CS 286 - Homework 3, Consensus and Coverage

Adolfo Roquero, Leticia Schettino, Kobi Greene

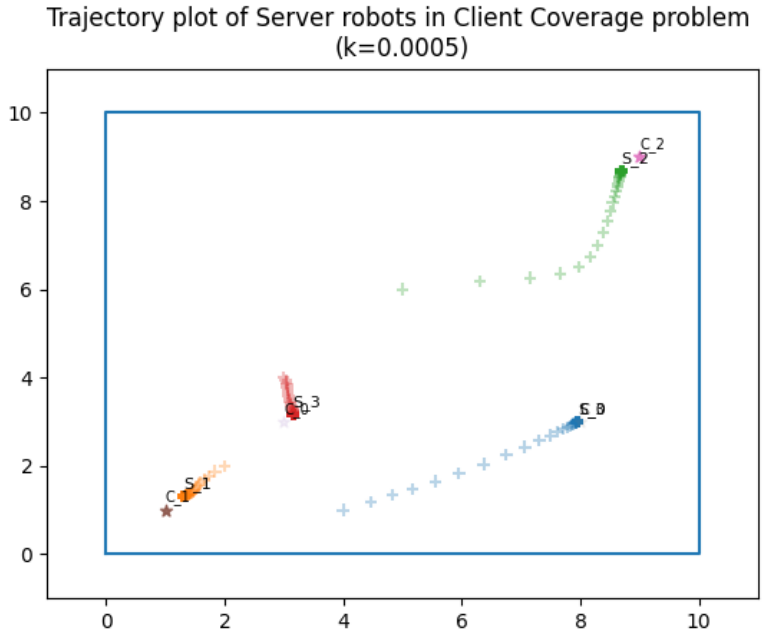
March 2022

1 Coverage with Spoofed Agents

a)

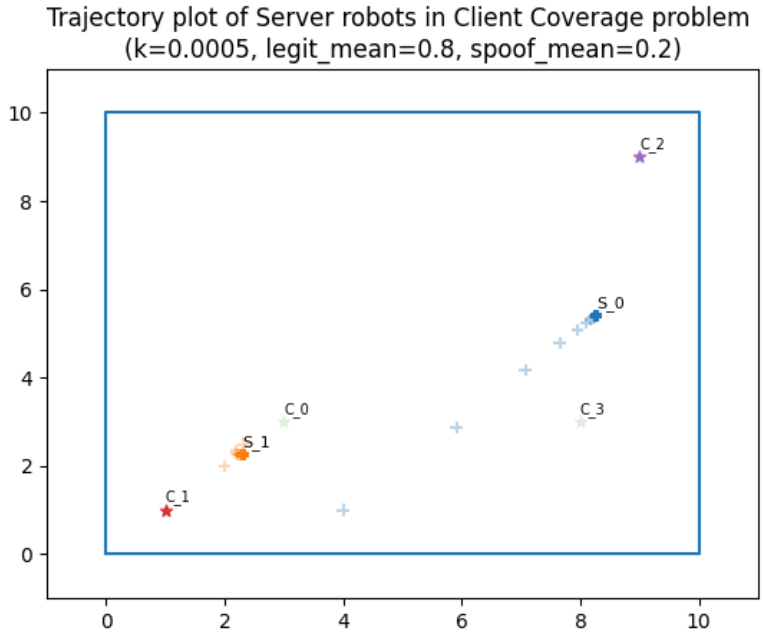
See code deliverable.
Note that for all the plots we added a label for the final position of each server and client. This is particularly useful for servers for which there is a high overlap in the trajectory and it is hard to visualize where in the clutter, they are located (fading colors are no longer visible)

b)



The importance function is the sum of the decomposed terms for each client such that each client position corresponds to a peak in the global importance function. Therefore, in a scenario with 4 clients and 4 servers, each one of the server should converge to the position of the clients. We can observe this behavior in the graph above where after 200 steps, the server robots (denoted S_i) have moved to the position of the clients (denoted C_i). Note that in this implementation, the system is suffering from the Sybil attack as there are 2 spoofed clients (Client 0 and Client 3) which are successfully disturbing the system as they cause servers (S_0 and S_3) to converged to the spoofed positions.

To better appreciate the Sybil attack behavior, we run the simulation with 2 servers (instead of 4 proposed in the given code) such that there are not enough servers for every client



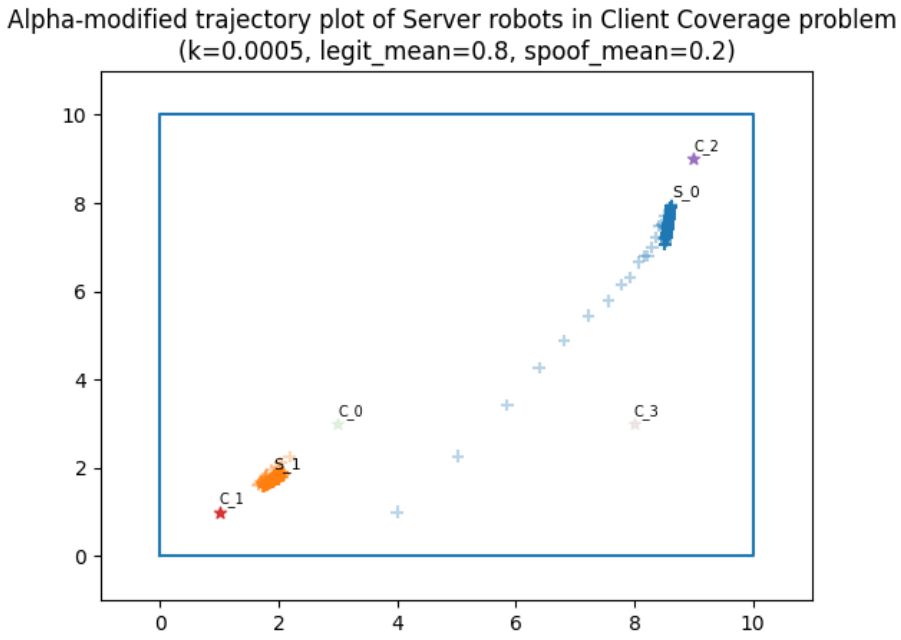
In this case, we can see that server S_1 converges to the midpoint between clients C_1 and C_0 and server S_0 converges to the midpoint between clients C_2 and C_3 which shows how the spoofed clients are hindering the system as S_1 should converge to C_1 only and S_0 should converge to C_2 only.

