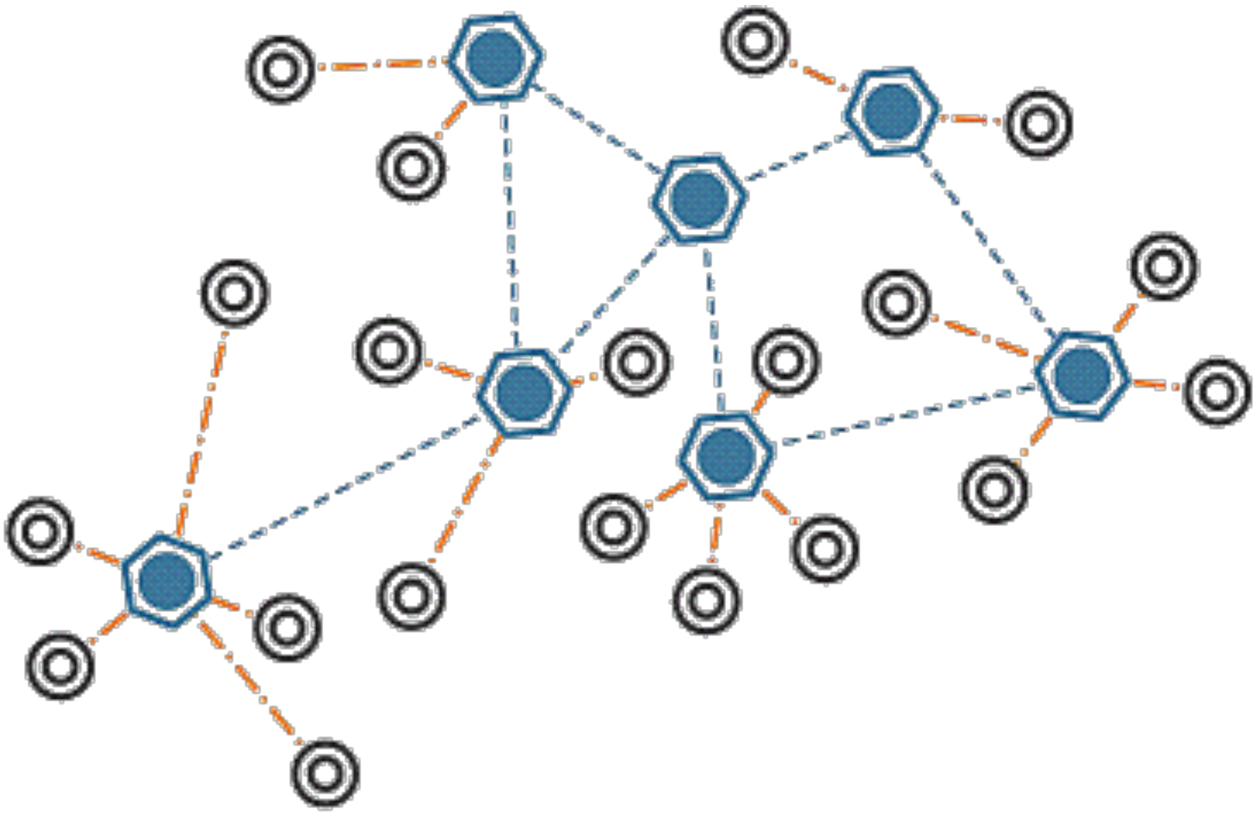
WSN FINAL EXAM



Topology Control Simulation

Dr. Yusuf Ozturk

Prepared by: Tasneem Singh (Red ID: 822046102)

24 December 2018

INTRODUCTION

The final exam is in continuation with where we left for the midterm. Hence, the initial few steps remains the same.

The midterm problem was aimed at developing a better understanding of concepts that go under developing a topology control simulation.

The first task of the problem set was to create a network of area 10,000 m x 10,000 m with 300 randomly deployed nodes.

