Multi-Agent Learning In a Prey-Predator Environment

İhsan Doğramacı Bilkent University

CS-461: Artificial Intelligence

Final Report

Project Team: 09

Team Members:

Tuğberk Dikmen, İdil Karagöz, Hasan Yarkın Kurt, Egemen Öztürk, Emre Uğur

Abstract

In this paper, we address the problem of multi-agent learning in a prey-predator environment. Initially, we experimented with evolutionary algorithms; however, we found their sample efficiency to be too low. Subsequently, we explored more sample-efficient reinforcement learning algorithms, such as Deep Deterministic Policy Gradients (DDPG) and its multi-agent counterpart, Multi-Agent Deep Deterministic Policy Gradients (MADDPG). We also experimented with the concept of a single shared critic, which we refer to as Shared Critic Deep Deterministic Policy Gradients (DDPG). This paper outlines the performance of these three algorithms in a prey-predator setup. Additionally, we conducted experiments in a fully cooperative environment, which highlighted the differences among the three algorithms more effectively.

I. Introduction

In our project we aimed to simulate a complex prey-predator environment. Our goal was to utilize artificial intelligence in a setting where multiple agents interact, learn, and evolve. This paper covers advanced techniques in deep reinforcement learning, specifically focusing on Multi-Agent Deep Deterministic Policy Gradient (MADDPG), Deep Deterministic Policy Gradient (DDPG), Shared Critic Deterministic Policy Gradient (SCDDPG) and Neuro Evolution of Augmenting Topologies (NEAT). We aimed to observe and analyze how these artificially intelligent agents adapt, learn and collaborate in a dynamic and competitive environment that challenges them with limited vision and partial observability. This report presents our final achievements in detail to understand and develop an artificial intelligence system capable of demonstrating the relationship between prey and predator agents.

II. Background

Markov Decision Process is a framework for mathematically modeling a decision-making process. It is used in economics, game theory, and artificial intelligence[1]. The components of an MDP are: States S represent the different states a system can be at any given point. Actions A: the action space of an agent; these are the set of all the actions available for an agent. Transitions $T(s, a, s')$: probabilities associated with transitioning to state S' from S with the action A. Rewards $R(s, a, s')$: rewards related to the transition s to s' with action a. Policy $P(s, s)$ is the policy that the agent follows while taking the actions. It gives the probability distribution of the action space in a given state. A deterministic policy will go with the most probable action, which is $argmax_a Policy(a|s)$[2].

Markov Games are the multi-agent extension of the Markov decision process[1] since an MDP deals with the decision of a single agent. A Markov Game for N agents is defined by a set of states S; in our prey-predator case, the position and the velocities of the agents make the state of the system $S = \{Pos_1 \dots Pos_n, Vel_1 \dots Vel_n\}$. A set of action spaces $A_1 \dots A_n$ and, in the case of partially observable Markov games, a set of observations $O_1 \dots O_n$. An observation is a limited portion of the state locally visible by an agent. An agent i tries to maximize its own expected sum of rewards $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where γ the discount factor and r are the rewards associated with the transition $s_t \, to \, s_{t+1}$ with action a[1].[2]

Reinforcement Learning is a type of machine learning concerned with how an agent should act in an environment to maximize its total reward[3]. It is instrumental in situations where it's not feasible to gather data that covers all the cases for an agent to take[4]. It is a process of trial and error. While interacting with the environment, agents learn to

give higher probability to specific actions in certain states through trial and error. Some related concepts in RL that will be useful in this paper:

Q value: q-values are defined as $Q(s, a)$ for a state s and action a that gives the value of taking the action a in the state s. A higher value indicates a good action. In the context of deep RL, a neural network can be fitted to the q-function $Q(S, A)$ since it is usually impractical to represent it as a table due to the sheer scale of the resulting table.

