2023 The 23rd International Conference on Control, Automation and Systems (ICCAS 2023)
Yeosu Venezia Hotel & Utop Marina Hotel, Yeosu, Korea, Oct. 17∼20, 2023

Attention-Based Randomized Ensemble Multi-Agent Q-Learning

Jeongho Park, Obin Kwon and Songhwai Oh*

Department of Electrical and Computer Engineering and ASRI, Seoul National University,
Seoul, 08826, Korea (jeongho.park@rllab.snu.ac.kr, obin.kwon@rllab.snu.ac.kr, songhwai@snu.ac.kr)
* Corresponding author

Abstract: Cooperative multi-agent scenarios are prevalent in real-world applications. Optimal coordination of agents requires appropriate task allocation, considering each task's complexity and each agent's capability. This becomes challenging under decentralization and partial observability, as agents must self-allocate tasks using limited state information. We introduce a novel multi-agent environment in which effective sub-task assignment is crucial for high-scoring performance. In addition, we propose a new multi-agent reinforcement learning framework named as attention-based randomized ensemble multi-agent Q-learning, or AREQ for short. This approach integrates a unique network structure using a multi-head attention mechanism, efficiently extracting task-related information from observations. AREQ also incorporates a randomized ensemble method, enhancing sample efficiency. We explore the impact of this attention-based structure and the random ensemble method through an ablation study and show AREQ's superiority compared to existing MARL methods within our proposed environment.

Keywords: Multi-agent reinforcement learning, Attention, Ensemble network

1. INTRODUCTION

Cooperative multi-agent problems are common in real-world applications, such as factory, warehouse and surveillance systems. In these settings, agents are expected to enhance overall system efficiency rather than pursuing individual benefits. For better cooperation, it's essential to accurately allocate sub-tasks to these agents, taking into account the complexity of each task and the capacity of each agent. However, real-world applicability often necessitates the constraints of decentralization and partial observability, which substantially increase problem complexity.

Multi-agent reinforcement learning (MARL) has recently drawn much attention due to its ability of understanding complicated multi-agent systems and providing efficient decentralized policies that can operate in partially observable environments. Several multi-agent environments, such as Starcraft2 Challenges [1], were introduced to evaluate MARL algorithms, but the concept of task assignment is implicitly embedded in those environments, thus making it hard to verify if the tasks are being properly allocated.

To acquire better insight about multi-agent cooperation, we design a new multi-agent environment, named as goal assignment and pathfinding problem, where effective sub-task assignment is crucial for high-scoring performance. At the same time, agents have to navigate a highly-structured map to complete each of the task. Fig. 1 shows an example of the map, composed of multiple one-way corridors and frequently appearing corners. The objective of the agents is to collect all the goals in minimal time steps while avoiding any collision.

To solve the goal assignment and pathfinding problem, we propose a new MARL framework combined with a specialized network structure, named as attention-based randomized ensemble multi-agent Q-learning (AREQ). Here, we introduce a novel feature embedding network structure based on the multi-head attention mechanism, which can effectively extract the task-related information from the the observations. Also, we introduce a new MARL framework that employs the randomized ensemble method [2], which uses the output features of the multi-head attention networks as an Q-function ensemble. Randomized ensemble method of AREQ improves the robustness to increasing update-to-data (UTD) ratio, which is the number of updates taken by the agent compared to the number of actual interactions with the environment. With high UTD ratio setting, AREQ can actively utilize the costly state information and reach the top performance in a sample-efficient way.

To assess the performance of AREQ, we first conduct the ablation study in the proposed goal assignment and pathfinding task environment to investigate the influence of each element that composes AREQ. Also, we compare AREQ with other MARL algorithms on the high UTD setting. Our results show that AREQ is well-designed to solve the multi-agent goal assignment and pathfinding task with a significant performance gain.

