TEXAS A&M
UNIVERSITY®

CSCE-689
Reinforcement Learning

Ant-Based Multi-Agent Reinforcement Learning

Link to github: `https://github.com/acramer/rl-project`
Link to video: `https://youtu.be/nnnvUvHV_8s`

Tejas Rajratna Adsul
430001123

Andrew Cramer
924005617

Contents

1 Introduction

1.1 Multi Agent Reinforcement Learning (MARL)

Multiagent systems involve autonomous agents that can interact with each other and share a common environment. These systems have found applications in a number of areas such as robotic teams, distributed control, resource management, collaborative decision support systems, etc. [1] In such a system, every agent tries to optimize its own reward by interacting with the environment and with each other. The agents can be competitive, cooperative, or a mixture of two. Numerous algorithms emerge from this categorization [2]. Q-learning [3] is one of the most well-understood and applied algorithm in single agent reinforcement learning (RL). Thus, some of the earliest works involving (MARL) include suitable modifications to Q-Learning. Tan [4] extended RL to systems with multiple agents (the example given was of different hunters trying to catch a prey in a gridworld) under different assumptions such as sharing sensations, sharing policies, and joint tasks. A form of decentralized Q-learning was proposed, and since then various algorithms have been shown to successfully emulate cooperation of different kinds among the agents of the system. This project attempts to use these ideas of MARL in a simple domain and test its working.

1.2 Learning in social insects

Social insects have a remarkable ability of solving problems. Every member of the colony is an individual agent that follows the same set of behavioral strategies towards a common goal [5]. While they may exhibit some flexibility in their jobs, most colonies have specific work distribution among their members, such as foraging of food, protection of nest, nurturing the young, etc. Such insects do not have a centralized control. Information is dispersed throughout the colony, and members collect it either through the environment itself or through interactions with other members. [5]. Thus, they are of a particular interest in designing learning algorithms. Swarm reinforcement learning methods [6, 7] are the result of a combination of inspiration from these social insects and optimization algorithms. Many ant based algorithms such as Ant-Q [8] and Phe-Q [9] involve modifications to the original Q-learning algorithm by including aspects of ant foraging methods. Using the guidelines of such algorithms, this project works on creating an ant-based domain that is slightly different from those generally used for ant-based learning. MARL methods like decentralized learning are then tried within this domain and their efficiency is tested.

2 Ant Gridworld

2.1 Ant Ecosystem

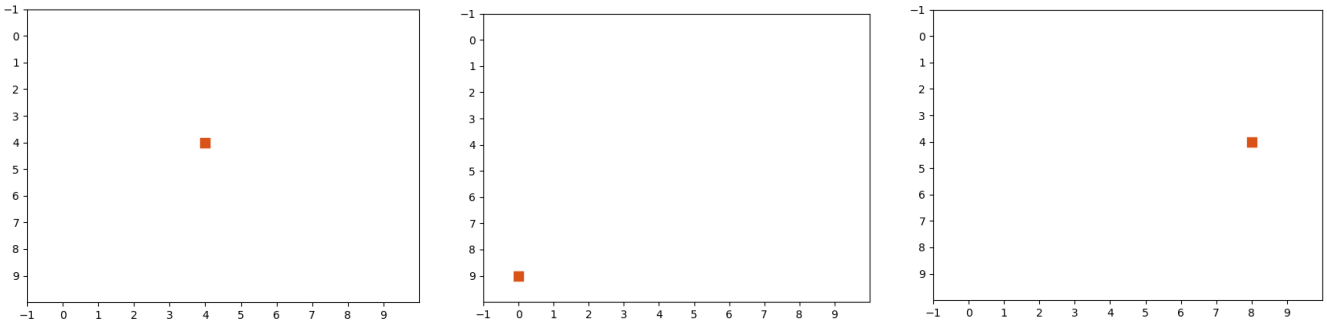
Though majority of the behavioral methods of ants remain the same, different species show different tactics when it comes to foraging food. Some ants, especially Desert ants [10], have no way of communicating due to extreme weather. They perform solitary foraging, where every ant leaves the nest, finds food, and brings it back to the nest without exchanging any kind of information with other ants. However, most of the species use pheromone trails as a way of communicating information about food, predators and nest to other members. Fire ants [11] further distribute the job of foraging among members of a colony. The

foragers of the colony are only involved in finding new resources of food and laying a trail from those resources back to the nest. The workers on the other hand follow these trails and collect all the food they can find at the end. This depicts a separation between the exploring and exploiting phases of foraging. This work intends to use this idea and create two different sets of agents in a multiagent system - one focused on exploration and one on exploitation. The following section details the domain.

