Population dynamics RL*

Antoine Debouchage[1,2], Guillaume Levy[1,2], and Clement Wang[1,2]

[1] CentraleSupélec, France
[2] Master MVA, Ecole Normale Supérieure Paris-Saclay, France

Abstract. In this project, we delve into the dynamics of ecological systems through a reinforcement learning (RL) approach, employing the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm. We focus on a 2D environment where prey and predators move freely, constantly planning to survive.
The environment is structured as a dynamic map where prey aims at evading capture while predators seek to hunt them down. What sets our model apart is the incorporation of a nuanced species concept, wherein entities can exhibit varied roles—prey, predator, or even neutrality—towards one another. This introduces a layer of complexity, reflecting real-world ecological systems where multiple species coexist and interact in intricate ways. Another focus is to establish interactions between agents, fostering dynamic and realistic behaviors within the simulated ecosystem.

Keywords: Prey-predator model · Reinforcement learning · Multi-agent Reinforcement learning · DDPG · MADDPG

1 Basic environement and agents

1.1 Environement map

The basic setting is a 2D space where $n = \sum_{i=1}^{k} n_i$ specimens, divided into k species with predation relations, freely roam and interact. This space can either be a rectangular field with borders, or it can loop around depending on the setting we choose, creating a continuous infinite playground (Fig 1).

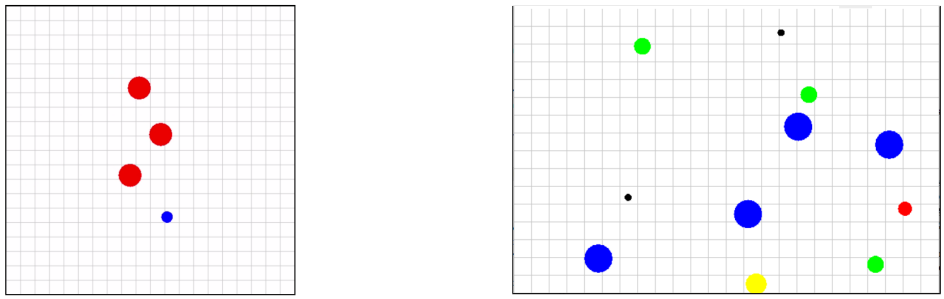


Fig. 1: (a) A simple environment with 2 species: 1 prey vs 3 predactors. (b) A complex environment with 5 species: 1 target, 4 low agents, 1 mid agent, 3 high agents, 1 super agents

1.2 The agents: Prey and predators

Foodchain Every agent embodies a member of a particular species. In the initial setup, one species assumes the role of prey while the other adopts the role of predator. The objective of the predator species is to catch the prey, while the prey strives to prolong its survival. Each agent's identity is determined by its coordinates. When the distance between a prey and a predator falls below a specific threshold, the prey is considered captured. After a few experiments, we moved to more complex interactions between species where some species can be both preys and predators ((b) of Fig 1 & Fig 2).

Observation Every agent receives information about the positions of all other agents relative to itself. This observation is conveyed through a vector with $2m+2n$ dimensions, where each value falls within the range of $[-1, 1]$, denoting the relative positions of other agents, plus its previous action.

* Supported by CentraleSupélec

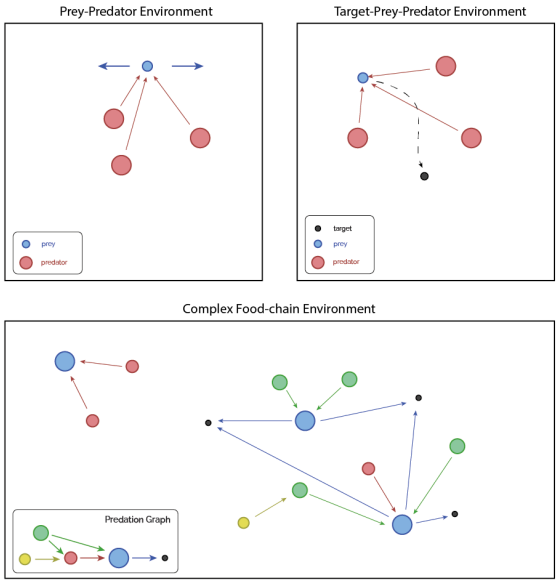


Fig. 2: Foodchain

Action During each step, an agent executes a continuous action, represented by a 2D vector containing values ranging from -1 to 1. The agent moves in the direction specified by this vector, adjusted by the speed characteristic of its species.

Reward The reward is designed to mimic real-world dynamics of predator-prey interactions, comprising multiple components that also encourage collaboration among agents of the same species. For agent a of species p, the reward is the sum of:

- *Caught by a predator*: -10 per collision with predators. In practical terms, we simply assess whether the distance to a prey falls below a specified threshold value.
- *Catching preys*: + 10 if an agent of the same species catches a prey.
- *Evading predators*: $\alpha \sum_{b \in \text{preds}_p} \text{dist}(a, b)$
- *Hunting collaboration*: Collective reward based on the minimization of the distance of all agents of the same species to their closest prey

$$-\beta \sum_{a^* \in \text{species}_p} \min_{c \in \text{preys}_p} \text{dist}(a^*, c)$$

- *Border penalization:* $-(\max(0, |x| - 0.9)) * 10 - \max(0, |y| - 0.9)) * 10$

Done In a straightforward environment with just one prey, the termination condition is triggered when the prey is captured. However, in a more intricate setting, there's no early stopping condition; episodes conclude only when the maximum episode steps are reached.