2. RELATED WORK

MARL algorithms have witnessed much attention as well as significant advances in recent years. MADDPG [3], COMA [4], and DOP [5] try to solve the multi-agent problems in policy-based approach. They maintain a centralized critic which estimates the value, while utilizing decentralized actors that only takes individual observations. Value-based methods, such as VDN [6], QMIX [7], QPLEX [8], Weighted-QMIX [9], and ResQ [10] factorize the joint action-value into decentralized individual

This work was supported by Korea Evaluation Institute of Industrial Technology(KEIT) grant funded by the Korea government(MOTIE) (No.20024368, Development of Integrated Control System for Multi-mobility and Autonomous Delivery Service Demonstration)

action-value functions under *Individual-Global-Max* [11] (IGM) condition.

Reinforcement learning has been actively used to study general multi-agent task assignment problems. [12] proposes a structured approach to multi-agent coordination where the assignment is obtained by solving a centralized optimization problem whose objective function is parameterized by a learned scoring model. HiT-MAC [13] introduces a hierarchical multi-agent coordinator which decomposes the target coverage problem into two-level tasks: targets assignment by a coordinator and tracking assigned targets by executors. Both of the algorithms maintain a centralized coordinator which is different from our fully decentralized approach.

3. BACKGROUND

3.1 Decentralized Partially Observable MDP

We consider a fully cooperative multi-agent task that is modeled by a decentralized partially observable Markov Decision Process (Dec-POMDP) [14]. It is defined by a tuple $\mathcal{M} =< \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \nabla, \gamma, \mathcal{P}, \otimes >$, where $\mathcal{N}$ is a finite set of n agents, A is a finite action set, and $s \in \mathcal{S}$ is a finite set of global states. At each time step, each agent $i \in \mathcal{N}$ chooses an action $a_i \in \mathcal{A}$, which forms a joint action vector $\boldsymbol{a}$. It results in a joint reward $r(s, \boldsymbol{a})$ and the environment then transitions to a new state s' based on the transition function $P(s'|s, \boldsymbol{a})$. $\gamma \in [0, 1)$ is a discount factor. Each agent receives an individual partial observation $o_i \in \Omega$ according to some observation function $O(s, i)$. Also, each agent i has its own action-observation history $\tau_i \in \mathcal{T} \equiv (\otimes \times \mathcal{A})^*$, and construct its own stochastic policy $\pi_i(a_i|\tau_i)$ based on it.

3.2 Centralized Training Decentralized Execution

Centralized training decentralized execution (CTDE) is a popular paradigm in MARL [3, 4, 6–10], that relaxes the partial observability condition during the centralized training and allows using extra state information. One popular CTDE approach is *individual-global-max* [11] (IGM) which suggests a condition that guarantees the local greedy action selections in the individual action-value functions to be consistent with the global joint actions selected from the joint action-value functions.

QMIX [7] employs a *mixer* network that estimates joint action-values through a nonlinear monotonic combination of individual agents' values, effectively satisfying IGM. Within the CTDE framework, QMIX's *mixer* network leverages extra state information, producing joint action-value as a function of global state information and agents' individual action-values,

$$Q_{tot}(\boldsymbol{\tau}, \boldsymbol{a}, s) = \mathit{mixer}(s, Q_1(\tau_1, a_1), \ldots, Q_n(\tau_n, a_n)), \tag{1}$$

where Q_{tot} denotes the joint action-value function, Q_i is the i-th agent's action-value function, and $\boldsymbol{\tau}$ denotes the joint action-observation history of the agents. Throughout the paper, we omit s in Q_{tot} and *mixer*'s arguments for presentational simplicity. QMIX is then trained end-to-end to minimize the following loss between the online values and target values:

$$\mathcal{L} = \sum_{b=1}^{B} \left(y_b - Q_{tot}(\boldsymbol{\tau}_b, \boldsymbol{a}_b)\right)^2, \tag{2}$$

where B is the batch size, and y is the target value computed as $y = r + \gamma \max_{\boldsymbol{a}'} Q^-_{tot}(\boldsymbol{\tau}', \boldsymbol{a}')$. Q^- symbol denotes the target Q-function whose parameters are periodically copied from the original Q-function [15].