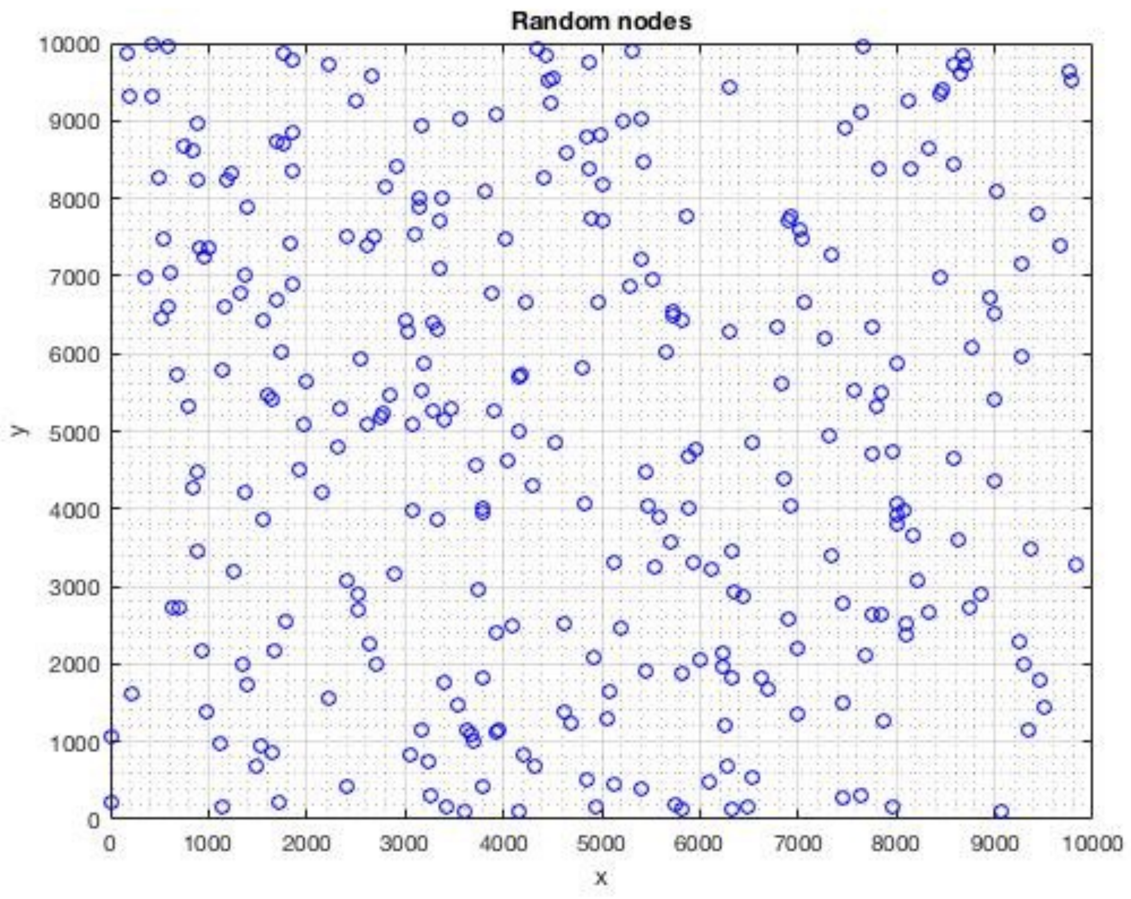


Fig 1: Random 300 nodes

BREAKING DOWN THE ALGORITHM FOR MIDTERM

- I started with first creating 300 nodes in a 10,000 m x 10,000 m area.
- Formed clusters around each node to get an estimate of how much overlapping can be observed. This helped form two different approaches that could've be implemented to solve the given problem.

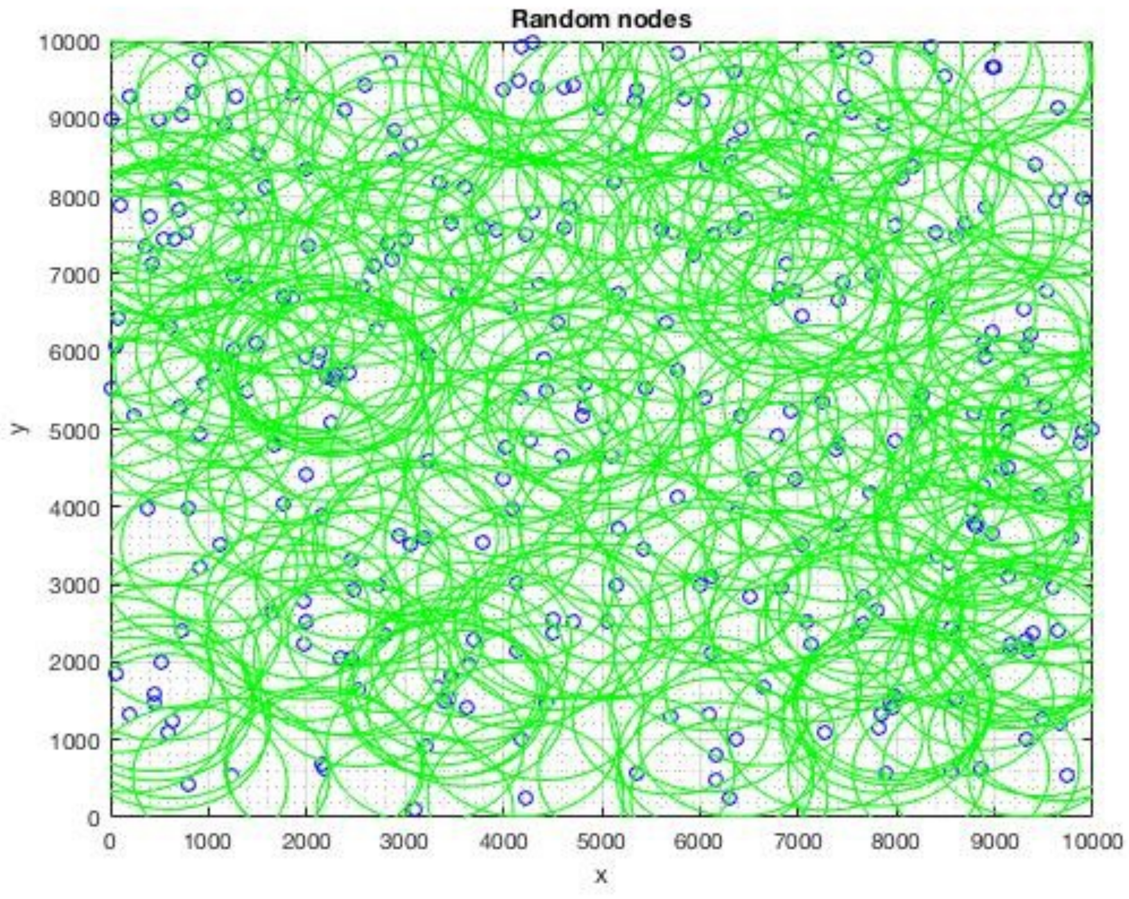
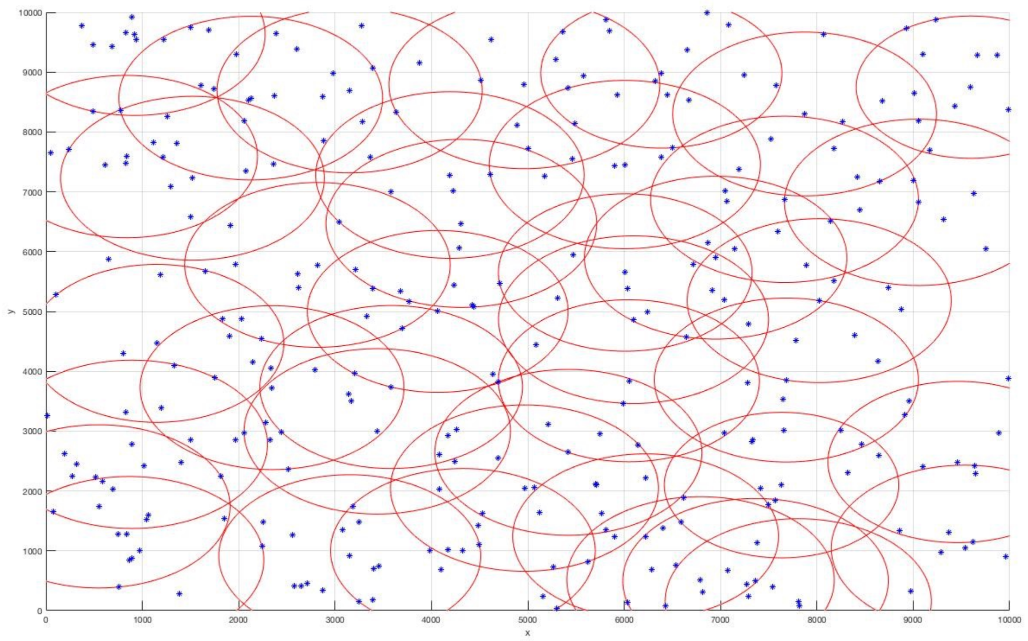


Fig 2: Cluster around every node

- K-means was chosen to form random Cluster centres across the given area. For this, I picked 40 number of clusters initially, which after later mixing and matching with different transmission powers gave me an almost full coverage at Tm_power ~ 14 dBm.

