Emeka Ezike, Dominic Garrity, and Victor Goncalves

Professor Gil

CS 286

02 March 2022

**Programming Assignment 2**

**Group Information**

Group number: 6

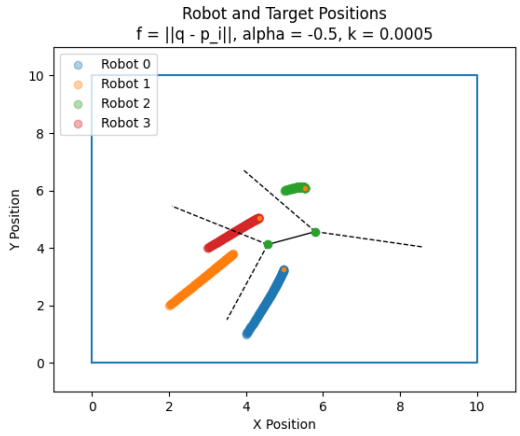
Group members: Dominic Garrity, Victor Goncalves, and Emeka Ezike

**Problem #1**

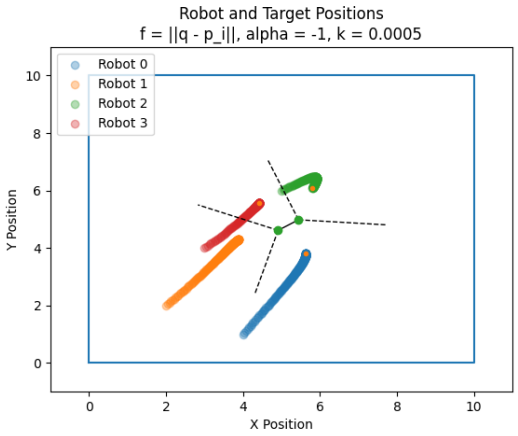
*Note: In coverage.py, we created two variables, "part" and "stoch," for easier evaluation. You can set "part" to the part of #1 that you wish to evaluate and "stoch" to True or False, depending on whether you want the system to be stochastic. For every part of this problem, we ran the simulation for 200 time steps and let the sensing function* $f(p_i, q)$ *be the Euclidean distance between the ith robot's location* $p_i$ *and some point in the environment* $q$*. We also conducted a case study to determine the extent to which Mac Schwager and his colleagues' algorithm could coordinate multi-agent coverage for different values of alpha. In specific, just as Schwager did in "Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment," we considered when* $\alpha = (-0.5)$*, when* $\alpha = (-1)$*, when* $\alpha = (-5)$*, and when* $\alpha = (-10)$*. Ultimately, we discovered that the robots achieve better coverage as as* $\alpha$ *approaches* $-\infty$*. This, as Schwager et al. stated, is because the*

$(f(p_i, q)/g_\alpha)^{\alpha-1}$ *term of the gradient-based controller approaches the indicator function of the Voronoi cell* $V_i$ *as* $\alpha$ *approaches* $-\infty$.

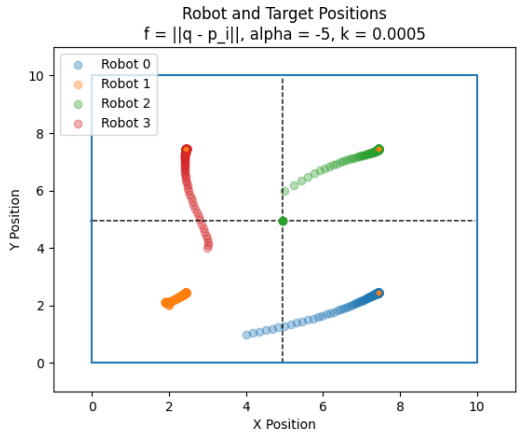
A. Please find our program output in Figure 1 below. Each plot is subtitled with its corresponding alpha value. In this part, we set k equal to 0.0005 and used the α values studied by Schwager and his colleagues: -0.5, -1, -5, and -10. (In fact, as mentioned above, we used these α values in every part.)



