Attentive Relational State Representation for Intelligent Joint Operation Simulation

Renlong Chen[1], Ling Ye[2], Shaoqiu Zheng[2](✉), Yabin Wang[2], Peng Cui[2], and Ying Tan[1,3,4,5](✉)

[1] School of Intelligence Science and Technology, Peking University, Beijing 100871, China
ytan@pku.edu.cn

[2] Nanjing Research Institute of Electronic Engineering, Nanjing 210007, China
zhengshaoqiu1214@foxmail.com

[3] Key Laboratory of Machine Perceptron (MOE), Peking University, Beijing 100871, China

[4] Institute for Artificial Intelligence, Peking University, Beijing 100871, China

[5] Nanjing Kangbo Intelligent Health Academy, Nanjing 211100, China

Abstract. In the multi-agent task, due to the constant changes in the location and state of each agent, the information considered by each agent when making decisions is also constantly changing. This makes it difficult to model cooperatively among agents. Previous methods mainly used average embedding to model feature aggregation. However, this aggregation has the problem of losing permutation invariance or excessive information loss. The feature aggregation method based on attentive relational state representation establishes an insensitive state representation to permutation and problem scale. In our experiments on Intelligent Joint Operation Simulation, experimental results show that attentive relational state representation improves the baseline performance.

Keywords: Multi-agent · Intelligent joint operation simulation · Information aggregation · Attention mechanism

1 Introduction

A lot of real-world robotic tasks involve multiple agents with partial observability and limited communication [2]. Agents have capability to extract useful features from neighboring agents to make optimal decisions and cooperation emerges in the group. Typical examples include the swarm robotics [22], traffic signal control [21], collaborative filter [3], and social network analysis [24].

This work is supported by Science and Technology Innovation 2030 - *New Generation Artificial Intelligence* Major Project (Grant No.: 2018AAA0102301), partially supported by Basic Theory Research Foundation of The Science and Technology Commission of the Central Military Commission and the National Natural Science Foundation of China (Grant No. 62076010 and 62276008).

Y. Tan and Y. Shi (Eds.): DMBD 2022, CCIS 1744, pp. 62–74, 2022.
https://doi.org/10.1007/978-981-19-9297-1_6

For the learning paradigms in multi-agent systems, a centralized controller [5] is theoretically feasible but counters many problems, such as the curse of dimensionality and nonexistence of a possible centralized controller in some real tasks like intelligent transportation systems [10]. Therefore, we focus on a decentralized protocol in multi-agent reinforcement learning (MARL), where agents are connected by a time-varying topology structure, and they aggregate information from all their neighbors. This also promotes the scalability and robustness of multi-agent systems.

However, when decentralized artificial intelligence (AI) agents are trained in an interactive environment, it is tricky to handle the state representation issue because the neighborhoods are highly flexible and scalable. One of the previous approaches to represent the aggregated state is fixing the number of local team members and simply concatenating the information received from neighboring agents, as the input dimension must be invariant in neural-network policies and other machine-learning models. We argue that these formulations lack flexibility. Another popular protocol is pooling embedding, such as max-pooling and mean-pooling. Even though pooling method secures invariant input dimension, it loses much information among neighboring agents.

In this article, we utilize an attention based aggregation method called Attentive Relational State Representation (ARE) to efficiently aggregate information from neighboring agents. By modeling the attention between different neighbors, ARE can actively select relevant information discriminately. By pooling, it constructs a unified state representation for learning policies. With this embedding, we condition the policy and train them simultaneously by deep reinforcement learning (DRL). The compact representation makes the learned policy robust to the changes in the multi-agent system and also reduces the search space for the policy learning method. Enabling learning in this framework opens up the possibility of applying learning-based methods to multi-agent interacting environments where neighbors need to be modeled explicitly and both the quantity and identity are changeable over time.

In our experiments, we apply ARE to Intelligent Joint Operation Simulation, which is a confrontation simulation game. In game setting, the blue side is the defensive side and red side is the offensive side. The blue side relies on the ground, sea and air three-dimensional air defense fire to defend the key targets of the two command posts on your island. While the red side comprehensively uses sea and air assault and supports supporting forces to break through the blue air defense system and destroys the key targets of two command posts of the blue side. Experimental results show that ARE helps decision making progress and outperforms baseline algorithm which indicates that ARE is a more efficient information aggregation method than conventional methods (Fig. 1).

