Planning Stage Reliable Operation of Dynamic Microgrids

by

Xuefei Zhu

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 1, 2023

Keywords: Dynamic Microgrid, Artificial Intelligence, Steady State Analysis, Load Shedding, Economic Load Dispatch

Abstract

The deployment of renewable energy generation is increasing at the distribution level in power systems. The concept of microgrids reduces the complex control burden from the system operator and lowers electricity prices. As the number of nested microgrids grows in the distribution system, there is a paradigm transition from static to dynamic microgrids. This research introduced reliable planning for the operation of dynamic microgrids in the planning stages to provide solutions in case of a real contingency, to provide solutions in case of real contingencies. The research begins with the partitioning of distribution networks in the planning stage based on different objective functions. Several constraints must be maintained during the partitioning process. This research proposes three modified algorithms to address different problems. The effectiveness of these algorithms was verified through the IEEE 33 Radial Bus System in PSCAD and the IEEE 39-Bus system in PSLF.

The worst case in a distribution system is the sudden trip of one or more generators. Load shedding is equivalent to increasing power generation in steps. With the objective of reducing interruption costs, a greedy-based sorting algorithm was proposed to minimize outage costs. The algorithm was implemented in Python and tested on the IEEE 39-Bus system. In addition to addressing economic issues on the demand side, economic considerations were also considered on the generation side. During the planning stage of reliable operation, the objective was to minimize the fuel cost of various generators while maintaining power balance. Particle swarm optimization was utilized to generate the best results, and the algorithm was implemented in Python. The test results from a three-generator and thirteen-generator systems were compared with those published in technical papers.

Acknowledgments

This work represents the culmination of five years of research, starting in August 2018, during which I received invaluable support from the committee members and the Power Lab. First and foremost, I wish to express my gratitude to my advisory committee chair, Dr. Eduard Muljadi, for his continuous guidance along my research journey, his academic and mental support, and his assistance in my smooth adaptation to a new environment. I would also like to extend my appreciation to my research advisory committee members: Dr. Mark Nelms, Dr. John Hung, and Dr. Mark Halpin. I want to thank Dr. Mark Nelms for accepting me as a PhD student in the Electrical and Computer Engineering Department. Furthermore, his friendly nature and generosity have consistently been helpful in my research career. Dr. Hung left an impression on me with his expertise in state variables and control classes, always delivering complex concepts in a comprehensible manner. His guidance during my general oral exam was particularly beneficial. I would also like to acknowledge Dr. Mark Halpin for imparting the fundamental knowledge of power systems. His unique teaching style has made the complex subject matter more organized and easily understandable. I am profoundly grateful for the tremendous assistance I received at Auburn University and consider myself fortunate to have studied in the Electrical and Computer Engineering Department.

Throughout this journey, the Power Lab has consistently felt like a home to me at Auburn University. I want to extend my gratitude to Dr. Jinho Kim for his patient guidance at the beginning of my research career at Auburn University. I'd also like to express my appreciation to Sangwon Seo for delivering presentations that facilitated my rapid comprehension of new concepts. Furthermore, Patrick Haney's humor and optimism never failed to brighten my spirits during the challenging moments of my research career. I'm thankful for his hospitality.

I would like to express my gratitude to my girlfriend, Amber, who has been with me through my most difficult times and continues to stand by my side, whether in good times or bad times. I appreciate her for always being a patient listener and a caring person. I want to thank her for consistently expressing love, regardless of her emotional state. Her kindness and compassionate nature always serve as a guiding light during my darkest times. I am truly grateful to have such a strong and kind-hearted woman as my teammate.

My father, Jianwen Zhu, and my mother, Liping Guo, raised me for 18 years and served as my life teachers for 31 years. I am immensely thankful to my parents for their unconditional love and for constantly giving without expecting anything in return. The most difficult aspect of my time at Auburn University has been the inability to be with them and assist them when they face difficulties. Filial piety is a highly esteemed virtue in our culture, and I am deeply indebted to them for the sacrifices they made to support my pursuit of a research career at Auburn. I am truly thankful to have parents like them who remain open-minded even when they may not fully understand. I hope that I can repay them someday in the future.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

REG	Renewable Energy Generation
RoCoF	Rate of Change of Frequency
UFLS	Under Frequency Load Shedding
DER	Distributed Energy Resource
DG	Distributed Generation
MPPT	Maximum Power Point Tracking
MG	Microgrid
PCC	Point of Common Coupling
SCADA	Supervisory Control and Data Acquisition
MMC	Microgrid Centralized Controller
MGCC	Microgrid Centralized Controller
DOE	Department of Energy
SGs	Synchronous Generators
GFG	Grid-Forming Generator
GFL	Grid Following
GFM	Grid Forming
BFS	Breadth-First Search
NP-hard	Non-deterministic Polynomial-time
PSO	Particle Swarm Optimization

1. **Microgrid**

1.1 Microgrid

Due to the technical advances and declining cost of renewable energy generation (REG), deployment of renewable energy generation (REG) is increasing at the distribution level of power systems. REG is not dispatchable and is generally decentralized. In that way, the distribution network is experiencing a transition from a centralized control scheme to a decentralized control scheme. On the one hand, bringing the REG into the distribution network can secure the power supply of critical loads when there is a fault upstream. Hence, the reliability of the distribution network was increased. It can also ease the emission of greenhouse gas because of the environmentally friendly nature of REG. On the other hand, bringing the REG into the distribution network may decrease the inertia of the system since most REGs are inverter-based resources and have small system inertia [1, 2] compared to traditional synchronous generators. The low inertia will mainly cause a rapid rate of change of frequency (RoCoF). Large RoCoF over a certain range will sometimes lead to instability of synchronous generators' synchronization, grid-following failures of inverted-based generation, and deficient responses of under-frequency load shedding (UFLS). Frequency control problems and declined resilience to frequency disturbances are sometimes caused by low inertia in the system. Hence, there is a need to maintain system inertia within a certain range even the penetration level of the system continues increasing.

Distributed energy resources (DER) are defined as sources or groups of sources that are not directly connected to the bulk power system [3]; they include both generators and energy storage technologies capable of exporting power. To integrate distributed generation (DG) into the distribution system, power electronic inverters are the most suitable interfaces to connect both parts. The typical DGs in this industry are traditional power generation synchronous generators (e.g., diesel genset, microturbine) and renewable energy generation (e.g., wind turbine generator, fuel cell, and photovoltaic). Distributed energy resource (DER) covers both the DG and also the energy storage (e.g., battery, flywheel, supercapacitor). Most of the REGs and energy storage have a power electronics converter at interface with the grid. These types of devices are also known as Inverter Based Resources (IBRs). The IBR can be operated in a grid-following mode (GFL) where the real and reactive power can be controlled independently [4-6] and Grid Forming (GFM) mode to provide grid-synchronized to control voltage and system frequency in an islanded microgrid (MG). Normally, the IBR operating in GFM mode are REG operating with energy storage systems or Energy Storage (i.e., Battery Energy Storage System - BESS). The IBR operating in GFL mode synchronized to the grid frequency of the distribution system. REGs can be operated under two modes; the first one is the maximum power point tracking mode (MPPT) to maximize the power output from DG [7, 8], and the second mode is the curtailed mode to support the frequency regulation from the system operator.

As the penetration level of the DERs into the distribution system continues increasing, there is an urgent need to generate a new concept to manage various DGs. A microgrid (MG) is defined as a group of interconnected loads and distributed energy resources with clearly defined electrical boundaries that function as a single controllable entity with respect to the grid and can connect and disconnect from the grid to enable it to operate in both grid-connected mode and islanded mode. Usually, the microgrid is connected to the main distribution grid through the Point of Common Coupling (PCC). MGs could be classified as AC MG, DC MG, and hybrid MG. The AC MG has drawn a lot of attention from both the industry and university scholars because of its practicality in existing AC power systems. A general structure of AC MG is illustrated in Figure 1-1.

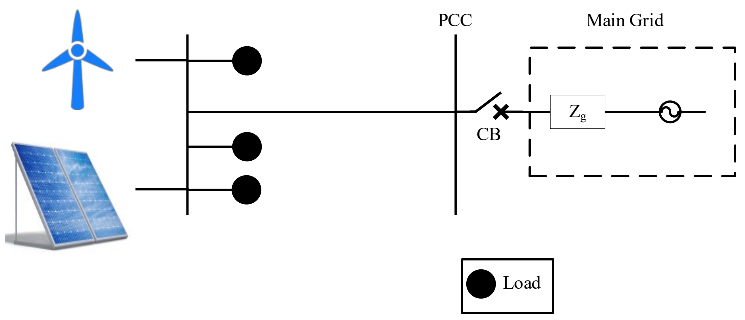


Figure 1-1: Structure of AC microgrid.

In Figure 1-1, the main grid was represented by a system impedance and a voltage source. The AC-DC inverter was utilized to connect REGs to the microgrid, and a circuit breaker is used to connect the microgrid to the main grid. When a microgrid is disconnected from the primary grid because of a contingency in the upstream network, the microgrid will form an islanded system. Compared to traditional power systems, microgrids are able to operate under both islanded mode and grid-connected mode [9-12]. In Figure 1-1, the MG's islanded operation can be achieved by opening the circuit breaker. The MG would disconnect itself from the main grid and operate as a small sub-system to avoid disturbance from the main grid. Meanwhile, the islanded MG should maintain power balance within the small system.

MGs should achieve smooth (seamless) transition from islanded mode to grid-connected mode or vice versa. Since most microgrid is often equipped with energy storage systems, the excess power generation in the microgrid could charge these energy storage systems or can be exported to the main grid. Based on the fact that MGs are usually lack of inertia, islanded MGs are more vulnerable to the system frequency fluctuations, which could be caused by the variability of renewable generation and load consumption. While traditional synchronous generators are often assumed to be supplied with sufficient fuel. That is the reason why energy storage systems are becoming an important component in MG operation.

Traditionally, the Supervisory Control and Data Acquisition (SCADA) will handle communication between different components in power system [13, 14]. The whole system is a centralized structure with a microgrid control center to give all operation commands. However, the control strategy within MGs is more distributed and more flexible. It could change from centralized control strategy with MG centralized controller (MMC) distributed control strategy that only needs sparse communication links. Furthermore, timely and reliable communication are more crucial for MG operation considering the small inertia in the system.

The concept of dynamic microgrid reduces REG's effect on a whole power system, lowering the electricity price. Because the investment in most renewable generation doesn't contain fuel cost such as PV and wind power plant, while almost all traditional synchronous generators are energized by fossil fuel. Because of all these advantages brought by MGs, more and more researchers are focused on this area and they are trying to achieve reliable and optimal operation of the MGs.

1.2 Static Microgrid VS Dynamic Microgrid

Traditionally, MGs are regarded as a single group of DGs and loads that connect together and form a small sub-system with clearly defined electric boundaries and fixed PCC. When there are multiple MGs in the system, they are generally treated as independent structure that could not connect to each other directly except they are embedded in the system through utility companies. Since the number of DGs are increasing at a very high rate, more and more MGs are integrated into the system. With intelligent inverters, fast communication and advanced hardware, a new control strategy that connects the MGs was developed.

The concept of Nested MG [15, 16] or Interconnected MG was developed to link multiple MGs. Compared to traditional MGs that operate individually and only interact with the system via single PCC. Nested MG may have many PCC. A general structure of nested MG developed by Department of Energy (DOE) is displayed in Figure 1-2.

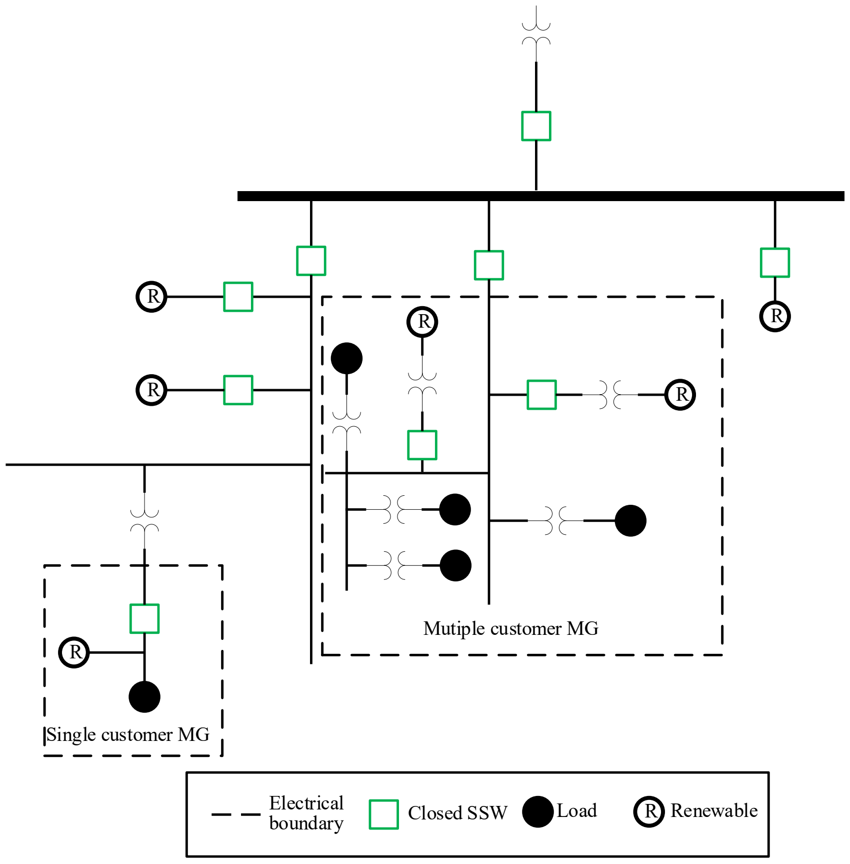


Figure 1-2: Structure of nested MG.

The cooperation between each nested MG is enabled in both islanded mode and grid-connected mode. It can provide the utility companies with the most efficient generation method. The generation mix of interconnected MGs could improve the system's economic performance and operational benefits. The reliability, resilience and robustness of traditional customers and MG customers will all be improved because of the advanced control strategy [17, 18]. System reliability could be improved through power exchange between MG with deficit generation and MG with surplus power generation. And there are mainly two challenges that need to be addressed.

The first one is that the complexity of controlling and coordination between various MGs significantly increases as the number of MGs increases. Old time-consuming control strategy needs to be replaced with more efficient control strategy. The second one derives from managing multiple PCC. It will bring more challenges to the planning and operation of system protection. Meanwhile, interaction between nested MGs requires careful planning to coordinate the control efforts.

Previously, all these nested MGs or interconnected MGs are all static MGs since their electrical boundaries are fixed during the planning stage. Distribution system operators used to form static microgrids (MGs) with static electrical boundaries. Because of the high penetration level of REG at the distribution level, simply forming static MGs cannot guarantee the supply-demand power balance in small sub-systems, especially when no power is transferred from the main grid (under islanding operation). Under this circumstance, the concept of dynamic MG was brought to the industry using smart switches [19-21]. A dynamic MG is defined as "a MG with flexible boundaries that can expand or shrink to keep the balance between generation and load at all times." The boundaries of these dynamic MGs could be substation breakers and smart switches. The structure of static MG is fixed during the operation, while the topological structure of dynamic MG could be frequently varied as per request from system operator. For instance, the structure change of dynamic MG could be caused by integration of another renewable generation or restoration of an islanded system during a natural disaster.

The general differences between static MG and dynamic MG are represented in Figure 1-3.

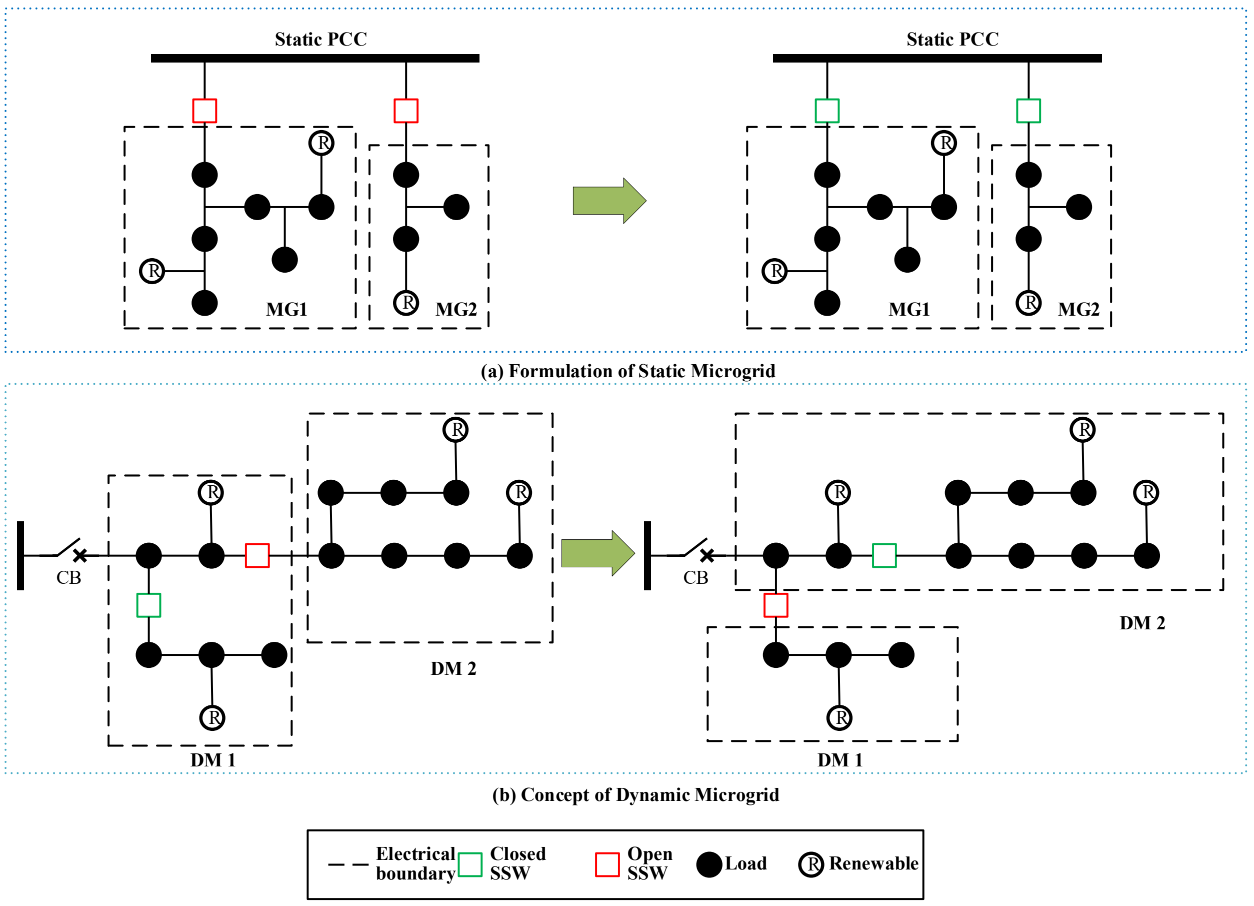


Figure 1-3: Difference between static MG and dynamic MG.

As displayed in Figure 1-3(a), the connection between two MGs is achieved through static PCC and initially static breaker at PCC is open. On the right side of Fig 1.3(a), the two MGs are connected since the static breaker at PCC is closed. A typical structure of dynamic MG is displayed in Fig 1.3(b), where DM in this Figure means dynamic MG and CB means circuit breaker. The topological change between left side and right side of Fig 1.3(b) is achieved through the closing and opening operations of the smart switches. And their connection points would also change. On the left side of Fig 1.3(b), the system is divided into the left side and the right side. On the right side of Fig 1.3(b), it was divided into upper part and lower part. Such transitions would bring multiple benefits to distribution system.

For example, suppose there is a fault upstream and the system is under islanded operation. Grid forming IBR (e.g., BESS) could be used to provide voltage at nominal frequency and form multiple MGs with other grid following DGs and loads. In this sense, multiple dynamic MGs could provide power supply to critical loads in the system. And they can also change their electrical boundaries [22, 23] to maintain as much load as possible. Another example is that suppose some MGs in the system has a lack of power generation while their adjacent MGs have enough reserve. A possible solution to balance reserve between different MGs is the topological transition of dynamic MGs in the system.

The optimal way to manage the distribution-level network with a high-level penetration of renewable energy is to separate the distribution system into small sub-systems. With dynamic MG schemes, the islanded distribution system can maintain a power balance in a real-time fashion with a low computational burden. Thus, the complexity of controlling and managing both grid-connected and islanded modes significantly decreases. Furthermore, the energy storage systems should be considered to compensate for the shortage brought by wind and solar when partitioning MGs because of their intermittent intermittency nature of the REG [24, 25].

1.3 Research Topic

Since there are mainly two operation modes of dynamic MGs, it is reasonable to separate the research topic into two scenarios. The difference between these two scenarios is the grid-connected mode MG can exchange power with the main grid considering the intermittency nature of renewable energy. Thus, different constraints and objective functions should be considered for these two scenarios.

With a grid-connected mode MG, the power balance is not the main objective since we can get enough power from the main grid. But if we require self-adequacy of each MG, we can still put power balance for each MG as a constraint. The more important aspect is that suppose there is a contingency in the upper stream, and we need to disconnect the whole system from the main grid, a smooth transition from a centralized (grid-connect MG) power system to a decentralized (islanded MG) system with several sub-systems is of great importance. Since a smooth transition would minimize the consequence brought by disconnection from and reconnection to the main grid.

The main objective of islanded mode is to enhance MGs' response to unexpected disturbances on the grid and secure supply to critical loads. The priority of islanded mode is to secure as much critical load as possible. After that, we will consider providing power supply to non-critical loads. In order to maintain the power balance under islanded mode, sometimes, load shedding is inevitable if there is a shortage of power supply. After the load shedding process, it is crucial to maintain power balance within each sub-system through topological change of dynamic MGs, which corresponds to the nature of dynamic MGs.

1.4 Organization of the Dissertation

In the next three chapters, it will mainly discuss three separation methods that targeted at three objective functions under grid-connected mode. Chapter 3 will first address the load shedding problems. In the meanwhile, the objective function is to minimize the interruption cost. Chapter 4 will extend the concept to economic load dispatch. Chapter 5 discusses the conclusions of this work and ideas for future work.

2. Separation Methods Under Grid-Connected Mode

When the distribution network is connected to the grid, there is enough power reserve available from the grid. Because we designed the generator to have plenty of headroom in the planning stage. Many researchers in this area are trying to find the optimal partitioning method for intelligent distribution networks [26-33], and plenty of objective functions have been put forward within the last decade. The most renowned objective functions in this field include minimizing power exchange and reducing power loss on distribution lines.

In section 2.1, we will explore the significant requirement for a smooth transition from a centralized system to a decentralized system: the minimization of power exchange. In section 2.2, we will further expand the utilization of the algorithm introduced in section 2.1. Moving on to section 2.3, we will enhance a traditional algorithm to reduce power losses when implementing separation method of dynamic microgrids.

2.1 Minimization of Power Exchange

When the system is running with sufficient reserve, the disturbance from renewable energy could be mitigated through large synchronous generators. However, in some cases, it may be better for the microgrid to operate independently. For example, if there is a persistent disturbance in the main grid, separating the microgrid is a better option for the reliability of each system. Similarly, any disturbance within the microgrid may interact with the primary grid (or another microgrid) and separating the microgrid may solve the issue. In order to prepare for the separation of MGs, we need to guarantee a smooth transition in advance.

In this research, we consider the minimization of sum of power exchange as the prerequisite for a smooth transition because it is reasonable to disconnect the lines with less power flow. Hence, we can get a smooth transition from a centralized distribution network to a decentralized intelligent distribution network.

2.1.1 Modified K-way Partitioning

Derived from graph theory, the k-way partitioning [34-36] was introduced to minimize the sum of edge weights that connect different partitions. If we select absolute value of active power flow as the edge weights, we can ensure that the sum of power exchange between different partitions is minimized. Since our primary concern is the minimization of power exchange, we do not need to consider the direction of the power flow. Using the absolute value of active power flow as edge weight would save us a lot of effort. Since the k-way partitioning is derived from graph theory, the first thing we need to do is to project the distribution network to a graph.

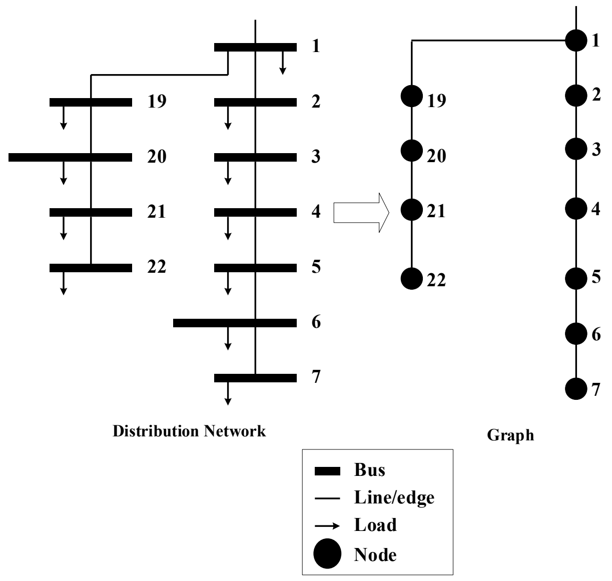


Figure 2-1: Projection from distribution network to graph.

Figure 2-1 illustrates the projection from the distribution network into a weighted graph. As the weight on each edge does not have a direction, it can also be treated as an undirected graph. If we take a closer look at the distribution network, we may find that the distribution network is similar to a weighted graph. We can treat each bus in the distribution network as a node in a graph. And each distribution line is an edge in a graph. As for the weight for each edge, we can choose different physical values according to our research topics, such as active power flow and resistance. Now we have the essential elements for a weighted graph; all we have to do is to apply the modified k-way partitioning to tackle our problem.

In order to implement the k-way partitioning algorithm, the first thing we need to do is define the value of *k*. The value *k* in k-way partitioning means the number of partitions we want to design. There are mainly two constraints considered in this research when defining value of *k*.

$$N_{p,max} \leq \min\,(N_{SG}, \frac{N_G}{2}) \tag{1}$$

where $N_{p,max}$ is the maximum number of partitions, N_{SG} is the number of synchronous generators (SGs) in the system, and N_G is the number of generators. The first term means that the least SG in one partition is equal to one based on the assumption that only SG can provide voltage at a nominal frequency. The second term mainly consider the reliability issue.

In order to get a better understanding of weighted graph, there are three matrices we need to introduce first. Degree matrix, adjacency matrix and Laplacian matrix. Suppose a graph is given by G=(V,E), where V represents n vertices, and E represents all edges. The edge weight a_{ij} is assigned for connected vertices i and j, Suppose the edge weight in Figure 2-1 is equal to 1 and degree for each vertex in graph is calculated:

$$d(i) = \sum_j a_{ij} \tag{2}$$

where *j* represents all vertices that connect to *i* and a_{ij} is the edge weight on the connection line. Based on this equation, we could get the degree for each vertex in weighted graph. The degree matrix is a diagonal matrix with its diagonal elements equal to the degree for each vertex. The calculated degree matrix is displayed in Table 2-I.