2.2 Environment

2.2.1 Nest

The environment in the testing domain consists of a gridworld of a given size. For preliminary testing, it is initialized as a square gridworld. However that can be changed easily. A nest is placed at a location specified by one of the three options - center, corner, random. The random option places the nest at a random gridpoint in the environment, including the center and any of the four corners. Fig. 1 shows the position of the nest for these three options in a 10x10 gridspace. The nest acts as the starting and the ending position for the ants. The location of the nest is fixed for the simulation. Some area around the nest, defined by the number of gridpoints in each direction, is kept clear. This ensures that ants have to travel a bit in order to interact with other aspects of the environment.



(a) Nest at center (b) Nest at a corner (c) Nest at a random gridpoint

Figure 1: Nest locations

2.2.2 Food

Food in the gridworld is represented by morsels with specific weights placed on random gridpoints. This is achieved with the help of two variables, *food_num* and *max_wt*. The variable *food_num* determines the number of morsels of food to be distributed, while *max_wt* provides an upper bound for the weight of these food morsels. The weight of the food implies that it can be collected multiple times. A single ant collects one part of the food morsel. The food is extinguished when it has been collected for times equal to its weight. The food morsels are constructed by randomly selecting *food_num*

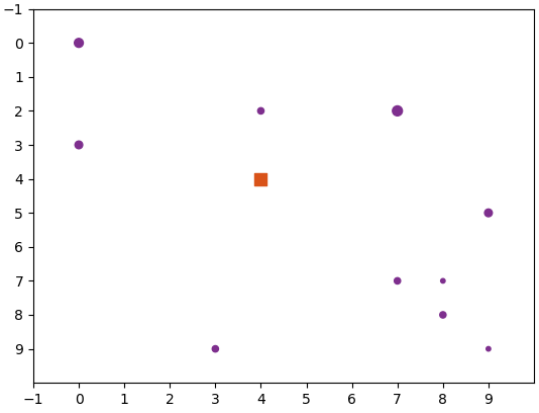


Figure 2: Food morsels

number of values between 1 and *max_wt*. Every morsel is then placed on a gridpoint not occupied by the nest or the area surrounding it that it kept clear. Fig. 2 shows the distribution of these morsels, with the nest at the center. The size of the morsel is indicative of its weight.

2.2.3 Obstacles

Besides food, there are obstacles scattered around the environment. These act as immovable objects and the ants have to circumnavigate them. The obstacles are distributed on the gridpoints where no food or nest (along with its surrounding) exists. Fig. 3 shows the complete environment with nest (square marker), food morsels (purple markers with variable sizes) and obstacles (black markers with same size). The locations of the food as well as the obstacles are randomly chosen, but they are kept constant throughout the simulation. This can be changed over time to form a stochastic environment, making the domain closer to the natural environment of ants.

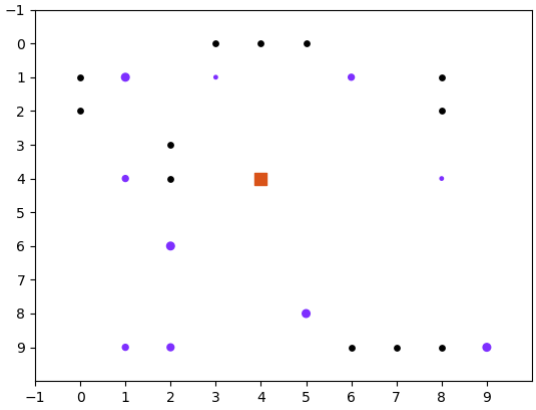


Figure 3: Obstacles along with nest and food morsels

2.3 Agents

Ants form the agents in the environment. As mentioned before, the foraging process has been divided into two parts - finding food sources, and collecting them. Thus, two different types of agents are used in the model.

