Cooperation and Competition: Flocking with Evolutionary Multi-Agent Reinforcement Learning

Yunxiao Guo†, Xinjia Xie†, Runhao Zhao†, Chenglan Zhu, Jiangting Yin, and Han Long

1 College of Sciences, National University of Defense Technology, Changsha 410073, China
2 College of Computer Science, National University of Defense Technology, Changsha 410073, China
3 College of System Engineering, National University of Defense Technology, Changsha 410073, China
4 Beijing International Studies University, Beijing 100000, China

Abstract. Flocking is a very challenging problem in a multi-agent system; traditional flocking methods also require complete knowledge of the environment and a precise model for control. In this paper, we propose Evolutionary Multi-Agent Reinforcement Learning (EMARL) in flocking tasks, a hybrid algorithm that combines cooperation and competition with little prior knowledge. As for cooperation, we design the agents’ reward for flocking tasks according to the boids model. While for competition, agents with high fitness are designed as senior agents, and those with low fitness are designed as junior, letting junior agents inherit the parameters of senior agents stochastically. To intensify competition, we also design an evolutionary selection mechanism that shows effectiveness on credit assignment in flocking tasks. Experimental results in a range of challenging and self-contrast benchmarks demonstrate that EMARL significantly outperforms the full competition or cooperation methods.

Keywords: Flocking · Multi-Agent Reinforcement Learning · Evolutionary Reinforcement Learning · Swarming Intelligence

1 Introduction

Such as migratory birds and wasps, flocking in swarm intelligence refers to the behavior of a large population that tend to gather in groups and move orderly [21]. It also be widely employed in numerous applications such as social evolution[4], multi-robotics control[5], and traffic model[10]. The research on the
flocking model can help people understand the macroscopic dynamics of the population and its dependence on parameters\cite{9} and also guide people in designing the control methods for the robotic population, such as UAVs\cite{11}.

The Traditional flocking method usually uses partial differential equations (PDEs) to describe the agent behavior in the population\cite{3,19,26}, by solving the PDEs with numerical techniques, the flocking strategies can be found. However, the PDEs are generally difficult to solve, and when the agent number is large, and the geometry is complex, the computation cost will increase. It also needs the model to know the full knowledge of the environment\cite{20}.

For the above reasons, the methods using deep reinforcement learning (DRL) to train the agents flocking self-organized have attracted lots of interest in recent years\cite{20,11,6}, especially the model-free multi-agent DRL (MADRL), which can handle the complicated tasks well without modeling the complex environment. Compared to the single-agent DRL algorithms, MADRL can also deal with more complicated problems\cite{8}, but also bring about new challenges like credit assignment, which is difficult to solve, particularly in the full-cooperative tasks like flocking.

Besides, the reward design in the flocking task is essential. For the full-cooperative tasks, if the individual agent gets the same reward as the rest of the group no matter what it does, then the agent will find it hard to know their contribution to the group, leading to failed learning. In biological, the behavior of agents to pursue higher individual rewards in the group can be seen as competition, and the lack of competition among the agents will constrain the performance of learning \cite{2}. Nevertheless, we should not apply the competitive MARL methods such as the MARL based on Nash equilibrium. It will lead to the local optimal solution for the cooperative tasks because the agent would not sacrifice for the greater team reward.

In this paper, we proposed the Evolutionary Multi-Agent Reinforcement Learning (EMARL) algorithm, which introduces competition to solve the credit assignment in the full-cooperative flocking task. Our main contributions are as follows: (1) We redesign the reward of the flocking task based on the boids model and solve the credit assignment challenge in the flocking task. We propose EMARL that combines competition (Evolutionary RL) and cooperation (Multi-agent RL). (2) In the part of evolutionary RL, we propose evolutionary selection to sift agents with poor performance and prove the convergence of policy gradient with Evolutionary Selection which further demonstrates the effectiveness. (3) We illustrated our approach to the flocking task with two obstacles and demonstrated the results at different levels of competition and cooperation.

## 2 Background

### 2.1 Boids Model

To guide the agents to flock efficiently, literature\cite{22} proposed the three basic rules in agents’ observable space (inside the red dotted circle in Fig. 1:  

– **Separation**: To avoid the crash of agents during flocking process, agents are required to keep a distance from others observed. For example, the agent (blue arrow) in Fig.1 (a) will change its toward to the orange dotted arrow.

– **Alignment**: It drives the agents to move in the same speed direction. Fig.1 (b) shows an agent who gets the acceleration to follow the other agents.

– **Cohesion**: To expand the scale gradually, agents who are nearby are attracted to join. The agent (blue arrow) in Fig.1 (c) will get an acceleration along the orange dotted arrow to keep the flock close to the blue node.

