Jazel A. Suguitan
Project 2 - CS 455

Requirement 1 - Algorithms
Matlab R2019a
Independent Q-Learning Algorithm: proj2_ind.m
Cooperative Q-Learning Algorithm: proj2_coop.m

# of episodes = 6
# of nodes = 10
time step = 0.003
initial epsilon for epsilon-greedy policy = 0.0001
desired distance of nodes = 30
epsilon for sigma norm = 0.1
alpha = 0.2
gamma = 0.9
c1_alpha = 5.6
c1_mt = 1.1

initial w for cooperative learning = 0.5

Note: The nodes are overshooting their targets, so it can be difficult to visually determine which safe place each node is fleeing to. However, printing out the actions a_t for all nodes in each iteration shows that a) the nodes start cooperative Q learning by choosing different actions, and b) all nodes eventually converge to the same safe place/action, usually action 1 [540, 400].

- Sample actions for nodes at the beginning of cooperative Q learning:

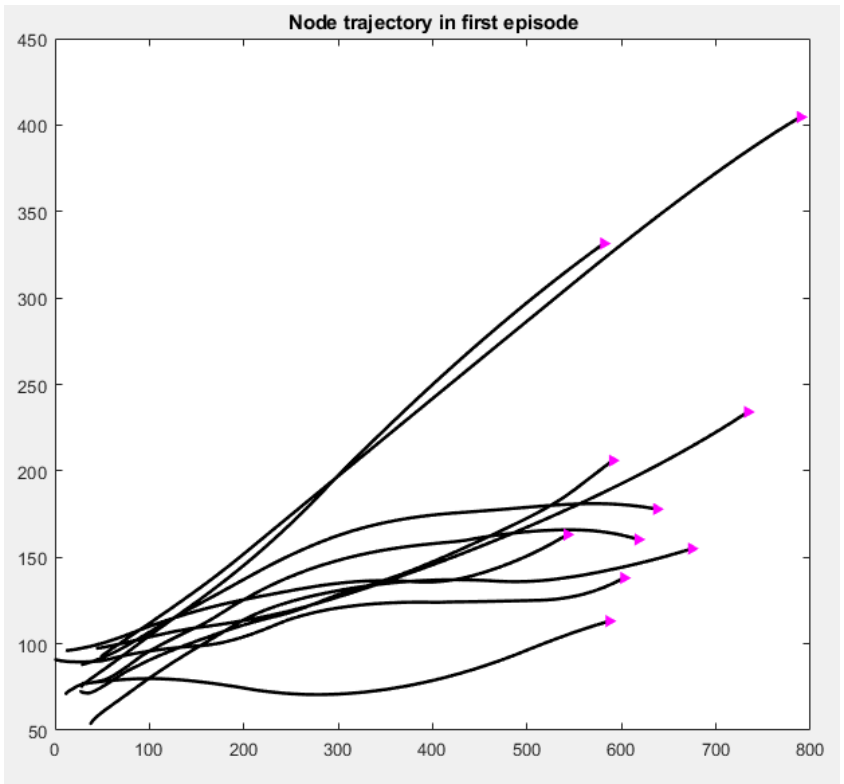
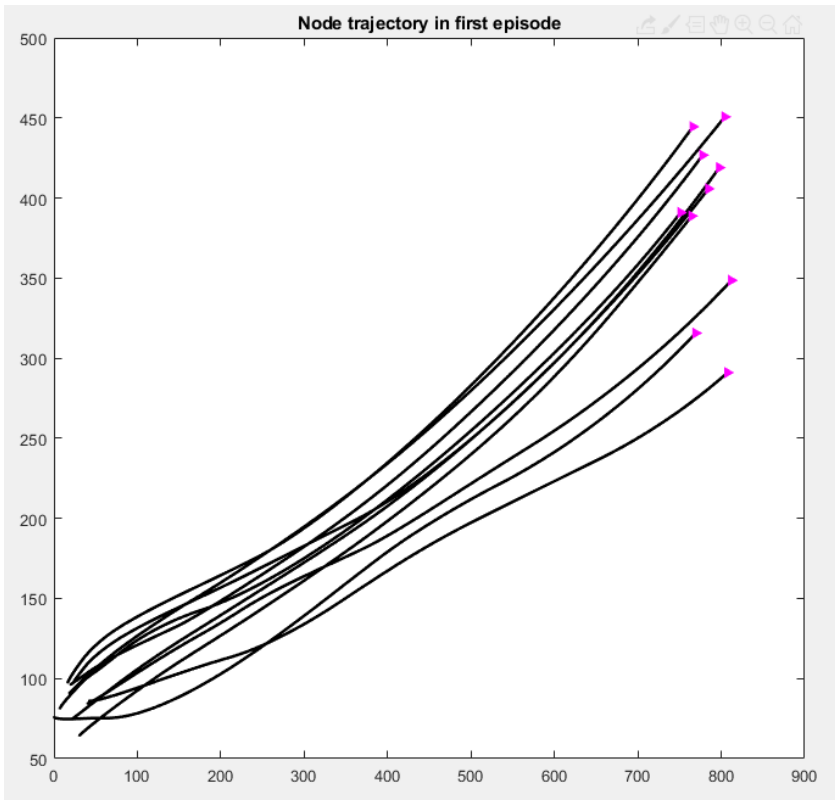
```
a_t =

     2     2     2     4     4     4     4     3     4     3


a_t =

     1     4     4     2     3     4     4     3     4     3


a_t =

     2     2     2     4     4     4     4     3     4     3
```

- Sample actions for nodes later in the cooperative Q learning process:

```
a_t =

     1     1     1     1     1     1     1     1     1     1


a_t =

     1     1     1     1     1     1     1     1     1     1


a_t =

     1     1     1     1     1     1     1     1     1     1
```

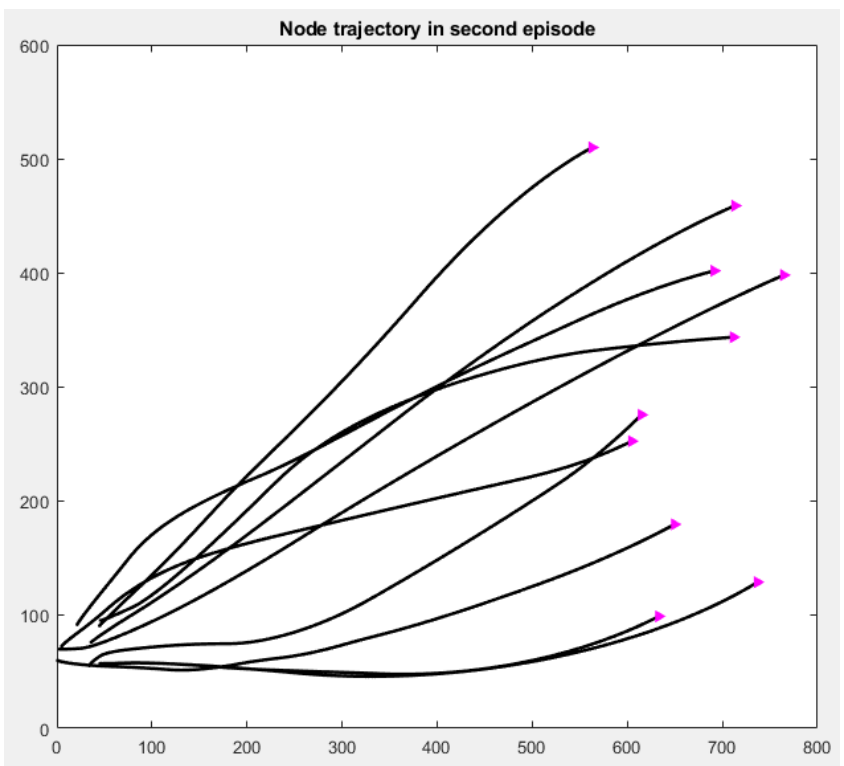
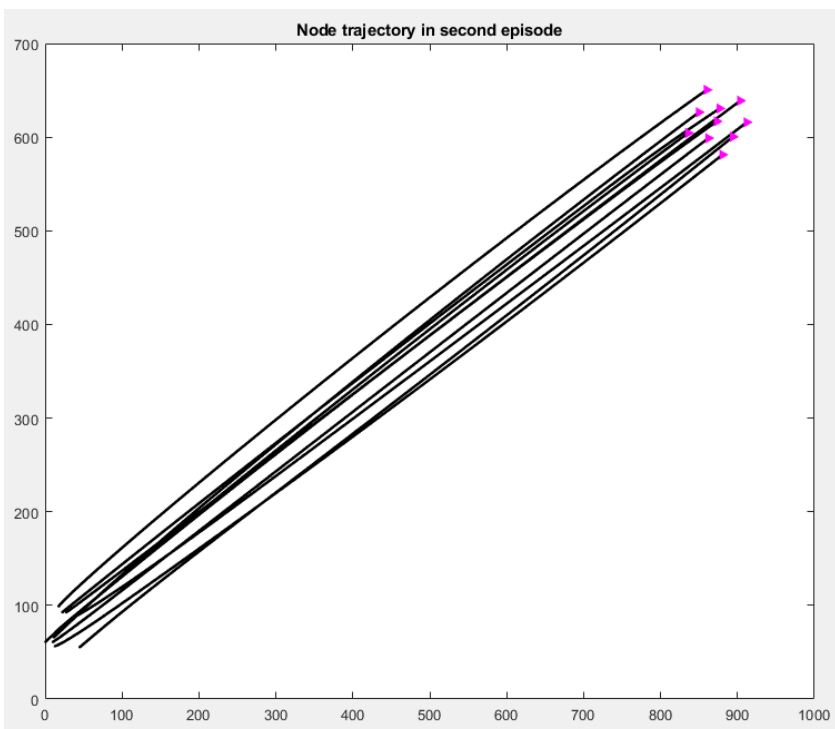
Requirement 2 - Node Trajectories

Episode 1

Independent	*Cooperative*
	

Here, the nodes in both algorithms do not flock very well, but unlike in the independent algorithm, the nodes in the cooperative algorithm seem to travel in the general same direction.