2 Algorithm

2.1 Deep Deterministic Policy Gradient (DDPG) and Multi-Agent DDPG (MADDPG)

DDPG [1] is an actor-critic algorithm based on the result of the deterministic policy gradient theorem. It utilizes a neural network to approximate both the actor, which represents the policy, and the critic, which evaluates the policy's action-value(Q-) function. The Bellman equation are used to update the Q function which is then used to learn a good policy. The objective is to for the actor to output a near optimal policy, while the critic network learns to predict the expected return (i.e., cumulative reward) associated with taking that action in that state.

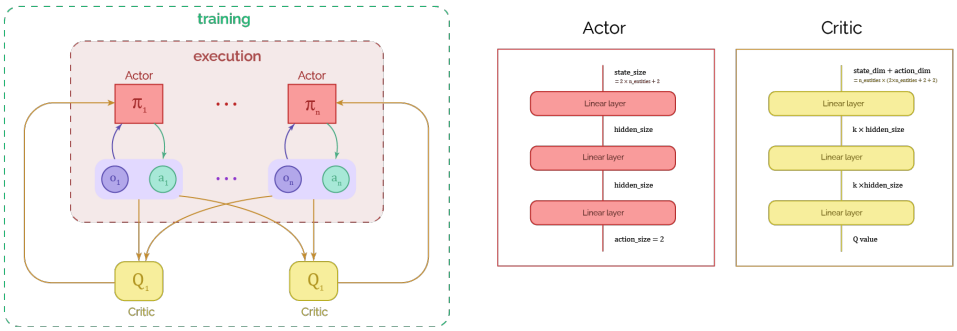


Fig. 3: Left: MADDPG setting, Right: DDPG Actor-critic networks

MADDPG [2] extends DDPG to multi-agent settings, enabling agents to learn policies in cooperative or competitive environments. MADDPG adopts a centralized training with decentralized execution approach, where agents share information during training but make decentralized decisions during execution.

In MADDPG, each agent has its actor network, which learns its individual policy, but shares a centralized critic network. The centralized critic network takes as input the states and actions of all agents and learns to estimate the value function for each agent. This allows agents to take into account the impact of their actions on other agents' rewards, fostering coordination and cooperation. To simplify our setup, we use separate networks for each of the $m + n$ agents to calculate their Q-function, instead of using one network to calculate all the Q-functions.

Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a'_1, \ldots, a'_N)|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i} \boldsymbol{\mu}_i(o_i^j) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)\big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$

  end for
end for

Fig. 4: MADDPG pseudo code

Implementation details

Replay buffer During training, DDPG employs a replay buffer to store and sample experiences, which helps to decorrelate the data and stabilize training. Additionally, it utilizes target networks to mitigate issues related to moving target problems. The actor and critic networks have target

counterparts that are periodically updated to slowly track the learned networks' parameters, which stabilizes the learning process.

Neural network architecture The actor network is designed as a 3-layered feed-forward network, which accepts a vector with $2n$ dimensions representing the observation state as input and produces a 2D vector of actions as output. The critic receives as input a concatenated vector of the observation state and the actions taken by each agent, resulting in a vector of $4n$ dimensions. It then outputs a single value for each agent, which serves as the estimate of the Q-value for the agent in the given state and action.

We implemented n actors and n critics in our setting (Fig 5), one for each agent. In this configuration, we extended the classical MADDPG approach, which entails a one-to-one correspondence between the agents and the networks, to incorporate more intricate networks. The rationale behind this extension was to observe how agents of the same species would interact when they share the same network. This stems from the intuition that there exists an implicit invariance in the environment among agents of the same species, as an agent would likely take the same action as another if they were in the same position. Therefore, employing a shared network was aimed at enhancing network capability and convergence.