Based on these rules, different flocking models were proposed adding natural population properties, such as [6,24]. We introduces the three rules into the action-value function estimation process of the multi-agent RL approach.

### 2.2 Multi-Agent & Evolutionary Reinforcement Learning

The flocking task can be considered a full-cooperation stochastic game, so it can be modeled as a partially observable Markov decision process (POMDP). POMDP also can be defined by a tuple \((S, U, P, Z, O, R, \gamma)\). At time \(t\), \(s_t \in S\) represents the true state of the environment, the joint action of all agents is \(\{u^1_t, u^2_t, \cdots, u^n_t\} = \mathcal{U}_t \subset \mathcal{U}^n\), where \(u^k_t \in \mathcal{U}\) represent \(k\)th agent action. The map \(\mathcal{P} : S \times \mathcal{U} \times S \rightarrow [0, 1]\) represents the probability that agents take the joint action \(\mathcal{U}_t\) at the state \(s_t\) then transit to \(s_{t+1}\). The state that the \(k\)th agent gains from the environment \(z^k_t \in \mathcal{Z}\) is partially observable. The map \(\mathcal{O} : S \rightarrow \mathcal{Z}\) is the observe function that maps the true state to the observed state. The observation-state history \(\tau^k_t = (z^k_0, u^k_0, z^k_1, u^k_1, \cdots, z^k_t)\) recorded the state and action of \(k\)th agent from time 0 to \(t\). After \(k\)th agent takes the action at the observed state, it will receive a reward \(r^k_t\) defined by: \(R : S \times \mathcal{U} \rightarrow \mathbb{R}\).

To solve POMDP, we maximize the discount return: \(R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}\) by learning a joint policy decomposed of all \(n\) agent independent stochastic policies: \(\pi(u | s, \theta) = \prod_{i=1}^{n} \pi_i(u^i | \tau^i, \theta_i)\).

A common MARL framework is Centralized Training with Decentralized Execution (CTDE) [517]. Each agent in the CTDE paradigm has a Critic for
global information training and several actors for local information training, while COMA\cite{8} tells an agent how much the selected action contributes to the current reward and avoids estimating the local action-value. However, they cannot address the credit assign problem effectively, as paper \cite{8} pointed out that using global rewards as the estimation of $R_0$ fails.

Evolutionary Reinforcement Learning (ERL) \cite{13,14,29,18} combines evolutionary algorithm and RL. A common ERL can be summarized as follow: we generate an agent group and train them through RL. After training, a fitness function evaluates all the agents’ performance, then select several elites by ranking their fitness value. Then, introduce the mutation mechanism like evolutionary algorithms for generating next-generation agents, and keep selecting the agent until the stop conditions. It shows highly competitive among the agents’ learning process. During the process, the best agents will be retained.

3 Proposed Method

3.1 Cooperation: The Flocking Task

In this section, we introduce the agents’ state and action space of the flocking tasks, and design the reward function.

State & Action Space

Considering a general flocking environment that consists of agents and obstacles. Each agent has the same velocity module $||v||$ and only observe the nearest $k$ agents’ state. The state of each agent includes its position and velocity (both are vectors). Set the action of agents to rotate in the clockwise direction, then, given the rotation size $\omega$ and the action space size $|A|$ of individual agent, the action space can be generated as $\{-\omega, -(|A|-1)\cdot\omega, \cdots, 0, \cdots, (|A|-1)\cdot\omega, \omega\}$.

Reward Function

Three flocking rules of Boids model: separation, alignment, cohesion, were modeled respectively by giving rewards to the behavior of agents in the environment. To avoid obstacles and focus on these rules, the reward of the whole Multi-agent flocking system at the time $t$ is given as follows:

$$R(t) = R_{sep}(t) + R_{ali}(t) + R_{coh}(t) + R_{obs}(t)$$

where $R_{sep}, R_{ali}, R_{coh}$ represent the reward obtained from flocking rules, and $R_{obs}$ is the reward from avoid obstacle.

For the reward of separation, the farther the $i$th agent is from all observed agents, the higher the reward is. Therefore, we utilize the average module of position vectors’ difference between $i$th agent and all $n_i$ observed agents as the single agent’s reward. Meanwhile, to avoid the distance between agents increasing infinitely by directly using the vectors’ difference module as the reward, the penalty parameter $d_s$ is given to punish the agent. The agent has received a negative reward for getting too close to other agents (less than $d_s$), but this reward does not increase with a distance greater than $d_s$:

$$R_{sep,i}(t) = \begin{cases} -\frac{1}{n_i} \sum_{j=1}^{n_i} e^{-\beta_{sep} ||r_i-r_j||}, ||r_i-r_j|| < d_s \\ 0, \text{other} \end{cases}$$