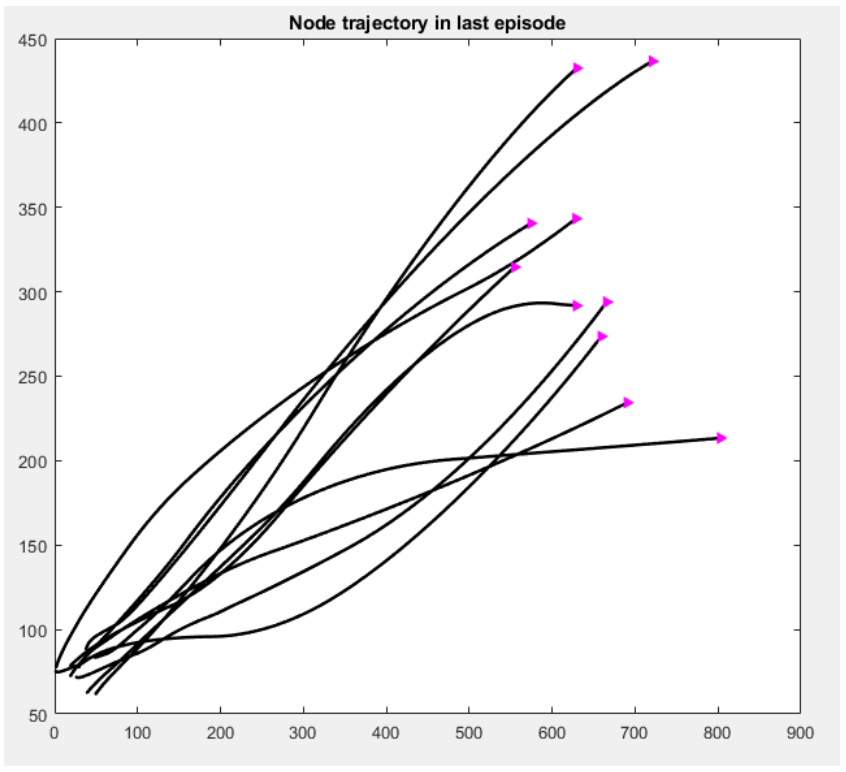
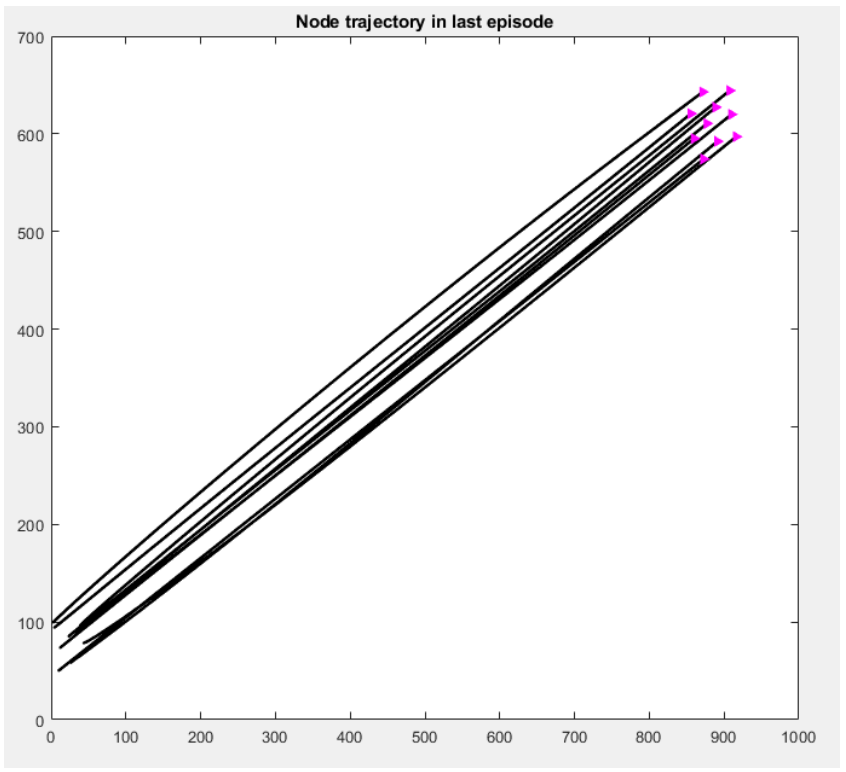
Episode 2

Independent	*Cooperative*
	

Here, the independent algorithm nodes are still scattered/not flocking, while the cooperative algorithm nodes are much more uniform in their flocking and action selection.

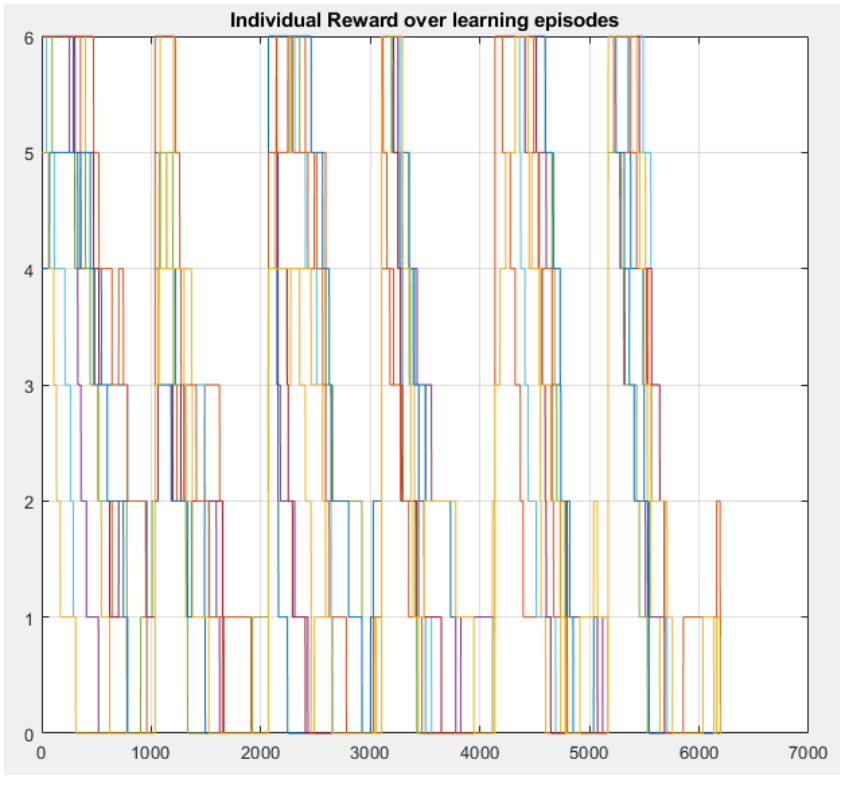
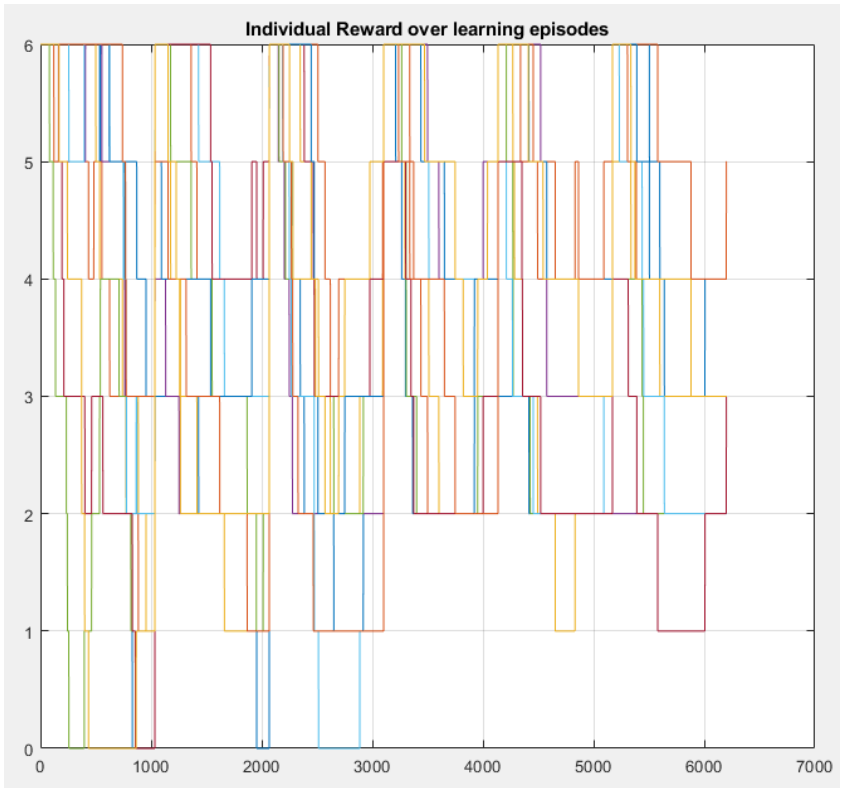
Final Episode

Independent	*Cooperative*

Here, the independent algorithm nodes have not yet decided on a single action. Meanwhile, the cooperative algorithm nodes travel in a tight lattice directly to the safe place deemed most optimal over the course of their Q learning.

Requirement 3 - Individual Reward

Independent	*Cooperative*
	

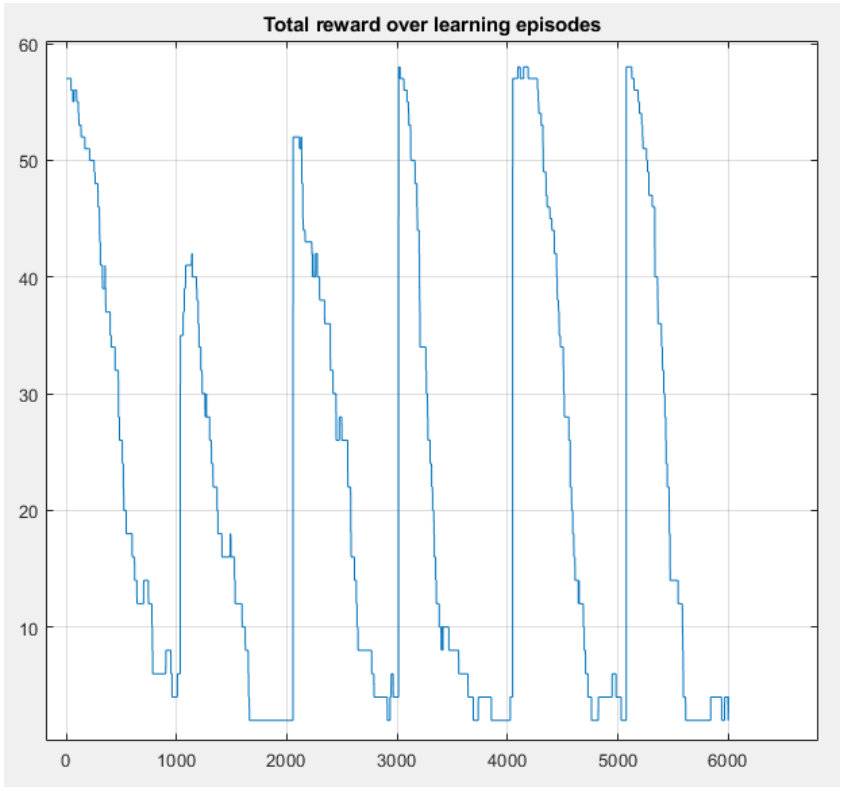
Reward is assigned in this manner:

