OPTIMIZATION AND HEURISTIC FACILITY LOCATION ALGORITHMS

FOR THE SMART GRID

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Nikhitha Kaparthi

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

March 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

OPTIMIZATION AND HEURISTIC FACILITY LOCATION ALGORITHMS FOR THE SMART GRID

By

NIKHITHA KAPARTHI

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall Nygard
Chair

Dr. Vasant Ubhaya

Dr. Maria Alfonseca-Cubero

Approved:

03/11/2013
Date

Dr. Brian Slator
Department Chair

ABSTRACT

One of the most crucial aspects in maintaining a smart grid Network is to monitor its stability, control and minimize the outages, blackouts etc. which can be achieved by Phasor Measurement Units (PMUs). Despite having a number of advantages, PMUs are expensive and cannot be placed at every node in a network for constant observation. Therefore, it is economically feasible to place these PMUs at optimal locations, depending on the demand of each node.

Our current research attempts to resolve the issue of positioning PMUs in a dynamic network at optimal locations which is accomplished by using two different heuristic algorithms, K-Median and K-Center heuristic algorithms. A limited number of PMUs will be used such that all nodes are served efficiently. Further, we will compare the time taken for the optimization and heuristic procedures of both algorithms in a dynamic network.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my adviser, Dr. Kendall Nygard, for his continued support throughout this paper. I am grateful for the ideas and suggestions given by Dr. Nygard, and the amount of guidance he gave is enormous. Also, special thanks go to my advisory committee members, Dr. Vasant Ubhaya and Dr. Maria Alfonseca-Cubero, for their input which helped me complete this paper.

I thank all my graduate faculty members for sharing their knowledge and experience with me. I would like to thank Ms. Carole Huber, Ms. Stephanie Sculthorp, and Ms. Betty Opheim personally as well as all the computer science department staff members for their support. I would to like to convey my special thanks to my friends, Satheesh Chakravarthi, Keerthi Sathiraju, Manu Bhogadi, Sterling Volla, Barb Volla, Archana Rajpalem, Anjana Guruprasad, Aishwarya Kakarla, and Samuel Sudhakar Kondamarri, for their valuable suggestions and continued moral support.

I would like to recognize my colleagues, Nishad Mohite, Gaurav Singh, Kate Vanstone, and William Very, for their continued support, and also I take this opportunity to thank all my fellow students for their encouragement and support.

Finally, words alone cannot express the thanks I owe to my family members for their enormous support.

Once again, I would like to extend my sincere thanks to my adviser, Dr. Nygard, for his valuable suggestions and support that helped me to complete this paper successfully.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

1. INTRODUCTION

This chapter mainly focuses on a detailed description of the research problem, smart grid, Phasor Measurement Units (PMUs) and different types of facility location problems.

1.1. Problem Description

Many decision problems are largely concerned with selecting; choosing the number; and placing the facilities, such as manufacturing plants, libraries, fire stations, hospitals, etc. One such example is placing PMUs in a smart grid network. In this paper, we will resolve the issue of optimally placing the PMUs in a smart grid network.

1.2. Smart Grid Network

A smart grid network delivers electrical energy from suppliers to consumers with the help of digital technology in a duplex communication to manage appliances. The three main features of smart grid are reliability, efficiency and flexibility [2].

1.2.1. Characteristics

The main characteristics of a smart grid network are as follows [1]:

- It has the ability to self-heal during the events of a power disturbance.
- Consumers actively participate in demand response.
- Smart grids need to be secured, and they have the ability for self-recovery during the time of physical and cyber-attacks.
- Distributed generation and storage are well accommodated in the smart grid.
- In order to operate efficiently, assets are optimized. Smart grid also enables new markets, services and products.

- A smart grid is really more than meter reading. It helps the utility in managing and monitoring power delivery to homes or businesses. It also helps to handle outages very efficiently and effectively. Furthermore, it offers good demand management.

1.2.2. Components

The main components of a smart grid network are as follows:

- Consumers and generators of electrical energy
- Phasor Measurement Units (PMUs)
- Demand response
- Smart meters
- Advanced sensors
- Workforce application

Furthermore, visualization software, servers, and management used for running the grid are also significant components of a smart grid which requires a quick, duplex communication structure.

1.3. Phasor Measurement Units (PMUs)

PMUs on a smart grid network measure electrical waves with the help of a common time source for synchronization. PMUs are one of the crucial devices used for measuring electrical power waves, voltage phase angle and voltage magnitude in a smart grid network. They have a unique feature of measuring voltages in a synchronized fashion in a smart grid network, making the PMUs a critical measuring device [3].

Placing the PMUs in every node of the smart grid network helps observe the state of the network. However, it is a rare scenario to have PMUs on all nodes because having a

lot of PMUs in the network is very expensive. PMUs help measure the voltage on the other end of the edge using Ohm's Law [4]. PMUs measure at a rate of 30 observations per second.

1.3.1. Uses

PMUs are primarily used for the purposes mentioned below:

- PMUs provide the state estimation of the power system at exact and regular periodic intervals, even in a dynamic scenario, and also take immediate action if any issues occur in the network.
- The primary use of PMUs on a smart grid network is for measuring voltage phase angle and voltage magnitude.
- PMUs ensure that good-quality power is being sent to the consumers.
- PMUs are also used to analyze the burden of the network system regarding severe circumstances, i.e., congestion management.
- Even after a disturbance in the network, PMUs are used for processing the sequence of steps from the very beginning.
- Depending on the synchronized phasor measurements, a progressive action of protection is taken [5].

1.3.2. Applications

The primary applications of PMUs in a smart grid are as follows:

- PMUs are used in smart grid networks for monitoring and serving all network nodes.
- In order to manage the power system, PMUs are applied in load-control techniques.
- PMUs help avoid blackouts and outages as well as detect errors by increasing the network reliability.

- In wide area monitoring and control, PMUs play a major role in finding a station sensor's errors, maintaining the preciseness of time synchronization, and basic error updates [6].

1.4. Facility Location Problems

Two factors that we need to consider for placing a facility in a network are as follows [7]:

- Location
- Layout

Location is defined as the place where we locate a facility, whereas layout defines how we locate facilities inside a chosen location. Both location and layout decisions are equally important. Location decisions are strategic and qualitative in nature even though several other quantitative models are available for location. The decision about the location also depends on many qualitative factors, such as [7]

- Raw materials: For example, if we are planning to build a new manufacturing plant or facility which is raw-material intensive, then it is customary to place these facilities near the locations where these raw materials are available.
- Labor availability: The decision of having facilities also depends on the areas where labor is available in abundance and where labor charges are not very expensive to manage.
- Node demand: For example, retail stores, banks, etc. are all located very close to the demand area.

- Transportation decision: Locating facilities also depend on the transportation factor to move a large amount of material to and from the location.
- Policies: The government may give subsidies to locate particular industries in certain locations, depending on the availability and possible benefits for having a facility in that particular location.
- Reach ability: This factor plays an important role for a few facilities, for example, a fire station. In such a case, travel distance and the amount of time to cover the distance play a major role because we try to minimize the time and distance between locations where facility is placed and those of the consumer/customer.

The quantitative models that are available for location and layout decisions are as follows:

- K-Median Problem: Although this model was originally created for location and layout decisions, it is very useful in several other applications, such as optimally locating PMUs in a smart grid network. This problem can be resolved by minimizing the total average transportation distance of a node to the facility. Hence, this problem is used in non-emergency scenarios [7]. This problem is also known as the minisum location-allocation problem [8].
- K-Center Problem: In many cities, retail outlets need to deliver goods often, so we need to choose locations which are optimal for distribution centers which can reduce the cost of delivery to each outlet and minimize the number of distribution centers. This problem can be solved by minimizing the maximum distance between the distribution centers. Hence, this problem is mainly used in emergency scenarios. This problem is also known as the minimax location-allocation problem [8].

- Requirements Problem: This model is characterized by pre-specified standards which are defined for facilities such that all requirements are met. Because these standards are inconsistent, these problems can be used in both emergency and non-emergency scenarios [9].

1.5. Objectives

Due to cost limitations, we can only have PMUs on those nodes which have high demand and requirements for incomplete observability. Therefore, this research focuses on two main objectives:

- To write a heuristic for the two problems: K-Center and K-Median. Depending on the node demand, a limited number of PMUs can be placed at incumbent locations in a network. Furthermore, the nodes which are rarely observed can be conspicuous.
- To compare optimization and heuristic procedures based on time.

2. LITERATURE REVIEW

A large number of approaches exist to help resolve facility–location problems, such as non-deterministic polynomial-time (NP) hard problems which include the uncapacitated facility location problem (UFLP) and the quadratic assignment problem. These problems are also used for locating multiple facilities in the network [10].

The uncapacitated facility location problem is an important NP-hard problem that includes many algorithms, such as the p-approximation algorithm, integer programming algorithm, approximation algorithm, and max-product linear programming (MPLP) [11].

The other facility location problems are constant-factor LP rounding, primal-dual algorithms, and greedy local-search algorithms [12].

For an uncapacitated facility location problem, let us consider a network G (N, E), where N gives a set of nodes, i.e., {1, 2, ..., n}, and E gives a number of edges. Let the length of each edge be d_{ij}, where i,j ϵ N; for each location i ϵ I, we have a facility such that cost is associated with it, and for each j ϵ J, we have demand locations for the facility such that I,J $\subseteq$ N. Let the cost of each facility be c_i, where $c_i \geq 0$, and let the cost of accommodating each demand location, j, to the facility location, i, be c_{ij} per unit demand and let demand be d_j. In this case, we are assuming that costs are equal to or greater than 0; that they satisfy the symmetry condition, i.e., $c_{ij} = c_{ji}$; and also that they satisfy the triangle inequality, i.e., $c_{ij} + c_{jk} \geq c_{ik}$, where i, j, k ϵ N. According to UFLP, we need to assign demand nodes to facility nodes such that the total cost incurred is minimized; i.e. [12],

$$\text{Minimize} \quad \sum_{i \epsilon I} c_i y_i + \sum_{i \epsilon I} \sum_{j \epsilon J} d_j d_{ij} x_{ij} \tag{1}$$

Subject to

$$\sum_{i \epsilon I} x_{ij} = 1, \forall j \epsilon J \tag{2}$$

$$x_{ij} \leq y_i \text{ , } \forall \text{i} \in \text{I, j} \in \text{J} \quad (3)$$

$$x_{ij} \in \{0, 1\}, \forall \text{i} \in \text{I, j} \in \text{J} \quad (4)$$

$$y_i \in \{0, 1\}, \forall j \in J \quad (5)$$

Constraints (2) and (3) make sure that every location, j ϵ J, is connected to some, or at least one, location in i ϵ I. Also, i ϵ I is always open when j ϵ J wants to get connected to it [12].

We can obtain linear programming (LP) relaxation for this problem by making x_{ij} and y_i zero or positive real numbers. Let α_j be the contribution of node j to connect to node i which has the facility to connect. The above constraints can be modified, and the LP program's dual program is as follows [13]:

$$\text{Maximize} \quad \sum_{j \in J} \alpha_j \quad (6)$$

Subject to

$$\alpha_j - \beta_{ij} - d_{ij} \leq 0 \ \forall \text{i} \in \text{I, j} \in \text{J} \quad (7)$$

$$\sum_{j \in J} \beta_{ij} \leq \text{c}_\text{i} \ \forall \text{i} \in \text{I} \quad (8)$$

$$\alpha_j \geq 0 \ \forall \text{ j} \in \text{J} \quad (9)$$

$$\beta_{ij} \geq 0 \ \forall \text{i} \in \text{I, j} \in \text{J} \quad (10)$$

With the dual program's inequality, we have

$$\sum_{j \in J} Max(0, \ a_j - d_{ij}) \leq c_i \quad (11)$$

For equation (11), if the dual program's inequality holds equality, the total cost, c_i, for establishing a facility would be enough [13]. For a better understanding, a new formula approach is defined, i.e., a star with a facility and several demand nodes. The cost of that star is the sum of the cost to establish a facility and the cost of connecting the demand

nodes to the facility. Now, let us consider a huge set of stars in set S such that every demand node exists in at least one star. An integer program can be defined such that [14]

$$\text{Minimize} \quad \sum_{S \in S} c_s x_s \tag{12}$$

Subject to

$$\sum_{S: j \in S} x_s \geq 1 \; \forall \; \text{j} \in \text{C} \tag{13}$$

$$x_s \in \{0, 1\} \; \forall \; \text{S} \in \text{S} \tag{14}$$

For the same problem, we can define an LP-relaxation as follows [14]:

$$\text{Minimize} \quad \sum_{S \in S} c_s x_s \tag{15}$$

Subject to

$$\sum_{S: j \in S} x_s \geq 1 \; \forall \; \text{j} \in \text{C} \tag{16}$$

$$x_s \geq 0 \; \forall \; \text{S} \in \text{S} \tag{17}$$

For the same problem, the dual program is defined as follows [14]:

$$\text{Maximize} \quad \sum_{j \in C} \alpha_j \tag{18}$$

Subject to

$$\sum_{j \in s \cap C} \alpha_j \leq c_s \; \forall \; \text{S} \in \text{S} \tag{19}$$

$$\alpha_j \geq 0 \; \forall \; \text{j} \in \text{C} \tag{20}$$

In this case, x_s makes sure star S is taken into consideration, and c_s defines the cost of the star. α_j defines the amount of city j included in the total actual expenses.

Let T be the cost for finding a solution for the facility location problem such that [14]:

$$\sum_{j \in J} \alpha_j = \text{T} \tag{21}$$

For $\gamma \geq 1$ and $\forall$S, we have

$$\sum_{j \epsilon J \cap C} \alpha_j \leq \gamma \, c_s \quad (22)$$

Then, it has, at most, γ as its approximation ratio [14].

The extension of the above algorithms gives us the definition of a greedy algorithm.

For the same star S, where the cost is c_s and where there are clients a_1, a_2, etc. until a_k and facility p, the greedy algorithm is defined as follows [15]:

$$\sum_{i=1}^{k} c_{p,a_i} \; + \; f_p(k) \quad (23)$$

The above definition says, sum of the costs of the connection of the clients to the facility p added to the facility cost of p for k clients [15].

