Self-Organizing Multiagent Systems

Partaker-Sharer Advising Framework

Denaldo Lapi and Hailey S. Miles

May 23, 2022

Contents

List of Figures

List of Tables

1 Introduction

Following the work proposed in the "A Q-values Sharing Framework for Multiple Independent Q-learners" paper [1], we aim to replicate the results of the predator/prey scenario. The goal of recreation being two fold:

- further explore the parameters chosen in the paper such as environment size and communication budget
- understand the effect the proposed parameters have on the time-to-goal and overall learning of the system

The implementation will be performed in the following order:

- Firstly, by implementing and validating a Q-learning algorithm to be used by each agent in the independent Q-learning scenario.
- Secondly, by obtaining the performance of two independent Q-learners without communication
- Thirdly, by implementing the communication between the agents.

This report outlines the technical details of all tests performed, as well as the rationale for choosing such tests. All details necessary for experiment recreation are given. The results of the recreated experiments and new experiments are discussed and reflected upon. Finally some conclusions are drawn from the original paper and the additional experiments.

[1] Zhu, Changxi, et al. "A Q-values sharing framework for multiple independent Q-learners." *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems.* 2019.

2 Environment and Model Design

2.1 Environment

The environment for these simulations is a grid world of varying size. Within the environment 2 agents will be present, which are referred to as the predators. An example of the grid world with 2 agents is shown in 1 where the two predators are shown in red. Each grid space represents a unit-less area of space that a predator can occupy at a given time instance. Additionally, a prey is randomly defined within the environment but it is important to note that it is not an agent. The prey is shown in blue in 1. The prey is able to move 1 grid space for each time step by selecting an action from: up, down, left, right, or none. The action chosen will be the one that moves it in the direction away from the two predators 80% of the time and taking a random action 20% of the time. This makes the environment dynamic as it can change every time step. The goal state of the system is then when one predator is in the same grid space as the prey and the other predator is in any of the adjacent grid spaces.

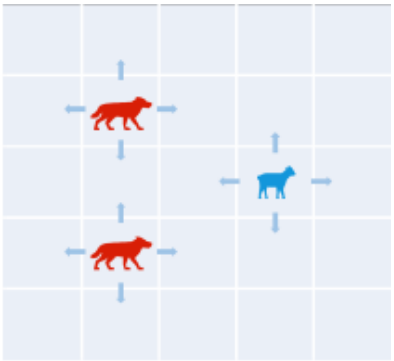


Figure 1: Example of grid world with 2 predators and 1 prey.

Each agent is comprised of an independent Q-learning model. We assume that each agent is identical. At each time step the predator agent can make a choice to move either up, down, left, right, by one grid space, or stay in the same grid space they are currently in. At each time step a predator can also choose to ask the other for advice on the action it must take. The other predator can choose to share information if it feels confident in its answer. In this way the two predators can communicate with one another to capture the prey more quickly.

2.2 Q-learning Model

Firstly we consider the single agent scenario, where we consider 1 predator and 1 prey. We assume the reader has basic knowledge of the Q-learning algorithm

(more details about Q-learning algorithm in a single-agent case and its adaptations to multi-agent systems are explained in the paper "A Comprehensive Survey of Multi-agent Reinforcement Learning" [2]).

At the beginning of a simulation the predator and prey are random initialized in a grid space with the restriction that they cannot be initialized in the same grid space. They predator and prey are then free to select an action and move around the grid. Due to the grid nature of the environment these actions are limited. The possible action are shown in Table 1.

Action	Description
Up	Agent moves 1 grid space in the x direction
$Down$	Agent moves 1 grid space in the -x direction
$Right$	Agent moves 1 grid space in the y direction
$Left$	Agent moves 1 grid space in the -y direction
$Nothing$	Agent stays in the current grid space

Table 1: Actions available to each agent

As the environment is a finite grid world, the predator and prey must be limited in some way so they do not exceed the grid limits. If the agents or the prey select an action that would have them move outside of the grid, we block the action and force the agent to stay in the same grid space. In the next step the predator and the prey can select a new action. Additionally, the predator agent is a reinforcement learning agent, whereas the prey is not. The prey takes an evasive action (i.e. moves away from the predator) 80% of the time, and takes a random action 20% of the time.

The state for the single agent case is comprised of relative distance coordinates of the predator agent to the prey. The coordinates take the form (x1, y1) where x1 represents the relative x distance to the prey and y1 represented the relative y distance. In a 4x4 grid world the possible coordinate values are -3, -2, -1, 0, 1, 2, 3. The total number of states would then be the combinations of these values: 7^2 which is 49 total states. It is easy to see that as the size of the grid world increases so do the number of states. This should be taken into account when considering the number of episodes and steps to ensure all states can be sufficiently explored.

The predator is considered to have captured the prey when it is in the same grid space as the prey. An example of the goal state for the single agent scenario is shown in Figure 2. The reward function for the single agent case is then defined as follows:

[2]L. Busoniu, R. Babuska and B. De Schutter, "A Comprehensive Survey of Multi-agent Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* vol. 38, no. 2, pp. 156-172, Mar. 2008.

