IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 14 June 2024, accepted 30 July 2024, date of publication 6 August 2024, date of current version 15 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3439354

RESEARCH ARTICLE

Farthest Agent Selection With Episode-Wise Observations for Real-Time Multi-Agent Reinforcement Learning Applications

HYUNSOO LEE[1], GYU SEON KIM[1], MINSEOK CHOI[2], HANKYUL BAEK[1], AND SOOHYUN PARK[3], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, Korea University, Seoul 02841, Republic of Korea
[2]Department of Electronic Engineering, Kyung Hee University, Yongin 02447, Republic of Korea
[3]Division of Computer Science, Sookmyung Women's University, Seoul 04310, Republic of Korea

Corresponding authors: Minseok Choi (choims@khu.ac.kr) and Soohyun Park (soohyun.park@sookmyung.ac.kr)

This work was supported by the National Research Foundation under Grant 2022R1A2C2004869.

ABSTRACT Multi-agent reinforcement learning (MARL) algorithms have been widely used for many applications requiring sequential decision-making to maximize the expected rewards through multi-agent cooperation. However, MARL faces significant challenges, particularly in resource-limited real-time computing environments. To tackle this problem, this paper considers the selection of agents for training which can be beneficial in terms of computation-overhead reduction. For the selection, a farthest agent selection (FAS) is proposed, inspired by the farthest point sampling for representative sample selection in 3D point cloud processing. The proposed FAS method is able to choose agents based on their episode-specific observations in real-time. Additionally, the number of selected agents can be determined based on the real-time variances in the observations of each agent. The proposed FAS method is rigorously evaluated using the StarCraft Multi-Agent Challenge (SMAC) and Predator-Prey (PP) tasks, demonstrating superior performance compared to existing MARL algorithms. This research is scalable, and thus, it can contribute to the development of more efficient MARL training methodologies for various applications such as real-time strategy games and human-robot cooperation scenarios requiring multi-agent cooperation under partial observability.

INDEX TERMS Multi-agent reinforcement learning, agent selection, StarCraft multi-agent challenge (SMAC).

I. INTRODUCTION

A. BACKGROUND AND MOTIVATION

Reinforcement learning (RL) is a branch of machine learning where agents learn sequential decision-making through interactions with their environment. The primary goal of RL is to maximize cumulative expected rewards over time, thereby enabling agents to develop strategies for optimal action sequences. Recent advancements in RL have demonstrated significant progress in various complex decision-making tasks in emerging applications [1], [2], [3], [4], [5]. Particularly in the realm of multi-agent reinforcement learning (MARL), which involves complex interactions and teamwork among agents, e.g., real-time strategy (RTS) games like StarCraft providing a variety of scenarios. These scenarios are conducive to evaluating agent performance across different situations. This approach allows researchers to analyze results and develop strategies applicable to real-world settings where agents do not have access to complete information [6], [7], [8]. Since real-world scenarios often involve agents operating with incomplete information or partial observation, a lot of studies utilize platforms that simulate such complex conditions to explore real-time learning strategies.

One of the challenging tasks in MARL is cooperation among heterogeneous agents [9], [10], [11]. The

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

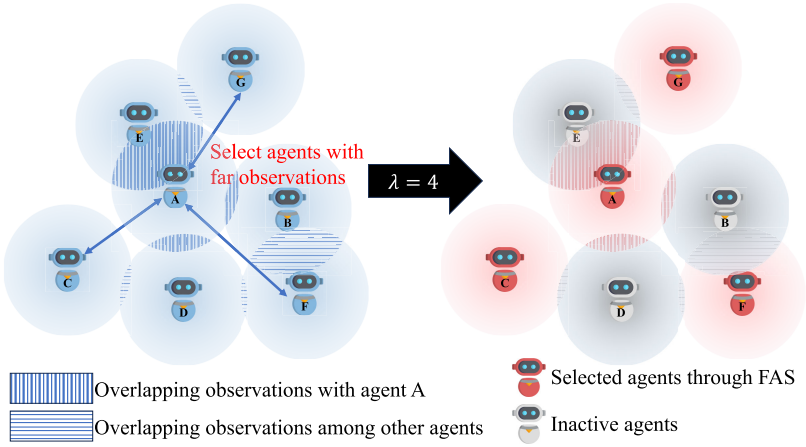


FIGURE 1. Concept of farthest agent selection in MARL.

collaboration among agents with different characteristics reflects the high complexity of real-world systems [12], [13], [14]. To achieve this goal, a centralized training and decentralized execution (CTDE) approach is effective as it facilitates learning from each agent's partial observations [15], [16]. However, when the partial observations among multiple agents are similar, using all agents' partial observations for learning can be wasteful and even hinder the training process from the perspective of the central network. Therefore, selecting a representative subset of agents and using their observations only to update the central network can yield computational benefits relative to where all agents are involved in the neural network training scheme.

Considerable research has focused on algorithms such as attention, which enhances overall system efficiency and performance [17]. This is achieved by prioritizing key information from agents and allocating less focus to less important details, thereby managing the computational resources more efficiently. The *attention* mechanism has also been utilized in RL under stochastic partial observability, where it has been employed to solve decentralized coordination problems [18], [19], [20]. However, to the best of our knowledge, there has not been any research for controlling the actual number of agents, which is our main objective.

B. ALGORITHM DESIGN RATIONALE

This paper proposes a control algorithm that adaptively selects agents for training the central network to reduce the model size and learning time, as described in Fig. 1. In Fig. 1, each agent independently observes the partial information of the entire system, some of which can overlap with those of nearby agents. The central network for taking global action-value functions can be trained using the partial observations of all agents, but the overlapped observations can delay and even hinder the training process. For example, in Fig. 1, agent A's observation overlaps with those of agents B, D, and E, and there is no overlap with agents C, F, and G. In this case, it would be beneficial to have agents A, C, F, and G at least participate in updating the central network to accelerate the training while maintaining the learning performance. The selection process involves determining the number of agents to participate in the current epoch and choosing specific agents. At first, given the number of agents participating in each epoch, we propose the farthest agent selection (FAS) algorithm that selects appropriate agents for updating the central network, inspired by farthest point sampling (FPS), where the FPS is widely used in 3D point cloud environments for downsampling by representing the entire population with a few representative points; therefore, the union of partial observations of agents selected by our FAS algorithm can represent the critical state information for updating joint action-value functions. Furthermore, the guideline of determining the number of agents for updating the central network is presented. Intuitively, a small variance of partial observations can be measured among agents having large overlaps within their observations; on the other hand, a large variance indicates that agents' observations are exclusive. Based on this concept, we experimentally demonstrate that our FAS algorithm can accelerate the training process of MARL and reduce the model parameters for the joint action while maintaining the learning performance.