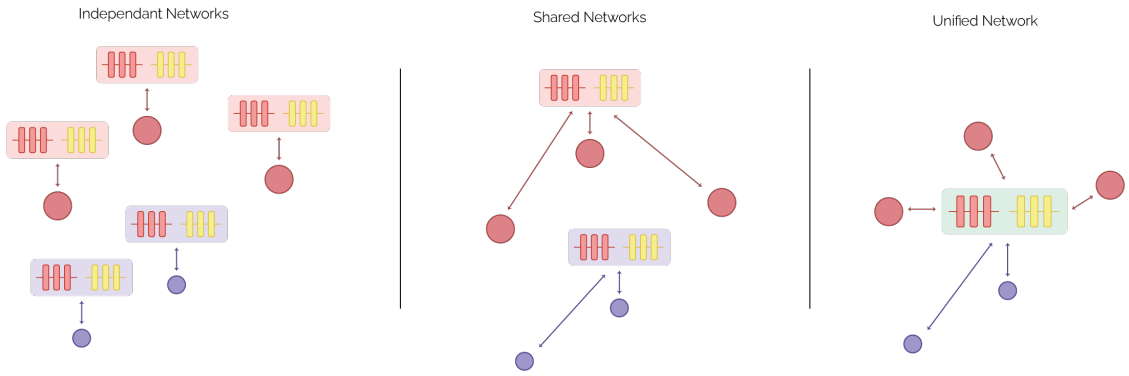
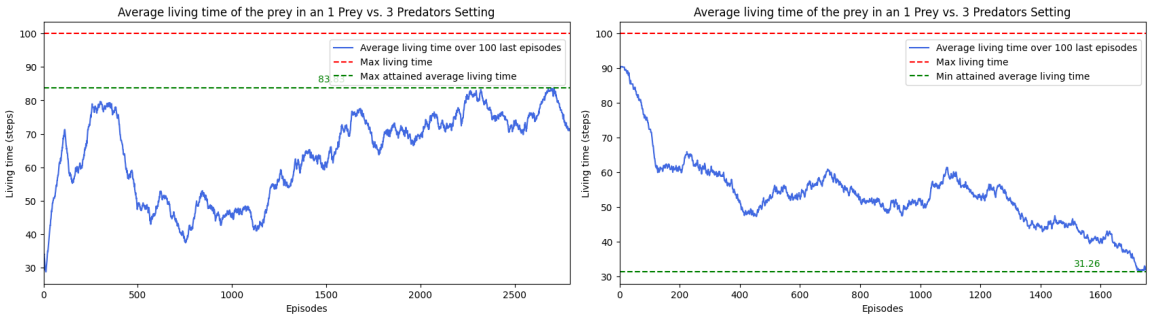


Fig. 5: MADDPG settings. We only implemented and tested the first setting.

3 Results

Crafting a suitable reward scheme to favor cooperation between agents turned out to be the primary challenge in this project. Initially, our approach involved providing a positive reward solely to the agent capturing the prey. However, under this configuration, we observed a lack of cooperation among agents: all predators just pursued the prey. When the prey's speed was slower than that of the predators, it was invariably caught; if the prey's speed surpassed that of the predators, it consistently managed to evade capture (Fig 6a). To mitigate this imbalance, we adjusted the prey's speed to ensure its consistent escape. Consequently, any instance where the prey was captured indicated successful cooperation among the predators. Then, we introduced a shared reward mechanism among all predators. This adjustment aimed to encourage collaborative efforts among predators (Fig 6b).



(a) No cooperation between predators

(b) With cooperation between predators

Fig. 6: Average Lifespan of Prey Throughout Training in 1 Prey vs 3 Predactors setting

The extension to the complex food chain was relatively straightforward as the MADDPG algorithm is already based on the number of agents. One of the main observation is that due to the environment's objectives, there is a constant battle to catch preys and escape predators giving rise to chaotic behaviours that highly depends on the initial position of the agents. Something interesting that also arose in the simple prey predator environment is that when the preys have no goal they will simply chose the action that will maximize their distance to the preys which in the case where the initialisation was in favor of the prey and the predator were slower than it, would result in simple escape given the environment surface is large enough. To counter this, the food chain environment allows for the creation of idle agents named targets that the preys will try to catch, to get additional reward, while dodging the prey leading to interesting results.

As for our MADDPG extension, we found that using shared networks require more backend work on the gradient computation. Introducing a single network for different species created high instability during training for all the species concerned. This problem seems to be coming from the update of the network by all the different entities. Since they all have different reward and behavior, it makes the credit assignment task very difficult for the model to track. Thus, the training was taking a bit longer to converge because of the number of gradient updates at each step proportional to the number of entities of the species which enforced the model to update in several different direction at once.

Furthermore, as previously mentioned, we attempted to model the environment as a torus, wherein the borders connect to each other. We observed that when the number of predators is insufficiently small, the prey can easily evade them, even without possessing extraordinary speed. This phenomenon arises because an agent can swiftly traverse the board, reaching any part from one corner to the opposite in just a few steps. Compared to the other, it cannot get stuck in the corner of the map. Consequently, this configuration presents challenges for predators to coordinate and exhibit cooperative behaviors effectively in capturing the prey.

4 Conclusion

In this report, we studied the performance of a multi agent reinforcement learning approach in a cooperative and competitive setting. Multiple agents had to interact in this foodchain environment and learn to cooperate with each other to be effective hunters. To achieve this goal, we used a multi agent reinforcement learning framework called MADDPG for Multi Agent Deep Deterministic Policy Gradient based on the off policy algorithm DDPG. It introduced a way to learn a centralized value function for all agents and still keeping the decentralized execution. With this algorithm, the agent were able to cooperate with each other to reach a faster prey quicker and quicker. From there, several improvements are possible. Carrying a deeper analysis of a more complex foodchain environment with a notion of energy for each agent that will be depleted with the agent's movement and refilled when eating could be an interesting way to see how they are able to deal with energy management.

References

1. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
2. 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)