**FOA Project Midterm Report submitted by Shambhavi Vikram Kulshrestha, Deepika Patra, Chandu Kamanuru, Vinay Madhav Jonnalagadda**

Based on the paper- Budget Constrained Relay Node Placement Problem for Maximal “Connectedness” by Anisha Mazumder, Chenyang Zhou, Arun Das and Arunabha Sen

**INTRODUCTION:**

This paper describes the possible solutions to the relay node placement problem in the wireless sensor network. In the given scenario, there are several sensor nodes that are placed in an area and each of them has their sensing radius R which makes it easier for them to communicate with another sensor lying within their radius. Taking this into consideration, all the sensors which are communicating with each other form a connected component.

The goal of this paper is to place the fewest number of the relay nodes in the deployment area so that the network formed by the sensor nodes and the relay nodes is connected. There are two metrics for connectedness in an incomplete graph furnished by this paper. The first definition of connectedness tries to decrease the number of connected components while the second one tries to maximize the size of the largest connected components.

We tried to implement the two heuristic based algorithms which place relay nodes to have a higher degree of connectedness of the network based on the two metrics definitions.

**PROBLEM FORMULATION:**

**Given:**

1. The locations of sensor nodes as a set of points P on an Euclidean Plane.
2. The communication range of sensors R.
3. A graph could be constructed G= (V, E) where all sensors are vertices and two sensors share an edge if their distance is less than or equal to R.

**Our goal:**

Placement of the relay nodes in such a manner that the new graph **G’ = (V’, E’)** is as connected as possible making it a connectedness**-maximization problem.** We have taken two definitions of correctness which have separate problem formulation from the paper.

* **Budget Constrained Relay node Placement with Minimum Number of Connected Components (BCRP-MNCC)** Given the locations of n sensor nodes in the Euclidean plane P = {p1,p2,...,pn}, positive integers R, C, and a budget B1 on the number of available relay nodes, is it possible to find a set of Q = {q1,q2,...,q|B1|} points in the same plane where relay nodes can be deployed, so that the number of connected components in the graph G0 = (V0, E0) corresponding to the point set P and Q is at most C? \*
* **Budget Constrained Relay node Placement with Maximum size of Largest Connected Component (BCRP-MLCC)** Given the locations of n sensor nodes in the Euclidean plane P = {p1,p2,...,pn}, positive integer R, C, and a budget B2 on the number of available relay nodes, is it possible to find a set of Q = {q1,q2,...,q|B2|} points in the same plane where relay nodes can be deployed, so that the size of the largest connected component in the graph G0 = (V0, E0) corresponding to the point set P and Q is at least C? \*

\*both the definitions are taken from the paper mention above.

The above two problems are special cases of the STP-MSP problem. The paper tries to find approximate heuristic-based solutions using a Minimum Spanning Tree approach which we have tried to implement in code.

**################### Algorithm 4#################################**

**Algorithm 4: Heuristic for BCRP-MNCC problem:**

* The heuristic solution implemented by the paper was based on Minimum Spanning Tree on the terminal points i.e. sensor nodes. In this algorithm, a complete graph was constructed which had terminals points with weights on the edges. The weighted edge e connecting two nodes is equal to the Euclidean distance and is divided by a communication range R.
* **w(e)=[length(e)/R]** **-1**, where w(e) is no. of relay nodes that will be required to enable the communication between the sensor nodes at the two ends of this edge.
* After this, the graph was used to make MST where the length of its edge was compared with the communication range R and if it was at most R then those two sensors connected by that edge needed no relay node for communication lest if the length was greater than R; it required relay nodes.
* The calculated relay nodes placed on the edge to enable the communication between two sensor nodes will be equal to w(e).
* If the budget on the number of available relay nodes is sufficient the connected component is one otherwise we learn that we are short of edges and we remove the edges of T’ until the number of relay nodes becomes less than or equal to the budget.
* **By removing one edge from the MST we are increasing the number of connected components by one.**

**###################Algorithm 5################################  
 Algorithm 5: Heuristic for BCRP-MLCC problem:**

* Taking the inputs of the sensor nodes as x,y coordinates from a csv file and adding them as nodes onto an empty graph.
* Assigning weights **w(e)=[length(e)/R]** **-1**, where w(e) is no. of relay nodes that will be required to enable the communication between the sensor nodes at the two ends of this edge.
* After this, the graph was used to make MST where the length of its edge was compared with the communication range R and if it was almost R then those two sensors connected by that edge needed n
* Create an n-MST using Prim’s Algorithm for the given terminal points, i.e. sensor nodes
* From the n-mst created, Iterating over k=n to 2, create an approximate k-mst and checking if the budget is matching the no. of nodes required for this k-MST
* If yes, then return this MST