Table 2-I: Degree Matrix

Vertex	1	2	3	4	5	6	7	19	20	21	22
1	2	0	0	0	0	0	0	0	0	0	0
2	0	2	0	0	0	0	0	0	0	0	0
3	0	0	2	0	0	0	0	0	0	0	0
4	0	0	0	2	0	0	0	0	0	0	0
5	0	0	0	0	2	0	0	0	0	0	0
6	0	0	0	0	0	2	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0
19	0	0	0	0	0	0	0	2	0	0	0
20	0	0	0	0	0	0	0	0	2	0	0
21	0	0	0	0	0	0	0	0	0	2	0
22	0	0	0	0	0	0	0	0	0	0	1

The adjacency matrix is a matrix that reveals the connection relations between different vertices. The element a_{ij} in adjacency matrix means there is a line connect vertex *i* and vertex *j* and a_{ij} is the edge weight. The adjacency matrix is also a symmetric matrix, since the edge weight is same whether we measured from vertex *i* or measured from vertex *j*. The adjacency matrix calculated for Figure 2-1 is:

Table 2-II: Adjacency Matrix

Vertex	1	2	3	4	5	6	7	19	20	21	22
1	0	1	0	0	0	0	0	1	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0
3	0	1	0	1	0	0	0	0	0	0	0
4	0	0	1	0	1	0	0	0	0	0	0

5	0	0	0	1	0	1	0	0	0	0	0
6	0	0	0	0	1	0	1	0	0	0	0
7	0	0	0	0	0	1	0	0	0	0	0
19	1	0	0	0	0	0	0	0	1	0	0
20	0	0	0	0	0	0	0	1	0	1	0
21	0	0	0	0	0	0	0	0	1	0	1
22	0	0	0	0	0	0	0	0	0	1	0

After we calculate degree matrix and adjacency matrix, next matrix we need to calculate is Laplacian matrix, which is calculated by:

$$L = D - A$$

(3)

There are several features of the Laplacian matrix we can use to verify if the result is correct:

1) Since all edge weights are the absolute value of active power flow, L can only have real eigenvalues.

2) L is positive semidefinite.

3) The smallest eigenvalue is λ_1=0, and the elements in the corresponding eigenvector are all identical.

If we select absolute value of active power flow as edge weights, the process to calculate degree matrix, adjacency matrix and Laplacian matrix stays the same. The only difference is that originally in the previous example, we assumed edge weight is equal to 1. Now we have the essential elements for a weighted graph, the detailed process for k-way partitioning is displayed in Figure 2-2.

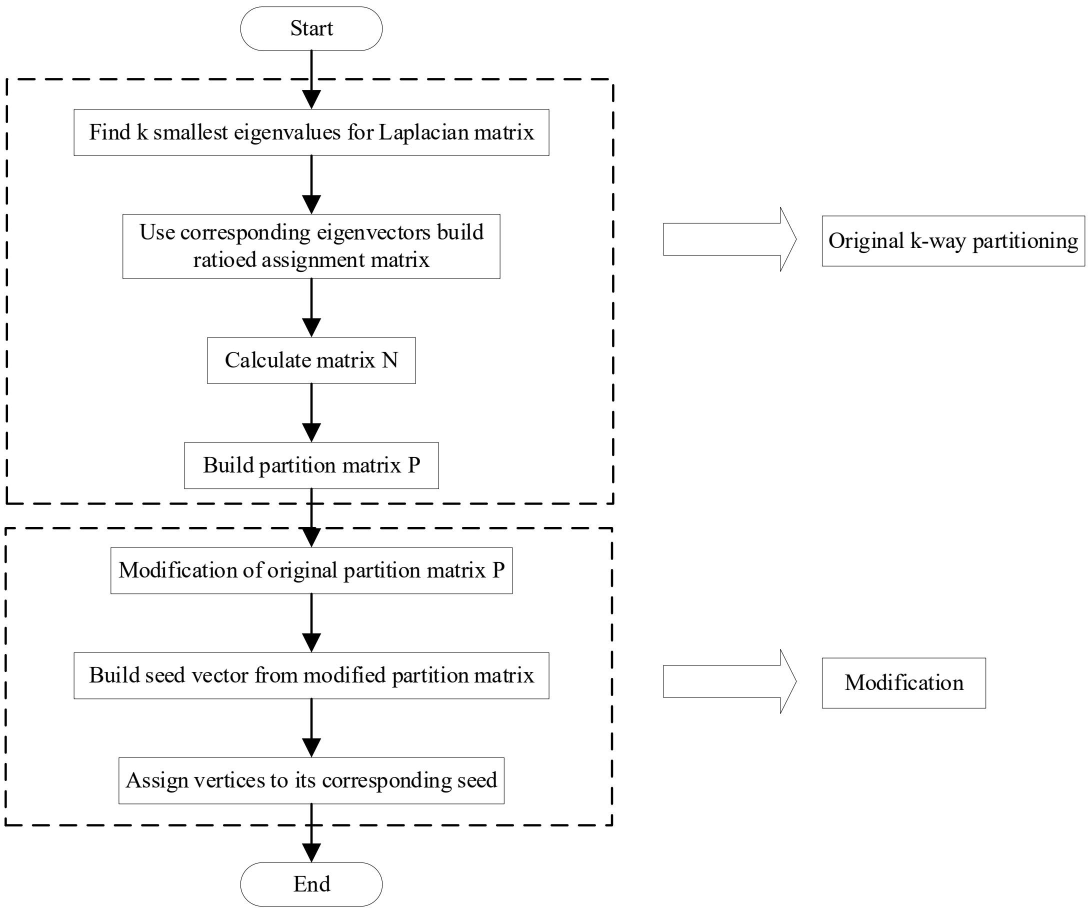

Start
Find k smallest eigenvalues for Laplacian matrix
Use corresponding eigenvectors build ratioed assignment matrix
Calculate matrix N
Build partition matrix P
Original k-way partitioning
Modification of original partition matrix P
Build seed vector from modified partition matrix
Assign vertices to its corresponding seed
Modification
End

Figure 2-2: Detailed process of modified k-way partitioning.

The initial step of k-way partitioning is to find k smallest eigenvalue and eigenvector pairs in Laplacian Matrix. Let [λ_1, λ_2, λ_3, …, λ_k] represent k smallest eigenvalues, their corresponding eigenvectors are believed to provide an embedding from our weighted graph's vertices to a k-dimensional subspace. The corresponding eigenvectors [x_1, x_2, x_3, …, x_k] that form ratioed assignment matrix X is:

$$X = [x_1, x_2, x_3, \dots, x_k] = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix} \tag{4}$$

where *X* is ratioed assignment matrix, Matrix *X* is an *N*×*K* matrix with *n* rows and *k* columns. Each row in *X* represents a vertex in our weighted graph. Only in this way can we map a weighted graph to a k-dimensional subspace.

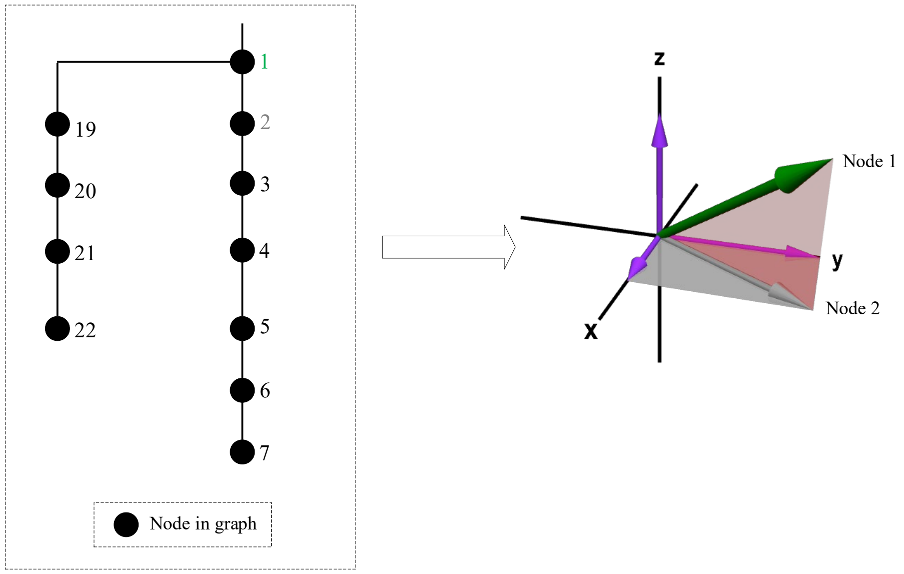


Figure 2-3: Mapping of node 1 and node 2 in a weighted graph to a 3-dimensional subspace.

Figure 2-3 gives the example of mapping 2 vertices (node 1 and node 2) in the graph to a 3-dimensional subspace. In the k-way partitioning algorithm, the value of '*k*' also defines the number of dimensions we are mapping. The next step is to calculate matrix *N*.

$$N_{ii} = \frac{1}{\sqrt{\sum_{h=1}^{k} x_{ih}^2}} \tag{5}$$

Apparently, matrix *N* is a diagonal matrix built from *X*. Its diagonal elements are reciprocal of the norm of each row in *X*. Now we can build the original partition matrix *P*.

$$P_{ori} = NXX^T N \tag{6}$$

where p_{ij} in the original partition matrix represents the cosine of the angle between two row-vectors: row_i and row_j in *X*. It can reflect how close these two row-vectors are in the k-dimensional subspace. Since these row vectors represent two vertices in a weighted graph through the mapping process, the element p_{ij} also represents a measure of distance between vertex i and j in a weighted graph. Thus, this partition matrix measures how close the vertices in the weighted graph are to each other.

The benefit of choosing k-way partitioning algorithm for dynamic MG is that if we select active power flow as edge weights, the resulting separation obtained can guarantee that the sum of edge weights connecting different partitions is the minimum due to the inherent nature of k-way partitioning. The second advantage is that we can achieve dynamic separation of the system, corresponding to dynamic MG's nature. Since the data we need is power flow and power flow is constantly changing in the system.

The elements in the original partition matrix represent the cosine of the angle between two row-vectors in ratioed assignment matrix X, it can be either positive or negative, while the measure of distance between vertices can only be positive. It may also cause another problem that the sum of two distances between two pairs of vertices might be smaller than either of these two distances. In this sense, we improved the algorithm by calculating the inverse of the cosine of all elements in the original partition matrix:

$$P = cos^{-1}(P_{ori}) \tag{7}$$

In this way, the sum of the distance between two vertices will only become larger and it corresponds to our common sense. The last step is to select k vertices as the seeds of k partitions. We can start by choosing the node that has the shortest electrical distance to the main grid. Let the seed vector S store the selected k seeds.

$$S = [S_1, S_2, S_3, \dots, S_k] \tag{8}$$

After we select the first seed, the next seed can be selected based on the modified partition matrix. The next vertex selected should follow the principle:

$$\arg\max_{v}(\min\{p_{vi}\}) \quad v \in [1, n], \mathrm{i} \in \mathrm{S} \tag{9}$$

where *v* represents each vertex in the weighted graph except the vertices already stored in seed vector *S*, *i* represents each vertex in seed vector *S*, and p_{vi} is the element in the modified partition matrix. In this way, we can guarantee that the newly selected seed is as far as possible from the previously stored seeds. After we find k seeds in the weighted graph, we also need to assign the rest vertices to their corresponding seeds based on:

$$\arg\min_{i}\{p_{vi}\} \quad v \in [1, n], \mathrm{i} \in \mathrm{S} \tag{10}$$

where *v* represents each vertex in the weighted graph, *i* represents each vertex in seed vector *S*, and p_{vi} is the element in the modified partition matrix. We are trying to assign the vertex to its nearest seed. After we assign vertices to their corresponding seeds, we successfully separate the distribution network into *k* partitions.

2.1.2 Merge Algorithm

There are still some issues with the *k*-way partitioning algorithm. *The first significant issue* is that it doesn't consider bus categories. The power system categorizes buses as load buses and generation buses. The traditional k-way partitioning may cluster several generation buses in one partition and lead to adjacent partitions' lack of generation. The reason is that the k-way portioning algorithm only seeks to minimize the sum of connection lines' edge weights; it will not consider the physical meaning of vertices.

The first issue causes *the second issue*. Since the adjacent partitions don't have generation, they cannot satisfy the self-adequacy requirement. It cannot guarantee that the power balance is satisfied within each partition. The proposed solution is to merge the partition that doesn't have a sufficient generation with adjacent partitions that have enough reserve [37, 38]. Since the k-way partitioning algorithm minimizes the sum of power exchange between partitions in the first place, such a merge operation would still ensure the newly generated separation method can maintain this advantage.

The key idea of the merge algorithm is to merge adjacent partitions so that the combined partition can satisfy our constraints. In this research, there are mainly three constraints considered. Power balance constraint is our priority. To provide a reliable power supply to customers, we need to guarantee that the power balance is satisfied within each partition.

$$\sum_{partitions} G \geq D \tag{11}$$

where *G* is the sum of power generation within each partition, and *D* is the power demands within each partition. The second constraint is that each partition should have at least one grid-forming resource. Since we assume a grid-forming (GFM) generator is the only resource that can provide voltage at nominal frequency. The third constraint is the power generation limit for all generators.

There are various merging approaches, but finding the optimal one necessitates using a search algorithm. There are many search algorithms in artificial intelligence. The most straightforward one is brute-force search. It enumerates all possible states regarding a given problem and chooses one with the highest score function. The score function is defined according to the requirements of the system operator. In this research, we will introduce the Breadth First Search (BFS). On the contrary to the brute-force search that enumerates all possible states regarding a given problem, BFS [39-43] solves the problem iteratively and as a result, each time, it only needs less search space than brute-force search does.

The first step of BFS is to perform k-way partitioning algorithm. Then, we generate a tree structure based on BFS. Each node in the tree represents a separation method of the distribution network. The root node in the tree structure is the original separation method obtained from the k-way partitioning algorithm. If each partition can guarantee active power balance and has at least one GFG, then the initial state (root node) is stored in a goal state set. Considering whether a newly generated state (node) is stored in the goal state set, these three constraints will be considered standard. However, the situation where the initial state (root node) can satisfy those constraints is relatively rare in the existing distribution system. Hence, the possible solution is to merge the imbalanced partitions with adjacent balanced ones. Suppose there are $N_{p,max}$ partitions in the initial state (root node), the theoretical maximum number of child nodes of the initial state is $C_{N_{p,max}}^{2}$ considering combination. However, some child nodes would never be generated because there is no direct connection between those two partitions. There are $N_{p,max}$ partitions in depth 0, whereas the nodes in depth $N_{p,max}$-2 will only have two partitions. As depth increases by 1, the number of partitions will decrease by 1 because of the merge operation.

The next step is to iteratively merge adjacent partitions to generate a new node in the tree structure. If the new state (node) can satisfy all three constraints, it is stored in the goal state set. The termination condition is that all nodes in this depth have only two partitions. In the next depth, there will be only one partition, and that is the distribution network itself.

The final step is to find the optimal solution from the goal state set. The extreme condition is that none of the nodes we have explored can satisfy these three constraints, and the goal state set is empty. Then, load shedding should be considered. Figure 2-4 illustrates the process of the merge algorithm.

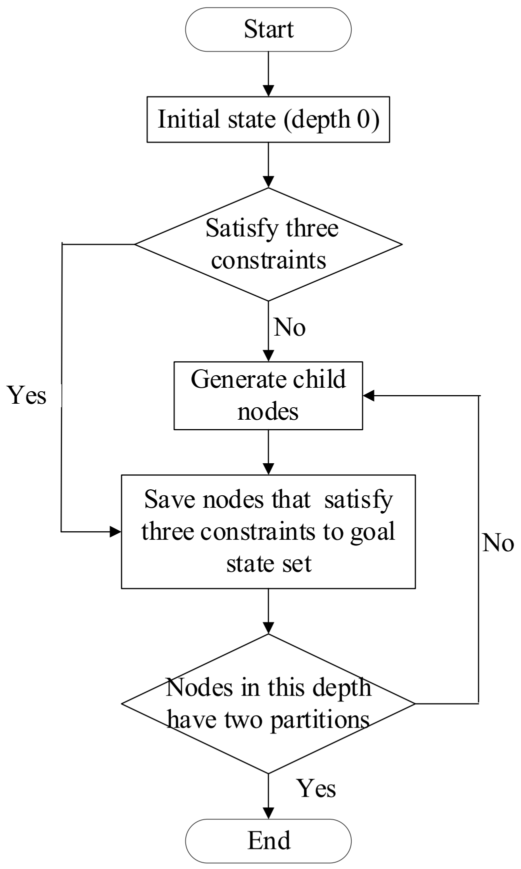

Start
Initial state (depth 0)
Satisfy three constraints
No
Yes
Generate child nodes
Save nodes that satisfy three constraints to goal state set
No
Nodes in this depth have two partitions
Yes
End

Figure 2-4: Process of merge operation.

When the algorithm terminates, if the goal state set has plenty of separation methods, the score function needs to be considered to determine which is the optimal one. Score function can be defined according to the objective function required by system operators. In this research, the score function is considered as the sum of the weighted reserve margin [44, 45].

$$Score = \sum_i W_i \cdot RM_i \tag{12}$$

where *Score* is the score calculated for the current separation method, W_i is the weight for each partition, and RM_i is the reserve margin for the corresponding partition. The sum of W_i is equal to one. The difference between generation and demand can be calculated by the reserve margin divided by the generation capacity. We can get a value for each state in our goal state set through this calculation process, and we pick one with the highest score. The weight for each partition is defined based on the weighted load.

$$W_i = \frac{\sum LP_j \cdot L_j}{\sum_i \sum LP_j \cdot L_j} \tag{13}$$

where W_i is the weight for i^{th} partition, LP_j is load importance, and L_j is load. The numerator represents the sum of the weighted load in the corresponding partition, and the denominator represents the sum of the weighted load in the whole system. In this research, we classify the system's load into three levels:

- The first level is the most critical load
- The second level is the crucial load, and
- The third level is the average (curtailable) load.

The load is still classified into different layers within each level according to frequency sensitivity coefficients. The loads with a larger frequency dependency will absorb more active power. It will need more reserve. If the system has enough reserve, it is reasonable to provide a higher weight to this load. The detailed classification approach is shown in Table I.

Table 2-III: Frequency Sensitivity Coefficients

Level	Layer 1	Layer 2	Layer 3	…	Layer M
1	$P_{1,1}$	$P_{1,2}$	$P_{1,3}$	…	$P_{1,m}$
2	$P_{2,1}$	$P_{2,2}$	$P_{2,3}$	…	$P_{2,m}$
3	$P_{3,1}$	$P_{3,2}$	$P_{3,3}$	…	$P_{3,m}$

2.1.3 Simulation With Radial Network

To verify the performance of the modified *k*-way partitioning algorithm, a case study is simulated in a modified IEEE 33-bus radial distribution system shown in Figure 2-5. Three scenarios were considered in this case study. The first one is that the system has enough reserve margin. The second one is that the system can barely keep the supply demand balance. The last one is that generation is less than demand because of loss of generation in the system.

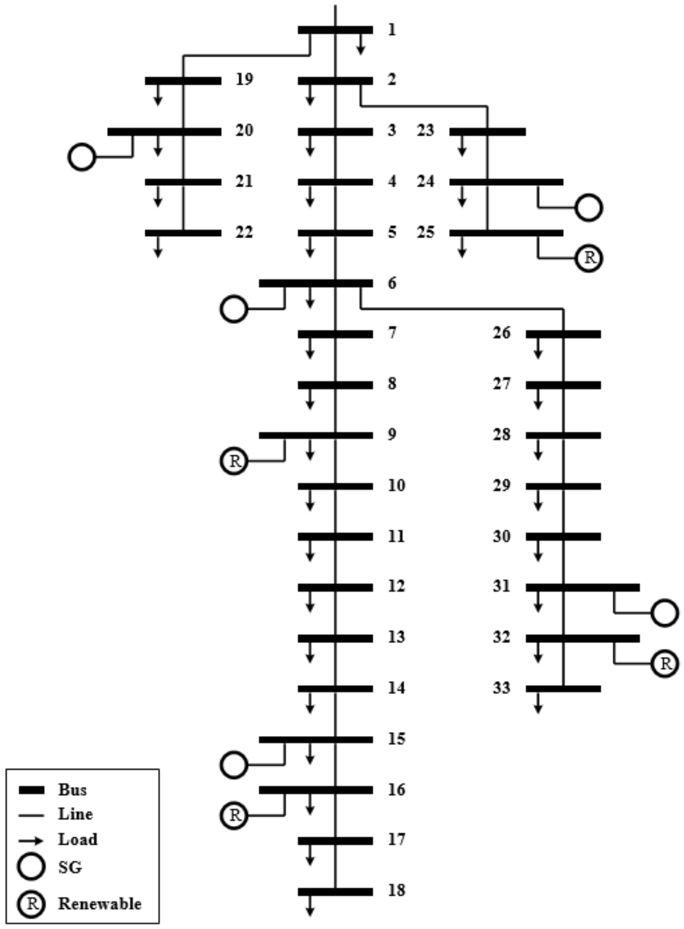


Figure 2-5: IEEE 33-bus radial distribution system.

Five diesel generators are connected to buses 6, 15, 20, 24, and 31, and four renewable resources are connected to buses 9, 16, 25, and 32. The capacities of the five diesel generators are identical to 1 MW. Since we require self-adequacy for each partition after the separation method, we can assume that the system is disconnected from the main grid and operating in the islanded mode.

Table 2-IV: Generation Capacity and Output for IEEE 33-Bus System

Bus	Generation type	Output [MW]	Capacity [MW]
6	Diesel	0.638	1.00
9	DG	0.200	0.20
15	Diesel	0.623	1.00
16	DG	0.150	0.15
20	Diesel	0.645	1.00
24	Diesel	0.642	1.00
25	DG	0.100	0.10
31	Diesel	0.634	1.00
32	DG	0.100	0.10

To separate the system into multiple dynamic Microgrids, we must determine the number of k if we implement a *k*-way partitioning algorithm. In this paper, we considered five partitions as the maximum because there are five synchronous generators in this system with the assumption that renewable resources are all grid-following resources.

Scenario 1

This scenario refers to a situation where the system has a sufficient reserve margin. It means that the sum of generation capacity is larger than the sum of loads in the IEEE-33 distribution system. In this scenario, it is probable that the original separation method generated from *k*-way partitioning algorithm can be put into the goal state set.

The first step of the proposed algorithm is to perform power flow calculations. After we obtain power flow data, the next step is to run the *k*-way partitioning algorithm. The result for the *k*-way partitioning algorithm is shown in Table 2-V.

Table 2-V: Result for *K*-way Partitioning

Partition	Nodes	Supply/kW	Demand/kW	Number of SG
P_1	**2** 3 19 20 21 22 23 24 25	2100	1480	2
P_2	4 5 6 7 26 27 **28**	1000	620	1
P_3	8 9 10 11 12 13 14 **15**	1200	665	1
P_4	29 30 31 **32** 33	1100	740	1
P_5	16 17 **18**	150	210	0

The P1, P2, P3, P4, and P5 seeds are nodes 2, 28, 15, 32, and 18, respectively. All partitions have enough reserve margin except for partition 5, and there is no synchronous generation in partition 5 because there are two synchronous generators in partition 1. The reason is that the k-way partitioning algorithm cannot differentiate between the generator and load buses, as we mentioned before. Hence, we must implement a merge algorithm to provide voltage at nominal frequency for partition 5. Since the only partition connecting to partition 5 is partition 3, depth 1 will merge these two. The merge operation in depth 1 is displayed in Figure 2-6.

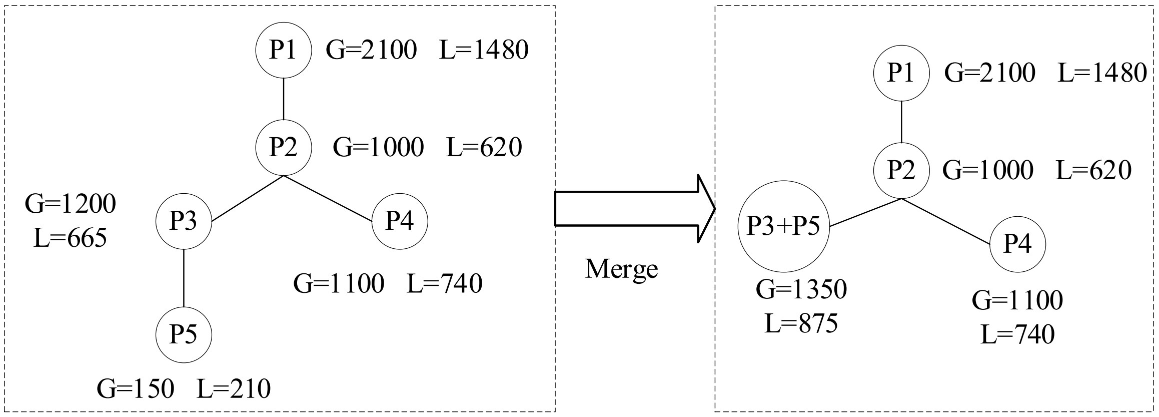


Figure 2-6: Merge operation in depth 1.

The *G* in this figure means total generation capacity within current partition and *L* means sun of load in the current partition. After we merge partition 3 with partition 5, the combined partition can guarantee that it has a synchronous generator and that the power balance is satisfied. In this sense, the current separation method is stored in goal state set. In depth 1, there will be only one separation method generated through the merge operation of partition 3 and partition 5. We can still generate numerous separation methods in depth 2 based on the merge operation. We still perform the merge operation because we want to generate as many as separation methods. In this sense, we can select the optimal separation method based on the score function.

At depth 2, there are three possible merge operations, it means that there will be three child nodes generated from separation method in depth 1, which is the parent node. The first merge operation is to merge partition 1 and partition 2.

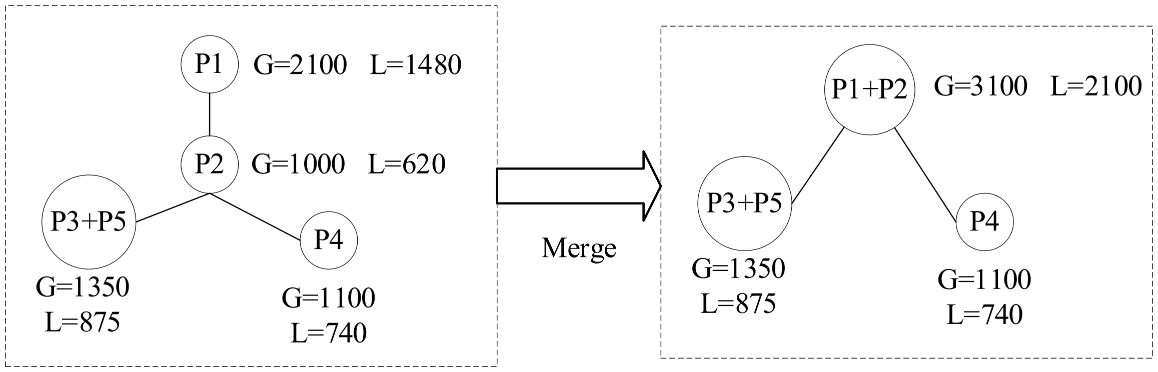


Figure 2-7: Merge partition 1 and partition 2.