- Since, there's a high chance the centroids formed by using k-means would not necessarily be the real nodes, I calculate individual distances between all the nodes and all the cluster centres. The nodes that have the least distance with the respective calculated centroid becomes the new_centroid_coordinate.
    - To check which nodes are actually in the cluster of every centroid made, we calculate the rec_power and maintain a vicinity_distance and nodes_and_center_distance matrix. The nodes that satisfy the condition, rec_power > -89 are stored in inodes_and_center_distance ```matrix, which is 40x300 matrix, representing all the nodes that are within the reach of the 40 cluster heads made.
- Next, I make an orphan_coordinates matrix, which stores the node coordinates that have not been covered by the centroid clusters. Here, I consider two conditions: First, only those nodes are added which are represented by 0s in the intracluster_distance matrix. They are 0, because this matrix was derived from the rec_power matrix, which assigned 0s to all the nodes that didn't fell in the vicinity range of centroids. But intracluster_distance can also be zero, if the node in question is the centroid itself which means the distance will be 0 and hence rec_power = 0. So, I check if this is actually the case, confirm this with the new_centroid_coordinates and if they match, just put a zero in the orphan_coordinates matrix, representing that this particular node is not an orphan but actually a CH.



As shown in the above figure, there are 40 clusters operating at 14dBm, covering almost all the nodes in the given area.

- In the next for loop, I check if there are any orphan nodes remaining ie. if the matrix has any non-zero values remaining. If yes, then I'd like to put a cluster around it, so that I can cover more nodes because I have already set the transmission power to the highest value of 14dBm.
- The entire process is repeated, wherein I run the loop to check the orphan_coordinates matrix for any non zero values. If this condition evaluates to true, I increment the number of clusters by 1.
- After this, since the number of clusters have increased and I have applied k-means to it, I check minimum distance of every node to every k-mean calculated cluster head coordinate made, and those who are nearest are made the new CH. The results are stored in new_centroid_coordinates matrix.
- Again, the receiver power is checked for each node and viscircles is used to plot circles around the no_of_clusters computed.

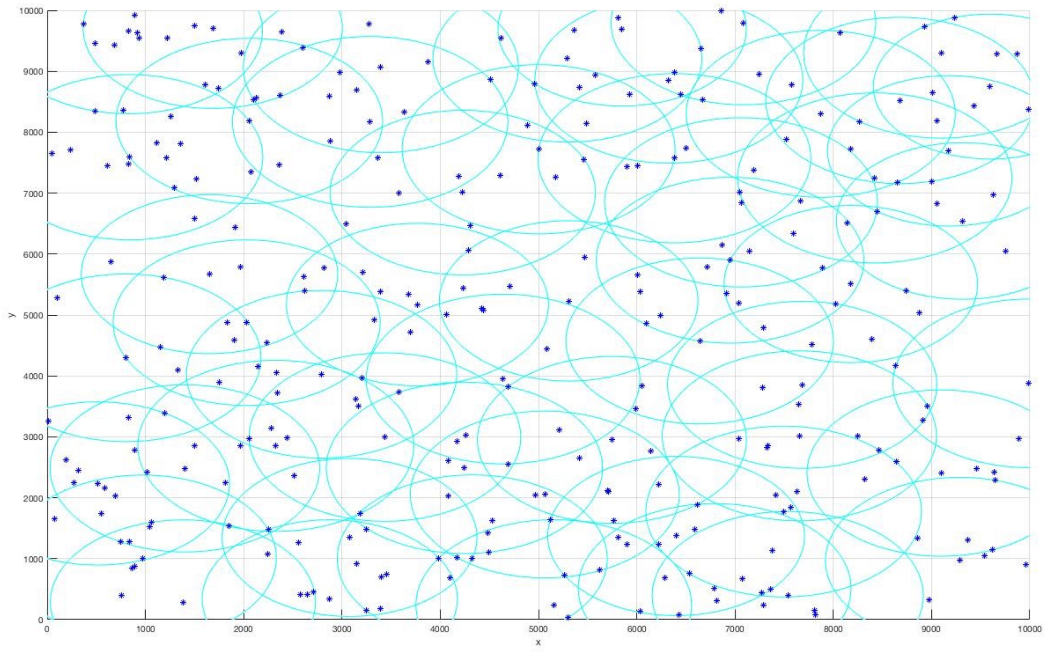


Fig 4: Increased number of clusters to accommodate all orphan nodes

The number of clusters in the above figure are 53, wherein all nodes have been covered and almost every cluster is connected to another. They are operating at transmission power of 14dBm.

- For further optimising the algorithm, I tried to create a power array with all the possible transmission values, ranging from -5dBm to 14dBm. Here, I wanted to check the orphan nodes and the centroids they are closest to. The power for those respective centroid were then supposed to be increased by 1 until they were operating at the max value of 14 dBm. The results obtained from this part had multiple clusters operating at 14dBm with circumferences that covered the entire area.

