Dark Forest Theory and Multi-Agent Reinforcement Learning

Théo Michel, Selim Gmati, Elena Immen

Dark Forest Theory and Multi-Agent Reinforcement Learning

Théo Michel **Selim Gmati** **Elena Immen**

Abstract

Multi-Agent Reinforcement Learning (MARL) has proven effective in many areas. These include robotics, computer networks and even traffic control. It is also increasingly used in game theory to solve complex problems. Taking this idea further, we decided to apply MARL to a new type of problem, interplanetary communication. To do this, we created an environment that simulated the potential civilisations interacting in the universe. We then ran reinforcement learning agents representing the civilisations, each with the goal of surviving in this universe. We then analysed the strategies they came up with and saw that the strategies reflected what humans have been observing for the last hundred years: complete silence.

1 The Dark-Forest Hypothesis

1.1 Predicates

The predicates of the Dark Forest hypothesis, taken from the book "The Dark Forest" by Cixin Liu, are :

- The universe is teeming with life, and a significant percentage of it is capable of communication. (See Drake equation (6))
- Suppose that survival is the primary need of a civilization.
- Suppose that civilisations expand continuously over time, but the total amount of matter in the universe remains constant.

We can also add to them :

- Assuming that civilisations are at about 1.3 on the Kardashev scale, they can send civilisation-ending kinetic projectiles at speeds close to the speed of light that are unavoidable.
- Suppose communication between planets is limited by the speed of light.

If we accept these predicates as true, then the question arises: where are the aliens? This is known as the Fermi Paradox: why have we not yet been able to see them? One possible explanation is the Dark Forest hypothesis.

1.2 Dark Forest hypothesis

The Dark Forest hypothesis suggests that other civilisations do not communicate with each other for fear of their location being discovered, which could lead to a deadly strike from the heavens. The other civilisations would have an incentive to kill each other on sight; the speed of a message is the same as the speed of a lethal projectile, making negotiations difficult, leading to the logical choice for survival: to annihilate the other solar system rather than take any risks.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2 Building the environment

To build an environment that simulates a small universe to test the Dark Forest hypothesis, we chose a stable and tested solution that also met the requirement for high performance: DeepMind's Melting Pot framework. (4; 5). Melting Pot is a tool for testing and comparing different algorithms used in multi-agent reinforcement learning environments. In particular, it focuses on testing new and unfamiliar social environments containing a variety of social interactions such as: cooperation, competition, deception, reciprocity, trust, stubbornness. Our aim was to build a base case of an environment that would allow us to test the hypothesis. We wanted to build a foundation that could be easily extended to build more sophisticated versions of the universe to model scenarios closer to the real development of civilisations.

We assume that the primary need of a civilisation is survival. Therefore, the goal of all agents in the environment is to survive as long as possible.

2.1 Environment

The universe is implemented as a 16x16 grid. We assume that the distribution of life in the universe does not follow a specific pattern and that each planet has different initial conditions. Within the universe we therefore spawn a random number of agents $\in [2, 6]$ with a random amount of initial resources $\in [200, 300]$. Per time step, each agent loses 1 resource to simulate the finite nature of life, as agents die when they run out of resources.

2.2 Reward

Each agent receives 1 reward per time step. There is no reward for any other behaviour, as we want to keep the self-induced bias as small as possible.

2.3 Actions

Each agent has 3 available actions. Performing an action costs the agent resources. Currently, resources are allocated arbitrarily, based on the agent's perception of how much such an action might cost in a real environment.

1. Broadcast own position - Cost: 7 resources
2. Private message own position - Cost: 5 resources
3. Kill other agent if their position is known to the killer - Cost: 10 resources

Due to framework restrictions, agents can still perform the Kill action on other players even if they do not know their position. This is a useless action as it does not kill the other player. This means that agents are performing an action that gives them neither a positive nor a negative reward. To mitigate this problem, we have penalised this action by making it cost 2 resources.

3 Multi-Agent reinforcement learning

Similar to game theory, much of the research in MARL revolves around social dilemmas. More specifically, it focuses on what policies agents would learn as they try to maximise their own payoffs through a trial-and-error process. This often involves a conflict between the needs of the agent and the needs of the group, and brings a particular insight into what is the best behaviour in each situation. Since we were interested in how MARL agents would learn to behave in our environment, in the situation described by the Dark Forest theory, we decided to experiment with two algorithms. Beyond analysing the learned policy, our goal was to study the impact of each implementation on the results, as each would value different factors.