C. CONTRIBUTIONS

The major key contributions of this research can be summarized as follows.

- A novel agent selection algorithm that selects representative agents of the entire system states and utilizes their observations only for updating the mixing network which estimates the joint action-value function. This approach accelerates the training of MARL and effectively reduces the computational complexity of the neural network while maintaining performance. Additionally, the number of agents required for training the mixing network can be adaptively determined based on the variance of partial observations, allowing the

proposed algorithm to be applied in a variety of environments and time-varying conditions.
- Extensive simulation results demonstrate that the proposed scheme can achieve comparable learning performances to the conventional method in which observations of all agents are used for updating the mixing network, despite allowing fewer agents to participate in each epoch.

D. ORGANIZATION

This paper is organized as follows: Sec. II surveys related research, and Sec. III discusses the necessary background knowledge. Sec. IV provides the details of the proposed FAS algorithm, while Sec. V offers an in-depth analysis of the performance evaluation. Sec. VI concludes this paper.

II. RELATED WORK

A. MARL AND AGENT SELECTION ALGORITHM

In the early days of multi-agent systems, methods to find the optimal policy by having agents directly learn value functions or policies are widely studied. However, this methodology, i.e., independent Q-learning (IQL) ignores the existence of other agents and processes each agent independently, one agent's strategy cannot affect other agents [21]. It makes the environment unpredictable for agents and makes the learning process unstable. The communication network (CommNet), designed to tackle this challenge, facilitates communication among agents during the training phase. It employs a centralized network architecture for the dissemination of information among agents, thereby fostering the development of a more cooperative multi-agent system [22].

Besides, in the field of MARL, numerous efforts have been made to solve multi-agent cooperation problems where each agent has partial observations. To solve these problems, the QMIX algorithm integrates a collective action-value function formulated by merging the action-value functions of individual agents, as indicated in [6].

QTRAN also learns cooperation policies for agents by transforming the joint action-value function in a way that allows for efficient and effective optimization [7]. Unlike typical MARL approaches, QTRAN transforms the joint action-value function to align it with a factored representation. Additionally, role-diverse Q-learning for MARL, i.e., roles to decompose (RODE) introduces a novel approach to MARL by incorporating the concept of role diversity into the learning process [23]. It focuses on learning diverse roles for agents based on their situational actions and interactions. However, these MARL algorithms suffer from large computations and difficulty in learning convergence as the state/action space and number of agents increase. Therefore, it is important to reduce the size of the observation using a selection algorithm such as FPS. In this context, by using RL and FPS in the point cloud to reduce the number of points and network size, the combination of RL and FPS optimized the trajectory of the mobile 3D sensor as quickly as possible [24].

B. COOPERATIVE MISSION EXECUTION APPLICATION USING MARL

MARL encourages multiple agents to achieve cooperative mission performance, such as the system in this paper. An efficient air transportation service algorithm is proposed in [25] and [26] in which multiple urban aerial mobilities (UAMs) cooperate with each other to transport passengers to target vertiports using CommNet method utilizing centralized training and distributed execution (CTDE). The QMIX finds application in trajectory optimization for multiple electric vertical takeoff and landing vehicles (eVTOLs), particularly in the domain of aerial drone-taxi services, as discussed in [27].

Additionally, in many multi-agent applications such as distributed networks, it has been confirmed that agent networks learn cooperatively, showing the effectiveness of MARL in distributed systems [28]. By replacing the optimal resource allocation problem with distributed decision-making using autonomous agents, an interaction mechanism was proposed that maintains a balance between competition and cooperation to encourage agents [4].

III. PRELIMINARIES

In this section, we aim to provide a brief discussion on FPS and QMIX to facilitate a better understanding of the technology we propose. We discuss the key features of FPS and the fundamental principles of the QMIX algorithm.

A. FARTHEST POINT SAMPLING (FPS)

FPS is a technique commonly used in various fields, including computer graphics, computational geometry, and machine learning. Its primary purpose is to efficiently sample a subset of points from a larger set, ensuring that the sampled points are spread out evenly across the entire set. This is particularly useful in applications to reduce the size of a dataset while maintaining its overall structure and diversity.

The key feature of FPS is its focus on spreading out the sampled points. The concept of FPS is also illustrated in Fig. 2. On the left side of the figure, in the beginning of the algorithm, p_1 and p_3 are selected randomly. p_1 is chosen randomly, and the red line indicates the distance between p_1 and p_3, which is the farthest among all point pairs originating from p_1, as shown in the table next to it. The table lists the distances between p_1 and all other points, with the farthest distance encircled to indicate it is the maximum. On the right side, the figure shows p_3 and p_5 as selected points, with a blue line connecting them. After p_3 is selected, p_5 is the farthest point from p_3, so p_5 is selected through the FPS algorithm. FPS algorithm is particularly useful in scenarios like downsampling a point cloud in 3D modeling to retain as much of the original shape's detail as possible with fewer points or in machine learning for selecting diverse training samples [29], [30].

B. QMIX

QMIX [6] is inspired by the *value decomposition networks (VDN)* [31], which apply a centralized yet factored total

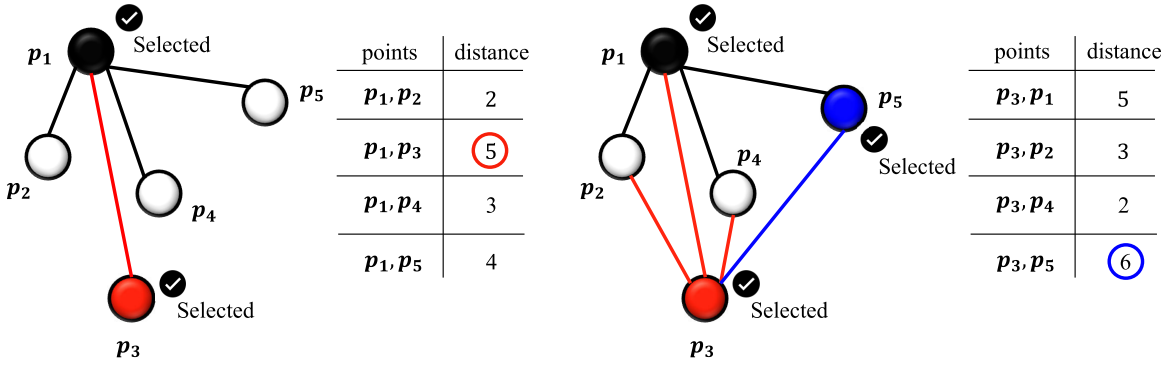


FIGURE 2. Farthest point sampling concept.