There are also several placement models that consider the reliability of PMUs. These placements models are defined in the two-stage optimization model where the first stage is integer linear programming which is used to get the optimal placement solution and compensates the reliability and observability constraints. In the later stage, we maximize the reliability of the entire system [16].

The initial stage of the two-stage optimization model also considers cost optimization by minimizing the number of PMUs. If R is the system-level reliability and r is the minimum reliability required for the bus system in order to reach the level of system reliability, then [16]

$$\mathrm{r} = \sqrt[n]{R} \quad (24)$$

$$\text{Minimize} \;\; \sum_{i=1}^{n} x_i \quad (25)$$

Subject to

$$f_i \geq \log(1\text{-r}) \,/\, \log \mathrm{q} \;\; \forall \mathrm{i} \quad (26)$$

$$r^n \geq \mathrm{R} \quad (27)$$

In the later stage of the two-stage optimization model, we try to maximize the reliability of the total system [16]:

$$\text{Maximize} \quad R \tag{28}$$

Subject to

$$f_i \geq \text{b} \ \forall \text{i} \tag{29}$$

$$\sum_{i=1}^{n} x_i \leq \text{c} \tag{30}$$

where c is the number of PMUs from the minimization problem.

3. K-MEDIAN AND K-CENTER PROBLEM

This chapter provides a detailed description of the K-Median Optimization and Heuristic procedures and the K-Center Optimization and Heuristic procedures. This chapter also includes examples, flowcharts, algorithms, and pseudo code.

3.1. K-Median Problem

For an undirected network G (N, E) where N = number of nodes and E = number of edges, the k number of PMUs to be placed in network G (N, E) in optimal locations requires minimizing the total average distance [9]. Let the k points be a set,

$X_K = \{x_1, x_2, x_3, \ldots, x_{k-1}, x_k\}$, and the shortest distance for any one of nodes, $x_1 \in X_K$ and node l in G, can be represented as $d(X_K, l)$

$$d(X_K, l) = Min_{x_i \in x_k} d(x_i, l) \tag{31}$$

K-Medians on network G can be defined as follows [9]:

$$X_K^{*} \text{ on G is a set of k-PMUs on G for every } X_K \in G \tag{32}$$

$$J(X_K) = \sum_{l=0}^{n} h_l \mathrm{d}\,(X_K, l)\,| \tag{33}$$

$$J(X_K^{*}) \leq J(X_K) \tag{34}$$

That is, the PMUs are placed such that the total average distance for the K-Median problem is minimized.

3.1.1. K-Median Optimization Problem

This section describes the algorithm, flowchart, and example of the K-Median Optimization Problem.

3.1.1.1. Algorithm and Flowchart for the Multi-Median Optimization Problem

The K-Median Optimization algorithm is an optimization problem where, for n nodes and k optimal locations, we calculate the total average distance of all possible nC_k combinations.

The steps of computation are as follows:

1. The user is allowed to input the graph in the form of an .xml file and also chooses the number of optimal nodes to locate.
2. The program generates nC_k combinations of n nodes.
3. For each combination, total average travel distances are calculated.
4. Among those total average distances, the minimum value needs to be picked [9].
5. All combinations with the minimum value as the total travel distance are optimal combinations to locate the facilities.

Figure 1 presents the pseudo code of the Multi-Median Optimization Problem, providing detailed computational steps for the algorithm.

```
Input graph 'G' with 'n' nodes
Input k
Initialize m <- 1
TotalDemand <- Sum of demands of all 'n' nodes
Generate nCk combinations with 'n' nodes
While m <= nCk
TotalSumM = Sum of the 'shortest distances' between every node of mth combination with
every other node in 'n'.
TotalAverageDistanceOfM = TotalSumM / TotalDemand
AverageDistanceArray[m-1] = TotalAverageDistanceOfM
Increment m by 1
EndWhile
OptimalSolution = Minimum of all the elements in the AverageDistanceArray (all nCk
combinations)
OptimalCombination = All nCk combinations with OptimalSolution as their Total Average
Distance
```

Figure 1. Pseudo Code of the Multi-Median Optimization Problem.

Figure 2 presents the flowchart of the Multi-Median Optimization Problem with detailed steps of computation right from the point of user input to the point of obtaining the optimal solution.

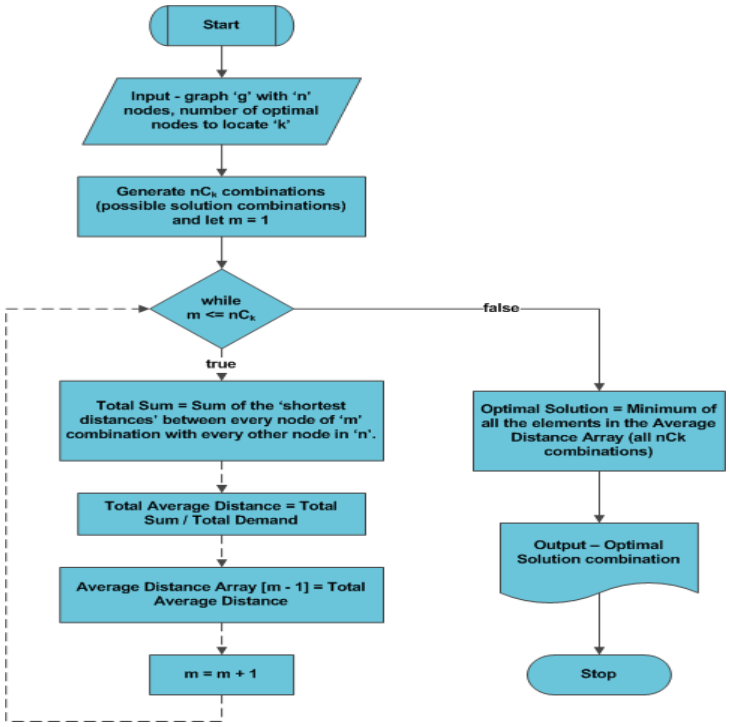

Figure 2. Flowchart of the Multi-Median Optimization Problem.

3.1.1.2. Example for the Multi-Median Optimization Problem

Let us consider the following network in Figure 3:

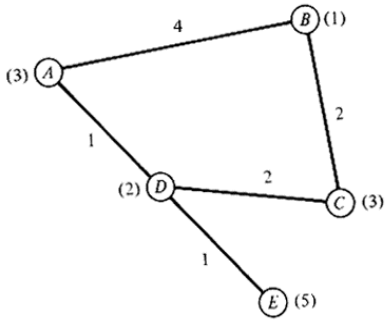

Figure 3. Example Network for the Multi-Median Optimization Problem.

Step 1: We need to construct the shortest distance matrix for this network.

$$d_{ij} = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 0 & 4 & 3 & 1 & 2 \\ B & 4 & 0 & 2 & 4 & 5 \\ C & 3 & 2 & 0 & 2 & 3 \\ D & 1 & 4 & 2 & 0 & 1 \\ E & 2 & 5 & 3 & 1 & 0 \end{array}$$

Step 2: We need to construct the node weight matrix with the help of the shortest matrix in Step 1, i.e., demand of the node multiplied by distance between the demand node and the facility node

$$h_j * d_{ij} = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 0 & 4 & 9 & 2 & 10 \\ B & 12 & 0 & 6 & 8 & 25 \\ C & 9 & 2 & 0 & 4 & 15 \\ D & 3 & 4 & 6 & 0 & 5 \\ E & 6 & 5 & 9 & 2 & 0 \end{array}$$

Step 3: For a single facility location, we need to calculate the total average distance of each node to every other node, i.e.,

$$Total\ average\ distance = \left\{ \frac{Total\ distance}{Sum\ of\ all\ node\ demand} \right.$$

	A	B	C	D	E	$Total\ average\ distance$
A	0	4	9	2	10	$\frac{25}{14} = 1.78$
B	12	0	6	8	25	$\frac{51}{14} = 3.64$
C	9	2	0	4	15	$\frac{30}{14} = 2.14$
D	3	4	6	0	5	$\frac{18}{14} = 1.28$
E	6	5	9	2	0	$\frac{22}{14} = 1.57$

Because 1.28 is the minimum total average distance of all other nodes, 1.28 is the optimal total average distance, and D is the optimal location.

For the multiple facility location problem, say for the same network, we need to place 3 locations such that n = 5 and k = 3. We need to check total average distances of $5C_3$ = 10 combinations, i.e., nC_k combinations. Let us consider one of the ten combinations, i.e., A, B, D:

$$h_j * d_{ij} = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 0 & 4 & 9 & 2 & 10 \\ B & 12 & 0 & 6 & 8 & 25 \\ C & 9 & 2 & 0 & 4 & 15 \\ D & 3 & 4 & 6 & 0 & 5 \\ E & 6 & 5 & 9 & 2 & 0 \end{array}$$

In this case, the shortest distance for each node is the distance of the node to the nearest facility. The total average distance for case A, B, D is (0 + 0 + 6 + 0 + 5) / 14 = 11/14 = 0.78.

Similarly, find the total average distance for all 10 combinations, such as "A, B, C"; "B, C, D"; "A, B, E"; etc. Using this exhaustive process, we obtain the total average distance of all combinations from which the minimum value is our optimal average distance, and the respective combination is optimal locations for the facilities.

Therefore, as the network becomes larger, it requires a lot of computational time to generate all combinations, such as a 100-node network with 50 facilities. In such a scenario, we end up calculating $100C_{50}$ combinations. To circumvent this problem, our current research explains the heuristic algorithm for the multi-median problem.

3.1.2. K-Median Heuristic

This section describes the algorithm, flowchart, and example of the K-Median Heuristic Procedure.

3.1.2.1. Algorithm and Flowchart for the Multi-Median Heuristic

The Multi-Median Heuristic algorithm is a heuristic procedure where, for n nodes and k optimal locations, we calculate the total average distance of only a few combinations. The steps of computation are as follows:

1. The user is allowed to input the graph in the form of an .xml file and also chooses the number of optimal nodes to locate.
2. The one-median solution needs to be calculated.

   The current procedure is iterative, where 2 optimal locations are found from 1 optimal location, 3 optimal locations from 2 optimal locations, 4 optimal locations from 3 optimal locations, etc., until k optimal locations are found from k-1 optimal locations.
3. Once the single optimal location is obtained, the location is moved to set S. Say, m = 1 where 'm' defines number of current optimal locations.
4. A new facility is added from the remaining nodes set S, resulting in a list. The total travel distance of each combination in the list needs to be computed. Let the minimum value for the total travel distance of all combinations be the current minimum distance and the combination that gives the minimum total travel distance be the current optimal combination.
5. Now, every node of the current optimal combination needs to be replaced one by one with the other nodes that are not present in the current optimal combination and compute the total travel distance of those combinations. All the total travel distances need to be compared with the current minimum distance.
6. If any combinations result in a total travel distance less than that of the current minimum distance, it is replaced with the new total travel distance, and the current

optimal combination is replaced with the new combination. Repeat steps 5 through 6; otherwise, the current combination is the optimal solution. m is incremented such that m = m +1.

7. Once an incumbent combination is obtained, repeat steps 4 through 7 until m = k [9].

```
Input graph 'G' with 'N' nodes
Input k
Generate 1−Median solution and move it to set S
Initialize m <− 1
While m <= k
Add a new facility to the set S from 'N' nodes which are not already present in S and move
the combinations to S` such that set S` = S x (N−S)
Calculate the total average distance of each combination from set S` and get the minimum
value (cmin) of total average distances and the sets with cmin as total average distance can
be moved to set S`` such that the set S`` = S U {xm} where xm ϵ (N−S)
Improve the solution set by replacing elements of S`` systematically one by one with the
nodes of (N−S``)
If cmin >= set of average distances
        cmin = new improved minimum average distance
        S`` = Improved set with minimum average distance as cmin
Call "Improve the solution set by replacing elements of S`` systematically one by one with the
nodes of (N−S``)"
Else
        If all nodes are not substituted
Call "Improve the solution set by replacing elements of S`` systematically one by one with the
nodes of (N−S``)"
        Else
                m = m + 1
                S = S``
                EndIf
        EndIf
End While
Optimal solution combination is set S
```

Figure 4. Pseudo Code of the Multi-Median Heuristic Procedure.

Figure 4 presents the pseudo code of the Multi-Median Heuristic Procedure, providing the detailed computational steps of the algorithm. Figure 5 presents the flowchart for the Multi-Median Heuristic Procedure, providing the detailed steps of computation from the point of user input to the point of obtaining the final solution.