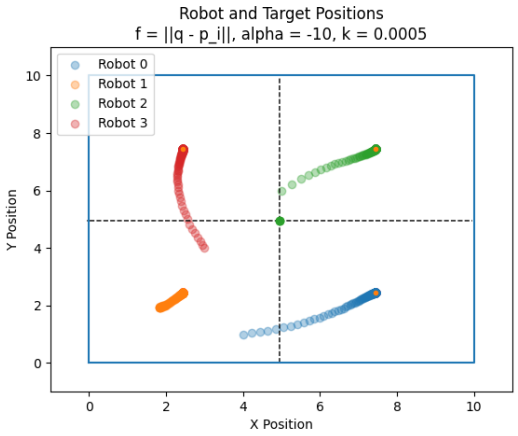

*(a)*




*(b)*



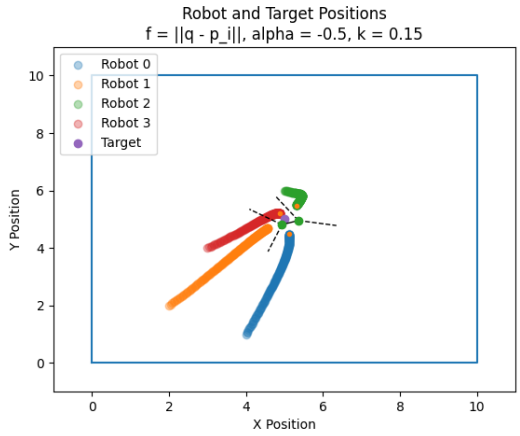

*(c)*



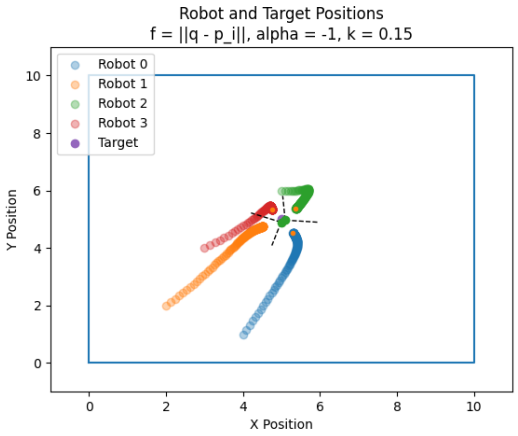

*(d)*

*Figure 1: We considered multi-agent coverage in the case with no target and the above parameters.*

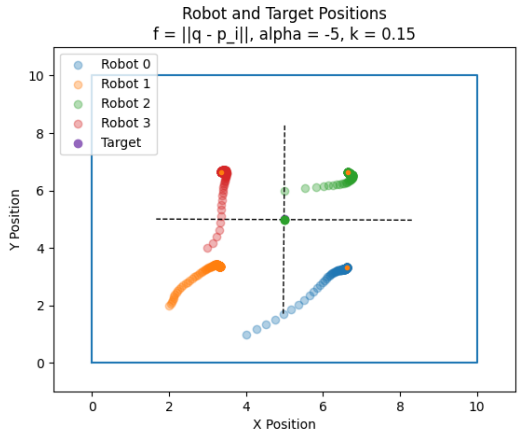
B. Please find our program output in Figure 2 below. In this part, we set k equal to 0.15, because, for smaller k, robots that are far away from the target were not extremely responsive. This is expected, because environmental "importance" is normally distributed around the target. The target, (5, 5), is included in purple.