As we can see, the new generated separation method from merge operation between partition 1 and partition 2 can guarantee that each partition has a synchronous generator and that the power balance is satisfied. Hence, this separation method will be stored in the goal state set. The second merge operation would be partition 2 and partition 4, which is the second child node generated from the parent node.

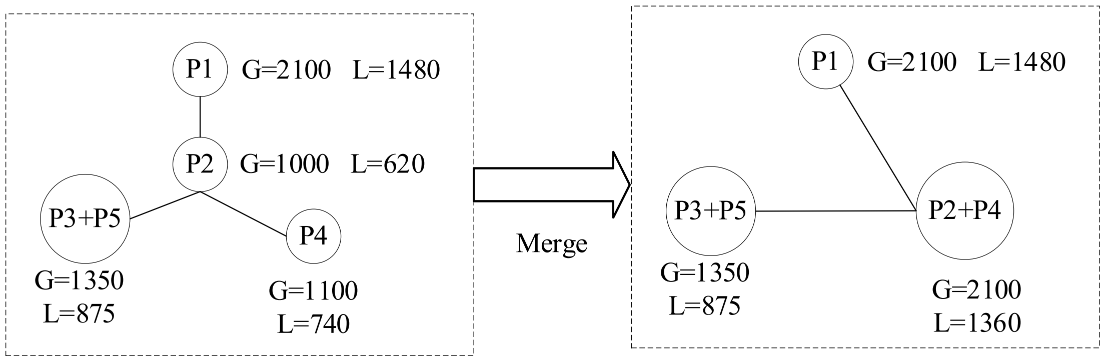


Figure 2-8: Merge partition 2 and partition 4.

Similarly, the newly generated separation method can also guarantee that there is a synchronous generator in each partition and within each partition, power balance is satisfied. Hence, this separation method will be stored in the goal state set. The last possible merge operation is to merge partition 2 and partitions 3 & 5. This merge operation will generate the third child node from the parent node.

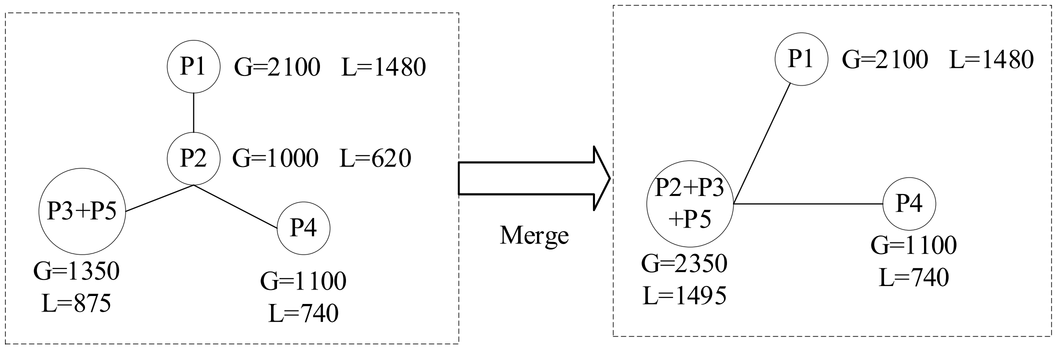


Figure 2-9: Merge partition 2 and partition 3 & 5.

We can still find out that the newly generated separation method can maintain the constraints. As a result, this third child node will be stored in the goal state set. We may find out that if the parent node can satisfy the constraints, all generated child nodes can satisfy the constraints, and they will all be put in the goal state set. In summary, there will be three nodes in-depth 2.

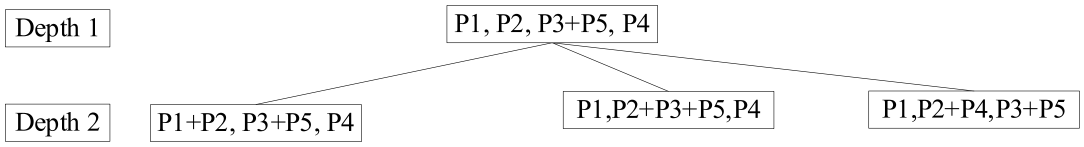


Figure 2-10: Three child nodes in depth 2.

Since there are three nodes in depth 2, there will be more child nodes generated by these three separation methods. We will start with the first node in depth 2.

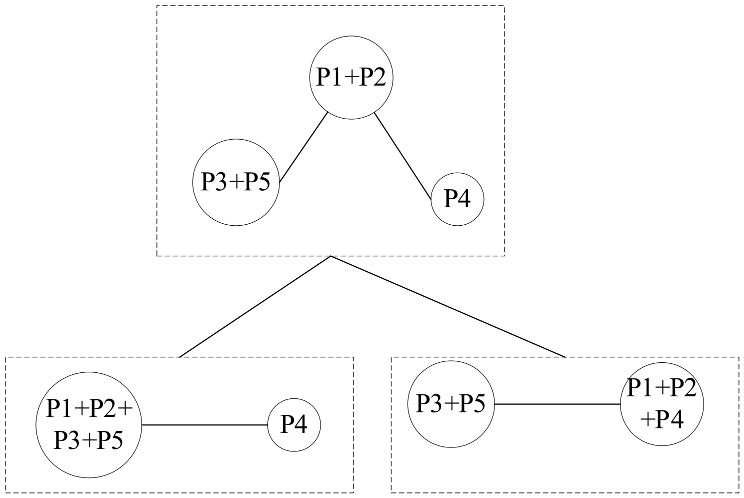


Figure 2-11: Child nodes generated from first node in depth 2.

There are two possible merge operation that could be implemented for the first node in depth 2. Hence, this node will generate two child nodes and they all can satisfy the two constraints. These two child nodes will be stored in goal state set. Now we will perform merge operation on the second node in depth 2.

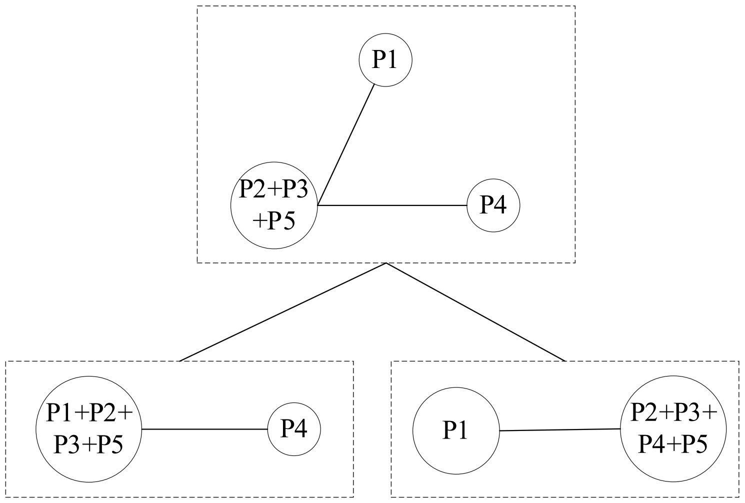


Figure 2-12: Child nodes generated from second node in depth 2.

Similarly, there will be 2 child nodes generated from the second node in depth 2 and they can all satisfy the two constraints. Another thing needs to be mentioned is that the left child node in Figure 2-12 is and the left child node in Figure 2-11 is the same. Since we cannot create duplicate nodes in the goal state set, we will only store the right child node in Figure 2-12 to the goal state set. Now we need to perform merge operation to the third node in depth 2.

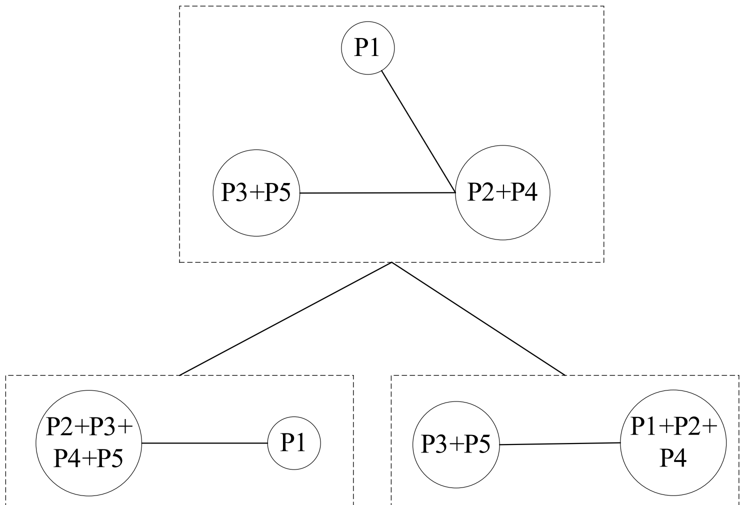


Figure 2-13: Child nodes generated from third node in depth 2.

We can find out that the left child node in Figure 2-13 and right child node in Figure 2-12 is the same. In this sense, we cannot store this node into the goal state set. The only node that can be stored in goal state set is the right child node. Now all three depths are all created and is displayed [46].

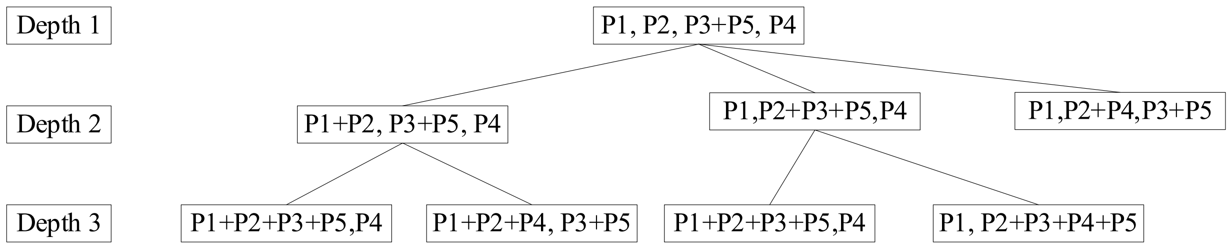


Figure 2-14: Tree structure generated for scenario 1.

Since the first node in this tree structure can satisfy our constraints, all nodes in this tree structure will be stored in goal state set. In this sense, we need to use the score function to select the optimal one as our separation strategy. Score function in this research is defined as the sum of the weighted reserve margin. We have to determine load importance among various loads. The load importance in partition one is shown in Table IV.

Table 2-VI: Load Importance in Partition One

Bus	Load	Level	K_{pf}	Load Importance
2	100	3	0.56	1.68
3	90	3	0.42	1.26
19	90	9	0.59	5.31
20	90	3	0.72	2.16
21	90	6	0.63	3.78
22	90	3	0.54	1.62
23	90	3	0.85	2.55
24	420	6	0.53	3.18
25	420	3	0.75	2.25

K_{pf} is the frequency sensitivity coefficient for the corresponding load. In the level column, Level=9 at bus 19 represents critical load, Level=6 represents crucial load and Level=3 represents average (curtailable) load. The load importance is calculated by multiplying level factor and frequency sensitivity coefficient. The final step is to calculate the weighted reserve margin for the separation methods in our goal state set. The result is shown in Table V.

Table 2-VII: Score for Separation Methods

Separation Method	**Score**
P1, P2, P3+P5, P4	485.1045
P1+P2, P3+P5, P4	727.0511
P1, P2+P3+P5, P4	652.9245
P1, P2+P4, P3+P5	634.3060
P1+P2+P3+P5, P4	**1213.0690**
P1+P2+P4, P3+P5	1153.3270
P1, P2+P3+P4+P5	997.7814

This table shows that the best score comes from the separation method that merges partitions 1, 2, 3, and 5 to form a big microgrid. So, we chose this separation method as our separation strategy at scenario 1.

Scenario 2

In this scenario, the supply can barely meet the demand. There is not much reserve margin in the system. Hence, the main goal of the separation of the network is to keep the supply demand balance. The capacity for renewable generation stays the same, but the capacities for SGs will be changed to their output powers in the first scenario. Since it's not practical to form a stable simulation result when generation is equal to load in PSCAD, we use the power flow data from the first scenario. Same as the first scenario. The first step is to implement the *k*-way partitioning method and the corresponding result is shown in Table VI.

Table 2-VIII: Supply and Demand in Scenario 2

Partition	Nodes	Capacity/kW	Demand/kW	Number of SG
P_1	2 3 19 20 21 22 23 24 25	1387	1480	2
P_2	4 5 6 7 26 27 28	638	620	1
P_3	8 9 10 11 12 13 14 15	826	665	1
P_4	29 30 31 32 33	734	740	1
P_5	16 17 18	150	210	0

From this table, we can find that several partitions cannot satisfy the power balance. Besides, there is no SG in partition 5. The initial step to form a tree structure is still to merge partition 3 and partition 5.

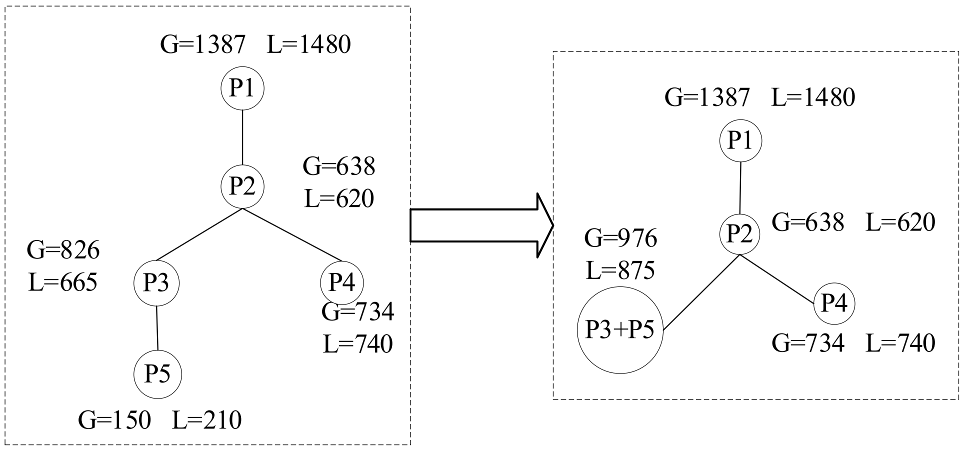


Figure 2-15: Merge of partition 3 and partition 5 in scenario 2.

After the first merge operation, we can find the only partition that cannot satisfy the supply and demand balance is partition 1. Since there is only one partition that connects to partition 1 and has enough reserve, the merge operation at depth 2 will merge partition 1 and partition 2.

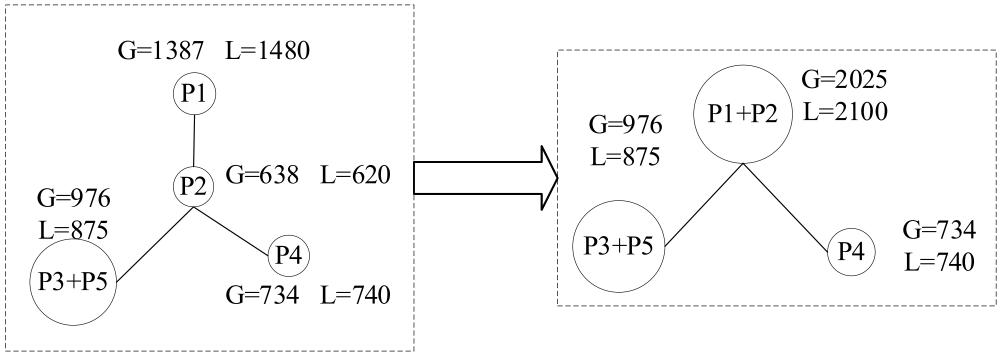


Figure 2-16: Depth 2 at scenario 2.

Even we merge partition 1 and partition 2, it still cannot satisfy power balance. Hence, we still have to implement merge operation. There are only two merge operations that could be implemented at depth 3. In this sense, the tree structure for the merge operation that generated at scenario 2 is:

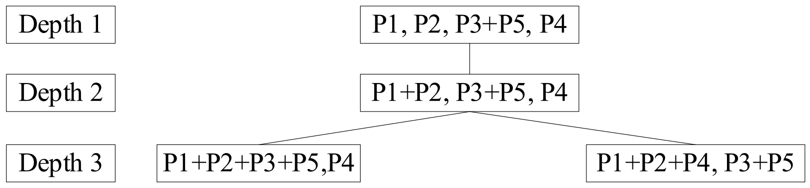


Figure 2-17: Tree structure in scenario 2.

Since the only separation method that can guarantee power balance in this tree structure is the first node at depth three, we will select this separation method as our separation strategy at scenario 2.

Scenario 3

It is expected that the test system will face a sudden trip of SGs. In this scenario, the most important requirement from a system operator is to secure the power supply for critical loads. We suppose there is a sudden trip of generators at bus 20 and bus 31. To maintain power balance within the system, load shedding is inevitable. The detailed load shedding process will be illustrated in chapter 3. After we implement the load shedding strategy, we can still apply the modified *k*-way partitioning algorithm and merge operation to provide separation methods to system operators.

2.1.4 Simulation with Mesh Network

Most papers published in this area only focused on radial distribution network because of its simplicity when dealing with case studies. For example, suppose we need to separate a radial network into *k* partitions, and we want to minimize the sum of power exchange. The easiest way is to choose *k*-1 edges with the smallest active power flow as our connecting edges based on the nature of the radial network. Contrary to the radial network, simply choosing *k*-1 edges sometimes cannot separate the mesh network into *k* partitions. Simply choosing edges with the smallest power flow sometimes cannot separate the system into *k* partitions. In this sense, *k*-way partitioning would solve the separation problem when considering minimizing power exchange. This research will employ the IEEE-39 bus system, to validate the effectiveness of modified *k*-way partitioning.

Assume generators placed on bus 30 are replaced with renewable generation. And assume the power output of the renewable generation is the same as that of the synchronous generator originally put on bus 30 during the given research period. In this research, we assume that each line has a smart switch, and the system operator can control each switch's status.

The initial step of the proposed approach is to perform a power flow analysis and get the active power flow from PSLF. After that, we need to build an adjacency matrix, degree matrix and Laplacian matrix. Since there are nine synchronous generators in the system and there are ten generators together in the system, the biggest number of *k* is equal to five. In this research, we choose *k* equal to four.

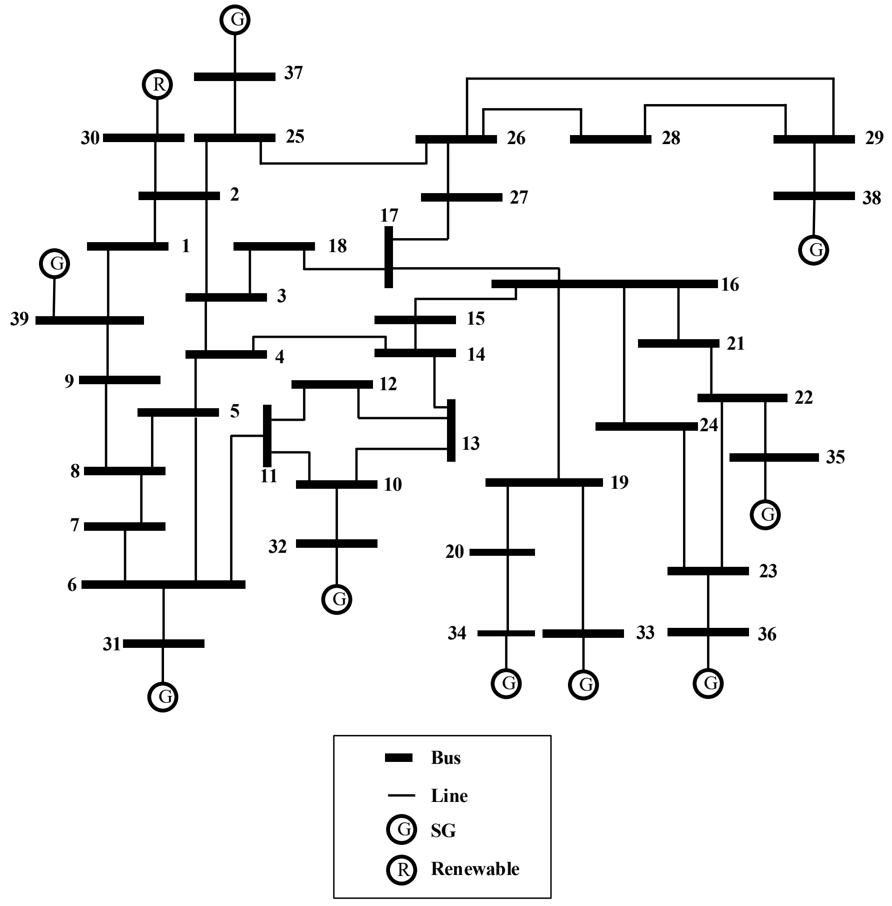


Figure 2-18: IEEE 39-Bus system.

Through the calculation of *k* smallest eigenvalue and eigenvector pairs of Laplacian matrix, we can map the weighted graph to a *k*-dimensional subspace. Then we have to calculate the modified partition matrix and try to find seeds for each partition. The result of seeds and their corresponding vertices is display in Table 2.

Table 2-IX: Results for IEEE 39-Bus System

Seed	Partition
1	1, 2, 3, 25, 26, 27, 28, 29, 30, 37, 38, 39
12	12
32	4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 31, 32
36	15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 33, 34, 35, 36

As we can see, the second partition only has one bus in the partition. The reason behind this is that *k*-way partitioning only seeks to minimize the sum of edge weights, it cannot guarantee that each partition will contain similar amounts of vertices. But if we take a closer look at this separation method, we can find that bus 12 is surrounded by third partition. It is natural to merge the second partition and third partition.

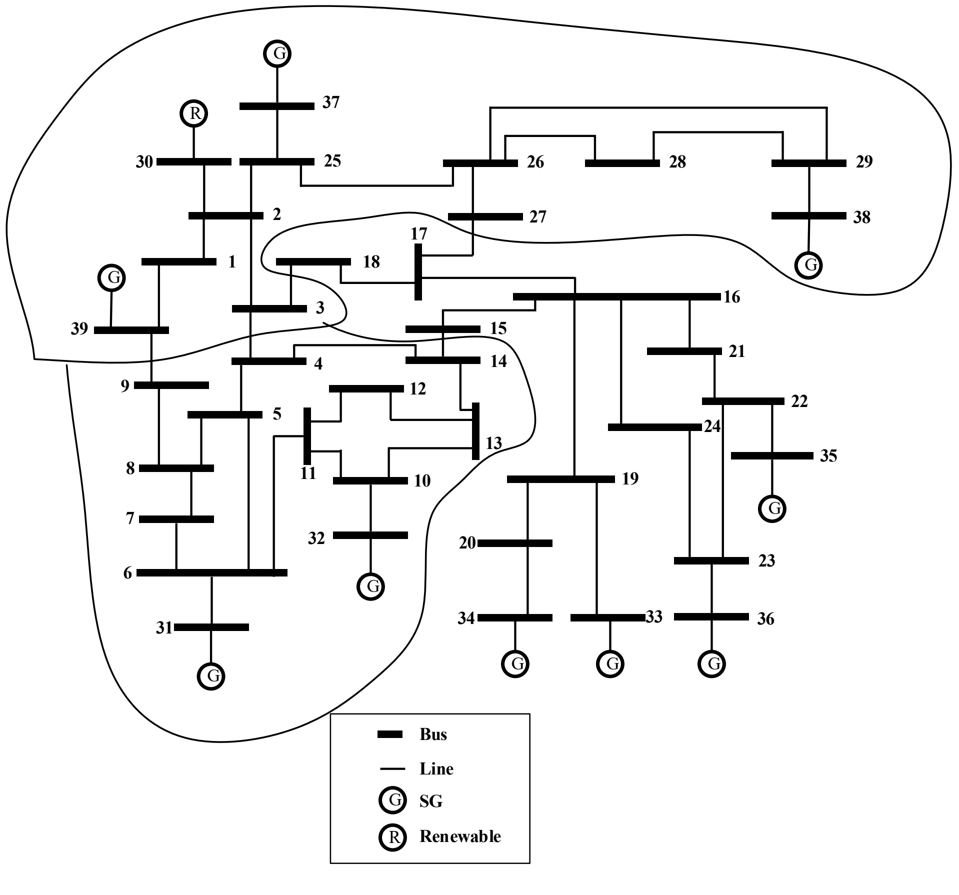


Figure 2-19: Separation method for IEEE 39-Bus system.

The newly generated separation method is displayed in this figure. Since we have minimized the sum of edge weights (active power flow) in the first place, the merge operation will keep sum of edge weights minimized. The newly generated three partitions can also satisfy the constraint that there should be at least one grid-forming generator in each partition. The three partitions can also satisfy the power balance.

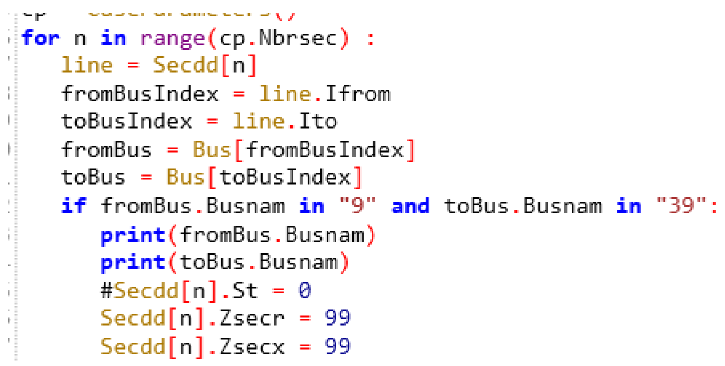
The whole process of modified *k*-way partitioning starts from the load flow analysis until the separation of the system was integrated using Python in PSLF. In that way, every time there is a sudden change in the system, we need to quickly respond to that change and generate a new separation method. All we have to do is to click the run button and generate the separation method we need. Now, we have to verify that the sum of power exchange is minimized. The first seven smallest absolute value of active power flow in the IEEE 39-Bus system is displayed.

Table 2-X: Smallest Absolute Value of Active Power Flow

Connecting Edge	From Bus	To Bus	Absolute Value of Power Flow/MW
1	17	27	11.5
2	9	39	12.1
3	14	15	30.4
4	3	18	34.1
5	16	24	42.7
6	22	23	42.7
7	3	4	75.2

Based on our modified *k*-way partitioning algorithm, the selected edges for the three partitions are the first, second, third, fourth, and last row in table. We probably won't separate the system into three partitions if we select the fifth and sixth row instead of the last row in table. The algorithm did not choose the fifth and sixth rows because of the compromise between minimization of the sum of connecting edges' weights and successful separation of the mesh network.