3.3 Randomized Ensemble Method for Q-learning

Q-learning is known to have the overestimation bias issue which can seriously harm its performance [16], especially when training with a large UTD ratio. Randomized ensemble double Q-learning (REDQ) [2] adds an ensemble of Q-functions on soft-actor-critic [17] framework to address this issue. REDQ maintains the ensemble of N Q-functions and conducts in-target minimization across a random subset of Q functions from the ensemble to compute the target value y for updating the critic

$$y = r + \gamma \left(\min_{m \in \mathcal{M}} Q^{m-}(s', \tilde{a}') - \alpha \log \pi_\phi(\tilde{a}'|s') \right), \quad \tilde{a}' \sim \pi_\phi(\cdot|s') \tag{3}$$

where $\mathcal{M}$ is a randomly sampled subset of the ensemble Q-functions with fixed size M. Q^m and Q^{m-} denotes the m-th Q-function among the ensemble and its corresponding target Q-function, respectively, while π denotes the policy and ϕ refers to its parameters. By carefully adjusting N and M, REDQ can control the average Q-function bias, and improve the performance in high UTD settings.

4. PROBLEM FORMULATION

4.1 Environment Setup

Our environment consists of a finite-sized, two-dimensional, four-connected grid world. The grid cells can contain agents, goals, and obstacles. Goals disappear from the map once an agent reaches them. Agents can move to any of the four neighboring grid cells or remain stationary each step. Collisions occurs if agents move to a cell occupied by obstacles or if multiple agents move to the same cell, causing the involved agents to remain in their current positions. At every episode's start, n agents and g goal points are randomly spawned on a randomly generated map. We adopt the map generating algorithm introduced in [18]. Agents share the objective to collect all existing goal points in minimal time steps, while avoiding collisions with obstacles and other agents.

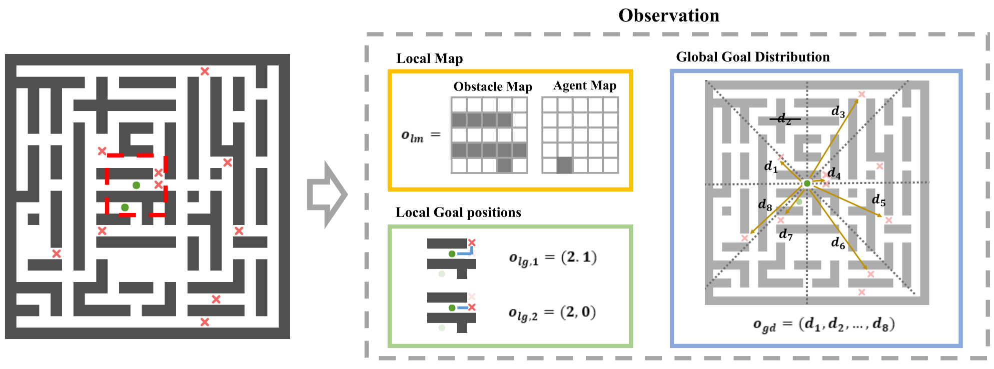


Fig. 1.: An example of the highly-structured grid map environment and its observation space of the agents (here, for the agent at the center of the map) when FOV is (5×5). Three different kinds of observations are provided. First, the local map o_{lm} that depicts the locations of the obstacles and agents inside the FOV. Second, o_{lg}'s which are the position coordinates of the nearby goals inside the FOV. Last, the o_{gd} which is a vector composed of the euclidean distance to the closest goal in each sector. Note that if there are no goals in a sector (in this case, the sector corresponding to d_2, -1 is observed instead.

4.2 Reward Formulation

The environment only provide task-related shared reward values to the agents without any information about individual agents' credits. We penalize them with a negative reward ($r_t = -1.0$) every time step, and as soon as all existing goal points are collected, the episode is terminated. A success reward is given whenever an agent collects a goal point ($r_t = +5.0$). We penalize the agents whenever they collide with the obstacles or other agents ($r_t = -1.0$). At each time step, all types of rewards are summed up and provided equally to all agents.

4.3 Observation Space

In our partially observable environment, agents receive local observations and limited global information. Local observations provide agents with the relative positions of obstacles, goals, and other agents within a square field-of-view (FOV) of 11×11 grid cells. This information is conveyed through a two-channel spatial map for obstacles and other agents, and coordinate vectors for goals. Beyond local data within the FOV, we also provide restricted information on the global distribution of uncollected goals. The entire map is divided into eight 45-degree sectors, centered on each agent as illustrated in Fig. 1. For each sector, we compute the Euclidean distance to the nearest goal, creating an eight-size goal distribution observation vector o_{dist}.