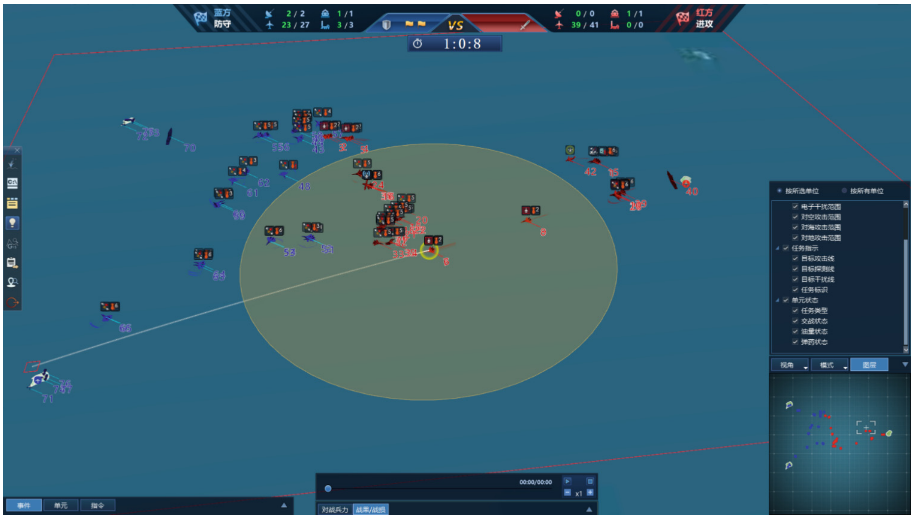


Fig. 1. Joint operation simulation

2 Related Work

2.1 Multi-agent Reinforcement Learning

Learning in the multi-agent system is essentially more difficult than in the single-agent cases, as multiple agents not only interact with the environment but also with each other [1,4,17]. Directly applying the single-agent RL algorithms to the multi-agent system as a whole is a natural approach, which is called the centralized MARL (also called joint action learner [5]). Centralized MARL models the interaction between agents by tightly coupling everything inside the model. Although feasible in execution, it suffers from the curse of dimensionality [4] due to the large-scale joint input space and action space. Thus, decentralized structure has more advantages toward scalability, robustness, and speedup [7,14,18,26].

In the decentralized MARL, a lot of attention has been given to the problem of modeling other agents [1]. In this article, we focus on how to aggregate the information collected from multiple agents, and we make a short survey on the information aggregation approaches in MARL.

2.2 Feature Aggregation in MARL

Concatenation. Concatenation is the simplest and most popular approach in multi-agent RL. By concatenating other features, the augmented state contains all necessary information for decision making. MADDPG [11] constructs the critic for each agent by concatenating other agents' observations and actions, from which the agents can effectively train their actors. A centralized critic is also used in COMA [6], to implement difference reward by comparing with a counterfactual baseline. These methods are under the paradigm of centralized learning with decentralized execution [6,11,12,14,20] which is inspired from DEC-POMDPs [13]. However, concatenation will make the dimension increase linearly, which scales poorly to the large size system. Also, the agent number and identities must be fixed, which is impractical in changeable environments.

Mean Embedding (ME). ME is a workable approach when dealing with a variable dimension problem. By calculating a mean representation, the output has an invariant dimension no matter how many agents are involved. CommNet [19] learns the communication model by rescaling the communication vector by the number of agents to aggregate information. [25] introduced the mean-field theory to MARL. The interactions within the group are approximated by those between a single agent and the average effect from the overall population or neighboring agents. [9] also used the ME method to tackle the representation learning problem in the swarm system. The ME has the advantage of scalability, including dimension and permutation invariance. However, the mean computation is isotropic. The agent has no knowledge of each of its neighbors when pooling averagely around its local view, which may cause ambiguous estimation in many multi-agent tasks where pairwise interactions are important for cooperative decision making.

3 Method

In this section, we first introduce background and give the definition of notations, then introduce ARE structure.

