirements faster than the baseline across almost all scenarios. Notably,

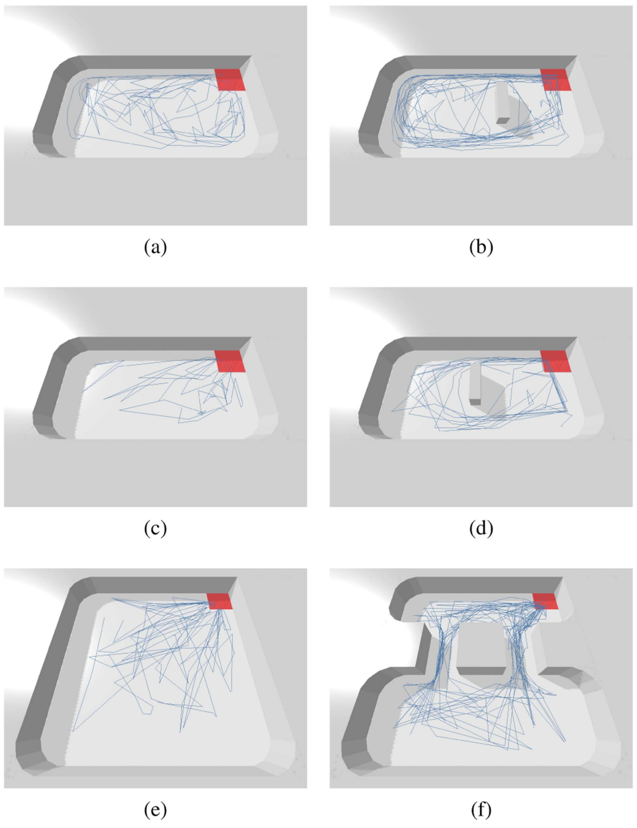

Fig. 6. Cooperation of homogeneous robots in different scenarios with pushing robots (a), (b) and lifting robots (c)–(f).

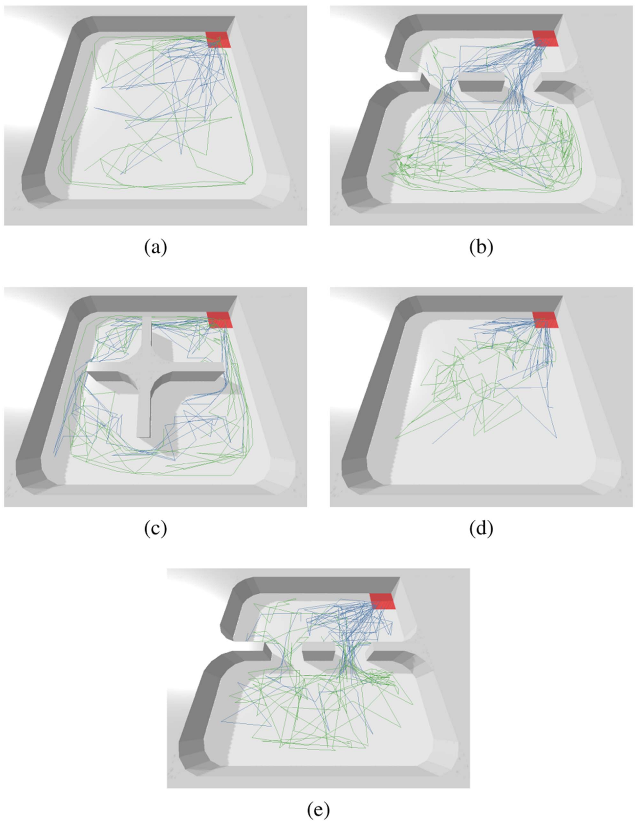

Fig. 7. Cooperation of heterogeneous robots in different scenarios with a combination of lifting and pushing robots (a)–(c) and a combination of lifting and throwing robots (d), (e). Blue lines indicate trajectories of lifting robots and green lines indicate trajectories of pushing robots or throwing robots.

in complex scenarios such as LargeDoors and LargeTunnels with lifting robots, LargeDoors and LargeRooms with lifting and pushing robots, and LargeRooms with lifting and throwing robots, robots trained using our method can save approximately 8% to 16% of the time. However, in the LargeEmpty scenario with pushing robots, the baseline method outperforms our method in terms of time cost. This may be due to the excessive collision avoidance of our trained policy.