c)

By default, legitimate clients sample from a distribution with $\mathrm{E}(\alpha_i) = 0.8$ and spoofer clients sample from a distribution with $\mathrm{E}(\alpha_i) = 0.2$. According to the paper, for successful spoofer detection, a threshold ω_i must be found with the provable property of differentiating spoofed clients whereby spoofed clients are bounded below this threshold, i.e., $E[\alpha_i] \leq \omega$, and legitimate clients are bounded above this threshold $E[\alpha_i] \geq 1 - \omega$.
In this case, we can use the same threshold of $\omega = 0.5$ (as found in empirically in the paper) as it meets the requirements for spoofed clients to have a an expected α bounded above the threshold $1 - \omega$ (since $E[\alpha_i] = 0.8$ for legitimate clients and $0.8 \geq 1 - 0.5$). Similarly, spoofed clients are bounded below the threshold ω (since $E[\alpha_i] = 0.2$ for legitimate clients and $0.2 \leq 0.5$). In addition, 0.8 and 0.2 are in the range $(0, 1)$ which is the range that alpha values take. Due to these reasons, we can see that the proposed legitimate mean and spoofer mean might be good values.

It is worth mentioning that the standard deviation used when sampling will also play a key role when differentiating between spoofed clients and legitimate clients. This is because if standard deviations are large, there will be too much overlap between the distributions of *alpha* for spoofed and legitimate clients which means that some values of alpha will fall on the incorrect side of the threshold and will yield a misclassification. However, we know from the "Guaranteeing Spoof-Resilient Multi-Robot Networks" paper that the alpha values are determined using SAR fingerprints and Signal to Noise Ration which allow for clients to be distinguished using a threshold. Therefore, since the samplers from the problem are modelling the real life results of the confidence metric, we can choose a relatively small value for the variance such that there is not much overlap. In the problem, we are using a standard deviation of 0.1 such that all 99% of the legitimate clients have sampled alpha values in the range $[0.0, 0.4]$ and 99% of the spoofed sampled clients have alpha values in the range $[0.6, 1.0]$

d)

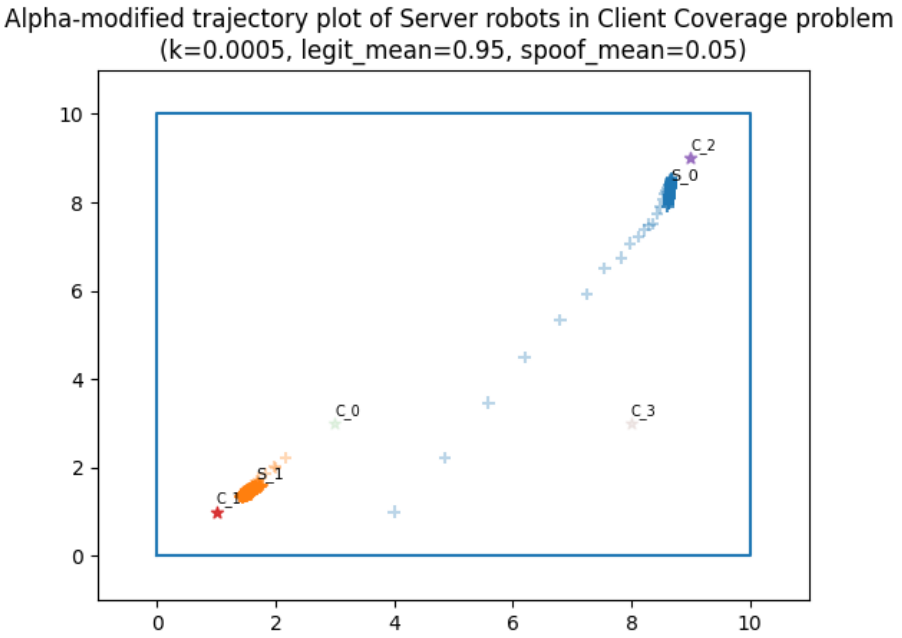


Running the simulation with an alpha-modified importance function shows improvement in the convergence of the servers towards the legitimate clients. Indeed, instead of converging to the midpoint between a legitimate client and a spoofed client, the servers move to a point closer to the legitimate clients. This shows that using the confidence metric *alpha* allows the system to be more robust against spoofing agents since it manages to reduce the impact that such agents have. This being said, we can see that the servers did not fully ignore the spoofed agents and did not converge to the exact location of the legitimate clients. This is due to the fact that the spoofed clients still carry some importance (even if it is less than the legitimate clients) and therefore, they still have a negative impact on the system (although such impact is reduced)

e)