3.1 Background and Notations

We consider multiple agents operating in a partially observable stochastic environment, modeled as a partially observable Markov decision process (POMDP). A stochastic game G is defined by a tuple $< S, U, P, r, Z, O, N, A, \gamma >$, where N agents, $A = \{a_1, a_2, \dots, a_N\}$, are in an interactive environment. $s \in S$ is the true state of the environment. At each time step, all agents simultaneously execute actions yielding a joint action $\mathbf{u} \in U$ then receive observation $\{o_i\}$ determined by observation function $O(s,u) : S \times U \to Z$, and rewards $r(s,u) : S \times U \to \mathbb{R}$ for profits.$P(s' \mid s, u) : S \times U \times S \to [0,1]$ is the state transition probability function, and γ is the discount factor. We denote joint quantities over agents in bold, joint quantities other than a specific agent a with the subscript a, i.e., $\mathbf{u} = [u_a, \mathbf{u}_{-a}]$. All agents take the goal of maximizing the discounted reward of r_t.

We consider the parameter-sharing decentralized control [8]. For simplicity and focusing on the representation problem, we assume that each agent can perceive the features of its neighbors in a local sensing range, and there is no other communication protocols.

3.2 Attentive Relational Encoder

An overview of the inference flow is illustrated in Fig. 2. All agents may behave in a possibly time-varying relation network $\mathcal{G}_t = (A, E_t)$, where E_t stands for the set of neighborhood links connecting agents at the time t. In an agent-centric view, we denote the neighboring feature set for the agent i as $\mathcal{N}_i = \{o_j\}_{j \in \mathcal{G}^i}$ where $\mathcal{G}^i$ is the sub-graph of $\mathcal{G}$ induced by all agents adjacent to agent i (we leave out t for brevity). Therefore, our task is to design a function f with trainable weights θ to map the neighborhood feature set to a fixed size of aggregated high-level features, $y : y_i = f(\mathcal{N}_i, \theta)$, where $i \in 1, 2, \ldots, N$.

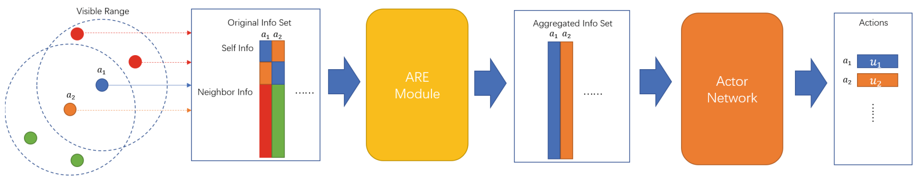


Fig. 2. Overview of the ARE in policy inference.

We utilize a compact neural-network-based architecture, ARE, to aggregate the information from neighboring agents group, whose size is changeable either due to the join or quit of agents. The basic idea of our ARE module is to learn an attention score for each neighbor's feature in the entire neighborhood set. The learnt score can be regarded as a credit that automatically selects useful latent features. For example, within a team of robots moving toward their separate goals, one robot may not care about some neighbors which are behind its moving direction although they are very close. The selected features are then pooled across all elements of the set to aggregate the information and finally served as the state representation for the subject agent.

Figure 3 illustrates the main components of our approach and its execution flow. ARE consists of three encoders, E^f, E^c, and E^a, where $\{f, c, a\}$ stand for feature embedding, communication embedding, and attention embedding. In particular, as shown in Fig. 3, we first feed all the original features (self-feature as well as neighboring features) into two shared encoders E^f and E^c. E^f can be regarded as an intrinsic encoder, which keeps valuable latent features for constructing representation, while E^c is an extrinsic encoder, which reserves the crucial information for interactive relation modeling. Thus, we obtain two streams of latent vectors, e^f and e^c:

$$e_i^f = E^f(o_i) \tag{1}$$

$$e_i^c = E^c(o_i) \tag{2}$$

Second, ARE computes attention scores using the latent vectors e^c for each corresponding neighboring agent through E^a, taking the self feature e_i^c, the cor-