Table IV presents the comparison of robot collisions in 16 scenarios. The results indicate that our method outperforms the baseline in avoiding robot collisions across all scenarios. While the difference is not so significant in simple scenarios, our method's superiority in collision avoidance is significantly evident in larger and more complex scenarios. The robot team trained using our method experiences nearly half the number of collisions. Notably, in the LargeDoors scenario with lifting robots, our robot team encounters only 40% of the collisions experienced by the baseline. Fig. 5 shows a visual comparison of robot collisions using our method and the baseline method.

On the map, red crosses represent locations where robots collide with each other, while blue lines indicate their trajectories. It is evident that our trained policy performs better in both path planning and collision avoidance, as indicated by the reduced number of red crosses and the sparsity of trajectories. We hypothesize that this difference is due to our method providing the robot team with full access to all situations during training, without any mist. This results in more efficient and intelligent decentralized execution. In contrast, the baseline method is limited by its use of only partial observation during training. We also observed that robot collisions occur more frequently near receptacles and narrow passages. This may be due to the layouts being insufficiently large for a four-robot team, causing them to congregate more often. Additional trajectories of homogeneous robot cooperation are depicted in Fig. 6. In scenarios where obstacles are placed in the middle, robots have learned to move around the obstacles one by one, forming a circle to avoid collisions.

In heterogeneous multi-robot cooperation tasks, our method achieves satisfactory results. As shown in the bottom five rows of Tables III and IV, our trained policies provide better plans for reducing time costs and avoiding collisions in heterogeneous tasks. This effect is more pronounced in complex scenarios. Fig. 7 illustrates the visual cooperation of heterogeneous robots in various scenarios using a combination of different robot types.

The blue trajectories represent lifting robots, while the green trajectories represent pushing robots in (a)–(c) and throwing robots in (d)–(e). It is noteworthy that pushing robots have learned to push objects into the receptacle using borders and to carry multiple objects simultaneously. Lifting robots assist by bringing objects left behind by pushing robots into the receptacle. Throwing robots focus on throwing objects far from the receptacle, allowing lifting robots to save time by not having to travel long distances or through doors. When objects are close to the receptacle, throwing robots can also assist by pushing objects to improve efficiency.

V. Conclusion

In this letter, we propose a novel architecture for multi-robot decision-making and control based on multi-agent reinforcement learning. We first propose an improved MARL algorithm called Qrainbow, which incorporates numerous improvements to the value decomposition network to accommodate asynchronous and heterogeneous environments. These improvements have been demonstrated to be effective in the SMAC environment. Furthermore, we apply our architecture to homogeneous and heterogeneous multi-robot cooperation tasks and achieve superior performance in terms of reducing time costs and avoiding collisions between robots. During our experiments, we discover that incorporating the global state during training facilitates more efficient and intelligent cooperation among the robot team.

In our future work, we plan to conduct physical experiments and address the sim-to-real problem. Due to the constraints of the scene size, it is disappointing that we cannot entirely prevent touches between robots and guarantee collision-free paths. Applying our approach in a larger space with more aggressive optimizations for obstacle avoidance, rather than merely assigning a penalty in the reward function, may yield better performance. Moreover, obtaining overhead images, which provide abundant information in our experiments, may not be as straightforward in real-world scenarios involving larger spaces. Therefore, further research is needed on how to leverage sensor data, such as from IMUs and LiDARs, more effectively and how to handle errors at the transport level.

References

[1] W. Dai, Y. Liu, H. Lu, Z. Zheng, and Z. Zhou, “A shared control framework for human-multirobot foraging with brain-computer interface,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6305–6312, Oct. 2021.

[2] J. Banfi, A. Messing, C. Kroninger, E. Stump, S. Hutchinson, and N. Roy, “Hierarchical planning for heterogeneous multi-robot routing problems via learned subteam performance,” *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4464–4471, Apr. 2022.