*(a)*




*(b)*

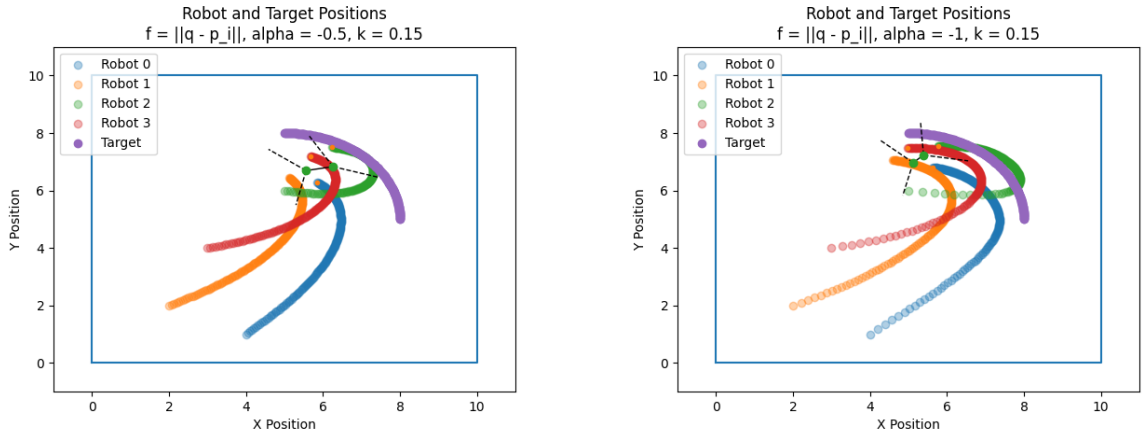



*(c)*

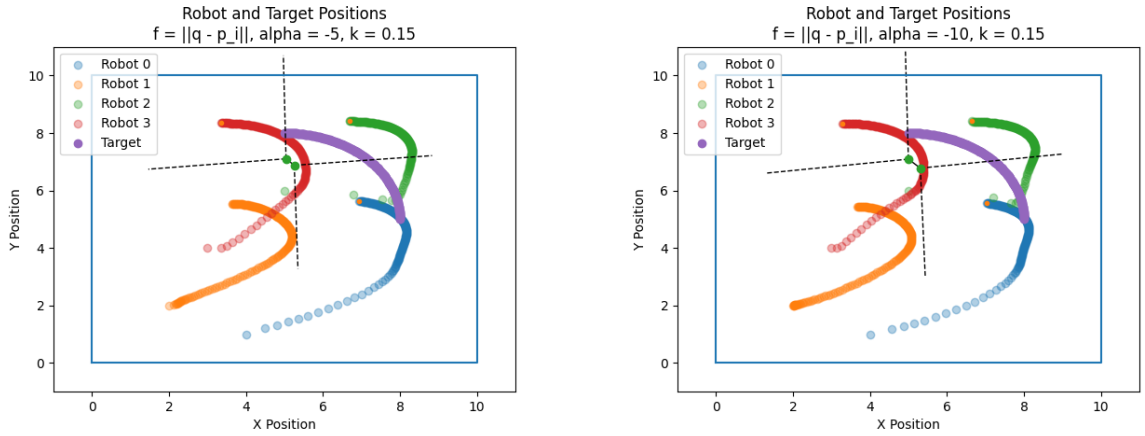

*(d)*

*Figure 2: We considered multi-agent coverage in the case with a target at (5,5) and the above parameters.*

C. Please find our program output in Figure 3 below. We also set k equal to 0.15 in this part, for the same reason described in part (B). The target, which moves in a quarter circle around (5, 5) with a period of 800 time steps, is included in purple.




*(a)* *(b)*

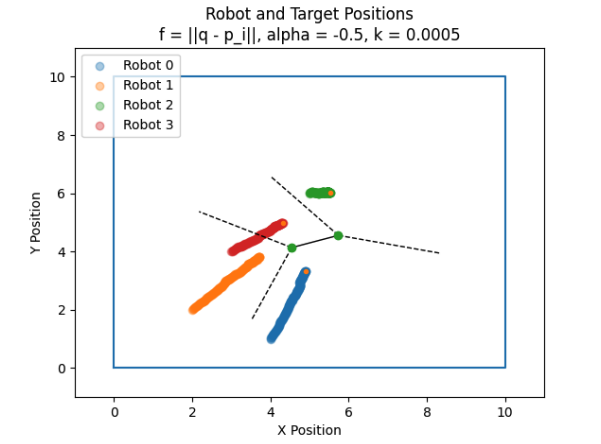



*(c)* *(d)*

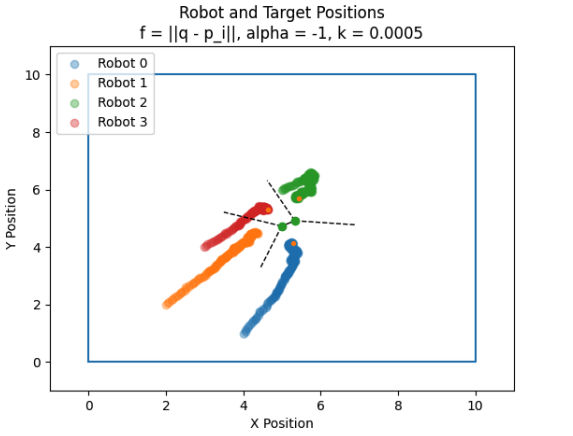
*Figure 3: We considered multi-agent coverage in the case with a moving target and the above parameters.*

D. All of the robots tended toward the target, even as it moved, for all 200 time steps–never settling on convergent positions. This is different from part (A), in which we assumed a uniform importance function, but makes sense, because environmental "importance" is normally distributed around the target; areas of higher importance will be closer to the target than areas of low importance, regardless of whether or not the target is moving.