**4.Problem Solution**

**4.1. Implementation of BCRP-MNCC Algorithm (Algorithm 4)**

As described in section 4.1, the heuristic approach has been beneficial for budget constrained relay node placement problem. For BCRP-MNCC algorithm, the best heuristic is to remove the edge which was contributing maximum to the total weight of the graph.

Following is the algorithm:

1: Create an MST T’ on the set of given terminal points P.

2: Assign each edge e of T’ a weight of w(e) =-1

3: while > B1 do

4: Remove the edge that has the maximum weight

from T’; breaking ties arbitrarily

5: end while

6: Return the resulting forest obtained from T’

In our code, the user is asked to enter the communication range and budget of relay nodes. The code is reading a sensor\_location.csv file to get the x and y coordinates of all nodes including the names of nodes. After getting the information of sensor nodes, a complete graph\* is created along with an assignment of the respective weight to all the edges. The function for creating a minimum spanning tree is called to create MST in the complete graph. We are using Prim’s algorithm here. Once the MST is created, the code is following the rest of the steps in the above algorithm.

**4.2. Implementation of BCRP-MLCC Algorithm (Algorithm 5)**

1: Create an MST T’ on the set of given terminal points P.

2: Assign each edge e of T’ a weight of w(e) =-1

3: for i in range n to 2 do

4: Construct an i-mst

if sum of weights <=Budget

return Resulting tree

5: end while

In our code, the user is asked to enter the communication range and budget of relay nodes. The code is reading a sensor\_location.csv file to get the x and y coordinates of all nodes including the names of nodes. After getting the information of sensor nodes, a complete graph\* is created along with an assignment of the respective weight to all the edges. The function for creating a minimum spanning tree is called to create MST in the complete graph. We are using Prim’s algorithm here. Once the MST is created, we are creating a k-MST over here by trying to remove the edge with the highest weight.

**5. Results**

**5.1. Analysis of BCRP-MNCC Algorithm (Algorithm 4)**

As mentioned in the previous section, Python code was implemented for the Algorithm 4 of BCRP-MNCC. Here, we are giving the reports generated for different number of sensors. For the number of sensors=10, the graph generated is the following:

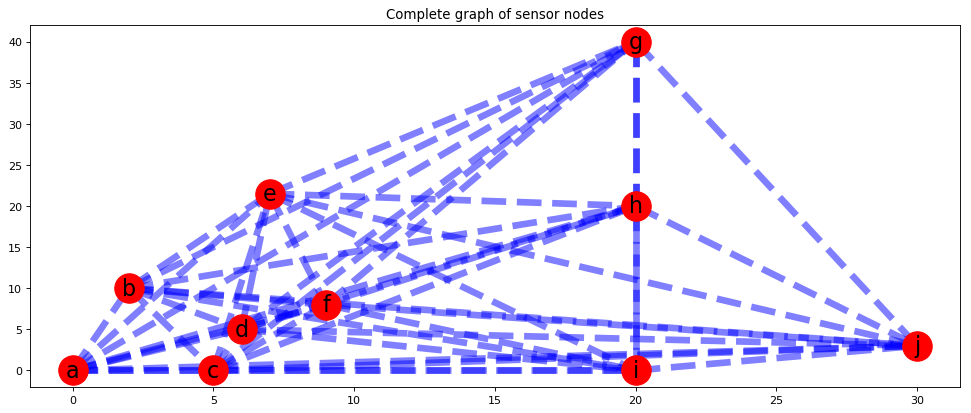


Fig:1 Complete Graph with 10 nodes

The nodes are represented in red with the alphabets indicating the name of the nodes. These are taken as inputs to our code through csv file submitted along with this project report. As it is clear from the image that the graph generated is a complete graph.

The next step is to generate Minimum Spanning Tree(MST) for the above graph. The implementation of MST from scratch is done in the code “Algorithm\_4\_BCRP\_MNCC”.