MARL algorithms can be centralised or decentralised. When centralised, agents can share observations, value functions or even policies. When decentralised, agents do not share any of these parameters and learn individually. Of course, this has a major impact on the learning process, as it limits the information available at each iteration. It also raises the problem of non-stationarity.

In fact, a multi-agent environment, as observed by each agent, changes greatly from one observation to another. Since all agents change their policies as they learn, the Markov property is violated, which is an additional obstacle. While these methods do not necessarily converge, it has been shown experimentally that some still perform well.

In our study, we therefore experimented with two algorithms, PPO and QMIX.

3.1 Experiment 1

PPO is a single agent reinforcement learning algorithm motivated by the question "how can we take the biggest possible improvement step on a policy using the data we currently have, without going so far that we inadvertently cause performance collapse". PPO takes multiple steps of SGD to maximise the objective. Here L and θ are given by the following expressions:

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_k}} [L(s, a, \theta_k, \theta)]$$

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right)$$

Research has shown that PPO-based multi-agent algorithms achieve surprisingly strong performance and can therefore be a good starting point. (1)

3.2 Experiment 2

While the previous approach is decentralised, we also wanted to experiment with centralised reinforcement learning. QMIX is a novel value-based method that has been shown to be very effective in mixed environments with both cooperation and competition, which is what we are ultimately considering for our environment. It can train decentralised strategies in a centralised way. In fact, it uses a network that estimates joint action values calculated from all the action values of the individual agents. The main constraint is to ensure that Q is monotonic with respect to each individual Q :

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \quad \forall a \in A$$

To achieve this, all the weights of the mixing network are positive, which ultimately ensures that :

$$\underset{\mathbf{u}}{\operatorname{argmax}}\, Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u^1} Q_1(\tau^1, u^1) \\ \vdots \\ \operatorname{argmax}_{u^n} Q_n(\tau^n, u^n) \end{pmatrix}$$

However, despite our attempts, we have not been able to train agents with the QMIX algorithm on our environment. Therefore, we will only see the results of our first experiment.

3.3 Implementation

In practice, we have used self-play to train the agents, with a separate PPO policy for each agent (IPPO), which leads to interesting training results, as the strategies adopted by the agents during training can be extremely varied.

We used RLlib in order to train the agents and to track the progress.

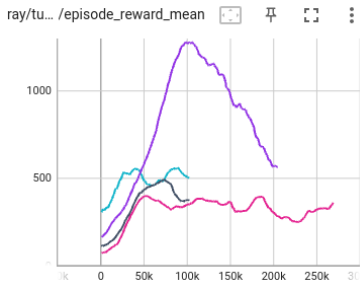

Figure 1: The average episode reward

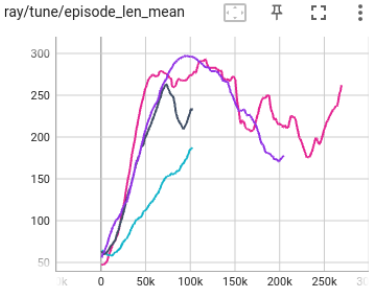

Figure 2: The average episode length

Figure 3: Rose: 2 player game, Violet: 5 player game, 3 player game with useless kill penalty, Cyan: 3 player game with useless kill penalty and kill reward

4 Experimental results

4.1 Training

In fig. 3 we can see that on all 4 different configurations see that they quickly hit the maximum reward possible, which is *num players* $\times$ *resources*, with $resources = 300$. After that, the result is unstable as a result of the IPPO algorithm.