4.4 State Information for Centralized Training

During centralized training, we provide extra task-related state information to *mixer*. We compute each agent's geodesic distance to each goal point using the A^* algorithm, gathering this into a geodesic distance state vector $s_{geodesic} \in \mathbb{R}^{ng}$. However, with increasing agents, goals, and map size, these computations can be time-consuming. We also generate each agent's three-channel local map, stacking them into a local map state vector $s_{map} \in \mathbb{R}^{n \times 11 \times 11 \times 3}$ for an 11-sized FOV. This map incorporates obstacle, agent, and goal point channels. Together with $s_{geodesic}$, we formulate the state information s for centralized training: $s = [s_{geodesic}, s_{map}]$, where $[\cdot, \cdot]$ denotes concatenation.

5. METHOD

In this section, we introduce AREQ, a new MARL framework designed for multi-agent goal assignment and pathfinding. First, we introduce a unique network structure employing a multi-head attention mechanism that efficiently transforms given observations into multiple feature embedding vectors. Next, we propose a novel MARL framework, built on a randomized ensemble method that mitigates overestimation bias in high UTD settings. This framework utilizes feature embedding vectors from the multi-head attention module to construct a Q-function ensemble.

5.1 Multi-Head Attention-Based Network Structure

At each time step, the agent should decode the complex relationships between a varying number of local goal points and other agents while considering the observed geographical information. Furthermore, it should decide whether to focus locally or explore farther regions.

To address this issue, we introduce a novel multi-head attention mechanism-based network structure for the goal feature embedding module within each agent's individual value function, Q_i, which is illustrated in Fig. 2a. This module first converts different observation types into equal-sized embedding vectors using fully-connected and CNN layers. Given the count of local goal points, l_t^i, for the i-th agent at time step t, we obtain $l_t^i + 2$ embedding vectors: one local map embedding e_{lm}, one global goal distribution embedding e_{gd}, and l_t^i local goal position embedding vectors $e_{lg,i}$, where $i = 1, \ldots l_t^i$. l_t^i can vary per time step depending on the number of goal points within the agent's FOV.

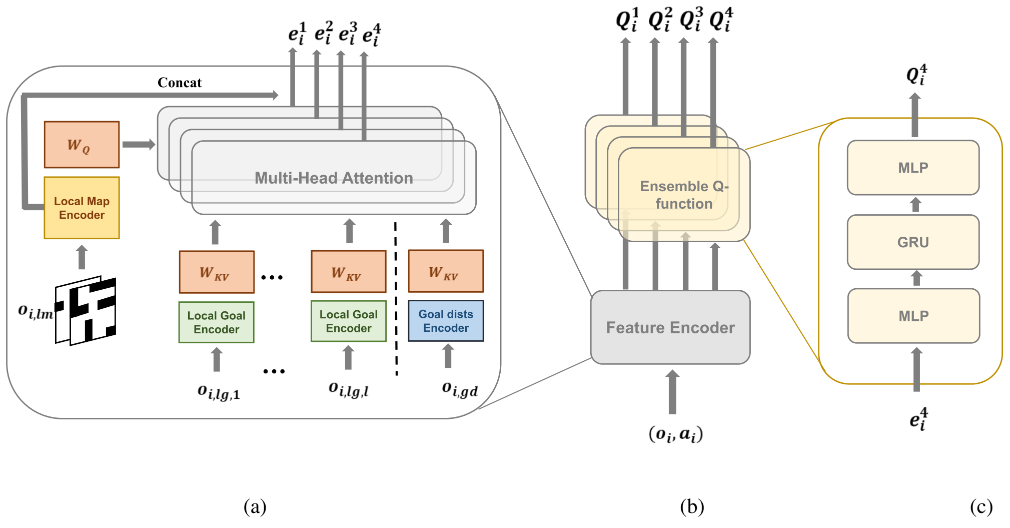


Fig. 2.: Schematic of the Q-network for i-th agent. (a) The 4-headed attention network structure that encodes the given observations. $o_{i,lg,p}$ denotes the position information of l-th local goal from the observation of i-th agent. (b) The overall structure of i-th agent's Q-function that outputs different Q values as much as the number of the heads. (c) The network structure inside each of the ensemble.

