SwarmAI

Seunghwan Song, TaeHyung Kim, Sanghyun Lee
School of computing
KAIST

Abstract

The simulation of predator-prey interactions is critical in community ecology for understanding swarm behavior and adaptive strategies. In existing research, the training was conducted by directly rewarding the presence of fish nearby to induce schooling behavior. However, in this study, by utilized Multi-Agent Deep Deterministic Policy Gradient (MADDPG), we were able to demonstrate schooling behavior even in situations where there is no direct reward for schooling but rather a penalty for being eaten by sharks. Our experiments involved varying grid sizes to test the model's robustness. The study highlights the potential of MARL in modeling complex behaviors without extensive rule-based systems, offering insights into decentralized intelligence and applications in various real-world scenarios.

1 Introduction

The simulation of complex systems requiring interactions, such as predator-prey ecosystems, is a critical issue in community ecology. For instance, the classical Lotka-Volterra model uses differential equations to represent the relationship between predators and prey[1]. Models that can be represented by such equations do not account for the individual intelligence of organisms, and therefore cannot accurately describe real ecosystems. In reality, organisms learn to avoid predators and sometimes exhibit swarm behavior[2, 3]. To enable such behaviors in organisms, well-designed simulations are required. In this study, we utilize reinforcement learning to train agents and present a simulation involving sharks and fish. Through this research, we aim to observe swarm behavior and adaptations for survival.

With the advancement of deep learning, reinforcement learning has demonstrated success in various fields[4–8]. Researchers aim to address problems in more complex environments through reinforcement learning, where interactions among various agents occur. To model such environments, they employ multi-agent reinforcement learning (MARL)[9]. MARL systems can simulate real-world scenarios such as competitive ecosystems, cooperative tasks, and dynamic decision-making processes[10–14]. This approach enhances our understanding of decentralized intelligence in various applications. However, existing research creates complex rules to handle agents, which hinders the diversity of agent behaviors. Therefore, our research conducts MARL using a maximally simplified reward system.

The remainder of this paper is organized as follows. We explain the framework of the shark-fish model in Section 2. Then in Section 3, we explain the results of the experiments. Finally, discussions and conclusions are provided in Section 4 and 5.

2 Methods

For this experiment, we consider individual agents moving at constant speed in a two-dimensional area with periodic boundary conditions with a predator, a shark. The minimum unit 'block' is 20 pixels, which is the size of a fish, shark, and a movement. Agents, called fish, are initially spawned randomly and get input **state** of the shark's direction and number of fishes in each direction: up, down, left, right. Then, for each step, the agent chooses to go one block up, down, left, or right according to their policy (**action space**). If other fish are blocking the way, it cancels the movement, which will make the fish gather. After the fish's move, the shark approaches the closest prey if its measured size is smaller than its size. If a shark touches a fish, it is considered eaten, after which the fish respawns in a random position.

2.1 Environment

Environment space is a 2D-pixel space with periodic boundary. The environment contains the size of the whole map, all the information about the shark, and reward criteria. The reward is given -1 each time a fish is caught and eaten by the shark.

2.2 Fish

Each fish agent will have its own individual policy network and critic network. Each of these networks will have a structure similar to the MADDPG[15] networks. However, the environment involves a discrete action space instead of the continuous action space typically addressed by Deep Deterministic Policy Gradient(DDPG)[16]. To address this issue, we utilized the softmax function on the output of the actor network to create a probability distribution, from which actions were sampled. When calculating the policy gradient, we used the result of the softmax as the input to the critic network to address the problem of gradient vanishing. The algorithm for fish agents are shown in Algorithm 1.

Algorithm 1 Discrete Action Multi-Agent Actor Critic for N agents

```
 1: for episode $= 1$ to $M$ do
 2:     Receive initial state $x$
 3:     for $t = 1$ to max-episode-length do
 4:         for each agent $i$ do
 5:             Select action $a_i \sim \mu_{\theta_i}(o_i)$ w.r.t. the current policy and exploration
 6:         end for
 7:         Execute actions $\mathbf{a} = (a_1, \ldots, a_N)$ and observe reward $r$ and new state $x'$
 8:         Store $(x, a, r, x')$ in replay buffer $\mathcal{D}$
 9:         $x \leftarrow x'$
10:     end for
11:     for agent $i = 1$ to $N$ do
12:         Sample a random minibatch of $S$ samples $(x^j, a^j, r^j, x'^j)$ from $\mathcal{D}$
13:         Set $y^j = r_i^j + \gamma Q^i_{\theta_i'}(x', a_1, \ldots, a_N)\big|_{a_k = \mu_{\theta_k}(o_k)}$
14:         Update critic by minimizing the loss
                $$\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q^i_{\theta_i}(x^j, a_1^j, \ldots, a_N^j)\right)^2$$
15:         Update actor using the sampled policy gradient
                $$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\mu_{\theta_i}(o_i^j)\nabla_{a_i} Q^i_{\theta_i}(x^j, a_1^j, \ldots, a_N^j)\big|_{a_i = \mu_{\theta_i}(o_i^j)}$$
16:     end for
17:     Update target network parameters for each agent $i$:
                $$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$
18: end for
```