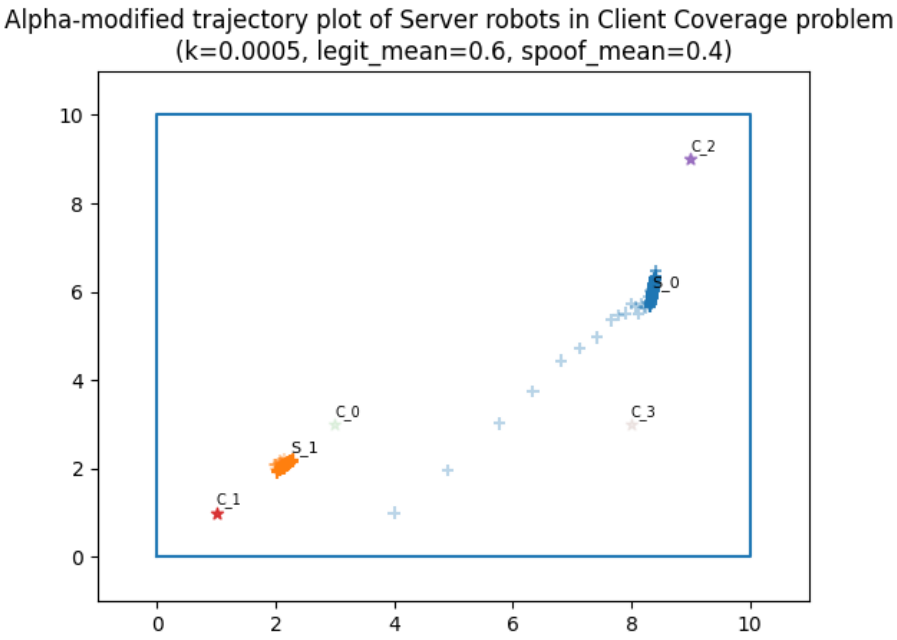
i)

We first test the scenario with more extreme mean *alpha* values with $legit\_mean = 0.95$ and $spoof\_mean = 0.05$



As expected, we can see that the impact of the spoofed clients is even more reduced than in the $(0.8, 0.2)$ case since in average, legitimate clients are given even more importance (average of 0.95) and spoofed client's importance is reduced (average of 0.05). On the graph, we observe that the final position of S_1 is closer to the legitimate client C_1 (compared to question part d.) and that the final position of S_0 is closer to the legitimate client C_2 (compared to question part d.)

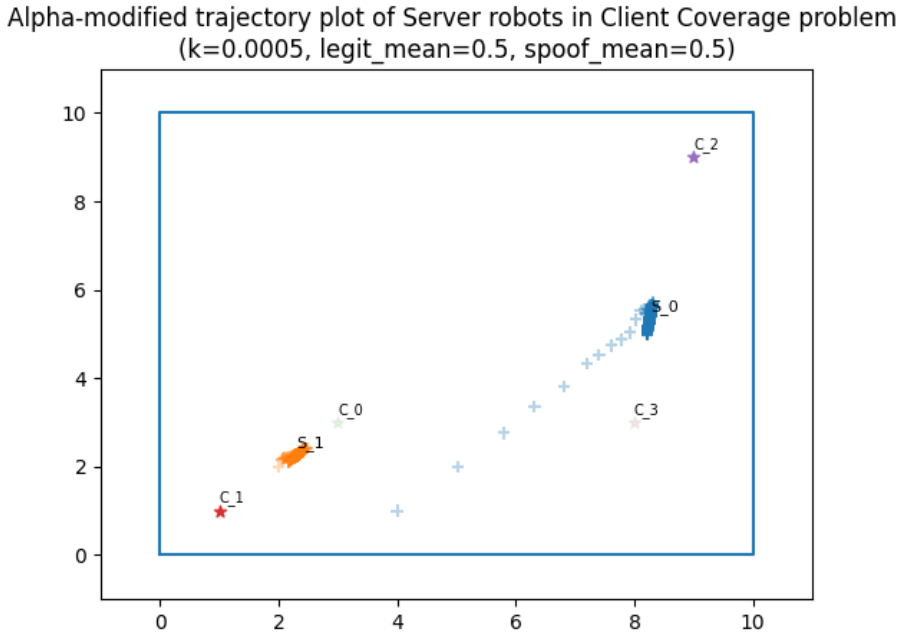
Additionally, we test the scenario with mean alpha values being closer to each other with $legit\_mean = 0.6$ and $spoof\_mean = 0.4$



We can see that the performance of the system is much worse in this second scenario than in the previous one (with $legit\_mean = 0.95$ and $spoof\_mean = 0.05$) and that servers converge to the midpoint between the legitimate client and the spoofed client. This is the worse case scenario in which spoofed agents are able to negatively impact the system as servers are not optimally covering the legitimate servers. This phenomenon is due to the fact that the confidence weights are sampled from two distributions that are similar with means close to each other and same standard deviation. In fact, the mean of the legitimate agents normal distribution falls within 2 standard deviations of the mean of the spoofed agents normal distribution and viceversa. Such overlap in the distributions will cause weights for spoofed agents and legitimate agents to be relatively close and, therefore, their importance will be weighed equally which means that the system is not able to discern between the 2. In some cases, the weight of a legitimate agent might even be below the weight of spoofed agents meaning that it is possible for spoofed agents to be given more importance than legitimate agents. Due to these reasons, the system is not being able to correctly discern between spoofed and legitimate agents, which causes the poor performance when it comes to resilience to spoofing attacks.

CAVEAT: It is worth noting the importance of the standard deviation of the sample distributions which is what causes the distributions to overlap. We explore this in the exploratory section below.