Q-value Q_{tot}. In QMIX, Q_{tot} is represented as the sum of each agent's value function, conditioned only on their individual observations and actions, enabling agents to greedily choose actions based on their Q-values Q_a. However, VDN has limitations in that it significantly restricts the complexity of representable centralized action-value functions and may ignore possible extra state information during the training. To overcome these limitations, QMIX does not fully factorize as VDN does for extracting decentralized policies. Instead, Q_{tot} in QMIX is divided and represented by individual agent networks, a mixing network, and a set of hypernetworks.

IV. FARTHEST AGENT SELECTION (FAS) METHOD

A. BACKGROUND

1) DECENTRALIZED PARTIALLY OBSERVABLE MARKOV DECISION PROCESS

A cooperative multi-agent task can be modeled as a *decentralized partially observable Markov decision process* (Dec-POMDP) [32], consisting of $G = \langle S, A, I, P, r, O, Z, n, \gamma \rangle$. Here, $s \in S$ represents the true state of the environment. At time t, each agent i where $i \in I \equiv \{1, \cdots, n\}$, selects an action a^N, resulting in a joint action $\mathbf{a} \in \mathbf{A} \equiv A^i$. This action causes a transition in the environment governed by the state transition function $P(s_{t+1}|s_t, \mathbf{a}_t) : S \times \mathbf{A} \times S \rightarrow [0, 1]$. All agents share the same reward function $r(s, \mathbf{a}) : S \times \mathbf{A} \rightarrow \mathbb{R}$. Lastly, $\gamma \in [0, 1)$ denotes a discount factor. Concerning partial observability, each agent i has its individual observations $o \in O$, as determined by the corresponding observation function $Z(s, i) : S \times I \rightarrow O$. The agents' respective action-observation histories $\tau^i \in T \equiv (O \times A)$, form the foundation for conditioning their stochastic policies, expressed as $\pi^i(a^i|\tau^i) : T \times \mathbf{A} \rightarrow [0, 1]$. These policies cumulatively induce a joint action-value function:

$$Q^{\pi}(s_t, \mathbf{a}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{a}_{t+1:\infty}}[R_t|s_t, \mathbf{a}_t] \tag{1}$$

where $R_t = \sum_{k=0}^{\infty} \gamma^i r_{t+k}$ represents the discounted accumulated reward. The objective of our proposed method is for identifying the joint optimal policy which can be denoted as π^* that satisfies,

$$Q^{\pi^*}(s, \mathbf{a}) \geq Q^{\pi}(s, \mathbf{a}) \tag{2}$$

for every policy π and each pair $(s, \mathbf{a})$ within the Cartesian product $S \times \mathbf{A}$. The Bellman optimality operator is defined as follows,

$$\mathcal{T}^* Q(s, \mathbf{a} := \mathbb{E}[r + \gamma \max_{\mathbf{a}'} Q(s', \mathbf{a}')], \tag{3}$$

where the expectation is over the next state $s' \sim P(\cdot|s, \mathbf{a})$ and reward $r \sim r(\cdot|s, \mathbf{a})$. As QMIX is based on Q-learning, it utilizes samples from the environment to calculate the expectation in (3), to update their estimates of Q^*. QMIX estimates the optimal joint action value function Q^* as Q_{tot}, which combines the utilities of each agent through the continuous monotonic function [6].

2) SAMPLING METHOD

The proposed algorithm utilizes FPS-inspired approaches on top of the MARL framework because it should be able to select the most representative agents from the entire population. One possible approach for selecting agents is uniform sampling, in which each data has an equal probability of being chosen. However, this method may not be able to perfectly represent the characteristics of the population, and especially when the population size is small, it can yield inaccurate results due to overfitting. Methods such as systematic sampling and stratified sampling are available to address this. Here, systematic sampling involves the random selection of the first sample and then determines subsequent samples. This method has the potential to introduce sample bias and poses a risk of manipulation with desired data when setting up the system initially. Additionally, stratified sampling can be used when there are various types of data. It involves first dividing the population into strata and then sampling from each stratum, ensuring the sampled results are not biased towards one group. This method requires prior knowledge about the population to classify the strata and can be subjective as it may incorporate personal opinions in the stratification process.

B. ALGORITHM DESIGN CONCEPT

In QMIX, the partial observations of all agents were fed into the mixing network as an input. However, this method can lead to computational overheads by fully utilizing

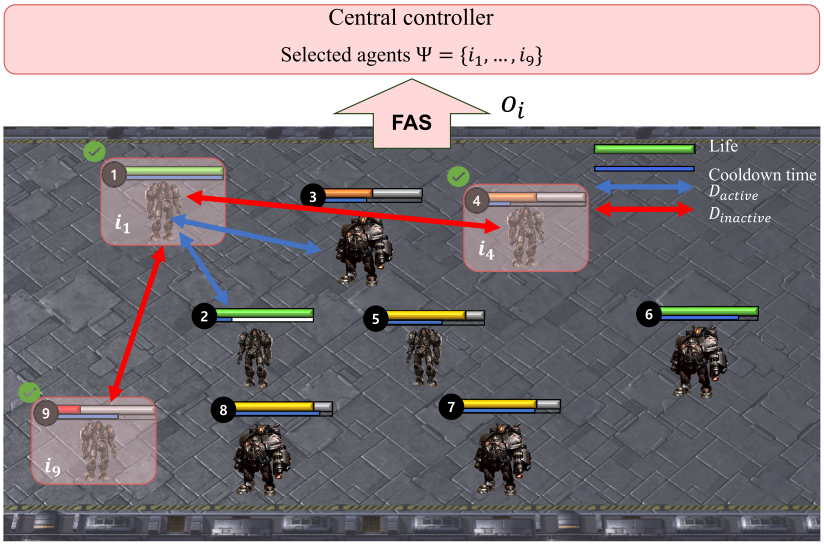


FIGURE 3. Farthest agent selection through observations.

massive amounts of unnecessary information. We select agents based on FPS to reduce computational burdens while employing relatively necessary information for learning, in a selective way. Typically, the information that an agent can obtain from its environment is highly diverse. For example, in the StarCraft Multi-Agent Challenge (SMAC) [33] environment, agents have partial observations for both allies and enemies [34]. In SMAC, the observations of agent i, i.e., o_i, include relative position, health, and cooldown time for its own skills. However, each agent includes the position and health information of the enemies for those in their field of vision.

Fig. 3 depicts the concept of the proposed algorithm in the simulation environment. Based on the partial observations of each agent, λ number of agents can be selected. The observations of selected agents at $t(o_i)$ and their actions from the previous time step a_{t-1} are forwarded to each respective agent network. The mixing network takes the calculated Q-values as an input, and mixes them to maximize Q_{tot} (not individual Q-values).