(2)
For the second rule: **alignment**, we normalized all the agent’s velocity vector: $$\vec{u}_i = \frac{\vec{v}_i}{||\vec{v}_i||}$$, and use vectors’ difference like Eq.2:

$$R_{ali,i}(t) = \frac{1}{n_i} \sum_{j=1}^{n_i} e^{\beta_{ali} ||\vec{u}_i - \vec{u}_j||}$$ (3)

If we know the angle between each agent and the coordinate system: $$\theta_1, \cdots, \theta_n$$, where $$\theta = \arctan(\frac{v_y}{v_x})$$, the alignment reward can be rephrased as follow:

$$R_{ali,i}(t) = \frac{1}{n_i} \sum_{j=1}^{n_i} e^{-\beta_{ali} [2-\cos(\theta_i - \theta_j)]}$$ (4)

**Cohesion** can be modeled as the distance between agent $$i$$ and all $$n_i$$ observed agents. As we didn’t hope that all the agents get too close, a penalize parameter $$d_c$$ like Eq. 5 is introduced:

$$R_{coh,i}(t) = \begin{cases} e^{-\beta_{coh} ||\vec{r}_i - \frac{1}{n_i} \sum_{j=1}^{n_i} \vec{r}_j||}, & ||\vec{r}_i - \frac{1}{n_i} \sum_{j=1}^{n_i} \vec{r}_j|| \geq d_c \\ 0, & \text{other} \end{cases}$$ (5)

For driving agents to avoid obstacles in the environment, construct the obstacle reward as a constant function determined by whether agents hit the obstacle:

$$R_{obs,i}(t) = \begin{cases} -\beta_{obs}, & \text{If agent hits obstacle} \\ 0, & \text{other} \end{cases}$$ (6)

Finally, we obtain the adding-up reward of an agent at time $$t$$. For the multi-agent system, the total reward is represented as the sum of all agents’ rewards:

$$R(t) = \sum_{i=1}^{N} (R_{sep,i}(t) + R_{ali,i}(t) + R_{coh,i}(t) + R_{obs,i}(t))$$ (7)

As a fully cooperative game, the flocking system hopes the overall return as large as possible, and the target is to maximize the total reward Eq[7] instead of each agent’s reward. Thus we pursues the Pareto dominance (globally optimal solution) instead of Nash equilibrium (locally optimal solution) [10].

### 3.2 Competition: Evolutionary Multi-Agent Reinforcement Learning

We developed an algorithm named **Evolutionary Multi-Agent Reinforcement Learning (EMARL)**, which use MARL to drive the agents to complete flocking task full-cooperatively. Meanwhile, the trick of ERL is introduced simultaneously to encourage the agents to learn competitively and solve credit assignment in full-cooperatively MARL. The pseudocode of EMARL is shown in AppendixA.1 Algorithm 1 and the whole framework in Fig. 2(a).
Fig. 2: (a) The framework of Evolutionary Multi-Agent Reinforcement Learning (EMARL); (b) The schematic diagram of Evolutionary Selection

For a cooperative task, flocking require the agents to cooperate and maximize the team reward and an agent must sacrifice its reward in exchange for the higher team reward. To achieve cooperation, we use centralized critic to estimate the team reward to train actors decentralized. Randomly initialize the actors and critic’s parameters \( \{ \theta^\pi_1, \ldots, \theta^\pi_n \} \) and \( \theta^Q_1 \) respectively, all the actors and critic are neural networks. The actors interact with the environment simultaneously, storing their individual history \( \tau_{i,t} \) into their individual buffer \( \mathcal{R}_i \). After all the buffers are updated, the critic minimizes the loss
\[
\mathcal{L}(\theta^Q) = \frac{1}{T} \sum_{i=1}^{T} (y_i - Q(s_i, u_i|\theta^Q))^2
\]
by sampling the transition \((o_i, u_i, r_i, o_{i+1})\) from the total buffer \( \mathcal{R} = \bigcup_{i=1}^{n} \mathcal{R}_i \). When updating the critic, because the train is centralized, we use the true global state \( s_t \) to replace the observation \( o_t \). Then, update all the actors by sampling the transition from their individual buffer \( \mathcal{R}_i \).