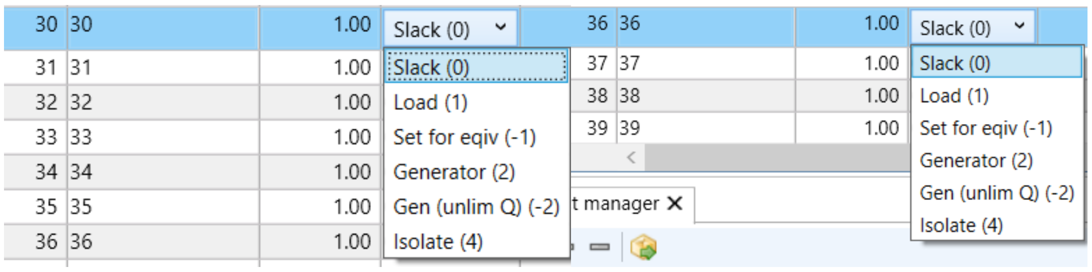
Now, we can implement steady-state analysis in PSLF. In the initial version of PSLF, we had to create the single-line diagram, manually disconnect the connection lines, and run the power flow analysis to check for bus voltage violations or line limit violations. With the newer version of PSLF, we could just change the code to disconnect those lines and select a slack bus in each a partition.



```
for n in range(cp.Nbrsec) :
   line = Secdd[n]
   fromBusIndex = line.Ifrom
   toBusIndex = line.Ito
   fromBus = Bus[fromBusIndex]
   toBus = Bus[toBusIndex]
   if fromBus.Busnam in "9" and toBus.Busnam in "39":
      print(fromBus.Busnam)
      print(toBus.Busnam)
      #Secdd[n].St = 0
      Secdd[n].Zsecr = 99
      Secdd[n].Zsecx = 99
```

Figure 2-20: Disconnect the line between bus 9 and 39 in PSLF.

To conduct a steady-state analysis in PSLF, it is necessary to choose a slack bus after disconnecting those lines and separating the system into three partitions. In the original IEEE 33-Bus system, bus 31 was designated as the slack bus. Therefore, we can continue to use bus 31 as the slack bus for its respective partition. For the other two partitions, we randomly selected two additional slack buses: bus 30 and bus 36.



30	30	1.00	Slack (0)
31	31	1.00	Slack (0)
32	32	1.00	Load (1)
33	33	1.00	Set for eqiv (-1)
34	34	1.00	Generator (2)
35	35	1.00	Gen (unlim Q) (-2)
36	36	1.00	Isolate (4)

36	36	1.00	Slack (0)
37	37	1.00	Slack (0)
38	38	1.00	Load (1)
39	39	1.00	Set for eqiv (-1)
			Generator (2)
			Gen (unlim Q) (-2)
			Isolate (4)

Figure 2-21: Selection of slack bus in PSLF.

Since the original input for modified *k*-way partitioning is active power flow, and in reality, power flow is constantly changing, we can implement the algorithm in a real-time fashion. The assumption under this operation is that SSWs in each distribution line and system operator can control each line's status according to the new separation method. We assumed that we update the dynamic MG structure in a real-time fashion at a similar frequency with which grid condition is changed, which corresponds to the nature of the dynamic microgrid.

2.2 K means Clustering for Separation

In the preceding section, we discovered that the modified k-way partitioning defines the measure of distance between buses through the projection from weighted graph to *k*-dimensional subspace, which is represented by the values in the modified partition matrix. Regarding the separation phase, in addition to identifying seeds for each partition and assigning buses to their respective seeds, an alternative method exists to accomplish this separation.

Because the modified *k*-way partitioning algorithm does not distinguish between generation and load buses and cannot ensure supply-demand balance within some partitions, we must seek an alternative algorithm to differentiate between generator and load buses while maintaining power balance within each partition. To meet this criterion, the *K*-means clustering algorithm was employed [47-49].

The advantage brought by *K*-means clustering algorithm is that it can update clusters and centroids according to user-defined distance. The distance between nodes in the distribution network can reflect the requirements of the system operator. In this research, the distance between buses has already been calculated through the calculation of modified partition matrix.

2.2.1 K Means Clustering

Traditionally, *K*-means clustering algorithm was utilized to divide a set of two-dimensional data points. It uses Euclidean distance to decide which two points has shorter distance:

$$D_{Eu}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{14}$$

where (x_1, y_1) and (x_2, y_2) are two data points in two-dimensional space. The value of k means the number of clusters. The algorithm will first randomly select k points as initial centers and use Euclidean distance to group the rest of the points into different clusters. And then the algorithm will calculate the mean of the x coordinate and y coordinate of each cluster as the new center for the next iteration. The termination policy for the iteration process can be classified into two categories. The first one is that the distortion or cost function finds the minimum. The second one is that the difference between two iterations is small enough. Under two-dimensional scenario, the distortion function can be defined as:

$$D = \sum_{clusters} \sum_{x_i \in C} D_{Eu}(x_i, c_j) \tag{15}$$

where x_i is data point in current cluster C, and c_j is the center for the current cluster. The distortion function can indicate the effect of the current clustering approach; smaller D will reflect more compact clustering results.

Suppose there are six random points in the 2-D frame and their coordinates are: 1-[1, 2], 2-[1.5, 1.8], 3-[5, 8], 4-[8, 8], 5-[1, 0.6], 6-[9,11].

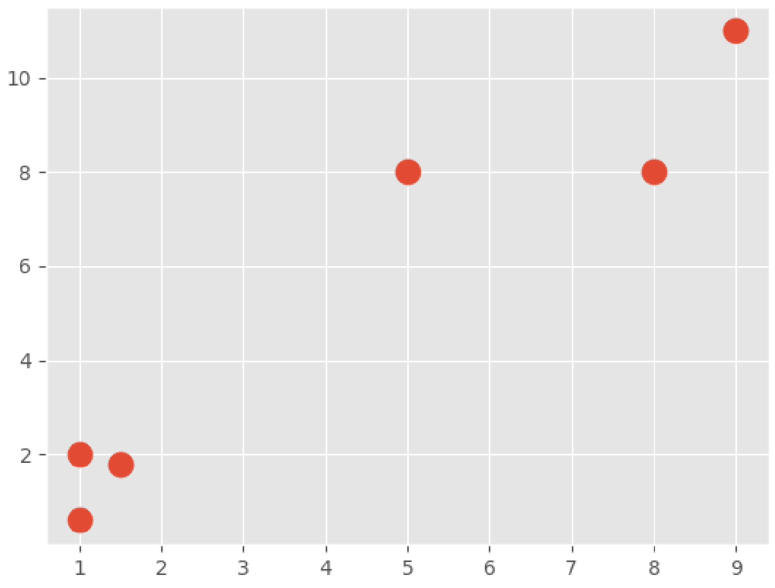


Figure 2-22: Six points in 2-dimensional space.

If we choose the initial centroids as [1, 0.6] and [9, 11], it becomes evident that these six points should be divided into two clusters. The first cluster comprises points 1, 2, and 5, while the second cluster consists of points 3, 4, and 6. The distortion calculated using this separation method is 10.86227766.

Now we need to find the centroids for the next iteration. Since the first cluster contains points 1, 2, and 5, the new centroid for this cluster is $[\frac{1+1.5+1}{3}, \frac{2+1.8+0.6}{3}]$.

We can apply the same process to the second cluster to find its new centroid. After that, we must recalculate the distortion based on the newly generated centroids. The results of the five iterations of *K*-means clustering on these six points are listed below.

Table 2-XI: Five Iterations for Six Points

Iteration	Centroid 1	Centroid 2	Distortion
1	1, 0.6	9, 11	10.8623
2	1.167, 1.467	7.333, 9.0	8.256
3	1.167, 1.467	7.333, 9.0	8.256
4	1.167, 1.467	7.333, 9.0	8.256
5	1.167, 1.467	7.333, 9.0	8.256

The algorithm quickly converges at the third iteration, corresponding to the fast convergence nature of K-means clustering. Another reason is that the six points can be clearly divided into two clusters.

2.2.2 Modification

In the previous discussion, traditional K-means clustering updated centroids based on the calculation of the mean of coordinates in each dimension. However, in a distribution network, even if we can map the distribution network to a weighted graph and then project the weighted graph to a k-dimensional subspace, we never assign coordinates to the buses in the distribution network. Instead, we only calculate the distances between different buses through the projection to a k-dimensional subspace stored in a modified partition matrix. Therefore, we need to find a new method for updating the centroids.

$$\underset{i}{\arg}\ min \sum_{j=1}^{m} P_{ij} \quad i \in [1, n] \quad j \in [1, m] \tag{16}$$

where i represents each bus in the distribution network, n is the number of buses in distribution system, j represents each bus in current cluster, m is number of buses in current cluster and P_{ij} is the distance in modified partition matrix. The updating strategy will find a node in the distribution network that has the minimum sum of distances with all nodes in the current cluster. In this sense, it can guarantee that through each iteration, the algorithm will reduce the distortion function, which can be expressed by:

$$D = \sum_{Clusters}(\sum_{j=1}^{m} P_{cj}) \tag{17}$$

where c is the centroid in each cluster, j is a single bus in the current cluster, and P_{cj} is the value in modified partition matrix. The distortion function is the sum of the distances between each centroid and all nodes in their respective clusters. It is evident that we want to minimize the distortion function in each iteration. The definition of the distortion function is inspired by k-way partitioning. In k-way partitioning, finding seeds and their corresponding partitions follows the logic of minimizing the sum of distances between seeds and nodes within partitions. Suppose the k seeds stored in the seed vector is $S = [S_1, S_2, S_3, \dots, S_k]$. When separating the system, we are trying to assign the node to its nearest seed. In this way, we can minimize the sum of distances (value in modified partition matrix) between seeds and nodes in their respective partitions. Based on this logic used in k-way partitioning, we can ensure that through the proposed distortion function in K-means clustering, the minimization of power exchange is achieved.

We assume the synchronous generators are the only source that can provide a voltage at grid frequency based on the assumption that DGs in the distribution network are all grid-following sources. The buses connected to synchronous generators are selected as initial centroids. In this way, it can guarantee that at least there is one SG in one cluster.

The detailed procedure of the proposed K-means clustering algorithm will be illustrated as the flowchart [50].

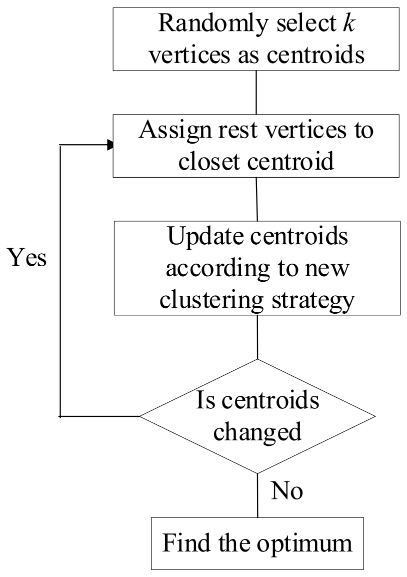

Randomly select k vertices as centroids
Assign rest vertices to closet centroid
Update centroids according to new clustering strategy
Is centroids changed
Yes
No
Find the optimum

Figure 2-23: Process of K-means clustering.

2.2.3 Simulation

The test system will still use IEEE 33-bus distribution system. It is assumed the system is isolated from the main gird. The distribution system includes five diesel generators and four renewable resources assigned to different buses. The output and capacity of the generation sources are displayed.

Table 2-XII: Generation Output and Capacity

Bus	Generation Type	Output [MW]	Capacity [MW]
6	Diesel	0.638	1.00
9	DG	0.200	0.20
15	Diesel	0.623	1.00
16	DG	0.150	0.15
20	Diesel	0.645	1.00
24	Diesel	0.642	1.00
25	DG	0.100	0.10
31	Diesel	0.634	1.00
32	DG	0.100	0.10

With the data given in the distribution system, the load flow is performed using PSCAD. The absolute value of active power flow is selected as edge weights. The detailed steps to calculate the modified partition matrix is illustrated in 2.1.1. Even it's an IEEE 33-bus system, the system's adjacency matrix and degree matrix are all 32×32 square matrix; because Bus 1 is assigned as the point of interconnection with the main system and there is no load connected when the 33-bus system is disconnected from the main grid. As a result, the Laplacian matrix is also a 32×32 symmetric matrix. After calculating the modified partition matrix, we determine the number of partitions. The system includes only five synchronous generators and the DGs are assumed as grid-following resources in this research. Thus, the maximum number of partitions is five for the test system.

Now we utilize the K-means clustering algorithm to deal with the separation part for the distribution system. At the same time, we used the values in modified partition matrix as the measure of distance between buses. Since there are five synchronous generators in the test system, we select the buses that connect to these generators as initial centroids. The following procedure is to assign rest buses to these centroids and then update centroids. The principle behind this process is to assign buses to generators with the least distance (values in modified partition matrix). When finding new centroids for the next iteration, we can compute the distortion function.

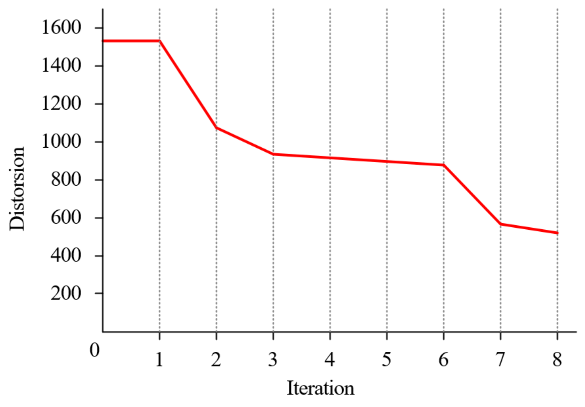


Figure 2-24: Distortion for eight iterations.

The algorithm was implemented with Python and the iteration process can be terminated after eight iterations. This is consistent with the quick convergence nature of K-means clustering algorithm. The separation result after iteration is displayed in the table.

Table 2-XIII: Separation Result for K-means Clustering

Cluster	Centroid	Nodes	Supply [kW]	Demand [kW]	Number of SG
1	20	1 2 19 20 21 22	1000	460	1
2	24	3 23 24 25	1100	1020	1
3	6	4 5 6 7 26 27 28	1000	620	1
4	15	8 9 10 11 12 13 14 15 16 17 18	1350	875	1
5	31	29 30 31 32 33	1100	740	1

There are two features of the K means clustering. Based on the selection of centroid, the proposed algorithm can guarantee that at least there is one synchronous generator in each cluster when the algorithm terminates. The second one is that the sum of the distance from each node to the corresponding centroids is minimized, which means that the weighted sum of edges that connect different clusters is still small. This feature derives from the inherent nature of the proposed distortion function.

2.3 Economic Separation Method

This section will primarily focus on discussing the separation method with the objective of minimizing power loss on distribution lines. Line loss [51-53] on distribution lines is primarily determined by resistance and current. Since we cannot change the resistance on distribution lines, the main focus in this research to minimize power loss is reducing the electrical distance between each load and its corresponding generator. In the meanwhile, we also want to add power balance to the objective function.

Suppose there are *m* generators and *n* load in the system. Let Cap_i be the generator capacity at generator *i*, L_i be the sum of load that assigned to generator *i*. Assume load bus *j* is assigned to generator *i* and the electrical distance is ED_{ij}. The electrical distance is defined as the sum of resistance on the shortest path from load bus to its corresponding generator. The problem can be formulated as:

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} ED_{ij} \tag{18}$$

$$ED_{ij} = \sum_r R \quad r \in \{rs_{ij}\} \tag{19}$$

Subject to:

$$\sum_{i=1}^{m} L_i < Cap_i \tag{20}$$

where rs_{ij} represents all possible paths from generator *i* to load bus *j*, *R* is resistance on the selected path. In this research, the supply-demand balance within each partition is considered when assigning buses to their corresponding generators.

Based on the theory from computer science, separating the network into several partitions while satisfying power balance within each partition is a non-deterministic polynomial-time (NP-hard) problem. During the planning stage of dynamic microgrid partitioning, the objective is to identify the optimal locations to install smart switches. Suppose the number of lines in the system is N_l. It is possible to use one line for separation, or alternatively, two lines could be utilized. Moreover, as many as N_l-1 lines can be used for separating the system. The total number of separation methods is:

$$N_S = C_{N_l}^{1} + C_{N_l}^{2} + C_{N_l}^{3} + \cdots + C_{N_l}^{N_l-1} \tag{21}$$

where N_S is the number of separation methods in the system and C represents combination. Since solving NP-hard problem would consume large amount of computing power for a larger system with numerous nodes and edges. Hence this section will not separate the system based on the selection of connection lines. Instead, the system will be separated by assigning each bus to its corresponding generator, based on two objective functions: minimizing electrical distance and maintaining power balance within each partition.

2.3.1 Original Dijkstra's Algorithm

Dijkstra's algorithm [54-56] was introduced to find the shortest path between a source node to all other nodes in a graph. This technique is derived from graph theory. There are also several algorithms to find the shortest path, such as the Bellman-Ford algorithm and Depth-First Search. The reason why this research chose Dijkstra's algorithm is that it can be applied to an undirected graph, whereas the Bellman-Ford algorithm only works for a directed graph. As this research aims to minimize the electrical distance, which is an undirected value, Dijkstra's algorithm is the perfect fit for this purpose.

The next thing we need to address is to select proper physical values as edge weights. In Dijkstra's algorithm, the definition of the shortest path is determined by the weight assigned to each edge. Since we aim to minimize the electrical distance, it is natural to use line resistance as the edge weight. In this sense, Dijkstra’s algorithm would use the edge weight to find the path that minimizes the total electrical distance. The definition of the weight on each edge can be easily extended to other physical values based on different research purposes and that is the beauty of Dijkstra's algorithm. The pseudo code of modified Dijkstra's algorithm for calculating electrical distance on the shortest path is displayed in Fig. 2.

Modified Dijkstra's Algorithm

Let ***dist*** be an array that contains the current distances from the source to other vertices, Graph = (vertices, edges) be a weighted graph, edges(u, v) is the edge weight between node u and node v, Q be a queue that store nodes to be analyzed.

```
function Dijkstra(Graph, source):
    dist[source] = 0
    for each vertex v in Graph.vertices do
        dist[v] = INFINITY
        add v to Q
    end for

    while Q is not empty do
        u ← vertex in Q with minimum dist[u]
        remove u from Q

        for each neighbor v of u still in Q:
            distance = dist[u] + Graph.edges(u, v)
            if distance < dist[v] then
                dist[v] = distance
        end for
    end while
    return dist[]
```

Figure 2-25: Pseudo code for modified Dijkstra's algorithm.

We will use a small weighted graph to illustrate the detailed process of the original Dijkstra's Algorithm.

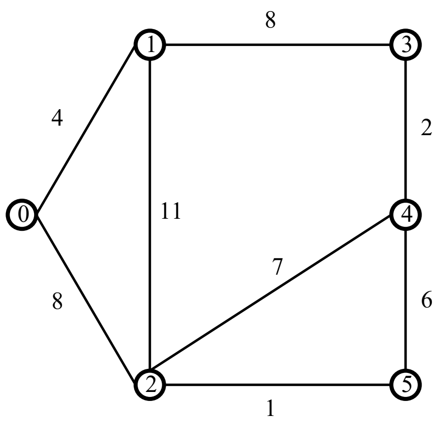


Figure 2-26: Weighted graph for Dijkstra's algorithm.

The number on each line represents the edge weight, which will be changed to resistance in subsequent research. Suppose we want to determine the shortest distance from node 0 to every node in this graph. This problem is similar to finding the shortest electrical distance from the generator to all the load buses in the system, assuming that node 0 is the generator and all other nodes serve as load buses. Suppose all nodes in the system haven't been explored, and we start with node 0. The initial distance from all nodes to node 0 is set to be infinity.

Node 0 has two neighbors: node 1 and node 2. Since the distance between node 0 and itself is naturally 0, the distance between node 1 and node 0 is 4 + 0 = 4, and the distance between node 0 and node 2 is 8 + 0 = 8. We can update the distance array.

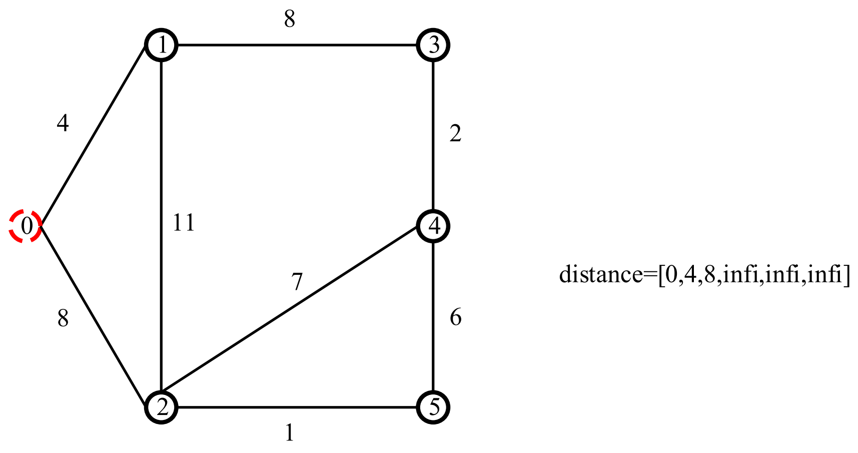


Figure 2-27: Exploration of node 0.

The distance array is sorted by the node order, starting with node 0 and ending with node 5. The dashed line surrounded by node 0 indicates that node 0 has already been analyzed. Our task is to determine the next node that needs to be analyzed. Looking at the distance table, we can see that node 1 has the shortest distance from node 0. Consequently, we will choose node 1 as the next node to analyze. Node 1 has two neighbors (node 0 has already been analyzed), node 3 and node 2. The distance between node 1 to node 0 is equal to 4 according to the distance table in Figure 2-

27. The distance between node 3 and node 0 is equal to 4(0-1) + 8(1-3) = 12. The distance between node 2 and node 0 is equal to 4(0-1) + 11(1-2) = 15. Since we have a distance between node 2 and node 0 in Figure 2-27 and it is 8, which is smaller than 15, we will not update the distance between node 2 and node 0. The updated distance is displayed.

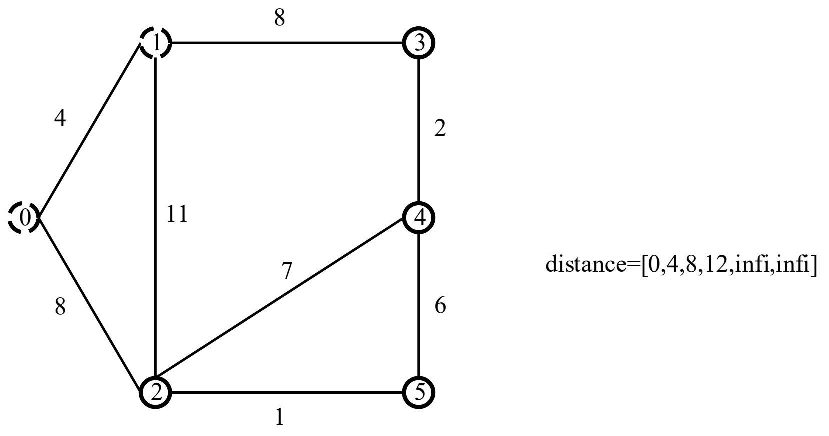


Figure 2-28: Exploration of node 1.

We have already discovered nodes 0 and 1, and the next node to be explored is node 2 because the distance between node 2 and node 0 is the shortest among the nodes that have not yet been explored. The distance between node 2 and node 0 is 8 based on the distance table in Figure 2-28. Node 2 has two neighbors: node 4 and node 5. The distance between node 4 and node 0 is 7(4-2) + 8(2-0) = 15, and the distance between node 5 and node 0 is 1(5-2) + 8(2-0) = 9. Since these distances are all smaller than infinity, we can update the distance table.

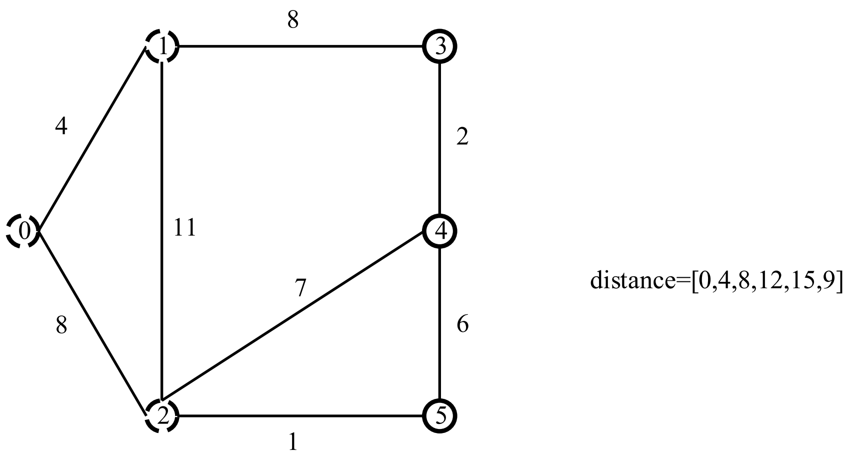


Figure 2-29: Exploration of node 2.

Even though we have substituted infinity with actual values in the distance table, three nodes remain undiscovered. Since 5 has the shortest distance to node 0 among the undiscovered nodes, it will be chosen as the next node for analysis. The distance from node 5 to node 0 is equal to 9, according to the distance table in Figure 2-29. Node 5 has only one neighbor that hasn't been explored, which is node 4. We can compute the distance between node 4 and node 0, which is 9(0-5) + 6(5-4) = 15. This distance matches the value in the distance table shown in this figure, so there is no need to update the distance table.

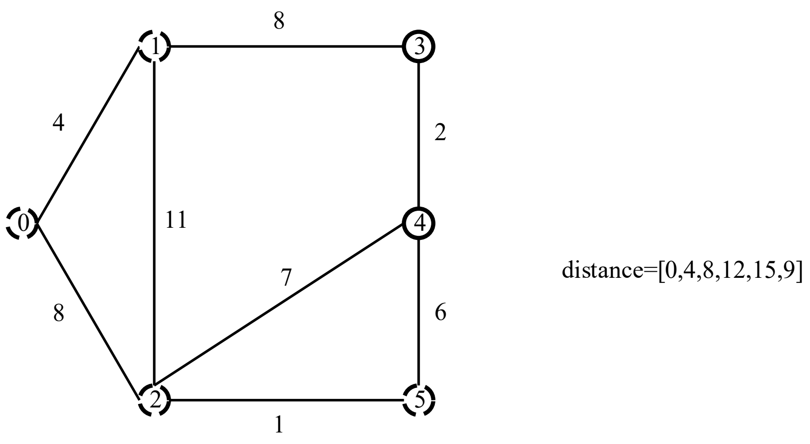


Figure 2-30: Exploration of node 5.

Referring to the distance table, we can determine that the next node requiring analysis is node 3. The distance between node 3 and node 0 is 12 based on the distance table in Figure 2-30. Node 3 has just one neighbor that hasn't been analyzed, which is node 4. Consequently, the distance between node 4 and node 0 is calculated as 12(0-3) + 2(3-4) = 14. This value is less than the corresponding distance of 15 in the distance table shown in Figure 2-30, so we update the distance table.