Given the observation, agents should determine which goal-related information to prioritize based on the current local distribution of obstacles and other agents. To reflect this, we calculate the query vector V_Q from the local map embedding, while both local goal position embeddings and global goal distribution embeddings are transformed into key vectors V_K and value vectors V_V. After performing scaled dot-product attention on the tokens, we attain the aggregated goal embedding e_{goal} through

$$\{e^1_{goal}, \dots e^N_{goal}\} = MHA(e_{lm}, e_{lg}, e_{gd}), \tag{4}$$

where *MHA* refers to the multi-head attention mechanism function, and N denotes the number of heads. We then concatenate e_{lm} with each e^h_{goal} to form the total embedding e, integrating information from both the local map and goal embeddings. These multi-head embeddings works as the ensemble of Q-functions for each agent. Note that, when there are no local goal points around, the global goal distribution embedding naturally becomes the goal embedding.

5.2 Randomized Ensemble for MARL

While additional refined state information for centralized training can improve the performance, it takes extra computing time to acquire. However, naively increasing UTD ratio is known to aggravate overestimation bias issue [2] in single-agent reinforcement learning. We show that increasing UTD ratio harms the performance of MARL as well through our experiments and introduce a new learning framework of AREQ that employs a randomized ensemble approach to QMIX, thereby improving robustness against UTD ratio.

We use three key hyper-parameters, G, N, and M, where G denotes UTD ratio, N denotes the size of the ensemble, and M denotes the size of the randomly sampled subset. The i-th agent has an ensemble of N Q-functions Q^k_i as well as their corresponding N target Q-functions Q^{k-}_i, where $k = 1, \dots, N$. Note that AREQ only maintains one *mixer* network.

Following the framework of QMIX, AREQ also restricts the mixer network to satisfy monotonicity condition. Therefore, the target value of AREQ is obtained by passing individual maximal action-values from n agents into the monotonic mixer network.

$$\begin{aligned} y &= r + \gamma \max_{\boldsymbol{a'}} Q^-_{tot}(\boldsymbol{\tau'}, \boldsymbol{a'}) \\ &= r + \gamma mixer(Q^-_{1,max}(\tau'_1), \dots, Q^-_{n,max}(\tau'_n)), \end{aligned} \tag{5}$$

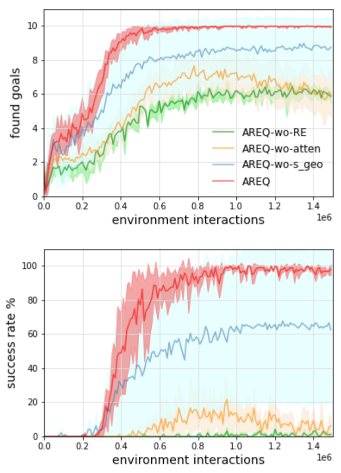
where we represent the individual maximal action-values on the next state as $Q^-_{i,max}(\tau'_i)$. QMIX simply conducts the maximization over the target Q-function for each agent, so it can be written as $Q^-_{i,max}(\tau'_i) = \max_{a'_i} Q^-_i(\tau'_i, a'_i)$. In contrast, AREQ conducts the maximization after the in-target minimization across random subsets of target Q-functions from the ensemble.

$$Q^-_{i,max}(\tau'_i) = \max_{a'_i} \min_{m_i \in \mathcal{M}_i} Q^{m_i -}_i(\tau_i, a'_i), \quad i = 1, \dots, n \tag{6}$$