2.3.1 Exploring Ants

Exploring ants are responsible for finding new food sources. They leave the nest, try to explore as much area as possible and search for food. Once a food morsel is found, they collect one part from it and return to the nest. On the way back, they lay a pheromone trail. After reaching the nest, they deposit the food and go back to search for new food sources. Exploring ants are discouraged from following the trail they just laid to go back to the food source and collect more of it. They collect from a food source only if it is completely new or there aren't any pheromone trails surrounding it, meaning for some reason other ants weren't able to collect the food completely. The percentage of exploring ants out of the total population is decided at the beginning of the simulation. Since the number of food morsels is typically lesser than the overall quantity of food, this percentage is usually kept lesser than 20%.

2.3.2 Exploiting Ants

Contrary to exploring ants, exploiting ants are incentivized to find food sources already discovered and collect them as fast as possible. Thus, they leave the nest when at least one pheromone trail has been formed. After reaching the end of the trail, they collect the food and return on the same trail to the nest. On the way back, they reinforce the trail with more pheromone. The strength of the pheromone

communicates the quality of the food resource at the end of the trail. Thus, ants tend to travel along the trails with the strongest pheromones. If the ants do not find any food at the end of the trail, they return to the nest and do not reinforce the trail on the way back. At every time step, small amount of pheromone in the environment evaporates from all the trails. Thus, trails which don't get reinforced evaporate completely over time.

3 Model Parameters

This section describes all the model parameters used in the design of experiments. The default values wherever applicable are also provided. All these values were set through trial and error seeking a good balance of exploration difficulty and sparsity of environment elements.

1. $env\_size = 20$: The side length of the area of the gridspace. It is assumed that the environment is square, but can be modified to be a rectangle easily.
2. $food\_num = 15$: Number of food morsels.
3. $max\_wt = 5$: Maximum weight of each food morsel.
4. $nest\_loc =$ center: Location of the nest with respect to the gridspace.
5. $nest\_range = 1$: Area around the nest kept clear of food
6. $obstacle\_no = 10$: Number of obstacle blocks in the gridspace
7. $memory\_len = 20$: Number of states that the ants can remember

To support learning about the environment through interactions, a reward function was designed. Different activities were assigned either positive or negative rewards corresponding to whether they helped in finding food and depositing it at the nest faster. The following list mentions these rewards and provides minimal reasoning for them.

1. Dropping food at nest [**20**]: The most reward is given to the ants for depositing food at the nest. This is same for both exploring and exploiting ants. This allows the Q-values of the gridpoints near the nest to give highest probability for actions that take the ant to the nest.

2. Returning to the nest using a pheromone trail [**0**]: The idea behind this reward is to incentivize ants to use existing pheromone trails while returning to the nest, under the presumtion that the trail signifies the shortest path to the nest. However, since ants take very long initially to return to the nest after finding food, this reward was kept to zero to discourage ants from following their own trails in the hope of achieving a reward. Again, this is kept same for both exploring and exploiting ants.

3. Returning to the nest without using a trail [**0,-1**]: This reward is kept 0 for an exploring ant but -1 for an exploiting ant. Ideally, the exploring ant is supposed to create a trail between the food and the nest while the exploiting ant should follow such trails and only collect food. The negative reward attempts to encourage exploiting ants towards following more trails.

4. Reducing distance between the nest and the current location while returning to the nest [**1**]: In order to help the ants realize that they are supposed to return to the nest with the food, this positive reward is given to an ant every time it takes an action that gets it closer to the nest. This is true for both types of ants. Moreover, to further push this behavior onto the ants, they are given a negative reward of the same value if they take a step that takes them away from the nest.

5. Finding new gridpoints while finding food [**1,0**]: This rewards incentivizes exploring ants to find new gridpoints of the environment as fast as possible in the beginning in order to find more food. The exploiting ants are discourages from learning this behavior.

6. Finding food [**10**]: This is the second highest reward given to the ants. One of the reasons behind this high reward is to let the ants reach the same position again when the food has large weight and can be collected multiple times. Once the food has been extinguished, the ants gradually learn to not go to that spot again.

