Kenneth Bello and Hongshan Liu

Professor Shilad Sen

Collective Intelligence

Homework 3

Part 1 and Part 4:

In the Kleinberg’s 2 dimensional structure, we firstly build an dictionary keep tracking of all the neighbors of any node in the matrix, including the immediate neighbors and the long-range neighbors, to represent the network of that point. For any given point A, we can easily find out the immediate neighbors by going off 1 units in each four directions. However, in order to randomly pick a long-range neighbor while giving the Manhattan distance between two points a considerable amount weight, we have to use the following approach:

First, we randomly pick a manhattan distance from the point A using the following formula:

D = Rand(0,1)^aplha \* Max\_Manhattan\_Distance

Then, we loop through all the points whose Manhattan distance is D, and randomly pick one to be the long-range neighbor of point A.

With the setup ready, we can now start the simulation. Firstly, we randomly pick a sender and a recipient. Then, we loop through the neighbors of the sender to find out the neighbor with the shortest path to recipient. Then, we set that newly chosen neighbor as the new sender and run the function recursively, until the new sender is the same as the recipient, which means that the information is successfully transferred. When the recursion ends, we print out the number of recursion, representing the number of steps it has taken to transmit the message.

In the result, we observed that, for a 10\*10 grid, we are getting an average 3.9 steps of information flow, which is consistent with the prediction. Also, we are getting a median 5 step of information flow.

When we lowered the density down to 75%, success rate drops to somewhat around 90%. When we lowered the density down to 50%, success rate drops to somewhat around 70%.

In our model, we sometimes reach dead-ends. These dead-ends occur when a neighbors who currently has the message has no other users to relay the message to, thus ending the chain and causing the message delivery from the original sender to the recipient to fail.

Part 2:

According to the preferential attachment, people with more inbound connections are more likely to receive future connection. In order to implement the theory into the code, we modified the long range neighbor selection process in the previous code.

For the setup, we prepare a dictionary that keeps track of the inbound connection for every nodes in the graph. When picking the first long range neighbor in the graph, we use the same method we did in the Kleinberg’s model. However, every time we pick a new long range neighbor, we update the dictionary that we have set up. From now on, there are two possible outcome every time we pick a new long range neighbor. Half of the time, the system is going to find the neighbor according to the manhattan distance. The other half of the time, the system will look up the dictionary we set up earlier and randomly pick the nodes with top three incoming connections. That way, the nodes with more incoming connections will experience the rich-gets-richer effect while other nodes still have a chance of receiving incoming connections.

In this model, using a 10\*10 grid, we are getting an average 6.2 steps of information flow, and a mean 7 steps of information flow.

Part 3:

In this part, we look at the rich-gets-rich effect from another angle, and introduced a new concept history. During the simulation, we keep track of the number of times that interaction happens between two nodes. If the number is high, which means two person tend to exchange information frequently in the past, it’s more likely that they will exchange information in the future. In the model, we used a dictionary to keep track of the information of history.

Then, we devised a selecting algorithm that smartly combines the importance of distance and the importance of history connection. In the algorithm, we decide which neighbor to choose according to the value S:

S = Distance - History

Since in reality, the probability of choosing a neighbor should be proportional to the history and inversely proportional to the distance, in the model, the lower the value S is, the greater the chance we pick that node.

In the end, we apply the graph that we generated using the preferential attachment techniques on the history model, in order to get the most realistic simulation data.

For the result, at the first, when the history of every pairs are 0, the result is largely like the original model, where the message flows rather quickly and efficiently. However, when the history starts to stack up, we can observe the role that history plays in the model. When a large number of message is being delivered, the information starts to wander around because of the “old friends effect” that takes the information to places that is not necessarily closest to the recipient. As the progress goes, the step of information flow grows significantly.