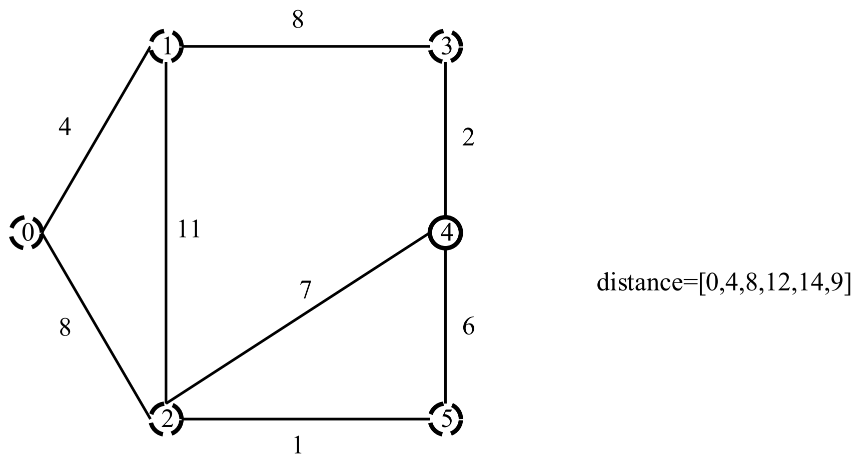


Figure 2-31: Exploration of node 3.

Now all nodes except node 4 have been explored and there is no need to analyze node 4. The shortest distance between each node to node 0 is displayed in the distance table.

2.3.2 Modification

Another objective we need to consider when separating the system is the power balance within each partition. Because Dijkstra's algorithm can only minimize the electrical distance. The proposed modification to Dijkstra's algorithm is displayed.

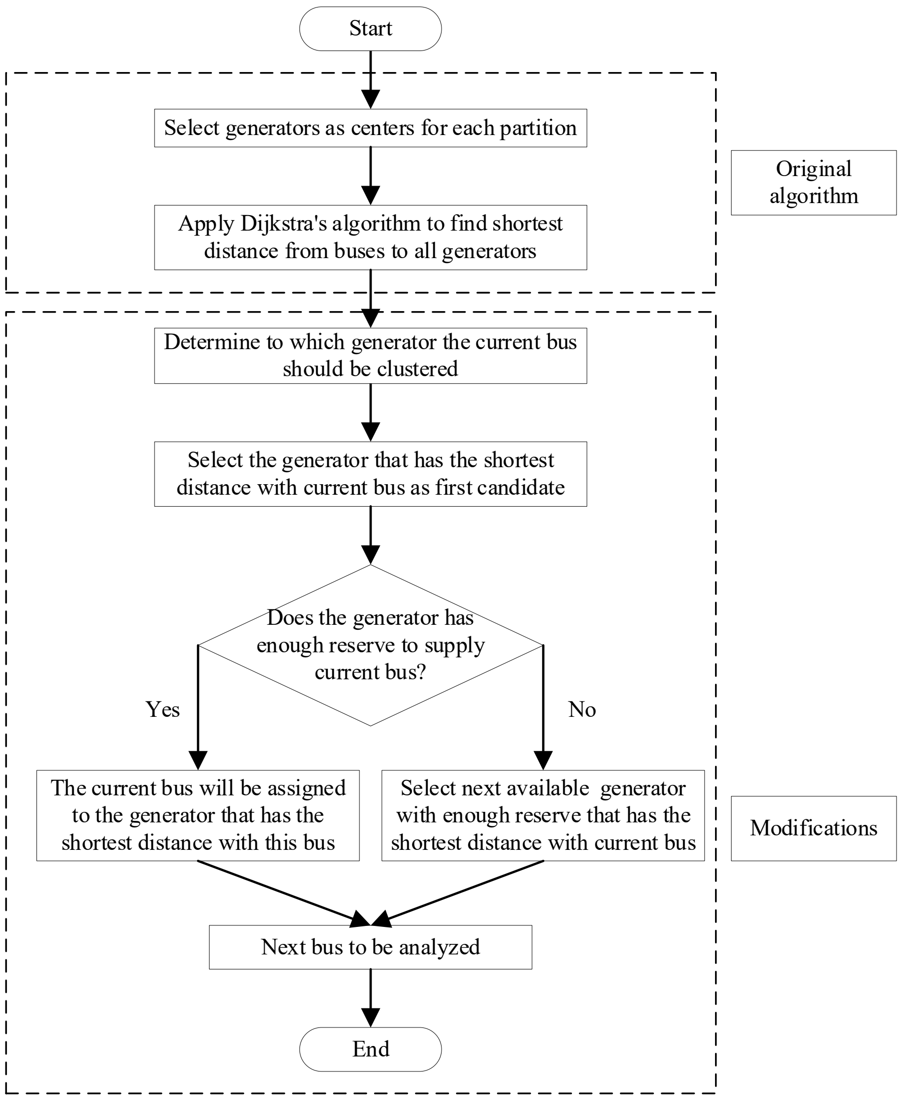

Start
Select generators as centers for each partition
Apply Dijkstra's algorithm to find shortest distance from buses to all generators
Original algorithm
Determine to which generator the current bus should be clustered
Select the generator that has the shortest distance with current bus as first candidate
Does the generator has enough reserve to supply current bus?
Yes
No
The current bus will be assigned to the generator that has the shortest distance with this bus
Select next available generator with enough reserve that has the shortest distance with current bus
Modifications
Next bus to be analyzed
End

Figure 2-32: Detailed process of modified Dijkstra's algorithm.

Suppose bus *j* is assigned to generator *i* according to Dijkstra's algorithm. However, before implementing the partitioning process, we first need to calculate the sum of the load that has already been assigned to generator *i*. If generator *i* still has enough reserve to supply bus *j*, we will assign bus *j* to generator *i*. If generator *i* lacks reserve, bus *j* will be clustered to the next available generator that has sufficient reserve and is with shortest distance to bus *j*. Basically, we want to minimize losses on distribution lines while maintaining power balance in each partition.

2.3.3 Simulation

The test system used in this simulation is the IEEE 39-Bus system.

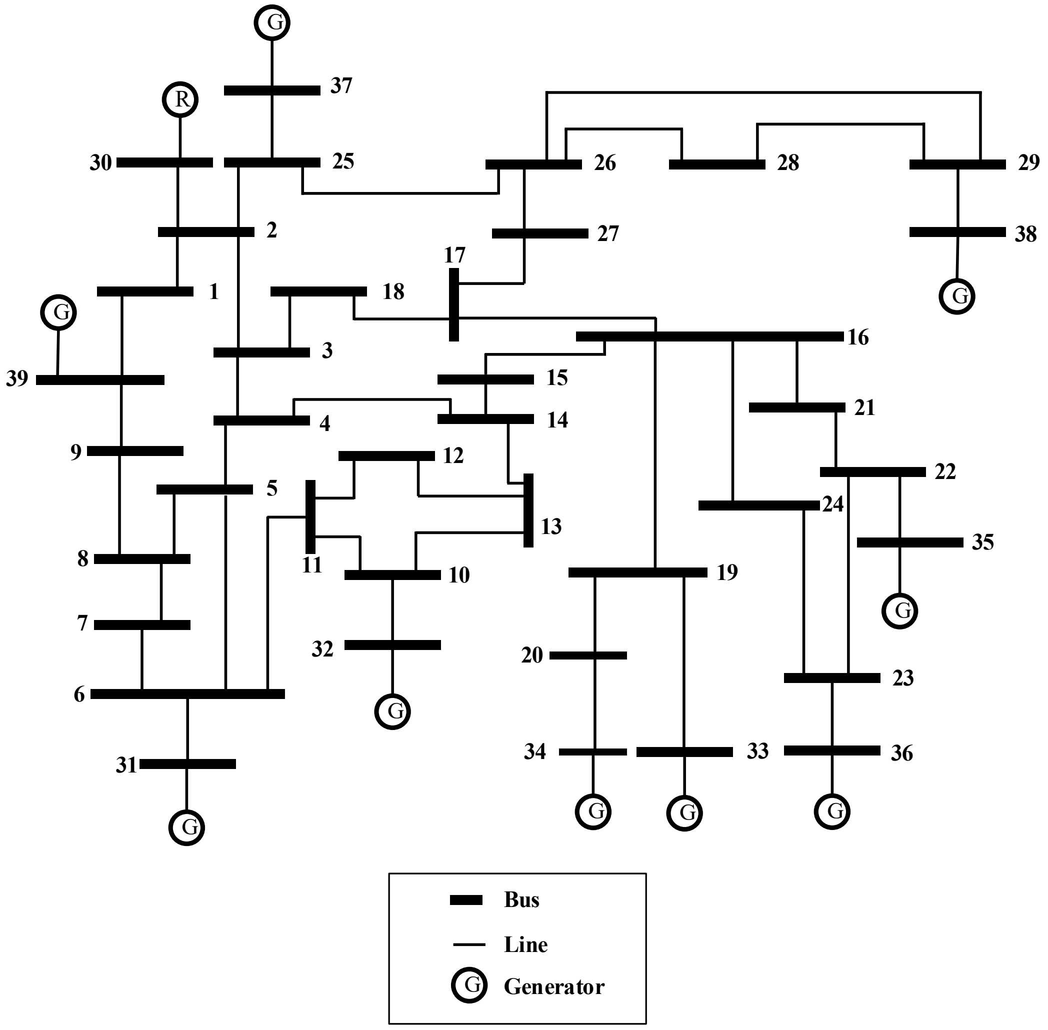


Figure 2-33: IEEE 39-Bus system.

Most researchers in this area focus only on radial distribution networks due to their simplicity when dealing with partitioning of distribution network. Imagine a situation where we have to partition a radial network into *k* partitions, while minimizing the electrical distance. The easiest way to achieve this is by choosing *k*-1 lines with the largest resistance as connecting edges, based on the nature of the radial network. However, unlike radial networks, simply selecting *k*-1 edges sometimes fails to separate mesh networks into *k* partitions. This simulation uses a mesh network to perform case studies that thoroughly demonstrate the effectiveness of the proposed algorithm.

In this case, assume generators placed on bus 30 are replaced with renewable generation. And assume the power output of the renewable generation is the same as that of the synchronous generator originally put on bus 30 during the given research period. Originally, the generator capacity was set to a much higher value than the generator output. However, in this simulation, we will consider a worse scenario by adjusting the generator capacity to be 1.5 times that of the generator output. The generator power output and its corresponding capacity is displayed.

Table 2-XIV: Generator Output and Capacity for IEEE 39-Bus System

Generator	Output (MW)	Capacity (MW)
30	250	375
31	571.3	856.95
32	650	975
33	632	948
34	508	762
35	650	975
36	560	840
37	540	810
38	830	1245
39	1000	1500

The proposed approach begins by retrieving all the necessary information from the IEEE 39-Bus system using PSLF in Python. The maximum number of partitions that can be generated is equal to the number of generators in the system. In this research, we assume that each generator in the system can supply voltage at nominal frequency. Since there are ten generators in the system, the maximum number of partitions that can be created is also ten. However, the generator connected to bus 33 and generator connected to bus 34 were combined to provide power supply to the system through bus 19. In this sense, they were treated as one source when partitioning the system.

Another thing needs to be mentioned is that in the system, some lines are actually transformers [57, 58].

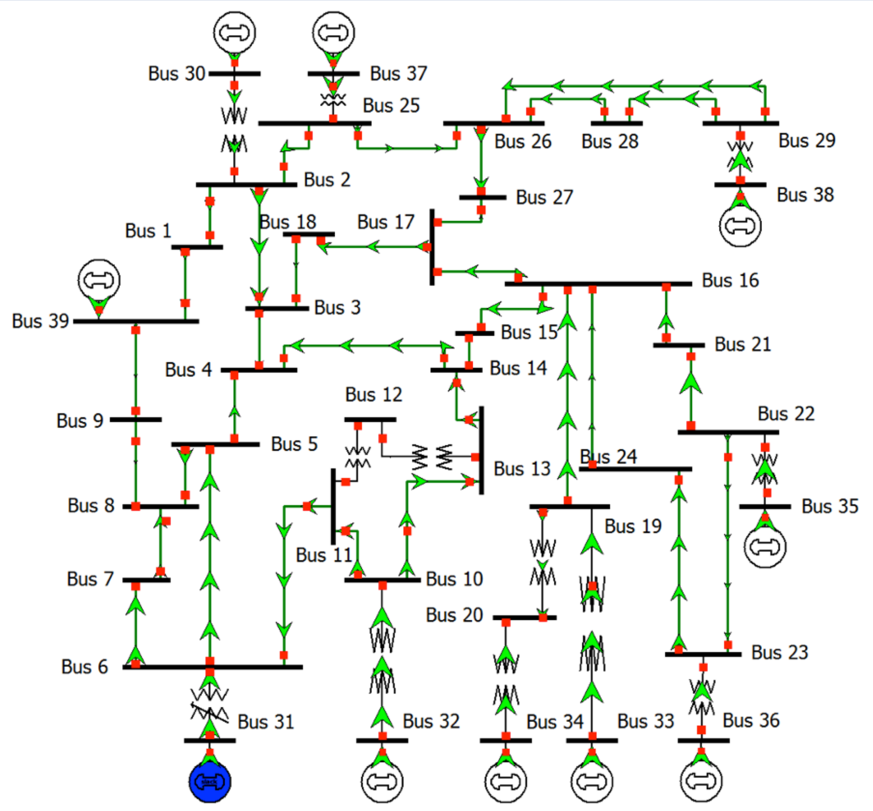


Figure 2-34: Lines and transformers in IEEE 39-Bus system.

Given that we are only considering system separation through the disconnection of lines, it is logical to group buses that have transformers and buses with generators together in the first step. For instance, it makes sense to put bus 2 and bus 30 together, and it is reasonable to group bus 31 with bus 6.

Table 2-XV: Group Generator Bus and Transformer Bus

Partition	Center	Buses within Partition
1	30	2, 30
2	31	6, 31
3	32	10, 32

4	33/34	20, 34, 19, 33
5	35	22, 35
6	36	23, 36
7	37	25, 37
8	38	29, 38
9	39	1, 39

Although there are 39 buses in the system, some of them don't have any loads. We will list the buses that have loads in the system.

Table 2-XVI: Loads in IEEE 39-Bus System

Bus	load (MW)	Bus	load (MW)
3	322	23	247.5
4	500	24	308.6
7	233.8	25	224
8	522	26	139
12	7.5	27	281
15	320	28	206
16	329.4	29	283.5
18	158	31	9.2
20	680	39	1104
21	274		

Now the next step is to find the shortest distance from the rest buses to the nine centers through the implementation of Dijkstra's algorithm in Python.

Table 2-XVII(a): Distance from Buses to Centers

Center	1	2	3	4	5	6	7	8	9	10
30	0.0035	**0**	**0.0013**	0.0026	0.0034	0.0036	0.0042	0.0042	0.0055	0.0047
31	0.0053	0.0036	0.0023	**0.001**	**0.0002**	**0**	**0.0006**	**0.001**	0.0033	0.0011
32	0.0064	0.0047	0.0034	0.0021	0.0013	0.0011	0.0017	0.0021	0.0044	**0**
33/34	0.0089	0.0054	0.0041	0.0051	0.0059	0.0061	0.0067	0.0067	0.009	0.0056
35	0.0089	0.0054	0.0041	0.0051	0.0059	0.0061	0.0067	0.0067	0.009	0.0056
36	0.0095	0.006	0.0047	0.0057	0.0065	0.0067	0.0073	0.0073	0.0096	0.0062

37	0.0105	0.007	0.0077	0.009	0.0098	0.01	0.0106	0.0106	0.0125	0.0106
38	0.015	0.0115	0.0102	0.0115	0.0123	0.0125	0.0131	0.0131	0.0154	0.0131
39	**0.001**	0.0045	0.0058	0.0049	0.0041	0.0043	0.0037	0.0033	**0.001**	0.0054

From this table, we can see that the distance between bus 2 and bus 30 is 0, and the distance between bus 6 and bus 31 is also 0. This is because there is a transformer between bus 2 and bus 30, and we grouped them together in the first place. The smallest values from the buses in the system to the generators are marked in bold in the table.

Table 2-XVII (b): Distance from Buses to Centers

Center	11	12	13	14	15	16	17	18	19	20
30	0.0043	2.1E+09	0.0043	0.0034	0.0047	0.0038	0.0031	**0.0024**	0.0054	2.1E+09
31	0.0007	2.1E+09	0.0015	0.0018	0.0036	0.0045	0.0041	0.0034	0.0061	2.1E+09
32	**0.0004**	2.1E+09	**0.0004**	**0.0013**	0.0031	0.004	0.0047	0.0045	0.0056	2.1E+09
33/34	0.006	2.1E+09	0.0052	0.0043	**0.0025**	**0.0016**	**0.0023**	0.003	**0**	2.1E+09
35	0.006	2.1E+09	0.0052	0.0043	**0.0025**	**0.0016**	**0.0023**	0.003	0.0032	2.1E+09
36	0.0066	2.1E+09	0.0058	0.0049	0.0031	0.0022	0.0029	0.0036	0.0038	2.1E+09
37	0.0107	2.1E+09	0.0102	0.0093	0.0075	0.0066	0.0059	0.0066	0.0082	2.1E+09
38	0.0132	2.1E+09	0.0127	0.0118	0.01	0.0091	0.0084	0.0091	0.0107	2.1E+09
39	0.005	2.1E+09	0.0058	0.0057	0.0075	0.0083	0.0076	0.0069	0.0099	2.1E+09

An interesting observation in this table is that the distance from bus 12 to all generators is infinite, and the same applies to bus 20. The reason is that all lines connected to bus 12 and bus 20 are transformers, and since we are only separating the system by disconnecting lines, we do not consider the resistance of transformers. Another thing that needs to be mentioned is that the distance from buses 11-20 to generators 33/34 and 35 is the same, except for bus 19. This is because buses wanting to reach generator 33/34 must go through line 16-19, and buses going to generator 35 must use lines 16-21 and 21-22. Since the distance between 16-19 is equal to the sum of the distances between 16-21 and 21-22, the distances from buses in the system to generators 33/34 and 35 will often have the same values.

Table 2-XVII (c): Distance from Buses to Centers

Center	21	22	23	24	25	26	27	28	29
30	0.0046	0.0054	0.006	0.0041	0.007	0.0058	0.0044	0.0101	0.0115
31	0.0053	0.0061	0.0067	0.0048	0.01	0.0068	0.0054	0.0111	0.0125
32	0.0048	0.0056	0.0062	0.0043	0.0106	0.0074	0.006	0.0117	0.0131
33/34	0.0024	0.0032	0.0038	**0.0019**	0.0082	0.005	**0.0036**	0.0093	0.0107
35	**0.0008**	**0**	0.0006	**0.0019**	0.0082	0.005	**0.0036**	0.0093	0.0107
36	0.0014	0.0006	**0**	0.0022	0.0088	0.0056	0.0042	0.0099	0.0113
37	0.0074	0.0082	0.0088	0.0069	**0**	**0.0032**	0.0046	0.0075	0.0089
38	0.0099	0.0107	0.0113	0.0094	0.0089	0.0057	0.0071	**0.0014**	**0**
39	0.0091	0.0099	0.0105	0.0086	0.0115	0.0103	0.0089	0.0146	0.016

Now that we have the shortest distances from all buses to generators, all we have to do is assign each bus to the generator with the least electrical distance. Meanwhile, we still need to check if the corresponding generator has enough reserve. The initial result from modified Dijkstra's algorithm is displayed.

Table 2-XVIII: Initial Separation Method for Modified Dijkstra's Algorithm

Center	Buses	Capacity	Demand
30	2, 3	375	322
31	4, 5, 6, 7	856.95	743
32	8, 10, 11, 12, 13, 14	975	529.5
33/34	15, 16, 17, 18, 19, 20	1710	1487.4
35	21, 22, 24, 27	975	863.6
36	23	840	247.5
37	25, 26	810	363
38	28, 29	1245	489.5
39	1, 9	1500	1104

From this table, we can see that the supply-demand balance is maintained within each partition. Additionally, based on the nature of Dijkstra's algorithm, the sum of electrical distance is minimized. Therefore, power loss minimization is achieved.

However, we can identify some issues when we examine the IEEE 39 Bus system using the separation method. For instance, generator 31 is connected to four buses: bus 4, bus 5, bus 6, and bus 7. If we attempt to connect bus 4 and bus 5 to generator 31, it will go through bus 8, which is assigned to generator 32. In fact, there is a long path from bus 8 to generator 32, involving many buses. The cause of this problem lies in our approach to assigning buses to generators, which primarily considers minimizing electrical distance and ensuring power balance. We have not taken into account whether all buses on the path with the least electrical distance to the generator have already been assigned to the same generator. In order to solve that problem, we need to find another solution.

Previously, we found that bus 15, 16, 17, 24, and 27 have the same shortest electrical distance to generator 33/34 and generator 35. We can begin by allocating the remaining buses and then deal with them later. Let's assign buses to the generators that have the shortest electrical distance to them and can ensure that the shortest path stays within the partition.

Table 2-XIX: Initial Separation Method

Center	Buses	Demand (MW)	Capacity (MW)
30	2, 3, 18	480	375
31	4, 5, 6, 7, 8	1255.8	856.95
32	10, 11, 12, 13, 14	7.5	975
33/34	19, 20	680	1710
35	21, 22	274	975
36	23	247.5	840
37	25, 26	363	810
38	28, 29	489.5	1245
39	1, 9	1104	1500

This table shows that all partitions can maintain power balance except for generator 30 and generator 31. However, if we take a closer look for the buses assigned to generator 30, we can see that there is a transformer between generator 30 and bus 2, and there is a connection line between bus 2 and bus 3. To supply bus 18 with power from generator 30, bus 18 needs to connect to bus 3 first. In this sense, it is reasonable to remove bus 18 from generator 30. Similarly, we can remove bus 4 from generator 31. Now all partitions can satisfy power balance. We will assign bus 4, 18, 15, 16, 17, 24 and 27 in the next step. First, we list the distance from these buses to generators in the system.

Table 2-XX: Distance from Buses to Centers

Centers	4	15	16	17	18	24	27
30	0.0026	0.0047	0.0038	0.0031	**0.0024**	0.0041	0.0044
31	**0.001**	0.0036	0.0045	0.0041	0.0034	0.0048	0.0054
32	**0.0021**	0.0031	0.004	0.0047	0.0045	0.0043	0.006
33/34	0.0051	**0.0025**	**0.0016**	**0.0023**	**0.003**	**0.0019**	**0.0036**
35	0.0051	**0.0025**	**0.0016**	**0.0023**	**0.003**	**0.0019**	**0.0036**
36	0.0057	0.0031	0.0022	0.0029	0.0036	0.0022	0.0042
37	0.009	0.0075	0.0066	0.0059	0.0066	0.0069	0.0046
38	0.0115	0.01	0.0091	0.0084	0.0091	0.0094	0.0071
39	0.0049	0.0075	0.0083	0.0076	0.0069	0.0086	0.0089

Let's analyze bus 4 first. Since bus 4 cannot be assigned to generator 31, the second shortest distance from bus 4 to these generators is to generator 32. As generator 32 has enough reserve capacity to supply the load at bus 4, and all the nodes on the shortest path from bus 4 to generator 32 have already been assigned to generator 32, bus 4 can be allocated to generator 32. The next bus needs to be analyzed is bus 18. The second shortest distance from bus 18 to these generators is to generator 33/34 or 35.

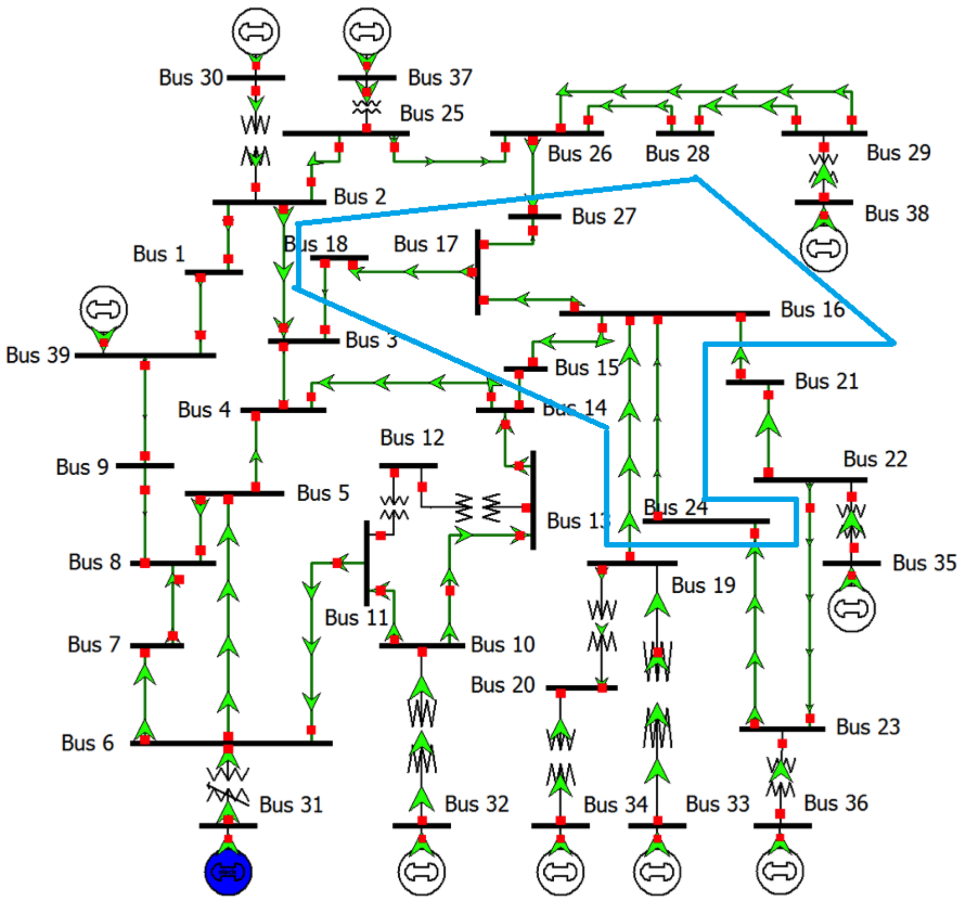


Figure 2-35: Buses have the same shortest distance to generator 33/34 and 35.

This figure groups all buses that have the same shortest electrical distance to generator 33/34 and generator 35. We initiate the allocation process between these buses and generator 33/34 and generator 35. If we assign bus 16 to generator 35, buses 24 and 15 will also be assigned to generator 35 since they are all directly connected to bus 16. The total load on buses 16, 15, and 24 amounts to 958 MW, which exceeds the remaining capacity of generator 35. Therefore, it is reasonable to assign all three of these buses to generator 33/34.

Since bus 15, 16 and 24 are assigned to generator 33/34. Bus 17, 18 and 27 cannot be assigned to generator 33/34 or generator 35, since bus 16 has been assigned to generator 33/34. Since bus 18 can not be assigned to generator 30 because of power balance issue, bus 17, bus 18 and bus 27 needs to be combined together to consider. The sum of load is equal to 439 MW. These three loads could be allocated to generator 37 or generator 38 through bus 26. All these three buses have a less electrical distance to generator 37 and generator 37 has enough reserve to supply loads on these three buses. Hence it is reasonable to assign 17, 18 and 27 to generator 37.