Following is the MST that our code gives:

MST for BCRP-MNCC Algorithm 4: defaultdict(<class 'set'>, {'a': {'d', 'c'}, 'd': {'b', 'f'}, 'c': {'i'}, 'f': {'e'}, 'i': {'j'}, 'e': {'h'}, 'h': {'g'}})

A function “” is also called which uses the function MST available in networkx package. This is to confirm if our code from scratch is giving the correct MST. The following MST was found for the above graph after calling this function:

Just\_To\_Confirm MST:

[('a', 'c', {'weight': 0}), ('a', 'd', {'weight': 1}), ('b', 'd', {'weight': 1}), ('c', 'i', {'weight': 2}), ('d', 'f', {'weight': 0}), ('e', 'f', {'weight': 2}), ('e', 'h', {'weight': 2}), ('g', 'h', {'weight': 3}), ('i', 'j', {'weight': 2})]

We found that our algorithm was giving the correct MST.

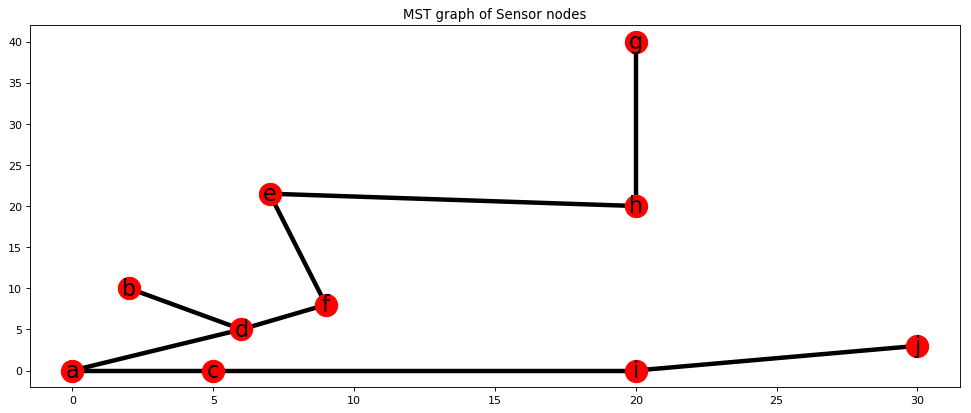


Fig:2 Minimum Spanning Tree of Graph with 10 nodes

The value of communication range and budget are taken from the user as input. For our analysis, we have taken communication range as 5 and Budget as 5.

The sum of weight of all edges in the above MST graph is 13.

The code is run and every time the edge with maximum weight was getting removed from the MST. It was running till the weight of MST becomes lower than the total weight.

Following are some images which we collected while running the code:

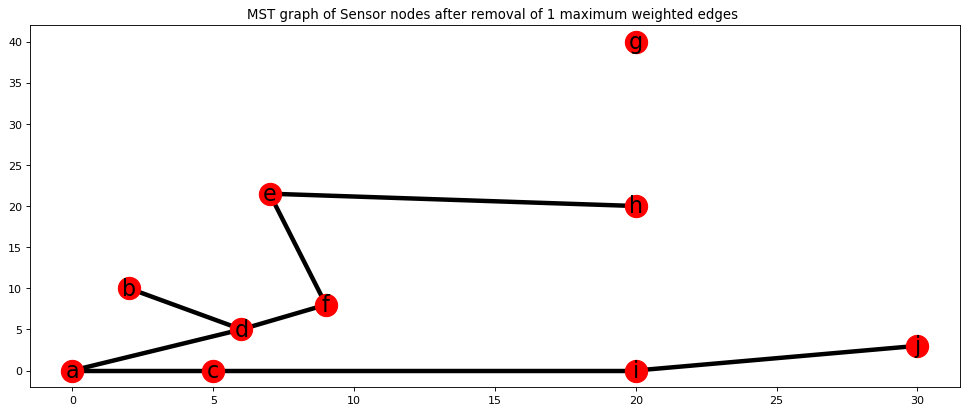


Fig:3 After removal of edge with maximum weight

The edge between g and h node was the highest i.e. 3. So, it got removed.

But the MST weight is still 10 which is more than the budget of 5. So, it is important keep removing edges.