7. Walking on a trail while finding food [**0,1**]: Exploring ants are not given any rewards for following trails since they are supposed to explore newer areas. Exploiting ants on the other hand are rewarded for following trails under the presumption that they would learn to follow trails to reach food quickly and exploit it.

8. Stepping towards nest while searching for food [**-1**]: A negative reward is given to both types of ants in case they step closer to the nest while searching for food. This allows exploring ants to travel farther from nest, and exploiting ants to take trails that go farther from nest.

These reward values were chosen arbitrarily for initial testing. The results from the algorithms are obtained using these rewards.

4 Algorithms

In order to understand the learning process within the domain, different algorithms were constructed from scratch. Primarily Q-Learning was used in all these algorithms. Each of them is detailed below.

4.1 Procedural Ant

The baseline was considered to be an ant that learns deterministically rather than through reinforcement. The following pseudocode elaborates the algorithm.

```
Initialize environment, ants with location at nest
Repeat while True:
    Repeat for every ant:
        Get neighboring gridpoints
        If the ant is 'searching for food':
            If food is present on one of the gridpoints:
                Step on that gridpoint, collect food
                Change status to 'returning to nest'
            Else if trail is present on one of the gridpoints:
                If the ant is 'exploring' type:
```

```
                    Disregard the trail that the ant has already traveled recently (while
                        searching for food or returning to nest)
                Disregard the trail that doesn't take the ant closer to the nest
                Choose one of the valid trails based on pheromone strength
            Else:
                Do one step of correlated random walk
        Else:
            If nest is present on one of the gridpoints:
                Step on that gridpoint
                Deposit food
                Change status to 'searching for food'
            Else if trail is present on one of the gridpoints:
                Check that the trail takes the ant closer to the nest
                Choose one of the valid trails based on pheromone strength
                Reinforce the gridpoint with pheromone
            Else:
                Choose the gridpoint that reduces the distance between nest and current
                    location
                Reinforce the gridpoint with pheromone
    If all food has been collected:
        Break
```

The algorithm ensures that exploring ants, after returning to the nest with food and laying a trail, do not follow the same trail to go to the same food resource again. Instead, they choose to go in a different direction to search for more food. Exploiting ants on the other hand prefer walking on a trail and going to the food to collect it, irrespective of whether they have traveled on it or not. In this case, the ant doesn't learn anything about the environment. It can only look one step ahead and decide what to do. The only way of communicating information about the environment is through pheromone trails. Fig. 4 shows the result of a run of this algorithm. The maximum time steps was kept to be 1000 and the model was run for 20 epochs. While there is no learning, the multiple epochs act as an averaging function for the plots. The ants were able to find all the food in less than half the maximum steps.

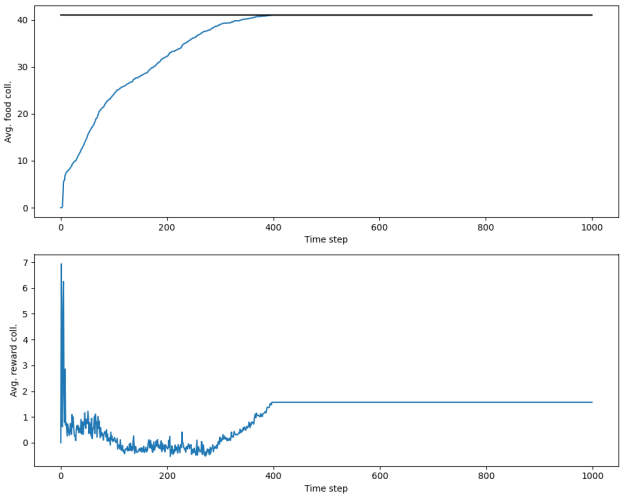


Figure 4: Procedural Ant

This acts as the baseline against which the learning algorithms can be tested. The procedural ant starts knowing that it has to find food and bring it back to the nest in the shortest way possible. This is something learning ants are not given initially. The efficiency of the algorithms thus can be indicated by the amount of time required by the ants to understand this behavior and act accordingly.