Fig 2(a) illustrates all the setup. In addition, to reduce the variance, we introduce the state value function \( V(s_t) \) as the baseline [15,25], with the policy gradient:
\[
g_{\theta^\pi_i} = \mathbb{E}_\pi[\nabla_{\theta^\pi} \log \pi_i(s, u|\theta^\pi)(Q(s, u|\theta^Q) - V(s|\theta^Q))]
\] (8)

There is an unovercome challenge of credit assignment on this method. Inspired by ERL that introduces evolutionary rules to enforce agents’ competition with each other, and left high fitness one, eliminate low fitness one, we design an evolutionary selection stage that partially replaces low perform agents’ parameters to solve this problem. As is shown in Fig 2(b), after the \( k \)th updating, calculate every agent’s \( \xi \) steps accumulative reward as its fitness value. Then, ranking the fitness value and selecting the first \( n_S \) agents as senior agents, the last \( n_J \) agents as junior agents, and using the percentage of the fitness value of the \( i \)-th junior agent to the sum fitness of all senior agents as the probability of the \( i \)-th agent parameter being replaced. We call it happens as inherit with probability. Other agents that are not included in junior agents will be kept and continue to be updated. For the pseudocode, see Algorithm 2 in the Appendix A.1.
Unlike traditional ERL, **Evolutionary Selection** does not evolve all the agents. It only selects well-performed \( n_S \) agents to form senior agents and replace parameters of junior agents, which consist of worse-performed \( n_J \) agents. That’s because **not all agents that do not perform as well as others have poorer policies**. In many cases, the policy did not show their advantage because the agent local on a bad position (e.g., the agent in the obstacle area like the blue arrows in Fig4), so the evolutionary selection only selects the worst \( n_J \) agents to let them inherit the parameters from senior agents. In the view of the evolutionary algorithm, a larger \( n_S : n_J \) ratio indicates that the algorithm is more competitive (the experiments will be designed to explore it).

Next, we give the convergence analysis of policy gradient with evolutionary selection.

### 3.3 Convergence Analysis

Before we analyze the convergence of the EMARL, we give following lemma:

**Lemma 1.** The policy gradient of the actor-critic algorithm without evolutionary selection in \( i \)th iteration:

\[
g_{\theta_i} = \mathbb{E}_\pi [\nabla_{\theta_i} \log \pi(u|\tau) A^\pi(s, u)]
\]

will converge when:

\[
\lim_{k \to \infty} \|\nabla \mathcal{J}\| = 0, \text{w.p.} \ 1
\]

Here we give the idea of the proof; the detail can see in AppendixA.2.

**Lemma 2.** If \( \pi_{j,i} \) is the policy of junior agent, after enough updates, the \( \pi_{j,i} \) will inherit the policy parameter from senior agents with probability 1.

The detail of proof can see in AppendixA.3. Using the lemmas, we give the convergence analysis of policy gradient with evolutionary selection:

**Theorem 1.** The policy with evolutionary selection: \( \pi_j^*(u_j|\tau_j), \forall j \in 1, 2, ..., n \), the gradient:

\[
g_{\theta_k}^* = \mathbb{E}_\pi \left[ \sum_{j=1}^{n} \nabla_{\theta_k} \log \pi_j^*(u_j|\tau_j) A^j(s, u) \right]
\]

will converge as well:

\[
\lim_{k \to \infty} \|g_{\theta_k}^*\| = 0, \text{w.p.} \ 1
\]

**Proof.** After the \( k \)th update, collect the actors that have never been selected as junior agents, append their policy to \( P_{NA} \), and append the rest policies to \( P_{NB} \):

- \( P_{NA} = \{\pi_{A,1}^*, \pi_{A,2}^*, \cdots, \pi_{A,N_A}^*\} \), where \( \pi_{A,i}^* \) is the policy that parameters never been inherit from before \( N \) iteration.
- \( P_{NB} = \{\pi_{B,1}^*, \pi_{B,2}^*, \cdots, \pi_{B,N_B}^*\} \), where \( \pi_{B,i}^* \) is the policy that parameter was inherit in previous some iteration.
Assume $P_{N_A}$ is not empty. It is easy to know, $\forall \pi_{A,j}^s$, if $\pi_{A,j}^s \notin P_{N_A}$, then $\pi_{A,j}^s \in P_{N_B}$, and $\pi_{A,j}^s$ will never be the junior agent. $\forall \pi_{A,j}^s$, the parameters of them have not be inherit by others, so, according to Lemma 1, we have $\forall \varepsilon > 0, \exists N_{A,j} > 0$, when $k > N_{A,j}$, for $\pi_{A,j}$, the following equation hold:

$$P\{|\|E_\pi[\nabla_\theta \log \pi_{A,j}^s(u_j|\tau_j, \theta_{\pi_{A,j}^s}) \cdot A_{\pi_{A,j}^s}(s,u)]\| < \varepsilon\} = 1 \quad (13)$$

So we only select $N = \max\{N_{A,1}, N_{A,2}, \cdots, N_{A,N_A}\}$, let $k \geq N$, the policies in $P_{N_A}$ are convergent. For the policy $\pi_{B,i}$ in $P_{N_B}$, according to Lemma 2, $\exists N_{B,i} \geq 0$, when $n \geq N_{B,i}$, $\pi_{B,i}$ will inherit the parameter from senior agent.