Following is the image after removing the edge with highest weight i.e. c-i with weight ‘2’:

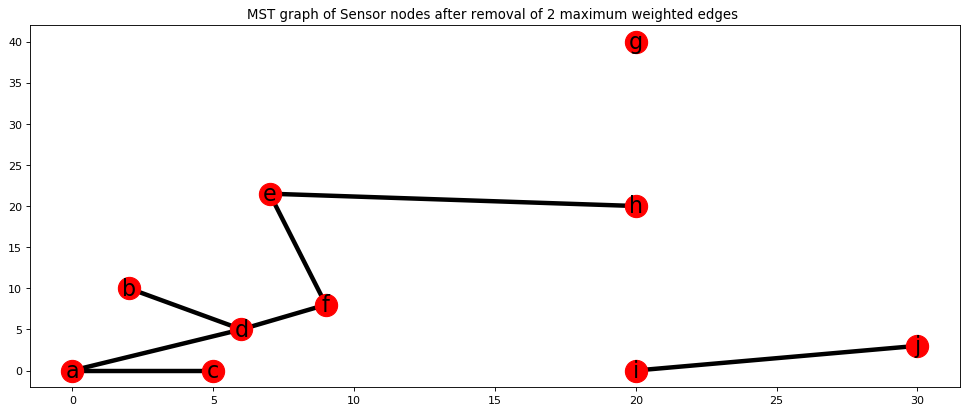


Fig:4 After removal of edge with a maximum weight

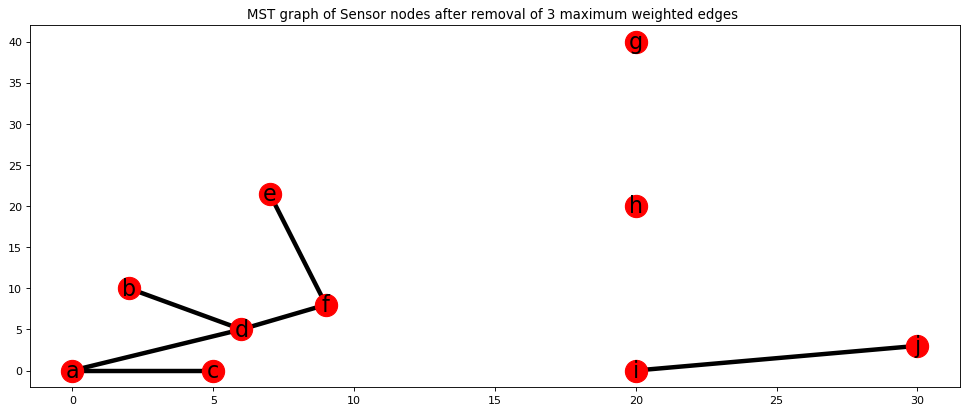


Fig:5 After removal of edge with maximum weight eh-2

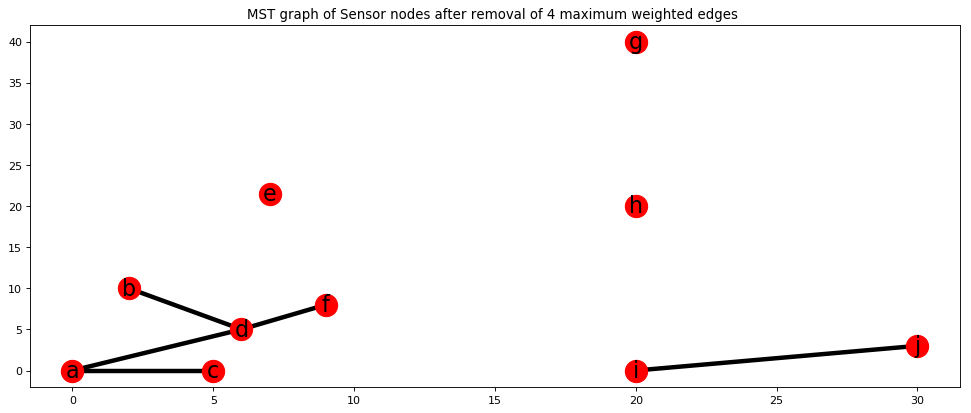


Fig:6 After removal of edge with maximum weight ef-2

Since the weight of the MST now is 4 which is below 5,the code stops and the number of connected components is 5. The above plot is the final resulting forest from our MST.

Connected components are: <(a,c,d,b,f),e,g,h,(i,j)>

**5.2. Analysis of BCRP-MLCC Algorithm (Algorithm 5)**

Taking an input for locations of sensor nodes from the csv file and making it a graph with edge weights to be no. of relay nodes needed to ensure connectivity between the nodes.