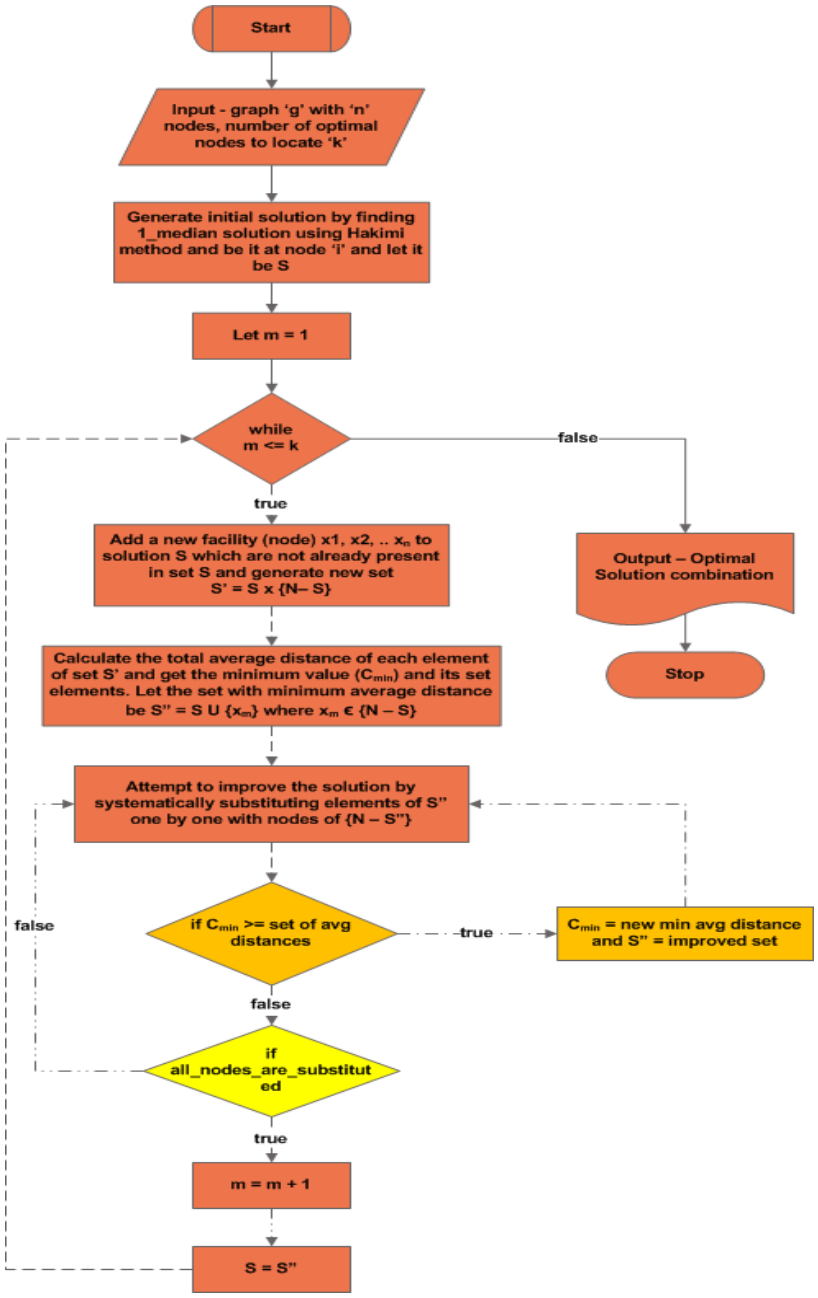

Figure 5. Flowchart of the Multi-Median Heuristic Procedure.

3.1.2.2. Example for the Multi-Median Heuristic

Suppose we are calculating for the two incumbent locations by taking the following network in Figure 6 into consideration:

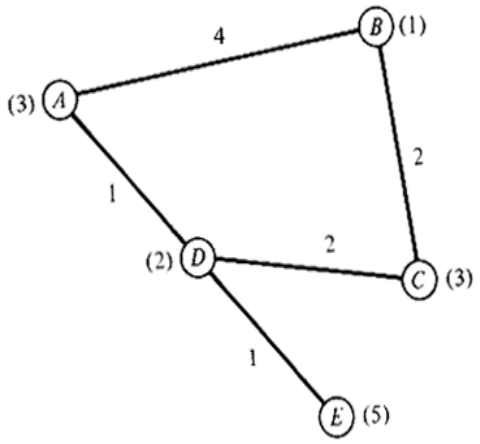

Figure 6. Example Network for the Multi-Median Heuristic Procedure.

For any value of k, the single optimal facility location is the first step of computation for the Multi-Median heuristic procedure.

Step 1: We need to construct the shortest distance matrix for this network

$$d_{ij} = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 0 & 4 & 3 & 1 & 2 \\ B & 4 & 0 & 2 & 4 & 5 \\ C & 3 & 2 & 0 & 2 & 3 \\ D & 1 & 4 & 2 & 0 & 1 \\ E & 2 & 5 & 3 & 1 & 0 \end{array}$$

Step 2: We need to construct the node weight matrix with the help of the shortest matrix in Step 1 i.e. demand of the node * distance between the demand node and the facility node

$$h_j * d_{ij} = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 0 & 4 & 9 & 2 & 10 \\ B & 12 & 0 & 6 & 8 & 25 \\ C & 9 & 2 & 0 & 4 & 15 \\ D & 3 & 4 & 6 & 0 & 5 \\ E & 6 & 5 & 9 & 2 & 0 \end{array}$$

Step 3: For a single facility location, we need to calculate the total average distance of each node to every other node; i.e.,

$$Total\ average\ distance = \left\{\frac{Total\ distance}{Sum\ of\ all\ node\ demand}\right.$$

	A	B	C	D	E	$Total\ average\ distance$
A	0	4	9	2	10	$\frac{25}{14} = 1.78$
B	12	0	6	8	25	$\frac{51}{14} = 3.64$
C	9	2	0	4	15	$\frac{30}{14} = 2.14$
D	3	4	6	0	5	$\frac{18}{14} = 1.28$
E	6	5	9	2	0	$\frac{22}{14} = 1.57$

Because 1.28 is the minimum total average distance of all other nodes, 1.28 is the optimal total average distance, and D is the optimal location. Move the current solution to set S. Therefore, S = {D}.

Step 4: This step is a facility addition step where we generate complementary combinations for set S such that {D, A}, {D, B}, {D, C}, {D, E}, and their respective travel distances are 15, 14, 10, and 13. So far, we have the minimum travel distance value as 10 with the node combination {D, C}. Now, S = {C, D} is our incumbent solution but not our final solution.

Step 5: This solution – improvement step is where we replace each node of set S and try to get a better solution by generating candidate combinations such as {C, A}, {C, B}, {C, E}, {A, D}, {B, D}, and {E, D}. We already calculated travel distances for the last three combinations, and we may avoid calculating those combinations again. The travel distances for candidate combinations {C, A}, {C, B}, and {C, E} are 14, 28, and 10. Now, we need to check if we have any travel distances less than or equal to 10. If yes, we need to repeat for the new incumbent solution until we get a value less than 10.

Step 6: If we get a travel distance less than 10, say "w," then "w" is our new total travel distance, and the respective combination is our final set; otherwise, 10 is our final incumbent solution, and all the combinations where the maximum travel distances to the facilities are 2 are considered incumbent solutions for the heuristic. Many times, this solution happens to be our final, optimum solution.

3.1.3. Visual Representation for the Multi-Median Problem

Let us consider the IEEE Standard 14-Bus System, a test system used for research and educational purposes, with 14 nodes and 19 edges [19]. Let us take an example of placing 7 PMUs at different locations.

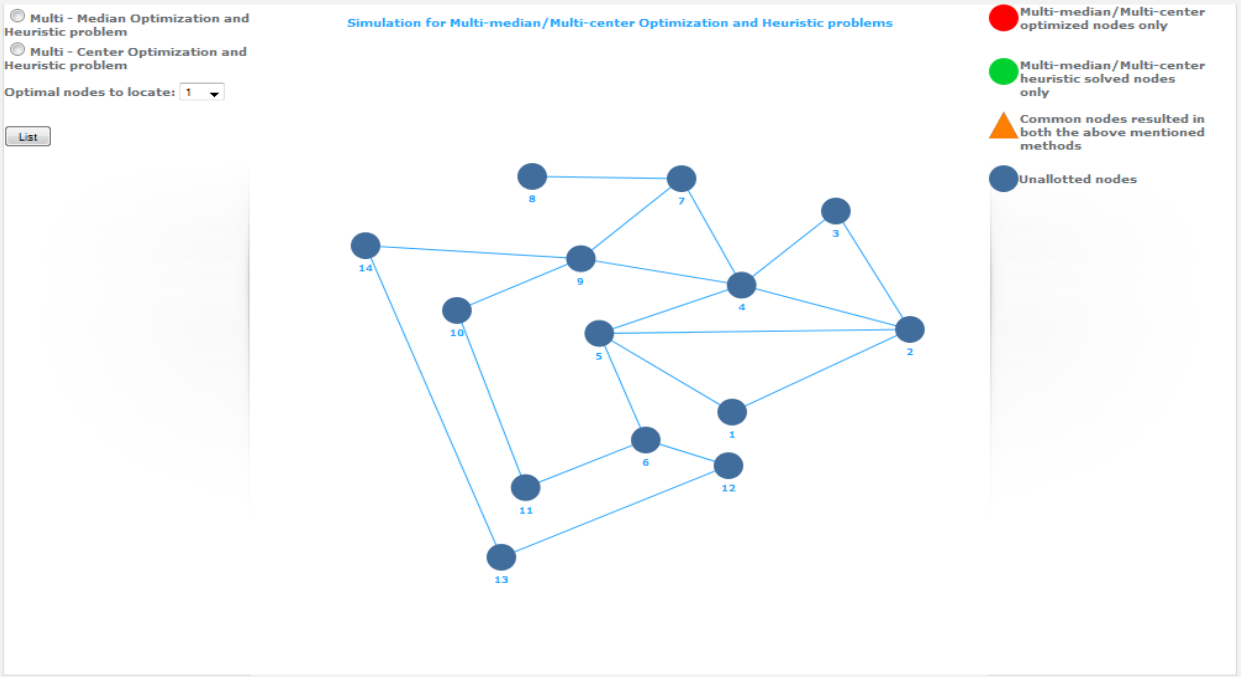

Figure 7. IEEE 14-Bus System for the Multi-Median Problem.

Figure 7 illustrates the view of the IEEE 14-Bus System. The left navigation of the page allows the user to select the number of optimal nodes to locate for PMUs. When the list is clicked, the nodes are highlighted, depending on the selected problem method.

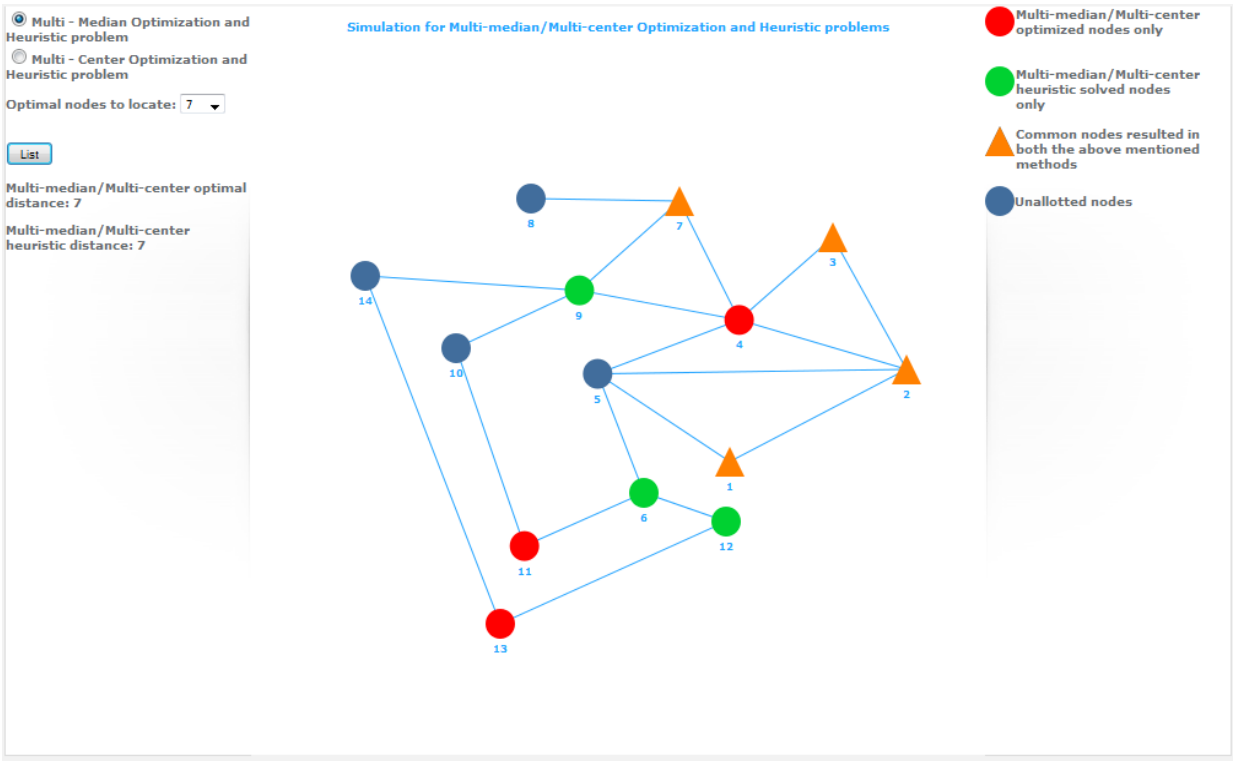

Figure 8. Visual Example of the Multi-Median Optimization and Heuristic Results.

Figure 8 illustrates different nodes located at optimal positions for the optimization problem and incumbent locations for the heuristic procedure. In our example with seven optimal locations, green circle nodes represent only Multi-Median optimization; red circle nodes represent only the Multi-Median heuristic; and yellow triangles represent the common nodes. Hence, the optimal solution of the IEEE Standard 14-Bus System for locating seven nodes is 1, 2, 3, 4, 7, 11, and 13, and the incumbent solution for locating seven nodes using the Multi-Median heuristic is 1, 2, 3, 6, 7, 9, and 12.

3.2. K-Center Problem

For an undirected network G (N, E), where N = number of nodes and E = number of edges, ‘k’ PMUs need to be placed in the network in optimal locations such that

minimizing the maximum distance of every node to every other node [9]. Let the k points be a set X_K, where

$X_K = \{x_1, x_2, x_3, \ldots, x_{k-1}, x_k\}$, and the shortest distance of any one node, $x_1 \in X_K$, and node l in G can be represented as $d(X_K, l)$

$$d(X_K, l) = Min_{x_i \in x_k} d(x_i, l) \quad (35)$$

K-Centers on network G can be defined as follows [9]:

X_K^* on G is a set of k PMUs on G; for every $X_K \in G$ (36)

$$J(X_K) = Max_{l \in N}\, d(X_K, l) \mid \quad (37)$$

$$J(X_K^*) \leq J(X_K) \quad (38)$$

That is, the PMUs are placed to minimize the maximum distance for the K-Center Problem.

3.2.1. K-Center Optimization Problem

This section describes the algorithm, flowchart, and example of the K-Center Optimization Problem.

3.2.1.1. Algorithm and Flowchart for the Multi-Center Optimization Problem

The Multi-Center Problem is an optimization problem where, for n nodes and k optimal locations, we calculate the maximum for the shortest distances of all possible nC_k combinations.

The steps of computation are as follows:

1. The user is allowed to input the graph in the form of .xml file and chooses the number of optimal nodes to locate.
2. The program generates nC_k combinations of n nodes.