4.2 Centralized Learning Ant

This algorithm uses the idea of Q-Learning while offline. The idea is to use a Q-table that is common to the environment and shared among all the ants. Before the ants begin exploring, a simulated ant is allowed to carry out all the required activities in the environment while updating the Q-values of state-action pairs. This allows ants to access a readymade Q-table when they start foraging, thereby reducing the amount of time required to learn the process of finding food and returning to the nest. A state is in this algorithm is defined by a combination of the gridpoint, the neighboring eight gridpoints, presence of food, trail and obstacles on any of these gridpoints, and information about the ant visiting the gridpoint. This information related to the ant includes the position of the ant, the last action that the ant took, the states present in the memory of the ant and a general vector pointing to the direction of the nest. The algorithm attempts to create a continuous state space by including all the information that is available to an ant when visiting a gridpoint.

4.3 Joint Learning Ant

In this case, the ants contribute to a joint Q-learning method while interacting with the environment. The ants initially start with no knowledge of food, trails or what to do when they happen on one. Every ant gets some reward through this interaction and updates a central Q-value table for the environment. This implicitly represents a shared knowledge of the environment among all the ants. While such sharing doesn't occur in nature with real ants, it is an important step towards understanding decentralized learning.

The algorithm was run for 10000 steps and over 20 epochs. Each epoch was terminated either when all food was collected or when the specified maximum number of steps had been reached. A soft reset was performed at the end of each epoch, whereby the food morsels were reinitialized, albeit in the same positions and same quantities as before. Fig. 5 shows the results of running this simulation. The top plot shows average food collected over all the epochs. This gives an idea of how fast is the food being collected as the agents learn over time. The middle plot gives the average reward collected over all the epochs. This too explains whether the ants were able to overcome the initial randomness. Finally, the bottom figure shows the amount of food that was collected in every epoch. Initially, the ants weren't able to collect all the food within 10000 steps. However, this quickly changed and they were collecting all the food by the third epoch. The black bar in the top and bottom figures is indicative of the amount of food present initially in every epoch. Thus, ants were able to learn about the positions of food and obstacles considerably fast uding the Joint Q-Learning method.

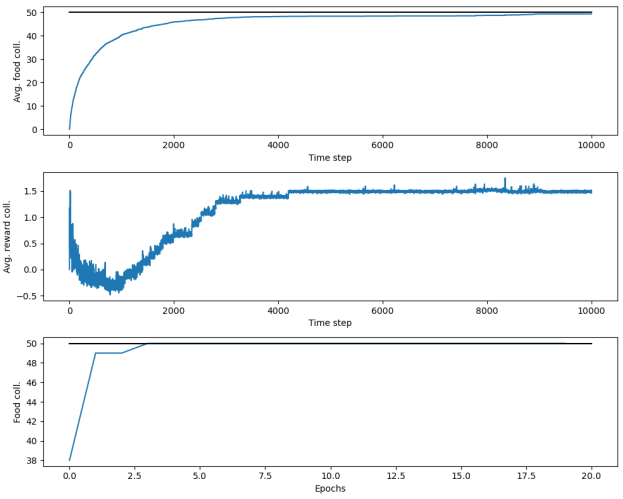


Figure 5: Joint Q-Learning

Algorithm 1: Joint Q-Learning

Initialize environment, ants with location at nest
Initialize state space as $\mathcal{S} := ((x, y), \textit{has\_food})$, where (x, y) indicates coordinates of a gridpoint and *has_food* shows whether the ant is searching for food ($\textit{has\_food} = 0$) or returning to nest with food ($\textit{has\_food} = 1$)
Initialize action space as $\mathcal{A} := \{0, 1, 2, 3, 4, 5, 6, 7\}$ where the indices correspond respectively to north, north-west, west, south-west, south, south-east, east, north-east
while *True*:
    for *every ant*:
        Get state
        Get action from policy
        Take action to get next state and reward
        Update Q table for given state-action pair
    if *all food has been collected*:
        Break

4.4 Decentralized Learning Ant

This case is similar to the Joint Learning Ant, except that instead of a joint Q-table, every ant has access to only its own Q-table. Thus, this algorithm brings the model closer to the way real ants behave. Since the experience of an ant is limited to what it learns during its foraging, the learned Q-values of a gridpoint by an ant are not available to the other ants. The algorithm has the same steps as Joint Q-Learning, except the Q-table updates are ant-specific.