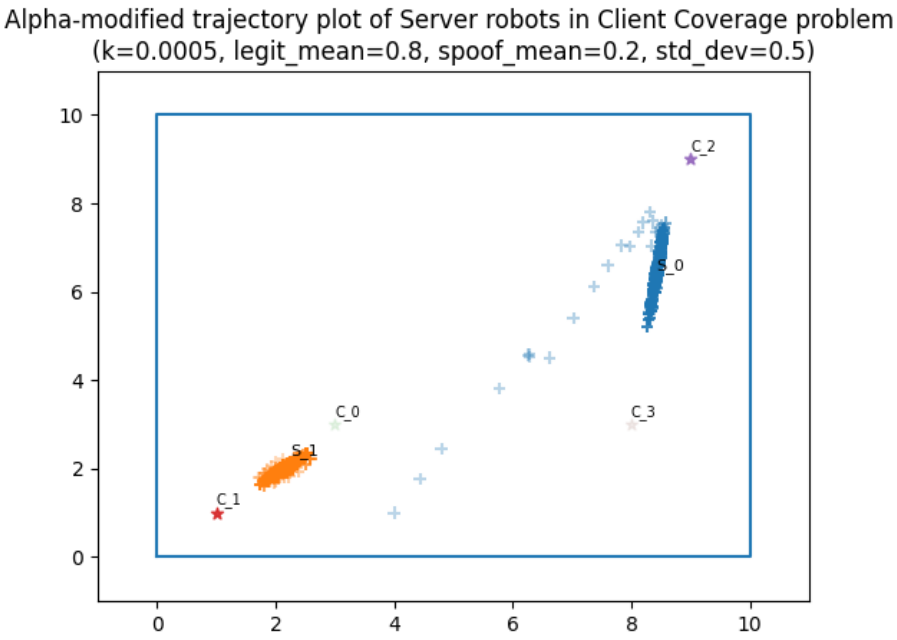
ii)Setting $spoofer\_mean = legit\_mean = 0.5$



We can see that in the scenario in which both spoofing agents and legitimate agents have the same mean value for the alpha sampler, the system performs worse and servers converge again to the the midpoint area between the Clients (just like in the non alpha-modified trajectory in question 1.b). This is due to the fact that spoofed agents and legitimate agents are given the same weights in the weighing of their importance which is equivalent of not giving weight at all (if we ignore the stochasticity of the sampling). Therefore, this alpha modified version yields a very similar result as the non-weighted version. If this were to happen in real life, it would mean that our technique for constructing the confidence value α would be completely useless as it wouldn't be making any distinction between spoofed and legitimate agents

1.1 Further exploration of the parameters

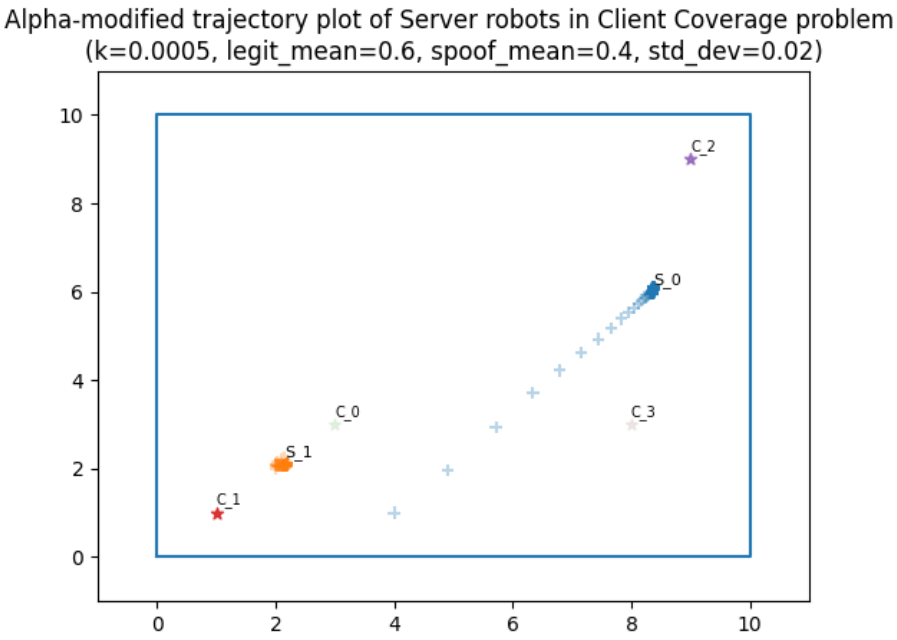
In this section, we explore the scenario with a relatively large standard deviation, simulating a real life case in which the SAR techniques for building α have a high variance (paradoxically means that the confidence metric has a high level of uncertainty). We test the case of a standard deviation of 0.5



We can see that in this case, the system performs worse than in the case with 0.1 standard deviation and that the Servers converge to the midpoint between Clients. This is due to the fact that if the values of α vary a lot it means that in some cases spoofed agents might have alpha weights higher than 0.5 and higher than the weights of the legitimate agents.
One could argue that the final position of Server 0 is actually closer to the legitimate client C_2 than to the midpoint. However, having such a high variance causes the server to not converge to a specific point but rather to an entire region through which it moves. Therefore, the final observation being closer to C_2 might be due to randomness in the sampling for the last iteration.

The observations above allow us to conclude that if the methods for determining α (SAR fingerprinting, etc) have a high uncertainty (don't converge to a small range of values), then spoofer detection can not be completed and the system is not robust against Sybil attacks. Therefore, even if on average the robots are given appropriate weights, when the variance of the weights is to high so as to create an over lap in the distributions, the systems fails.

Alternatively, we test a scenario with a small standard deviation of 0.02 but with close sampling means of 0.4 and 0.6.