- Reward = 1, if the agent is in the same square as the prey
- Reward = 0, in all the other cases

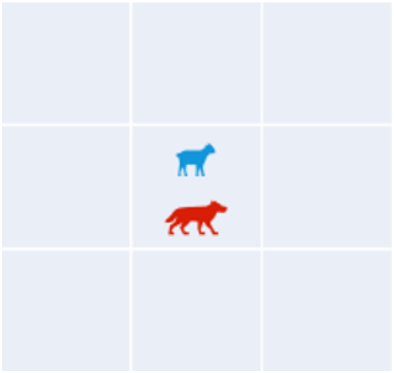


Figure 2: Example of goal state with 1 predator and 1 prey.

2.3 Model Training and Validation

We first test the Q-learning model proposed in the single agent setting to ensure it is adequately learning. The Q-learning algorithm will be executed with an epsilon greedy strategy for exploration. This is where an epsilon parameter is established and a random probability is generated. If the random probability is less than the set epsilon a random action is selected from the current state. Otherwise, the maximum Q-value from the state is selected and the corresponding action is selected. A learning rate (α) and a discount factor (γ) must also be defined. The parameters used in Q-model training are the same that were used in the paper. Table 2 shows all parameters used in Q-learning and their values.

Parameter	Value
α	0.1
γ	0.9
ε	0.1
Episodes	20000
Steps	5000
Environment grid size	10x10
Number of predators	1

Table 2: Parameters for training Q-learning model

An episode is composed of 5000 steps. In each step the predators determine their state, select and action and observe the reward. The episode ends when either

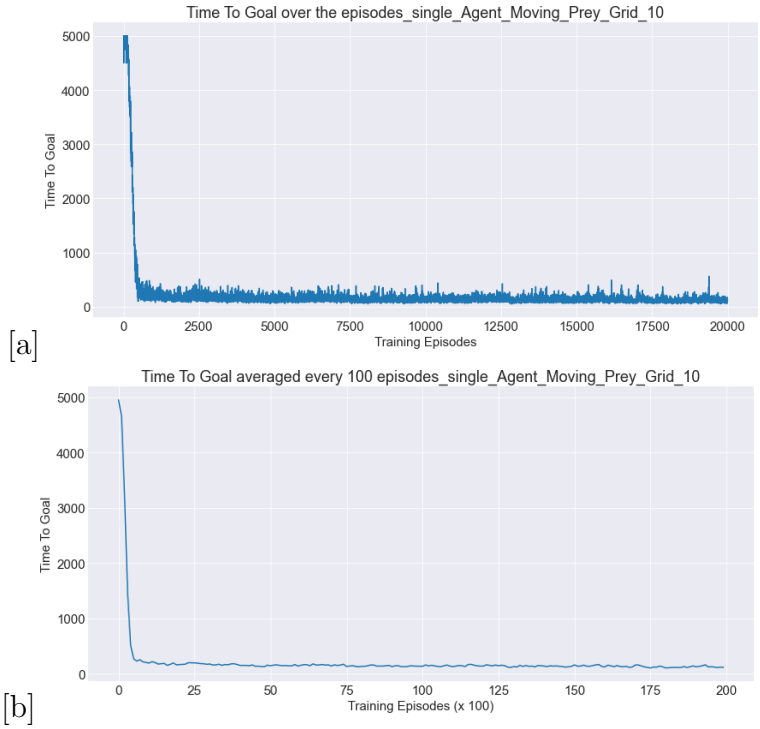


Figure 3: [a] Q-learning single agent time to goal over the 20000 training episodes [b] Q-learning single agent time to goal averaged every 100 training episodes

the 5000 steps are reached or the predators arrive to the goal state. The time to goal is the number of steps necessary to reach the goal state, or 5000 if the goal state was never reached. Every 100 episodes the time to goal for the 100 episodes is averaged and these averages are then plotted. This process can be repeated many times. To evaluate the learning of the Q-learning agent we repeated this process 20 times for the parameters in Table 2. The time to goal for these 20 runs is shown in Figure 3.

It is clear to see from Figure 3 that the Q-learning algorithm is able to learn to catch the prey. After around 6000 episodes the time to goal approached a steady value close to zero. The time to goal is then maintained over the remainder of the episodes. It can also be noted that in a grid size of 10x10 the number of states becomes 19^2, for a total of 361 states. We can extract the Q-table at the end of the training process to see that most of the states have been visited during training. With this initial test we can deem the Q-learning algorithm appropriate to be used in the multiple agent scenarios.

3 Simulation Design

In this section we explain how the model from Section 2 is used in simulations as well as the procedure to follow for each experiment proposed.

3.1 Multiple Independent Q-learners

With the validation of the single agent Q-learning algorithm, we can adapt the Q-learning model to multiple independent Q-learners. Independent Q-learning (IQL) is performed when each agent independently learns its own action-value Q function. That is to say, each agent keeps their own Q-table with their own independent Q-values. In this way, an agent forms part of the environment for the other agents in the system (more details in the paper "A Comprehensive Survey of Multi-agent Reinforcement Learning"[2]). The prey follows the same behaviour as the single agent case. For all simulations the number of agents in the system will be 2.