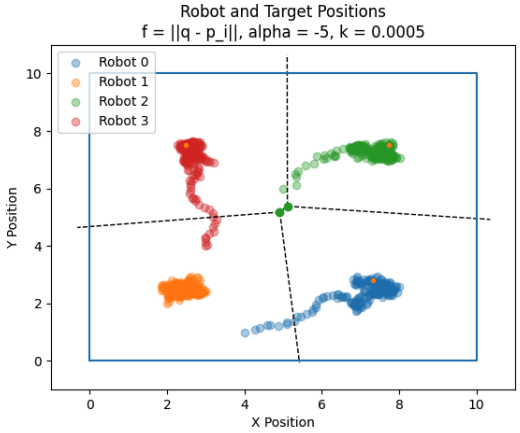
E. Please find our program output in Figure 4 below. Again, we set k equal to 0.15, for the same reason described in part (B).

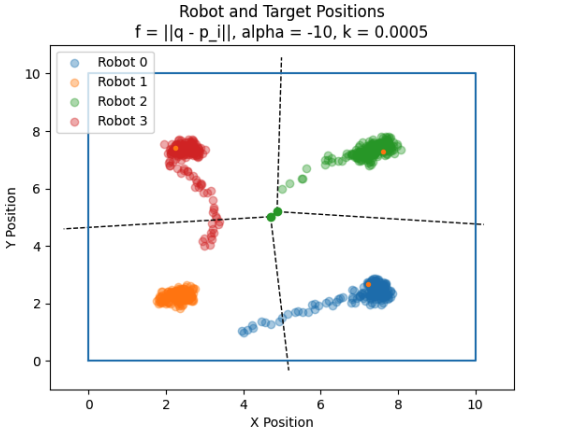



*(a)*




*(b)*



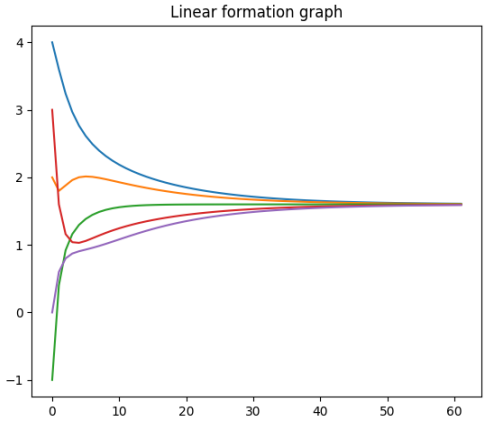
*(c)* *(d)*

*Figure 4: We considered multi-agent coverage in the case with stochasticity and the above parameters.*