Policy is defined as $\Pi(a|s)$. It is the optimum action given a state. Similarly to the case with q-functions, a neural network can be used to model this function since a lookup table will be huge for many environments. Therefore, it can be defined as $\Pi_\theta(a|s)$ the policy parameterized with θ. θ These are the parameters of the neural network to model the policy; these parameters are the weights and biases of the network.[3]

Policy Gradient Algorithms are a popular choice for a reinforcement learning task. Policy gradient algorithms modify the policy parameters to maximize the objective function $J(\theta)$[1]. The objective function can be used to observe the quality of the policy that is parameterized θ. It is defined as $J(\theta) = E_{s\sim p,\, a\sim\Pi_\theta} Q(s,a)$. The idea is to take small steps in the direction of $\nabla_\theta J(\theta)$ if we imagine θ as the input domain and the J(θ) as the output. We want to find the maxima of this surface. The gradient of $J(\theta)$ can be written as:

$$\nabla_\theta J(\theta) = E_{s\sim p_\Pi, a\sim\Pi_\theta}[\nabla_\theta log\, \Pi_\theta(a|s) Q_\Pi(s,a)]$$

Policy gradient algorithms primarily differ in how the Q function is defined[1]. In the simplest case, it can be defined as the sum of rewards on a trajectory. $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, which gives rise to the fundamental REINFORCE algorithm. However, REINFORCE suffers from high variance, which leads to unstable learning. Q function may be directly fitted with a neural network, which is the main idea behind a family of algorithms called actor-critic algorithms.

Deep Deterministic Policy Gradient (DDPG) algorithms are the extension of the policy gradient algorithms to work with deterministic policies μ : S → A [i] . The gradient of $J(\theta)$ can be written as

$$\nabla_\theta J(\theta) = E_{s\sim D}[\nabla_\theta \mu_\theta(a|s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}]$$

Multi-agent deep Deterministic Policy Gradient (MADDPG) is an algorithm proposed by OpenAI to solve multi-agent problems with a DDPG approach. The main idea of MADDPG is that the critic network Q has access to central information. That is for an environment with N agents. The critic network for agent i is $Q_i(s_1 \ldots s_n, a_1 \ldots a_n)$. However, the actor-network still has access to only local information. For agent i, the actor-network is $\Pi_i(s_{,i}, a_i)$. This centralized critic is fitted per agent to enable learning in a mixed cooperative-competitive environment.

Shared Critic Deep Deterministic Policy Gradients (SCDDPG). We mentioned that in MADDPG, there is a per-agent critic network. We wondered what would happen with a single shared critic for all agents. This idea is demonstrated in the paper[13]. Although there isn't an RL algorithm named SCDDPG we will refer to the shared critic counterpart of MADDPG as as SCDDP in this paper. This algorithm is generally applicable in full cooperative environments, but we managed to use it in the prey-predator environment by separating the critics of the competing agents.

III. Literature Review

The Topic of prey versus predator environments is a well-researched domain with many simulations and environments that emphasize different aspects of artificial intelligence. During our literature review for the project, we analyzed many different videos, tutorials and repositories. In this section of the report, our motive is to provide the pros and cons of these algorithms and environments. Additionally we provide our decision process behind choosing one over the other.

A. Evolutionary algorithms

Evolutionary algorithms represent a powerful computational approach inspired by the principles of natural selection and genetic evolution. When applied to prey versus predator environments, these algorithms simulate the dynamic interplay between species, capturing the essence of survival and adaptation. In such scenarios, prey species evolve strategies for evasion, camouflage, and efficient resource utilization. On the other hand, predators undergo continuous adaptation to enhance their hunting techniques, intelligence, and stamina. Evolutionary algorithms in this context enable the modeling of genetic variation, selection, and reproduction within populations, mirroring the biological processes that shape species over time. These algorithms find applications in optimizing strategies for prey and predator, offering insights into cooperation and competition amongst agents for understanding complex ecological dynamics and co-evolutionary processes.

Bibites environment[4] is one example of such environments utilizing evolutionary algorithms. In this environment NeuroEvolution of Augmenting Topologies(NEAT) [5] is chosen as the evolutionary algorithm. This algorithm allows for the evolution of structure (topology) of neural networks. Agents adapt to the environment over time by discovering the optimal neural network architecture for the given task.

Our initial project idea was to develop a separate environment and utilize the NEAT algorithm for the agents. However, our environment could not reach the sample generation efficiency required for an evolutionary algorithm so we decided to proceed with more sample-efficient algorithms and more computationally efficient environments.