C. FARTHEST AGENT SELECTION FOR MARL

This section proposes a novel QMIX-based MARL approach with FAS for reducing computational resource usage. "Farthest agents" are defined as the agents with the most heterogeneous observations, where the heterogenous observations differ from the physical distances. The reason why the most heterogeneous agents are considered is that it can be the most representative for the observations which are not covered by the other agents.

1) AGENT NETWORK

To deal with the partial observation, each agent i employs a deep recurrent Q-network (DRQN) with θ_i to evaluate its own Q-value $Q_i(\tau^i, a_t^i; \theta_i)$. The DRQN-based approach substitutes the full connection layer in DQN with the gated recurrent unit (GRU). Therefore, it takes the current observation o_t^i and the last action a_{t-1}^i as input. Then the hidden state of the agent network h_{t-1}^i, which demonstrates the memory of action-observation history τ^i, is circularly input and updated. Then, whole state history can be utilized to fit the Q-value in place of only adapting the current observation, thus reducing the adverse effects caused by partially observable conditions. After obtaining the Q-value of each possible action, ϵ-greedy is adopted to select action a_t^i of agent i. Finally, the agent network outputs the value function of the selected action. Based on this design concept, the agent network Q-value can be updated as,

$$Q_i(\tau^i, a_t^i; \theta_i) \leftarrow Q_i(\tau^i, a_t^i; \theta_i) \\ \alpha \left[r_t^i + \gamma \max_{a'} Q_i(\tau_{t+1}^i, a'; \theta_i) - Q_i(\tau^i, a_t^i; \theta_i) \right], \quad (4)$$

where α is the learning rate, r_t^i is the reward received by agent i at t, and τ_{t+1}^i is agent i's updated action-observation history.

2) AGENT SELECTION

To accelerate the training process, a subset of agents, denoted by $\Psi \subseteq \{1, 2, \cdots, n\}$, is selected by FAS to update both mixing and agent networks. Here, the number of selected agents is predetermined as $|\Psi| = \lambda$ and can vary with each training batch. The computational procedure of our proposed algorithm is described in Algorithm 1. In our proposed MARL-based agent selection algorithm, the diversity of agents' observations is utilized to select agents. The whole agent selection process is described in Fig. 4. Our proposed agent selection algorithm begins with a randomly selected initial agent i_0 and an initial set of selected agents $\Psi = \{i_0\}$, given the set of the observations of each agent i, i.e., $O = \{o^1, \cdots, o^N\}$ from transition τ. Denoting the Euclidean distance between observation vectors of agents i and j as $d(o^i, o^j)$, the agent whose observation has the furthest distance from the agent i_0 is added to Ψ. Similarly, we sequentially add agents with observation vectors furthest

Algorithm 1 Farthest Agent Selection

Input : Set of the normalized observations $\tilde{O}$,
Number of samples λ
Output : Subset of agents Ψ such that $|\Psi| = \lambda$
Initialize an empty set: $\Psi \leftarrow \phi$
Select an initial agent i_0 from I
$\Psi \leftarrow \{i_0\}$
for $n = 1$ to $\lambda - 1$ **do**
  Find the agent $i_n \in I \setminus \Psi$ according to (8)
  $\Psi \leftarrow \Psi \cup \{i_n\}$
end for
return Ψ

from the previously selected agents to Ψ until λ agents are selected. To fairly consider multiple observations, the proposed algorithm normalizes the observations component-wisely via min-max normalization [35], as follows,

$$\tilde{o}^i = \frac{o^i(k) - o_{\min}(k)}{o_{\max}(k) - o_{\min}(k) + \delta}, \tag{5}$$

$$o_{\min}(k) = \min\{o^1(k), \ldots, o^N(k)\}, \tag{6}$$

$$o_{\max}(k) = \max\{o^1(k), \ldots, o^N(k)\}, \tag{7}$$

where $o^i(k)$ is the kth component of the observation vector o_i, i.e., $o_i = [o_i(1), \cdots, o_i(K)]$, and δ is a very small constant. Accordingly, the proposed FAS algorithm fills Ψ until $|\Psi|$ becomes λ by sequentially adding the agent whose normalized observation vector is the most distant from observations of all selected agents, and it can be described as follows:

$$i_n = \arg\max_{i \in I \setminus \Psi} \min_{j \in \Psi} d(\tilde{o}^i, \tilde{o}^j), . \tag{8}$$

where $\Psi \leftarrow \Psi \cup \{i_n\}$. However, performing FAS at every time step requires a significant amount of computation. To maintain performance while reducing computational load, after executing FAS for each episode extracted from the experience buffer, the most frequently selected agents across mini-batches are identified. In each mini-batch, the top-λ most frequently selected agents are input into the mixing network. It's important to note that only information from these selected agents is included in the training process, ensuring that the neural network focuses on the most representative data while maintaining efficiency. Note that the weights are not assigned based on the importance of specific observations to avoid bias in the algorithm.

3) MIXING NETWORK

It is based on QMIX [6] and it satisfies,

$$\arg\max_{\mathbf{a}} Q_{tot}(\boldsymbol{\tau}, \mathbf{a}) = \begin{pmatrix} \arg\max_{a^1} Q_1(\tau^1, a^1) \\ \vdots \\ \arg\max_{a^n} Q_n(\tau^n, a^n) \end{pmatrix}. \tag{9}$$

However, we dynamically select λ agents based on episodes sampled from the experience buffer, and the action-value functions of the selected agents are only used as inputs to the mixing network which is parameterized by ϕ. Let $\Psi = \{i_1, \cdots, i_\lambda\}$ and the vectors of their trajectories and actions are denoted as $\boldsymbol{\tau}^\Psi \triangleq [\tau^{i_1}, \cdots, \tau^{i_\lambda}]$ and $\mathbf{a}^\Psi \triangleq [a^{i_1}, \cdots, a^{i_\lambda}]$, respectively. Then, our mixing network satisfies,

$$\arg\max_{\mathbf{a}^\Psi} Q_{tot}(\boldsymbol{\tau}^\Psi, \mathbf{a}^\Psi) = \begin{pmatrix} \arg\max_{a^{i_1}} Q_{i_1}(\tau^{i_1}, a^{i_1}) \\ \vdots \\ \arg\max_{a^{i_\lambda}} Q_{i_\lambda}(\tau^{i_\lambda}, a^{i_\lambda}) \end{pmatrix}, \tag{10}$$

which represents the partial components of (9). This approach allows each agent i to participate in decentralized execution by simply choosing actions greedily based on its own Q_i. There exists a monotonic relationship between each agent's Q_i and the output of mixing network Q_{tot}, i.e.,

$$\frac{\partial Q_{tot}}{\partial Q_i} \geq 0, \ \forall i \in \Psi. \tag{11}$$

To satisfy this monotonicity, the weights of the mixing network are constrained to be non-negative, allowing hyper-networks to exist independently to generate the weights for the mixing network. If we update both the agent networks and mixing networks for many iterations sufficient to select all agents multiple times to satisfy that their individual argmax operations on own action-value functions yield the same result as the global argmax on $Q_{tot}(\boldsymbol{\tau}, \mathbf{a})$ as in (9).

