Assignment 3: Computational Game Theory

BOUSSANT-ROUX Luc / ULB / 000470709

22 December 2018

1 N-Armed Bandit

For the duration of this exercise, we will use table 1 for the reward distributions of each of the 4 actions

1.1 Exercise 1

We can see in figure 1 that the algorithm with the highest average reward is "Greedy 0.1". We can also see that most of the algorithms have a low standard deviation except for the "Greedy 0" and "Softmax 0.1". We can explain this phenomenon by looking at the algorithms more closely. Indeed, if we look at "Greedy 0", this means that we always take the action with the highest Q. However, at first all of the different Qs will be equal to 0 but after one action is chosen randomly the Q of this action is higher than 0. Thus, for every step after that the action will be chosen again. Since we choose the first action randomly, the average reward is then very scattered.

In the figure 2 there is no standard deviation, it is the graph of only the last experiment.

We can see in the figure 2 the rewards distribution of the different Actions. Indeed, in the graph for Action 4 you can see the dispersion is greater than for the other Actions. This is because the sigma is much higher for action 4 compared to the rest. We can also see that the final mean for all of the actions is close to the mu of the rewards distribution.

When we look at figure 3, we can see 3 different type of graph. Indeed, if we look at the graphs "Greedy 0.1", "Greedy 0.2" and "Softmax 1", we see that action 1 is the one that has been chosen the most, then action 2, action 3 and lastly action 4. This is normal because this order is the one from highest to lowest reward in average. We can also see a graph with the random method where the average is 250 ± 15 which is expected in a 1000 steps experiment. Finally, for the "Softmax 0.1" and "Greedy 0" we can see that the average for each action is very close to 250. However, the standard deviation is very high for these ones this is because of what has been explained above. The number of actions for an experiment is very high for one and low for the others.