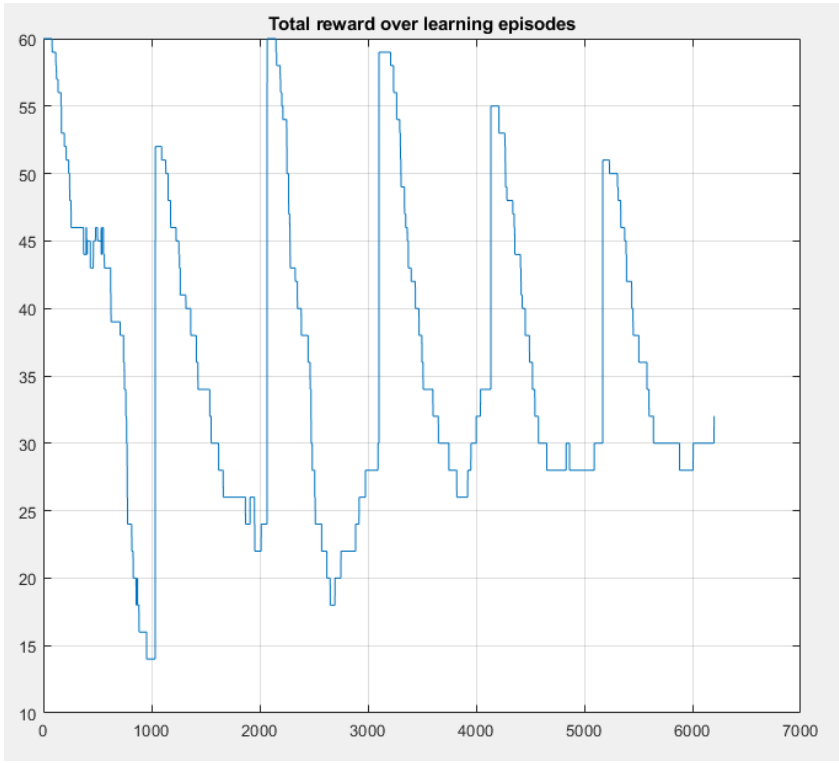
- **The reinforcement reward**
- If $|N_i| < 6$, $r_i = |N_i|$
- Else $r_i = 6$ (keep an alpha lattice formation)

Thus, it is expected that not all the nodes in a system will receive the maximum reward (6) at all iterations of an episode. However, there are trends in the above plots that demonstrate how the cooperative algorithm generates more reward than the independent algorithm. For the independent algorithm, there is higher reward only at the beginning of each episode when nodes are randomly clustered together; the reward drop to lower values as each episode progresses. On the other hand, the individual rewards for the cooperative algorithm nodes are less predictable and more dependent on the position of the individual node within the lattice-shaped network. By the end of the final episode, the nodes in the cooperative algorithm generally have higher rewards than the nodes in the independent algorithm.

Requirement 4 - Total Sum of Reward

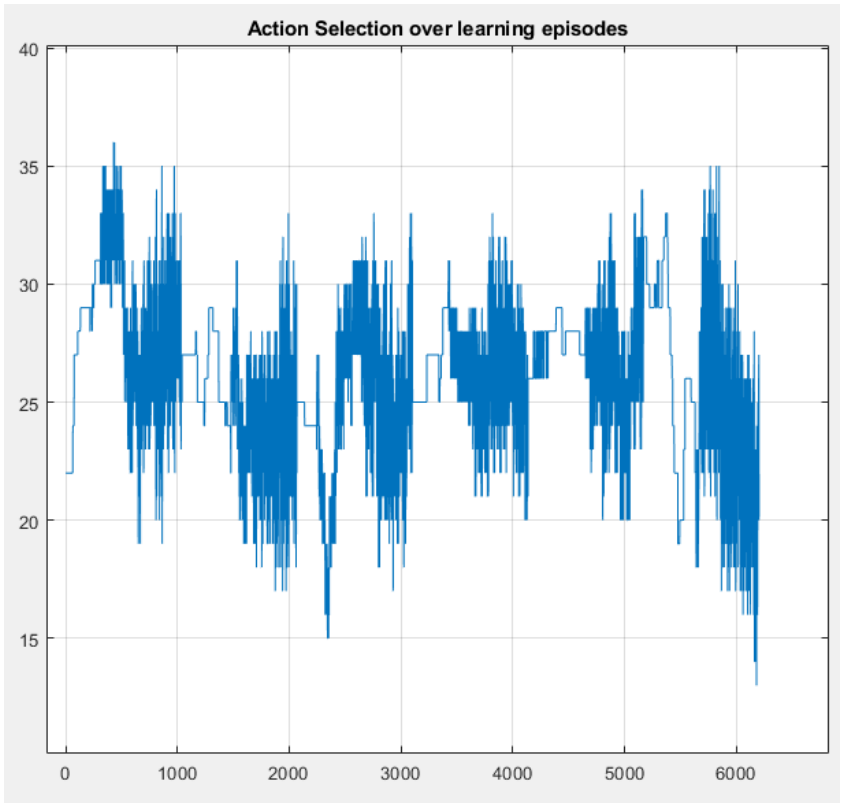
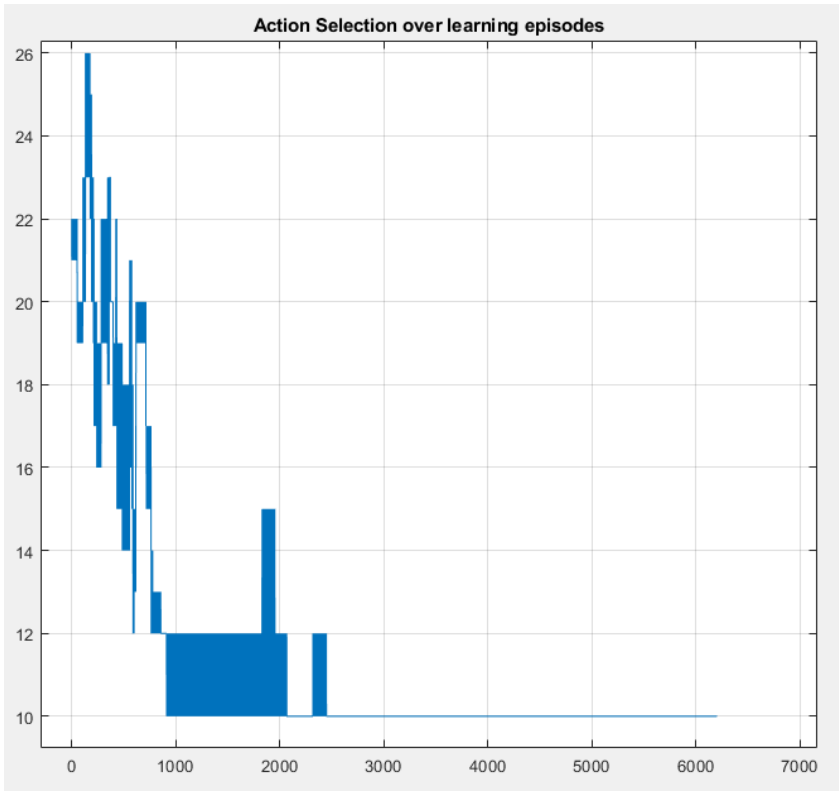
Independent	*Cooperative*





This is a clearer demonstration of how the cooperative algorithm usually generated more reward than the independent algorithm, especially after the nodes converged to a single action.

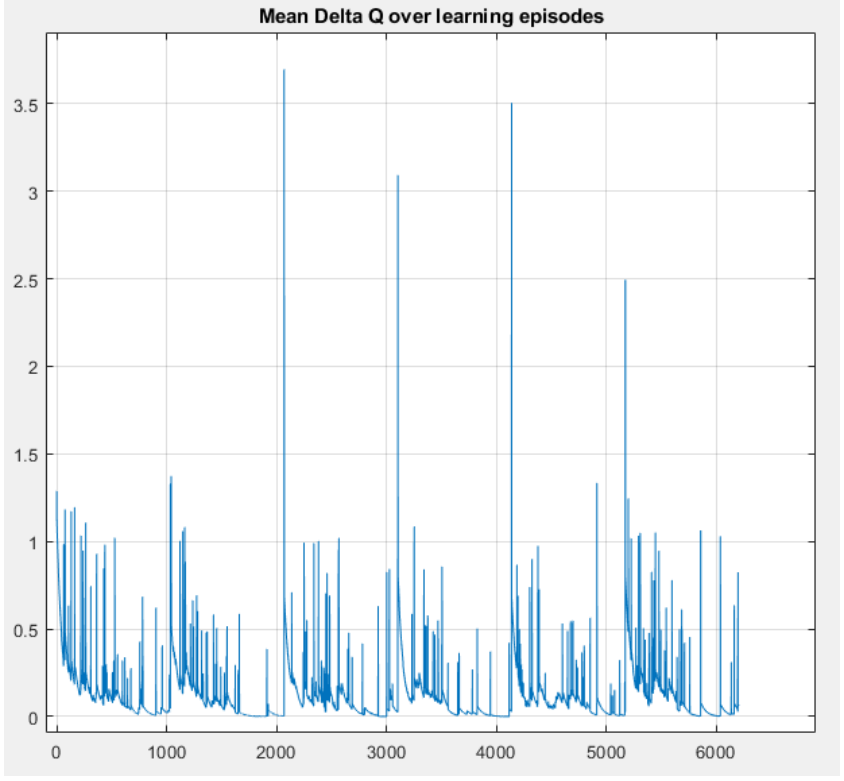
Requirement 5 - Action Selection

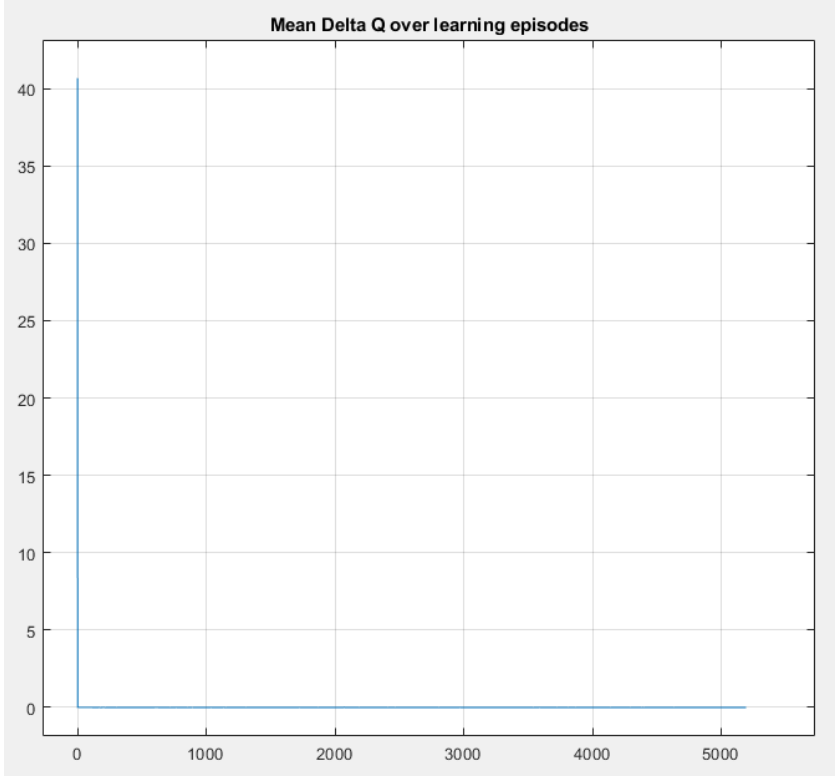
Independent	*Cooperative*
	

Since there are 10 nodes and 4 possible actions (encoded 1-4), the goal is to have these graphs stabilize at a y-value of 10, 20, 30, or 40. The independent algorithm graph for action selection never stabilizes, while the cooperative algorithm graph eventually stabilizes to a value of 10, meaning that the nodes learned to select action 1 together.