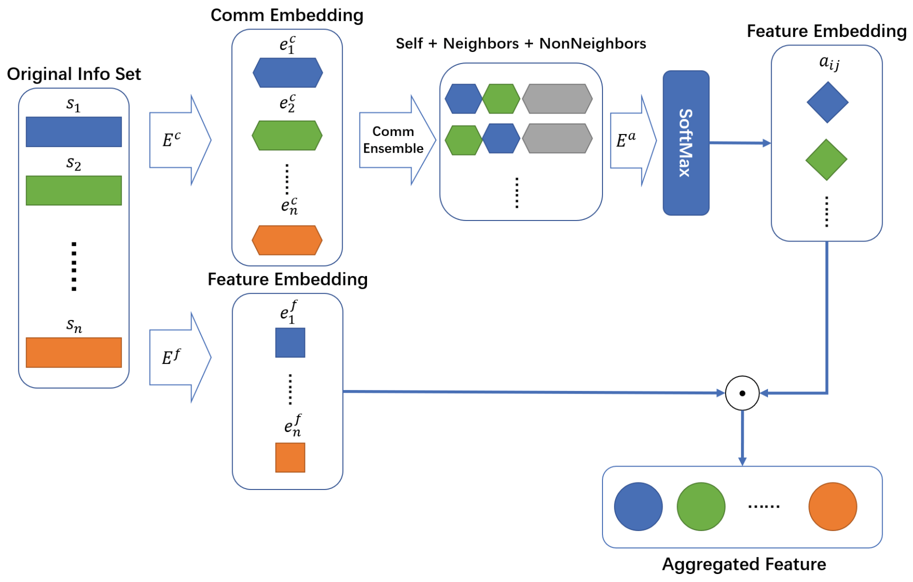


Fig. 3. ARE module on aggregating neighboring features.

responding neighbor's feature e_j^c, as well as an Mean Embedding for other neighboring agents in $\mathcal{G}_i$ other than the agents i and j

$$e_{ij}^a = E^a\left(e_i^c, e_j^c, \overline{e^c}_{-ij}\right) \tag{3}$$

$$\overline{e}^c_{-ij} = \frac{\sum_{k \in \mathcal{G}_i - \{i,j\}} e_k^c}{\|\mathcal{G}_i - \{i,j\}\|} \tag{4}$$

It is worth emphasizing that the self feature is also included in the neighboring feature set in ARE to evaluate the attention score for each agent itself.

We add another channel of ME $\overline{e^c}_{-ij}$ besides pairwise features, in order to model the other neighbors' effect on the pairwise interaction. The output of the function E^a is a set of learnt attention activations $\left\{e_{ij}^a\right\}_{j \in \mathcal{G}_i}$. This procedure is similar to the query-key system [23].

$$e_{ij} \propto \phi\left(e_i^T W_k^T W_q e_j\right) \tag{5}$$

where each sender broadcasts a key transformed by W_k, while the receiver broadcasts a query transformed by W_q. The multiplication of these two parts interprets the relevance or utility of the message. However, we implement this by a neural layer E^a, where the high-level hidden state in neural net can model more abundant interactions between two agents than the query-key system, and generate the attention scores for aggregation.

Third, the learnt attention activations are normalized across the neighborhood set computing a set of attention weights $\overrightarrow{a_i} = \{a_{ij}\}_{j \in \mathcal{G}_i}$. We choose softmax as the normalization operation, so the attention weight for the j-th neighboring feature is

$$a_{ij} = \frac{\exp\left(e_{ij}^{a}\right)}{\sum_{k \in \mathcal{G}_i} \exp\left(e_{ik}^{a}\right)} \tag{6}$$

Subsequently, the computed attention weights are multiplied by their corresponding intrinsic latent features in e^f, generating a new set of deep weighted features. Finally, these weighted features are pooled by summing up across the neighborhood set, producing a fixed size of aggregated features which are then fed into a shared decoder to the downstream control policy, as illustrated in Fig. 2.

$$y_i = \sum_j a_{ij} e_j^f \tag{7}$$

$$\pi_i = \text{decoder}\left(y_i\right) \tag{8}$$

In essence, as the weighted features can be parallelly computed and pooled, the output of the ARE module y_i is permutation invariant with regard to the input order.

We here highlight the specific form of the attention weight. In 3, the attention embedding is generated in the scalar value form. To model complex interaction, we can design e_{ij}^a as vector. Therefore, the attention score a_{ij} in 6 is also vector and 7 is revised

$$y_i = \sum_j \left(a_{ij} \cdot W_a\right) \odot e_j^f \tag{9}$$

where we first unify the dimension by multiplying a matrix W_a, then do the Hadamard product with e_j^f. For simplicity, we set the dimension of the attention vector a_{ij} to be the same with e_j^f, thus W_a becomes an identity matrix, and can be ignored.