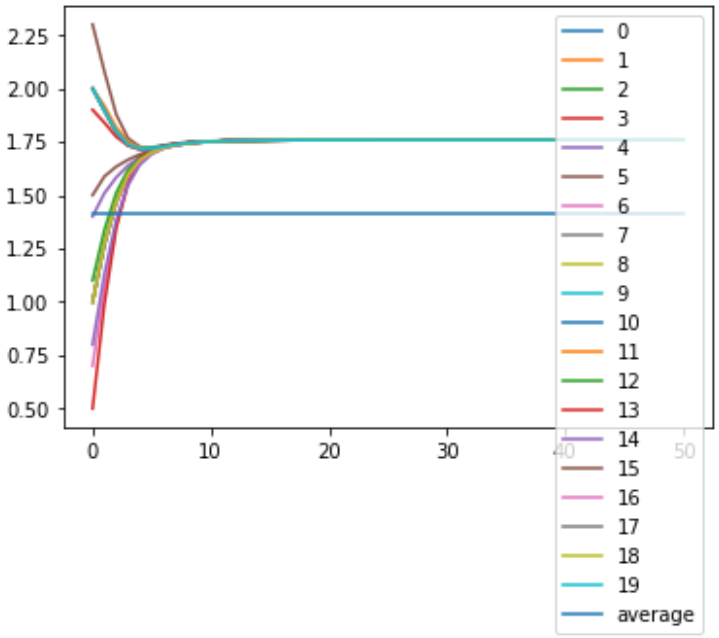
Surprisingly, the system fails again to account for spoofed agents as the servers converged to the midpoints between the legitimate and the spoofed clients.
Even though the threshold of $\omega = 0.5$ is strictly enforceable i.e. there is no overlap in the distributions and evaluating the value of *alpha* would allow to perfectly determine legitimacy, the system still fails. This is due to the fact that the value of α is used as the actual value for the weighting of the importance instead of being used in deciding whether an agent is spoofed or not (binary decision based on *alpha*). Therefore, as in the cases presented in earlier questions $spoofer\_mean = legit\_mean = 0.5$ and $spoofer\_mean = 0.4$ and $legit\_mean = 0.6$, the weights will be too similar for spoofed and legitimate clients and so will be the importance awarded.

To conclude this problem and the exploratory section, we observed that the values of the confidence metric α need to be sufficiently different in average if they are to be used as weights for the individual importance, in order for the system to be spoof-resilient. In addition, a real threshold separating spoofed agents from legitimate agents can only exist if the variance of the confidence metric is small i.e the system is certain about the confidence metric generated. Otherwise, the overlap in the distributions of α will cause the system to fail.

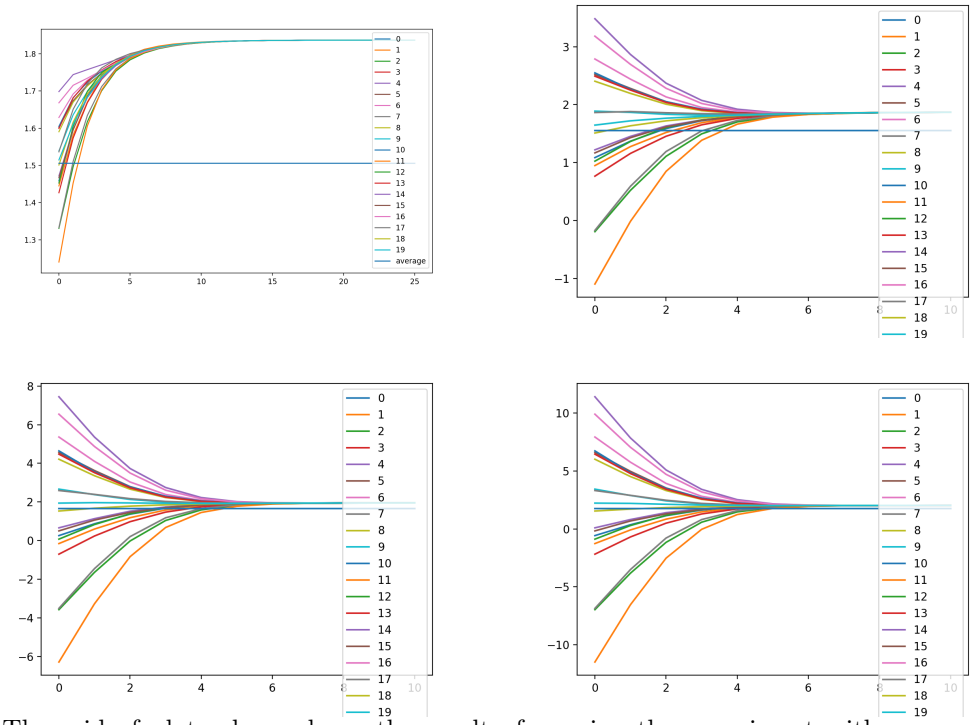
2 Using Spoofed information

a) Implement transition W

The plot below shows the result we obtain after implementing the transition_W function and running the algorithm for 50 iterations. The final convergence value obtained is 1.75806313 and the true average of the legitimate states is 1.41.