Requirement 6 - Average Delta Q

Independent	*Cooperative*



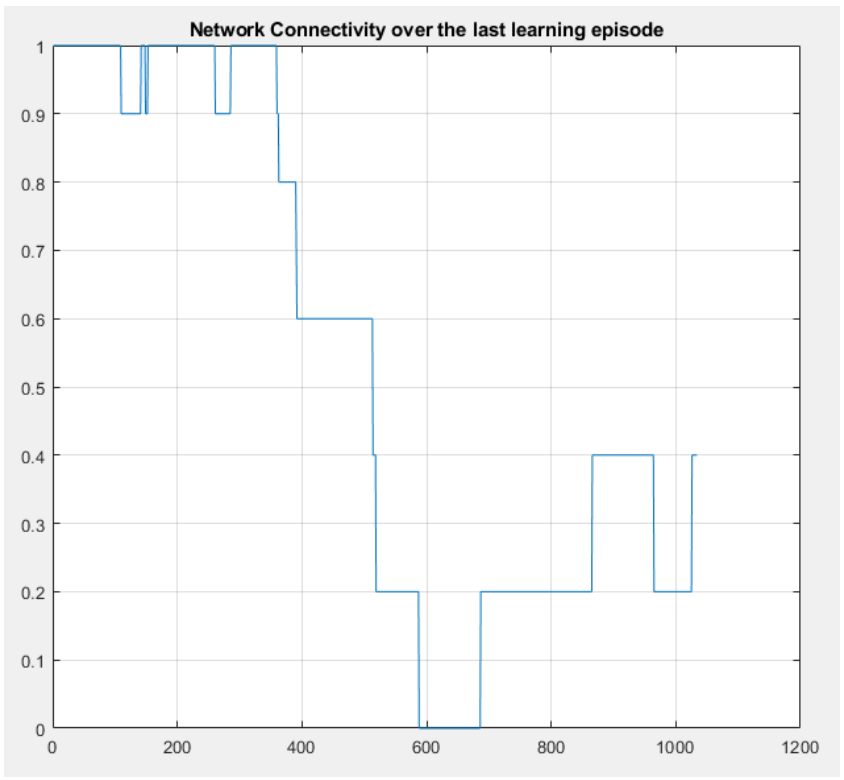


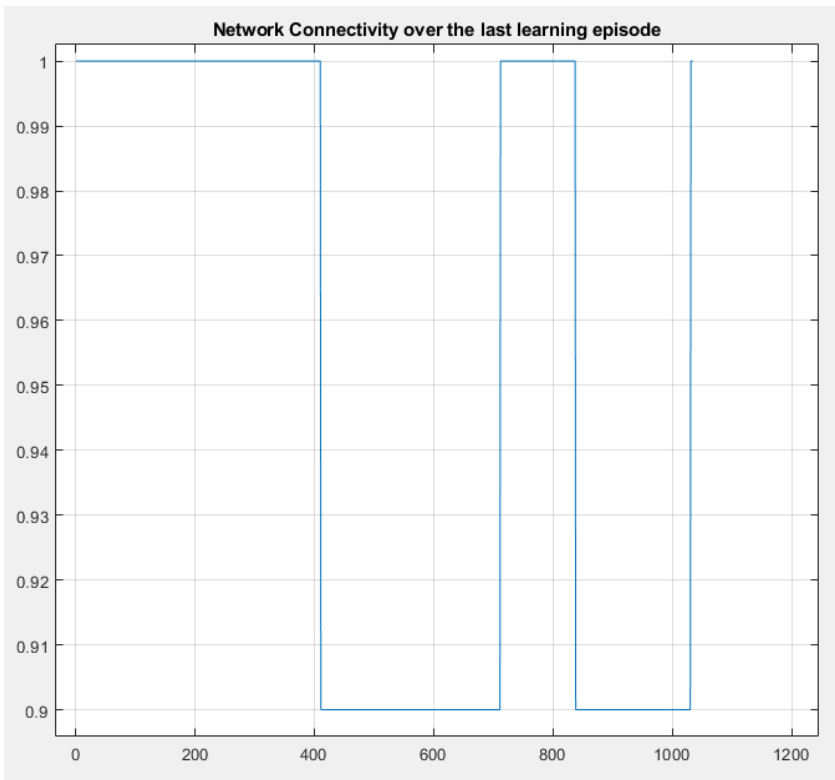
We can determine that the nodes in the network have converged to a single action when the change in Q table values stabilize to a value of 0. The cooperative algorithm does this, but the independent algorithm does not.

EXTRA PLOTS

Connectivity (Last Episode)

Independent	*Cooperative*

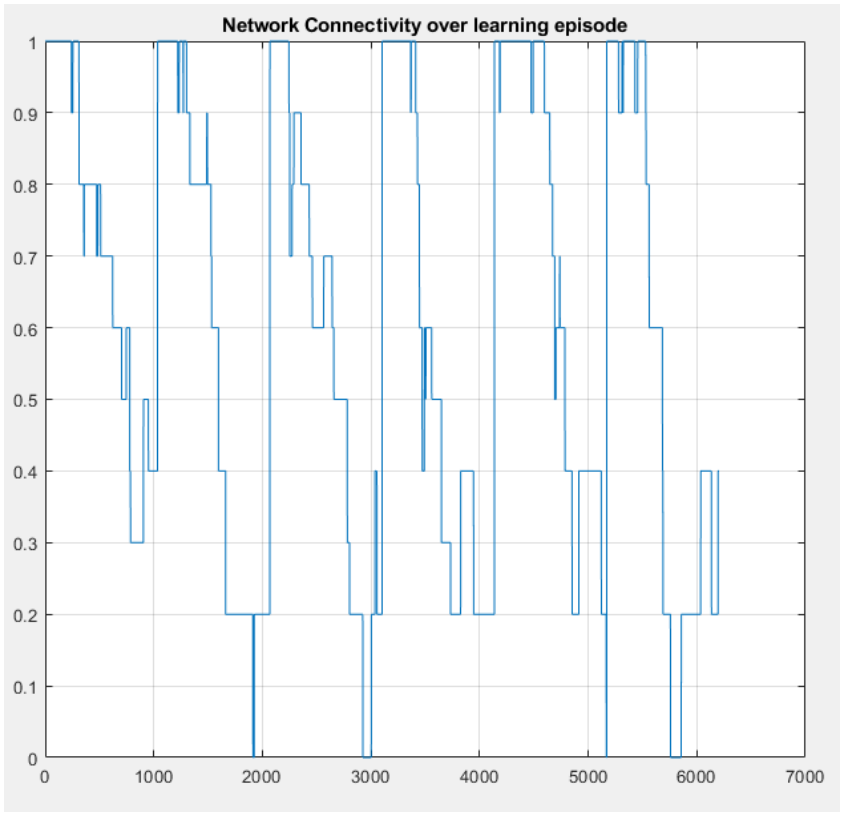


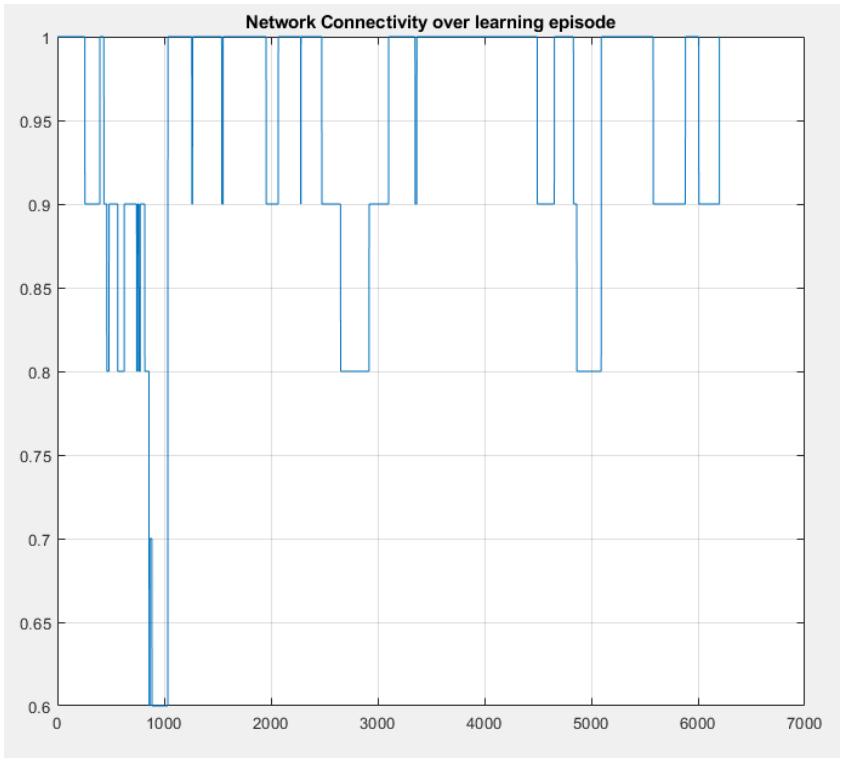


The independent Q-learning seems to disturb the flocking nature of the network, while the cooperative algorithm helps ensure the nodes flock together and stay connected.

Connectivity (All Episodes)