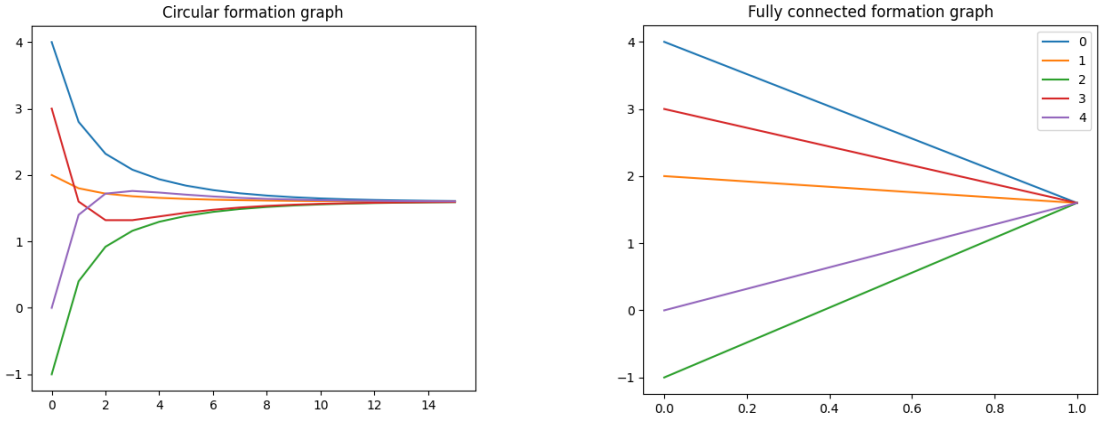
**Problem #2**

*Note: When plotting update iterations for these graphs in consensus.py, we created a threshold to determine whether a graph has reached consensus. Furthermore, we set the threshold to 0.20 in the case with stochastic noise, and 0.04 otherwise. This is because node values never reach the exact same value, but can differ by a couple of tenths.* ***Furthermore, we have environment labels like add_noise and add_adversary that need to be changed to view different cases.***

A. The fully connected graph formation converged fastest, and was able to do so in 1 time step.



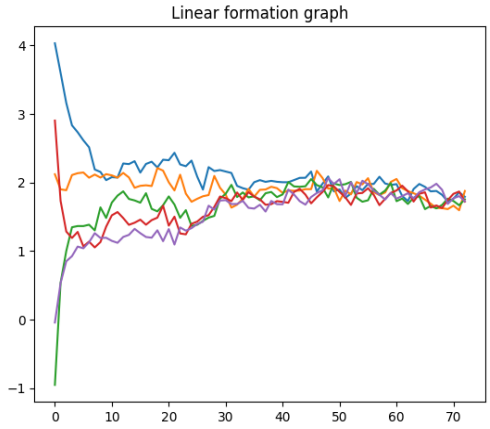

*(a)*



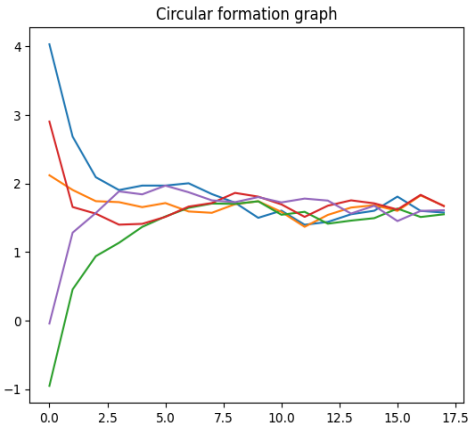

*(b)* *(c)*

*Figure 5: We plotted each of the five nodes' states over time within the linear graph formation, the partially connected circle graph formation, and the fully connected graph formation. The plotting terminated when the graph reached consensus.*

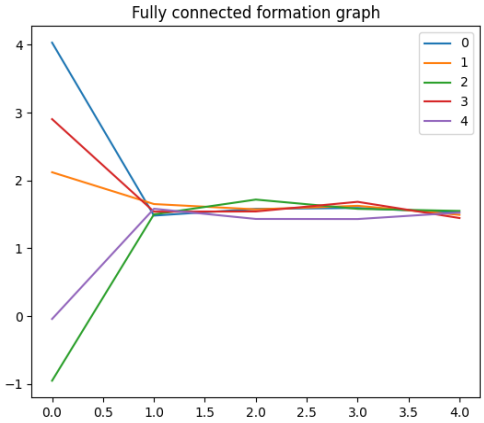
B. With the addition of stochastic noise, every graph took longer to reach consensus. Interestingly, not even the fully connected graph reached consensus in one time step. Further, once a graph reaches our threshold for consensus, we choose to stop it from updating. However, with noise, graphs could theoretically (and, in practice, did) jump in and out of consensus; we decided to stop recording once the nodes' states were within a predefined threshold (0.04) of the consensus value.




*(a)*




*(b)*




*(c)*

*Figure 6: We reconducted the experiment of #2(A), but added a Gaussian value (μ = 0, σ = 0.1) to the nodes' update states in order to simulate stochastic noise.*