3. For each combination, the maximum for the shortest distance of facilities of the combination to the nodes is calculated.
4. Among those maximum values, the minimum value needs to be picked.
5. All combinations with the minimum value as their maximum of shortest distances to every other node are the optimal combinations to locate the facilities [9].

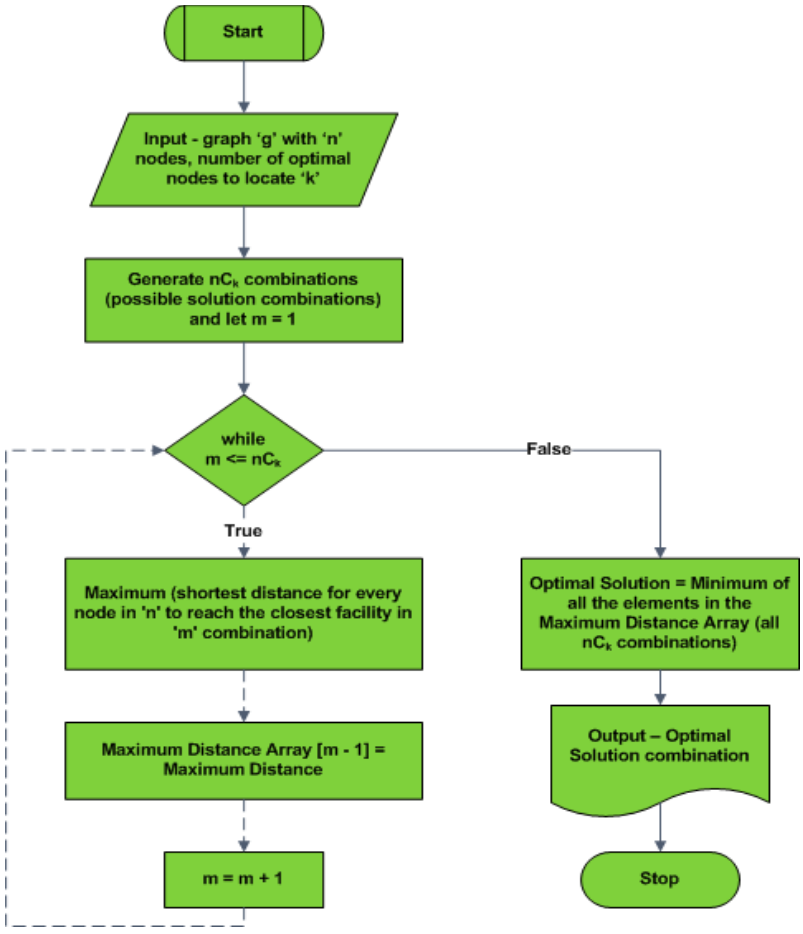

Figure 9. Flowchart of the Multi-Center Optimization Problem.

Figure 9 presents the flowchart for the Multi-Center Optimization Problem, providing the detailed steps of computation from the point of user input to the point of obtaining the final solution. Figure 10 presents the pseudo code of the Multi-Center Optimization Problem, providing the detailed computational steps for the algorithm.

```
Input graph ‘G’ with ‘n’ nodes
Input k
Initialize m <– 1
Generate nCk combinations with ‘n’ nodes
While m <= nCk
MaximumDistanceM = Maximum (Shortest distance for every node in 'N' to reach the closest facility in 'm' combination) MaximumDistanceArray [m–1] = MaximumDistanceM
Increment m by 1
EndWhile
OptimalSolution = Minimum of all the elements in the MaximumDistanceArray (all nCk combinations)
OptimalCombination = All nCk combinations with OptimalSolution as their Maximum Distance
```

Figure 10. Pseudo Code of the Multi-Center Optimization Problem.

3.2.1.2. Example for the Multi-Center Optimization Problem

Let us consider the following network in Figure 11:

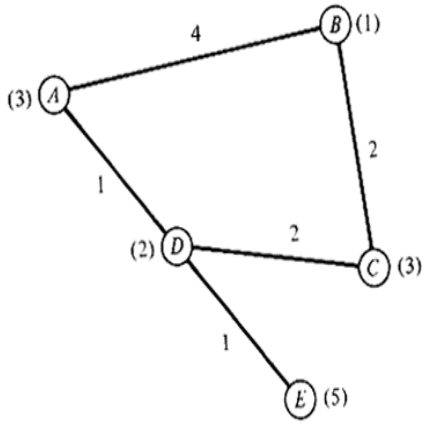

Figure 11. Example Network for the Multi-Center Optimization Problem.

Step 1: We need to construct the shortest distance matrix for this network.

$$d_{ij} = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 0 & 4 & 3 & 1 & 2 \\ B & 4 & 0 & 2 & 4 & 5 \\ C & 3 & 2 & 0 & 2 & 3 \\ D & 1 & 4 & 2 & 0 & 1 \\ E & 2 & 5 & 3 & 1 & 0 \end{array}$$

Step 2: For a single facility location, we need to calculate the maximum travel distance of each node to the facility; i.e.,

$$Maximum\ distance\ of\ each\ node =$$

$$\{Maximum\ (all\ shortest\ distance\ of\ each\ node\ to\ the\ facilities)$$

	A	B	C	D	E	$Max\ distance\ of\ each\ node$
A	0	4	3	1	2	$\text{Max}\ (0, 4, 3, 1, 2) = 4$
B	4	0	2	4	5	$\text{Max}\ (4, 0, 2, 4, 5) = 5$
C	3	2	0	2	3	$\text{Max}\ (3, 2, 0, 2, 3) = 3$
D	1	4	2	0	1	$\text{Max}\ (1, 4, 2, 0, 1) = 4$
E	2	5	3	1	0	$\text{Max}\ (2, 5, 3, 1, 0) = 5$

Because 3 is the minimum of the maximum distances of each node to the facility, 3 is the optimal travel distance, and C is the optimal location.

For the multiple facility location problem, say for the same network, we need to place three locations such that n = 5 and k = 3; we need to check the total average distances of $5C_3 = 10$ combinations, i.e., nC_k combinations. Let us consider one of the ten combinations, i.e., A, B, D:

	A	B	C	D	E
A	0	4	3	1	2
B	4	0	2	4	5
C	3	2	0	2	3
D	1	4	2	0	1
E	2	5	3	1	0

In this case, the shortest distance of each node is the distance of the node to the nearest facility. Therefore, the maximum total distances to the nearest facilities, A, B, and D, are Maximum (0, 0, 2, 0, 1) = 2.

Similarly, find the maximum total travel distances of all 10 other combinations, such as “A, B, C”; “B, C, D”; “A, B, E”; etc. Using this exhaustive process, we obtain the maximum total travel distances of all combinations from which the minimum value is our

optimal total travel distance and where the respective combination is optimal locations for the facilities.

Therefore, as the network becomes larger, it requires a lot of computational time to generate all combinations, such as a 100-node network with 50 facilities. In such a scenario, we end up calculating $100C_{50}$ combinations. To circumvent this problem, our current research explains the heuristic algorithm for the Multi-Center problem.

3.2.2. K-Center Heuristic

This section describes the algorithm, flowchart, and an example of the K-Center Heuristic Procedure. The K-Center Heuristic was developed following an approach that is inspired by the methodology used for the K-Median Heuristic. The procedure is a new advance that goes beyond the known heuristics for solving the K-Center problem.

3.2.2.1. Algorithm and Flowchart for the Multi-Center Heuristic

The Multi-Center Heuristic algorithm is a heuristic procedure where, for n nodes and k optimal locations, we calculate the maximum of the shortest distances for facilities in a combination, to the other nodes. Only a few combinations are calculated.

The steps of computation are as follows:

1. The user is allowed to input the graph in the form of an .xml file and chooses the number of optimal nodes to locate.
2. A one-center solution needs to be calculated.

   The current procedure is an iterative procedure where 2 optimal locations are found from 1 optimal location, 3 optimal locations from 2 optimal locations, 4 optimal locations from 3 optimal locations, etc. until k optimal locations are found from k-1 optimal locations.

3. Once the single optimal location is obtained, the location is moved to set S. Say, m = 1, where ‘m’ defines number of current optimal locations.
4. A new facility is added from the remaining nodes to the set S, resulting in a list. The maximum shortest distance of all other nodes to reach to the PMU of each combination in the list needs to be computed. Let the minimum distance value among all the maximum combination distances be the current minimum distance, and let the combination that gives the minimum distance value among all the maximum combination distances are the current optimal combination.
5. Now, every node of the current optimal combination needs to be replaced one by one with the other nodes that are not present in the current optimal combination, and we compute the maximum distance of each combination. All the maximum values need to be compared with the current minimum distance.
6. If any combination results in getting a maximum distance value less than the current minimum distance, the current minimum distance is replaced with the new maximum distance value, and the current optimal combination is replaced with the new combination; repeat Steps 5 through 6; otherwise, the current combination is the optimal solution. Increment ‘m’ such that m = m +1.
7. Once we have an optimal combination, repeat steps 4 through 7 until m = k [9].

Figure 12 presents the pseudo code of the Multi-Center Optimization Problem, providing the detailed computational steps of the algorithm. Figure 13 presents the flowchart for the Multi-Center Optimization Problem, providing the detailed steps of computation from the point of user input to the point of obtaining the final solution.

```
Input graph 'G' with 'N' nodes
Input k
Generate 1-Center solution and move it to set S
Initialize m <- 1
While m <= k
Add a new facility to the set S from 'N' nodes which are not already present in S and move
the combinations to S` such that set S` = S x (N-S)
Calculate the maximum of the shortest distances of each combination from set S` and get the
minimum value(cmin) of the maximum values and the sets with cmin as their maximum
values can be moved to set S`` such that the set S`` = S U {xm} where xm ϵ (N-S)
Improve the solution set by replacing elements of S`` systematically one by one with the
nodes of (N-S``)
If cmin >= set of maximum values
        cmin = new improved minimum of maximum distances
        S`` = Improved set with minimum of maximum distances as cmin
Call "Improve the solution set by replacing elements of S`` systematically one by one with the
nodes of (N-S``)"
Else
        If all nodes are not substituted
Call "Improve the solution set by replacing elements of S`` systematically one by one with the
nodes of (N-S``)"
        Else
                m = m + 1
                S = S``
                EndIf
        EndIf
End While
Optimal solution combination is set S
```

Figure 12. Pseudo Code of the Multi-Center Heuristic Procedure.

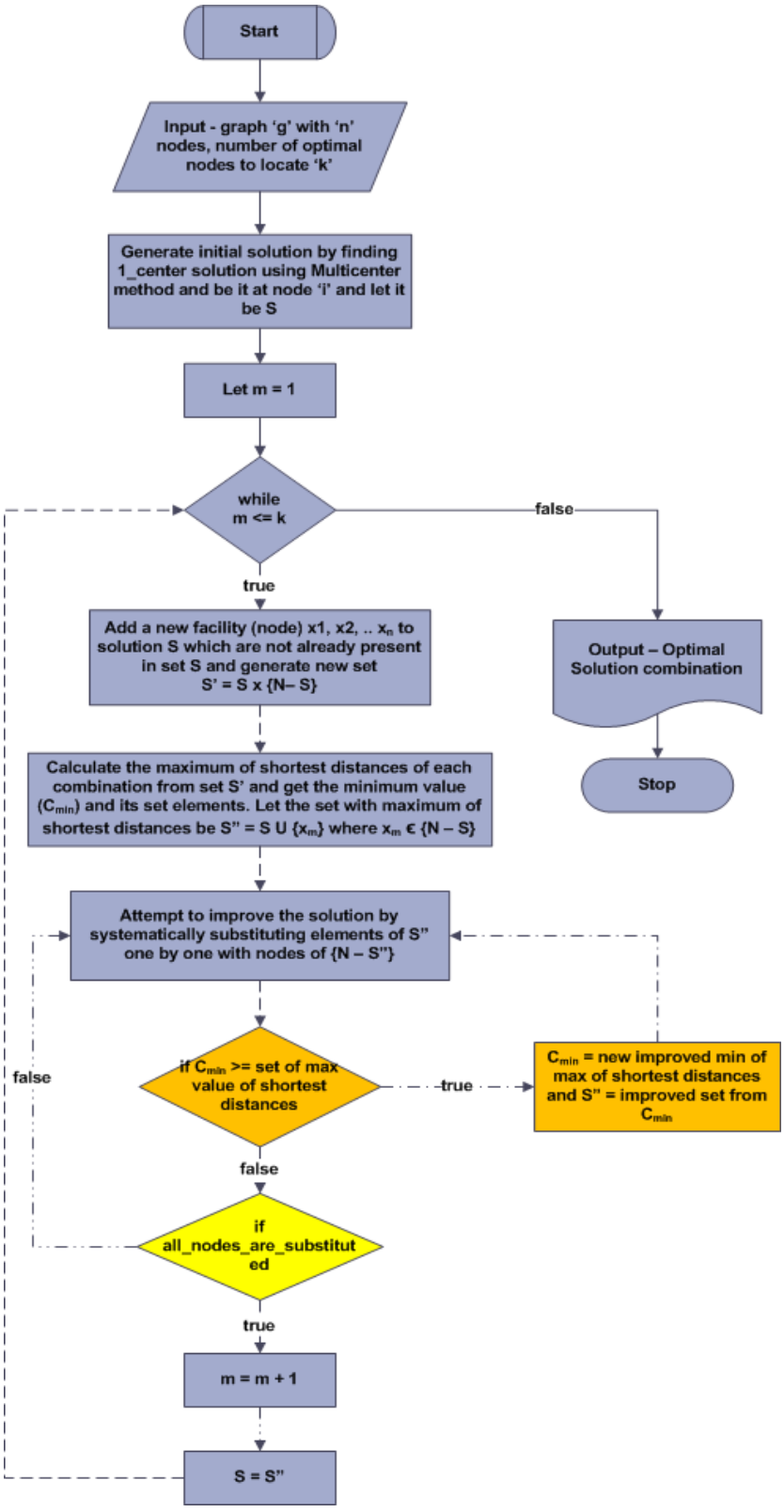

Figure 13. Flowchart of the Multi-Center Heuristic Procedure.