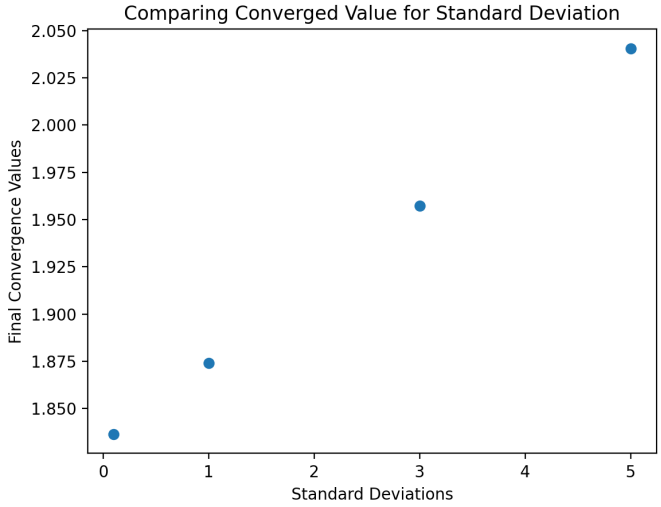


b) Setting the legitimate clients values using normal distribution with varying standard deviations



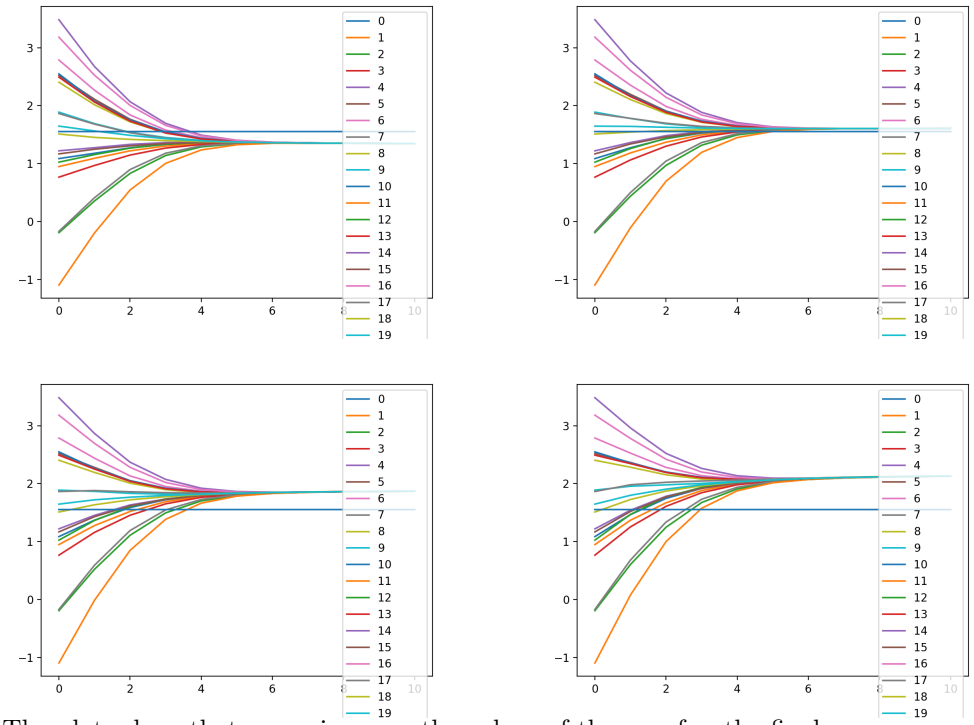
The grid of plots above shows the result of running the experiment with the standard deviation parameter in the distribution of the legitimate agents set to 0.1 in the top left, 1 in the top right, 3 in the bottom left and 5 in the bottom right. It is important to point out that the scale of the y-axis is different therefore, while it could seem that the bottom right plot (sd = 5) has the closest convergence to the true average, it is due to the larger range of values in teh y-axis (the plot below comparing the final convergence values shows this clearly).

For the rate of convergence, however, it is helpful to look at this grid of plots, noticing again the difference in the scale of the axis. Here, we observe that a higher standard deviation in the distribution of legitimate agents leads to a faster convergence rate since it takes 10 timesteps for convergence to be reached when sd = 0.1 compared 6 timesteps, 6 timesteps and 5 timesteps for sd = 1, 3 and 5 respectively.

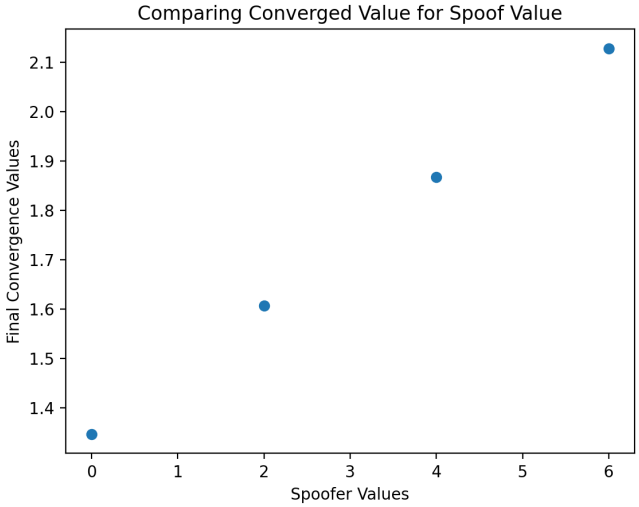


The plot above shows the different convergence values as we increase the standard deviation of the distribution of legitimate agents from 0.1 to 1, 3, and 5. As we can see in the plot, a larger standard deviation in the legitimate nodes leads to a decrease in the performance of the algorithm, since the final convergence value of the network is further from 1.5 (which is the mean of the distribution of the legitimate agents) and closer to the value of 4.0 of the spoofed agents.