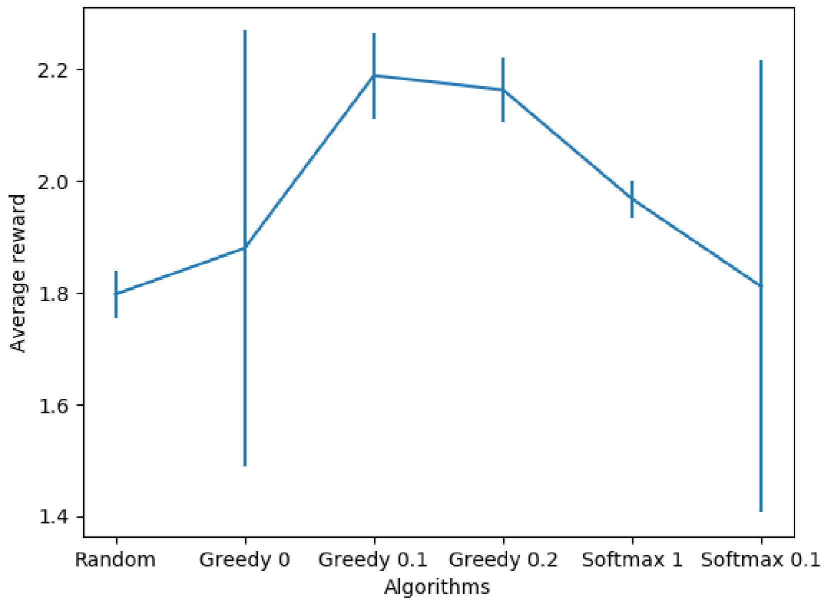

FIGURE 1 – Average Reward for each algorithm

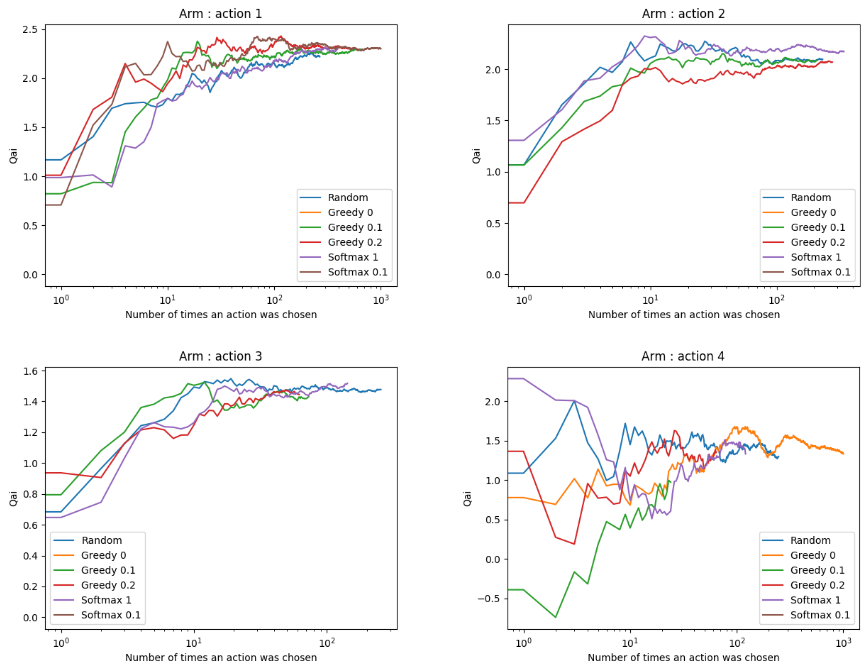

FIGURE 2 – $Q^*_{a_i}$ of that action along with the actual Q_{a_i} estimate over time