With two agents present in the environment the goal state of the system must be adapted. We consider the goal state (predators catching the prey) to be when one agent is in the same square as the prey and the other agent in an adjacent square. An example of the goal state is shown in Figure 4 with the 2 predators in red and the prey in blue.

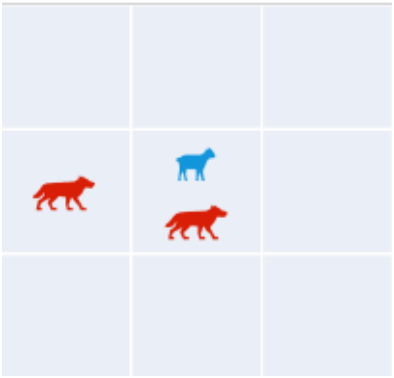


Figure 4: Example of goal state with 2 predators and 1 prey.

As the predators must work together to catch the prey, the same reward is returned to both agents (cooperative MARL setting [2]). The modified reward for the multi-agent case is as follows:

- Reward = 1, if one agent is in the same grid space as the prey and the other agent is in an adjacent grid space
- Reward = 0, in all the other cases

The actions available to each agent are the same as in the single agent case described in Table 1, and the same precautions are taken for agents or prey that exceed the limits of the grid world.

The state for each agent must be adapted for the multi-agent case, as the other agents in the system form part of the environment. The state can then be comprised of two sets of relative coordinates described by a tuple (x1, y1, x2, y2). The first set of coordinates contains the relative distance from one agent to the other. The second set of coordinates contains the relative positions of the current predator to the prey. Negative positional values are obtained when the current agent is below or to the left of the other agent or prey. The total number of states then depends on the grid size of the world. For example, in a grid world of size 4x4 there are 7 possible values: -3, -2, -1, 0, 1, 2, 3. The number of states is then the combination of these possibilities: 7^4, for a total of 2401 states. It is clear to see that for the same grid size the number of states significantly increases in the multi-agent scenario compared to the single agent scenario.

3.2 Partaker-Sharer Advising Framework

The authors in the considered paper[1] propose a framework based on Q-learning for communication between agents in order to decrease learning time called *partaker-sharer advising framework* (PSAF). The communication is structured in the form of partaker-sharer communication framework. The agents in question can choose to adopt a partaker role or sharer role to request or share information respectively.

When an agent adopts the partaker role, it deems that its current state has not been sufficiently explored and chooses to poll the other agents in the system for their maximum Q-value in that state. When an agent adopts the sharer role it evaluates its confidence level and shares the Q-value, if appropriate. In addition, two budgets are proposed for each agent: ask budget (for partaker role) and give budget (for sharer role).

To further understand each role, the algorithm for the partaker role is shown in Figure 5. The partaker role requires additional inputs: the $E^i_{degree}(s)$ which maps a state to the degree of exploration shown in Equation (1), and the probability of asking function shown in Equation (2) where v_p is a predefined parameter with a value of 0.7 for the experiments (as in the paper[1]).

$$E^i_{degree}(s) = \sqrt{n^i_{visit}(s)} \tag{1}$$

$$P^i_{ask}(s, E^i_{degree}) = (1 + v_p)^{-E^i_{degree}(s)} \tag{2}$$

For a given time step the agent begins the Q-learning process by evaluating the current state that the agent is in. Once the state has been established in

Algorithm 1 an agent i asks for and uses Q-values

Require: agent i, function E^i_{degree}, function P^i_{ask}, budget b^i_{ask}.

1: **for** each time step **do**
2: let current state be as s_i
3: **if** $b^i_{ask} > 0$ **then**
4: $P \leftarrow P^i_{ask}(s_i, E^i_{degree})$
5: $p \leftarrow getRandomNumber(0, 1)$
6: **if** $p < P$ **then**
7: **for** each agent in $G(s_i)$ **do**
8: each agent as a sharer provides its Q-value
9: **end for**
10: denote Π as the set of shared Q-values
11: **if** $\Pi \neq \emptyset$ **then**
12: $b^i_{ask} \leftarrow b^i_{ask} - 1$
13: $\Pi = \Gamma(\Pi)$
14: **for** each Q-value of Π **do**
15: denote $Q_j(s_i, a^*_j)$ as the Q-value of sharer j
16: $Q_i(s_i, a^*_j) \leftarrow Q_j(s_i, a^*_j)$
17: **end for**
18: execute greedy exploration strategy
19: **end if**
20: **end if**
21: **end if**
22: **if** no action is executed **then**
23: perform usual exploration strategy (i.e., ϵ-greedy)
24: **end if**
25: **end for**

Figure 5: Algorithm for agent adopting partaker role.

line 2 of Figure 5, the asking budget is checked. If the budget has not been all spent, a probability is generated in line 4 using (2). The probability of asking is designed such that, the more visits that are made to the specific state the lower the probability of asking will be. The idea being that the agents asks for Q-values when the state has not been sufficiently explored. In line 5 a random number is also generated between 0 and 1 and the values of the calculated probability is compared. If the random number is less than the probability, the sharer agents are asked to provide their q-values for the corresponding state the partaker agent is currently in. The returned Q-values are stored in line 10. If q-values have been returned then the asking budget is reduced (line 12), and the maximum q-value returned becomes the selected q-value (line 13). Then in line 16 the shared q-value is written into the partakers q-table at its current state and to the corresponding action in the sharers Q-table. It is important to note that the sharer provides the

q-value to be updated in the partakers Q-table and does not necessarily provide the partaker agent with the action the partaker agent should select: indeed, if the partaker receives a q-value, he selects the next action by applying a greedy policy. If the partaker process is not able to be followed, the partaker performs regular Q-learning with an epsilon greedy strategy as outlined in the single agent case.