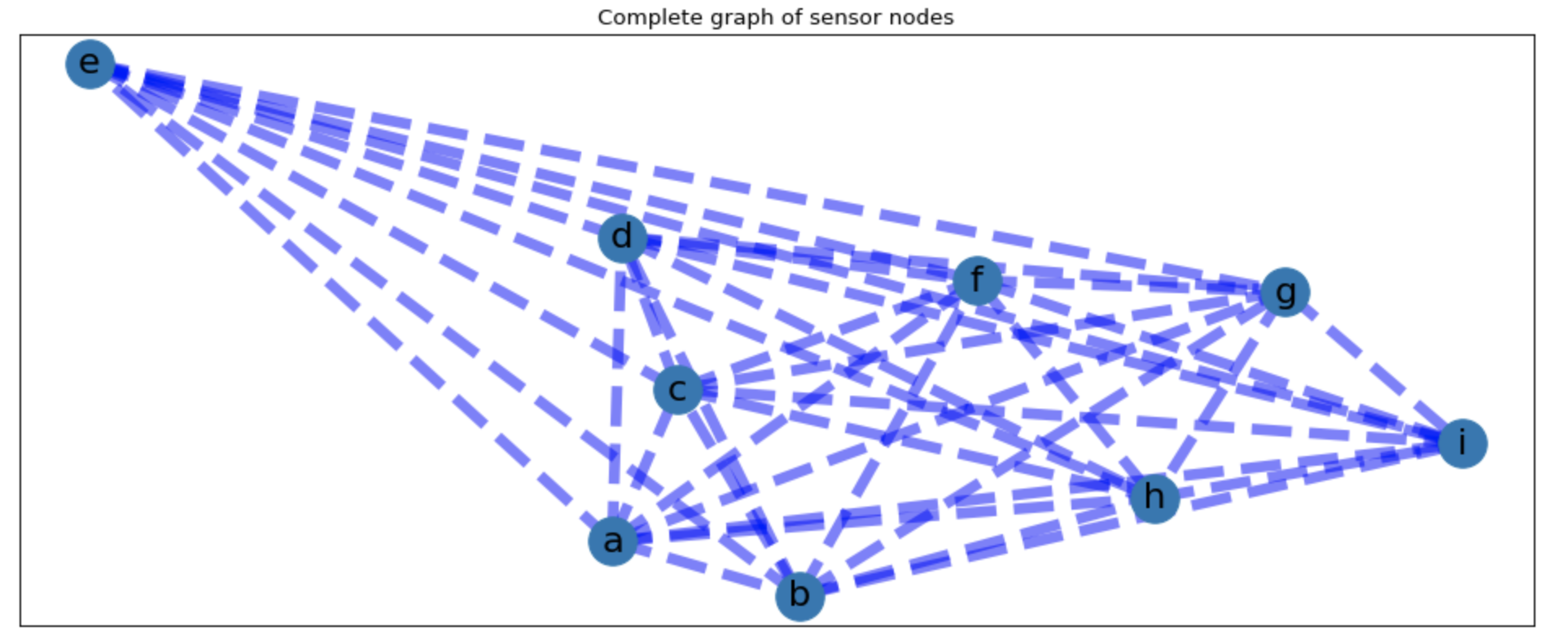
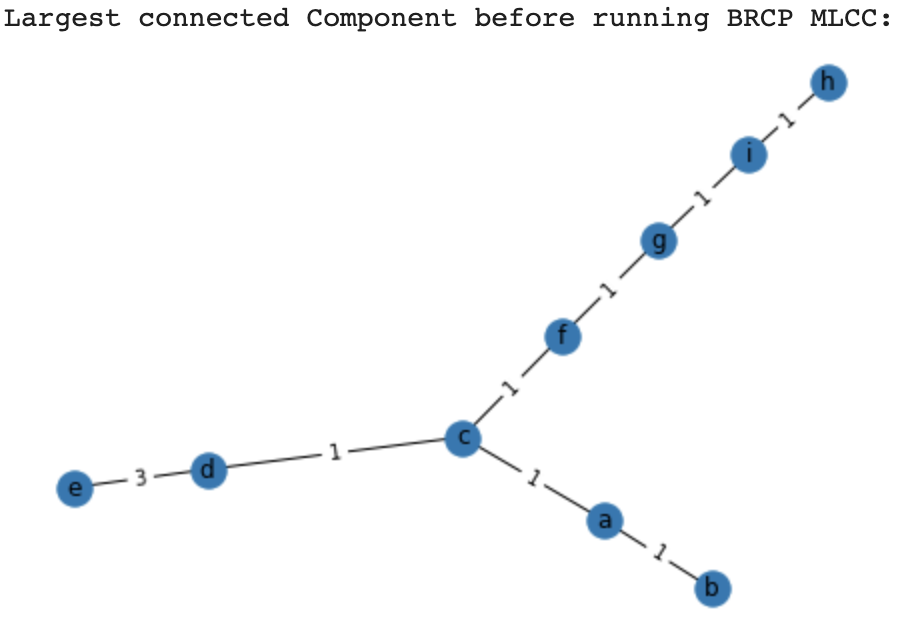
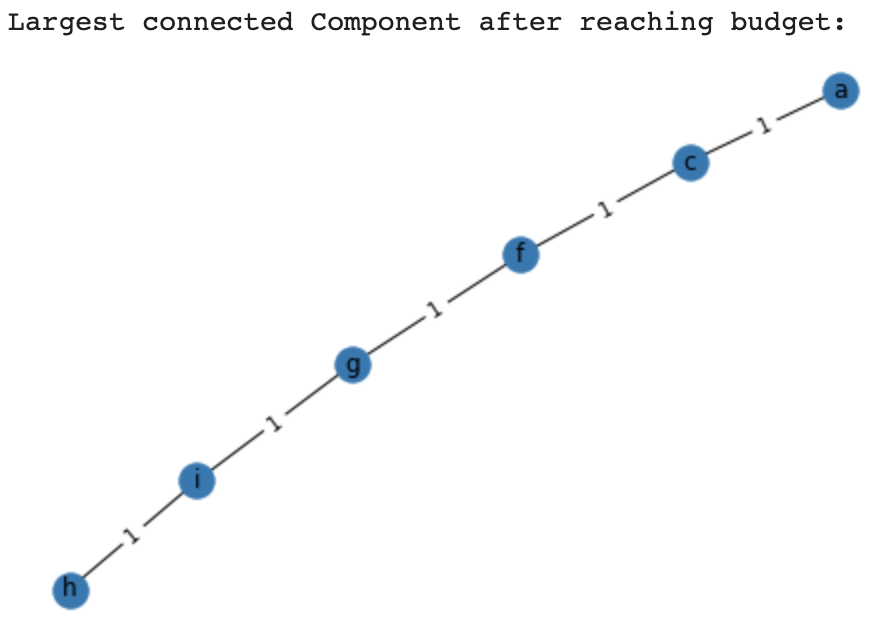