3.2.2.2. Example for the Multi-Center Heuristic

Suppose we are calculating for the two incumbent locations by taking the following network in Figure 14 into consideration:

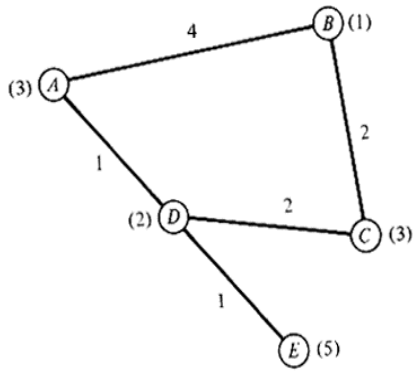

Figure 14. Example Network for the Multi-Center Heuristic Procedure.

For any value of k, the single optimal facility location is the first step for computing the Multi-Center heuristic procedure.

Step 1: We need to construct the shortest distance matrix for this network

$$d_{ij} = \begin{array}{c|ccccc} & A & B & C & D & E \\ A & 0 & 4 & 3 & 1 & 2 \\ B & 4 & 0 & 2 & 4 & 5 \\ C & 3 & 2 & 0 & 2 & 3 \\ D & 1 & 4 & 2 & 0 & 1 \\ E & 2 & 5 & 3 & 1 & 0 \end{array}$$

Step 2: For a single facility location, we need to calculate the maximum travel distance of each node to the facility; i.e., $Maximum\ distance\ of\ each\ node =$ $\{Maximum\ (all\ shortest\ distance\ of\ each\ node\ to\ the\ facilities)$

	A	B	C	D	E	$Max\ distance\ of\ each\ node$
A	0	4	3	1	2	$\text{Max}\ (0, 4, 3, 1, 2) = 4$
B	4	0	2	4	5	$\text{Max}\ (4, 0, 2, 4, 5) = 5$
C	3	2	0	2	3	$\text{Max}\ (3, 2, 0, 2, 3) = 3$
D	1	4	2	0	1	$\text{Max}\ (1, 4, 2, 0, 1) = 4$
E	2	5	3	1	0	$\text{Max}\ (2, 5, 3, 1, 0) = 5$

Because 3 is the minimum for the maximum distances of each node to the facility, 3 is the optimal travel distance, and C is the optimal location. Move the current solution to set S. Therefore, S = {C}.

Step 3: This facility-addition step is where we generate complementary combinations for set S such that {C, A}, {C, B}, {C, D}, {C, E}, and their respective maximum values of travel distances are Max (0, 2, 0, 1, 2), Max (3, 0, 0, 2, 3), Max (1, 2, 0, 0, 1), and Max (2, 2, 0, 1, 0), i.e., 2, 3, 2, and 2. We have the minimum value for maximum travel distances so far as 2 with the node combination {C, A}, {C, D}, and {C, E}. Now, S = {C, A}, {C, D} or {C, E} is our incumbent solution but not our final solution.

Step 4: This solution-improvement step is where we replace each node of set S (Let us consider set S = {C, D}.) and try to get a better solution by generating candidate combinations such as {C, A}, {C, B}, {C, E}, {A, D}, {B, D}, and {E, D}. We already calculated travel distances for the first three combinations, and we may avoid calculating those combinations again. Now, the maximum travel distances for candidate combinations {A, D}, {B, D}, and {E, D} are Max (0, 4, 2, 0, 1), Max (1, 0, 2, 0, 1), and Max (1, 4, 2, 0, 0), i.e., 4, 2, and 4. Now, we need to check if we have any maximum travel distances less than or equal to 2. If yes, we need to repeat Step 5 for the new incumbent solution until we obtain a value less than 2.

Step 5: If we obtain a travel distance less than 2, say “w,” then “w” is our new total travel distance, and the respective combination is our final set; otherwise, 2 is our final incumbent solution, and all the combinations where the maximum travel distances to the

facilities are 2 are considered incumbent solutions for the heuristic. Many times, this solution happens to be our final, optimum solution.

3.2.3. Visual Representation for the Multi-Center Problem

Let us consider the IEEE Standard 14-Bus System, a test system used for research and educational purposes, with 14 nodes and 19 edges [19]. Let us take an example of placing 7 PMUs at different locations.

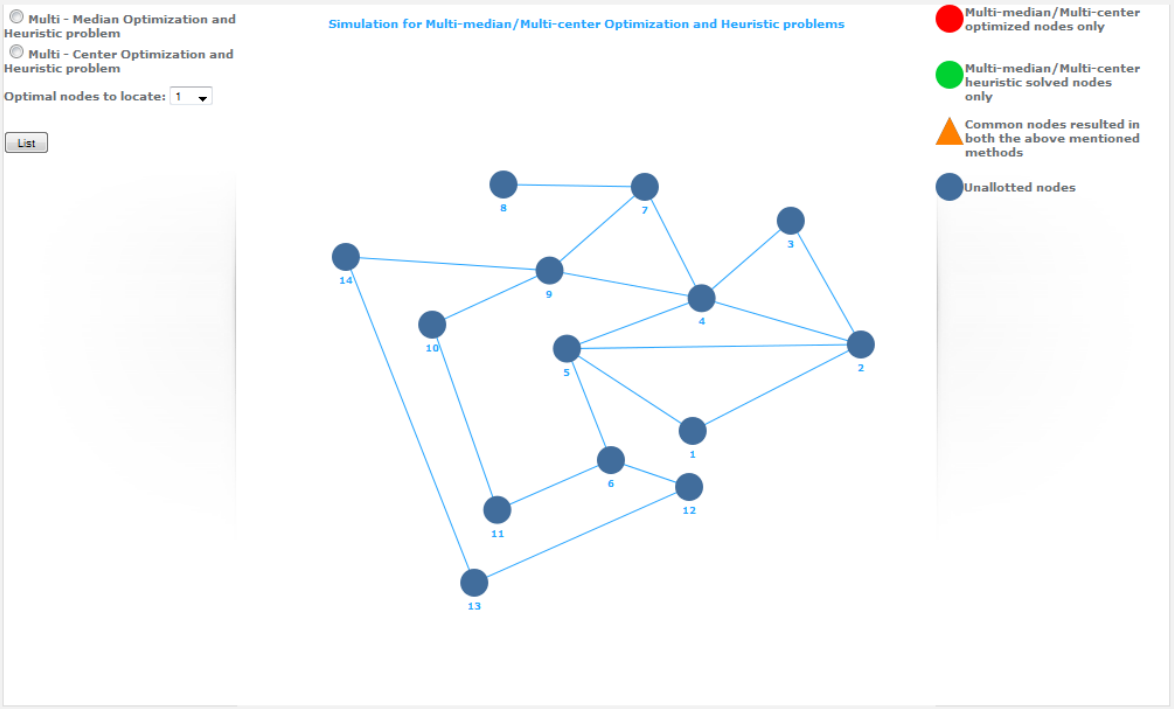

Figure 15. IEEE 14-Bus System for the Multi-Center Problem.

Figure 15 illustrates the IEEE 14-Bus System. The left navigation of the page allows the user to select the number of optimal nodes to locate for PMUs. When the list is clicked, the nodes are highlighted depending on the selected problem method.

Figure 16 illustrates different node locations at optimal positions for the optimization problem and incumbent locations for the heuristic procedure. In our example

with seven optimal locations, green circle nodes represent only Multi-Center Optimization; red circle nodes represent only the Multi-Center Heuristic; and yellow triangles represent the common nodes. Hence, the optimal solution of the IEEE Standard 14-Bus System for locating seven nodes is 1, 2, 3, 4, 7, 11, and 1,3 and the incumbent solution for locating seven nodes using the Multi-Median heuristic are 1, 2, 3, 4, 5, 6, and 7.

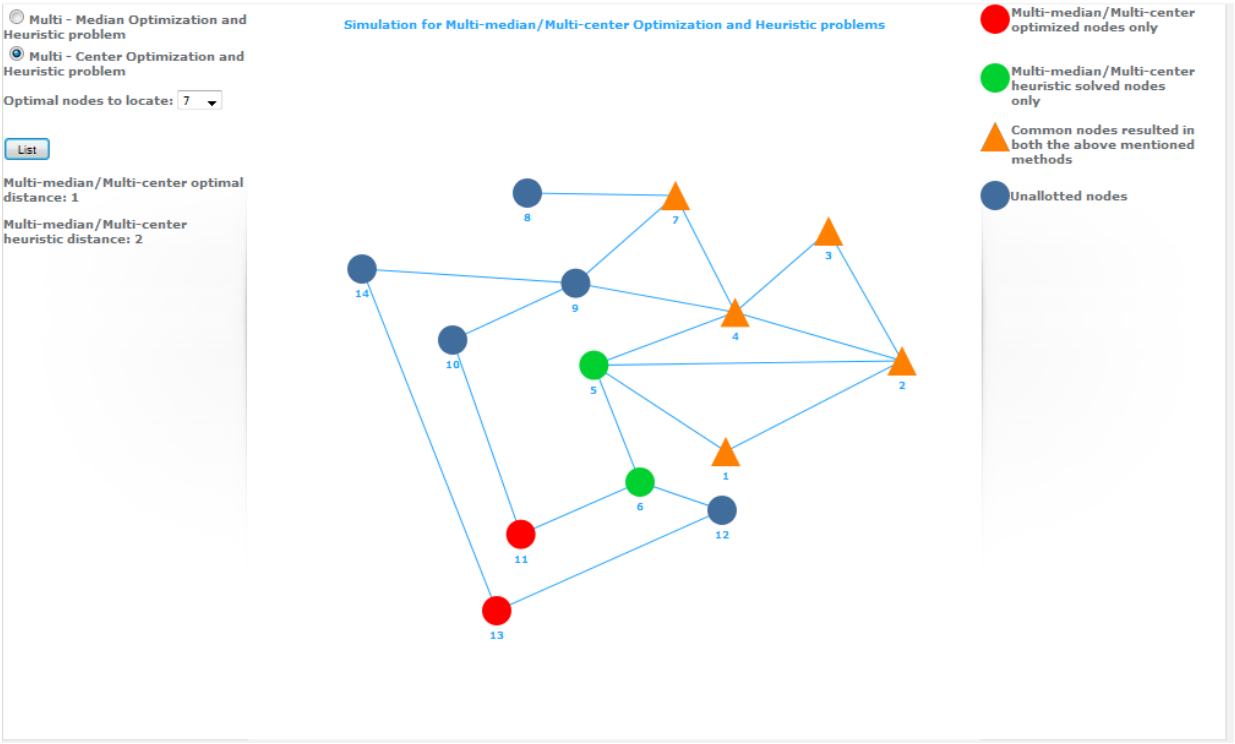

Figure 16. Visual Example of the Multi-Center Optimization and Heuristic Results.

4. IMPLEMENTATION

To implement the solution for the K-Median and K-Center problem, various tools are used in this research, including NetBeans, the JGraphT application programming interface (API), CombinatoricsLib API, and Random Network Generator.

4.1. NetBeans

NetBeans is an integrated development environment (IDE) that is primarily used for writing and organizing Java code and APIs in this research. It is an open-source project developed by the worldwide developer community. It is relatively simple and intuitive, and it provides support for working with third-party libraries in a project that has been developed. Based on these advantages, NetBeans is the primary choice for implementing the solution over other IDEs. Further, it also provides support for easy debugging which is handy when implementing solutions for complex problems [20, 21].

4.2. JGraphT

JGraphT is an open-source, Java-based graph library that provides data structure and algorithm support for mathematical graph-theory problems. It supports various graphs, such as directed graphs, undirected graphs, weighted graphs, unweighted graphs, simple graphs, multi graphs, pseudo graphs, and more [22]. In this research, JGraphT is primarily used for creating a Simple Weighted graph which is nothing but a data structure provided by JGraphT for manipulating graphs/networks. JGraphT also provides APIs for finding the shortest path between nodes in a graph using the famous Dijkstra's Shortest Path algorithm [23]. JGraphT also provides support for visualizing the graphs using the counterpart library called JGraph.

4.2.1. Sample Implementation: Simple Weighted Graph in JGraphT

The implementation in Figure 17 shows how to create a Simple Weighted graph using the JGraphT API as well as how to add vertices and edges after creating the graph. In this paper, the smart grid network is represented as a Simple Weighted Graph.

```
// create a simple weighted graph
SimpleWeightedGraph<String, DefaultWeightedEdge> g = new SimpleWeightedGraph<String,
DefaultWeightedEdge>(DefaultWeightedEdge.class);

// add a vertex – say node = "A"
g.addVertex(node);

// add an edge – say edge = {source = "A" -> target = "B"}
DefaultWeightedEdge edge = g.addEdge(source, target);

// add edge weight – say weight = 1
g.setEdgeWeight(edge, weight);
```

Figure 17. A Sample Implementation of a Simple Weighted Graph.

4.2.2. Sample Implementation: Shortest Path Using Dijkstra Algorithm in JGraphT

The implementation in Figure 18 shows how to find the shortest distance using the JGraphT API between two different nodes; in this example, it is nodes "A" and "E." The DijkstraShortestPath API employs the Dijkstra algorithm to find the shortest distance between any two vertices.

```
// find the shortest distance between the source = "A" and target = "E"
DijkstraShortestPath<String, DefaultWeightedEdge> d = new DijkstraShortestPath<String,
DefaultWeightedEdge>(g, source, target);
Int shortest_distance = d.getPathLength();
```

Figure 18. A Sample Implementation of a Dijkstra Algorithm.

4.3. CombinatoricsLib

CombinatoricsLib is a Java-based library which can be employed to generate different permutations or combinations for a given set of items. The library can be

employed to generate combinatorial objects, such as simple permutations, permutations with repetitions, simple combinations, combinations with repetitions, subsets, integer partitions, list partitions, and integer compositions [24].

In this research, CombinatoricsLib is employed to generate the nC_r combinations of items. For instance, if there are 10 nodes in a network and if we need to find 3 optimal nodes to locate and place the PMUs, then by using CombinatoricsLib, one can generate $10C_3$ combinations where one of the combinations results in an optimal combination.