Similarly, the algorithm for the sharer agent is shown in Figure 6. It is necessary for the partaker agent and sharer agent to define a confidence level for the state of the partaker agent. The confidence level of the partaker is calculated using (3) where m describes the number of times the state-action pair has been updated in the Q-table, i.e. the number of times the partaker has visited that pair. The confidence level of the sharer is calculated using an additional discrimination parameter (4). The equation used for the sharers confidence level is shown in (5).

$$C_p^i(s,a) = m_{visit}^i(s,a) \tag{3}$$

$$D^j(s) = 1 - \frac{1}{max_a Q_j(s,a) - min_a Q_j(s,a) + 1} \tag{4}$$

$$C^j(s,a^*) = m_{visit}^j(s,a^*) \times D^j(s) \tag{5}$$

From (5) we can notice that the sharers confidence exceeds the one of the partaker only if the sharer has visited many times the state-action pair and this is due to the (4), which is normalized between 0 and 1. The sharer agent then observes the state in line 1 and consults its give budget in line 2. If there is budget to share a Q-value, the sharer saves the maximum Q-value of that state (line 3). The sharer then compares the confidence level of the partaker agent to its own. If the sharer agent has a higher confidence level then it reduces the budget (line 5) and return the maximum Q-value to the partaker agent (line 6).

3.3 Experiment Designs

To understand the results shown in the paper[1], we propose a series of tests to recreate and further analyze the parameters in the system and their effects on the time to goal, which is the metrics of performance we take into consideration. It should be noted that due to the large run time of many of the experiments some of the tests performed are not identical to those performed in the paper[1]. In these cases the number of times the tests were repeated was reduced. If this was the case it is noted in the description of the test.

The first test proposed, *Test 1*, is to compare the communication and learning in the PSAF in a scenario with a moving prey and a non-moving prey. In the moving prey scenario, as outlined in previous sections, the prey can select an

Algorithm 2 an agent j provides its maximum Q-value

Require: agent j, function C_s^j, budget b_{give}^j, the partaker's state s_i, function C_p^i.

1: agent j observes state s_i
2: **if** $b_{give}^j > 0$ **then**
3: $\quad a_j^* \leftarrow \arg\max_a Q_j(s_i, a)$
4: $\quad$ **if** $C_s^j(s_i, a_j^*) > C_p^i(s_i, a_j^*)$ **then**
5: $\quad\quad b_{give}^j \leftarrow b_{give}^j - 1$
6: $\quad\quad$ return $a_j^*, Q_j(s_i, a_j^*)$
7: $\quad$ **end if**
8: **end if**

Figure 6: Algorithm for agent when adopting sharer role.

evasive action 80% of the time and chooses a random action 20% of the time. This creates some complexity for the Q-learning as the position of the goal state is dynamic. Additionally this could create differences in the Q-tables between the two independent agents. This could place an important emphasis on the communication between agents to arrive to the goal state more quickly. When the prey is stationary, it is initialized at the beginning of an episode and remains in that grid position until the next episode. By keeping the prey stationary the importance of communication could be much more reduced. This test is preformed in a grid world of size 12 with 2 predator agents that t communicate with one another. The learning parameters are maintained from 2 as well as the the number of episodes and steps. This test however will be performed 20 times, taking the average of all 20 runs.

The second test proposed, *Test 2*, is for comparing multiple agents adopting the *partaker-sharer advising framework* and simple independent Q-learning agents. The goal for this test is to observe some difference in the time to goal between the two methods. The PSAF should provide some decrease in learning time as the two agents are able to share information learned. This test will be produced with 2 predator agents and 1 prey that is able to take evasive actions as in the moving prey scenario from *Test 1*. The learning parameters are identical to the previous tests, as well as the number of steps and episodes. The test is run twice: once on a grid size of 10x10 and another on a grid size of 12x12. For each grid size, the experiment is repeated a total of 20 times and the response is averaged together. In the paper[1] this experiment is repeated 200 times. However, due to the large run times this was reduced to allow for other experiments to be performed as well.

The third test proposed, *Test 3*, is based on the previous test, with an increased grid size. Here we increase the grid size to a 12x12 grid to observe the effect com-

munication has on more complex environments. In a grid size of 10x10 there are 19 possible relative positions, meaning there are $19^4 = 130321$ possible states. In a grid size of 12x12 there are 23 possible relative positions, meaning there are almost double the amount of states than with the smaller grid size: $23^4 = 279841$ possible states. The goal of this test is to observe if communication is more important at larger grid sizes or if the grid size does not play a difference in communication. This test will be performed with the same learning parameters as tests 1 and 2, as well as the same number of episodes, steps, and agents. Importantly we will be comparing a grid size of 10 and 12 repeating each experiment 20 times.