B. Deep Reinforcement Learning Algorithms

Deep reinforcement learning (DRL) algorithms are powerful tools for modeling and solving complex decision-making problems, making them particularly relevant in prey versus predator environments. In such scenarios, DRL leverages artificial neural networks to represent the decision policies of both prey and predator agents. The prey agents employ DRL to learn adaptive strategies for evasion, resource gathering, and survival, while the predator agents utilize the same framework to develop effective hunting strategies, spatial awareness, and decision-making capabilities. DRL allows these agents to learn from interactions with the environment, adjusting their behaviors over time based on rewards and penalties. The dynamic nature of prey versus predator interactions demands a sophisticated understanding of the environment, and DRL excels in capturing and learning intricate patterns within these complex systems. This technology enables the simulation of strategic evolution in the virtual realm, offering insights into emergent behaviors, coevolution, and the optimization of survival strategies in predator-prey dynamics.

One example of DRL algorithms used in prey vs predator environments is in [6] where the Multi agent particle environment(MPE)[7] is used with the Multi Actor Deep Deterministic Policy Gradient(MADDPG)[8] algorithm. This environment is much simpler where predator agents try to catch prey agents, further details of this environment will be discussed in Section IV (Experiment). Simplicity of the environments allow for better performance of the algorithms where agents learning is much faster. We decided to use this environment and compare different DRL algorithms so that we can observe the differences in agents' learning.

IV. Methodology

A. Environment

1. Initial Environment

We initially implemented our own prey-predator environment using the Pymunk physics engine and the Pygame graphics library. We made the environment complex to enable complex behavior to emerge.

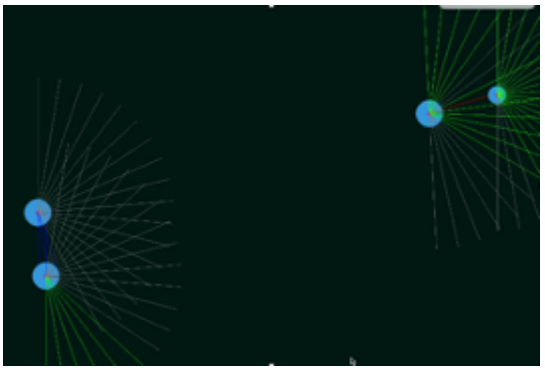

Fig. 1: Custom Prey-Predator Environment

In this environment, entities possessed a 180-degree field of vision denoted by lines. Predators, distinguished by their larger size, coexisted with smaller prey entities. The ability to rotate around themselves enabled entities to survey their environment. Despite the intricacy of this physics-based environment, the training process proved to be considerably time-consuming compared to the later environment utilized in subsequent iterations.

2. PettingZoo MPE Environment

PettingZoo MPE environment, which is based on the OpenAI MPE (Multiagent Particle Environment), is used for the training. In this environment, a slower cooperating adversary (Predator) is chasing a faster agent. There are landmarks and obstacles in the environment that can be used by the agent to get away from the predators.

Simple Tag:

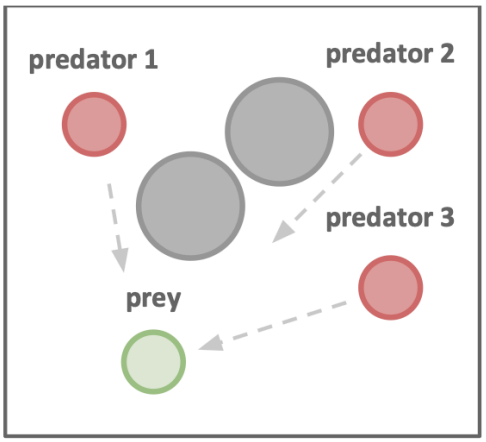

Fig. 2: Prey-Predator Environment [8]

Simple tag is a simple prey-predator environment. If the predators catch the prey agent, all of them get rewards (+10), which creates collaboration. Meanwhile, the agent gets a negative reward (-10), reinforcing the escaping behavior. If entities go into the obstacles or leave the area, they get negative rewards, forcing them to stay in the allowed spaces.