Fig. 6 shows the evolution of the learning capabilities of ants following this algorithm. The parameters were kept same as for Joint Q-learning: 10000 maximum steps and 20 epochs. Comparing Fig. 5 and Fig. 6, it is clear that Decentralized Q-Learning is much slower than Joint Q-Learning. This is intuitive, since the learning is limited in this case. However, the bottom plot in Fig. 6 does show an increase in the amount of food collected per epoch for the same number of time steps. This implies that for enough iterations of a given environment, ants will be able to collect all food and deposit it at the nest as fast as they can. In the limit of large number of epochs or large number of time steps, Decentralized Q-Learning should converge to Joint Q-Learning.

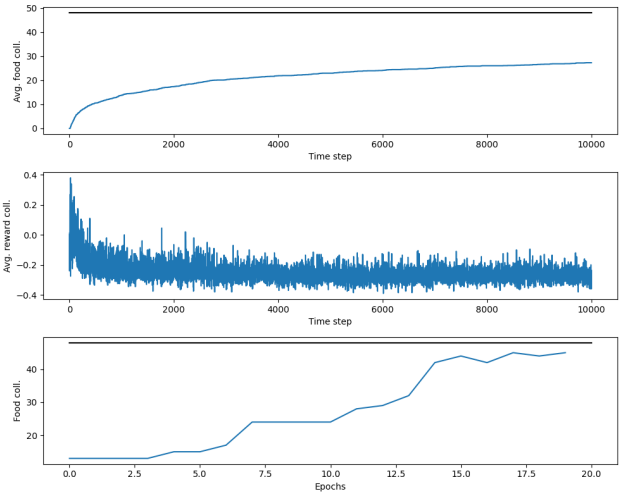
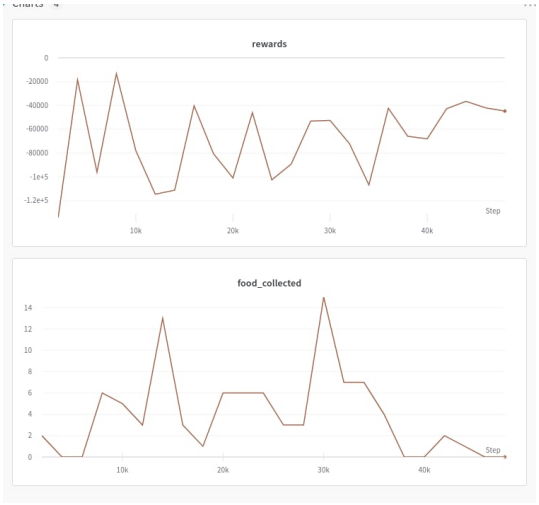


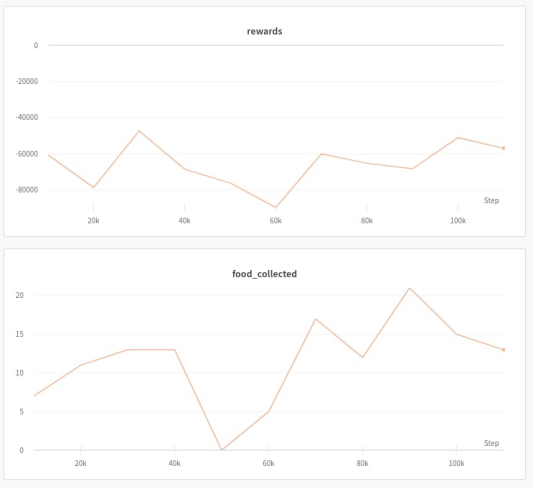
Figure 6: Decentralized Q-Learning

4.5 Centralized Deep Q-Learning Ant

Algorithms explored up to this point are unable to generalize knowledge learned to different environment, and they do not work well with large state spaces. To create a more robust model with behaviors similar to the procedural algorithm, we implemented a Centralized version of the Deep Q-Learning introduced by Mnih et al. [12], and to test how robust our model was, we reinitialized the environment entirely for each step of the training loop. Rather than a Q-table, we created a function approximator to estimate the value of taking an action at a given state [12]. For our function approximator we created a simple feed-forward neural network that takes a vector the size of our state space and outputs a vector the size of our action space. During the training loop we run the model for an episode before sampling state transitions (state, action, next state and rewards) from our model. We then take these samples compute loss of our model's reward prediction for the state action pair. To compute the loss, it is necessary to define a clone of the model that is updated periodically to approximate the Q-Value in the bellman equation used as the label.