The final test proposed, *Test 4*, explores the effect of budget size on PSAF learning. By considering smaller budgets, we can observe the effect it has on learning by how closely it approached the multiple IQL scenario. We can also observe if it is more important TO have budget throughout the entire learning process or if having budget is more important at the beginning of the training episodes when more learning is taking place. This test will be preformed using the same learning parameters as the previous tests as well as the same number of episodes and steps. This test will be performed at a grid size of 10x10 and repeated 20 times. The result of the 20 repetitions can then be averaged and the result plotted.

A summary of all four experiments and their descriptions is shown in Table 3

Proposed Test	Description
Test 1	Moving prey vs non-moving prey
Test 2	PSAF vs Multiple IQL
Test 3	Alternate grid size 12x12
Test 4	Reduced ask budget and give budget

Table 3: Summary of performed experiments

3.4 Python Implementation

This project was implemented in Python. In particular a main project folder *SOAS_project* consists of 2 identically structured sub-folders containing Python files:

- *one_agent*: contains code for the single agent/predator model
- *two_agent*: contains code for the model with 2 predators

As stated before, the structure inside each folder is exactly the same. We'll briefly explain how the files are arranged inside the *two_agent* folder, which con-

tains the most interesting code for what regards the implementation of the multi-agent reinforcement learning algorithm proposed in the analyzed paper[1].

Inside this folder we find the following Python scripts:

- a script for each type of environment we analyzed during our tests:
  - *environment_comm.py*, which defines the "environment" and "agent" Python classes for the case with a fixed prey and partaker-sharer communication
  - *environment_comm_prey.py*, which defines the "environment" and "agent" Python classes for the case with a moving prey and partaker-sharer communication
  - *environmnent_no_comm.py*, which defines the "environment" and "agent" Python classes for the case with a fixed prey and no communication
  - *environmnent_no_comm_prey.py*, which defines the "environment" and "agent" Python classes for the case with a moving prey and no communication
- *utils.py*, which contains the definitions of the main functions performing training simulations and plots (functions and their parameters are described as comments)
- *main.py*, which contains the main program that calls the functions defined in the *utils.py*

The folder *one_agent* is structured following the same structure, but the environment files are related to the single agent case, both with moving and fixed prey.

3.4.1 Additional details

As explained previously, we used different scripts for each type of environment. In particular, each environment script contains the definition of 2 Python classes:

- *Environment*: defines the grid world environment (by specifying the grid size) with the agents and the predator. Each agent is an object of the corresponding *Agent* class.
- *Agent*: defines the Q-learning predator moving inside the *Environment*. Each agent object has its own Q-table, structured as a Python dictionary where keys are represented as the *hash* of the state coordinates (the tuple of relative coordinates we mentioned in 3.1). This allows to boost the performance of

the training procedure, since we initialize the Q-tables with empty dictionaries and, as soon as new states are visited, they are added to the dictionary data structure. Besides that, the usage of the hash function allows a fast access to the values inside the Q-table. In the case of PSAF agents, they also contain the budget as inner variables.

As a last detail, we would like to remark that the main function performing the transitions inside the environment is the *transition()* function of the *Environment* class, which allows to move the predators and the prey inside the grid, by calling the respective functions.

For further details we suggest to read the code of the *environment_comm_prey.py* script, which show the most complete case, with moving prey and PSAF agents. All the functions are carefully commented in order to allow a clear understanding of the code. A *README* file is also provided in the folder with the instructions for executing all simulations performed.

4 Results

In this section the results of the tests introduced in Section 3.3 are presented and discussed.

4.1 Test 1

In *Test 1* we consider both the time to goal and budget used for a moving prey and a non moving prey in the PSAF. The time to goal results are shown in plot [a] and the communication budgets are shown in [b] of Figure 7. It is important to note that only the budget of one agent is shown as they are symmetrical. That is to say the asking budget of one agent is equal to the give budget of the other agent and the give budget of one agent is equal to the ask budget of the other agent.