Simple Spread:

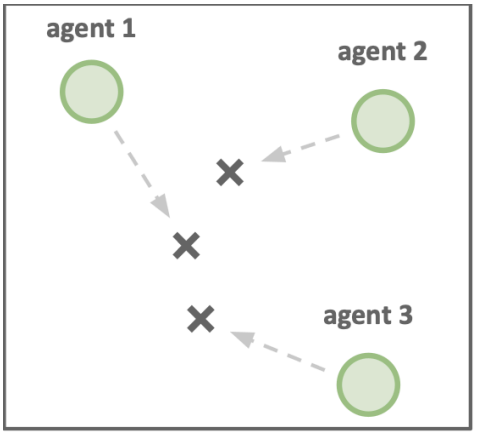

Fig. 3: Simple Spread Environment [8]

In the simple Spread Environment, there are N agents (indicated by blue circles) and N landmarks (indicated by black circles). The overarching goal is for the agents to cover all the landmarks while avoiding collisions collectively. The learning goal involves incentivizing agents globally by rewarding them based on the summed minimum distances between each agent and the landmarks. At a local level, agents receive penalties for collisions with one another, with a deduction of -1 for each collision. This dual-reward structure challenges agents to balance landmark coverage with the imperative to navigate the space without collisions cooperatively. The environment encourages the development of strategies that optimize both global and local interactions, making it an interesting setting for exploring multi-agent reinforcement learning.[9]

B Experiments

1. Simple Tag Experiments

The "Simple Tag" experiment introduces a classic prey-predator dynamic. In this setup, we have multiple predator agents whose objective is to chase and tag the prey agents within a defined environment. This experiment focuses on the ability of predator agents to develop chasing strategies and the prey's capability to escape effectively. It tests the agents' learning capabilities in terms of collaborative work and the use of environmental features.

a) Selection of Parameters Based on OpenAI Paper

1. MADDPG Parameters from OpenAI
We aligned our parameter choices for MADDPG with those recommended in OpenAI's repository. This included settings for learning rates, buffer capacity, and the soft update parameter (tau), ensuring our simulation closely follows the architecture proven effective in OpenAI's research. The rationale behind these choices is grounded in the success of these parameters in similar complex multi-agent environments studied by OpenAI. [10]

2. DDPG Parameters from OpenAI
The DDPG algorithm parameters are kept unchanged from the MADDPG training parameters. Our DDPG implementation is a modification of the MADDPG algorithm. We decentralized the critics of agents to get the DDPG algorithm. Other than that the two algorithms were kept identical to observe the effect of this centralized critic idea. [11]

3. To implement the SCDDPG algorithm we assigned the same critic to cooperating agents. For example, in a 3 predator 1 prey environment we used a total of 2 critics. 1 for the predators and 1 for the prey. Apart from this change, we kapt the parameters and the policy gradient calculations of the algorithm identical with MADDPG.

The choices for episode length, learning interval, and gamma (discount factor) are particularly aligned with OpenAI's guidelines, reflecting an emphasis on stable and efficient learning progressions in single-agent contexts.

b) Integration with Our Experiment Design

Episode Numbers (30000, 50000, 100000, 200000): While the exact numbers were chosen to extend beyond typical OpenAI scenarios, they follow the principle of sufficient training episodes to allow for deep learning and strategy development in complex environments.

Episode Length (25 steps): The fixed episode length ensured a consistent framework for each simulation run, allowing for a standardized comparison across different scenarios and algorithms.

Random Steps (50000) & Actor/Critic Learning Rates (0.01): These were selected to maintain a robust exploration-exploitation balance and steady learning progression, which are crucial elements highlighted in OpenAI's research.

Batch Size (1024) & Buffer Capacity (1000000): Following OpenAI's insights, a large batch size and buffer capacity were chosen to enable the algorithms to learn from a diverse set of experiences, enhancing the robustness of the learned strategies.

Learn Interval (100): This interval was chosen to balance frequent updates for rapid learning and sparse updates to avoid overfitting.

Tau (0.02): A relatively low tau value for soft updates ensures stable, incremental learning, avoiding drastic policy shifts that could destabilize the learning process.

Gamma (0.95): This discount factor balances the importance of immediate and future rewards, a crucial aspect in environments where long-term strategies can significantly impact outcomes.

We expect that the strategies and efficiencies developed by the agents in our simulation will align with the outcomes observed in OpenAI's studies. For example, we expect MADDPG to facilitate sophisticated cooperative strategies in multi-agent settings, as seen in OpenAI's research.

c) Agent Configurations

1. 3 Predators, 1 Prey, 2 Obstacles:

Expectation: The primary expectation here is for the predators to demonstrate collaborative strategies as they are in the majority against a single prey. The obstacles further add complexity, requiring predators to coordinate their movements and strategies efficiently.