(a) Hybrid



(b) Online

Figure 7: Deep Q-learning

In addition to this completely online version, we experimented with a partially offline version that begins by performing Q learning on multiple environments before performing offline learning on a final environment. Unfortunately, both the purely online (Fig. 7b) and the hybrid (Fig. 7a) models were very inconsistent, and did not perform nearly as well as our procedural algorithm. A few things we could have tried were to improve how our state space is represented and how we define our rewards. For our state space specifically, a point of volatility came from the fact that our models had to learn about food and obstacles 8 different (one for each direction). This lead to directional bias derived from the randomness of the environments generated during the training. A way we could have represented the state space better would be to represent a the food and obstacles relative to the direction of the action taken last, and to do the same for the approximation in the model.

5 Discussion

This work constructed a relatively new domain that was bio-inspired and implemented multi-agent reinforcement learning. Different algorithms were implemented corresponding to various levels of learning, starting from simple to implement but unrealistic, to difficult but realistic and robust. Some of the perks of these algorithms are:

1. The algorithms show that a simple Q-learning algorithm can be modified for an ant-based domain involving multi-agent reinforcement learning.
2. Different algorithms mentioned in this project show varying levels of implementation, but try to get closer to the real life scenario.
3. The complex algorithms such as Deep Q-Learning allow for more robust learning of the environment. They can also be easily implemented for larger environments where tabular Q-Learning becomes memory intensive.

However, the model along with the algorithms have a lot of space to improve. Some of the improvements that can be implemented are:

1. An optimization of the hyperparameters would allow for best calibration of the learning algorithms.
2. The behavior of the reward function tested over a large number of scenarios can allow for setting the reward values that ensure an optimal learning rate.
3. Ideas like experience replay, batch training, etc. can be used for better convergence and reduced bias in the algorithms.

One of the extensions that can be done to this project is making it more stochastic. The environment as in this work is not completely stochastic since food is being collected and trails are being laid. However, food and obstacles initially set remain at their respective positions till the end. A more stochastic environment would allow food morsels and obstacles to randomly appear and disappear. This would take the model closer to the natural environment of ants. The results provided from different algorithms for this bio-inspired ant-based domain show that it is an engaging domain with a lot to improve on. To provide an insight into this model and motivate further research is what this project intended to do.

References

[1] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[2] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.

[3] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[4] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

[5] A. E. Hirsh and D. M. Gordon, "Distributed problem solving in social insects," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 199–221, 2001.

[6] H. Iima and Y. Kuroe, "Swarm reinforcement learning algorithms based on particle swarm optimization," in *2008 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1110–1115, IEEE, 2008.

[7] H. Iima, Y. Kuroe, and S. Matsuda, "Swarm reinforcement learning method based on ant colony optimization," in *2010 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1726–1733, IEEE, 2010.

[8] L. M. Gambardella and M. Dorigo, "Ant-q: A reinforcement learning approach to the traveling salesman problem," in *Machine learning proceedings 1995*, pp. 252–260, Elsevier, 1995.

[9] N. Monekosso and P. Remagnino, "Phe-q: A pheromone based q-learning," in *Australian Joint Conference on Artificial Intelligence*, pp. 345–355, Springer, 2001.

[10] C. Buehlmann, P. Graham, B. S. Hansson, and M. Knaden, "Desert ants locate food by combining high sensitivity to food odors with extensive crosswind runs," *Current Biology*, vol. 24, no. 9, pp. 960–964, 2014.

[11] W. R. Tschinkel, "The organization of foraging in the fire ant, solenopsis invicta," *Journal of Insect Science*, vol. 11, no. 1, p. 26, 2011.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb 2015.