4.3.1. Sample Implementation: Generating Combinations Using CombinatoricsLib

The implementation in Figure 19 basically creates an initial vector with the given array (say {a, b, c}), and then using the initial vector and the given size (say 2), the API generates all possible combinations which are nothing but {{a, b}, {b, c}, {a, c}}. These generated combinations are used as needed in the research.

```
// create the initial vector
ICombinatoricsVector<String> vector = Factory.createVector(array);

// create a simple combination generator to generate 3-combinations of the initial vector
Generator<String> gen = Factory.createSimpleCombinationGenerator(vector, size);

// print all possible combinations
for (ICombinatoricsVector<String> combination : gen)
{
        print (combination.getVector());
}
```

Figure 19. A Sample Implementation for Generating Combinations.

4.4. Graphical User Interface

An interface has been developed to make it easier for importing the network in the form of an .xml file and getting the output of optimal or incumbent nodes based on the user's selection from among the options: Hakimi's Theorem (Multi-Median Optimization),

Multi-Median Heuristic Procedure, Multi-Center Optimization Problem, and Multi-Center Heuristic Procedure.

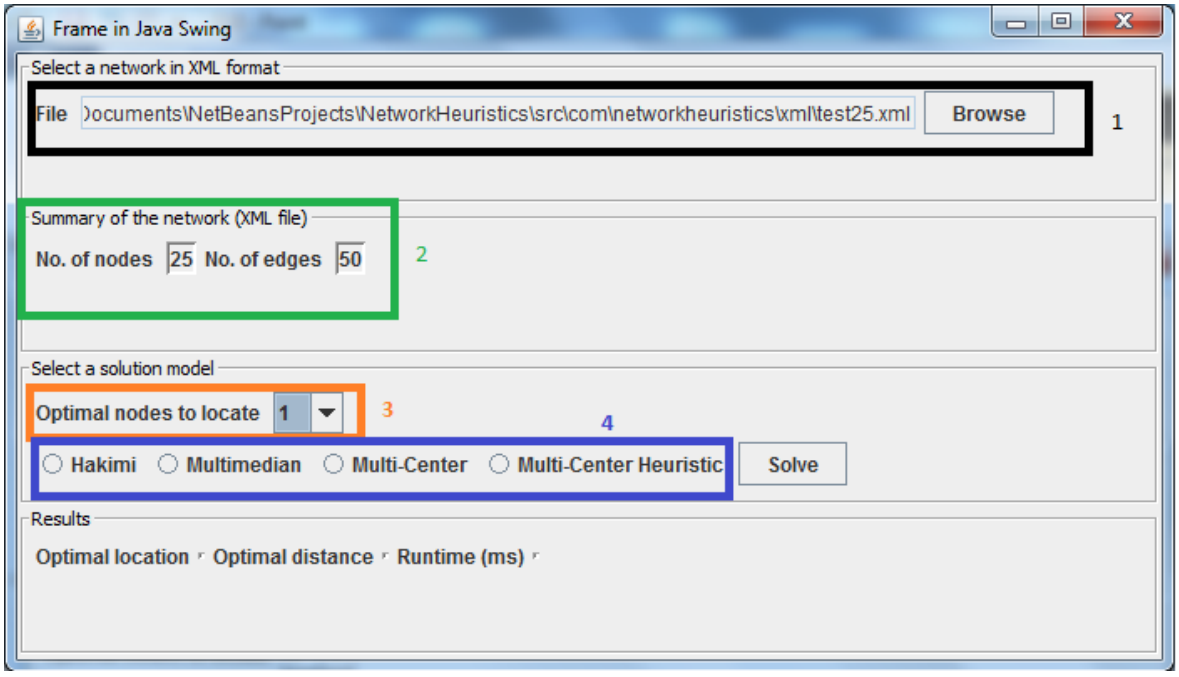

Figure 20. Graphical User Interface for Demonstration.

In Figure 20, each section represents as follows:

1. .xml file which contains a network can be imported.
2. Represents the summary of the network by providing information about the number of nodes and edges for the network that was imported in step 1.
3. The user is allowed to select the number of optimal/incumbent nodes to be located in the network.
4. The user is allowed to select the method from the options: Multi-Median Optimization, Multi-Median Heuristic, Multi-Center Optimization, and Multi-Center Heuristic.

Results: This section displays the final output required, i.e., optimal/incumbent locations, optimal distance, and the amount of time taken in milliseconds.

4.5. Random Network Generator

Given the number of nodes and edges, the Random Network Generator generates a random network in the form of an .xml file. Based on the following rules, the API generates a random network:

- We should ensure that every node is connected to at least one edge. (The source node and target node of an edge can be picked by selecting a random number between 1 and the number of nodes.)
- The source node should not be the same as the target node.
- For every edge, we should ensure that the edge does not already exist for that particular source node and target node.

The steps of computation are as follows:

1. User is allowed to input number of nodes and number of edges and index = 0
2. A random number from 1 to number of nodes is chosen for source and target.
3. If the edge is already created or source and target are same then we need to decrement index by 1.
4. If not, create an edge with the source and target and increment index by 1.
5. Repeat step 2 through step 4 until index is equal to number of edges.
6. A graph is created with all the edges in the form of .xml

Figure 21 presents the format of XML document which includes the list of node, list of edges and their respective lengths. Figure 22 illustrates the algorithm for generating a random network by taking user inputs for the number of nodes and edges.

```
<?xml version="1.0" encoding="UTF-8"?>
<graph>
   <nodes>
      <node>node1</node>
      <node>node2</node>
       ...
   </nodes>
   <edges>
      <edge>
         <source>source1</source>
         <target>target1</target>
         <distance>1</distance>
      </edge>
      <edge>
         <source>source2</source>
         <target>target2</target>
         <distance>1</distance>
      </edge>
       ...
   </edges>
</graph>
```

Figure 21. XML Format Generated by Random Network Generator.

```
Input number of nodes
Input number of edges
Initialize index <- 0
While index < number of edges
        Source = a random number from 1 to number of nodes is chosen
        Target = a random number from 1 to number of nodes is chosen
        If (edge already exists || Source = Target)
                Decrement index
                continue
        Else
                Create an edge with the Source and Target
        End If
        Increment index
End While
Create graph with all the edges in .xml format
```

Figure 22. Pseudo Code for Random Network Generator.

5. EXPERIMENTS AND RESULTS

This chapter mainly focuses on the computational results of experiments conducted on several networks, such as the IEEE Standard 14-Bus System, the 30-Bus System, and some random networks.

5.1. IEEE Standard Bus Systems

IEEE Standard Bus Systems are test systems which are used for research and education purpose in a power-system management environment. These test systems are preferred for experimenting [19]. In our research, we used the following Standard IEEE Bus Systems:

- *IEEE Standard 14-Bus System:* The Standard 14-Bus System contains five synchronous machines with IEEE type-I exciters, where three of them are synchronous compensators. These compensators are used for sustaining reactive power. The other 11 nodes are loads. In our research, this bus system is used to find optimal locations for PMUs such that it helps with voltage calculation [19].

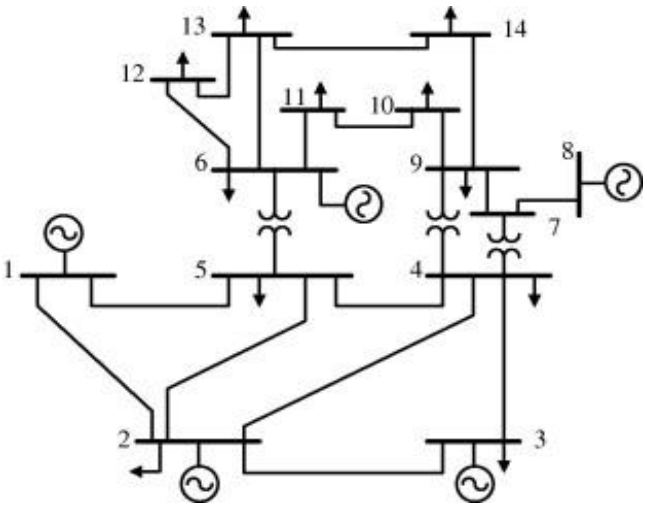

Figure 23. IEEE Standard 14 – Bus System [18].

Figure 23 illustrates the diagram for the Standard IEEE 14-Bus System.

- *IEEE Standard 30-Bus System:* This bus system is similar to the IEEE Standard 14-Bus System where six of the nodes are synchronous generators and four of the nodes are transformers. In this case, we have 21 loads [19]. Figure 24 illustrates the diagram for the Standard IEEE 30-Bus System.

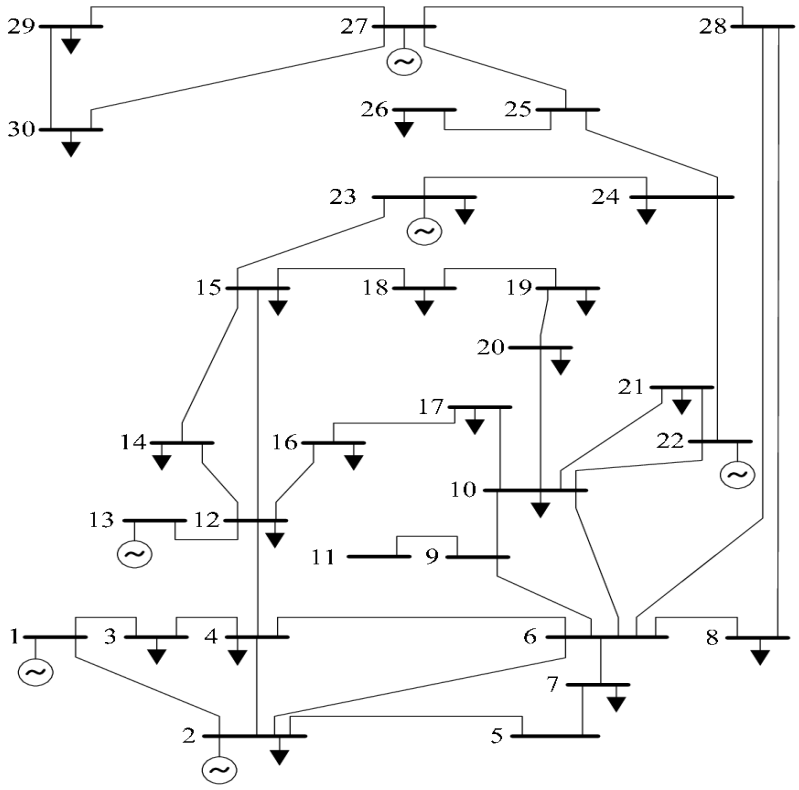

Figure 24. IEEE Standard 30-Bus System [17].

The experiments conducted are as follows:

- Time comparisons of Multi-Median Optimization Problem and Heuristic for the IEEE Standard 14-Bus System
- Time comparisons of the Multi-Center Optimization Problem and the Multi-Center Heuristic for the IEEE Standard 14-Bus System

- Time comparisons of Multi-Median Optimization Problem and Heuristic for the IEEE Standard 30-Bus System
- Time comparisons of the Multi-Center Optimization Problem and the Multi-Center Heuristic for the IEEE Standard 30-Bus System
- Time comparisons of the Multi-Median Optimization and the Multi-Median Heuristic for 10 random networks generated with 35 nodes, 40 edges, and 10 optimal locations
- Time comparisons of the Multi-Center Optimization and the Multi-Center Heuristic for 10 random networks generated with 35 nodes, 40 edges, and 10 optimal locations

5.2. Using the Standard 14-Bus System

Experiments have been conducted on these algorithms using the IEEE Standard 14-Bus System.

5.2.1. Experiment 1

In this experiment, we compare the time taken to compute optimal locations for 1-8 PMUs on a Standard 14-Bus System for the Multi-Median Optimization Problem and Multi-Median Heuristic methods.

Table 1 gives the time comparison values of the Multi-Median Optimization and Heuristic algorithms for the Standard IEEE 14-Bus System. The “Optimal Nodes to Locate/Type” column represents the number of facilities selected, and the “Multi-Median Optimization” and “Multi-Median Heuristic” columns are the amount of time taken by these algorithms (measured in milliseconds). Because the network is smaller in size, there is not much time difference. In this table, optimal node 5 has 0 as its time-taken value, which means the time, in decimals, is less than 0. Figure 25 is the bar-graph representation of Table 1.

Table 1. Runtime comparison between the Multi-Median Optimization and Heuristic (14 bus).

Optimal Nodes to Locate/Type	Runtime (in Milliseconds)	
	Multi-Median Optimization	Multi-Median Heuristic
1	63	31
2	31	15
3	31	31
4	15	16
5	0	0
6	31	15
7	31	16
8	31	31

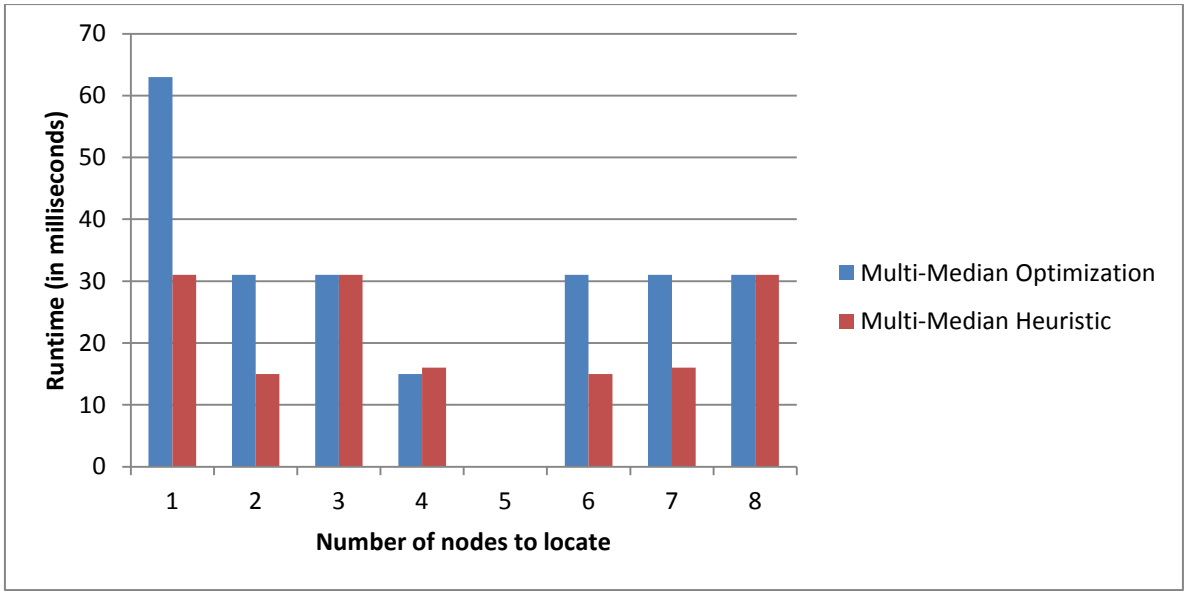

Figure 25. Bar Graph showing Runtime Comparison between the Multi-Median Optimization and Heuristic (14 Bus).