2.3 Predator

There is a predator, a shark, in the environment which hopefully forces the agents to school. The shark first checks whether the current target fish is alive, and is not forming a swarm size bigger than its threshold. If the swarm size is small, the shark moves toward the fish two times faster than the fish's speed. Otherwise, the target is reset after some time step. The algorithm for measuring the fish's swarm size is shown in Algorithm 2. WIDTH, HEIGHT, and SWARM RADIUS are hyper parameters for the size of the sea and a radius for detecting a swarm near the target fish.

Algorithm 2 Measuring the swarm size of a fish by the predator

```
 1: Get the target fish and its x,y coordinates $x$, $y$
 2: Initialize the $swarmsize = 0$
 3: for each $fish$ do
 4:     if $fish$ is dead or $fish$ is current target fish then
 5:         continue
 6:     else
 7:         Take care of periodic boundary as follows
 8:         $dx = min(abs(x - fish.x), WIDTH - abs(x - fish.x))$
 9:         $dy = min(abs(y - fish.y), HEIGHT - abs(y - fish.y))$
10:         $distance = sqrt(dx * *2 + dy * *2)$
11:         if distance <= SWARM RADIUS then
12:             $swarmsize$ += 1
13:         end if
14:     end if
15: end for
16: return $swarmsize$ + 1
```

2.4 Baseline agents

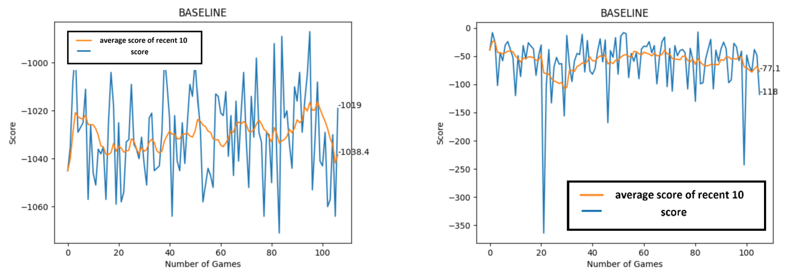
There is a baseline for comparing the scores plot with the MADDPG model. 'Random' and 'One Direction' are the simplest policies that can arise that does not consider other fish or shark's position.

2.4.1 Baseline agents: random policy

Each fish agents move in one of four directions randomly at each step. Since each action is random and does not even avoid a shark, its performance is expected to be very poor.

2.4.2 Baseline agents: one direction policy

Each fish agent moves in a fixed direction of up, down, left, or right throughout a whole episode. However, this naive implementation can make the fish school since the movement of the fish is canceled on collision among fishes. Hence we want to see whether our model performs better than going in a randomly chosen fixed direction, using this baseline. As seen in below result, there are sharp drops of score per episode for this policy, which means it is unstable and it relies on luck.



(a) Random policy (b) One direction policy

Figure 1: Reward plot for each episode with 600x600 box, 32 fishes

3 Experiment

3.1 Setup

The experiment simulates a dynamic environment where multiple fish must avoid a predatory shark by forming cohesive groups. The environment is implemented using `pygame`, and the behaviors of both fish and shark are governed by reinforcement learning algorithms, specifically the Multi-Agent Deep Deterministic Policy Gradient (MADDPG).

Three different game environments are defined, each with varying sizes and numbers of fish:

200x200 grid: 4 fish, shark size 4.

400x400 grid: 16 fish, shark size 6.

600x600 grid: 36 fish, shark size 10.

3.2 Results

The results of the simulation show that fish can learn to form effective groups to evade the shark. The training progress is visualized through plotted scores (Table 1), demonstrating improvements in the fish's survival time.

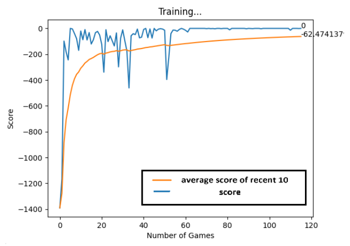
When testing different environmental variations, we observed the following results (Figure 2):

Grid Size and Number of Fish: Increasing the grid size and the number of fish generally led to more complex group behaviors and required longer training times. The fish were able to maintain group cohesion effectively even in larger environments.