4) TRAINING

The parameters of the mixing network and agent network of selected agents in Ψ, i.e., ϕ and θ, are updated as following equations,

$$\phi \leftarrow \phi - \beta \nabla_\phi \mathcal{L}(\theta), \tag{12}$$

$$\theta_\Psi \leftarrow \theta_\Psi - \beta \nabla_{\theta_\Psi} \mathcal{L}(\theta), \tag{13}$$

where $\theta_\Psi = [\theta_i, \ \forall i \in \Psi]$, and β is the learning rate. Our proposed algorithm is trained via the iterative minimization of the following loss function,

$$\begin{aligned}
\mathcal{L}(\theta) &= \sum_{i=1}^{\mathbf{b}} (\mathcal{Y} - Q_{tot}(\boldsymbol{\tau}, \mathbf{a}, s; \theta))^2 \\
&= \sum_{i=1}^{\mathbf{b}} \left(\mathcal{Y} - \mathcal{F}(Q_1(\tau^1, a^1), \cdots, Q_N(\tau^N, a^N))\right)^2 \\
&\approx \mathbb{E}_\Psi \left[\sum_{i=1}^{\mathbf{b}} (\mathcal{Y} - Q_{tot}(\boldsymbol{\tau}^\Psi, \mathbf{a}^\Psi, s; \theta))^2 \right] \\
&= \frac{1}{N_s} \sum_{n=1}^{N_s} \sum_{i=1}^{\mathbf{b}} (\mathcal{Y} - Q_{tot}(\boldsymbol{\tau}^{\Psi_n}, \mathbf{a}^{\Psi_n}, s; \phi, \theta_{\Psi_n}))^2 \\
&= \frac{1}{N_s} \sum_{n=1}^{N_s} \sum_{i=1}^{\mathbf{b}} \Big(\mathcal{Y} - \mathcal{F}(Q_{i_1}(\tau^{i_1}, a^{i_1}), \cdots, \\
&\quad \cdots, Q_{i_\lambda}(\tau^{i_\lambda}, a^{i_\lambda}))\Big)^2,
\end{aligned} \tag{14}$$

where $\mathbb{E}_\Psi$ is the expectation with respect to randomly chosen Ψ, N_s is the number of mini-batch, i.e.,

$$\hat{y} = r + \gamma \max_{\mathbf{a}'} Q_{tot}(\boldsymbol{\tau}', \mathbf{a}', s'; \theta^-), \tag{15}$$

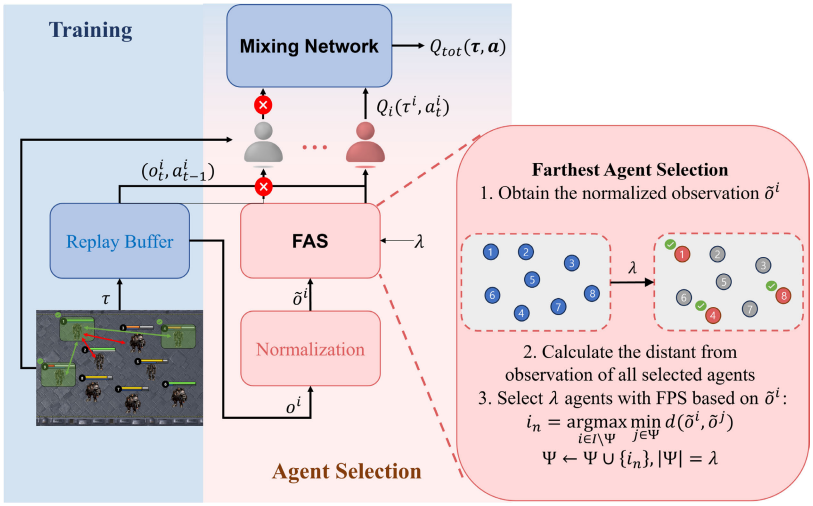


FIGURE 4. FAS-based agent selection in real-time MARL applications such as real-time strategy games.

TABLE 1. Various scenarios in SMAC for experimental performance evaluation.

Scenarios	Agents	Enemies
8m	8 Marines	8 Marines
10m_vs_11m	10 Marines	11 Marines
3s5z	3 Stalkers, 5 Zealots	3 Stalkers, 5 Zealots
MMM	7 Marines, 2 Marauders, 1 Medivac	7 Marines, 2 Marauders, 1 Medivac

and $\Psi_n = \{i_1, \cdots, i_\lambda\}$ is the set of selected agents based on the n-th episode sampled from experience buffer.

V. EXPERIMENTS

The performance of the proposed algorithm is evaluated on two widely used partially observable multi-agent cooperative tasks, i.e., SMAC [33] and the Predator Prey (PP) [36]. The training process of the proposed FAS method is compared with that of the full algorithm and other baseline approaches through a comprehensive performance analysis. We utilize the two FAS-applied methods, dynamic FAS (DFAS) and static FAS (SFAS). DFAS refers to experiments where λ is adjusted for each episode batch based on the corresponding variance values, while SFAS denotes experiments conducted with a fixed value of λ. Additionally, the impact of FAS is assessed both with a fixed number of agents and by dynamically determining the number of agents to be selected for each scenario. This approach provides insights into the adaptability and efficiency of FAS in varying multi-agent environments.