[3] Y. Chang et al., “LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments,” *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9175–9182, Oct. 2022.

[4] D. Koung, O. Kermorgant, I. Fantoni, and L. Belouaer, “Cooperative multi-robot object transportation system based on hierarchical quadratic programming,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6466–6472, Oct. 2021.

[5] E. Tuci, M. H. Alkilabi, and O. Akanyeti, “Cooperative object transport in multi-robot systems: A review of the state-of-the-art,” *Front. Robot. AI*, vol. 5, 2018, Art. no. 59.

[6] G. Rishwaraj and S. Ponnambalam, “Integrated trust based control system for multirobot systems: Development and experimentation in real environment,” *Expert Syst. Appl.*, vol. 86, pp. 177–189, 2017.

[7] T. Ming, “Multi-agent reinforcement learning: Independent versus cooperative agents,” in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, pp. 330–337.

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

[9] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

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

[11] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[12] K. Zhang, Z. Yang, and T. Başar, “Multi-agent reinforcement learning: A selective overview of theories and algorithms,” in *Handbook Reinforcement Learning Control*. Berlin, Germany: Springer, 2021, pp. 321–384.

[13] C. Yu et al., “The surprising effectiveness of ppo in cooperative multi-agent games,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 24611–24624.

[14] P. Sunehag et al., “Value-decomposition networks for cooperative multi-agent learning based on team reward,” in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 2085–2087.

[15] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Monotonic value function factorisation for deep multi-agent reinforcement learning,” *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 7234–7284, 2020.

[16] Y. Yang et al., “Qatten: A general framework for cooperative multiagent reinforcement learning,” 2002, *arXiv:2002.03939*.

[17] S. Mahadevan and J. Connell, “Automatic programming of behavior-based robots using reinforcement learning,” *Artif. Intell.*, vol. 55, no. 2/3, pp. 311–365, 1992.

[18] P. Stone and M. Veloso, “Towards collaborative and adversarial learning: A case study in robotic soccer,” *Int. J. Hum.- Comput. Stud.*, vol. 48, no. 1, pp. 83–104, 1998.

[19] C. Amato, G. Konidaris, L. P. Kaelbling, and J. P. How, “Modeling and planning with macro-actions in decentralized pomdps,” *J. Artif. Intell. Res.*, vol. 64, pp. 817–859, 2019.

[20] A. Christopher, D. K. George, and P. K. Leslie, “Planning with macro-actions in decentralized POMDPs,” in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2014.

[21] Y. Xiao, J. Hoffman, and C. Amato, “Macro-action-based deep multi-agent reinforcement learning,” in *Proc. Conf. Robot Learn.*, 2020, pp. 1146–1161.

[22] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, “Deep reinforcement learning for decentralized multi-robot exploration with macro actions,” *IEEE Robot. Automat. Lett.*, vol. 8, no. 1, pp. 272–279, Jan. 2023.

[23] O. Nachum, M. Ahn, H. Ponte, S. S. Gu, and V. Kumar, “Multi-agent manipulation via locomotion using hierarchical Sim2Real,” in *Conf. Robot Learn.*, 2020, pp. 110–121.

[24] R. Han et al., “Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards,” *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 5896–5903, Jul. 2022.

[25] M. Hessel et al., “Rainbow: Combining improvements in deep reinforcement learning,” in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innov. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 3215–3222.

[26] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1.

[27] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[28] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, “Dueling network architectures for deep reinforcement learning,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[29] K. De Asis, J. F. Hernandez-Garcia, G. Z. Holland, and R. S. Sutton, “Multi-step reinforcement learning: A unifying algorithm,” in *Proc. 32nd AAAI Conf. Artif. Intell. 30th Innov. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 2902–2909.

[30] M. G. Bellemare, W. Dabney, and R. Munos, “A distributional perspective on reinforcement learning,” in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, vol. 70, pp. 449–458.

[31] M. Fortunato et al., “Noisy networks for exploration,” in *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[32] J. Wu, X. Sun, A. Zeng, S. Song, S. Rusinkiewicz, and T. Funkhouser, “Spatial intention maps for multi-agent mobile manipulation,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8749–8756.