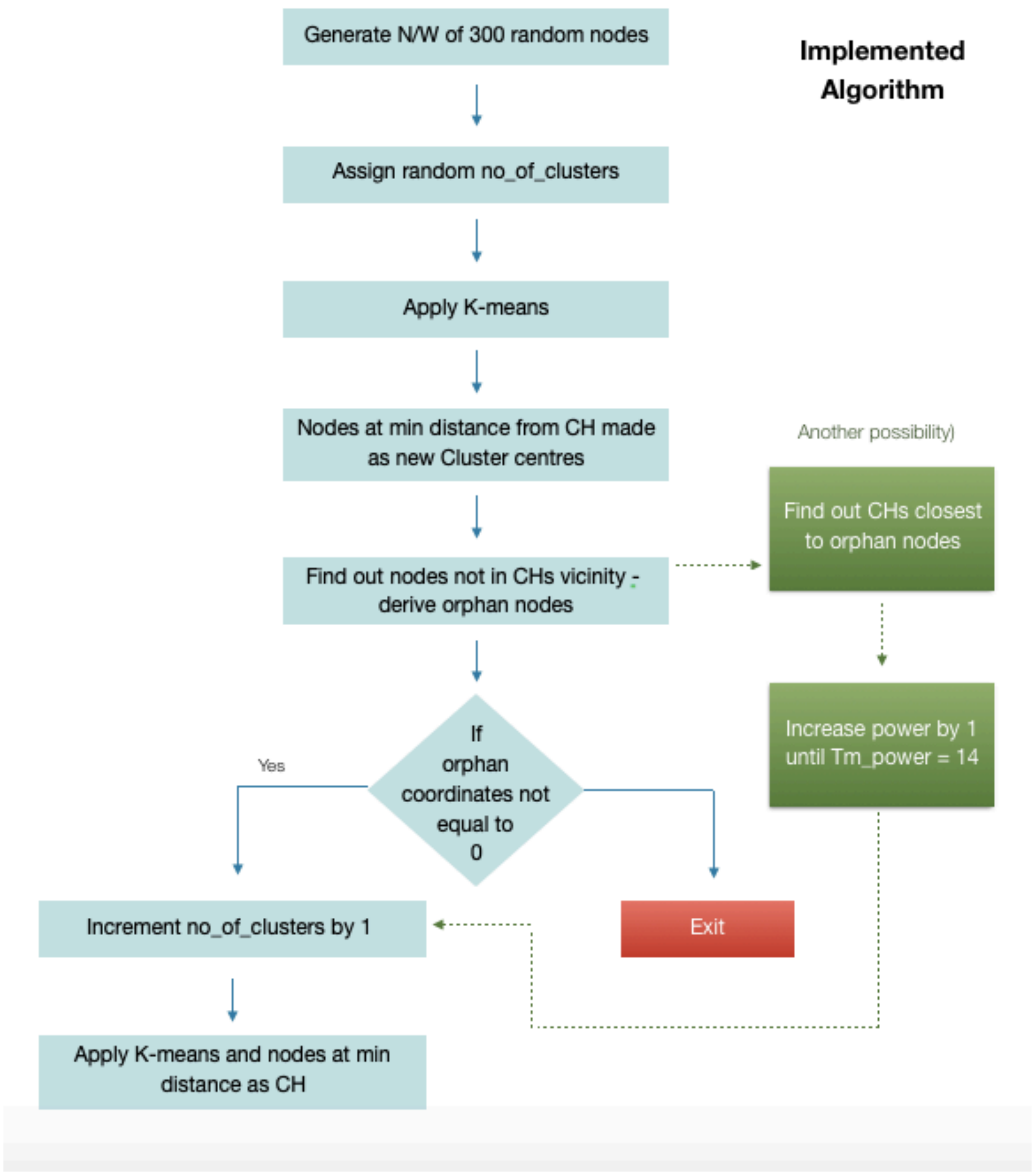

Generate N/W of 300 random nodes
Implemented Algorithm
Assign random no_of_clusters
Apply K-means
Nodes at min distance from CH made as new Cluster centres
Another possibility)
Find out CHs closest to orphan nodes
Find out nodes not in CHs vicinity - derive orphan nodes
Increase power by 1 until Tm_power = 14
Yes
If orphan coordinates not equal to 0
Increment no_of_clusters by 1
Exit
Apply K-means and nodes at min distance as CH

Fig 5: Flow of the midterm algorithm

//End of midterm

FINAL EXAM IMPLEMENTATION - ALGORITHM

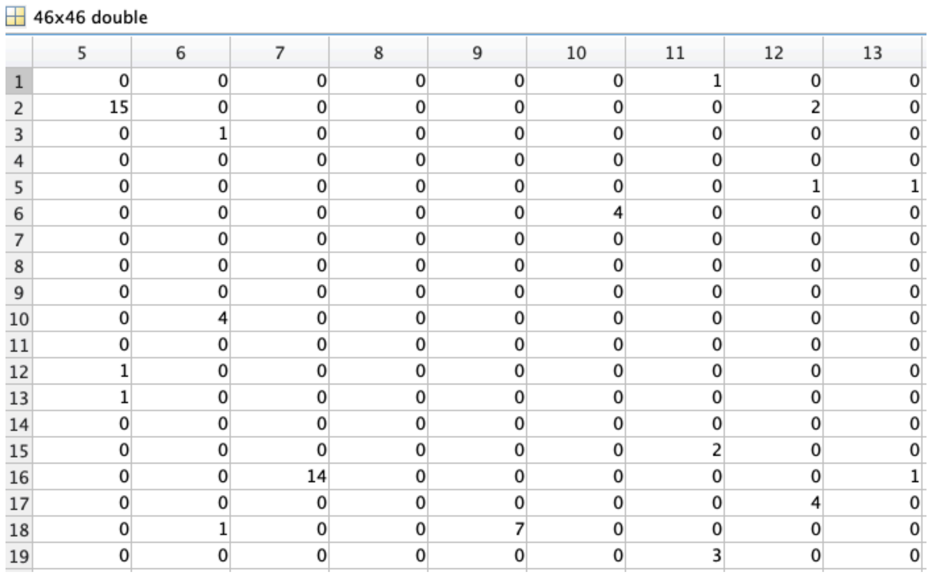
The main tasks for the final exam were:

1. Vary power levels so that instead of all clusters operating at power level of 14dBm, we can have clusters with varying power levels and still have no orphan coordinates.
2. Find common nodes shared between different clusters. Construct a matrix that keep a record of this and a graph - the thickness of edges depending on the number of nodes shared.
3. Follow Shortest Path (Djikstra's) algorithm for finding the shortest path from one cluster to another. This had to be done for 10,000 message transactions. To construct a graph for the same with respect to percentage of messages sent and no. of hops covered.
4. An energy consumed per cluster graph.

First and foremost, common nodes between different clusters were found. Using the receiver sensitivity, it was checked which all nodes were in the vicinity and which were not. These values were saved as 1s and 0s in the vicinity_mat correspondingly. For finding nodes, we had to compare successive adjacent column values and sum all the instances of wherever we found '1'. Hence we used three for conditions that were run over the transpose of the vicinity_mat since the dimensions varied.