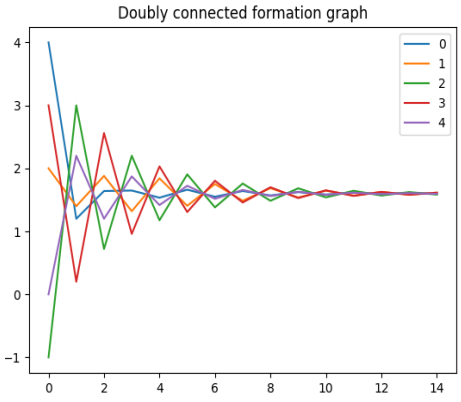
C. The effect of stochasticity differs significantly between the linear, partial, and fully connected graphs. Firstly, the linear graph is far more noisy (as demonstrated by steep edges jutting in random directions) than the circular and fully connected graphs and takes longer to reach consensus (on average takes 27 time steps, compared to an average of 13 time steps for the circular graph and 1 time step for the fully connected graph). This implies that the networks with more connected structures were better suited to counteract the effects of noise. We believe this to

be true because fully connected graphs converge faster, so noise has less of an effect. The longer it takes for a network to reach convergence, the more noise accumulates in the system since noise is added at each time step. We know that more connected graphs converge faster, so they will ultimately have less accumulated noise.

D. Although all of the nodes are connected to each other, each node's direct/numerical neighbors are given twice as much weight. Therefore, whenever a node takes a step towards consensus, it seems to overstep it's convergence update by about twice the correct amount. For this reason, it takes the graph longer to converge on/reach consensus than in the case with a fully connected graph. In Figure 7 below, we provide an adjacency matrix A that describes the graph formation that we used, as well as a plot that shows its convergence over time.

$$A = \begin{bmatrix} 0 & 2 & 1 & 1 & 2 \\ 2 & 0 & 2 & 1 & 1 \\ 1 & 2 & 0 & 2 & 1 \\ 1 & 1 & 2 & 0 & 2 \\ 2 & 1 & 1 & 2 & 0 \end{bmatrix}$$

*(a)*




*(b)*

*Figure 7:* We reconducted the experiment of #2(A), but only with the graph formation described by the adjacency matrix in (a). Considering the linear and circular formations would have been trivial.

E. The Fielder value for linear formation graph (Figure 8a) is **0.382**. The Fielder value for the circular formation graph (Figure 8b) is **1.382**. The Fiedler value for the fully connected formation graph (Figure 8c) is **4.9999999999999964**. For the three test cases, we generated three 5x5 adjacency matrices using the rand_adj function using a random probability *p*. The Fielder value for the random adjacency matrix made when *p* = 0.70, shown in Figure 8(d), is **1.382**. This is interestingly close to the Fiedler value of the circular graph, even though the circular graph is more sparse. The Fielder value for the random adjacency matrix made when *p* = 0.093, shown in Figure 8(e), is **0.0**; this makes sense, as the graph is not a connected one. The Fielder value for the random adjacency matrix made when *p* = 0.6574, shown in Figure 8(f), is **0.6972**.