Fig:1 Complete Graph with 10 nodes

The nodes are represented in red with the alphabets indicating the name of the nodes. These are taken as inputs to our code through csv file submitted along with this project report. As it is clear from the image that the graph generated is a complete graph. We then go ahead and create the MST for the graph. Once an MST is created, we iteratively creating an n-1 MST iteratively and checking if the required weights are less than budget. If so, then it will return the resulting MST as the largest component.





**TEAM-MATES AND WORK-SPLITTING:**

Deepika and Shambhavi worked on the various aspects of algorithm 4(Heuristic based algorithm for BCRP-MNCC problem). They worked on Prim’s algorithm to create Minimum Spanning Tree. While Vinay and Chandu woked on Algorithm 5 and figured out the implementation of an approximate k MST from the n-mst. After this, they implemented the BCRP-MLCC and BCRP-MNCC algorithms. Then Shambhavi and Deepika tested the algorithms by creating multiple graphs using various software and the ones provided by the TA’s. All four of us tried to implement the algorithm efficiently.

**MY CONTRIBUTION:**

I worked along with Chandu K and tried figuring out how to implement a k-MST from a complete MST. Since these are approximate algorithms, we know we can create in multiple ways. We chose to iteratively implement from the k+1 MST. We then sat down and coded the k-MST and algorithm 5 together and tested on various scenarios.

**PEER EVALUATION:**

|  |  |  |  |
| --- | --- | --- | --- |
| **TEAM-MATES** | **Shambhavi K** | **Chandu Kamanuru** | **Deepika Patra** |
| **EVALUATION** | **20** | **20** | **20** |