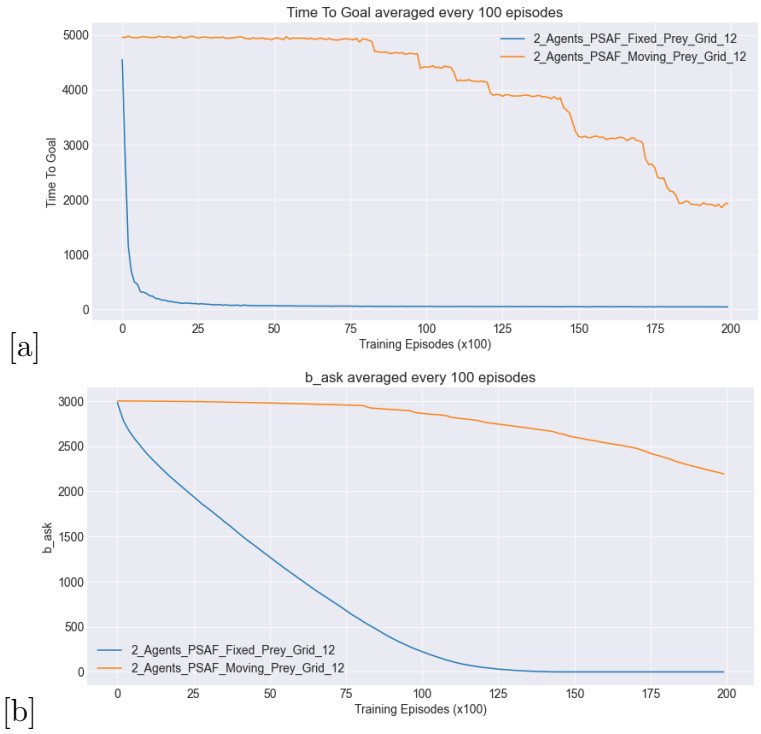


Figure 7: [a] time to goal for moving prey (orange) and stationary prey (blue) [b] budget used for moving prey (orange) and stationary prey (blue)

From Figure 7 [a] it is clear from the curve (blue) that the stationary prey scenario is much less complex for the two agents as learning happens in about 2500 episodes. The time to goal for the stationary prey maintains a very low value for the rest of the episodes. Additionally, the curve is very smooth meaning that

the time to goal over the 20 runs were very consistent. Looking at the curve for the moving prey (orange) the same cannot be said. There are more variations present suggesting less consistency over the 20 runs. The learning in the moving prey scenario also seems to occur significantly after 7500 episodes. This slower learning is to be expected from the increased complexity of the scenarios.

In Figure 7 [b] there is also a clear difference between the stationary and moving prey scenarios. The stationary prey scenario uses significantly more budget than the moving prey scenario. This difference could be due to differences in the Q-tables of the two agents, and the fact that sharing is based on the confidence levels of the agents in that state. The agents must wait more episodes before they build up the confidence levels to begin sharing Q-values. Additionally, in the non-moving prey scenario the agents are able to learn extremely fast so the importance of the messages being sent between agents after they have learned is greatly reduced. Conversely, the budget starts to reduce significantly in the moving prey case at around the same amount of episodes the time to goal begins to reduce.

4.2 Test 2

This test seeks to compare the time to goal for the PSAF and IQL methods. It was performed at two different grid sizes.

Firstly we consider a grid size of 10x10. The average time to goal results for 20 repetitions are shown in Figure 8 with an additional plot showing the first 4000 episodes of the process. In plot [a] the time to goal clearly diminishes after 2500 episodes. The partaker-sharer method (orange) seems to learn faster than the IQL version. However, between 2500 and 5000 episodes the IQL version reaches the time to goal after, and then after 5000 episodes the responses are very similar. Additionally, the slope of the two methods seems to be fairly similar. To examine the time to goal more closely we can plot the first 4000 episodes in [b] where the time to goal for both clearly begins to diminish. The PSAF clearly begins learning after 1500 episodes where time to goal begins to decrease, where the IQL is slightly delayed and begins to diminish after 2000 episodes. It is important to note that this experiment was repeated 20 times, whereas in the paper (1) this experiment was repeated 200 times. It is possible that with more repetitions clearer differences between the methods could be seen.

Secondly we considered a grid size of 12x12. The average time to goal results for 20 repetitions are shown in Figure 9 with an additional plot showing the first 14000 episodes of the process. The differences between the time to goal in Figure 9 using the PSAF and not are much clearer at this larger grid size. The PSAF (orange) seems to learn much faster. This would suggest that communication is more important at larger grid sizes for speeding up learning.

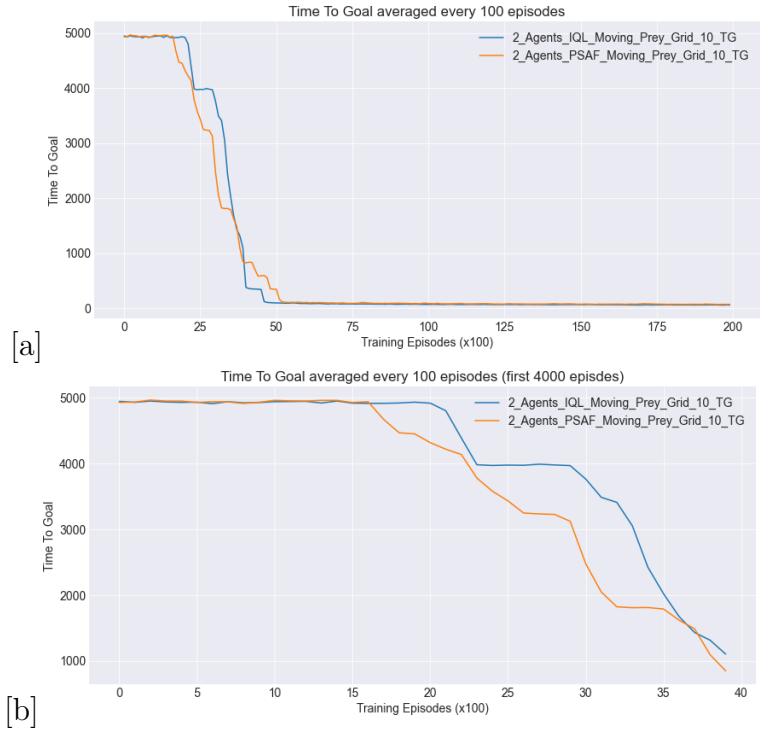


Figure 8: [a] time to goal for the PSAF (orange) IQL (blue) in a 10x10 grid [b] a zoom of the time to goal response for the first 4000 episodes in a 10x10 grid

4.3 Test 3

From the previous test it is clear that using a larger grid size increases the complexity of the Q-learning problem. To further examine this we can plot the time to goal for the PSAF at a grid size of 10x10 and 12x12 as well as the resulting budgets. The averaged response is plotted in Figure 10.