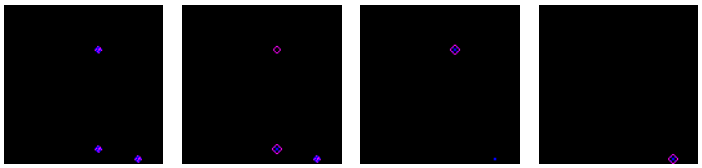

Figure 4: 3 Players without killing incentives, after 80 iterations see (2) for legend

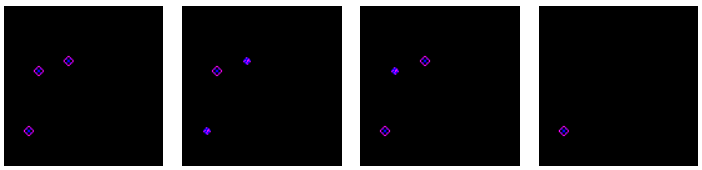

Figure 5: Environment with 3 players, an additional reward for killing, and useless kill penalty after 80 iterations see presentation (2) for legend

In fig. 4 we can see the bottom right, civilization has learned never to send messages. In fig.5 we can see that the agent's rarely ever try to send messages any more, they still try to kill each other all the time at this state of the training.

5 Challenges

5.1 Software

Software stability and reproducibility is a big problem in the field of machine learning, and the more niche the domain, the more obvious the problem. For example, MARL for social interactions is not a mainstream field, so many bugs are forgotten and documentation is outdated because the companies developing the software often have their own in-house tools and don't notice the external problems. An example of such a problem is a breaking commit to DMLab2d that broke the Ubuntu installations,

but two of our members downloaded the software before this commit and one after, this caused a lot of confusion and we discussed the problem with the developers of DmLab. We can also say that most of the examples on GitHub were outdated and no longer worked, but we have now updated them. This is just one example of how access to this beautiful technology is hindered by software engineering oversights.

5.2 Theoretical

Building the environment has also been a difficult task, as it is crucial not to let our bias towards an expected outcome influence the actual outcome. We don't think we've yet succeeded in this, as time constraints have not allowed us to refine the environment enough to model the problem perfectly, and some of the shortcuts we've taken are biased towards the outcomes we wanted, such as adding a reward for killing, or the fact that planets are only observable when they send messages. We hope to improve this in our next iteration of the environment.

6 Future Work

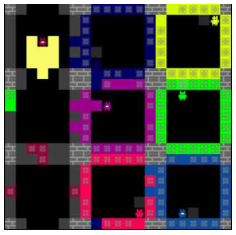

Figure 6: Melting pot environment : territory basis for future work

The next step is to use more complete environments using the experience we have learned here, right now if we want to add more functionality we have to expand the action space, the state space will stay relatively small. But RL algorithms learn better when you have a larger state space and a smaller action space. So we will probably have to start from scratch, the idea here would be to use the ideas from the territory 6 environment, it is an environment where agents are stuck in an 8 by 8 box, where they can peacefully get rewards for the entire duration, but also choose to invade other boxes and kill their inhabitants to get more resources. We plan to extend this to add communication and cooperation and killing through the walls.

7 Conclusion

In summary, our study focused on applying multi-agent reinforcement learning (MARL) to investigate the Dark Forest hypothesis, which suggests that civilisations in the universe remain silent and hidden for fear of being detected and attacked. We created a simulated universe environment using the Melting Pot framework and trained MARL agents to survive in this environment. Our experiments used two algorithms, IPPO, to analyse the strategies and behaviours learned by the agents.

Overall, our study contributes to the understanding of how MARL can be used to investigate the Dark Forest hypothesis, shedding light on the potential behaviours and strategies of civilisations in the universe. Further research and development in this area may provide valuable insights into the Fermi Paradox and the existence of extraterrestrial civilisations.

8 Contributions

Elena Immen: Understanding the Melting Pot framework and building the environment

Théo Michel: Development of the environment, integration of PPO agents and evaluation of results

Selim Gmati: Exploration of MARL algorithms, their interaction with the developed environment and integration of QMIX

Joint efforts: Environment design, presentation, quiz, final report

References

[1] The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games, Chao yu et al.,2021

[2] Link to the presentation where you can see the GIFs

[3] QMIX: Monotonic Value Function Factorization for Deep Multi-Agent Reinforcement Learning, Rashid et al. 2018

[4] Scalable Evaluation of Multi-Agent Reinforcement Learning with Melting Pot, Joel Z. Leibo, Edgar Duenez-Guzman, Alexander Sasha Vezhnevets, John P. Agapiou ete al. 2021

[5] Melting Pot Repository

[6] Click to see the Drake Equation

[7] Link to the oral presentation