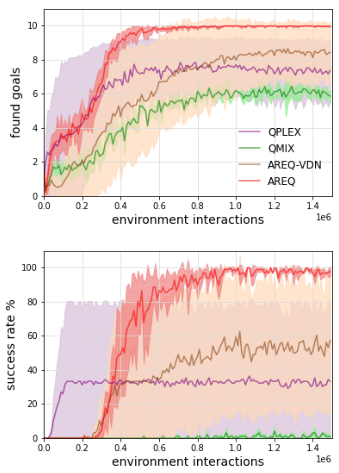
where $\mathcal{M}_i$ denotes the randomly sampled subset of ensemble for the i-th agent. Now, the target value y is computed by combining (5) and (6), and AREQ is trained to minimize the following loss:

$$\mathcal{L} = \sum_{k=1}^{N} \sum_{b=1}^{B} \left(y_b - Q^k_{tot} \boldsymbol{\tau}_b, \boldsymbol{a}_b\right)^2, \tag{7}$$

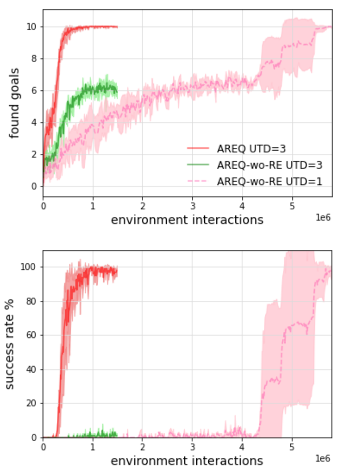
where B is the batch size sampled from the replay buffer. $Q^k_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) = mixer(Q^1_1(\tau_1, a_1), \dots, Q^1_n(\tau_n, a_n))$ refers to the online joint action-value computed from the agents' k-th Q-functions among the ensembles. Note that there is only one target value for each batch ob-



(a)



(b)



(c)

Fig. 3.: (a) Ablation study on AREQ. (b) Comparison of performance between agents trained in different UTD settings. (c) Comparison of performance between AREQ and other MARL methods.

tained through the in-target minimization across the random subsets, and it is shared across the ensembles. In practice, we stack a series of fully-connected layers and a GRU [19] layer on top of each attention head to construct ensembles of Q-functions as depicted in Fig.2c.

6. EXPERIMENTS

In this section, we evaluate AREQ's performance in the proposed multi-agent goal assignment and pathfinding environment. Episodes are terminated after 50 time steps. The evaluation is conducted every 10,000 environment interactions, with 32 unseen test episodes each. We evaluate the methods mainly on two metrics, the number of found goals during each episode and the success rate. The success rate depicts whether the agents end the episode by collecting all the goals. All experiments are conducted in UTD=3 setting and trained for 1.5M environment interaction with 3 different seeds. As an exception, the ablation experiments conducted in UTD=1 are trained for 6M environment interactions.

6.1 Ablation Study

We perform ablation study on AREQ in order to investigate the influence of each element that composes AREQ. Fig.3a plots the averaged number of the found goals and the success rate for each experiment.

First, we investigate if the attention-based network structure improves the performance. Specifically, we set a baseline, named as AREQ-*wo-atten*, that uses the network structure based on fully-connected layers in the feature embedding module. Fig. 3a shows that the attention-based network structure improves the averaged performance of AREQ. While AREQ-*wo-atten* fails to solve the task with very low success rate, AREQ stably collects all 10 goals and keeps high success rate near 100%.

Second, we test if $s_{geodesic}$ is worth the time it takes to compute. We design a variant of AREQ with a mixer that only takes s_{map} as extra information, and call it as AREQ-*wo-s-geo*. Fig. 3a shows that though AREQ-*wo-s-geo* achieves similar performance as AREQ in specific seeds, suffers from large variance of performance in general. This indicates that $s_{geodesic}$ helps the agents to find the solution in more easy and stable way.

Last, we investigate if the randomized ensemble method improves the robustness to high UTD ratio. Fig. 3b plots the performance comparison between AREQ in UTD=3, and AREQ without randomized ensemble method in UTD=3 and UTD=1, each named as AREQ-*wo-RE UTD=3* and AREQ-*wo-RE UTD=1*. AREQ-*wo-RE UTD=1* takes about 6M environment interactions to achieve the top performance, while AREQ with UTD=3 takes only 1M environment interactions. The fact that AREQ-*wo-RE UTD=3* does not shows equal performance with AREQ UTD=3 indicates that randomized ensemble method improves the robustness to high UTD settings.