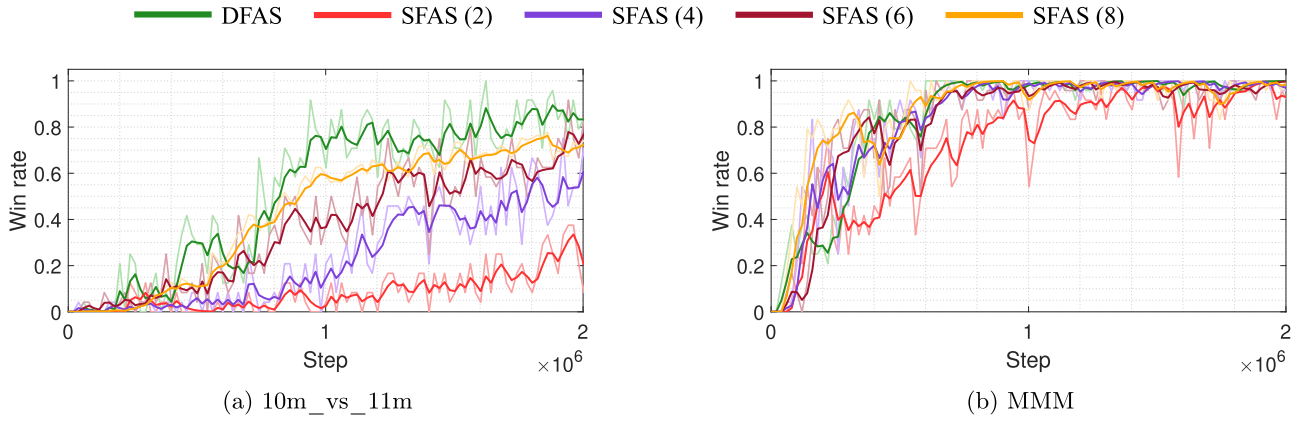
When determining λ, we focused on the variance of each component-wise observation among the agents. Since the observations are normalized between 0 and 1, where a higher average variance among agents' partial observations indicates greater dissimilarity. Therefore, more agents need to be included as inputs into the mixing network. Conversely, a lower average variance suggests similarity in the agents' partial observations, implying that overlapping observations need not be redundantly input into the mixing network. In DFAS experiments, the variance for each observation component is calculated for every episode, with the average variance being updated continuously after each episode. The value λ is then determined based on the ratio of the current observation variance to the cumulative average, relative to the total number of agents. This approach ensures that λ indicates the number of agents to be included in batch learning, adapting to the dynamic nature of the observations across episodes. Additionally, to prevent DFAS from selecting all agents, a limit is set to choose up to a maximum of 80% of the total agents (i.e., up to 6 agents in scenarios with 8 agents, and up to 4 agents in scenarios with 5 agents).

The research was conducted using Python 3.7.13, PyTorch 1.12.1, StarCraft II version 4.10, and SMAC version 1.0.0.

A. STARCRAFT MULTI-AGENT CHALLENGE (SMAC)

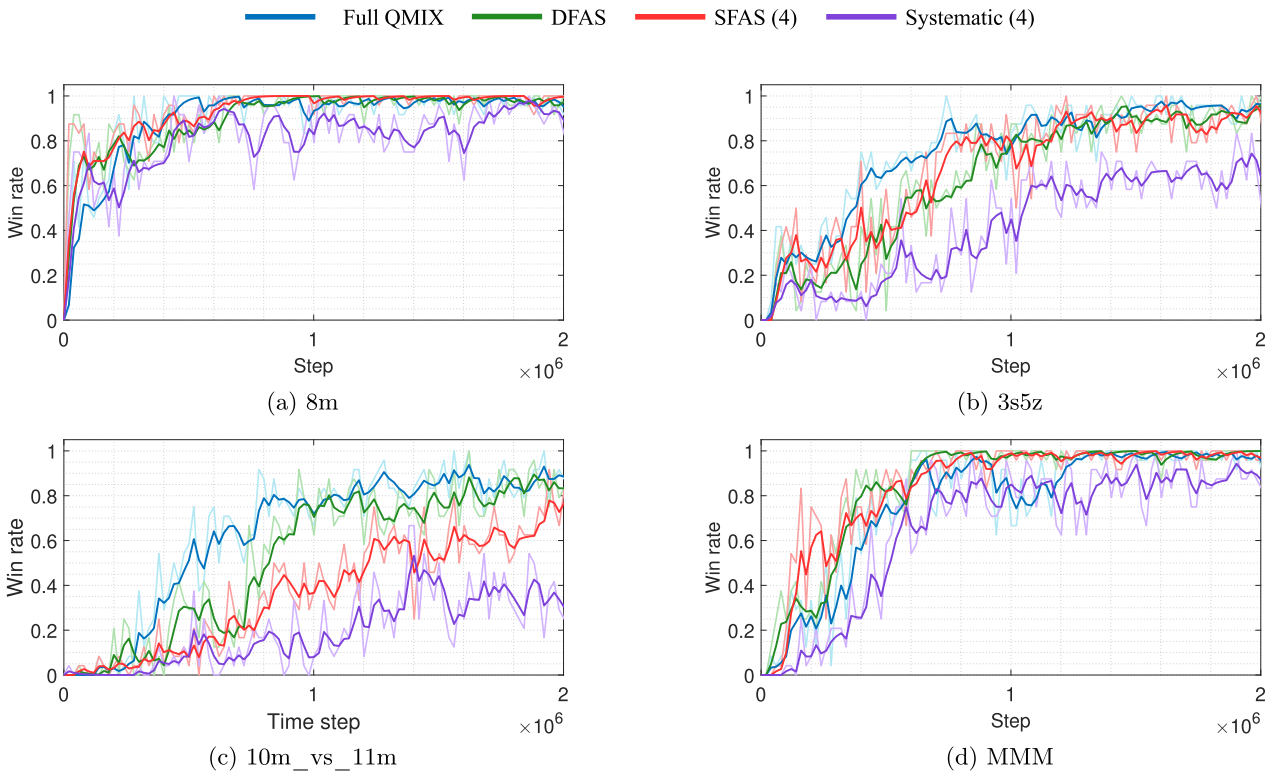
In SMAC, a team consists of a number of agents and has the objective of defeating an opponent team, which is governed by the StarCraft built-in AI algorithm. All of the experiments are evaluated in the level/difficulty of 7 (i.e., very hard). We selected environments evaluation under various conditions: symmetric-homogeneous agents (8m), asymmetric-homogeneous agents (10m_vs_11m), symmetric-2 heterogeneous agents (3s5z), and symmetric-3 heterogeneous agents (MMM). The details of the composition of agents for each environment are presented in Table 1.

When setting a static λ, the value of λ significantly impacts the performance. Fig. 5 illustrates the results of selecting



(a) 10m_vs_11m (b) MMM

FIGURE 5. Performance comparisons on different λ values.