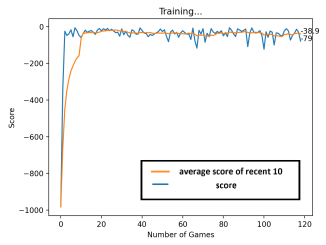
Swarm Size Criteria for Shark: Adjusting the shark's size threshold for determining if a group is too large to attack showed that smaller thresholds led to more aggressive shark behavior, disrupting fish groups more frequently. Larger thresholds allowed fish to form larger groups, which were more effective at evading the shark.

Individual Fish Behavior: When the number of fish exceeded the minimum required for the shark to consider them a swarm, some fish remained alone and did not join the groups. These isolated fish tended to move in a single direction, making them more vulnerable to being targeted by the shark.

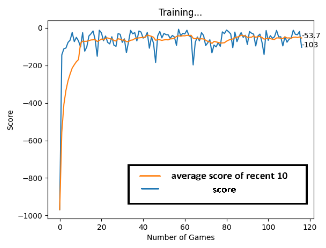
Overall, the model demonstrated strong adaptability and robustness across different settings, maintaining effective group behavior and efficient predator-prey interactions.



(a) 200x200 grid



(b) 400x400 grid



(c) 600x600 grid

Figure 2: Training progress for different grid sizes.

Grid Size	Number of Games	Convergence Score	Shark Size
200x200	120	0	4
400x400	120	-38.9	6
600x600	120	-53.7	10

Table 1: Statistical results of the training process, showing the number of games to convergence and the corresponding score.

4 Discussion

Our experiments demonstrated that fish can learn to form effective groups to evade a predatory shark, and the shark can adapt to disrupt these groups to capture individual fish. The results showed that the model was robust across various environmental conditions, including different grid sizes.

Environmental Variations and Robustness: The model's ability to maintain effective group behavior across different environmental conditions highlights its robustness. Increasing the grid size and number of fish led to more complex group behaviors.

Behavioral Observations: The observations indicated that fish tend to form cohesive groups to minimize the risk of being targeted by the shark. Additionally, when the number of fish exceeded the minimum required for the shark to consider them a swarm, some fish remained alone and moved in a single direction, making them more vulnerable to being targeted.

5 Conclusion

In conclusion, our study demonstrated the potential of MADDPG in simulating complex behaviors in a predator-prey ecosystem, providing valuable insights into swarm intelligence and adaptive behaviors. Future research will build on these findings to further enhance the understanding and application of MARL in diverse fields.

Limitations and Future Work: While the grid-based environment provided a simplified representation of predator-prey interactions, it may oversimplify the complexity of real-world ecosystems. The reward structure, designed to promote specific behaviors, may not capture all aspects of natural interactions. Computational constraints also limited the size and complexity of the simulations. Future work will address these limitations by exploring more complex and realistic environments, refining the model to minimize human intervention, and incorporating more sophisticated behavioral dynamics.

6 Contributions

TaeHyung Kim: Implemented shark logic, set appropriate variables (number of fish and shark size per grid size).

Sanghyun Lee: Make the algorithm for fish agent(Make discrete actor critic algorithm).Making softmax for discrete action. Finding paper realated with project.

Seunghwan Song: Proposing and modeling the fish-shark environment framework (state, action space etc) using pygame

References

[1] Peter J Wangersky. Lotka-volterra population models. *Annual Review of Ecology and Systematics*, 9(1):189–218, 1978.

[2] Randal S Olson, David B Knoester, and Christoph Adami. Evolution of swarming behavior is shaped by how predators attack. *Artificial life*, 22(3):299–318, 2016.

[3] Jonathan M Jeschke and Ralph Tollrian. Prey swarming: which predators become confused and why? *Animal Behaviour*, 74(3):387–393, 2007.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[5] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[6] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[7] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[8] Rongrong Liu, Florent Nageotte, Philippe Zanne, Michel de Mathelin, and Birgitta Dresp-Langley. Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review. *Robotics*, 10(1):22, 2021.

[9] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[10] Mihir Durve, Fernando Peruani, and Antonio Celani. Learning to flock through reinforcement. *Physical Review E*, 102(1):012601, 2020.

[11] Jun Yamada, John Shawe-Taylor, and Zafeirios Fountas. Evolution of a complex predator-prey ecosystem on large-scale multi-agent deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[12] Fabian Ritz, Daniel Ratke, Thomy Phan, Lenz Belzner, and Claudia Linnhoff-Popien. A sustainable ecosystem through emergent cooperation in multi-agent reinforcement learning. In *Artificial Life Conference Proceedings 33*, volume 2021, page 74. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2021.

[13] Xueting Wang, Jun Cheng, and Lei Wang. A reinforcement learning-based predator-prey model. *Ecological complexity*, 42:100815, 2020.

[14] Jianan Li, Liang Li, and Shiyu Zhao. Predator–prey survival pressure is sufficient to evolve swarming behaviors. *New Journal of Physics*, 25(9):092001, 2023.

[15] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[16] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.