So we only select $N_B = \max\{N, N_{B,1}\}$, after $N_B$ update steps, there is some agent in senior agent convergent, once the $\pi_{B,i}$ inherit their parameters, the $\pi_{B,i}$ will convergent. Extend this result to all the policy in $P_{N_B}$, we have when $\max\{N, N_{B,1}, \cdots, N_{B,N_B}\}$ update steps, the policy in $P_{N_B}$ convergent.

All the policies in $P_{N_A}, P_{N_B}$ will converge w.p.1, therefore, the policy with evolutionary selection is converge w.p.1.

Based on Lemma 1, Theorem 1 shows that under the assumption 15, the policy gradient with Evolutionary Selection is also convergence with probability 1.

4 Experiments

4.1 Experimental Setup

Experiments of EMARL with different $n_S$ and $n_J$ were designed for validation. We implement the flocking environment by OpenAI gym, the basic of the codes we refer to [1]. We mainly compare our algorithm with different levels of competition and cooperation by adjusting the ratio of $n_J$ and $n_S$.

**Agents setting:** In the flocking visualization environment, we use a green arrow to represent the agent, and the arrow direction is the agent’s toward. We test 15 and 30 agents in the environment, and these two group experiments include the different settings of $n_S, n_J$. The details of the environment setting can be see in the Appendix.

**Environment Setting:** We construct the environment with two circle obstacles (shallow red area), one has a radius of 0.1(lower-left corner) and the other 0.2 (center) (Fig.4). Our environment allows the agent to swarm inside of the obstacle to simplify the environment. We consider the obstacle as the forbidden zone and the agents who stay in the zone would suffer from the negative reward determined by Eq.6. In our experiments, the alignment reward is not introduced because the relationship between performance and alignment reward is weak in the tentative experiments. Nevertheless, instead of denying the function of alignment reward, our target is to simplify the environment.

**Baselines Setting:** As for comparisons, we selected two training methods. (a) **Non-training:** In general, we need to choose non-training as the baseline to compare the performance of flocking and obstacle avoidance with other methods. (b) **IQL:** There are several representative methods according to MARL. [23] has demonstrated that IQL achieves the state-of-the-art performance.
Fig. 3: Performance of Evolutionary Multi-Agent Reinforcement Learning in flocking task. Each row represents that the maximum number of agents is 15 and 30 respectively, while each column represents $n_S : n_J = 1, 2, 3$ respectively.

4.2 Evolutionary Reinforcement Learning Performance

The performance of the algorithms is shown in Fig. 3, where (a), (b), (c) represent the 15 agents with the number ratio of senior agents to junior agents $n_S : n_J$ equal to 1, 2, 3 respectively. In terms of the group return obtained by agents in the training process, the environment of 15 agents and 30 agents performs excellently when $n_S : n_J = 1 : 2$ or $1 : 1$. In the view of the evolutionary algorithm, a larger $n_S : n_J$ ratio indicates that the algorithm is more competitive.

From the perspective of ERL, when $n_S : n_J = 1 : 3$, Evolutionary Selection is almost consistent with the conventional ERL algorithm. Nearly all agents are selected as senior agents or junior agents. Then, the parameter replacement will happen on them. Our experiment shows that although it can improve the algorithm compared with the traditional way, the extent of improvement is far less than that of the previous two ratios of $n_S$ to $n_J$, which refer to appropriate levels of competition and cooperation.

4.3 Ablation Study

Flocking Visualization In general, the proposed algorithm performs better than the traditional one, and different $n_S$ and $n_J$ settings have different effects.
When $n_S : n_J = 1 : 2$, they perform best. Compared to the return curve of an agent during training, we are more concerned with the position distribution.

The visualized results are shown in Fig. 4, from which we see that before the training, the distribution of all agents is irregular, and some agents even moved into the obstacle. For the 30 agents’ environment, after training with IQL, the agents form the flock, but almost one-third of agents are in the area of the forbidden zone; after training with EMARL, the agents form the obvious flock, with no agent inside the obstacle (forbidden zone), and the movement of the agents tend to avoid the obstacle. While for the 15 agents’ environment, after training with IQL, although there is only one agent is in the forbidden zone, agents do not form any obvious flock; after training with EMARL, there is no agent inside the forbidden zone and the agents form the flock well. The movement of agents tends to avoid the obstacle which indicates the excellent effectiveness of EMARL.
Fig. 5: The visualization of frame by frame superposition of agents flocking. (a): The flocking of 30 agents environment trained with IQL. (c): The flocking of 30 agents environment trained with EMARL.

Frame by Frame Superposition Visualization The flocking visualization of frame by frame superposition obviously shows the obstacle avoidance performance from the dynamic perspective. From Fig. 5 we discover that after training with IQL, the agents moving into the forbidden zone are very intense and frequent, and the formation is not formed in good order. While after training with EMARL, most agents avoid the obstacles and they went on orderly, which proves the advantages of our method more forcefully.