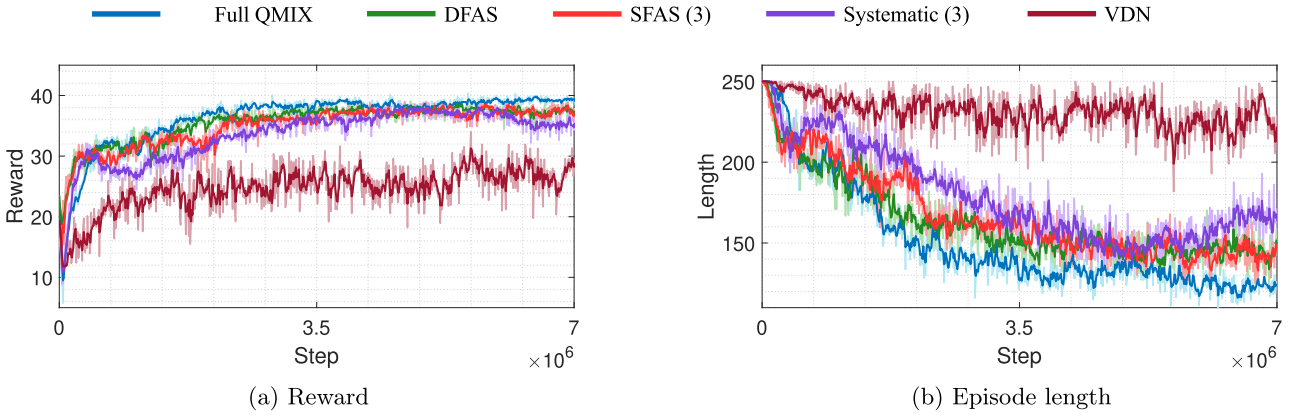
(a) 8m (b) 3s5z

(c) 10m_vs_11m (d) MMM

FIGURE 6. Performance comparisons on various SMAC environments.

agents through FAS with varying fixed λ values. Although there is a clear tendency for performance to improve with more agents, it is crucial to choose an appropriate λ considering the trade-off between computational load and performance. In DFAS experiments in each scenario, the average λ determined after training is obtained. In the 10m_vs_11m scenario, an average of 5.45 agents were selected, whereas in the MMM scenario, the number increased to an average of 7.02 agents. Additionally, DFAS shows the best performance in Fig. 5-(a), even if fewer agents are selected on average. It has been experimentally observed that in scenarios with homogeneous agents, such as 10m_vs_11m, the smaller difference in partial observations among agents leads to lower variance, which affects the value of λ.

Fig. 6 describes the test win rate on various SMAC environments. Additionally, to account for cases where the full QMIX algorithm can not be used because of the computational constraint, we have denoted the experiments that involve inputting only agents with systematic sampling into the mixing network as "Systematic." In both the SFAS and systematic sampling experiments, we only selected



(a) Reward (b) Episode length

FIGURE 7. Performance comparisons on predator prey.

four agents. The proposed DFAS demonstrated performance nearly equivalent to that of the Full QMIX, and the SFAS, which selects fewer agents than DFAS, also showed its potential. In the 8m and 3s5z scenarios, DFAS resulted in the selection of an average of 5.01 and 5.56 agents, respectively. These results also statistically confirm that more agents need to be selected in heterogeneous agent scenarios like 3s5z, where there is greater variance in the agents' observations.

B. PREDATOR PREY

In PP, $\mathbf{N}$ predators chase and surround to try to capture $\mathbf{M}$ preys. When an agent reaches a goal, it receives a reward of 2.0. If all prey are hunted, the agent receives an additional reward of 10.0. In cases where all goals are achieved, the agent receives an additional reward based on the length of the episode. The baselines include the Full QMIX and VDN. In our experiments, we set $\mathbf{N} = 5$, $\mathbf{M} = 15$ in 20×20 grid and required that all prey be hunted within a maximum of 250 steps to proceed to the next episode. In the SFAS scenario of the PP experiment, only 3 out of the 5 selected Predators are utilized for training. As shown in Fig. 7, the proposed DFAS and SFAS show acceptable and relatively superior performance relative to the Full QMIX, although it only includes less than 80% of the whole agents. In the DFAS scenario, an average of 3.7 agents out of the total 5 Predators were used during the training process.

VI. CONCLUDING REMARKS

This paper introduces a novel control method for the multi-agent reinforcement learning framework that leverages only a subset of agents instead of inputting all agents into the centralized neural network. By normalizing each agent's observations, we choose a representative subset of agents from the population. The cooperation among these selected agents is operated in the mixing network with a classical QMIX approach. In order to determine the number of agents to be selected, the variance among the observation of agents is considered according to the farthest agent selection (FAS) approach inspired by FPS. Our data-intensive performance evaluations with real-time strategy game platforms reveal that the proposed FAS algorithm can attain results comparable to those of the conventional method, where observations from all agents are utilized for updating the mixing network in QMIX. This is achieved while allowing a fewer number of agents to participate in each epoch, demonstrating the scheme's efficiency in balancing performance with reduced agent involvement. As future research directions, various applications of our proposed FAS algorithm can be considerable such as multi-drone cooperative mobile Internet access and autonomous surveillance applications. Furthermore, this approach can provide motivation for artificial intelligence research that requires the training with the selected agents from the diverse pool of entire agents.

ACKNOWLEDGMENT

The authors would like to thank Joongheon Kim for his advice and contribution to research and discussions.

REFERENCES

[1] H. Jin, Y. Wei, Z. Yang, Z. Liu, and G. Fan, "Multi-intersection management for connected autonomous vehicles by reinforcement learning," in *Proc. IEEE 43rd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Hong Kong, Jul. 2023, pp. 649–659.

[2] Y. Xiao, Y. Song, and J. Liu, "Collaborative multi-agent deep reinforcement learning for energy-efficient resource allocation in heterogeneous mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 6, pp. 6653–6668, Jun. 2024.

[3] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning-based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 729–743, Feb. 2020.

[4] J. Tan, R. Khalili, H. Karl, and A. Hecker, "Multi-agent distributed reinforcement learning for making decentralized offloading decisions," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, London, U.K., May 2022, pp. 2098–2107.

[5] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, 2nd Quart., 2021.

[6] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, Stockholm, Sweden, Jul. 2018, pp. 4292–4301.

[7] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, vol. 97, Long Beach, CA, USA, Jun. 2019, pp. 5887–5896.

[8] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 32, New Orleans, LA, USA, Feb. 2018, pp. 2974–2982.

[9] M. Shin, D.-H. Choi, and J. Kim, "Cooperative management for PV/ESS-enabled electric vehicle charging stations: A multiagent deep reinforcement learning approach," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3493–3503, May 2020.

[10] W. Du, S. Ding, C. Zhang, and Z. Shi, "Multiagent reinforcement learning with heterogeneous graph attention network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 6851–6860, Oct. 2023.

[11] Y. Zhong, J. G. Kuba, X. Feng, S. Hu, J. Ji, and Y. Yang, "Heterogeneous-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 25, no. 1, pp. 1–67, Jan. 2024.

[12] R. R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella, "Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings," in *Proc. Int. Joint Conf. Artif. Intell.*, Acapulco, Mexico, Aug. 2003, pp. 705–711.

[13] W. J. Yun, D. Mohaisen, S. Jung, J.-K. Kim, and J. Kim, "Hierarchical reinforcement learning using Gaussian random trajectory generation in autonomous furniture assembly," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.* New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 3624–3633.