Design Discussion. In terms of the flexibility in multi-agent state representation learning, the ARE architecture is designed with the following desirable properties and advantages over existing approaches.

– Computational Efficiency: ARE is computationally high efficient since all operations are parallelizable across the neighboring pairs and all modules are shared.
– Quantity Invariance: Although the size of the neighboring feature set can be arbitrary, the output representation is still irrelevant as sum pooling is utilized. This property makes ARE scalable to the changeable and dynamic interactive environments
– Differentiation Ability: Our method is capable of differentiating the utility of multiple neighbors. By feeding each neighbor's feature together with self feature to the attention module and applying the attention mechanism on these features, ARE is able to attach importance to more relevant neighbors' features.

3.3 Training: Proximal Policy Optimization

The ARE module is trained end-to-end by reinforcement learning. The utilized training backend algorithm is Proximal Policy Optimization (PPO) [16].

Proximal Policy Optimization, or PPO, is a policy gradient method for reinforcement learning. The motivation was to have an algorithm with the data efficiency and reliable performance of TRPO [15], while using only first-order optimization.

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi\theta(a_t|s_t)}{\pi_{\theta_{dd}}(a_t|s_t)}$, so $r\left(\theta_{\text{old}}\right) = 1$. TRPO maximizes a "surrogate" objective:

$$L^{\text{CPI}}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{dd}}\left(a_t \mid s_t\right)}\right)\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right] \tag{10}$$

where CPI refers to a conservative policy iteration. Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update; hence, PPO modifies the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1:

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \operatorname{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{11}$$

where ϵ is a hyperparameter, say, $\epsilon = 0.2$.

4 Experiment

We test our method on Joint Operation Simulation, which is a military operation scenario with red and blue sides which need players to make decisions to achieve intended goals respectively. We will first introduce the task and then give the comparison of our method and baseline algorithm on Joint Operation Simulation.

4.1 Joint Operation Simulation

Scenario Background. The Blue side has long occupied the Red side's islands and recently harassed the Red side's ships for daily operations at sea. In order to swear sovereignty and protect their own interests, joint air and sea combat forces are deployed to strike the Blue side's key targets on the islands to establish a basis for subsequent retaking of the islands. Blue side target (defensive side): rely on ground, sea and air three-dimensional anti-aircraft firepower, guard their own island 2 command post key targets. Red side objective (offensive side): to use a combination of air and sea assault and support forces to break through the blue side's air defense system and destroy the blue side's 2 key command post targets.

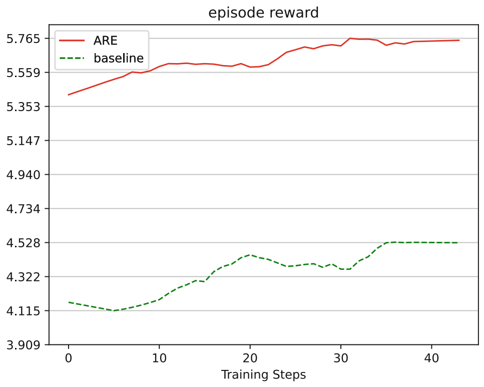


Fig. 4. Comparison of ARE and mean embedding baseline on joint operation simulation.

Scenario Settings. For the characteristics of the two sides, a reasonable set of force composition to confront the idea of rehearsal. Each side has only one airfield, and each airfield has only one runway for the takeoff and landing of combat aircraft. When an aircraft takes off and occupies the runway, no other aircraft can land, and when an aircraft lands, no other aircraft can take off. Through the airport takeoff and landing density to control the speed of force release, to achieve the control of AI available force, consider the tournament against compact, set the minimum interval of aircraft takeoff and landing (by the simulation platform internal control).