$$A_l = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

*(a)*

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

*(b)*

$$A_{fc} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

*(c)*

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

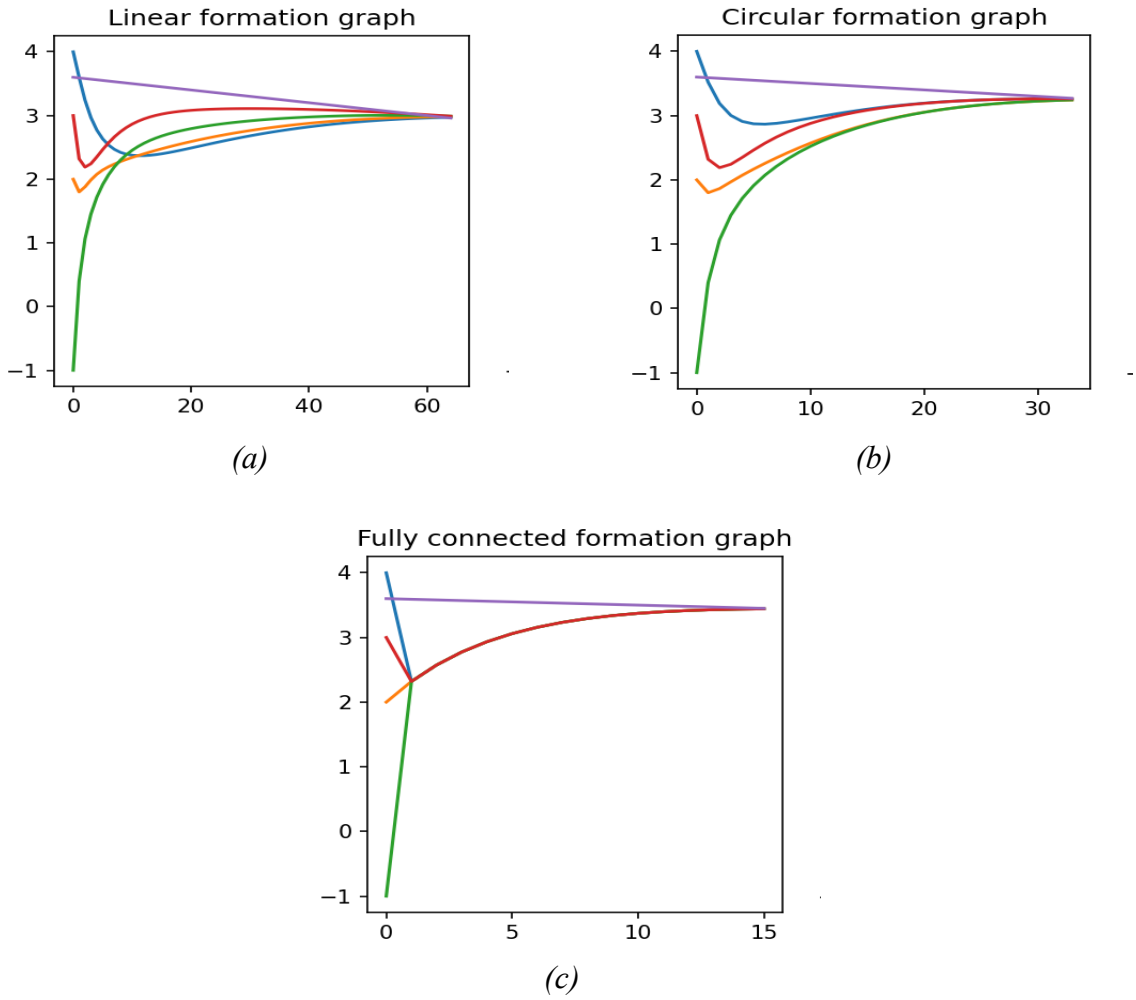
*(d)*

$$R_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad R_3 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

*(e)* *(f)*

*Figure 8: We defined adjacency matrices of various graphs and calculated their Fiedler values.*

F. In this experiment, we replaced our last node with a misbehaving agent that would be classified as a faulty agent because it moves towards its own unique target and makes decisions without awareness "of the structure and state of the network" (Pasqualetti 90). According to paper "Consensus Computation in Unreliable Networks: A System Theoretic Approach", the consensus algorithm we implemented for #2(A) has "no resilience to malfunctions, and the presence of a misbehaving agent may prevent the entire network from reaching consensus" (Pasqualetti 92). In our example, the presence of the faulty agent forced the other agents to converge to the state of the faulty agent, which makes intuitive sense: since the behaving agents are attempting to coalesce the network at one state, they will converge to the state of the faulty agent because the behaving agents can move towards other agents, but the faulty agent will move towards its own target.



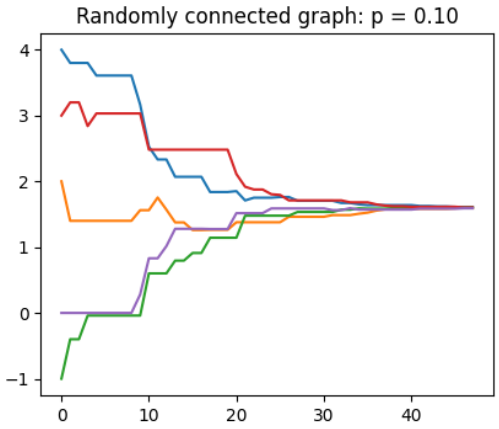

*(a)* *(b)* *(c)*

*Figure 9: We reconducted the experiment of #2(A), but replaced the fifth node with an adversarial agent (seen in purple) that had an initial state of 3.6 that approached its target state of 0.0 at a rate of 0.01 per time step.*