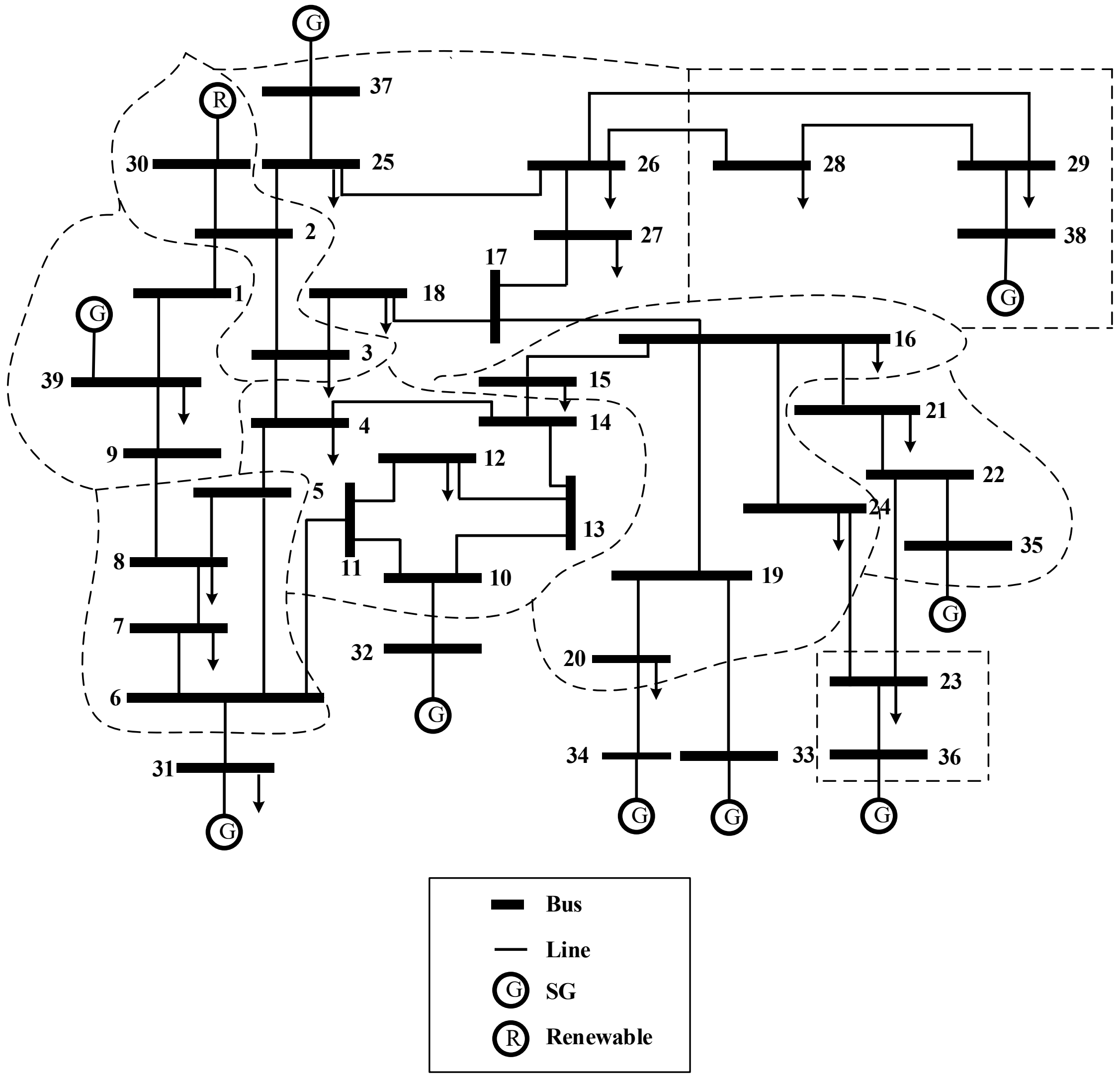


Figure 2-36: Final separation result.

Table 2-XXI: Final Separation Result

Partition	Generator	Buses	Demand (MW)	Capacity (MW)
1	30	2, 3	322	375
2	31	5, 6, 7, 8	765	856.95
3	32	4, 10, 11, 12, 13, 14	507.5	975
4	33/34	15, 16, 19, 20, 24	1638	1710
5	35	21, 22	274	975
6	36	23	247.5	840
7	37	17,18, 25, 26, 27	802	810
8	38	28, 29	489.5	1245
9	39	1, 9	1104	1500

As we can see, all partitions satisfy the power balance, and some partitions may have considerable amount of reserve. The reason is that the proposed algorithm will allocate load to the generator that has the shortest electrical distance with it if the generator has enough reserve to supply this load. We can observe from this table that some partitions in the system have only a few buses. This is because there are ten generators and thirty-nine buses in the system, which means that the average number of buses per partition should be around four ($39/10 \approx 4$).

3. Islanded Mode

When the distribution network is disconnected from the main grid due to a fault upstream, the dynamic microgrids will operate in islanded mode. If the MGs are designed with self-adequacy [59-61] in the planning stage, the system can be separated into several sub-systems. Suppose there is a sudden trip of generation, and the power balance for the system will be disrupted. Load shedding is the only solution to maintain power balance [62, 63]. The main advantage of load shedding is that it is equivalent to increasing generation in a step-function manner. The traditional method to increase generation involves controlling the governor to generate more power at a non-linear speed. In contrast, load shedding is similar to an instantaneous increase in power generation.

3.1 Problem Formulation

When load shedding is inevitable, the system should minimize the outage cost. Some advanced technology companies may incur up to one million dollars in losses per hour when faced with power outages, whereas most residential consumers experience comparatively smaller hourly losses. It is crucial to distinguish between the loads that experience higher hourly losses and those that incur lower hourly losses.

3.1.1 Load Cost

In this research, we aim to categorize loads based on their importance. We have divided the loads into three categories: average load, non-critical load, and critical load, and assigned different values to each based on their level of importance. Critical loads are assigned the highest level of importance, followed by non-critical loads and then average loads. In reality, each load point may contain one or more load categories. It may contain 30 percent non-critical load and 70 percent average load. To account for this, we calculate the weighted load importance for each load based on its portion.

Another way to classify different loads is based on their functionality, which we have categorized into six types: industrial load, commercial load, agricultural load, residential load, governmental & institutional load, and office & building load. We will use these load categories to determine the interruption cost of each type of load [64]. It is worth noting that the interruption duration may also affect its cost.

Table 3-I: Interruption Cost with Duration

Customer Type	Interruption Duration & Cost ($/kW)				
	1min	20min	60min	…	480min
Industrial	1.625	3.868	9.085	…	55.81
Commercial	0.381	2.969	8.552	…	83.01
Agricultural	0.06	0.343	0.649	…	4.12
Residential	0.001	0.093	0.482	…	16.69
Govt. & Inst.	0.044	0.369	1.492	…	26.04
Office&Bldg.	4.778	9.878	21.06	…	119.2

From this table, it becomes clear that the duration of the interruption determines the interruption cost for each customer type. As the duration of the interruption increases, so does the cost. However, some types of load experience a more rapid increase in cost than others. This research will assume the worst-case scenario, where the interruption cost is based on the longest duration. Additionally, each load point may have multiple types of customers, requiring the weighted interruption cost to be calculated based on each type's proportion.

In this research, when a load point contains a critical load, we classify it as a critical load point, given that critical loads hold the highest importance. Critical facilities, including hospitals, data centers, emergency services, and telecommunication infrastructure play a vital role in the proper functioning of our society and the people's lives. Therefore, it is reasonable to assign the highest priority to these load points. Conversely, if a load point exclusively comprises non-critical loads and average loads, we categorize it as a non-critical load point, and the load cost is determined by considering two key factors: the weighted load importance and the weighted interruption cost.

$$L_{nc} = L \cdot LP_{weighted} \cdot IC_{weighted}$$

(22)

where L_{nc} is the load cost (\$) for the non-critical load point. L is the load value, $LP_{weighted}$ and $IC_{weighted}$ correspond to the weighted load importance and interruption cost (\$/kW). For the load cost of the critical load point, we don't consider the load importance; the way to calculate the load cost is:

$$L_{cri} = L \cdot IC_{weighted}$$

(23)

where L_{cri} is the load cost (\$) for critical load point and L is the load value.

3.1.2 Objective Function

The objective function in load shedding aims to minimize the cost associated with the removal of loads while still meeting the load shedding requirements. This can be expressed as

$$min \sum LC(L_1 + L_2 + \cdots + L_n)$$

(24)

Subject to

$$\sum L_1 + L_2 + \cdots + L_n \in [T, T + T_{threshold}]$$

(25)

where LC is the function to calculate load cost of load point, $L_1, L_2, L_3 \ldots L_n$ denote the load removed from load shedding solution. T stands for the load shedding target. $T_{threshold}$ is the threshold considered for load shedding. It is important to note that our research diverges from previous studies by introducing the concept of a load shedding threshold. In contrast to previous studies that often employ a fixed load shedding target, our research introduces the concept of a load shedding threshold. This threshold allows for a more flexible approach to load shedding, considering that achieving a fixed amount of load removal can be challenging in real-world scenarios. The utilization of this threshold provides a more adaptive and practical solution to the load shedding problem.

3.2 Preliminary Solutions

In the problem formulation, it becomes apparent that this problem shares similarities with the Knapsack problem, with the primary difference being that the Knapsack problem seeks to maximize value, whereas in load shedding problems, our goal is to minimize the total load cost when implementing load shedding.

3.2.1 Knapsack Problem

In the Knapsack problem [65-67], you are provided with a set of boxes, each assigned specific values and weights, and your objective is to choose a subset of these boxes to place inside a bag.

Simultaneously, you must ensure that the total weight of the selected boxes does not exceed the bag's maximum capacity. The solution involves selecting a subset of boxes to maximize the total value within the constraints of the bag's capacity.

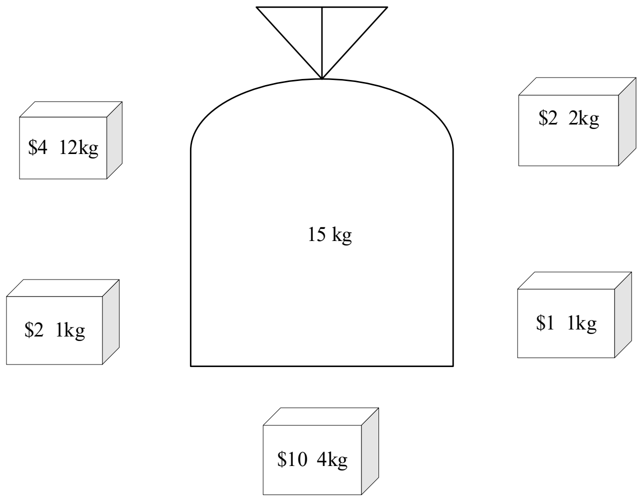


Figure 3-1: Knapsack problem.

As shown in this figure, five boxes are available for selection, and the bag's maximum weight capacity is 15 kg. Our objective is to select a subset of boxes that will maximize the total value. In this sense, the most straightforward approach is to enumerate all combinations of boxes to discover the one that satisfies the 15 kg constraint and maximizes the total value. If the number of boxes is relatively small, it is feasible to employ a computer to generate all combinations and subsequently perform the necessary calculations. However, when dealing with a substantial system comprising numerous boxes, this process can be time-consuming and resource-intensive.

3.2.2 Dynamic Programming

Dynamic programming [68-70] is the most used solution to solve knapsack problem. Since dynamic programming will divide a big problem into several subproblems, the solution is guaranteed to be global optimal.

The first step of dynamic programming is to build a 2-D table called "dp", we will use the 2-D table to store the solution to sub-problems. Suppose there are *n* loads in the system and the load shedding target is equal to *t*, 2-D table will be a *t*×*n* table. The value in 2-D table $dp[j][i]$ is to store minimal load cost achieved by considering the first *i* loads while ensuring that their cumulative load equals *j*.

$$L_x + L_y + \cdots L_z = j \tag{26}$$

$$dp[j][i] = \arg min\ C_{L_{1,2,\ldots i}} \tag{27}$$

where L_x, L_y…L_z is the solution to the sub-problem; they are a subset of first *i* loads. *C* stands for combination, and $dp[j][i]$ is the least load cost we can get from all subsets of first *i* loads. The complete 2-D table will be displayed.

Table 3-II: Dynamic Programming Table

	0	1	…...	i-1	i	…...	n
dp[1]	dp[1][0]=0	dp[1][1]	…...	dp[1][i-1]	dp[1][i]	…...	dp[1][n]
dp[2]	dp[2][0]=0	dp[2][1]	…...	dp[2][i-1]	dp[2][i]	…...	dp[2][n]
…...	…...	…...	…...	…...	…...	…...	…...
dp[j-1]	dp[j-1][0]=0	dp[j-1][1]	…...	dp[j-1][i-1]	dp[j-1][i]	…...	dp[j-1][n]
dp[j]	dp[j][0]=0	dp[j][1]	…...	dp[j][i-1]	dp[j][i]	…...	dp[j][n]
…...	…...	…...	…...	…...	…...	…...	…...
dp[t]	dp[t][0]=0	dp[t][1]	…...	dp[t][i-1]	dp[t][i]	…...	dp[t][n]

The 2-D table will be populated row by row, starting from the first row and continuing until the last row. Each row corresponds to a specific sub-problem with a distinct load-shedding target. The first row is dedicated to solutions for a load-shedding target of 1, the second row for a target of 2, and so on, with the *j*th row signifying a load-shedding target of j. The final solution is represented by $dp[t][n]$, with load shedding target equal to *t* through all *n* loads.

Within each row, the value in table still represents sub-problems specific to that particular row. For instance, column 0 in *j*th row represents the scenario where we employ the first 0 loads to address the sub-problem while ensuring that the cumulative load equals j. Similarly, the value in column *i* of *j*th row corresponds to employing the first *i* loads to tackle the sub-problem with a total load equal to *j*.

Suppose we need to calculate the sub-problem represented by $dp[j][i]$, and all rows before *j*th row have already been solved, along with all columns before column *i*. As we iterate through the loads, and the current load that needs to be analyzed is load i, we encounter two possible options. The first option is that we don't care about load i, we can still use first i-1 load to satisfy the requirement.

$$dp[j][i] = dp[j][i-1] \tag{28}$$

The second option is that we will consider load i, but this time we will use the first i-1 load to make up j-L_i, and we have to add load cost of load *i*.

$$dp[j][i] = dp[j-L_i][i-1] + cost(L_i) \tag{29}$$

The most important thing of dynamic programming is the state transition function. Since we are trying to find the minimum load cost we can get, the state transition function would be:

$$dp[j][i] = \min\left(dp[j][i-1], dp[j-L_i][i-1] + cost(L_i)\right) \tag{30}$$

The most important data structure used in dynamic programming is a 2-D table *dp*. In this table, the first index refers to the current load shedding target we want to achieve. In Python, it is limited to integers, but in reality, it can be a floating-point number. For example, if the load shedding target should be 203.4 MW, then dynamic programming won't work.

3.2.3 Original Greedy

When the problem involves maximizing or minimizing something, greedy algorithms [71-73] will certainly come to mind. In the load-shedding problem, we aim to minimize the cost of the load we remove. If we could calculate the per-unit value of the load cost for each load, we could always choose the load with the least per-unit cost.

Suppose there are six load points in the system and their corresponding load cost is displayed.

Table 3-III: Per-unit Load Cost

	1	2	3	4	5	6
Load (MW)	10	20	30	20	10	40
Load Cost ($ MM)	60	100	120	80	30	80
Per-unit ($ MM/MW)	6	5	4	4	3	2

The "MM" in the table stands for million dollars. This table shows that load point 6 has the lowest per-unit value of load cost. If the load shedding target is 50 MW, it would be reasonable to choose load point 6 and load point 5 as our load shedding solution.

The only factor preventing us from applying the greedy algorithm to real-world problems is the assumption that each load can be divided into 1 MW pieces. However, loads cannot be divided into such discrete units. Therefore, the original greedy algorithm is not suitable for practical engineering problems.

3.3 Proposed Load Shedding Strategy

Since we initially categorized the load into average load, non-critical load, and critical load, the load shedding strategy will also be divided into three stages [74-78]. Each stage will deal with a certain category of load.

3.3.1 Three stage load shedding

The first stage of our policy involves the load aggregator. Historically, demand response programs have only targeted large industrial and commercial customers, utilizing complex control strategies to shed loads during peak periods. However, with load aggregator [79-81], residential users can participate in demand response programs as a cohesive entity, receiving control schemes directly from the system operator. In this sense, during power supply shortage, the load aggregator can remove loads that have a contract with utility companies, prioritizing their removal over other loads. Let T represent the load-shedding target, G represents the total generation capacity, and L represent the total load in the system. The three-stage load shedding is displayed.

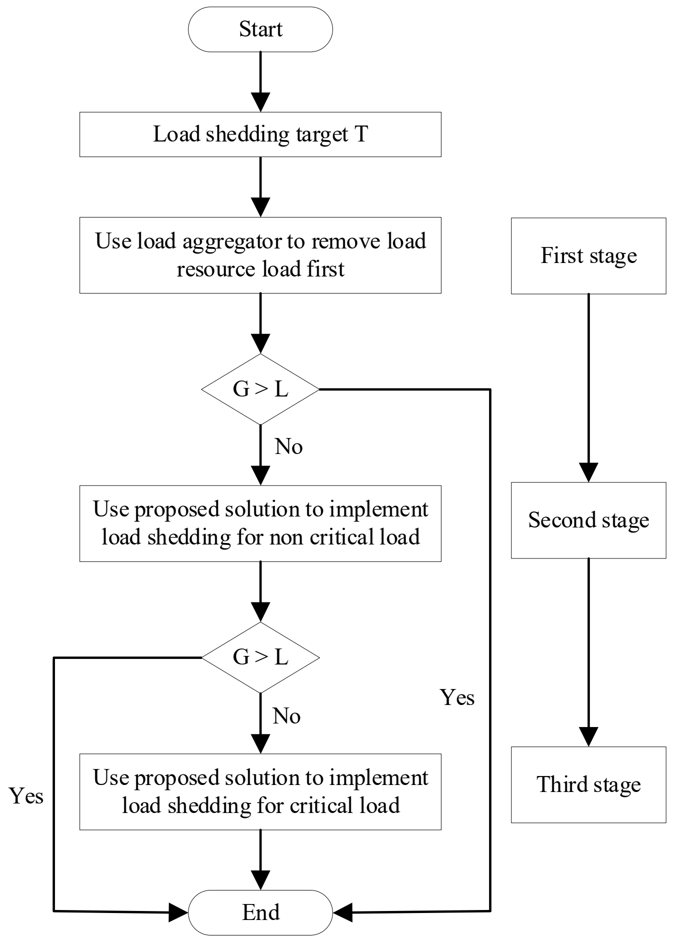

Start
Load shedding target T
Use load aggregator to remove load resource load first
G > L
No
Yes
Use proposed solution to implement load shedding for non critical load
G > L
No
Yes
Use proposed solution to implement load shedding for critical load
End
First stage
Second stage
Third stage

Figure 3-2: Three stage load shedding.

Following the implementation of the first stage of load shedding, if the system demand still exceeds the generation output, the second stage must be initiated. The primary objective of this stage is to minimize the load cost of the removed non-critical load point.

$$min \sum L_{nc}$$

(31)

Although the original greedy algorithm cannot be applied to practical engineering problems, we can still draw inspiration from its underlying concept. This time, we will treat each load point as a whole entity and not divide them into 1 MW pieces. The proposed load shedding method is a greedy-based sorting algorithm.

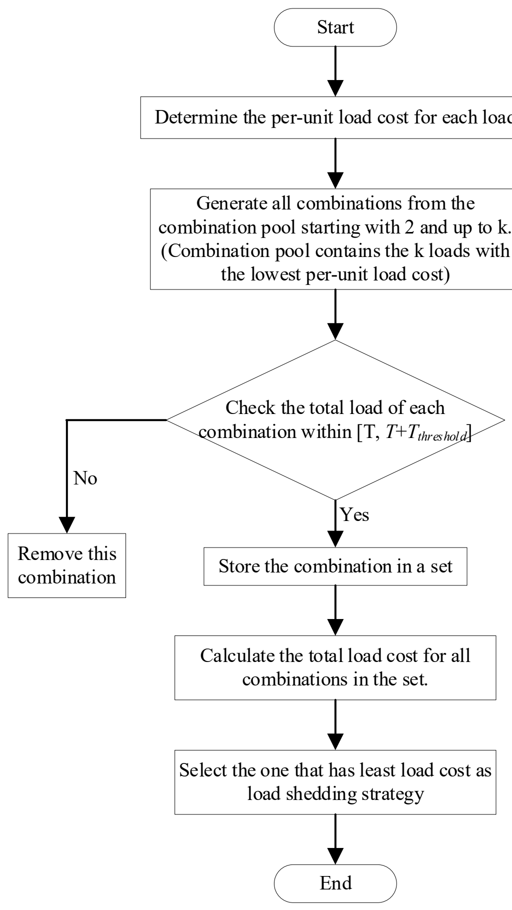

Start
Determine the per-unit load cost for each load
Generate all combinations from the combination pool starting with 2 and up to k. (Combination pool contains the k loads with the lowest per-unit load cost)
Check the total load of each combination within [T, T+T_threshold]
No
Yes
Remove this combination
Store the combination in a set
Calculate the total load cost for all combinations in the set.
Select the one that has least load cost as load shedding strategy
End

Figure 3-3: Proposed load shedding algorithm.

As displayed in this figure, the first step is to calculate the per-unit load cost for each load point.

$$L_{C\_unit} = \frac{L_C}{L}$$

(32)

where L_{C_unit} is the per-unit load cost for the current load (\$/kW), L_C is the load cost (\$), and L refers to the load (kW). The idea behind this equation is the greedy algorithm. The next step is to generate all combinations from the combination pool starting with 2 (C_k^2) and up to k (C_k^k). The combination pool is defined as a subset that only contains the first k loads with the lowest per-unit load cost. The value of k is determined as the largest number to implement combination, which will be illustrated later. By selecting the load with the lowest per-unit load cost as load shedding solution, we can ensure that the sum of load cost remains minimal.

The value k is the largest number for combination, which can be calculated through the process displayed in the figure below.

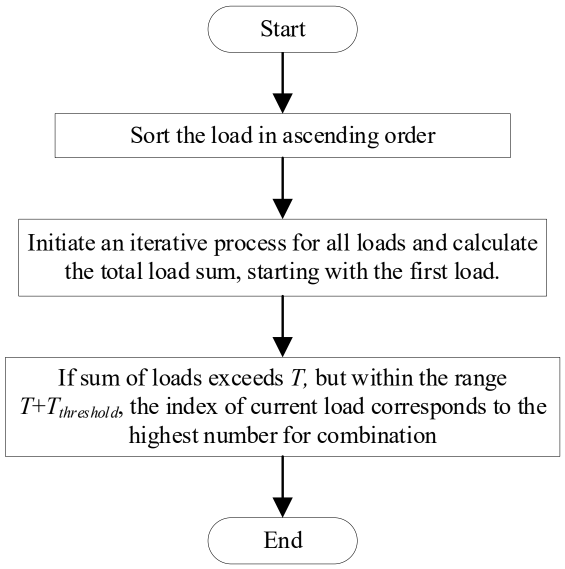

Start
Sort the load in ascending order
Initiate an iterative process for all loads and calculate the total load sum, starting with the first load.
If sum of loads exceeds T, but within the range $T+T_{threshold}$, the index of current load corresponds to the highest number for combination
End

Figure 3-4: Calculation of k.

The value of k is determined to reduce the proposed algorithm's computing time and memory usage. Without knowing the largest number for combination, and with n loads in the system, all combinations must be checked from 1 to n, as the answer may lie in any of these combinations. However, the value of k helps to improve the computing time of the algorithm by solving the following equation:

$$T_{im} = \frac{\sum_{i=1}^{n} c_n^i}{\sum_{i=1}^{k} c_n^i}$$

(33)

where T_{im} is the improvement in computing time, *C* represents combination. Since the loads are sorted in ascending order, we can ensure that *k* is the largest number for combination.

We build the combination pool in the second step to reduce memory usage. Initially, we have to choose i ($i\in [1, k]$) load from *n* loads, which could lead to a significant waste of memory. However, since the solution to our problem must include the load with the smallest per-unit value of load cost, we can reduce the combination pool from *n* to *k* by selecting only the first *k* loads with the smallest per-unit value of load cost. Since the largest number for combination is *k*, we can select *i* ($i\in [1, k]$) load from *k* loads, and the memory usage improved can be expressed:

$$S_{im} = \frac{\sum_{i=1}^{k} n}{\sum_{i=1}^{k} k}$$

(34)

where S_{im} is the improvement in memory usage, *k* represents the largest number for combination. Since the combination pool contains the first *k* load with the smallest per unit value of load cost, we can ensure that the solution to our problem lies within this set.

If the power balance is still not achieved after implementing the second load-shedding stage, the third stage must be initiated. The primary objective in this stage remains the same: minimizing the sum of load costs. In the third stage, the load-shedding approach differs from the second stage as interruption cost will not be considered. Interruption cost will only be considered for non-critical loads in the second stage and third stage only deal with critical loads. The proposed algorithm will also be utilized for load shedding in the third stage.

After implementing the three-stage load-shedding policy, the system's supply-demand balance will be maintained. The system can still be partitioned into multiple sub-systems to reduce the control complexity. With the method we proposed in the second part, we can successfully partition the system while achieving self-adequacy within each sub-system.