5 Related Work

Adopting RL to solve the flocking problem has received continuous attention. In literatures [11][12], the Q-learning based algorithms are introduced into the UAVs’ flocking task, and the flocking reward is designed well; this work needn’t know complete knowledge but still performs well when the UAV agent number is not too large. The paper [6] proposed a MARL framework for flocking tasks, and they designed a secondary reward that measures how an agent can perform complex cooperative learning driven by flocking. These works also pay more attention to cooperation than competition among the agents, and the algorithms can only deal with small-scale flocking tasks. The literature [5,28,27] introduced DRL and actor-critic architecture, which increase the number of agents the algorithm can handle. The literature [20] proposed a mean-field game-based method, which encourages the agents to reach the equilibrium solution by Soft Actor-Critic algorithm. It is paying more attention to the competition, but the potential problem is that the equilibrium solution may not be the global optimal.

6 Conclusion

In this paper, we propose Evolutionary Multi-Agent Reinforcement Learning (EMARL) which first applies ERL in flocking tasks. We combine cooperation for further
improvement of performance and competition for the credit assignment problem. For cooperation, the agents’ reward for flocking tasks is designed and the target is to maximize the total reward. For the competition, we design a Roulette to make junior agents inherit the parameters of senior agents and an Evolutionary Selection mechanism. Experimental results in a range of challenging and self-contrast benchmarks indicate this algorithm with effectiveness. For multi-agent tasks similar to flocking [7], we would further study on the relationship between the number of agents, the ratio of senior and junior agents, and the degree of emergence, making our method conducive to more complex environments.

References

1. "boid multi-agent rl environment & multi-agent rl agent" (2020), https://github.com/zombie-einstein/flock_env
2. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. arXiv preprint arXiv:1710.03748 (2017)
3. Bardi, M., Cardaliaguet, P.: Convergence of some mean field games systems to aggregation and flocking models. Nonlinear Analysis 204, 112199 (2021)
4. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the national academy of sciences 99(suppl _3), 7280–7287 (2002)
5. Chang, W., Lizhen, W., Chao, Y., Zhichao, W., Han, L., Chao, Y.: Coactive design of explainable agent-based task planning and deep reinforcement learning for human-uavs teamwork. Chinese Journal of Aeronautics 33(11), 2930–2945 (2020)
6. Chen, C., Hou, Y., Ong, Y.S.: A conceptual modeling of flocking-regulated multi-agent reinforcement learning. In: 2016 International Joint Conference on Neural Networks (IJCNN). pp. 5256–5262. IEEE (2016)
7. Drugan, M.M.: Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. Swarm and evolutionary computation 44, 228–246 (2019)
8. 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)
9. Grover, P., Bakshi, K., Theodorou, E.A.: A mean-field game model for homogeneous flocking. Chaos: An Interdisciplinary Journal of Nonlinear Science 28(6), 061103 (2018)
10. Hu, Y., Gao, Y., An, B.: Multiagent reinforcement learning with unshared value functions. IEEE transactions on cybernetics 45(4), 647–662 (2014)
11. Hung, S.M., Givi, S.N.: A q-learning approach to flocking with uavs in a stochastic environment. IEEE transactions on cybernetics 47(1), 186–197 (2016)
12. Hung, S.M., Givi, S.N., Noureddin, A.: A dyna-q (lambda) approach to flocking with fixed-wing uavs in a stochastic environment. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics. pp. 1918–1923. IEEE (2015)
13. Khadka, S., Majumdar, S., Nassar, T., Dwiel, Z., Tumer, E., Miret, S., Liu, Y., Tumer, K.: Collaborative evolutionary reinforcement learning. In: International conference on machine learning. pp. 3341–3350. PMLR (2019)
14. Khadka, S., Tumer, K.: Evolution-guided policy gradient in reinforcement learning. Advances in Neural Information Processing Systems 31 (2018)
15. Konda, V., Tsitsiklis, J.: Actor-critic algorithms. Advances in neural information processing systems 12 (1999)
16. Lighthill, M.J., Whitham, G.B.: On kinematic waves ii. a theory of traffic flow on long crowded roads. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences 229(1178), 317–345 (1955)
17. Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems 30 (2017)
18. Majumdar, S., Khadka, S., Miret, S., McAleer, S., Tumer, K.: Evolutionary reinforcement learning for sample-efficient multiagent coordination. In: International Conference on Machine Learning. pp. 6651–6660. PMLR (2020)
19. Mavridis, C.N., Tirumalai, A., Baras, J.S., Matei, I.: Semi-linear poisson-mediated flocking in a cucker-smale model. IFAC-PapersOnLine 54(9), 404–409 (2021)
20. Perrin, S., Laurière, M., Pérolat, J., Geist, M., Élie, R., Pietquin, O.: Mean field games flock! the reinforcement learning way. arXiv preprint arXiv:2105.07933 (2021)
21. Quera, V.Q.J., Salvador Beltrán, F., Dolado i Guivernau, R.: Flocking behaviour: Agent-based simulation and hierarchical leadership. Jasss-The Journal of Artificial Societies and Social Simulation, 2010, vol. 13, num. 2, p. 8 (2010)
22. Reynolds, C.: Boids background and update. http://www.red3d.com/cwr/boids/ (2001)
23. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning. pp. 330–337 (1993)
24. Toner, J., Tu, Y.: Flocks, herds, and schools: A quantitative theory of flocking. Physical review E 58(4). 4828 (1998)
25. Weaver, L., Tao, N.: The optimal reward baseline for gradient-based reinforcement learning. arXiv preprint arXiv:1301.2315 (2013)
26. Wu, J., Liu, Y.: Flocking behaviours of a delayed collective model with local rule and critical neighbourhood situation. Mathematics and Computers in Simulation 179, 238–252 (2021)
27. Yan, C., Xiang, X., Wang, C.: Fixed-wing uavs flocking in continuous spaces: A deep reinforcement learning approach. Robotics and Autonomous Systems 131, 103594 (2020)
28. Yan, C., Xiang, X., Wang, C., Lan, Z.: Flocking and collision avoidance for a dynamic squad of fixed-wing uavs using deep reinforcement learning. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4738–4744. IEEE (2021)
29. Zhu, S., Belardinelli, F., León, B.G.: Evolutionary reinforcement learning for sparse rewards. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1508–1512 (2021)
A Appendix