Algorithm Performance: This setup provides an ideal scenario to test the cooperative learning

capabilities of MADDPG and SCDDPG. DDPG will offer insights into how individual predators adapt in a team setting.

2. 1 Predator, 3 Preys, 2 Obstacles:

Expectation: In this scenario, the lone predator is expected to employ a more opportunistic approach, potentially targeting the closest prey. The challenge here lies in the predator's ability to navigate and strategize effectively against multiple prey.

Algorithm Performance: This configuration will particularly test the adaptability of the DDPG algorithm in managing multiple targets. MADDPG and SCDDPG's performance will be indicative of how these algorithms handle outnumbered situations.

3. Predators, 3 Preys, 2 Obstacles:

Expectation: A balanced number of predators and prey is expected to result in a mix of collaborative and solo strategies. This dynamic environment will likely showcase instances where agents choose to act independently or in sync with others based on the evolving situation.

Algorithm Performance: This balanced setup will be crucial in evaluating the flexibility of all three algorithms in adapting to varying group dynamics and changing environmental conditions.

4. Simple World (2 Predators, 1 Leader Predator, 2 Preys, 2 Baits, 2 Obstacles, 2 Forests):

Expectation: This is the most complex and advanced scenario, combining elements of leadership, baiting strategies, and environmental features like forests for hiding. We anticipate a rich blend of collaborative tactics, individual decision-making, and strategic use of environmental elements by both predators and prey.

Algorithm Performance: This scenario will test the upper limits of the algorithms' ability to handle complex strategies and environmental interactions. The role of the leader predator introduces an additional layer of hierarchy and strategy that could lead to more advanced collaborative behaviors.

All these scenarios will be run for each of the three algorithms (MADDPG, DDPG, SCDDPG) across a range of episode numbers (30,000, 50,000, 100,000, 200,000). The varying episode numbers will allow us to observe how the learning and strategy development evolve over time and whether longer training leads to more sophisticated and efficient behaviors.

d) Expected Results

We hypothesize that the learning performance and strategy sophistication will vary significantly across different agent configurations and algorithms. We anticipate that:

MADDPG and SCDDPG will outperform DDPG in scenarios with multiple interacting agents due to their ability to learn cooperative strategies. In scenarios with more predators than preys, we expect to see emergent collaborative hunting strategies. Conversely, with more prey than predators, we anticipate strategies involving evasion and opportunistic hunting. The most complex scenario (Experiment 4) is expected to demonstrate the highest level of strategy sophistication, including deceptive tactics and leadership roles.

In conclusion, our simulation design aims to provide a comprehensive understanding of how different reinforcement learning algorithms perform in varied and complex environments. Through this study, we hope to gain valuable insights into the dynamics of predator-prey interactions and the capabilities of current AI techniques in simulating such complex systems.

2. Simple Spread Experiments

a) Experiment Description

In the 'Simple Spread' experiment, we introduce a scenario that is distinct from the typical predator-prey dynamics. In this setup, we have three agents, each tasked with reaching one of three specified locations, while avoiding collisions with each other. This setup emphasizes navigation skills and collision avoidance within a multi-agent environment. We chose this environment for

demonstration purposes, as we found that its purely cooperative nature more effectively highlights the differences between the algorithms.

b) Parameters and Algorithms

The parameters for this experiment remain consistent with those used in the predator-prey scenarios. The same algorithms, MADDPG, DDPG, and SCDDPG, are employed in this experiment. The rationale behind using these algorithms is to assess and compare their effectiveness in a non-predatory context.

c) Expected Outcomes

MADDPG and SCDDPG: These algorithms are expected to excel in this scenario due to their capability to handle multi-agent dynamics and cooperative strategies. The focus will be on how agents learn to navigate efficiently to their respective goals without interfering with each other.

DDPG: As a single-agent algorithm, DDPG's performance in this experiment provides insight into individual learning and navigation capabilities in a multi-agent environment.

d) Significance of the Experiment

This "simple spread" experiment adds a significant dimension to our study, contributing to the predator-prey dynamic. It focuses on cooperative movement and spatial strategy in a shared environment. This scenario is significant for understanding multi-agent cooperation in tasks that require spatial coordination and shared goal achievement.

e) Integration with the Overall Study

Incorporating this experiment into our study allows us to create a wider understanding of multi-agent systems. By comparing the results from this experiment with the predator-prey scenarios, we can draw comprehensive conclusions about the versatility and adaptability of the algorithms in different environmental setups and objectives.