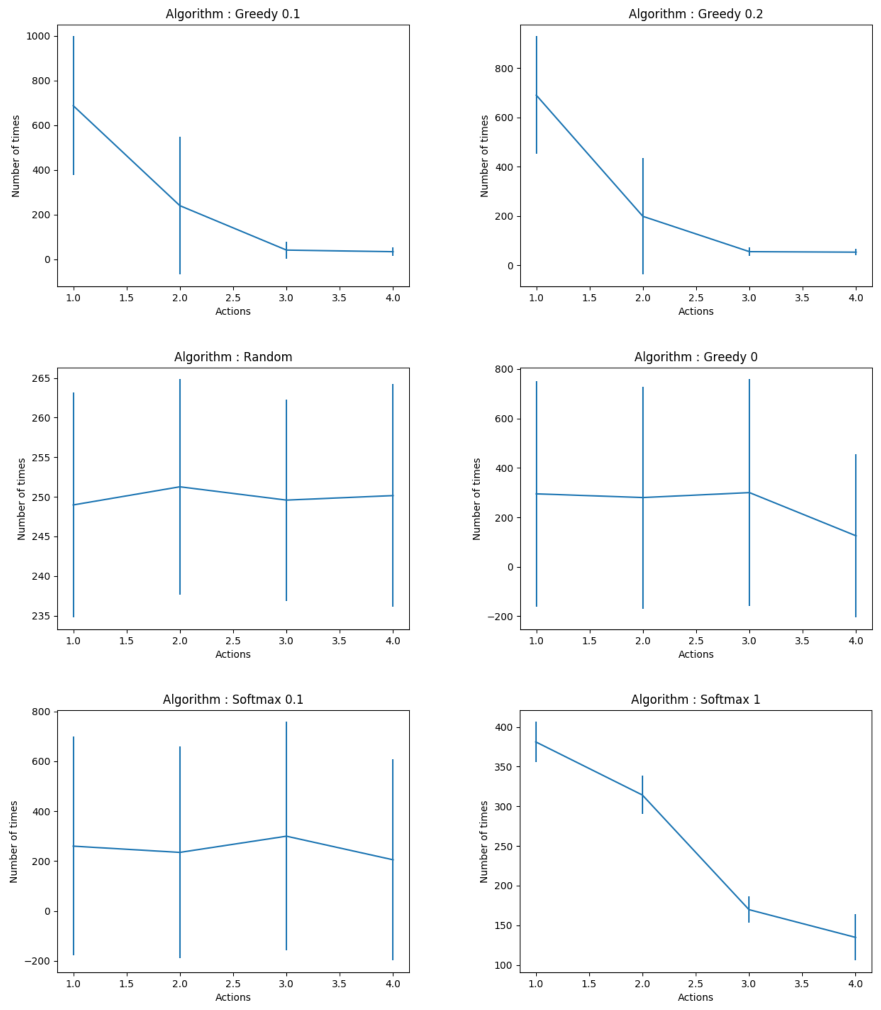

FIGURE 3 – Histograms showing the number of times each action is selected

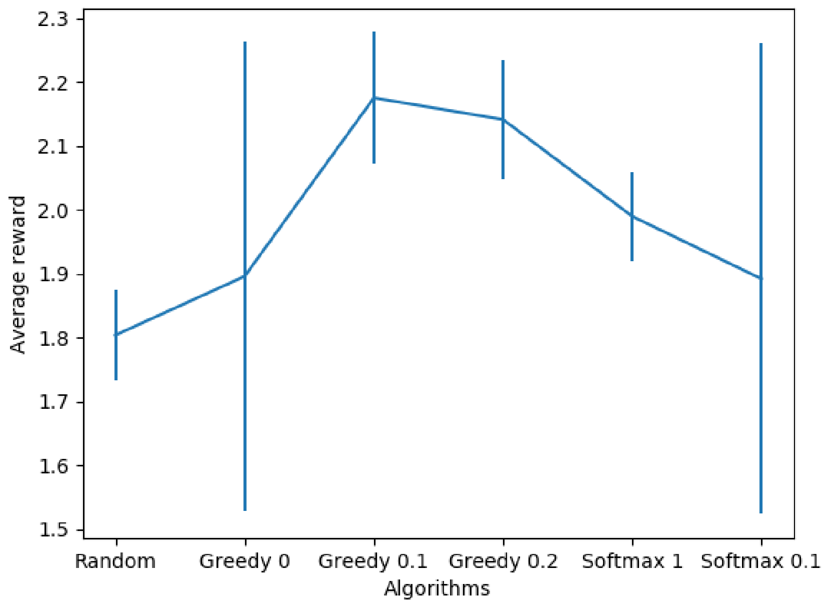

FIGURE 4 – Average Reward for each algorithm

To conclude, we can see that the algorithms that arrive close to the best arm after 1000 iterations during all of the experiments are : "Greedy 0.1", "Greedy 0.2", "Softmax 1". The algorithms that arrives the fastest are the ones with the highest average reward at the end (figure 1. Indeed, the fastest one is "Greedy 0.1", then "Greedy 0.2" and lastly "Softmax 1".

1.2 Exercise 2

After doubling the standard deviation of the reward distribution, we can see that most graphs have an higher standard deviation also (figure 4 and 6). In figure 5, we can see that the Q_{a_i} are more scattered than before.

We can also see that the same algorithms reach the best arm at the end of 1000 iterations but they take more time to do so. This can be explained by the closeness of Action 1 and Action 2 average reward and with the standard deviation doubled they seem to have an even closer reward. However, regarding the "Softmax 1" method, it seems that the time it took to find the best arm isn't longer than before.

To conclude, the problem is then a bit harder to learn because it takes more time to find a the best arm. The performance of the algorithms aren't affected much but the gap between the algorithms has decreased.