3.3.2 Case Study

The case study is still implemented in IEEE 33 bus system,

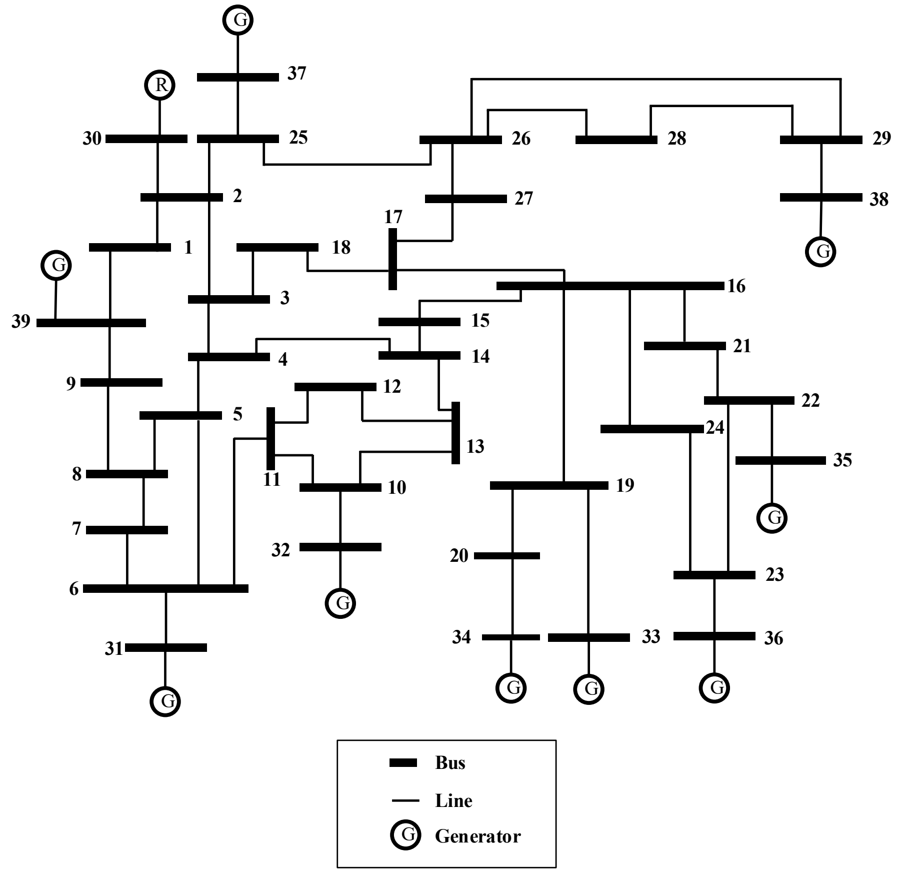


Figure 3-5: IEEE 39-Bus system.

There are ten generators and several load points in the system. First, we need to determine the load cost for each load point. To calculate the load cost for each load point, we must first calculate the weighted load importance and the weighted interruption cost.

Table 3-IV: Weighted Load Importance

Bus	Average load ratio	Non-critical load ratio	Critical load ratio	Weighted load importance
3	38	35	27	5.67
4	56	27	17	4.83
7	47	34	19	5.16

8	83	15	2	3.57
12	76	10	14	4.14
15	58	20	22	4.92
16	77	20	3	3.78
20	67	18	15	4.44
21	57	13	30	5.19
24	63	4	33	5.1
25	72	19	9	4.11
26	67	14	19	4.56
27	60	16	24	4.92
28	54	15	31	5.31
29	87	13	0	3.39
39	62	25	13	4.53

In this research, we assign a load importance value of 9 to critical loads, 6 to non-critical loads, and 3 to average loads. The load with the highest ratio of critical load will have the largest weighted load importance.

The load is also divided into six categories: industrial load, commercial load, agricultural load, residential load, governmental & institutional load, and office & building load. The weighted interruption cost must be calculated based on the ratio of each load category.

Table 3-V: Weighted Interruption Cost

Bus	Industrial /%	Commercial /%	Agricultural /%	Residential /%	Govt. & Inst. /%	Office& Bldg. /%	Weighted Interruption Cost($/kW)
3	11	0	28	27	18	16	35.5582
4	10	12	24	14	38	2	31.1468
7	3	19	24	14	25	15	45.1616
8	1	6	23	13	10	47	67.284
12	22	13	21	18	14	12	44.8885
15	23	5	23	17	17	15	43.0785
16	12	7	20	29	28	4	30.2312
20	9	13	24	14	13	27	54.7088
21	20	15	29	20	16	0	32.3127
24	29	9	28	12	21	1	33.4726
25	20	13	31	24	12	0	30.3609
26	0	3	26	21	17	33	50.8292
27	5	13	32	14	14	22	47.1064
28	17	16	23	10	17	17	50.0767

29	21	8	25	12	33	1	31.1789
39	2	18	30	21	13	16	43.2561

Now, we can calculate the load cost for each load using. The results are displayed in Table VI, The $ MM in Table VI stands for million dollars.

Table 3-VI: Load Cost

Bus	Load/MW	Weighted load importance	Weighted interruption cost($/kW)	Load cost/ $ MM
3	322	5.67	35.5582	64.9200
4	500	4.83	31.1468	75.2195
7	233.8	5.16	45.1616	54.4833
8	522	3.57	67.284	125.3864
12	7.5	4.14	44.8885	1.3938
15	320	4.92	43.0785	67.8228
16	329.4	3.78	30.2312	37.6418
20	680	4.44	54.7088	165.1768
21	274	5.19	32.3127	45.9506
24	308.6	5.1	33.4726	52.6812
25	224	4.11	30.3609	27.9514
26	139	4.56	50.8292	32.2176
27	281	4.92	47.1064	65.1255
28	206	5.31	50.0767	54.7769
29	283.5	3.39	31.1789	29.9650
39	1104.9	4.53	43.2561	216.5053

Suppose there is a sudden trip of generator encountered on bus 33, bus 36, and bus 37. Now the most effective way to maintain power balance is load shedding. Initially, the power output from these three generations is 632MW, 560MW, and 540MW. Therefore, our load-shedding target is 1732 MW.

To determine the load shedding threshold, we consider the sum of load in the system, 6149.5 MW, distributed across 19 buses with loads. This results in an average load of approximately 324 MW per bus. Thus, we set the threshold value at 324 MW.

At the first load-shedding stage, a load aggregator will remove the customers who have signed contracts with utility companies. Therefore, the load on bus 18, bus 23, and bus 31 will be removed through the load aggregator. Doing so alleviates a significant amount of load from the system, resulting in a decrease of 414.7 MW from the original load shedding target. Now, the new load shedding target is adjusted to 1317.3 MW.

Before implementing the proposed greedy-based sorting algorithm in Python, we need to find the largest value for combination. First, we have to sort the load in ascending order.

Table 3-VII: Loads in Ascending Order

Ascending order	Bus	Load/MW
1	12	7.5
2	26	139
3	28	206
4	25	224
5	7	233.8
6	21	274
7	27	281
8	29	283.5
9	24	308.6
10	15	320
11	3	322
12	16	329.4
13	4	500
14	8	522
15	20	680
16	39	1104.9

Our load shedding target is 1317.3 MW and load shedding threshold is 414.7 MW, leads to the load shedding range is [1317.3MW, 1732MW]. Since 7.5 + 139 + 206 + 224 + 233.8 + 274 + 281 = 1365.3 MW > 1317.3MW and 1317.3MW $\in$ [1317.3MW, 1641.3MW], the largest value for combination is 7 and *k* is equal to 7.

Next, we calculate the per unit value of load cost for the remaining loads. Since *k* is equal to 7, we will select seven loads with the smallest per unit value of load cost to form a combination pool. The combination pool is displayed in ascending order in Table VII, and \$ MM represents a million dollars.

Table 3-VIII: Per-unit of Load Cost

Bus	Load/MW	Load cost/ \$ MM	Per unit value (\$ MM/MW)
29	283.5	29.96495	0.10569647
16	329.4	37.64183	0.11427394
25	224	27.95146	0.12478329
4	500	75.21952	0.15043904
21	274	45.9506	0.16770291
24	308.6	52.68119	0.17071026
12	7.5	1.393788	0.18583839

The load at bus 29 appears to have the lowest per-unit load cost. Now we will need to perform the combination process in Python and select the combination with the lowest total load cost as our loading shedding solution. The pseudo-code for the proposed algorithm is displayed.

```
Greedy based sorting
1: for $i = 1, 2, \ldots, k$ do
2:     let combination = $C_k^i$
3:     if $\sum load\ in\ combination \in [T, T + th]$ then
4:         store combination in Set
5: end for
6: let min_cost = 1000000
7: for $combination\ in$ Set do
8:     let cost = $\sum_{load} Load\ Cost(load)$
9:     if cost < min_cost then
10:        min_cost = cost
11: end for
```

Figure 3-6: Pseudo-code for combination process.

In this figure, *k* represents the largest number for combination, which is 7, *T* is our load shedding target, and *th* refers to the threshold we set. By running the proposed algorithm, we identified a load shedding strategy that involved removing four loads: bus 29, bus 16, bus 25, and bus 4. These four loads totaled 1336.9 MW, which falls within the range of [*T*, *T* + *th*]. Notably, Table 3-VIII indicates that these four loads happen to have the lowest per-unit load cost. This result could be interpreted as the algorithm's greedy nature. However, it's also possible that this outcome was influenced by our selection of load shedding target and threshold.

Suppose we change the initial load shedding target from 1317.3 MW to 1000 MW while keeping the same threshold as 324 MW. As a result, the load shedding target range is [1000, 1324]. If we apply the proposed algorithm, it suggests shedding the loads at bus 25, bus 29, and bus 4. The sum of these three loads is 1007.5 MW, which falls within our range. Interestingly, the algorithm recommends shedding the load at bus 4 (the fourth row in Table 3-VIII) instead of bus 16 (the second row). This decision stems from the fact that the combined total load of the first three rows in the table is only 836.9 MW, which is much lower than the new load shedding target. If we were to add the load at the fourth row to this total, the resulting sum of 1336.9 MW would exceed the range of [1000, 1324] MW. Therefore, the algorithm settles on shedding the load at the fourth row instead of the load at the second row, striking a balance between meeting the new target and staying within the range.

The next thing we want to test is to find computing time improvement. The pseudo-code for the original sorting algorithm is presented.

```
Original sorting
1: for $i = 1, 2, \ldots, n$ do
2:     let combination = $C_n^i$
3:     if $\sum load\ in\ combination \in [T, T + th]$ then
4:         store combination in Set
5: end for
```

Figure 3-7: Pseudo-code for original sorting.

In this figure, the *n* represents the number of load buses in the system. To test the efficiency of the proposed algorithm, we ran both the proposed and original sorting algorithms six times and compared their respective running time. The comparison of computing time between these two algorithms (traditional greedy algorithm VS proposed algorithm) is displayed.

Table 3-IX: Comparison of Time Efficiency

Sequence	Original Sorting (s)	Proposed Algorithm (s)	Ratio
1	0.03119946	0.01196956	2.606567
2	0.03097517	0.00892901	3.469049
3	0.03048563	0.00997400	3.05651
4	0.03124166	0.007938147	3.935636
5	0.03124547	0.01495958	2.08866
6	0.03124189	0.00897789	3.47987

According to our theoretical calculation in previous equation, the computing time improvement resulting from the proposed algorithm is approximately 4.4 times faster than the original sorting algorithm. However, in Table 3-IX, the ratio between the proposed algorithm and the original sorting algorithm is slightly less than 4.4. This difference can be attributed to the second "for loop" in Figure 3-4. Even after identifying all suitable combinations in the first "for loop", the algorithm still needs to run a second loop to calculate the sum of load cost for each combination before it can select the combination with least load. This extra computational step has contributed to the difference between theoretical computing time improvement and actual improvement achieved.

Since the first and second stages of load shedding are sufficient to achieve power balance in this case study, implementing a third stage is unnecessary. With power balance achieved, we can separate the system into multiple partitions and ensure self-adequacy within each partition based on various algorithms discussed in chapter 2.

4 Economic Load Dispatch

The previous chapters made an assumption that the generation capacity remains constant, and when we consider the power balance within each partition, we will use generation capacity for consideration. In reality, power generation must align with economic load dispatch [82-86]. It is reasonable to assign generators with lower costs to produce more power. During the planning stage of dynamic microgrid operation, it is necessary to pre-determine the optimal generation output for each generator in order to maximize profit.

4.1 Problem Formulation

In order to achieve economic load dispatch among various generators, the generation cost function must be defined.

$$min \sum_{i=1}^{n} FC(P_i) \tag{35}$$

$$P_i \in [P_{i,min}, P_{i,max}] \tag{36}$$

$$FC(P_i) = a_i P_i^2 + b_i P_i + c_i + \left| e_i \times \sin\left(f_i \times (P_{i,min} - P_i)\right) \right| \tag{37}$$

where P_i is the power generated by the ith unit, $FC(P_i)$ represents the fuel cost of the generator. $P_{i,min}$ and $P_{i,max}$ are the lower and upper generation limits for the current generator. There are five parameters [87] defined in the fuel cost function: a_i, b_i, c_i, e_i, and f_i. The first three are the most commonly used in the fuel cost function, while e_i and f_i are considered based on the valve-point loadings of the generating units.

Traditionally when dealing with economic load dispatch problem, incremental cost has been most widely used in this area. If we could maintain equal incremental cost between different generators, we could guarantee an optimal solution is achieved. If the generation constraints are considered, we just have to set some generator output to their limits.

Since modern generators express strong nonlinearity in their input-output behavior due to factors such as valve-point loadings and rate limits. Traditional dispatch algorithms necessitate the use of approximations to account for these characteristics, but such approximations are not desirable in some practical problems, leading to significant long-term revenue loss. In order to deal with the non-linearity of fuel cost functions of generating units, heuristic algorithms and search algorithms might be more suitable for economic load dispatch problems.

4.2 Particle Swarm Optimization

Particle swarm optimization (PSO) [88-92] searches for a solution within the search space using a group of particles. Each particle in the group represents a potential solution, and we employ an objective function to evaluate its optimality. In each iteration, a particle adjusts its position towards the optimal solution based on the best solution found among the particles and the best solution that all particles have explored up to that point. The optimal solution is eventually achieved after numerous iterations.

Suppose there is a cost function *f(x, y)* that we either want to minimize or maximize. The first step of PSO is to randomly generate several particles.

$$P = [(x_1, y_1), (x_2, y_2), (x_3, y_3) \dots (x_m, y_m)] \tag{38}$$

where *P* represents the particles we generate, and the values within these particles represent their positions in a 2-dimensional space. In order to move these particles, we need to randomly generate initial velocities for each of them.

$$V = [(V_{x1}, V_{y1}), (V_{x2}, V_{y2}), (V_{x3}, V_{y3}) \dots (V_{xm}, V_{ym})] \tag{39}$$

Where *V* represents the velocity of all particles. Suppose the position for particle *i* at iteration *t* is:

$$P_i^t = [x_i^t, y_i^t] \tag{40}$$

The velocity for particle *i* at iteration t is:

$$V_i^t = [V_{x,i}^t, V_{y,i}^t] \tag{41}$$

The position for particle *i* at iteration *t* + 1 is:

$$P_i^{t+1} = [x_i^t + V_{x,i}^t, y_i^t + V_{y,i}^t] \tag{42}$$

At the same time, velocity for particle *i* at iteration *t* + 1:

$$V_i^{t+1} = wV_i^t + c_1 r_1 (P_{best} - P_i^t) + c_2 r_2 (G_{best} - P_i^t) \tag{43}$$

where *w* represents the inertia weight factor. A larger *w* implies that the new velocity will depend more on the previous velocity. P_{best} stands for the best position we've discovered for particle *i* in previous iterations at iteration *t*, while G_{best} represents the best position found among all particles in previous iterations. C_1 and c_2 are called cognitive and the social coefficients, where c_1 drives particle *i* toward its P_{best}, and c_2 drives particle *i* toward the G_{best}. They control how much weight should be given between refining the search result of the particle itself and recognizing the search result of the swarm. We can consider these parameters control the tradeoff between exploration and exploitation. R_1 and r_2 denote random numbers ranging from 0 to 1.

An interesting characteristic of this algorithm that sets it apart from other optimization methods is its independence from the gradient of the objective function. In gradient descent, for instance, we seek the function's minimum by heading in the direction of steepest descent. However, in the case of a particle next position, its movement isn't solely determined by the "downhill" direction; rather, it depends on the locations of "P_{best}" and "G_{best}." This property provides PSO especially well-suited for scenarios where differentiation for objective function is difficult. The flowchart of PSO is displayed.

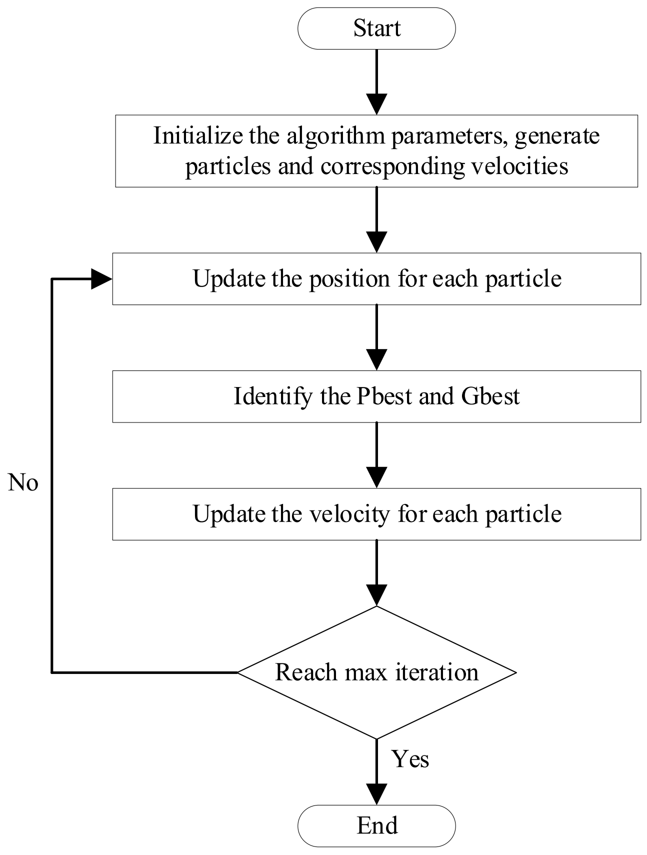

Start
Initialize the algorithm parameters, generate particles and corresponding velocities
Update the position for each particle
Identify the Pbest and Gbest
Update the velocity for each particle
Reach max iteration
No
Yes
End

Figure 4-1: Process of PSO.

4.3 Simulation

The test system was a three-generator system adapted from [87], their generating units' parameters are displayed in Table1.

Table 4-I: Parameters for Three Generators System

Generator	P_{min} (MW)	P_{max} (MW)	a	b	c	e	f
1	100	600	0.00156	7.92	561	300	0.0315
2	50	200	0.00482	7.97	78	150	0.063
3	100	400	0.00194	7.85	310	200	0.042

In this test system, the total load demand is 850 MW. The equation to update the velocity is:

$$V_i^{t+1} = wV_i^t + c_1r_1(P_{best} - P_i^t) + c_2r_2(G_{best} - P_i^t) \tag{44}$$

In this simulation, the parameters for PSO are swarm_size (number of swarms) =100, *w*=0.5, c_1=0.5, c_2=0.5, max iteration is set to 100. Based on the program in Python, the result generated result by PSO is:

Table 4-II: Results from PSO in Three Generator System

Simulation	Generator1(MW)	Generator2(MW)	Generator3(MW)	Cost ($)	Time
1	498.9756282	99.94603915	251.0783329	8241.34	0.12566
2	498.9356159	101.2066067	249.8577774	8242.74	0.1242
3	399.6804736	199.7174658	250.6020648	8251.13	0.12335
4	399.1533454	129.0329493	321.8137061	8361.76	0.12559
5	399.4492657	199.7470432	250.8037946	8250.87	0.1257
6	399.1993003	126.401218	324.3994817	8343.94	0.09996
7	599.71034	75.48773873	174.8020222	8480.86	0.13146
8	299.8726526	150.6075528	399.5197993	8234.8	0.12585
9	399.8794692	50.12166183	399.9988689	8242.07	0.09424
10	499.7922161	100.1455169	250.0622671	8242.98	0.11006
11	599.5364496	150.156325	100.3072256	8385.94	0.12035
12	498.9336183	176.2739528	174.7924298	8424.76	0.12571

We have conducted 12 simulations, and the minimum cost obtained from these simulations is 8234.8, which corresponds to simulation 8. This minimum cost aligns with the results reported in several papers. In [87], the author presents a table displaying minimum costs obtained from various algorithms, with the best result being 8234.07.

It is evident that each simulation generates different results due to the initialization of the swarm and velocity. These differences in the initial conditions lead the swarm in different directions, resulting in different solutions.

We have proved a small 3 generator system can work; we aim to analyze whether a larger system will also work. We extend the analysis to a larger system with 13 generators.

Table 4-III: Parameters for Thirteen Generators System

Generator	P_{min} (MW)	P_{max} (MW)	a	b	c	e	f
1	0	680	0.00028	8.1	550	300	0.035
2	0	360	0.00056	8.1	309	200	0.042
3	0	360	0.00056	8.1	307	200	0.042
4	60	180	0.00324	7.74	240	150	0.063
5	60	180	0.00324	7.74	240	150	0.063
6	60	180	0.00324	7.74	240	150	0.063
7	60	180	0.00324	7.74	240	150	0.063
8	60	180	0.00324	7.74	240	150	0.063
9	60	180	0.00324	7.74	240	150	0.063
10	40	120	0.00284	8.6	126	100	0.084
11	40	120	0.00284	8.6	126	100	0.084
12	55	120	0.00284	8.6	126	100	0.084
13	55	120	0.00284	8.6	126	100	0.084

The total demand is 1800 MW. This time the parameters for PSO stay the same: The parameters for PSO are swarm size=100, w=0.5, c_1=0.5, c_2=0.5, max iteration=100. The generated result from Python is:

Table 4-IV: Results from PSO in Thirteen Generator System

Simulation	Cost ($)	Time
1	18596.47159	0.22124052
2	18515.3101	0.19491291
3	18564.34267	0.18862915
4	18486.16615	0.210470438
5	18398.4136	0.2056036
6	18576.21752	0.190135002
7	18699.26961	0.209335804
8	18383.22355	0.20947957
9	18579.94865	0.200233459
10	18472.57628	0.200392962
11	18583.64757	0.203057528
12	18492.70441	0.210788965

The least cost from this simulation is 18383.22355, which corresponds to the 8th simulation. But the best result I got is still higher than the result in previous papers. Hence, I changed the iteration from 100 to 500.

Table 4-V: Results from 500 Iterations

Simulation	Cost ($)	Time
1	18619.7	0.4125
2	18729	0.3146
3	18440.7	0.73276
4	18327.1	0.41446
5	18429.6	0.4458
6	18546.9	0.45755
7	18384.9	0.27324
8	18709.3	0.60099
9	18127.7	0.40489
10	18422.6	0.7303
11	18454.5	0.41179
12	18195.7	1.02193

The best result I obtained is 18127.65126, corresponding to the 9th simulation. This result closely aligns with the calculation in previous papers. It is evident that if we increase the iterations, the probability of getting a better value is increased.

A higher cognitive coefficient c_1 means the particles tend to follow their own best position and a higher social coefficient, c_2 means the particles tend to follow the best position of their neighbors. It is reasonable to decrease c_1 and increase c_2. This time, we changed the parameters to swarm size=100, w=0.5, c_1=0.3, c_2=0.6, and max iteration=500. The result we get is:

Table 4-VI: Results from 500 Iterations after Changing Parameters

Simulation	Cost ($)	Time
1	18678.1	0.49303
2	18523.6	0.31957
3	18533.5	0.47811
4	18462.7	0.5173

5	18489.3	0.43166
6	18604.9	0.75589
7	18450.6	0.53202
8	18406.4	0.44536
9	18059.1	0.55483
10	18493.2	0.5699
11	18468	0.23615
12	18419.8	0.50819

From this table, we observe that the optimal result we've achieved is 18059.0611, which corresponds to the 9th simulation, and is close to the result obtained from [87], which is 18048.21. Notably, this minimum cost is lower than the result obtained when $c_1 = 0.5$ and $c_2 = 0.5$. A more general approach involves decreasing c_1 linearly as the iterations go and increasing c_2 linearly with each iteration [93]. This strategy is based on the assumption that with more iterations, we move closer to the optimal result.

5. Conclusions and Future Work

This research primarily addresses the reliable operation of multiple dynamic microgrids during the planning stage. Three partitioning algorithms have been proposed in this research, each with different objective functions. The constraints for these partitioning methods are the same, focusing on supply-demand balance and having at least one grid-forming generator within each partition. The first two algorithms prioritize minimizing power exchange between different partitions to ensure a smooth transition from a centralized system to a decentralized one. In the second algorithm, K-means clustering was employed to perform the partitioning step, overcoming the limitations of the original k-way partitioning. Simulation results from the IEEE 33 Radial Bus System and IEEE 39-Bus system were used to demonstrate the effectiveness of these proposed algorithms. Since the original data used in these two algorithms is based on active power flow, and in reality, power flow is constantly changing, we can achieve dynamic separation methods, which corresponds to the nature of dynamic microgrids. Future research will focus on implementing the proposed algorithms in larger systems and improving their computational efficiency. The third algorithm concentrates on minimizing power losses after partitioning, using electrical distance as the index for power loss. This modified algorithm offers several benefits, including reduced voltage drop along the connection. Future research could focus on recording nodes on the shortest path when coding the algorithm in Python.

In islanded mode, the primary concern is addressing the load-shedding problem. The proposed load-shedding strategy aims to secure as much critical load as possible and minimize the cost of shedding load. Once the power balance is restored after the load-shedding process, the algorithms discussed in Chapter 2 can be applied to ensure reliable partitioning of the distribution system. Future research will focus on implementing the proposed algorithms in larger systems, such as the IEEE 57-Bus System or IEEE 118-Bus System. Additionally, efforts will be made to enhance the time and space efficiency of the proposed algorithm.

As for the economic load dispatch, our goal is to achieve economic system operation during the planning stage. We introduce a nonlinear equation to incorporate more factors into the fuel cost function. In contrast to traditional gradient descent methodologies, particle swarm optimization does not require differentiation concerning variables. However, there is a risk that particle swarm optimization might get trapped in local optima due to initial speed and velocity. Future research could focus on strategies for escaping local optima through simulated annealing.

6. References

[1] K. Shi, H. Ye, W. Song and G. Zhou, "Virtual Inertia Control Strategy in Microgrid Based on Virtual Synchronous Generator Technology," IEEE Access, vol. 6, pp. 27949-27957, 2018, doi: 10.1109/ACCESS.2018.2839737.

[2] T. Kerdphol, F. S. Rahman, M. Watanabe and Y. Mitani, "Robust Virtual Inertia Control of a Low Inertia Microgrid Considering Frequency Measurement Effects," IEEE Access, vol. 7, pp. 57550–57560, 2019, doi: 10.1109/ACCESS.2019.2913042.

[3] IEEE Guide for Design, Operation, and Integration of Distributed Resource Island Systems With Electric Power Systems, IEEE Std 1547.4, Jul.20, 2011.

[4] W. Qiu et al., "A Grid Forming/Following Sequence Switching Control Strategy for Supporting Frequency Stability of Isolated Power Grids," 2023 5th Asia Energy and Electrical Engineering Symposium (AEEES), Chengdu, China, 2023, pp. 212-217, doi: 10.1109/AEEES56888.2023.10114155.

[5] Y. Xu, H. Nian, J. Kang, J. Zhao, Z. Wang and J. Zhou, "Impedance-based Analysis of Potential Stability Risk Between Grid-Forming and Grid-Following Wind Turbine Systems," 2021 6th Asia Conference on Power and Electrical Engineering (ACPEE), Chongqing, China, 2021, pp. 858-862, doi: 10.1109/ACPEE51499.2021.9437139.

[6] Z. Zhou, W. Wang, T. Lan and G. M. Huang, "Dynamic Performance Evaluation of Grid-Following and Grid-Forming Inverters for Frequency Support in Low Inertia Transmission Grids," 2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe), Espoo, Finland, 2021, pp. 01-05, doi: 10.1109/ISGTEurope52324.2021.9640034.