V. Results

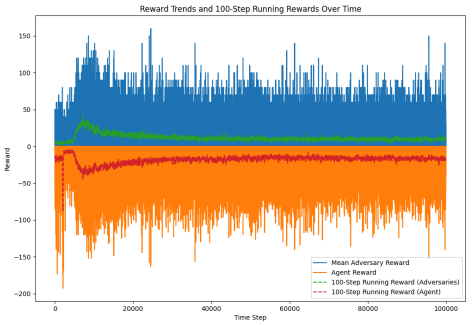

Fig. 4: DDPG Prey-Predator

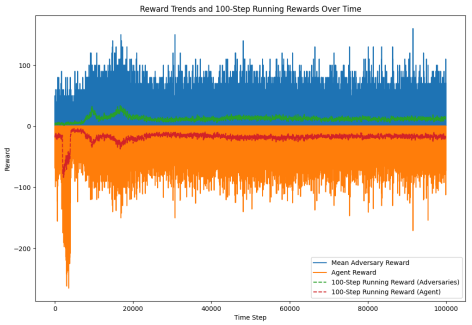

Fig. 5: MADDPG Prey-Predator

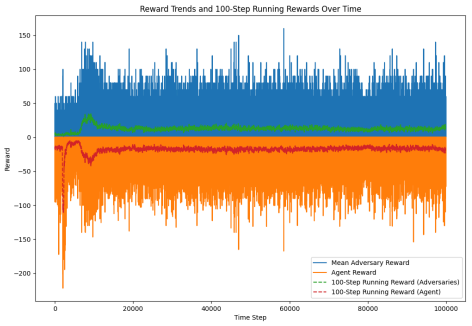

Fig. 6: SCDDPG Prey-Predator

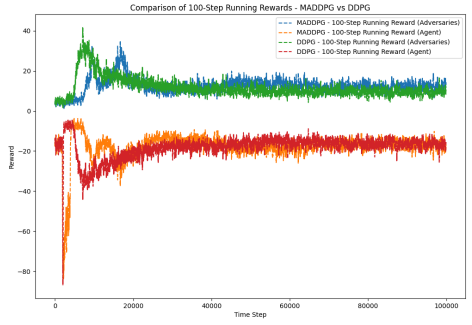

Fig. 7: MADDPG vs DDPG Prey-Predator

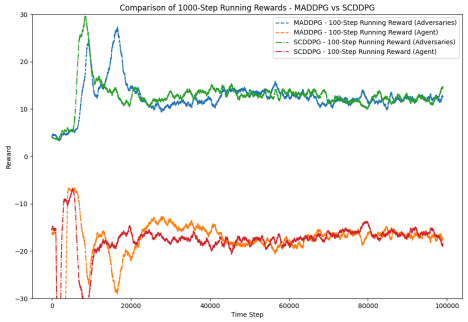

Fig. 8: SCDDPG vs MADDPG Prey-Predator

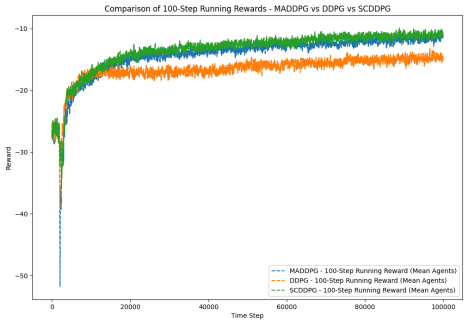

Fig. 9: MADDPG vs DDPG vs SCDDPG Simple-Spread

VI. Conclusion

All three algorithms mentioned (DDPG, MADDPG, SCDDPG) performed well in the predator-prey setup. In the predator-prey environment, we observed this mirroring kind of reward trend where the reward trend of one competing agent mirrors that of the other Fig.4,5,6. This results from the very fundamental nature of the environment: for one agent to increase its rewards, the other agent's rewards must decrease. In our case, we first observed an increase in the predators' rewards and a decrease in the prey's rewards. While the predators are learning to catch the prey, the prey is being caught. After this, we observed that the predators' rewards fell while the prey's rewards increased as the prey learned to escape from the predators. This behavior is observed prominently at the beginning of the training, but it's also repeated several times at different scales. The bumps in the trends are much more minor and have a broader base. We also observed that the converging patterns in the graphs might mislead one into believing that learning has stopped. However, we found that a steady pattern in the reward trends does not conclude finished learning. We observed a significant difference in agents' behavior between 30k and 100k training periods. This difference is visually reflected in our test runs after training, which can be found in the repository[13].