[14] W. J. Yun, S. Park, J. Kim, M. Shin, S. Jung, D. A. Mohaisen, and J.-H. Kim, "Cooperative multiagent deep reinforcement learning for reliable surveillance via autonomous multi-UAV control," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7086–7096, Oct. 2022.

[15] L. Deng, W. Gong, M. Liwang, L. Li, B. Zhang, and C. Li, "Towards intelligent mobile crowdsensing with task state information sharing over edge-assisted UAV networks," *IEEE Trans. Veh. Technol.*, early access, Feb. 23, 2024, doi: 10.1109/TVT.2024.3369089.

[16] L. Miuccio, S. Riolo, S. Samarakoon, M. Bennis, and D. Panno, "On learning generalized wireless MAC communication protocols via a feasible multi-agent reinforcement learning framework," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 298–317, Feb. 2024.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 30, Long Beach, CA, USA, Dec. 2017, pp. 6000–6010.

[18] T. Phan, F. Ritz, P. Altmann, M. Zorn, J. Nüßlein, M. Kölle, T. Gabor, and C. Linnhoff-Popien, "Attention-based recurrence for multi-agent reinforcement learning under stochastic partial observability," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Honolulu, HI, USA, Jul. 2023, pp. 27840–27853.

[19] H. Fei, Y. Zhang, Y. Ren, and D. Ji, "Optimizing attention for sequence modeling via reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3612–3621, Aug. 2022.

[20] H. Liu, Y. Liu, X. Wang, and H. Yang, "Exploring coarse-grained pre-guided attention to assist fine-grained attention reinforcement learning agents," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Padua, Italy, Jul. 2022, pp. 1–7.

[21] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Mach. Learn.*, Amherst, MA, USA, Jul. 1993, pp. 330–337.

[22] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, vol. 29, Barcelona, Spain, Dec. 2016, pp. 2244–2252.

[23] T. Wang, T. Gupta, A. Mahajan, B. Peng, S. Whiteson, and C. Zhang, "RODE: Learning roles to decompose multi-agent tasks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2021, pp. 1–24.

[24] M. Rosynski, A. Pop, and L. Buşoniu, "Active search and coverage using point-cloud reinforcement learning," in *Proc. 27th Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Timisoara, Romania, Oct. 2023, pp. 289–296.

[25] C. Park, G. S. Kim, S. Park, S. Jung, and J. Kim, "Multi-agent reinforcement learning for cooperative air transportation services in city-wide autonomous urban air mobility," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 8, pp. 4016–4030, Aug. 2023.

[26] C. Park, S. Park, G. S. Kim, S. Jung, J.-H. Kim, and J. Kim, "Multi-agent deep reinforcement learning for efficient passenger delivery in urban air mobility," in *Proc. IEEE Int. Conf. Commun.*, Rome, Italy, May 2023, pp. 5689–5694.

[27] W. J. Yun, S. Jung, J. Kim, and J.-H. Kim, "Distributed deep reinforcement learning for autonomous aerial eVTOL mobility in drone taxi applications," *ICT Exp.*, vol. 7, no. 1, pp. 1–4, Mar. 2021.

[28] C. Zhang and V. Lesser, "Coordinated multi-agent reinforcement learning in networked distributed POMDPs," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 25, San Francisco, CA, USA, Aug. 2011, pp. 764–770.

[29] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 5588–5597.

[30] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 5105–5114.

[31] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. ACM Int. Conf. Auto. Agents MultiAgent Syst. (AAMAS)*, Stockholm, Sweden, Jul. 2018, pp. 2085–2087.

[32] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Berlin, Germany: Springer, Jul. 2015.

[33] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The StarCraft multi-agent challenge," *CoRR*, vol. abs/1902.0404, pp. 2186–2188, Feb. 2019.

[34] W. J. Yun, S. Yi, and J. Kim, "Multi-agent deep reinforcement learning using attentive graph neural architectures for real-time strategy games," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Melbourne, VIC, Australia, Oct. 2021, pp. 2967–2972.

[35] S. G. K. Patro and K. K. Sahu, "Normalization: A preprocessing stage," 2015, *arXiv:1503.06462*.

[36] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Auto. Robots*, vol. 8, pp. 345–383, Jun. 2000.

HYUNSOO LEE received the B.S. degree from the School of Electronic Engineering, Soongsil University, Seoul, Republic of Korea, in 2021. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Korea University, Seoul.

His research interests include deep learning algorithms and their applications to mobility and networking. He was a recipient of the IEEE Vehicular Technology Society (VTS) Seoul Chapter Award in 2022.

GYU SEON KIM received the B.S. degree in aerospace engineering from Inha University, Incheon, Republic of Korea. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Korea University, Seoul, Republic of Korea.

His research interests include deep reinforcement learning algorithms and their applications to autonomous mobility systems. He received the IEEE Seoul Section Student Paper Contest Award (2023).

MINSEOK CHOI received the B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2011, 2013, and 2018, respectively. He was a Visiting Postdoctoral Researcher of electrical and computer engineering with the University of Southern California (USC), Los Angeles, CA, USA, and a Research Professor of electrical engineering with Korea University, Seoul, South Korea. He was an Assistant Professor with Jeju National University, Jeju, South Korea, from 2020 to 2022. He has been with Kyung Hee University, Yongin, South Korea, since 2022, and he is currently an Assistant Professor.

HANKYUL BAEK received the B.S. and Ph.D. degrees in electrical and computer engineering from Korea University, Seoul, Republic of Korea, in 2020 and 2024, respectively.

He was with LG Electronics, Seoul, from 2020 to 2021. He was a Visiting Scholar with the Department of Electrical and Computer Engineering, The University of Utah, Salt Lake City, UT, USA, in 2023. He has been a Postdoctoral Scholar with the Department of Electrical and Computer Engineering, Korea University, since March 2024. His current research interests include quantum machine learning and its applications to mobility and networking. He received the IEEE Seoul Section Student Paper Contest Award (2023).

SOOHYUN PARK (Member, IEEE) received the B.S. degree in computer science and engineering from Chung-Ang University, Seoul, Republic of Korea, in February 2019, and the Ph.D. degree in electrical and computer engineering from Korea University, Seoul, in August 2023.

She was a Postdoctoral Scholar with the Department of Electrical and Computer Engineering, Korea University, from September 2023 to February 2024. She has been an Assistant Professor with Sookmyung Women's University, Seoul, since March 2024. Her research interests include deep learning theory and network/mobility applications, quantum neural network (QNN) theory and applications, QNN software engineering and programming languages, and AI-based autonomous control for distributed computing systems. She was a recipient of the ICT Express Best Reviewer Award (2021), the IEEE Seoul Section Student Paper Contest Award, and the IEEE Vehicular Technology Society (VTS) Seoul Chapter Award.