Table 1. Comparison of force composition of both sides

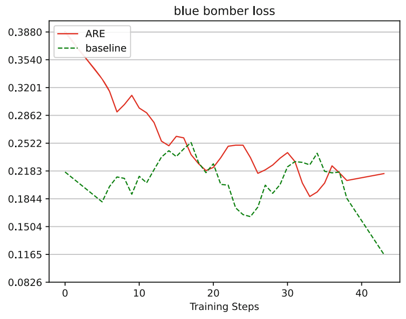
	Bomber	AWACS	Jammer	Fighter	Frigate	Radar	Airport	Camp	CC
Red	18	1	1	24	2	1	1	-	-
Blue	10	1	-	20	2	-	1	3	2

Table 1 shows the force composition of both sides. Red side as the offensive side consists of 18 Bombers, 1 AWACS, 1 Jammer, 24 Fighters, 2 Frigates, 1 Radar and 1 Airport. Blue side has a different setting, which consists of 10 Bombers, 1 AWACS, 2 Radars, 20 Fighters, 2 Frigates, 3 Camps, 1 Airport and 2 Command Centers.

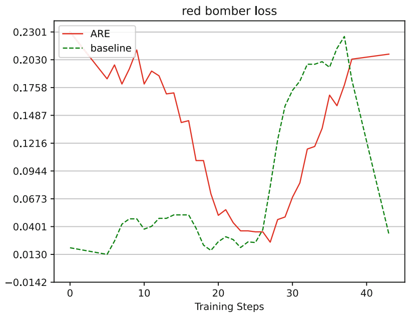
For reward settings, all enemy units share 10 points, blue side has a bonus for keeping Command Centers alive, 1 point for each Command Center. Considering the requirement of retaining own units, we also punish unit loss on each side.

Figure 7 gives the composition of entity features, including positions, side, type, speed, direction, damage, alive, weapon, locked. last mission type. ARE aggregates information from each entity, and yields an aggregated and embedded feature containing neighbors' information, which helps a better decision making.

4.2 Results

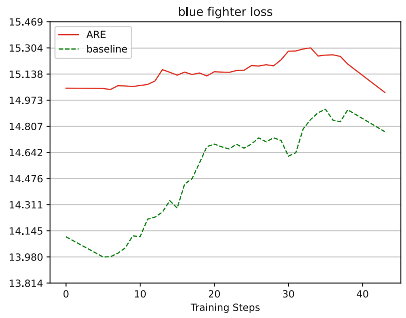


(a)

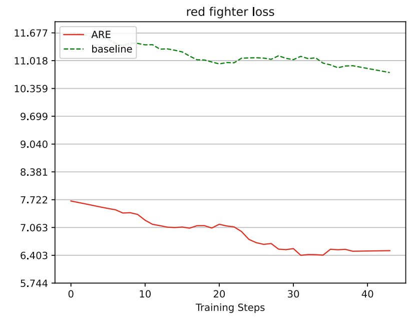


(b)

Fig. 5. Comparison of bomber loss of both sides



(a)



(b)

Fig. 6. Comparison of fighter loss of both sides

Both ARE and baseline use PPO as a training backend in our experiments. The red side is set to use RL training algorithms while the blue side is set to use a fixed rule-based method. Figure 4 shows that ARE helps PPO to achieve a higher episode reward throughout the whole training progress. Then more detail comparisons will be given.

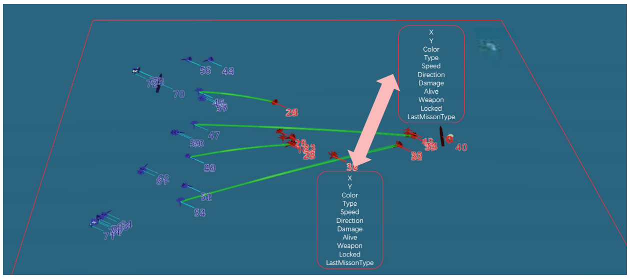


Fig. 7. Composition of entity features.

As is shown in Fig. 5a, ARE based algorithm achieves more average blue side bomber loss during the training progress. The red side bomber loss in Fig. 5b reflects the radical level how a strategy to use bombers, where both ARE and ME baseline vary a lot in different training phases.

Figure 6 shows that ARE outperforms ME baseline by a large margin for a higher damage to blue side and lower red side loss in fighters. It indicates that ARE helps fighters to coordinate with ally units to formulate a more efficient strategy than ME does.