A.1 The pseudo-code of EMARL

The pseudo-code of EMARL:

Algorithm 1 EMARL Algorithm

| Initialize the decentralized actors population \( P_0 = \{\pi_1, \cdots, \pi_n\} \), centralized critic \( Q \) with parameters: \( \theta^{\pi_1}, \theta^Q \) respectively.
| Initialize the empty cyclic replay buffer of \( \mathcal{R} = \bigcup_{i=1}^{n} \mathcal{R}_i \), \( \mathcal{R}_i \) is ith agent’s buffer.

Input: Senior agent number: \( n_S \); Junior agent number: \( n_J \)

for generation = 1 : \( \infty \) do

for \( \pi_i \in P_k \) do

fitness\(_i\) = 0

for \( j = 0 : \xi \) do

\( r_{i,t+j} = R_{\text{sep},i}(t+j) + R_{\text{ali},i}(t+j) + R_{\text{coh},i}(t+j) + R_{\text{obs},i}(t+j) \)

fitness\(_i\) ← fitness\(_i\) + \( r_{i,t+j} \)

end for

Append \( \mathcal{R}_i \) to \( \mathcal{R} \)

end for

\( \Theta_J = \{\} \), \( \Theta_S = \{\} \)

Rank the actors population by fitness value

Select first \( n_S \) actors’ parameters append to senior agents set \( \Theta_S \), the last \( n_J \) actors’ parameters append to junior agents set \( \Theta_J \), and write its corresponding agents’ set as \( P_S, P_J \)

\( P_k \leftarrow \text{Evolutionary Selection}(P_S, P_J, P_k) \)

Randomly sample a minibatch of \( T \) transitions \( (o_i, u_i, r_i, o_{i+1}) \) from \( \mathcal{R} \)

Compute \( y_i = r_i + \gamma \cdot Q(s_{i+1}|\theta^Q) \)

Update \( Q \) by minimizing the loss: \( \mathcal{L}(\theta^Q) = \frac{1}{T} \sum_{i=1}^{T} (y_i - Q(s_i, u_i|\theta^Q))^2 \)

for \( \pi_j \in P_k \) do

Sample history \( \tau_j \) from \( \mathcal{R}_j \) and update \( \pi_j \) by following decentralized policy gradient:

\( \nabla_{\theta^Q} J \sim \frac{1}{T} \sum_{t=1}^{T} \nabla_{\theta^Q} \log \pi_j(u_j|\tau_j, \theta^\pi_j)(Q(s_j, u|\theta^Q) - \sum_u Q(s_j, u|\theta^Q)) \)