Figure 10 [a] shows the time to goal at the two different grid sizes. There is a clear difference between the time to goal of the two grid sizes. The smaller grid size of 10x10 starts reducing the time to goal sooner than the larger grid size 12x12 - about 5000 episodes before. This makes sense are there are over double the amounts of states in the 12x12 grid than the 10x10. Additionally, the slope of the time to goal for the smaller grid size is much steeper than the larger grid size. This makes sense as the larger gird size presents many more states and so learning time is longer.

Figure 10 [b] shows the budget used for both the 10x10 (blue) and 12x12 (orange) grid sizes. The larger grid sizes uses far fewer messages in the 20000 episodes than the smaller grid size. This makes sense when considering the learning time and the amount of states. As the the larger grid takes longer to learn, the

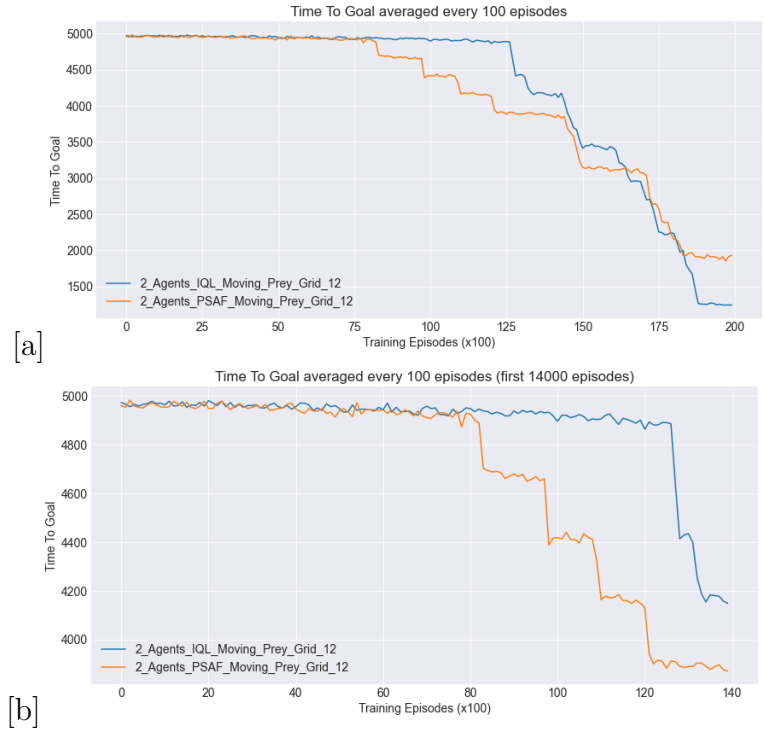


Figure 9: [a] time to goal for the PSAF (orange) IQL (blue) in a 12x12 grid [b] a zoom of the time to goal response for the first 14000 episodes in a 12x12 grid

budget is used more slowly as the agents must build up confidence at the states in order to share meaningful Q-values: in a large grid size they need more exploration to explore the states. For both scenarios it is clear that when the time to goal begins to diminish, so does the budget. For the larger grid size this requires more episodes.

4.4 Test 4

In this test, we observe the time to goal and budget for 2 partaker sharer agents with a grid size of 10x10 and asking/sharing budgets of 3000, 1000, and 500. These results are plotted in Figure 11.

Figure 11 [a] shows the time to goal for each of the different budgets considered. All three budgets seem to achieve a very similar time to goal learning pattern. Additionally, their slopes seem to be fairly similar. There is very little difference between the budget of 1000 and 500, where they both seem to begin learning at the same time and reach a stable time to goal at a similar number of episodes as well.