Here's a snapshot of the common_nodes matrix:

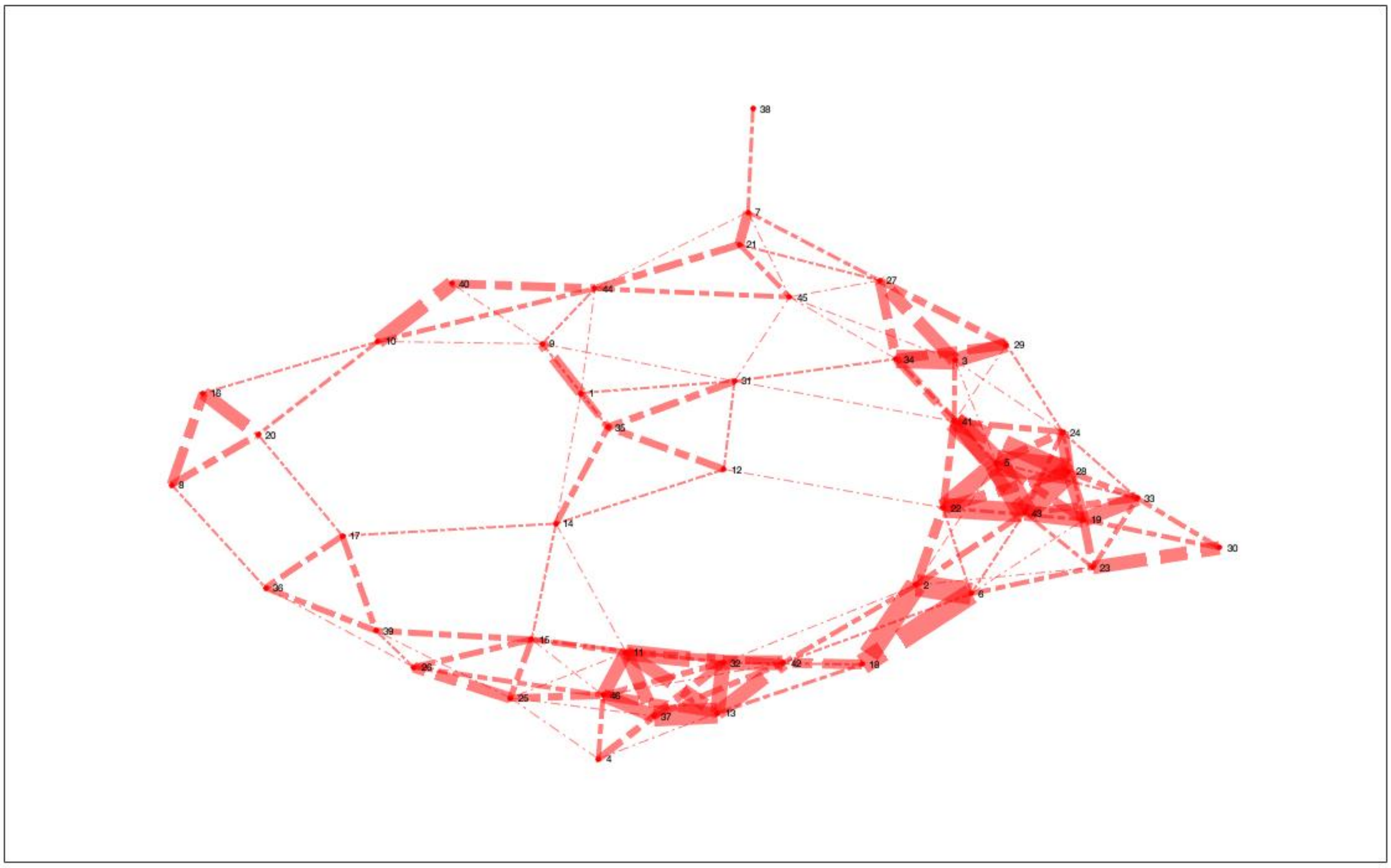
As evident, it's a square matrix of size - no_of_clusters x no_of_clusters. Hence, it gives the number of common nodes that every cluster shares with every other cluster. Common nodes are important to establish connectivity and a complete row/column of zeros indicate a lone disconnected cluster/node which is not the ideal case.


46x46 double

	5	6	7	8	9	10	11	12	13
1	0	0	0	0	0	0	1	0	0
2	15	0	0	0	0	0	0	2	0
3	0	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	1
6	0	0	0	0	0	4	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
10	0	4	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	0
13	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	2	0	0
16	0	0	14	0	0	0	0	0	1
17	0	0	0	0	0	0	0	4	0
18	0	1	0	0	7	0	0	0	0
19	0	0	0	0	0	0	3	0	0

Fig 6: Common nodes matrix

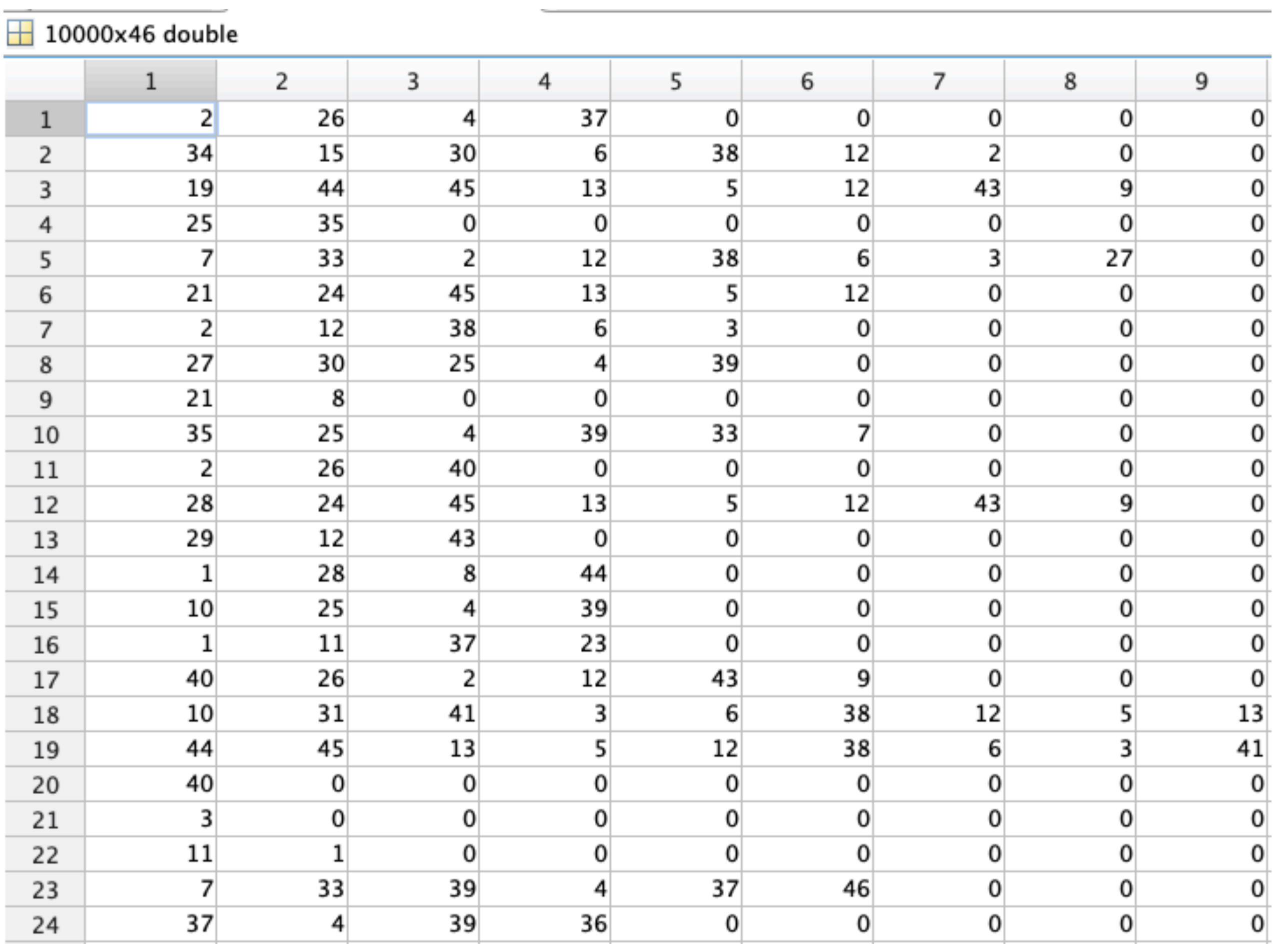
Using the common nodes matrix, a backbone graph(Fig. 7) is constructed, whose line width is dependent on the weight of the edges which in turn are actually the number of nodes