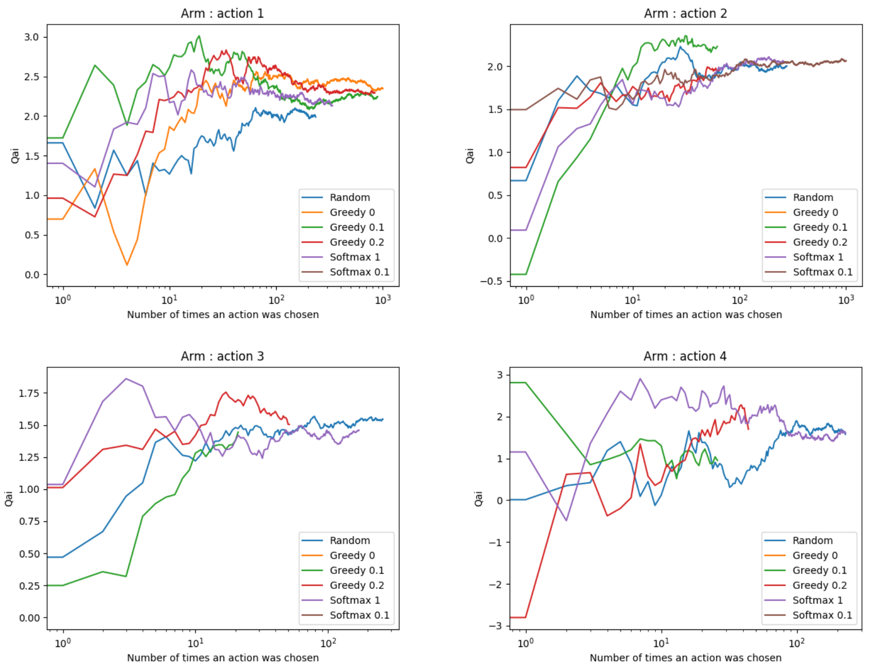

FIGURE 5 – $Q^*_{a_i}$ of that action along with the actual Q_{a_i} estimate over time