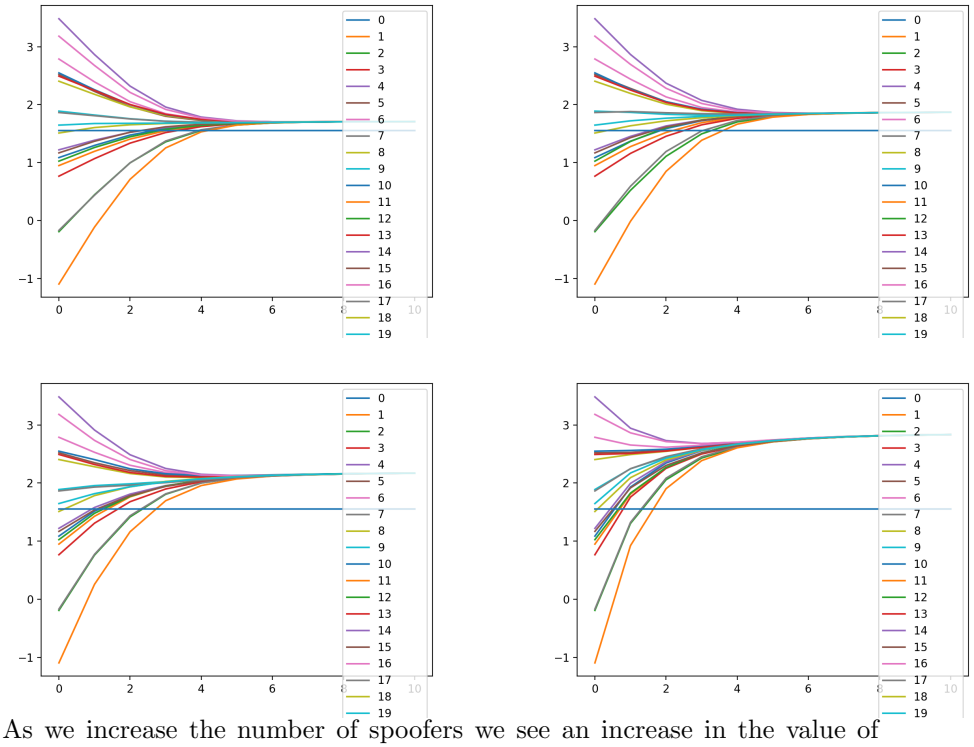
c) Varying the initial value of the spoofed agents



The plots show that as we increase the values of the spoofer, the final convergence value shifts up as well. There is not a noticeable change for how fast these. This makes sense because when the value of the spoofers are higher, they are tricking the legitimate nodes to converge higher and higher. The opposite happens when the spoof nodes have lower values and the legitimate nodes will converge lower than expected.

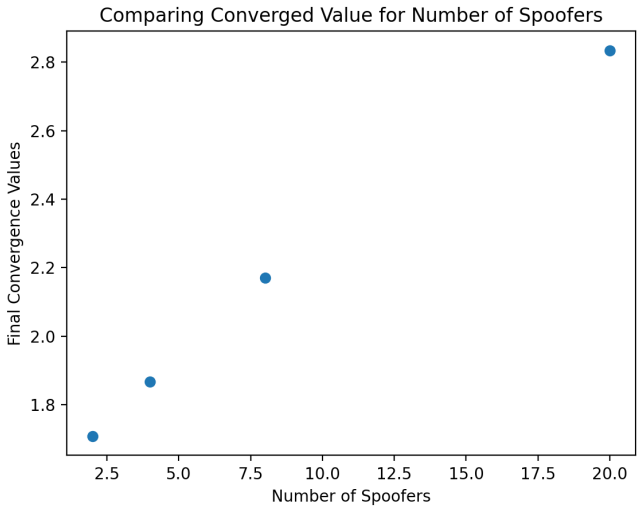


d) Varying the number of spoofer agents



As we increase the number of spoofers we see an increase in the value of our final convergence value. This makes sense because we have more nefarious

agents attempting to trick our legitimate nodes to converge to its value. We see a sharp rise when the spoofed nodes have a value of 20 since it is significantly higher than the other nodes. The abundance of nodes with the same value also leads the legitimate nodes to think they are real.

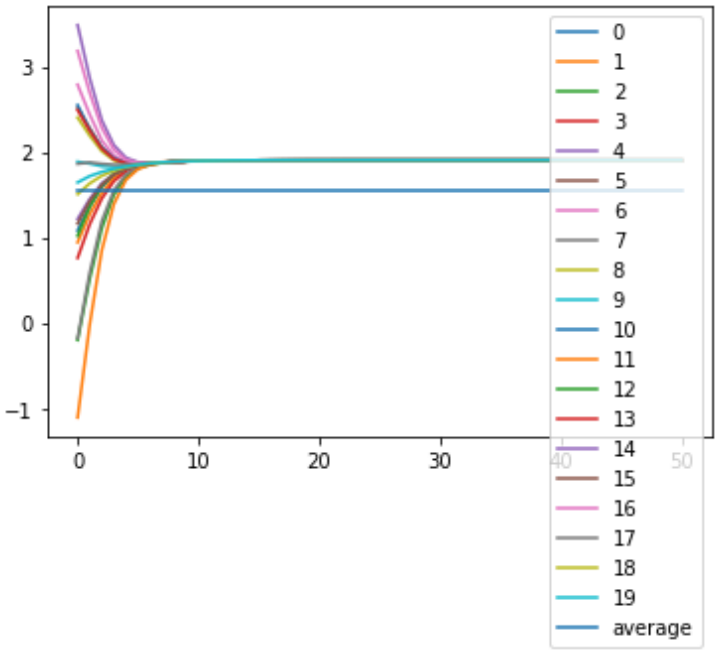


e) Changing spoofer value to follow a pattern

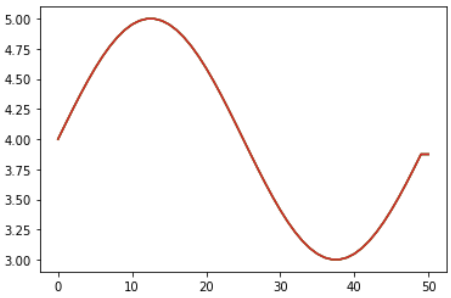
i) sine wave pattern

The plot below is the output of the program when we run for 50 iterations with the spoofer agents following a pattern of a sine curve with amplitude 1, displacement 4.0 and a period of 50. The convergence value for this experiment was 1.906702.