Soft update actor network: \( \theta^\pi_j \leftarrow \alpha \theta^\pi_j + (1 - \alpha) \theta^\pi_j' \)

end for

Soft update critic network: \( \theta^Q \leftarrow \alpha \theta^Q + (1 - \alpha) \theta^Q' \)

end for

The pseudo-code of evolutionary selection:
Algorithm 2 Evolutionary Selection

procedure Evolutionary Selection($P_S$, $P_J$, $P_k$)
    $P_k = P_k \setminus P_j$
    for $\pi_{J,i} \in P_J$ do
        Generate a random number $r_n \sim U[0,1]$
        mutate rate $i = 1 - \frac{\text{fitness}_{J,i}}{\sum_{k=1}^{nS} \text{fitness}_{S,k}}$
        if $r_n < \text{mutate rate}_i$ then
            $\theta_{\pi_{J,i}} \leftarrow \text{random select a parameter from } \Theta_S$
        end if
    end for
    $P_k = P_k \cup P_J$
    return $P_k$
end procedure

A.2 Proof of lemma \[1\]

Proof. Using the product form joint policy to replace $\pi$, and expanding the advantage function:

$$g_{\theta_k} = \mathbb{E}_{\pi} \left[ \sum_{j=1}^{n} \nabla_{\theta_k} \log \pi_j (u_j | \tau_j) \left( Q^{\pi_j} (s, u) - V^{\pi_j} (s) \right) \right]$$  \(14\)

$$g_{\theta_k} = \mathbb{E}_{\pi} \left[ \sum_{j=1}^{n} \nabla_{\theta_k} \log \pi_j (u_j | \tau_j) Q^{\pi_j} (s, u) \right] - \mathbb{E}_{\pi} \left[ \sum_{j=1}^{n} \nabla_{\theta_k} \log \pi_j (u_j | \tau_j) V^{\pi_j} (s) \right]$$  \(15\)

We expand the right part of $g_{\theta_k}$:

$$g_{\theta_k} = \mathbb{E}_{\pi} \left[ \sum_{j=1}^{n} \nabla_{\theta_k} \log \pi_j (u_j | \tau_j) Q^{\pi_j} (s, u) \right] - \sum_s d^\pi (s) \sum u \pi(u | \tau) \cdot$$

$$\sum_{j=1}^{n} \nabla_{\theta_j} \log \pi_j (u_j | \tau_j) V^{\pi_j} (s)$$

$$= \mathbb{E}_{\pi} \left[ \sum_{j=1}^{n} \nabla_{\theta_k} \log \pi_j (u_j | \tau_j) Q^{\pi_j} (s, u) \right] - \sum_s d^\pi (s) \sum u \sum_{j=1}^{n} \pi(u | \tau) V^{\pi_j} (s) \nabla_{\theta_k} \cdot 1$$

$$= \sum_{j=1}^{n} \mathbb{E}_{\pi} \left[ \nabla_{\theta_k} \log \pi_j (u_j | \tau_j) Q^{\pi_j} (s, u) \right]$$

\(16\)

The above result can illustrate that the joint policy gradient can be divided into the sum of the $n$ independent policy gradient. Then, for the single gradient, \[15\] can guarantee that:

$$\lim_{k \to \infty} \| \mathbb{E}_{\pi} [\nabla_{\theta_k} \log \pi_j (u_j | \tau_j) Q^{\pi_j} (s, u)] \| = 0, \forall j = 1, \ldots, n; \text{w.p.1}$$  \(17\)
Because \( n \) is also infinite, the sum of the gradient is converged to 0 with probability 1.

### A.3 Proof of lemma 2

**Proof.** In the \( k \)th update, the inherit probability of \( \pi_{J,i} \) is:

\[
p^k_J = 1 - \frac{\text{fitness}_{J,i}}{(1/n_S) \sum_{k=1}^{n_S} \text{fitness}_{S,k}}
\]  

Due to the junior & senior agents are selected according to the rank of fitness value, which means:

\[
\max_i \{ \text{fitness}_{J,i} \} \leq \min_j \{ \text{fitness}_{S,j} \}
\]  

So we have:

\[
\text{fitness}_{J,i} \leq \frac{1}{n_S} \sum_{k=1}^{n_S} \text{fitness}_{S,k}
\]  

Therefore, the \( p^k_J \in [0, 1] \). After \( m \)th update, the inherit probability of \( \pi_{J,i} \) is:

\[
P^m = 1 - \prod_{k=1}^{m} (1 - p^k_J)
\]  

Easy to see, there \( \exists N, s.t. n \geq N \), the \( P^m \) equal to 1, or :

\[
\lim_{n \to \infty} 1 - \prod_{k=1}^{n} (1 - p^k_J) = 1
\]

### A.4 Values of Parameters

Table 1: The values of mainly parameters in our experiment are shown as follows.

| Parameters | Value |
|------------|-------|
| \( \beta_{sep} \) | 40    |
| \( \beta_{obs} \) | 40    |
| \( \beta_{coh} \) | 40    |
| \( 2|A| + 1 \) | 5     |
| \( ||\vec{v}|| \) | 0.0125|
| \( ||\omega|| \) | 0.0225|
| \( \gamma \) | 0.98  |
| \( \xi \) | 20    |