5.2.2. Experiment 2

In this experiment, we compare the time taken to compute optimal locations for 1-8 PMUs on a Standard 14-Bus System for the Multi-Center Optimization Problem and Multi-Center Heuristic methods.

Table 2 gives the time comparison values for the Multi-Center Optimization and Heuristic algorithms of the Standard IEEE 14-Bus System. The “Optimal Nodes to

Locate/Type" column represents the number of facilities selected, and the "Multi-Center Optimization" and "Multi-Center Heuristic" columns are the amount of time taken by these algorithms (measured in milliseconds). Figure 26 is the bar-graph representation of Table 2.

Table 2. Runtime comparison between the Multi-Center Optimization and Heuristic (14 bus).

Optimal Nodes to Locate/Type	Runtime (in Milliseconds)	
	Multi-Center Optimization	Multi-Center Heuristic
1	47	16
2	47	31
3	31	16
4	16	15
5	16	15
6	31	16
7	31	16
8	31	31
9	47	47

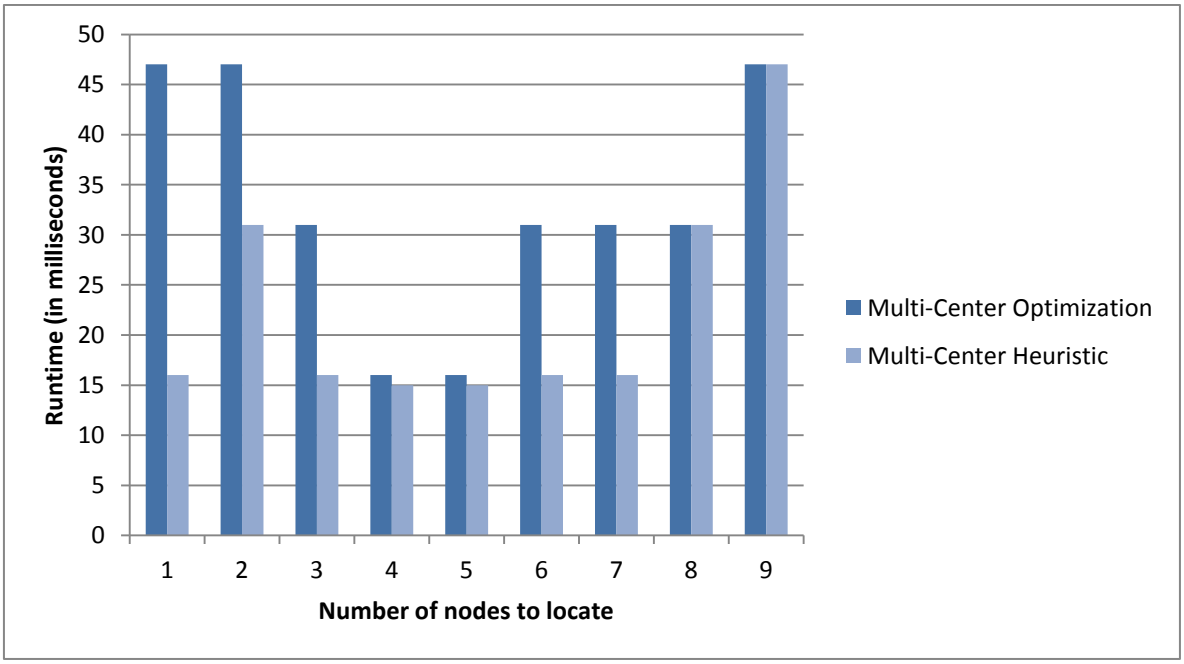

Figure 26. Bar Graph showing Runtime Comparison between the Multi-Center Optimization and Heuristic (14 Bus).

5.3. Using the Standard 30-Bus System

Experiments have been conducted on the algorithms using the Standard IEEE 30-Bus System.

5.3.1. Experiment 3

In this experiment, we compare the time taken to compute optimal locations for 1-8 PMUs with the Standard 30-Bus System for the Multi-Median Optimization Problem and Multi-Median Heuristic methods.

Table 3 gives the time comparison values for the Multi-Median Optimization and Heuristic algorithms of the Standard IEEE 30-Bus System. The "Optimal Nodes to Locate/Type" column represents the number of facilities selected, and the "Multi-Median Optimization" and "Multi-Median Heuristic" columns are the amount of time taken by these algorithms (measured in milliseconds). Figure 27 is the bar-graph representation of Table 3.

Table 3. Runtime comparison between the Multi-Median Optimization and Heuristic (30 bus).

Optimal Nodes to Locate/Type	Runtime (in Milliseconds)	
	Multi-Median Optimization	Multi-Median Heuristic
1	203	32
2	31	16
3	93	47
4	374	47
5	2059	281
6	10702	1263
7	-	5834

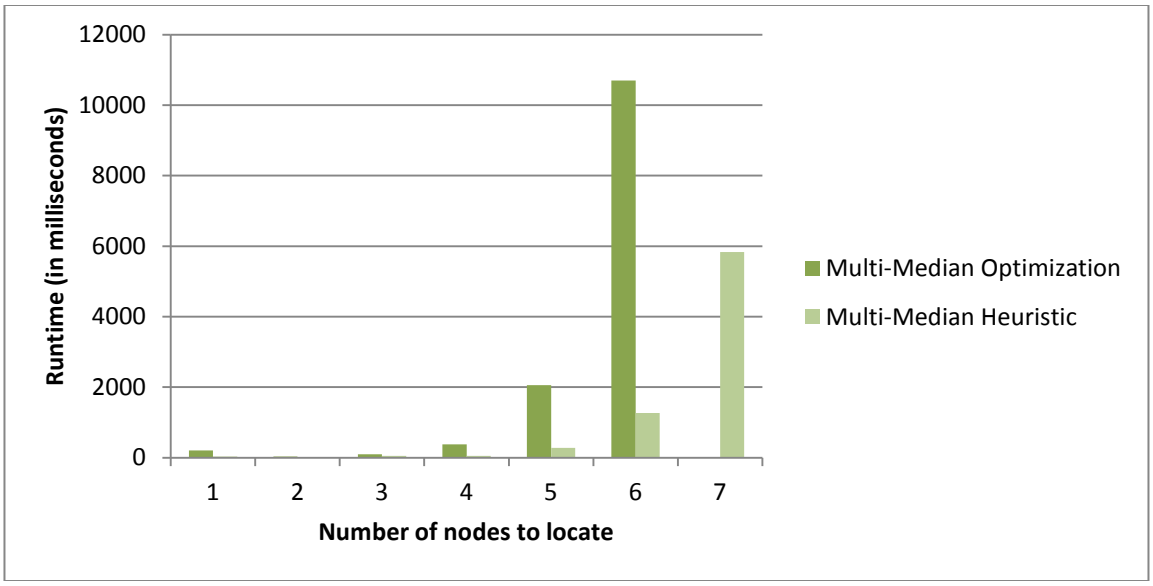

Figure 27. Bar Graph showing Runtime Comparison between the Multi-Median Optimization and Heuristic (30 Bus).

5.3.2. Experiment 4

In this experiment, we compare the time taken to compute optimal locations for 1-8 PMUs on a Standard 14-Bus System for the the Multi-Center Optimization Problem and Multi-Center Heuristic methods.

Table 4 gives the time comparison values of the Multi-Center Optimization and Heuristic algorithms for the Standard IEEE 30-Bus System.

Table 4. Runtime comparison between the Multi-Center Optimization and Heuristic (30 bus).

Optimal Nodes to Locate/Type	Runtime (in milliseconds)	
	Multi-Center Optimization	Multi-Center Heuristic
1	515	78
2	47	31
3	94	31
4	327	93
5	2652	218
6	10795	1108
7	-	5226

The “Optimal Nodes to Locate/Type” column represents the number of facilities selected, and the “Multi-Center Optimization” and “Multi-Center Heuristic” columns are the amount of time taken by these algorithms (measured in milliseconds). Figure 28 is the bar-graph representation of Table 4.

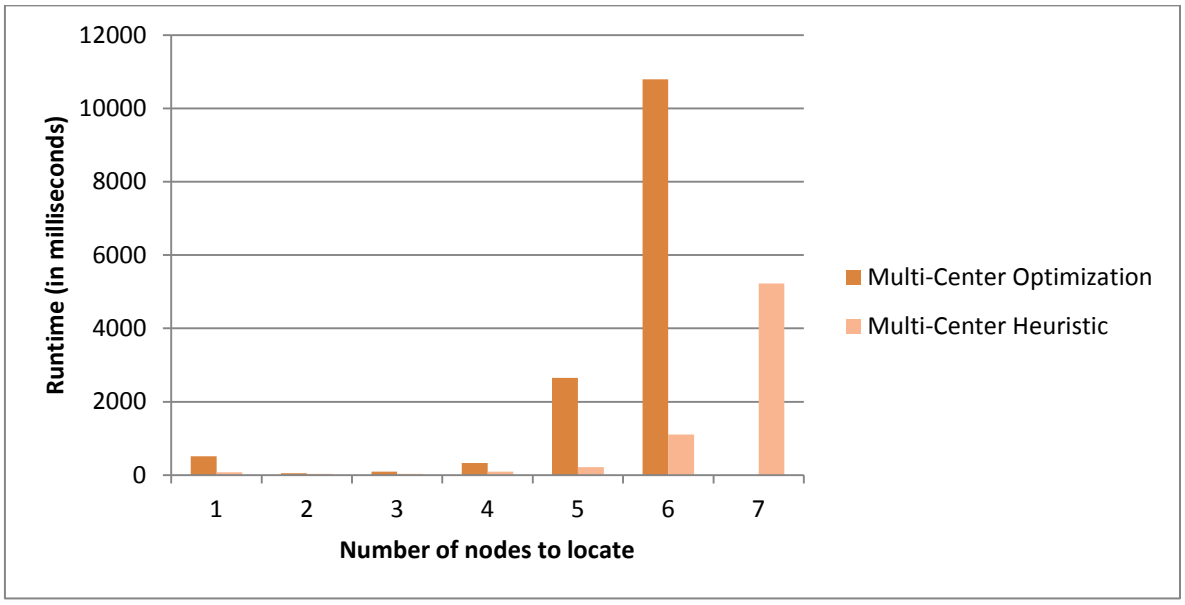

Figure 28. Bar Graph showing Runtime Comparison between Multi-Center Optimization and Heuristic (30 Bus).

5.4. Using Random Networks

Experiments have been conducted on these algorithms using random networks as follows.

5.4.1. Experiment 5

In this experiment, 10 random networks are generated with 35 nodes, 40 edges, and 5 optimal locations. This experiment is conducted for comparing the time constraints of different networks with same number of nodes, edges, and optimal locations for the Multi-Median Optimization and Heuristic Procedures.

The 10 random networks are generated using a Random Network Generator and ensuring that no node is left disconnected. The same set of networks is used to calculate average runtime, standard deviation, and variance.

Table 5 gives the time comparison values of the Multi-Median Optimization and Heuristic algorithms for 10 random networks labeled from 1 through 10 for the 35 nodes and 40 edges. The "Network" column represents the selected network, and the "Multi-Median Optimization" and "Multi-Median Heuristic" columns are the amount of time taken by these algorithms (measured in milliseconds).

Table 5. Runtime information for 10 random networks with 35 nodes, 40 edges, and 10 optimal locations for the Multi-Median Optimization and Heuristic.

Network	Runtime (in Milliseconds)	
	Multi-Median Optimization	Multi-Median Heuristic
1	8003	764
2	7457	624
3	7691	811
4	7520	796
5	7144	640
6	6770	436
7	6786	764
8	6897	655
9	6585	656
10	6601	562

Figure 29 is the bar-graph representation of Table 5. Here, we observe a large difference in the Multi-Median Optimization and Heuristic algorithms. Also, the heuristic is better than the optimization problem in terms of the time taken.