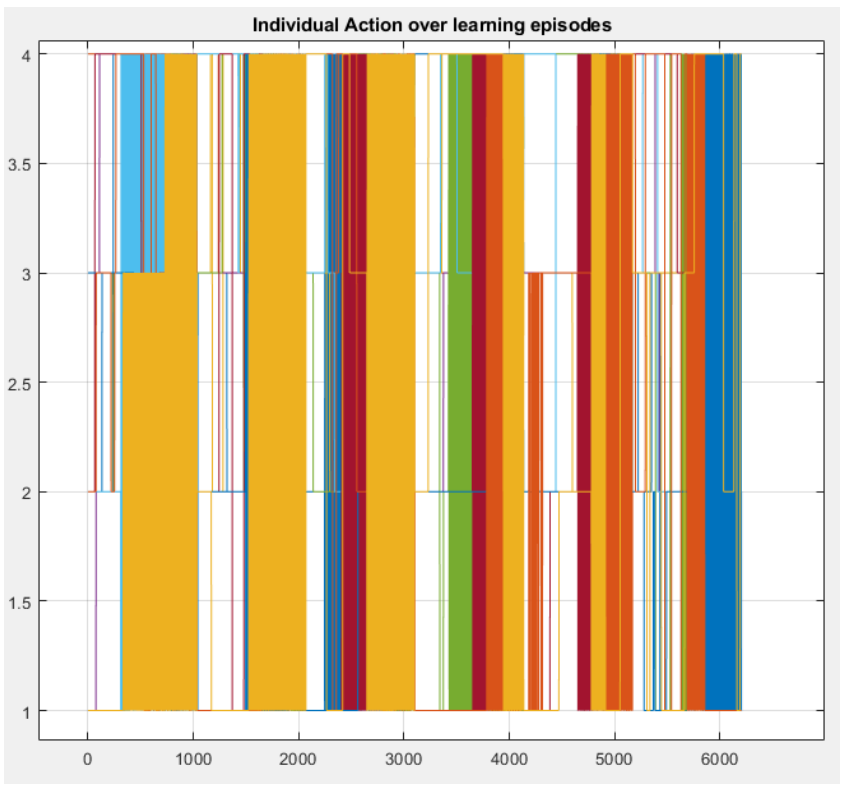
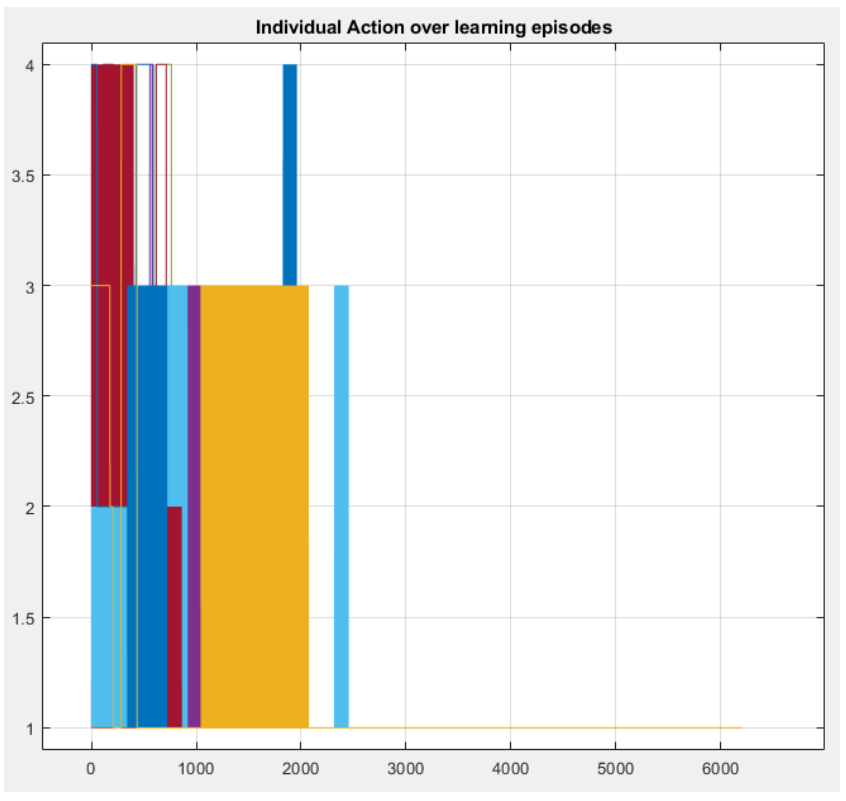
Independent	*Cooperative*





The network is generally more connected with the cooperative algorithm than with the independent algorithm.

Individual Action Selection

Independent	*Cooperative*
	

All the nodes in the cooperative algorithm converge to a single action (action 1), while the nodes in the independent algorithm choose different actions throughout the entire Q learning process.

Requirement 7 - Cooperative Q-Learning with Different w Values

The weight w value was used as shown in the project guidelines:

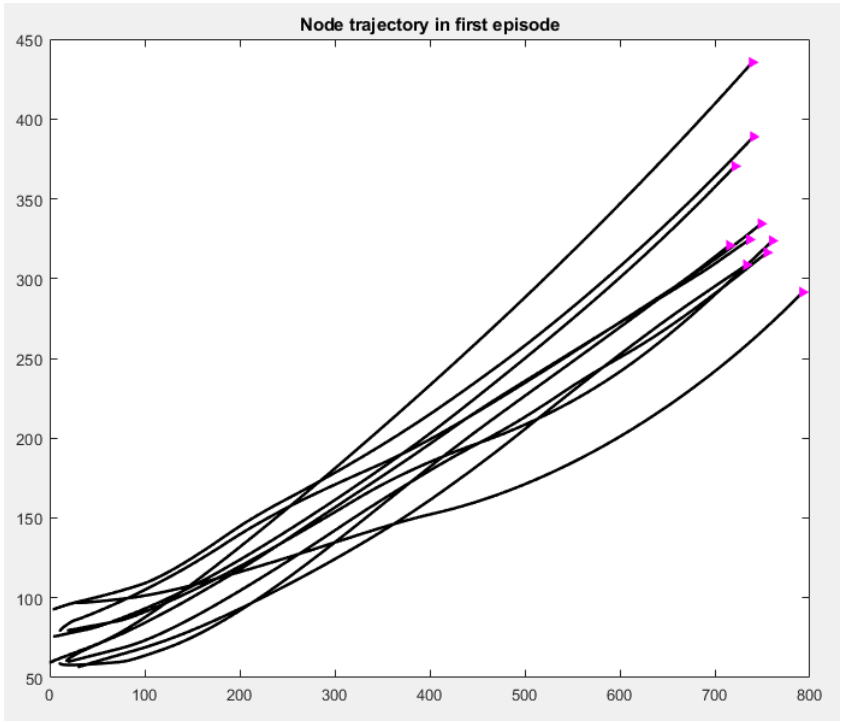
$$Q_i^{k+1}(s_i, a_i) \Leftarrow w Q_i^k(s_i, a_i) + (1 - w) \sum_{j=1}^{|N_i^a|} Q_j^k(s_j, a_j) \quad (8)$$

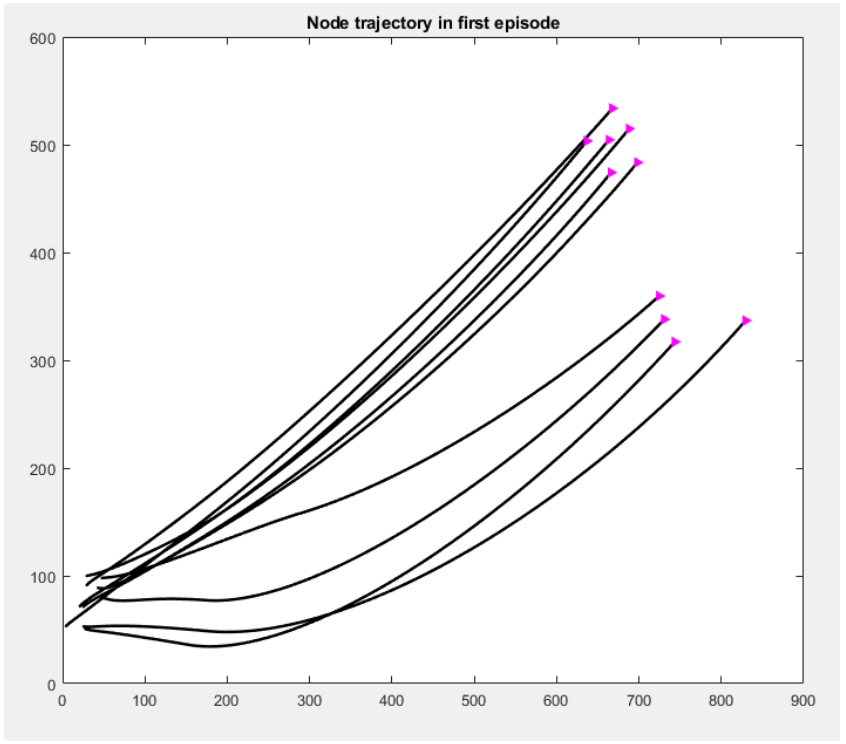
This means that a higher w value will make a node trust itself more than its neighbors, while a lower w value will make a node place more trust in the experiences of its neighbors.

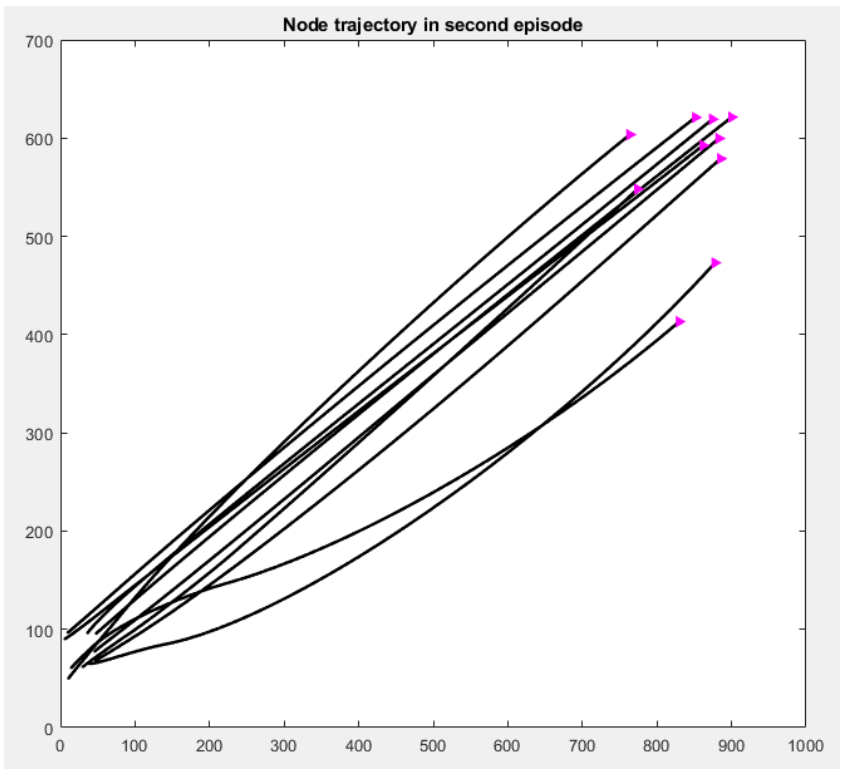
The results of the cooperative Q-learning algorithm with a weight w = 0.5 is shown in the sections above.

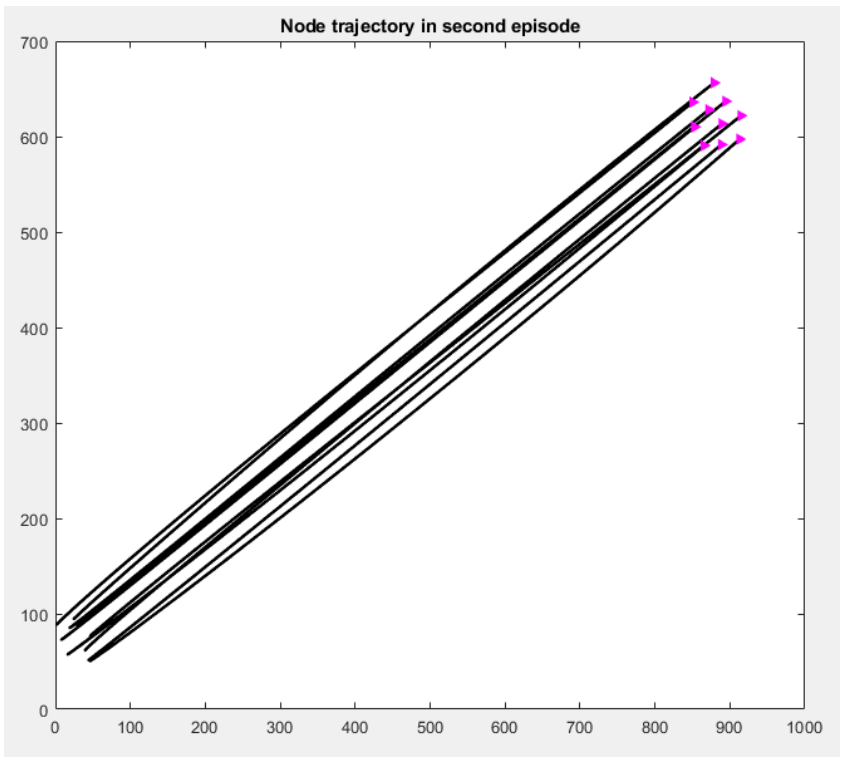
The results of the cooperative Q-learning algorithm with two other weight values (w = 0.3, 0.7) are shown below.