When the results of the three algorithms are compared Fig.7,8, we do not observe any tangible difference among the three. Predators in MADDPG seem to perform slightly better than the predators in DDPG Fig7 since the reward trend of predators for MADDPG is slightly above the reward trend of predators in DDPG. The differences are better reflected in the test runs that are observed visually in the environment. We did not observe any difference between SCDDPG and MADDPG in the predator-prey setting, as the reward trends are practically identical Fig.8. However, we see that a shared critic approach can be applied to the predator-prey environment, which is a mixed cooperative-competitive environment. A shared critic approach is already proven to be a solution in a purely cooperative environment in this paper [14], but we managed to adapt it to a mixed environment by separating the critics of the competing agents. We did not observe any behavioral difference with the agents trained with MADDPG and SCDDPG. However, a shared critic approach will result in better computational efficiency since performing backpropagation on neural networks is usually the performance bottleneck of RL algorithms. SCDDPG can be even more useful in a multi-agent environment where the number of agents is high.

When the three algorithms are compared in the simple spread environment, we observe significant differences among the algorithms since a pure comparative environment emphasizes the multi-agent capabilities of the algorithms better. We observe that the single-agent algorithm DDPG performs noticeably worse than the multi-agent algorithm, which is reflected in the plot (pno). We also see that SCDDPG performs better than MADDPG in the simple spread environment Fig.9.

References

[1] “Markov decision process,” Wikipedia, https://en.wikipedia.org/wiki/Markov_decision_process (accessed Dec. 17, 2023).

[2] ArXiv:1706.02275v4 [cs.LG] 14 Mar 2020, https://arxiv.org/pdf/1706.02275.pdf (accessed Dec. 17, 2023).

[3] “Reinforcement learning,” Wikipedia, https://en.wikipedia.org/wiki/Reinforcement_learning (accessed Dec. 17, 2023).

[4]“Bibites Community Shared Content,” *GitHub*, Nov. 11, 2023. https://github.com/TheBibites/Bibites_Shared_Content (accessed Dec. 17, 2023).

[5]K. O. Stanley and R. Miikkulainen, “Evolving Neural Networks through Augmenting Topologies,” *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, Jun. 2002, doi: https://doi.org/10.1162/106365602320169811.

[6]V. Gouet, “MADDPG,” *GitHub*, Nov. 02, 2023. https://github.com/Gouet/MADDPG-pytorch (accessed Dec. 17, 2023).

[7]“openai/multiagent-particle-envs,” *GitHub*, May 17, 2020. https://github.com/openai/multiagent-particle-envs

[8]R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments,” arXiv:1706.02275 [cs], Mar. 2020, Available: https://arxiv.org/abs/1706.02275

[9] “Pettingzoo documentation,” Simple Tag - PettingZoo Documentation, https://pettingzoo.farama.org/environments/mpe/simple_tag/ (accessed Dec. 17, 2023).

[10] Openai, “Openai/MADDPG: Code for the MADDPG algorithm from the paper ‘multi-agent actor-critic for Mixed Cooperative-competitive environments,’” GitHub, https://github.com/openai/maddpg (accessed Dec. 17, 2023).

[11] “Deep deterministic policy gradient¶,” Deep Deterministic Policy Gradient - Spinning Up documentation, https://spinningup.openai.com/en/latest/algorithms/ddpg.html#pseudocode (accessed Dec. 17, 2023).

[12] Gymnasium documentation. Gymnasium Documentation. https://gymnasium.farama.org/

[13] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. arXiv preprint arXiv:1705.08926, 2017.

[14] Emruur, "prey_predetor/Results_v0," GitHub repository, 2023. [Online]. Available: https://github.com/Emruur/prey_predetor/tree/main/Results_v0