6.2 Comparison with MARL Algorithms

We compare AREQ with other MARL algorithms with UTD=3 setting. QPLEX [8] implements the full IGM class by employing a duplex dueling network architecture, and is known to show good performance in Starcraft2 challenges. We add AREQ-*wo-RE* as a baseline denoted as QMIX [7]. Also, we make a variant of AREQ by using VDN-typed mixer [6] instead of the current QMIX-typed mixer, named as AREQ-*VDN*. Note that VDN does not utilize any extra state information and simply adds up the individual values. Fig. 3c shows that QPLEX and AREQ-*VDN* have significantly larger variance of their performance compared to AREQ, while QMIX converges in a low score. Note that QPLEX shows the fastest learning speed in some of the seeds, but completely fails to learn in other seeds. Learning speed of AREQ-*VDN* is more similar to that of AREQ, but again suffers from instability to the seeds. These results show that AREQ achieves significant improvement in convergence performance in our proposed environment with better stability, and therefore fully taking the advantage of increasing UTD ratio.

7. CONCLUSIONS

In this paper, we introduced a new multi-agent goal assignment and pathfinding environment which requires the agents to allocate themselves to different goal points. Furthermore, we have proposed AREQ, a novel randomized-ensemble-based MARL framework combined with a specialized multi-head-attention network structure. AREQ can efficiently interpret the relations between local geography, goals and other agents. Also, AREQ is robust to high UTD ratio, providing high sample efficiency, and therefore eases the burden of using costly state information. Consequently, AREQ could successfully solve the given tasks in the high UTD setting with better stability. In future work, we plan to extend the AREQ framework to various MARL applications.

REFERENCES

[1] M. Samvelyan, T. Rashid, C. S. D. 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 Extended Abstract," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

[2] X. Chen, C. Wang, Z. Zhou, and K. Ross, "randomized ensemble double q-learning:L Earning M Odel," in *International Conference on Learning Representations (ICLR)*, 2021.

[3] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," in *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[4] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual Multi-Agent Policy Gradients," in *the AAAI conference on artificial intelligence*, 2018.

[5] Y. Wang, B. Han, T. Wang, H. Dong, and C. Zhang, "DOP: Off-Policy Multi-Agent Decomposed Policy Gradients," in *International Conference on Learning Representations (ICLR)*, 2021.

[6] 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 *International Conference on Autonomous Agents and Multiagent Systems*, 2017.

[7] T. Rashid, M. Samvelyan, C. Schroeder De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," in *International Conference on Machine Learning (ICML)*, 2018.

[8] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex Dueling Multi-Agent Q-Learning," in *International Conference on Learning Representations (ICLR)*, 2021.

[9] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX : Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," in *Neural Information Processing Systems (NeurIPS)*, 2020.

[10] S. SHEN, M. Qiu, J. Liu, W. Liu, Y. Fu, X. Liu, and C. Wang, "Resq: A residual q function-based approach for multi-agent reinforcement learning value factorization," in *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022.

[11] K. Son, D. Kim, W. J. Kang, D. Hostallero, and Y. Yi, "QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2019.

[12] N. Carion, G. Synnaeve, A. Lazaric, and N. Usunier, "A Structured Prediction Approach for Generalization in Cooperative Multi-Agent Reinforcement Learning," in *Neural Information Processing Systems (NeurIPS)*, 2019.

[13] J. Xu, F. Zhong, and Y. Wang, "Learning Multi-Agent Coordination for Enhancing Target Coverage in Directional Sensor Networks," in *Neural Information Processing Systems (NeurIPS)*, 2020.

[14] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer Briefs in Intelligent Systems, 2016.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] S. Thrun and A. Schwartz, "Issues in Using Function Approximation for Reinforcement Learning," in *4th Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, 2018.

[18] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "PRIMAL2: Pathfinding Via Reinforcement and Imitation Multi-Agent Learning-Lifelong," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2666–2673, apr 2021.

[19] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv preprint arXiv:1412.3555*, 2014.