shared between every other cluster. The thickness of the connection between two matrices increases as the number of common nodes increases. As shown in the figure below, the 10

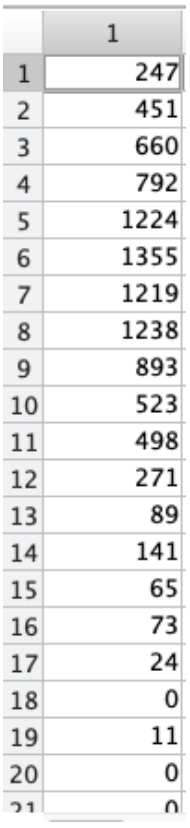
For the 10,000 random message transactions, we choose random to and from points from the given cluster centres and use the in built function called “shortest path” to compute the shortest distance from the to node to the from node. The hop matrix keeps a track of the number of hops that take place for all the 10,000 transactions. The shortest path matrix stores the complete path from every random to and from node.

The following id the shortest path matrix obtained. Here the first index for every row or column represents the ‘from’ cluster and the last represents the ‘to’ cluster.



10000x46 double

	1	2	3	4	5	6	7	8	9
1	2	26	4	37	0	0	0	0	0
2	34	15	30	6	38	12	2	0	0
3	19	44	45	13	5	12	43	9	0
4	25	35	0	0	0	0	0	0	0
5	7	33	2	12	38	6	3	27	0
6	21	24	45	13	5	12	0	0	0
7	2	12	38	6	3	0	0	0	0
8	27	30	25	4	39	0	0	0	0
9	21	8	0	0	0	0	0	0	0
10	35	25	4	39	33	7	0	0	0
11	2	26	40	0	0	0	0	0	0
12	28	24	45	13	5	12	43	9	0
13	29	12	43	0	0	0	0	0	0
14	1	28	8	44	0	0	0	0	0
15	10	25	4	39	0	0	0	0	0
16	1	11	37	23	0	0	0	0	0
17	40	26	2	12	43	9	0	0	0
18	10	31	41	3	6	38	12	5	13
19	44	45	13	5	12	38	6	3	41
20	40	0	0	0	0	0	0	0	0
21	3	0	0	0	0	0	0	0	0
22	11	1	0	0	0	0	0	0	0
23	7	33	39	4	37	46	0	0	0
24	37	4	39	36	0	0	0	0	0



	1
1	247
2	451
3	660
4	792
5	1224
6	1355
7	1219
8	1238
9	893
10	523
11	498
12	271
13	89
14	141
15	65
16	73
17	24
18	0
19	11
20	0
21	0

The message matrix represents the count which the random source nodes took to reach the destination nodes. As evident for this case, majority of the nodes took somewhere around 5 to 8 hops to reach their destinations.

Using this matrix we get the percent_message_mat which gives us the relation between number of hops and percentage of messages delivered.

The percent_message_mat is obtained by dividing the message_mat by 10,000 and multiplying the whole thing by 100.