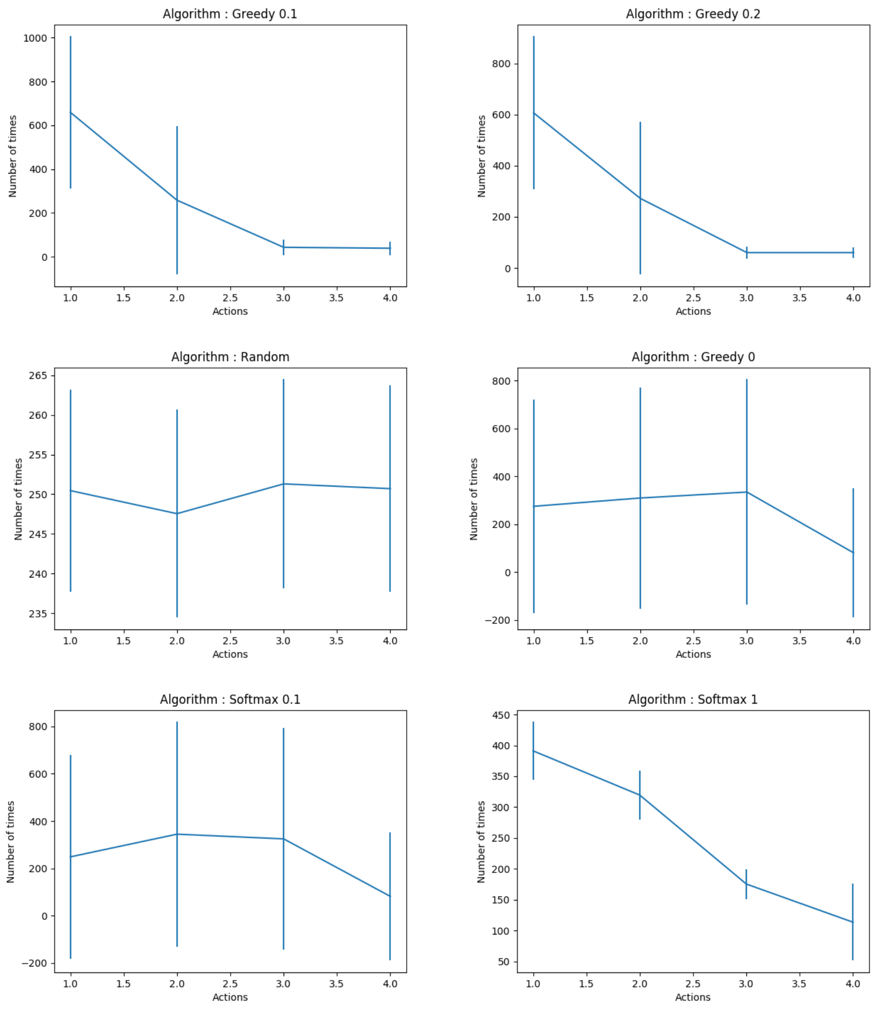

FIGURE 6 – Histograms showing the number of times each action is selected

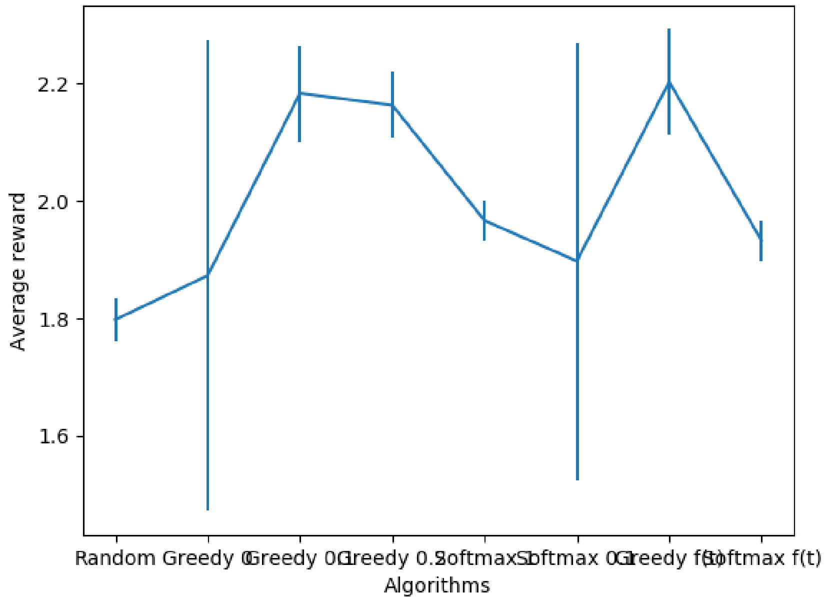

FIGURE 7 – Average Reward for each algorithm

1.3 Exercise 3

We showed only the graphs that changed between exercise 1 and exercise 3.

We can see that the algorithm "Greedy f(t)" is the one that is the fastest to find the best arm because it has the highest average reward (figure 7) and because when we look at the number of times each action has we can see that action 3 and 4 have is very close to 0 (figure 9). However, regarding the "Softmax f(t)" we can see that there isn't much difference with the "Softmax 1".

2 Windy Gridworld

We can see in figure 10 that the optimal path is more or less what we had expected with a wind of 2 in the 2 middle columns. Moreover, we can see that some of the grid was never visited because they have all of the actions as best. We can observe that the number of steps to get to the goal is inversely related to the total collected reward which could have been expected.

For the graph, we used a semi logarithmic scale to have a better sense of how the graph reaches convergence. We can see that after 100 iterations the system has learned the best path to take. However, as said above we have found out the best path to get to the goal but we haven't explored the whole grid. We have