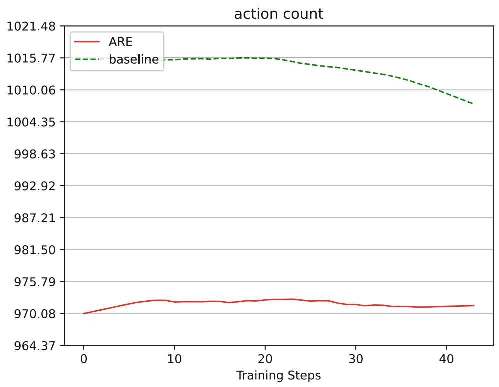


Fig. 8. Comparison of action count per episode.

More actions usually mean more cost in real battles. We also count the action numbers per episode of both methods in Fig. 8. ARE uses significant less actions per episode than ME baseline method, which means ARE helps algorithm learn a much more efficient strategy.

5 Conclusion

In this paper, we analyze the information aggregation problems in multi-agent systems and utilize a computational efficient, quantity invariant and differentiable aggregation method, ARE in a highly simulated joint operation game. Experimental results show that ARE helps decision making progress and outperforms baseline algorithm which indicates that ARE is a more efficient information aggregation method than conventional methods.

References

1. Albrecht, S.V., Stone, P.: Autonomous agents modelling other agents: a comprehensive survey and open problems. Artif. Intell. **258**, 66–95 (2018)
2. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of Markov decision processes. Math. Oper. Res. **27**(4), 819–840 (2002)
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. arXiv preprint arXiv:1301.7363 (2013)
4. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **38**(2), 156–172 (2008)
5. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. AAAI/IAAI **1998**(746–752), 2 (1998)
6. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
7. Guestrin, C., Lagoudakis, M., Parr, R.: Coordinated reinforcement learning. In: ICML, vol. 2, pp. 227–234. Citeseer (2002)
8. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multi-agent control using deep reinforcement learning. In: Sukthankar, G., Rodriguez-Aguilar, J.A. (eds.) AAMAS 2017. LNCS (LNAI), vol. 10642, pp. 66–83. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71682-4_5
9. Hüttenrauch, M., Adrian, S., Neumann, G., et al.: Deep reinforcement learning for swarm systems. J. Mach. Learn. Res. **20**(54), 1–31 (2019)
10. Li, W.: Notion of control-law module and modular framework of cooperative transportation using multiple nonholonomic robotic agents with physical rigid-formation-motion constraints. IEEE Trans. Cybern. **46**(5), 1242–1248 (2015)
11. Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
12. Mao, H., Gong, Z., Ni, Y., Xiao, Z.: ACCNet: actor-coordinator-critic net for "learning-to-communicate" with deep multi-agent reinforcement learning. arXiv preprint arXiv:1706.03235 (2017)
13. Oliehoek, F.A., Amato, C.: A Concise Introduction to Decentralized POMDPs. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-28929-8
14. Omidshafiei, S., Pazis, J., Amato, C., How, J.P., Vian, J.: Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: International Conference on Machine Learning, pp. 2681–2690. PMLR (2017)

15. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning, pp. 1889–1897. PMLR (2015)
16. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
17. Shoham, Y., Powers, R., Grenager, T.: If multi-agent learning is the answer, what is the question? Artif. Intell. **171**(7), 365–377 (2007)
18. Su, S., Lin, Z., Garcia, A.: Distributed synchronization control of multiagent systems with unknown nonlinearities. IEEE Trans. Cybern. **46**(1), 325–338 (2015)
19. Sukhbaatar, S., Fergus, R., et al.: Learning multiagent communication with backpropagation. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
20. Sunehag, P., et al.: Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:1706.05296 (2017)
21. Tan, T., Bao, F., Deng, Y., Jin, A., Dai, Q., Wang, J.: Cooperative deep reinforcement learning for large-scale traffic grid signal control. IEEE Trans. Cybern. **50**(6), 2687–2700 (2019)
22. Tan, Y., Zheng, Z.Y.: Research advance in swarm robotics. Defence Technol. **9**(1), 18–39 (2013)
23. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
24. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
25. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., Wang, J.: Mean field multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 5571–5580. PMLR (2018)
26. Zhang, K., Yang, Z., Liu, H., Zhang, T., Basar, T.: Fully decentralized multi-agent reinforcement learning with networked agents. In: International Conference on Machine Learning, pp. 5872–5881. PMLR (2018)