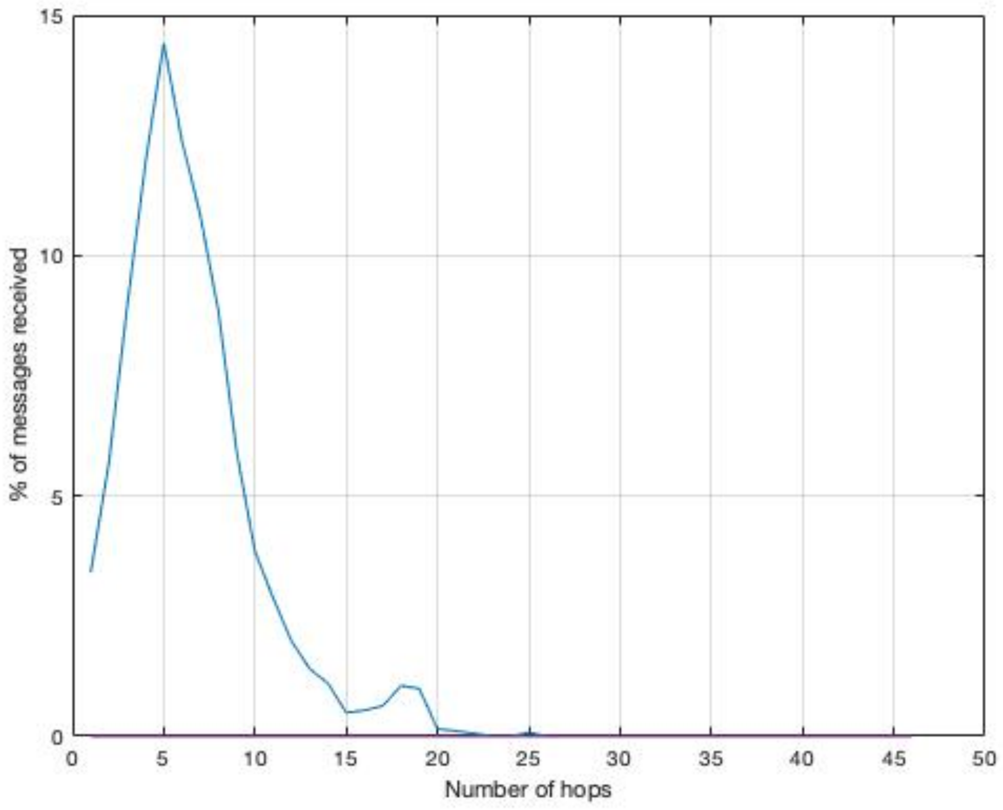


Fig. 9: Percentage of messages delivered vs no. of hops

For optimising power for different clusters, we start with various reference matrices to keep a track of previous values in case we want to revert back to them if things don't work such as in ideal case scenarios. At first, we give different power values to the tpower1 matrix with the help of a loop that loops of the power matrix which is nothing but the available power levels to us in descending order.

We go about the whole routine again, where in we check and calculate the distance,fspl,margin and receiver power.

Using the values obtained we again make the vicinity matrix. Also, we have cluster_power_mat that stores powers associated with each cluster.

Using the above calculated entities, we come up with common nodes for clusters now operating at different power levels.

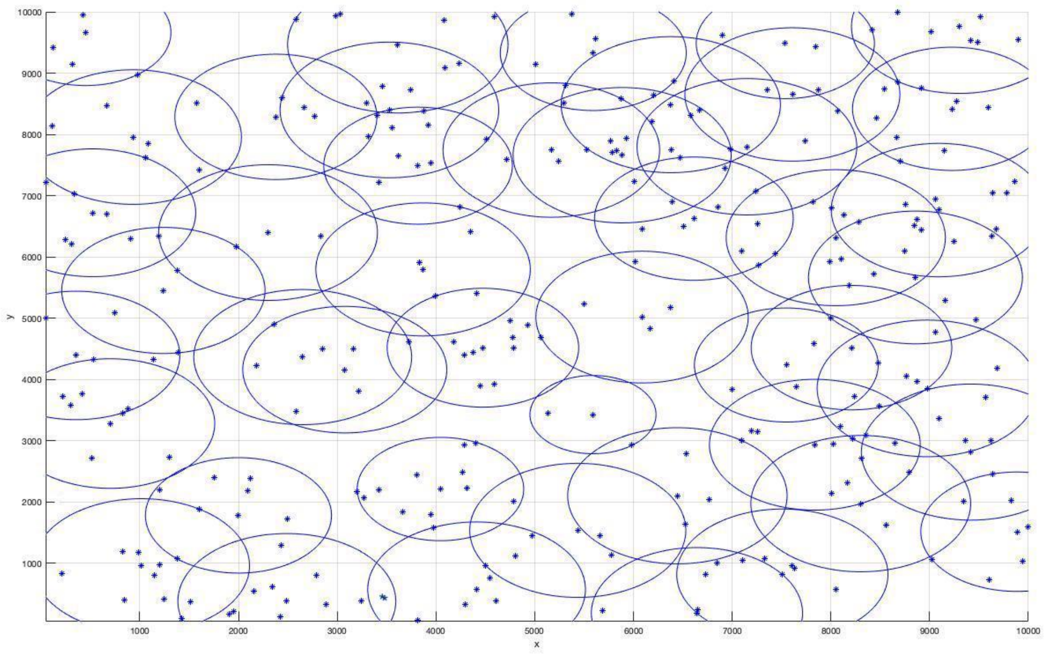


Fig. 9: Clusters operating at different power levels

Here we check if there are nodes that are not in the vicinity of any cluster head or clusters which have no common nodes with any of the other existing clusters. If the condition is true, we change the power at which these clusters operate back to 14dBm.

All the number of clusters are more(~46) but they cover all the nodes and operate at different power levels.

To calculate the per cluster power consumption, we make use of the shortest_path_matrix and use the formula dBm_to_mW(sb,1)=10^((ref_tpower1(sb,1))/10); where sb is a loop that runs from 1 to the no. of clusters. This gives us the power consumption numbers for each cluster. The resultant is the graph below.