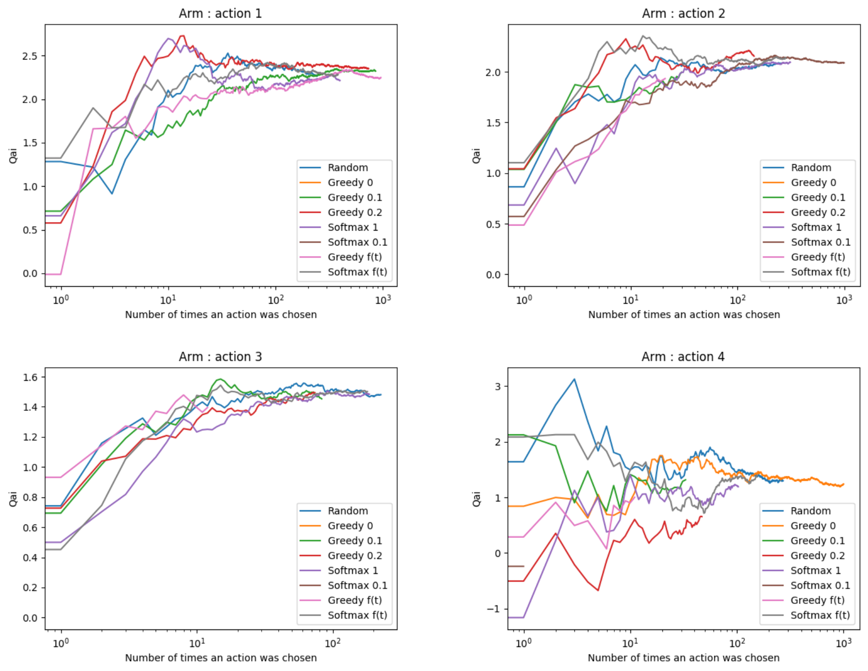

FIGURE 8 – $Q^*_{a_i}$ of that action along with the actual Q_{a_i} estimate over time

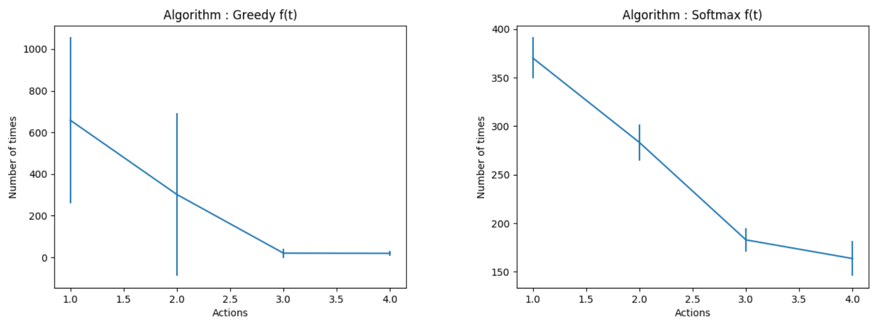

FIGURE 9 – Histograms showing the number of times each action is selected

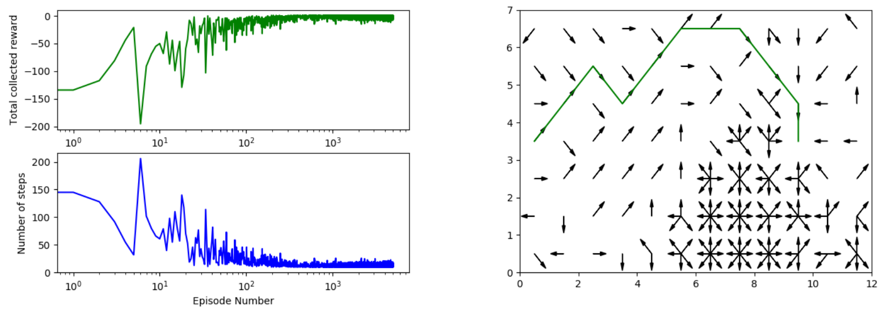

FIGURE 10 – WindyGrid world with $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.2$

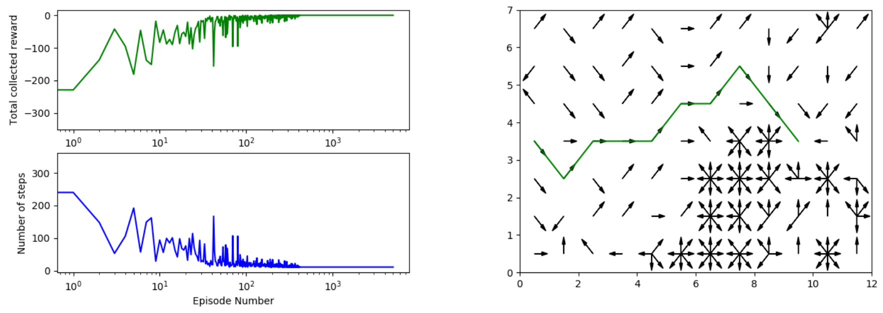

FIGURE 11 – WindyGrid world with $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0$

finished exploitation but not exploration.