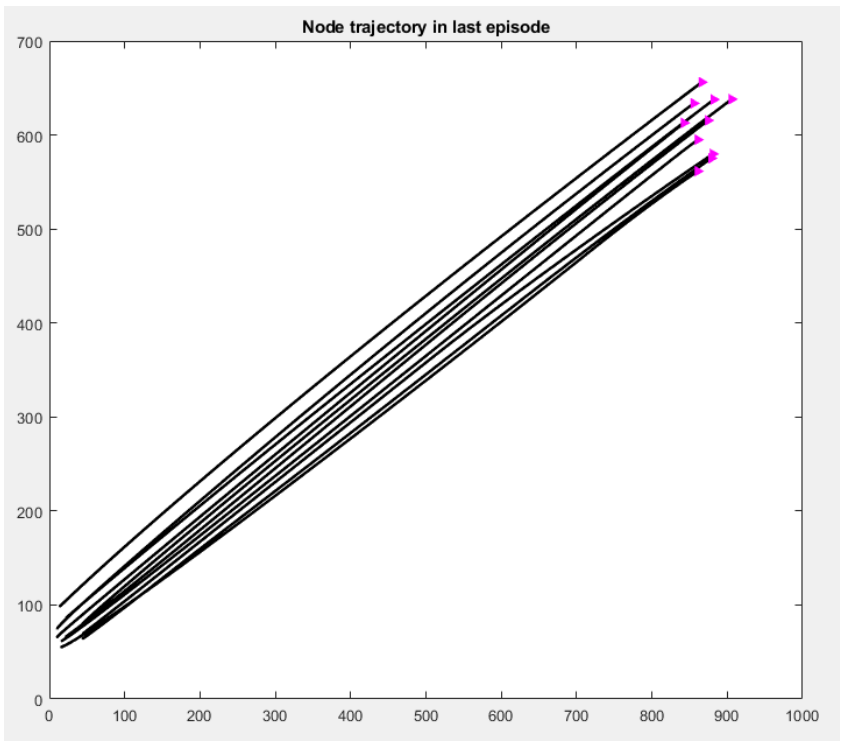
w = 0.3	*w = 0.7*

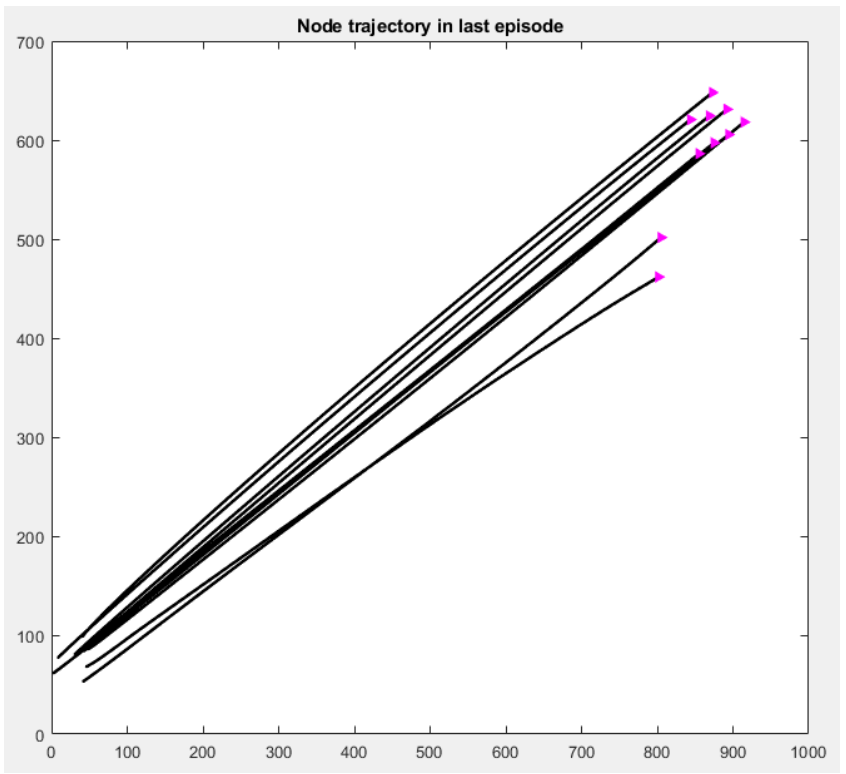

Node trajectory in first episode
450
400
350
300
250
200
150
100
50
0
100
200
300
400
500
600
700
800

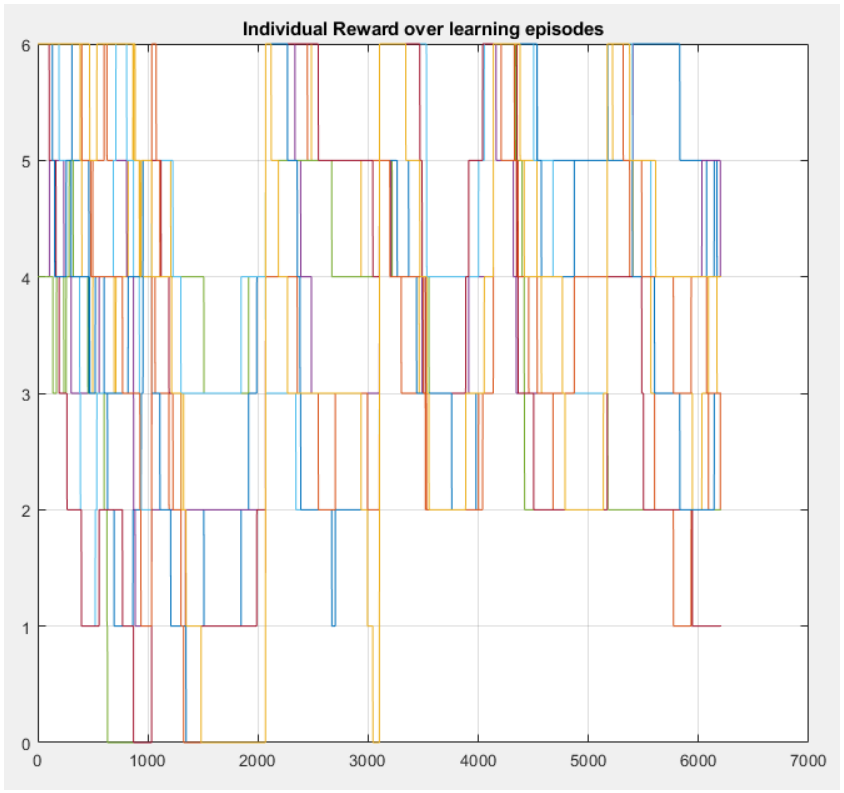

Node trajectory in first episode
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900

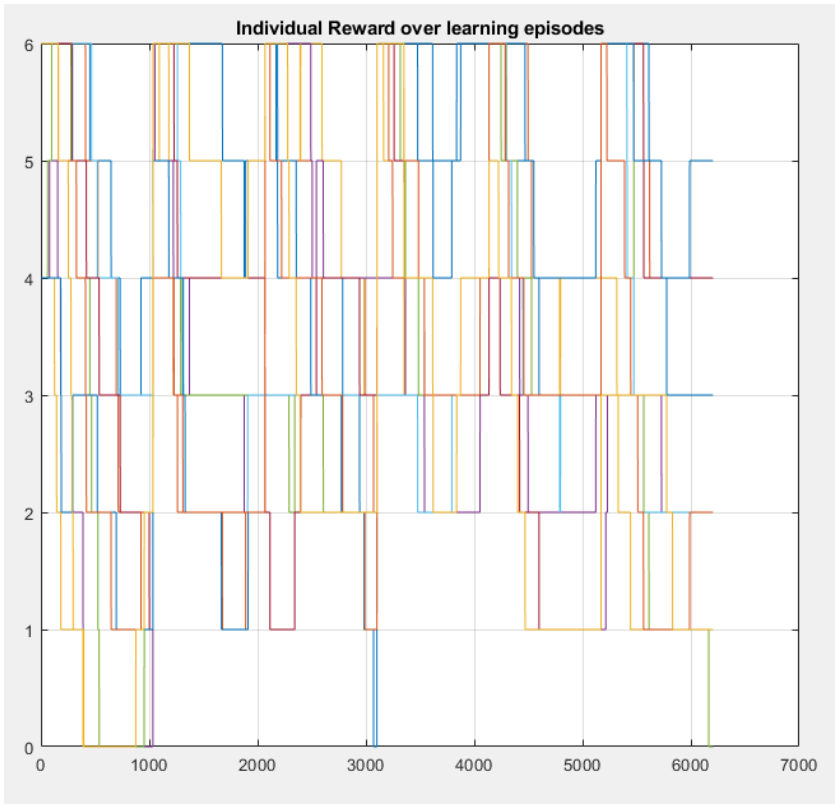

Node trajectory in second episode
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900
1000

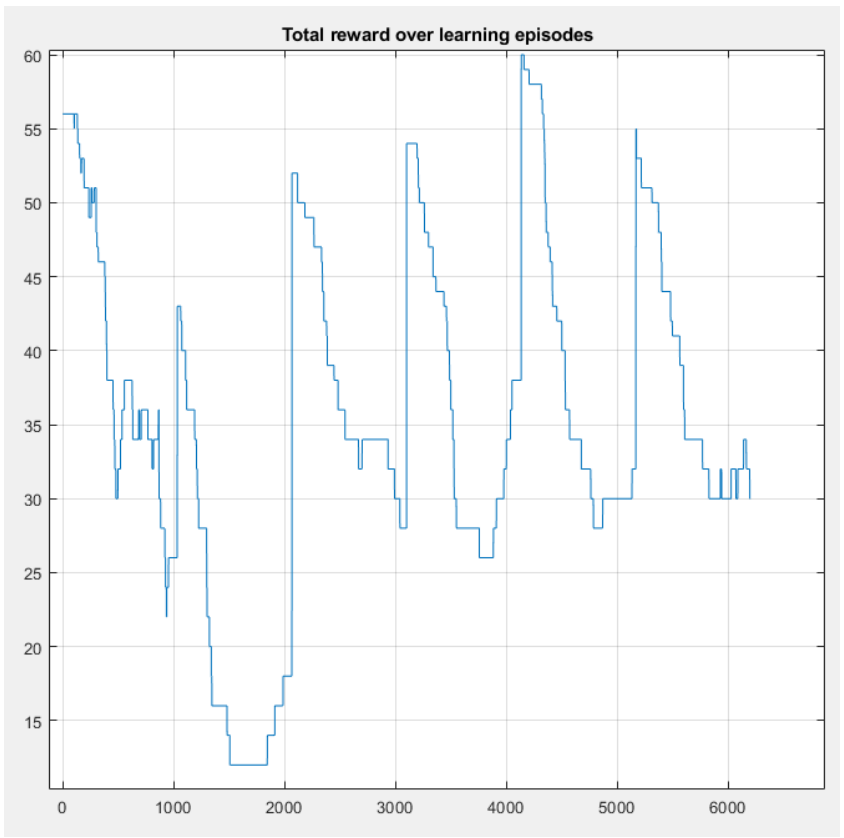

Node trajectory in second episode
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900
1000