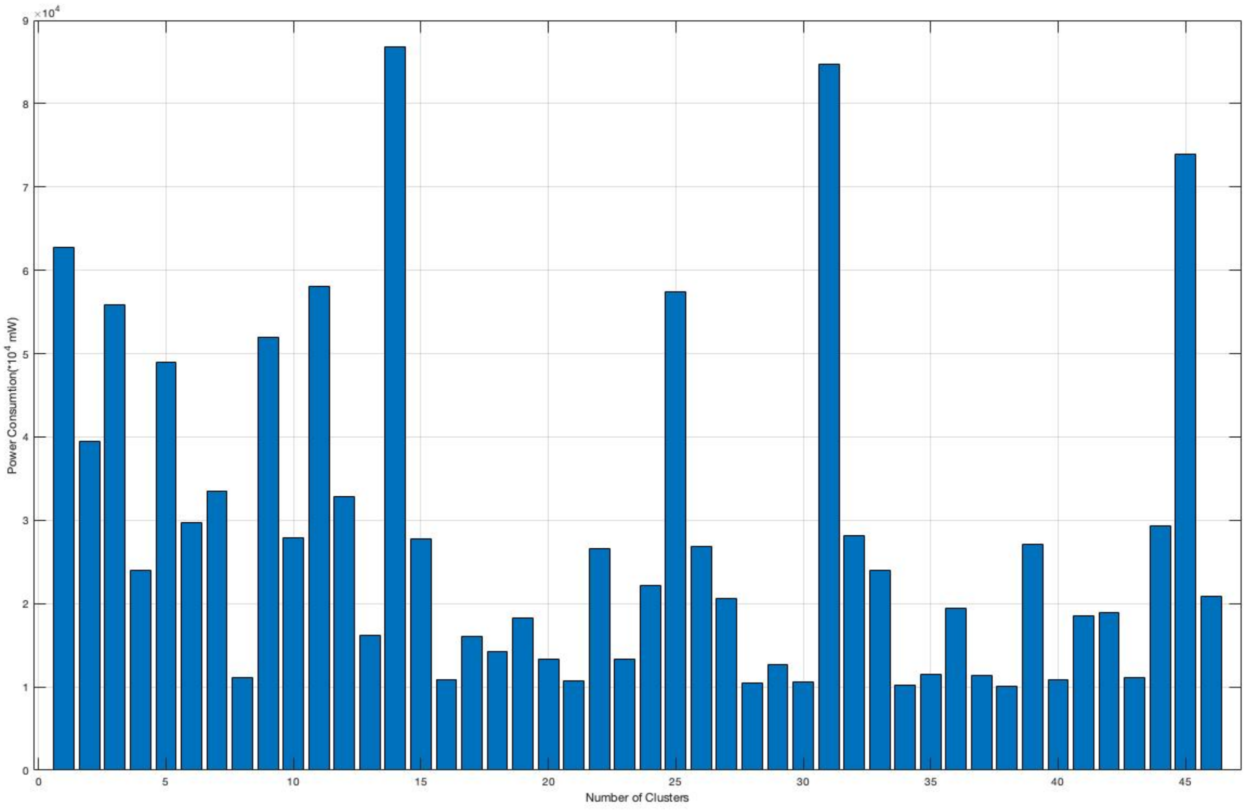
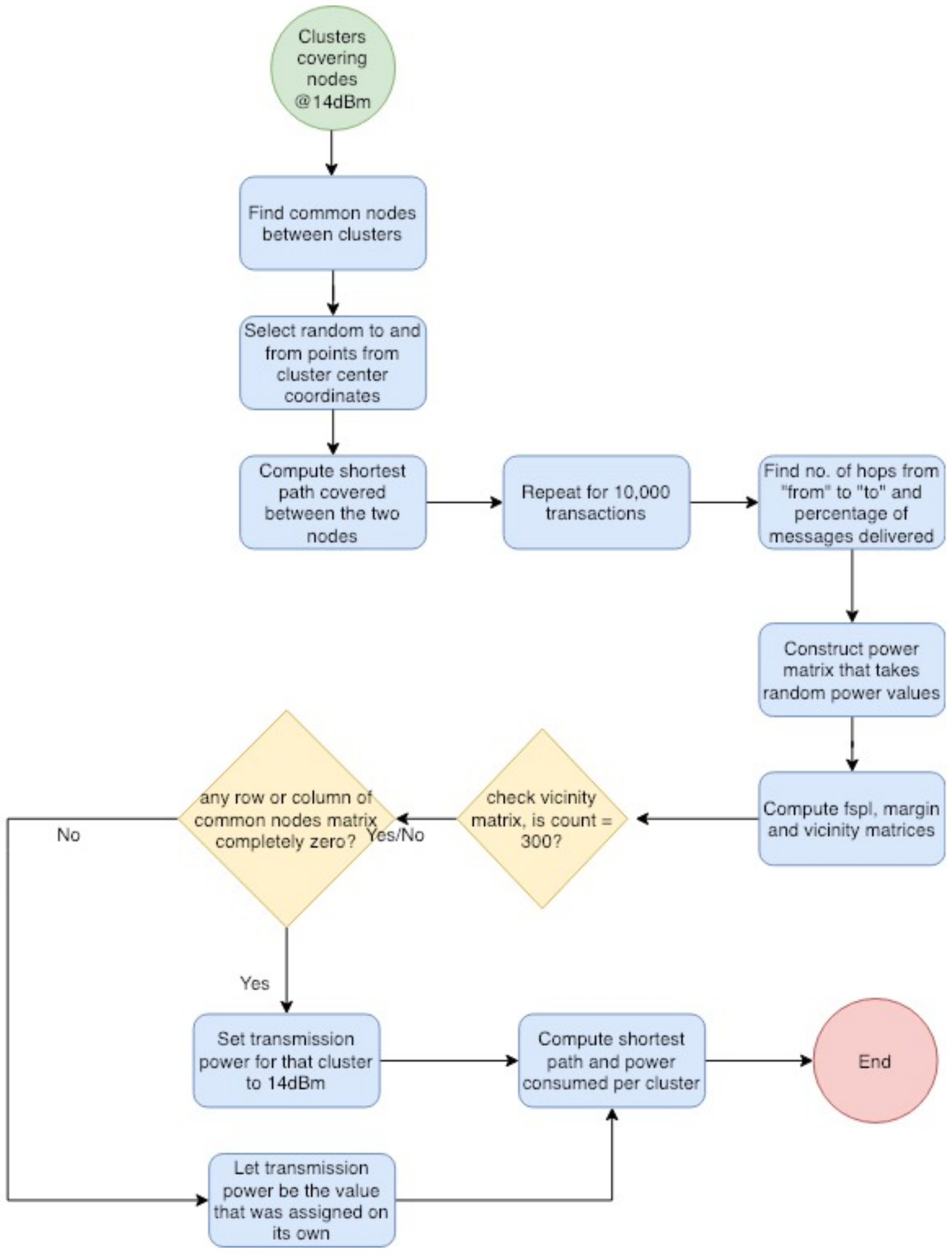


Fig. 10: Power consumption per cluster

BASIC FLOW OF THE ALGORITHM


Clusters covering nodes @14dBm
Find common nodes between clusters
Select random to and from points from cluster center coordinates
Compute shortest path covered between the two nodes
Repeat for 10,000 transactions
Find no. of hops from "from" to "to" and percentage of messages delivered
Construct power matrix that takes random power values
Compute fspl, margin and vicinity matrices
check vicinity matrix, is count = 300?
Yes/No
any row or column of common nodes matrix completely zero?
No
Yes
Set transmission power for that cluster to 14dBm
Compute shortest path and power consumed per cluster
End
Let transmission power be the value that was assigned on its own

CONCLUSION

With the simulations we observed that covering all the nodes with clusters while making sure that all nodes can communicate with each other is a tradeoff between power and number of clusters. On one hand, you can have lower number of nodes operating at high transmission power whereas on the other you have more number of clusters operating at lower power. The disadvantage of having high number of clusters is extensive overlapping which in turn consumes larger power. The disadvantage of increasing CH transmission power for all the cluster heads is that with the addition of more number of nodes(if in future), it will become a little more complex and a bit unfeasible because then you probably will end up with a some really clusters encompassing your entire network.

The algorithm for the final exam tries to find a middle ground by trying to limit the number of clusters operating at different transmission powers, by also ensuring that there are no orphan nodes left. The algorithm takes into account the concept of vicinity and common nodes matrices which are vital in establishing that the nodes are indeed connected with each other ie. they can communicate in spite of residing in different clusters. The algorithm was able to cover almost all the nodes besides working on the given deliverables. The limitation in this approach is the case of overlapping which is due to a little more number of clusters constructed.