2.1 Exercise 1

When we have $\epsilon = 0$, we can see that the number of steps it takes to reach the goal is more important at first but it reaches the optimal path only a bit slower than before. However, when it reaches the optimal path it never deviates from it this is because of the Totally Greedy method where we always choose the path with the highest Q. Finally, exploration is not more extended with these parameters. When $\epsilon = 0$, we give more importance to the current reward.

2.2 Exercise 2

When we have $\gamma = 1$, we can see that it takes as much time to get to the optimal path. However, the optimal path is reached with more disturbance. The collected reward for some episode are very low considering its number. The exploration is once again not much more important. When $\gamma = 1$, we give more importance to the long term reward.

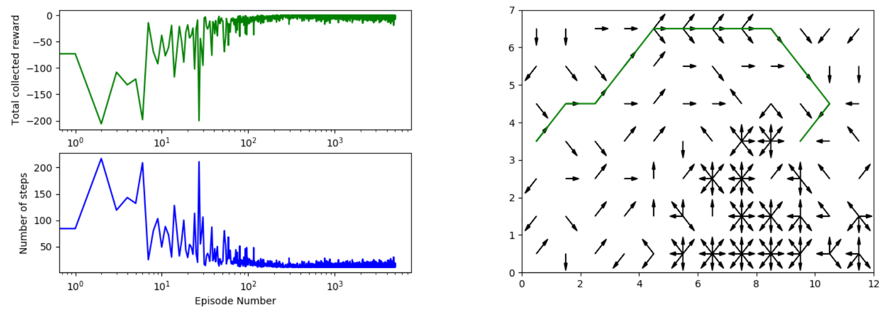

FIGURE 12 – WindyGrid world with $\alpha = 0.1$, $\gamma = 1$ and $\epsilon = 0.2$

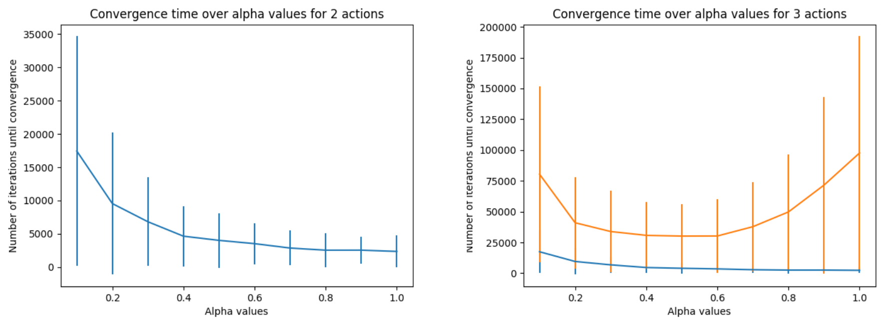

FIGURE 13 – Graphical Coordination

3 Graphical Coordination Game

3.1 Exercise 1

The payoff matrices for the coordination games are :

	Action 1	Action 2
Action 1	1	0
Action 2	0	1

	Action 1	Action 2	Action 3
Action 1	1	0	0
Action 2	0	1	0
Action 3	0	0	1

In figure 13, we can see that the standard deviation for 3 actions is much higher than the standard deviation for 2. Moreover, we can see that the convergence time is dependent on the values of alpha.

The role of α is to balance the moments when a player keeps his action or changes it. If alpha is low then the player changes his action often and when it is high the player rarely changes his action. This only happens if the two players that are being compared have chosen 2 different actions.

With my settings, the fastest convergence time with 2 actions is reached with the following α values 0.8, 0.9, 1 and, with 3 actions, 0.3 to 0.6. If $\alpha = 0$, then the algorithm fails to converge. Finally, we can also see that the number of actions affects the convergence time a lot because it is multiplied by 2.