In the plot below we see almost no differnce resulting from changing the pattern for the value of the spoofer agents to a sine curve. The algorithm performs slightly worse since the final convergence value of 1.906702 is more far off from the true average of 1.41 than the final convergence value of 1.874032 that we obtain when running the same experiment but with the fixed spoof value at 4.0 (also for 50 iterations). We suspect that this slightly worse performance of the algorithm comes from the fact that the sine curve in the first iterations will lead to a higher value for the spoof agent which is even further than 4.0 is from the true average of 1.41 and changes in the first iterations will impact the convergence more.

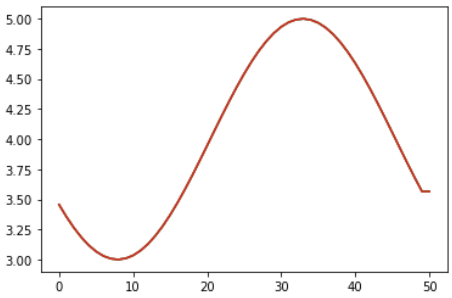


To confirm that the spoofer values was being updated correctly, we plotted their value at each step of the iteration, which generated the plot below (confirming correct implementation).



To confirm the intuition above regarding the sine curve and the larger value for the spoofer for the first iterations, we ran a similar experiment however with a phase shift in the sine curve such that in the first iterations the value would be made smaller than 4.0 and therefore would be closer to the true average of the legitimate agents. This led to a final convergence value of 1.78020116, which is not only better - closer to 1.41 - than the sine experiment above (as suspected) but it is also better than the benchmark we were working with when the spoofed value was fixed at 4.0.

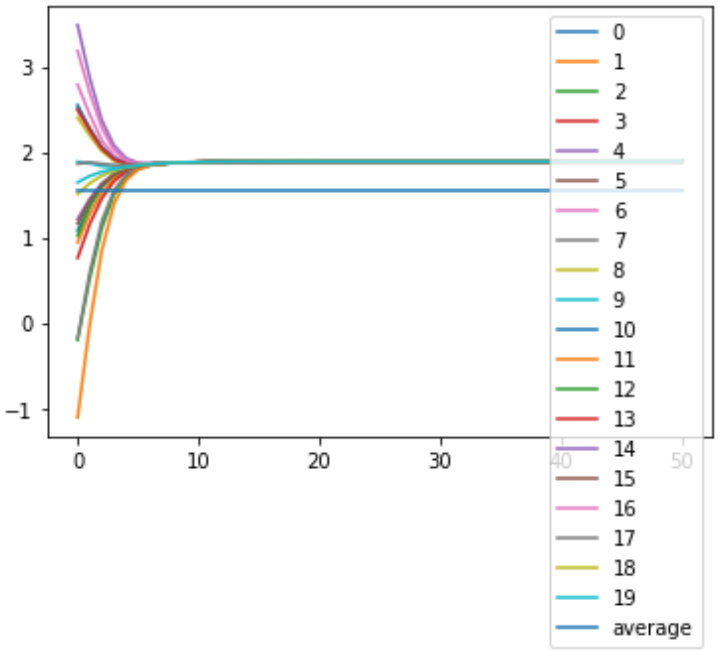
The image of the phase shifted sine curve is given by the plot below:



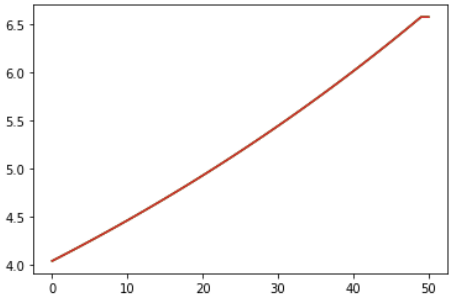
ii) exponentially increasing values

The plot below is the output of the program when we run for 50 iterations with the spoofer agents following a pattern of exponentially increasing values, starting with $x_S(1) = 4.0$ and $x_S(t+1) = 1.01 * x_S(t)$. The convergence value for this experiment was 1.89125.

We can see that this plot is very similar to the previous plot where the spoofer values all had a value of 4.0 and in fact, changing the pattern of the spoofer once again had a very small impact both in the final convergence value and in the convergence rate of the network. Compared to the sine pattern directly above the final convergence value was slightly closer to the true mean of the legitimate agents. This could be due to the the fact that for the initial iterations, the change in the value of the spoofer is more significant in the sine pattern than in the exponential pattern and changes in early iteration affect the algorithm more since the confidence in the α estimate is smaller.



To confirm that the spoofer values was being updated correctly, we plotted their value at each step of the iteration, which generated the plot below (confirming correct implementation).



Comments:

Initially, we were surprised by the result from experimenting with different patterns for the spoofed agents since we expected to see more significant changes in the convergence graphs. However, upon reflection, we realized that this results makes sense with the model we are using. In class we discussed two different models for identifying spoofer agents in a network, one based on analyzing the data being shared by each agent and the second model based on analyzing physical aspects (such as the fingerprint in the AOA profile) and using this to compute a value of α which estimates how trustworthy an agent is. The algorithm we have here follows the latter model and therefore when we change

the data (the value of the spoofer agent) according to the different patterns we don't see a significant change since the α value still encapsulates the lack of trust in data coming from the spooofed agents. As the algorithm progresses (timesteps), the change in the value of a spoofer agent away from the true average should impact even less in the convergence since the α estimate will be more accurate - i.e. in the first iteration there is little information to classify which agent is legitimate or spoofed but at each iteration the algorithm will have a better understanding of which agents are legitimate/spoofed.