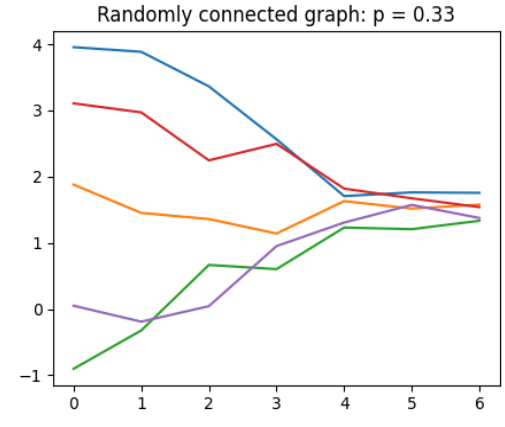
G. As shown in Figure 10, the networks with higher edge probabilities reach consensus faster, which is expected because, with higher edge probabilities, there is a higher number of expected connections and thus a greater amount of information exchange. Convergence is different to part 2a because in these switching networks, the neighbors that affect an agent are always changing. This is most evident in the switching network with probability 0.1, shown in Figure 10(a), where

an agent occasionally has no neighbors so it makes zero change in a time step. In Theorem 6 of the paper “Consensus and Cooperation in Networked Multi-Agent Systems,” Olfati-Saber et al. prove that a periodically connected switching network will eventually reach consensus, and we see in our examples that as the likelihood of forming connections increases, the network

*Without Noise*

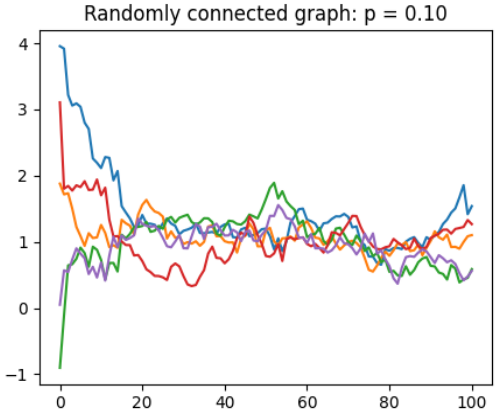



*(a)*

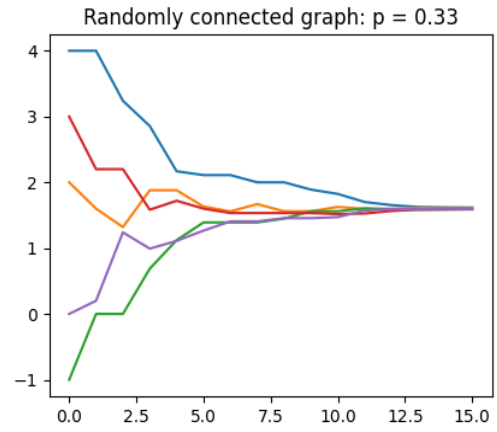



*(c)*

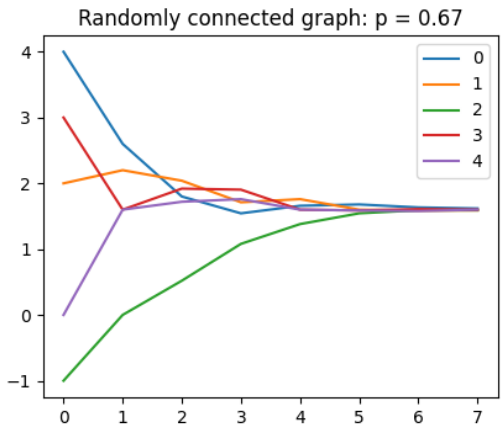
*WIth Noise*



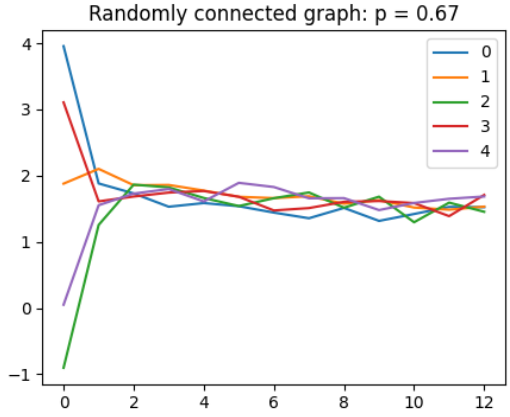

*(b)*




*(d)*




*(e)*




*(f)*

*Figure 10: We reconducted the experiment of #2(A), but used switching network formations that formed edges with probability p. We explored these results for three separate values of p.*