Figure 11 [b] shows how these budgets are used over the episodes. Similarly to

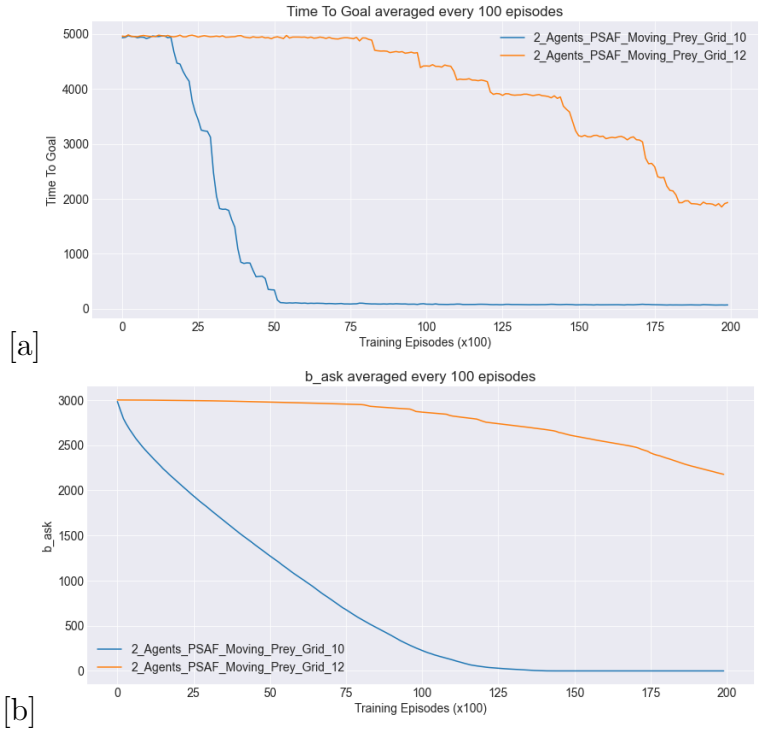
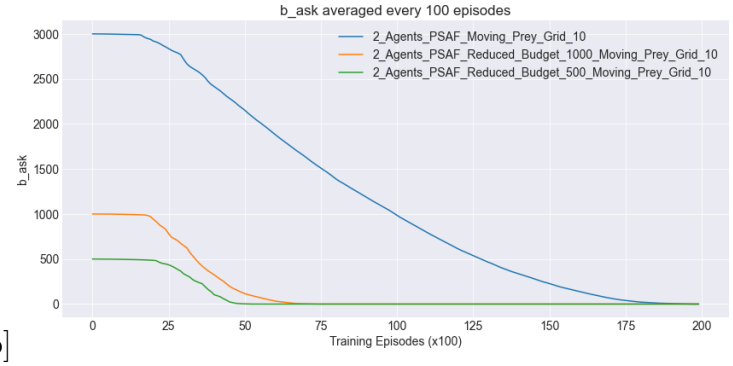


Figure 10: [a] time to goal for grid size 10x10 (blue) and 12x12 grid (orange) [b] budget for grid size 10x10 (blue) and grid size 12x12 (orange)

the time to goal, all three budgets follow a very similar pattern, starting to reduce at a similar number of episodes and following a very similar slope. The smaller budgets are used up before the larger budgets which makes sense considering that they are smaller.

[a]



[b]

Figure 11: [a] Time to goal of budget 3000 (blue), 1000 (yellow), and 500 (green) [b] budget used where maximum budget 3000 (blue), 1000 (yellow) and 500 (green)

5 Conclusions

From Test 1 it is clear that the moving prey scenario is much more complex than the fixed prey scenario. Although the budget of the fixed prey scenario is used over the 20000 episodes, from the time goal it appears that learning occurs much before the budget is depleted. This could suggest that in a low complexity scenario, such as the one with a fixed prey, communication between agents is not necessarily important or useful as they are able to learn very quickly using minimal budget.

From test 2 it is clear that in more complex scenarios, such as those with larger grid sizes, communication becomes more important in reducing the learning time. This was seen with larger difference between the partaker-sharer method vs the IQL method at a grid size of 12x12 compared to a smaller grid size of 10x10. This makes sense as increasing the grid size dramatically increases the number of states that must be explored. By relying on the PSAF that considers the confidence levels of the agents, they are able to share quality information that decreases the learning time.

From Test 3 it is clear to see that the communication has importance in both small and large grid sizes. The communication is reliant on the confidence level that agent has at a specific state. As such it stands to reason that when there are more states to explore the sharing of Q-values happens later with respect to the case with a grid size that presents less states to explore. However, when examining the slopes of the time to goal and budgets the smaller grid size presents a steeper slope, suggesting that learning does in fact happen faster at a smaller grid size.

Considering different communication budgets of 3000, 1000, and 500, in Test 4 it was very difficult to see differences reflected in the time to goal and almost no difference in the budget usage. Indeed the budget starts to be used for all the cases in the same episodes and it follows the same decreasing pattern. A possible conclusion we can draw (even if we repeated the training only for 20 iterations) is that the budget is much more important at the very beginning of the communication process, that is when the agents have enough confidence to start exchanging Q-values. During this phase, the communication allows to reduce earlier the time to goal with respect to the multi-IQL case (as we've seen in Test 2).

Overall, we were able to successfully replicate the experiments performed in the predator-prey scenario. As well as propose new tests to further understand the authors results. The main difference between this work and the paper 1 was the number of repetitions we were able to perform given the time constraints.

Our work can be further extended by repeating the tests (trainings) 200 times and taking the average results. Additionally, this framework can be applied to more complex scenarios for example, with more than two agents.

5.1 Personal Conclusions

This project allowed both students to practice good teamwork and good communication. Meetings occurred on a weekly basis as well as several phone calls. In this way, both students were able to contribute equally to the project while taking advantage of each students strengths. This topic of work was new to both students and provided a great challenge to implement. By breaking it down into sub-tasks both students were able to learn a great deal about multi-agent reinforcement learning.

6 Code and demo

- The full Python code is also available at the following *github* repository: `https://github.com/denaldo98/SOAS_project`
- A video-demo of the simulation of the PSAF setting with moving prey is available at: `https://youtu.be/UpeMG3ep6Y8`