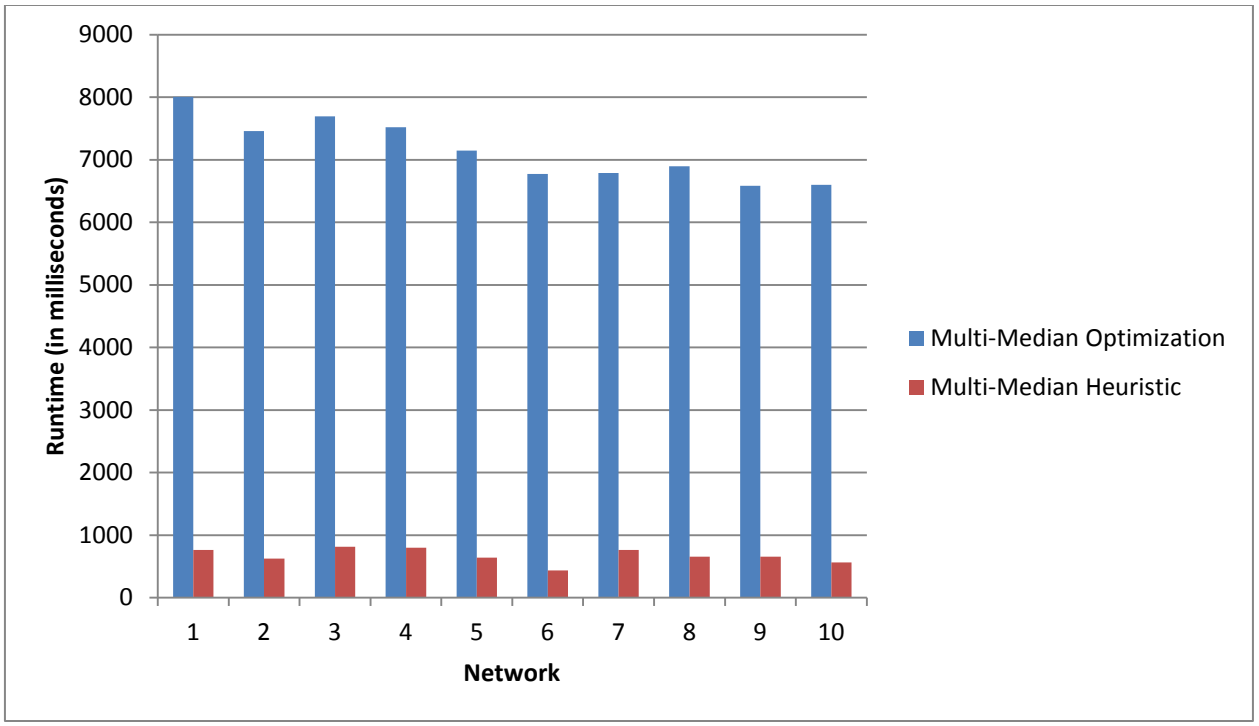

Figure 29. Bar Graph showing Runtime Comparison between the Multi-Median Optimization and Heuristic (10 Runs).

5.4.2. Experiment 6

In this experiment, 10 random networks are generated with 35 nodes, 40 edges, and 5 optimal locations. This experiment is conducted for test comparison and to test the average time taken by different networks with the same number of nodes, edges, and optimal locations for the Multi-Center Optimization and Heuristic Procedures.

Table 6 gives the time comparison values for the Multi-Center Optimization and Heuristic algorithms for 10 random networks labeled from 1 through 10 with 35 nodes and 40 edges. The “Network” column represents the selected network, and the “Multi-Center Optimization” and “Multi-Center Heuristic” columns are the amount of time taken by these algorithms (measured in milliseconds).

Table 6. Runtime information for 10 random networks with 35 nodes, 40 edges, and 10 optimal locations for the Multi-Center Optimization and Heuristic.

Network	Runtime (in Milliseconds)	
	Multi-Center Optimization	Multi-Center Heuristic
1	6412	624
2	6302	577
3	5974	624
4	6942	593
5	5865	483
6	6053	484
7	6350	499
8	6773	484
9	6148	546
10	6132	515

Figure 30 is the bar-graph representation of Table 6. Here, we observe a large difference in the Multi-Center Optimization and Heuristic algorithms. Also, the heuristic is better than the optimization problem in terms of the time taken.

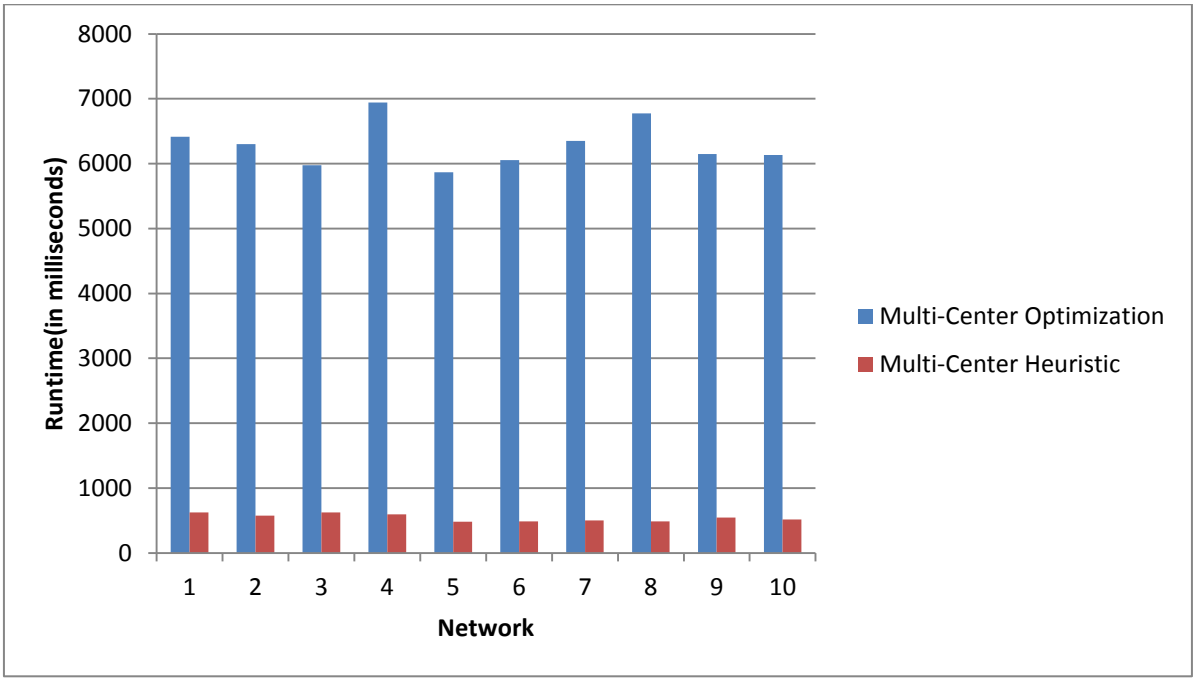

Figure 30. Bar Graph showing Runtime Comparison between the Multi-Center Optimization and Heuristic (10 Runs).

5.5. Observations

Let us consider Experiment 5: 10 random networks with 35 nodes, 40 edges, and 5 optimal locations. The average runtime, standard deviation, and variance are calculated for both the Multi-Median Optimization and Heuristic Procedures.

Average runtime (Multi-Median Optimization) = 7,145.4 milliseconds

Standard Deviation (Multi-Median Optimization) = 495.8488 milliseconds

Variance (Multi-Median Optimization) = 245,866 milliseconds^2

Average runtime (Multi-Median Heuristic) = 670.8 millseconds

Standard Deviation (Multi-Median Heuristic) = 117.0373 milliseconds

Variance (Multi-Median Heuristic) = 13,697.73 milliseconds^2

Let us consider Experiment 6: 10 random networks with 35 nodes, 40 edges, and 5 optimal locations. The average runtime, standard deviation, and variance are calculated for both the Multi-Center Optimization and Heuristic Procedures.

Average runtime (Multi-Center Optimization) = 6,295.1 milliseconds

Standard Deviation (Multi-Center Optimization) = 342.9510655 milliseconds

Variance (Multi-Center Optimization) = 117,615.4333 milliseconds^2

Average runtime (Multi-Center Heuristic) = 542.9 milliseconds

Standard Deviation (Multi-Center Heuristic) = 57.80129 milliseconds

Variance (Multi-Center Heuristic) = 3,340.989 milliseconds^2

As the network size increases in terms of the number of nodes and edges, the differences for optimization and heuristic procedures also increase. The larger the network, the better the heuristic procedure is over the optimization problem**.**

6. CONCLUSION, LIMITATIONS, AND FUTURE WORK

6.1. Conclusion

Phasor Measurement Units are used to monitor smart grid networks. Because PMUs are expensive and not economically feasible, we have limited the number of PMUs to locate. The current research aims to overcome these problems by proposing a heuristic method. This method can be applied to place the available PMUs on a smart grid network at incumbent locations. Various factors, such as minimizing the total average distance and minimizing the maximum distance of PMUs, are considered depending on the method. Based on these above factors we have utilized the K-Median algorithm for minimizing the total average transportation distance and the K-Center algorithm for minimizing the maximum distance of every node to the facilities.

A code was developed for K-Median Optimization and Heuristic procedures, K-Center Optimization and Heuristic procedures and for generating random network, given the number of nodes and edges. Further, testing has been conducted on IEEE Standard 14 and 30 bus system network for both optimization and heuristic procedure. Also, testing has been conducted on 10 random networks generated with equal number of nodes and edges.

The primary objective of this research is to develop a heuristic which can locate the PMUs on a smart grid Network in a shorter time and, therefore, calculate fewer combinations as compared to the optimization problem. The results were in tandem with our objectives laid down for this research. The computational time difference (in milliseconds) between the optimization problem and the heuristic procedure is presented by graphs in Chapter 4. Based on our research it was observed that as the network size increases, the heuristic method is more feasible than the optimization method.

6.2. Limitations

Although these optimization algorithms give the optimal solution for k locations, as the network size increases, the number of calculations for nC_k combination increases. One of the significant limitations for employing these algorithms is to compute nC_k combinations of data; we need a very large memory to store the data.

6.3. Future Work

In some cases, the heuristic solutions may not be exactly similar to optimal solutions. In such scenarios, a case-improvement algorithm can be defined. This algorithm aims to obtain the same solution as the optimal one and is much quicker than the optimization problem. The memory problem can be overcome by improving the algorithms in such a way that computing a few combinations can be avoided to improve and accelerate performance.

BIBLIOGRAPHY

[1] Energy.gov. (n.d.). *Smart Grid*. Retrieved October 15, 2012, from http://energy.gov/oe/technology-development/smart-grid

[2] Smart grid. (n.d.). *Wikipedia*. Retrieved October 15, 2012, from http://en.wikipedia.org/wiki/Smart_grid

[3] Phasor measurement unit. (n.d.). *Wikipedia*. Retrieved October 16, 2012, from http://en.wikipedia.org/wiki/Phasor_measurement_unit

[4] Gyllstrom, D., Rosensweig, E., & Kurose, J. (n.d.). Max Observability PMU Placement with Cross-Validation. *Max Observability PMU Placement with Cross-Validation*. Retrieved October 16, 2012, from http://people.cs.umass.edu/~dpg/pubs/tech11.pdf

[5] Singh, B., Sharma, N., Tiwari, A., Verma, K., & Singh, S. (2011). Applications of Phasor Measurement Units (PMUs) in electric power system networks incorporated with FACTS controllers. *International Journal of Engineering, Science and Technology, 3*(3), 64-82. Retrieved October 16, 2012 .

[6] Narendra, K. (n.d.). Role of Phasor Measurement Unit (PMU) in Wide Area Monitoring and Control. *Role of Phasor Measurement Unit (PMU) in Wide Area Monitoring and Control*. Retrieved October 17, 2012, from http://www.erlphase.com/downloads/application_notes/Roles_of_PMUs_in_Wide_Area_Monitoring_and_Control.pdf

[7] G, S. (2012, June 25). Mod-08 Lec-32 Location problems -- p median problem, Fixed charge problem. *YouTube*. Retrieved October 23, 2012, from http://www.youtube.com/watch?v=82OUSjOM2bg

[8] Facility location. (n.d.). *Wikipedia*. Retrieved October 24, 2012, from http://en.wikipedia.org/wiki/Facility_location

[9] Larson, R. C., & Odoni, A. R. (1981). Network Applications. In *Urban Operations Research*. Englewood Cliffs, NJ: Prentice-Hall. Retrieved November 15, 2012, from http://web.mit.edu/urban_or_book/www/book/

[10] J, R. (2005, August 11). *Methods for Solving the p-Median Problem: An Annotated Bibliography*. Retrieved October 27, 2012, from http://ramanujan.math.trinity.edu/tumath/research/reports/report96.pdf

[11] Lazic, N., Frey, B. J., & Aarabi, P. (2010). Solving the Uncapacitated Facility Location Problem Using Message Passing Algorithms [Abstract]. *Journal of Machine Learning Research*, *9*, 429-436. Retrieved October 27, 2012, from http://jmlr.csail.mit.edu/proceedings/papers/v9/lazic10a/lazic10a.pdf

[12] Shmoys, D. B., Tardos, É, & Aardal, K. (1998). Approximation algorithms for facility location problems (extended abstract). *STOC '97 Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, 265-274. doi: 10.1145/258533.258600

[13] Jain, K., Mahdian, M., Markakis, E., Saberi, A., & Vazirani, V. V. (2003). Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, *50*(6), 795-824. doi: 10.1145/950620.950621

[14] Jain, K., Mahdian, M., & Saberi, A. (n.d.). A new greedy approach for facility location problems. *STOC '02 Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing*, 731-740. doi: 10.1145/509907.510012

[15] Hajiaghayi, M. T., Mahdian, M., & Mirrokni, V. S. (2003). The facility location problem with general cost functions. *Networks, 42*(1), 42-47. doi: 10.1002/net.10080

[16] Khiabani, V., Yadav, O., & Kavesseri, R. (2011). *Reliability-based placement of Phasor Measurement Units in power systems*. Manuscript submitted for publication, North Dakota State University, Fargo.

[17] Younes, M., & Khodja, F. (n.d.). A Hybrid Harmony Search Algorithm Approach for Optimal Power Flow.

[18] Gitizadeh, M., & Kalantar, M. (2009). A novel approach for optimum allocation of FACTS devices using multi-objective function. *Energy Conversion and Management*, *50*(3), 682-690. doi: 10.1016/j.enconman.2008.10.009

[19] Test Systems. (n.d.). *Test System*. Retrieved December 26, 2012, from http://itee.uq.edu.au/pss-l/test system.htm

[20] NetBeans IDE Features. (n.d.). *NetBeans IDE*. Retrieved January 20, 2013, from http://netbeans.org/features/index.html

[21] NetBeans. (n.d.). *Wikipedia*. Retrieved January 20, 2013, from http://en.wikipedia.org/wiki/NetBeans

[22] JGraphT. (n.d.). *Welcome to JGraphT*. Retrieved January 20, 2013, from http://jgrapht.org/

[23] JGraphT: A free Java graph library. (n.d.). *JGraphT: A Free Java Graph Library*. Retrieved February 22, 2013, from http://www.jgrapht.org/javadoc

[24] Combinatoricslib 2.0 Features. (n.d.). *Combinatoricslib - Very Simple Java Library to Generate Permutations, Combinations and Other Combinatorial Sequences*. Retrieved January 22, 2013, from http://code.google.com/p/combinatoricslib/