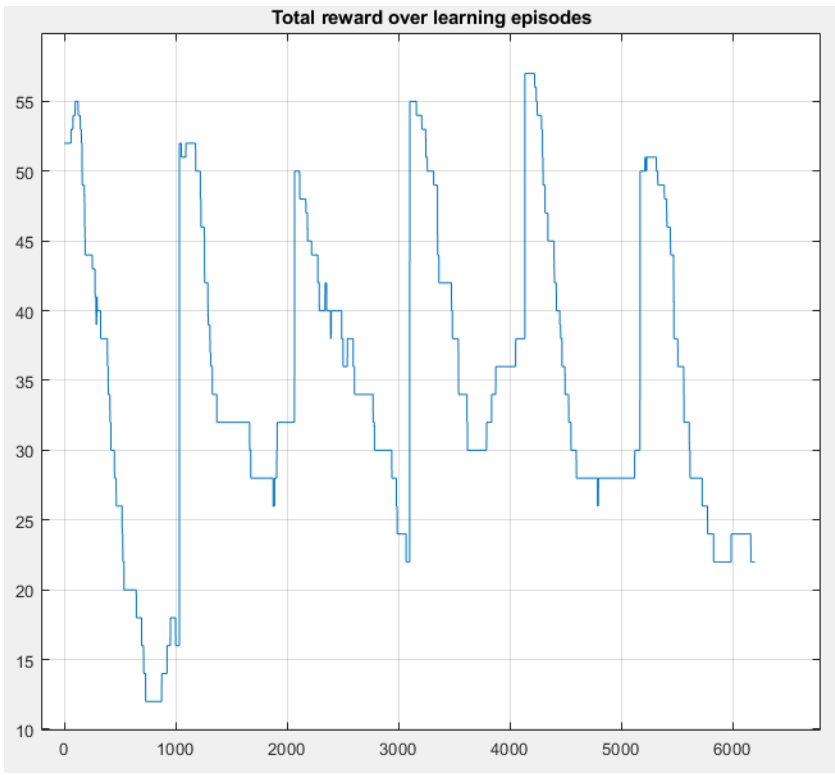

Node trajectory in last episode
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900
1000

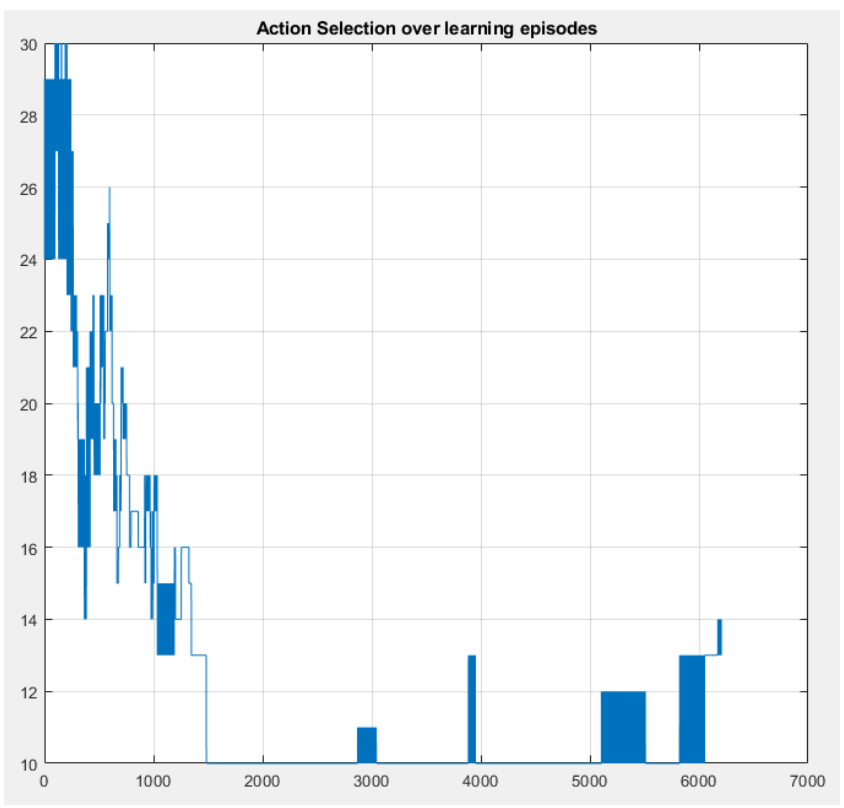

Node trajectory in last episode
700
600
500
400
300
200
100
0
100
200
300
400
500
600
700
800
900
1000

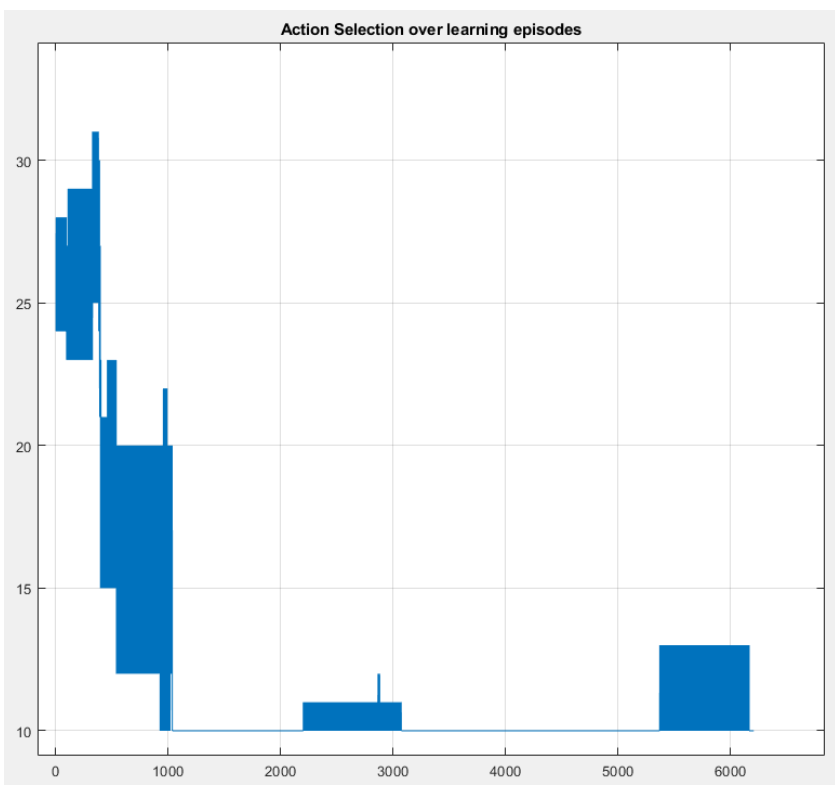

Individual Reward over learning episodes
0
1
2
3
4
5
6
0
1000
2000
3000
4000
5000
6000
7000

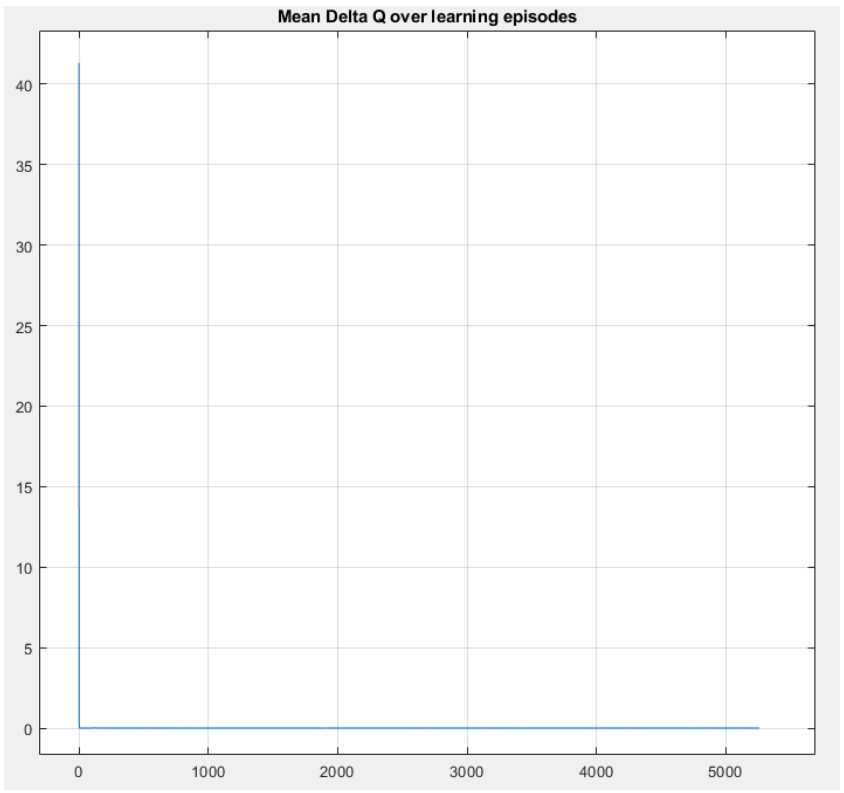

Individual Reward over learning episodes
0
1
2
3
4
5
6
0
1000
2000
3000
4000
5000
6000
7000

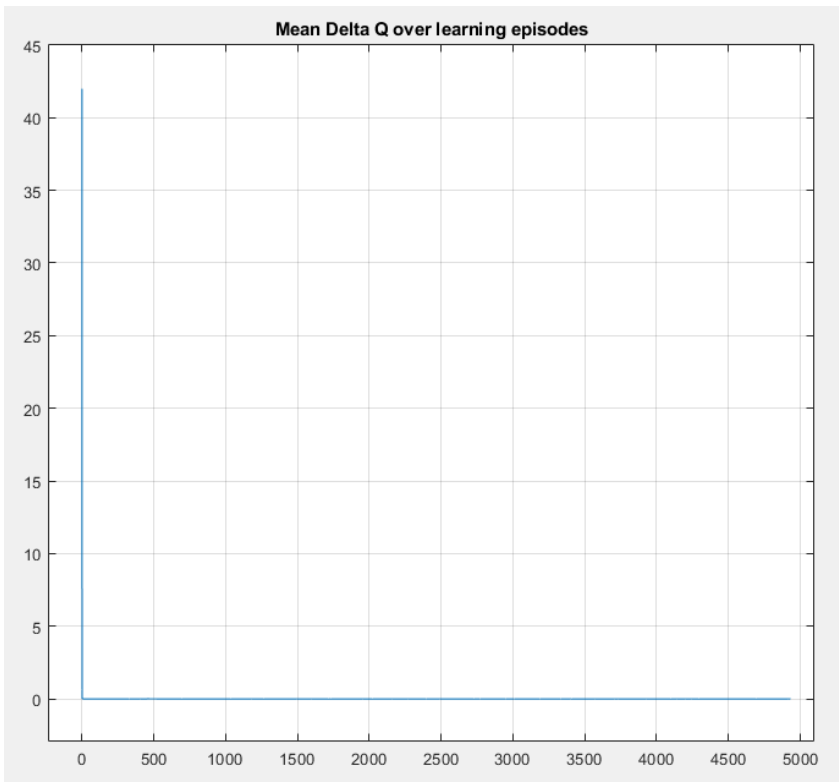

Total reward over learning episodes
15
20
25
30
35
40
45
50
55
60
0
1000
2000
3000
4000
5000
6000

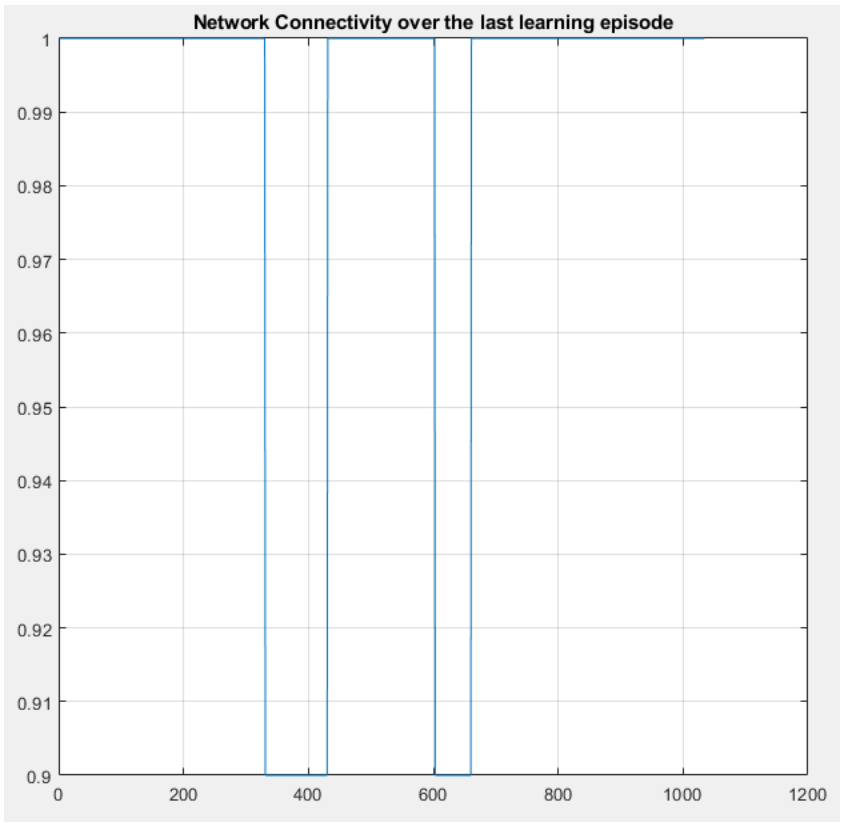

Total reward over learning episodes
10
15
20
25
30
35
40
45
50
55
0
1000
2000
3000
4000
5000
6000

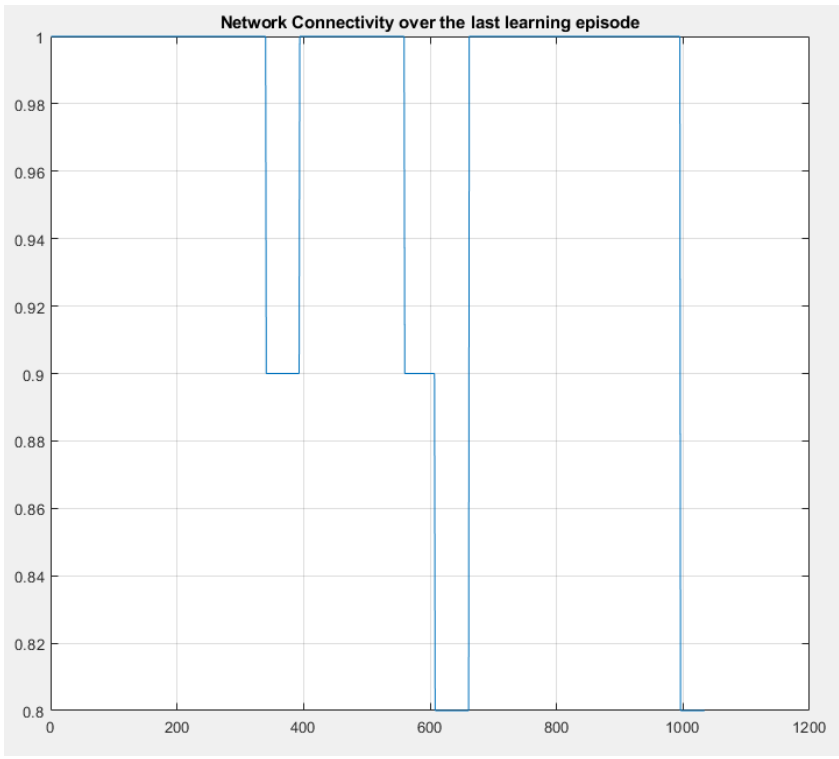

Action Selection over learning episodes
10
12
14
16
18
20
22
24
26
28
30
0
1000
2000
3000
4000
5000
6000
7000

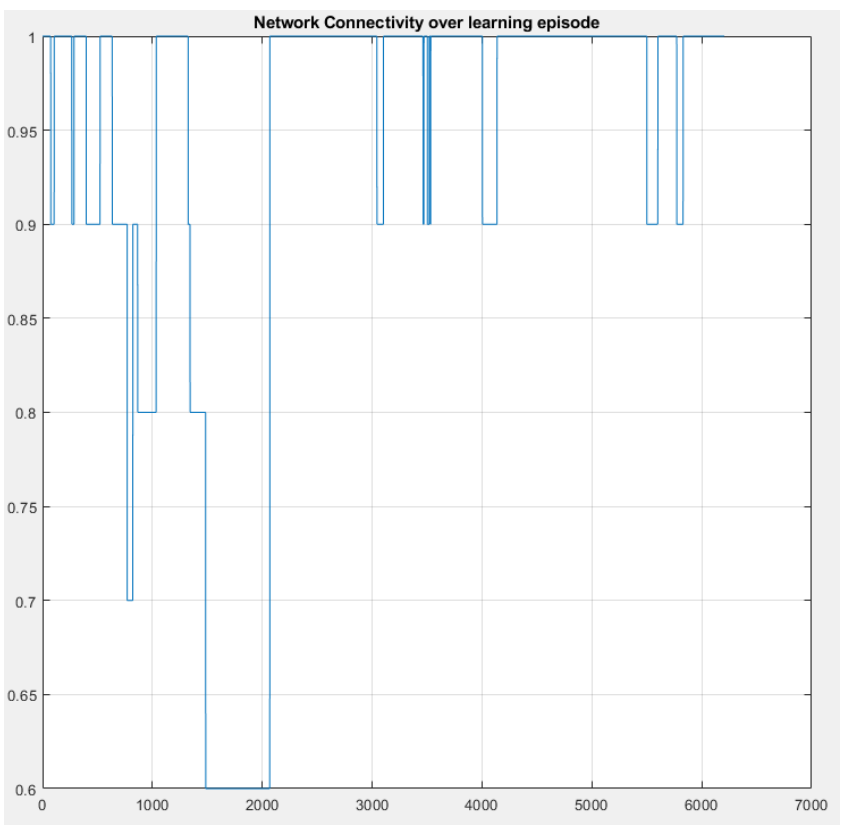

Action Selection over learning episodes
10
15
20
25
30
0
1000
2000
3000
4000
5000
6000

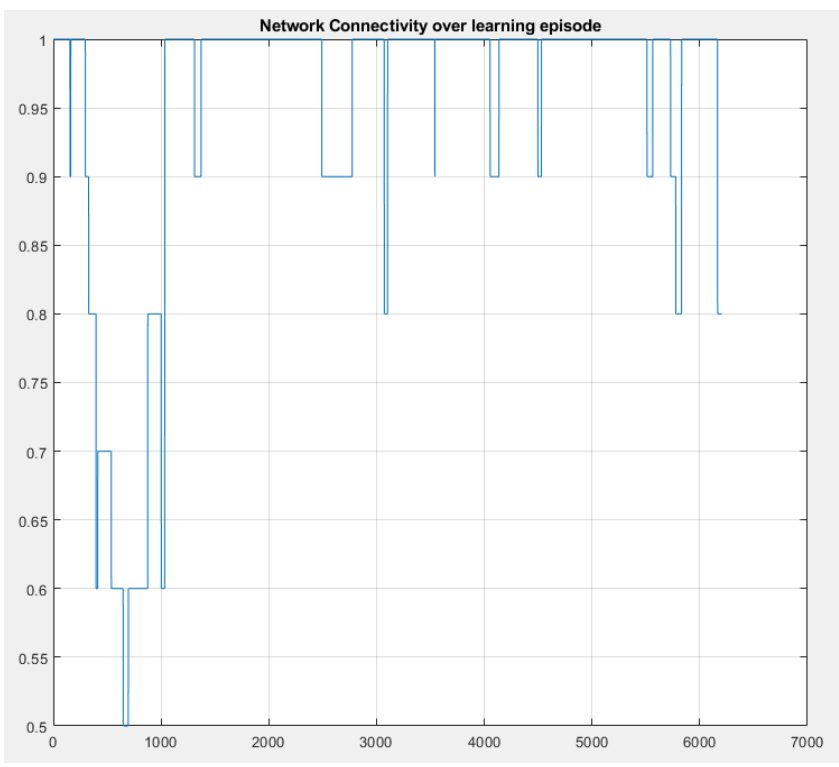












The graphs for both weight values are quite similar, but there are small differences that stand out.

- Node trajectory: In the last episode, nodes with w = 0.3 appear to travel more uniformly and directly than the nodes with w = 0.7.
- Individual reward: By the end of the Q-learning, w = 0.3 nodes seem to have fewer nodes with 0 reward than w = 0.7 nodes.
- Total reward: The low points of the w = 0.3 graph are not as low as those of the w = 0.7 graph.
- Connectivity: The w = 0.3 graph spends more time (iterations) stable at a fully-connected value of 1 than the w = 0.7 graph.

These small differences indicate that a lower weight value (in other words, updating a node's Q-value with more weight on its neighborhood) results in higher reward and better network flocking than with a higher weight value.

However, in the tests performed, it appears that the best network performance resulted from a weight value w = 0.5, so perhaps a more even balance of trust in oneself and trust in the neighborhood is optimal.