[7] S. Uprety and H. Lee, "22.5 A 93%-power-efficiency photovoltaic energy harvester with irradiance-aware auto-reconfigurable MPPT scheme achieving >95% MPPT efficiency across

650μW to 1W and 2.9ms FOCV MPPT transient time," 2017 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 2017, pp. 378-379, doi: 10.1109/ISSCC.2017.7870419.

[8] T. Latif and S. R. Hussain, "Design of a charge controller based on SEPIC and buck topology using modified Incremental Conductance MPPT," 8th International Conference on Electrical and Computer Engineering, Dhaka, Bangladesh, 2014, pp. 824-827, doi: 10.1109/ICECE.2014.7026999.

[9] Z. N. Popovic, S. D. Knezevic and B. S. Brbaklic, "A Risk Management Procedure for Island Partitioning of Automated Radial Distribution Networks with Distributed Generators," IEEE Trans. Power Syst. vol. 35, no. 5, pp. 3895-3905, Sept. 2020, doi: 10.1109/TPWRS.2020.2976763.

[10] V. Hosseinnezhad, M. Rafiee, M. Ahmadian, and P. Siano, “Optimal Island partitioning of smart distribution systems to improve system restoration under emergency conditions,” Int. J. Elect. Power Energy Syst., vol. 97, pp. 155–164, Apr. 2018, doi: org/10.1016/j.ijepes.2017.11.003.

[11] A. Khodaei, "Microgrid optimal scheduling with multi-period islanding constraints," IEEE Trans Power Syst., vol. 29, no. 3, pp. 1383-1392, May 2014, doi: 10.1109/TPWRS.2013.2290006.

[12] W. T. El-Sayed, H. E. Z. Farag, H. H. Zeineldin and E. F. El-Saadany, "Formation of Islanded Droop-Based Microgrids with Optimum Loadability," IEEE Trans. Power Syst., vol. 37, no. 2, pp. 1564-1576, March 2022, doi: 10.1109/TPWRS.2021.3099691.

[13] E. Lázár, R. Etz, D. Petreuş, T. Pătărău and I. Ciocan, "SCADA development for an islanded microgrid," 2015 IEEE 21st International Symposium for Design and Technology in

Electronic Packaging (SIITME), Brasov, Romania, 2015, pp. 147-150, doi: 10.1109/SIITME.2015.7342314.

[14] K. M. Tofani, P. A. A. Pramana, B. B. S. D. A. Harsono, D. R. Jintaka and K. G. H Mangunnkusumo, "SCADA Systems Design to Optimize and Automate Microgrids Systems in Indonesia," 2020 International Conference on Technology and Policy in Energy and Electric Power (ICT-PEP), Bandung, Indonesia, 2020, pp. 187-192, doi: 10.1109/ICT-PEP50916.2020.9249833.

[15] R. H. Lasseter, "Smart Distribution: Coupled Microgrids," Proceedings of the IEEE, vol. 99, no. 6, pp. 1074-1082, June 2011. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=655b0d7c6ce23e93e4ae99747a640793e3be852e.

[16] H. Al-Saadi, R. Zivanovic, and S. F. Al-Sarawi, "Probabilistic hosting capacity for active distribution networks," IEEE Transactions on Industrial Informatics., vol. 13, no. 5, pp. 2519-2532, Oct. 2017, doi: 10.1109/TSG.2020.3038651.

[17] S. Wang, Z. Li, L. Wu, M. Shahidehpour, and Z. Li, "New metrics for assessing the reliability and economics of microgrids in distribution system," IEEE Transactions on Power Systems., vol. 28, no. 3, pp. 2852–2861, Aug. 2013 , doi: 10.1109/TPWRS.2013.2249539.

[18] F. Farzan, S. Lahiri, M. Klienberg, K. Gharieh, F. Farzan, and M. Jafari," Microgrids for fun and profit: The economics of installation investments and operations," IEEE Power and Energy Magazine., vol. 11, no. 4, pp. 52–58, Jul./Aug. 2013, doi: 10.1109/MPE.2013.2258282.

[19] M. E. Nassar and M. M. A. Salama, "Adaptive Self-Adequate Microgrids Using Dynamic Boundaries," in IEEE Transactions on Smart Grid, vol. 7, no. 1, pp. 105-113, Jan. 2016, doi: 10.1109/TSG.2015.2420115.

[20] Y. Ma et al., "Real-Time Control and Operation for a Flexible Microgrid with Dynamic Boundary," 2018 IEEE Energy Conversion Congress and Exposition (ECCE), Portland, OR, USA, 2018, pp. 5158-5163, doi: 10.1109/ECCE.2018.8558364.

[21] S. Zhen, Y. Ma, F. Wang and L. M. Tolbert, "Operation of a Flexible Dynamic Boundary Microgrid with Multiple Islands," 2019 IEEE Applied Power Electronics Conference and Exposition (APEC), Anaheim, CA, USA, 2019, pp. 548-554, doi: 10.1109/APEC.2019.8721836.

[22] Y. Du, X. Lu, J. Wang and S. Lukic, "Distributed Secondary Control Strategy for Microgrid Operation with Dynamic Boundaries," IEEE Transactions on Smart Grid, vol. 10, no. 5, pp. 5269-5282, Sept. 2019, doi:10.1109/TSG.2018.2879793.

[23] N. Manna and A. Kumar Sil, "Multiple Objective Modelling by Forming Dynamic Clusters of Peak Loads and Distributed Generations for Energy Management in Grid Connected Mode," 2020 IEEE Applied Signal Processing Conference (ASPCON), Kolkata, 2020, pp. 46-50, doi: 10.1109/ASPCON49795.2020.9276667.

[24] N. Rahbari-Asr, Yuan Zhang and M. -Y. Chow, "Cooperative distributed energy scheduling for storage devices and renewables with resiliency against intermittencies," 2016 IEEE 25th International Symposium on Industrial Electronics (ISIE), Santa Clara, CA, USA, 2016, pp. 612-617, doi: 10.1109/ISIE.2016.7744959.

[25] N. A. Binte Shaikh Fauzan, R. T. Naayagi, T. Logenthiran and V. -T. Phan, "Integration of battery energy storage using single phase inverter for intermittency mitigation," 2016 IEEE Region 10 Conference (TENCON), Singapore, 2016, pp. 921-925, doi: 10.1109/TENCON.2016.7848139.

[26] F. S. Gazijahani and J. Salehi, "Optimal bilevel model for stochastic riskbased planning of microgrids under uncertainty," IEEE Transactions on Industrial Informatics., vol. 14, no. 7, pp. 3054–3064, Jul. 2018, doi: 10.1109/TII.2017.2769656.

[27] R. A. Osama, A. F. Zobaa and A. Y. Abdelaziz, "A Planning Framework for Optimal Partitioning of Distribution Networks Into Microgrids," IEEE Systems Journal., vol. 14, no. 1, pp. 916-926, Mar. 2020 , doi: 10.1109/JSYST.2019.2904319.

[28] M. Barani, J. Aghaei, M. A. Akbari, T. Niknam, H. Farahmand and M. Korpås, "Optimal Partitioning of Smart Distribution Systems Into Supply-Sufficient Microgrids," IEEE Transactions on Smart Grid, vol. 10, no. 3, pp. 2523-2533, May 2019, doi: 10.1109/PowerTech46648.2021.9494956.

[29] S. A. Arefifar, Y. A. I. Mohamed, and T. H. M. El-Fouly, "Supply adequacy-based optimal construction of microgrids in smart distribution systems," IEEE Transactions on Smart Grid., vol. 3, no. 3, pp. 1491–1502, Sep. 2012, doi: 10.1109/TSG.2012.2198246.

[30] H. Wang, H. Zhang, L. Chen, L. Chen, Y. Zhao and B. Duan, "A Method of Distribution Reconfiguration with Micro Grid Considering Dynamic Behavior of Thermal Loads," 2018 International Conference on Smart Grid and Electrical Automation (ICSGEA), 2018, pp. 1-5, doi: 10.1109/ICSGEA.2018.00009.

[31] S. M. Ferdous, F. Shahnia and G. M. Shafiullah, "Stability and robustness of a coupled microgrid cluster formed by various coupling structures," Chinese Journal of Electrical Engineering, vol. 7, no. 4, pp. 60-77, Dec. 2021, doi: 10.23919/CJEE.2021.000038.

[32] Z. Zhang, Z. Wang, H. Wang, H. Zhang, W. Yang and R. Cao, "Research on Bi-Level Optimized Operation Strategy of Microgrid Cluster Based on IABC Algorithm," IEEE Access, vol. 9, pp. 15520-15529, 2021, doi: 10.1109/ACCESS.2021.3053122.

[33] A. K. Verma, S. Sharma, "Network Partitioning for Parallel Power System Operation," 2021 IEEE 2nd International Conference on Electrical Power and Energy Systems (ICEPES), pp. 1-6, 2021, doi: 10.1109/ICEPES52894.2021.9699680.

[34] Hao Li, G. W. Rosenwald, J. Jung and Chen-ching Liu, "Strategic Power Infrastructure Defense," in Proceedings of the IEEE, vol. 93, no. 5, pp. 918-933, May 2005, doi: 10.1109/JPROC.2005.847260.

[35] C. H. Lee, M. Kim and C. I. Park, "An efficient k-way graph partitioning algorithm for task allocation in parallel computing systems," Systems Integration '90. Proceedings of the First International Conference on Systems Integration, Morristown, NJ, USA, 1990, pp. 748-751, doi: 10.1109/ICSI.1990.138741.

[36] P. K. Chan, M. D. F. Schlag, and J. Y. Zien, "Spectral k-way ratiocut partitioning and clustering," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 9, pp. 1088–1096, Sep 1994, doi: 10.1109/43.310898.

[37] M. Pietsch, A. Klein and F. Steinke, "Merging Microgrids for Optimal Distribution Grid Restoration under Explicit Communication Constraints," 2020 Resilience Week (RWS), Salt Lake City, UT, USA, 2020, pp. 48-54, doi: 10.1109/RWS50334.2020.9241251.

[38] Y. He, Q. Yu, Z. Cao, Y. Chen, Z. Shen and K. Zhang, "A Local Energy Transaction Strategy for Networked Microgrids Under Coalitional Game Theory," 2022 12th International Conference on Power and Energy Systems (ICPES), Guangzhou, China, 2022, pp. 429-433, doi: 10.1109/ICPES56491.2022.10073417.

[39] K. P. Swaroop, D. P. Garapati, P. K. Nalli and S. S. Duvvuri, "Service Restoration in Distribution System Using Breadth-First Search Technique," 2021 7th International

Conference on Electrical Energy Systems (ICEES), 2021, pp. 403-407, doi: 10.1109/ICEES51510.2021.9383670.

[40] T. Hubana, "Artificial Intelligence based Station Protection Concept for Medium Voltage Microgrids," 2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH), 2020, pp. 1-6, doi: 10.1109/INFOTEH48170.2020.9066305.

[41] Bike Xue, Fei Shi, Xinhong Shi, Jie Yu and Lili Liu, "Transmission path optimization method combining sensitivity analysis with breadth first search theory suitable for large scale direct electricity trade," 2016 IEEE International Conference on Power and Renewable Energy (ICPRE), Shanghai, 2016, pp. 481-484, doi: 10.1109/ICPRE.2016.7871257.

[42] Y. Kaneko, "Enhanced breadth first search-based routing in wireless sensor networks," 2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Phuket, Thailand, 2022, pp. 1-4, doi: 10.1109/ITC-CSCC55581.2022.9894872.

[43] H. Khalil and Y. Labiche, "Finding All Breadth First Full Spanning Trees in a Directed Graph," 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy, 2017, pp. 372-377, doi: 10.1109/COMPSAC.2017.128.

[44] T. Chongphipatmongkol and K. Audomvongseree, "Determination of Reserve Margin based on Specified Loss of Load Expectation," 2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Chiang Rai, Thailand, 2018, pp. 644-647, doi: 10.1109/ECTICon.2018.8619932.

[45] R. Diewvilai, R. Nidhiritdhikrai and B. Eua-arporn, "Reserve margin evaluation for generation system using probabilistic based method," The 8th Electrical Engineering/

Electronics, Computer, Telecommunications and Information Technology (ECTI) Association of Thailand - Conference 2011, Khon Kaen, Thailand, 2011, pp. 905-908, doi: 10.1109/ECTICON.2011.5947987.

[46] X. Zhu, J. Kim, E. Muljadi and R. M. Nelms, "Optimal Separation Method of Dynamic Microgrid Operation," 2021 IEEE Energy Conversion Congress and Exposition (ECCE), Vancouver, BC, Canada, 2021, pp. 1110-1115, doi: 10.1109/ECCE47101.2021.9595609.

[47] O. Penangsang, D. F. U. Putra and T. Kurniawan, "Optimal placement and sizing of distributed generation in radial distribution system using K-means clustering method," 2017 International Seminar on Intelligent Technology and Its Applications (ISITIA), Surabaya, 2017, pp. 98-103, doi: 10.1109/ISITIA.2017.8124062.

[48] N. V. V. K. Jureedi, N. Prema Kumar and K. M. Rosalina, "Clustering Analysis and its Application in Electrical Distribution System," International Journal of Recent Advances in Engineering & Technology, vol. 1, no. 1, pp. 130–136, 2013, doi: 10.46564/ijraet.2020.v08i06.006.

[49] W. Yanbo, L. Li, P. Xinfu and F. Enpeng, "Load Forecasting Based on Improved K-means Clustering Algorithm," 2018 China International Conference on Electricity Distribution (CICED), Tianjin, 2018, pp. 2751-2755, doi: 10.1109/CICED.2018.8592023.

[50] X. Zhu, J. Kim, E. Muljadi and R. M. Nelms, "Dynamic Separation of Microgrid System to Maximize Reliability in a Smart Grid," 2021 IEEE Green Technologies Conference (GreenTech), Denver, CO, USA, 2021, pp. 232-236, doi: 10.1109/GreenTech48523.2021.00045.

[51] B. N. Ha, S. W. Lee, N. H. Cho, W. Namkoong, J. W. Yoon and I. H. Lim, "Loss minimization and load balancing in a distribution network," 2015 International Symposium on

Smart Electric Distribution Systems and Technologies (EDST), Vienna, Austria, 2015, pp. 114-120, doi: 10.1109/SEDST.2015.7315193.

[52] H. Lasso, C. Ascanio and M. Guglia, "A Model for Calculating Technical Losses in the Secondary Energy Distribution Network," 2006 IEEE/PES Transmission & Distribution Conference and Exposition: Latin America, Caracas, Venezuela, 2006, pp. 1-6, doi: 10.1109/TDCLA.2006.311651.

[53] B. Chen, K. Xiang, L. Yang, Q. Su, D. Huang and T. Huang, "Theoretical Line Loss Calculation of Distribution Network Based on the Integrated Electricity and Line Loss Management System," 2018 China International Conference on Electricity Distribution (CICED), Tianjin, China, 2018, pp. 2531-2535, doi: 10.1109/CICED.2018.8592309.

[54] T. D. Sudhakar, N. S. Vadivoo, S. M. R. Slochanal and S. Ravichandran, "Supply restoration in distribution networks using Dijkstra's algorithm," 2004 International Conference on Power System Technology, 2004. PowerCon 2004., Singapore, 2004, pp. 640-645 Vol.1, doi: 10.1109/ICPST.2004.1460073.

[55] P. Prabawa and D. -H. Choi, "Multi-Agent Framework for Service Restoration in Distribution Systems With Distributed Generators and Static/Mobile Energy Storage Systems," in IEEE Access, vol. 8, pp. 51736-51752, 2020, doi: 10.1109/ACCESS.2020.2980544.

[56] Z. Jiangfeng et al., "Research on Decision-Making optimization for Operation System of Power Grid Framework under Smart Grid and Dijkstra Algorithm," 2021 IEEE International Conference on Data Science and Computer Application (ICDSCA), Dalian, China, 2021, pp. 378-382, doi: 10.1109/ICDSCA53499.2021.9650175.

[57] S. Subrahmanyam V, S. K. Jain and G. Narayanan, "Real-time Simulation of IEEE 10-Generator 39-Bus System with Power System Stabilizers on Miniature Full Spectrum Simulator," 2019 IEEE International Conference on Sustainable Energy Technologies and Systems (ICSETS), Bhubaneswar, India, 2019, pp. 161-166, doi: 10.1109/ICSETS.2019.8745001.

[58] C. A. Bartend Russell and S. Khan, "Single Line Outage Analysis on IEEE 39 Bus Network," 2022 25th International Conference on Computer and Information Technology (ICCIT), Cox's Bazar, Bangladesh, 2022, pp. 821-826, doi: 10.1109/ICCIT57492.2022.10055232.

[59] R. A. Osama, A. Y. Abdelaziz, R. A. Swief and M. Ezzat, "Microgrid self adequacy optimization using back tracking search algorithm," 2015 IEEE Power & Energy Society General Meeting, Denver, CO, USA, 2015, pp. 1-5, doi: 10.1109/PESGM.2015.7285881.

[60] R. Jovanovic and A. Bousselham, "A greedy method for optimizing the self-adequacy of microgrids presented as partitioning of graphs with supply and demand," 2014 International Renewable and Sustainable Energy Conference (IRSEC), Ouarzazate, Morocco, 2014, pp. 565-570, doi: 10.1109/IRSEC.2014.7059835.

[61] M. E. Nassar and M. M. A. Salama, "Adaptive Self-Adequate Microgrids Using Dynamic Boundaries," in IEEE Transactions on Smart Grid, vol. 7, no. 1, pp. 105-113, Jan. 2016, doi: 10.1109/TSG.2015.2420115.

[62] X. Zhou, L. Ning, B. Wang, C. Yang, H. Sun and Y. Liu, "An Industrial Park Load Shedding Scheme Based on Smart Contract," 2022 IEEE 6th Conference on Energy Internet and Energy System Integration (EI2), Chengdu, China, 2022, pp. 1811-1816, doi: 10.1109/EI256261.2022.10116235.

[63] S. Xing, "Microgrid emergency control based on the stratified controllable load shedding optimization," International Conference on Sustainable Power Generation and Supply (SUPERGEN 2012), Hangzhou, 2012, pp. 1-5, doi: 10.1049/cp.2012.1778.

[64] R. Billinton and P. Wang, "Distribution system reliability cost/worth analysis using analytical and sequential simulation techniques," IEEE Trans. Power Syst., vol. 13, no. 4, pp. 1245-1250, Nov. 1998 , doi: 10.1109/59.736248.

[65] A. Zerka, M. Ouassaid, M. Maaroufi and R. Rabeh, "Energy Exchange Management in Smart Grids Using a Knapsack Problem Inspired Approach," 2022 IEEE 21st Mediterranean Electrotechnical Conference (MELECON), Palermo, Italy, 2022, pp. 1034-1039, doi: 10.1109/MELECON53508.2022.9842885.

[66] A. L. Maharsi, F. D. Wijaya and I. W. Mustika, "Cost-based power distribution optimization scheduling in microgrid," 2017 3rd International Conference on Science and Technology - Computer (ICST), Yogyakarta, Indonesia, 2017, pp. 87-92, doi: 10.1109/ICSTC.2017.8011858.

[67] Tran Dang Khoa, L. T. Dos Santos, M. Sechilariu and F. Locment, "Load shedding and restoration real-time optimization for DC microgrid power balancing," 2016 IEEE International Energy Conference (ENERGYCON), Leuven, Belgium, 2016, pp. 1-6, doi: 10.1109/ENERGYCON.2016.7514092.

[68] W. Zhuo, "Control of a networked microgrid system with an approximate dynamic programming approach," 2019 Chinese Control Conference (CCC), Guangzhou, China, 2019, pp. 6571-6576, doi: 10.23919/ChiCC.2019.8865864.

[69] V. S. Borra and K. Debnath, "Dynamic programming for solving unit commitment and security problems in microgrid systems," 2018 IEEE International Conference on Innovative

Research and Development (ICIRD), Bangkok, Thailand, 2018, pp. 1-6, doi: 10.1109/ICIRD.2018.8376313.

[70] W. Chouaf, A. Abbou and A. Agga, "Optimal energy management for a connected microgrid using dynamic programming method," 2022 8th International Conference on Optimization and Applications (ICOA), Genoa, Italy, 2022, pp. 1-6, doi: 10.1109/ICOA55659.2022.9934437.

[71] R. Jovanovic and A. Bousselham, "A greedy method for optimizing the self-adequacy of microgrids presented as partitioning of graphs with supply and demand," 2014 International Renewable and Sustainable Energy Conference (IRSEC), Ouarzazate, Morocco, 2014, pp. 565-570, doi: 10.1109/IRSEC.2014.7059835.

[72] Z. Jiang, V. Sahasrabudhe, H. Grebel, A. Mohamed and R. Rojas-Cessa, "Greedy Algorithm for Routing Power and Source Assignment on a Digital Microgrid," 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, GA, USA, 2019, pp. 761-767, doi: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00141.

[73] A. Zerka, M. Ouassaid, M. Maaroufi and R. Rabeh, "Energy Exchange Management in Smart Grids Using a Knapsack Problem Inspired Approach," 2022 IEEE 21st Mediterranean Electrotechnical Conference (MELECON), Palermo, Italy, 2022, pp. 1034-1039, doi: 10.1109/MELECON53508.2022.9842885.

[74] S. Xing, "Microgrid emergency control based on the stratified controllable load shedding optimization," International Conference on Sustainable Power Generation and Supply (SUPERGEN 2012), Hangzhou, 2012, pp. 1-5, doi: 10.1049/cp.2012.1778.

[75] Z. Pan, J. Wang, J. Liu and J. Hou, "Friendly Load Control Scheme with Importance Degree and Intention," 2021 IEEE 12th Energy Conversion Congress & Exposition - Asia (ECCE-Asia), Singapore, Singapore, 2021, pp. 2075-2079, doi: 10.1109/ECCE-Asia49820.2021.9479147.

[76] T. Zhou, Z. Zhou and X. Li, "Vehicle-to-Grid Based Load Recovery Strategy for Power Network Considering Load Importance," 2023 IEEE PELS Students and Young Professionals Symposium (SYPS), Shanghai, China, 2023, pp. 1-5, doi: 10.1109/SYPS59767.2023.10268188.

[77] Q. Gao, C. Zhu, X. Zhang, N. Jiang and Q. Wu, "Partition load reduction algorithm based on Contribution coefficient and load importance," 2022 China International Conference on Electricity Distribution (CICED), Changsha, China, 2022, pp. 880-888, doi: 10.1109/CICED56215.2022.9929104.

[78] C. Li et al., "Active Control of Flexible Power Electronic Load Considering Importance Degree in Microgrid," 2018 IEEE International Power Electronics and Application Conference and Exposition (PEAC), Shenzhen, China, 2018, pp. 1-6, doi: 10.1109/PEAC.2018.8590554.

[79] I. Kim and D. Kim, "Pricing-based shared energy storage optimization for residential users with photovoltaic generation system and demand-side load management," 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN), Zagreb, Croatia, 2019, pp. 466-471, doi: 10.1109/ICUFN.2019.8806069.

[80] J. Ma, H. Yu, J. Zhou, Y. Peng, C. Lu and G. Geng, "Edge-End Based Demand Response System of Aggregated Residential Load: Case Study," 2022 5th International Conference on Energy, Electrical and Power Engineering (CEEPE), Chongqing, China, 2022, pp. 812-816, doi: 10.1109/CEEPE55110.2022.9783269.

[81] K. Bruninx, H. Pandžić, H. Le Cadre and E. Delarue, "On the Interaction Between Aggregators, Electricity Markets and Residential Demand Response Providers," in IEEE Transactions on Power Systems, vol. 35, no. 2, pp. 840-853, March 2020, doi: 10.1109/TPWRS.2019.2943670.

[82] D. Sun et al., "Integrated Generation-Grid-Load Economic Dispatch Considering Demand Response," 2020 IEEE/IAS Industrial and Commercial Power System Asia (I&CPS Asia), Weihai, China, 2020, pp. 375-379, doi: 10.1109/ICPSAsia48933.2020.9208351.

[83] T. Jianxing, S. Bin, C. Rui, Y. Yao and W. Guosong, "Economic Dispatching Method Based on Dynamic Network Loss Factor," 2018 International Conference on Power System Technology (POWERCON), Guangzhou, China, 2018, pp. 601-605, doi: 10.1109/POWERCON.2018.8601865.

[84] Guo-Li Zhang, Geng-Yin Li, Hong Xie and Jian-Wei Ma, "Environmental/economic load dispatch based on weighted ideal point and hybrid evolutionary algorithm," 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 2005, pp. 2466-2471 Vol. 4, doi: 10.1109/ICMLC.2005.1527358.

[85] Wen Liang, M. Liu, Fangchao Song, Wencheng Wu, Kai Zhou and Aimin Jin, "Power system dynamic economic dispatch with controllable air-conditioning load groups," 2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), Xi'an, 2016, pp. 962-967, doi: 10.1109/APPEEC.2016.7779637.

[86] Y. Wu, M. Wang, Y. Song and X. Zhang, "Economic Dispatch Considering Flexibility Provided by Load Static Voltage Characteristics," 2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2), Changsha, China, 2019, pp. 1980-1985, doi: 10.1109/EI247390.2019.9061919.

[87] N. Sinha, R. Chakrabarti and P. K. Chattopadhyay, "Evolutionary programming techniques for economic load dispatch," in IEEE Transactions on Evolutionary Computation, vol. 7, no. 1, pp. 83-94, Feb. 2003, doi: 10.1109/TEVC.2002.806788.

[88] PhanTu Vu, DinhLuong Le, NgocDieu Vo and J. Tlusty, "A novel weight-improved particle swarm optimization algorithm for optimal power flow and economic load dispatch problems," IEEE PES T&D 2010, New Orleans, LA, USA, 2010, pp. 1-7, doi: 10.1109/TDC.2010.5484396.

[89] N. J. Singh, J. S. Dhillon and D. P. Kothari, "Integrated particle swarm optimization variants for economic load dispatch problem," 2016 7th India International Conference on Power Electronics (IICPE), Patiala, India, 2016, pp. 1-5, doi: 10.1109/IICPE.2016.8079530.

[90] N. Kumar, U. Nangia and K. B. Sahay, "Economic load dispatch using improved particle swarm optimization algorithms," 2014 6th IEEE Power India International Conference (PIICON), Delhi, India, 2014, pp. 1-6, doi: 10.1109/POWERI.2014.7117665.

[91] X. Ma and Y. Liu, "Particle swarm optimization to solving economic load dispatch with spinning reserve," 2010 International Conference On Computer Design and Applications, Qinhuangdao, China, 2010, pp. V4-214-V4-217, doi: 10.1109/ICCDA.2010.5541218.

[92] I. Alzubi, H. M. K. Al-Masri and A. Abuelrub, "Modified Particle Swarm Optimization Algorithms for Solving Economic Load Dispatch," 2022 3rd International Conference on Smart Grid and Renewable Energy (SGRE), Doha, Qatar, 2022, pp. 1-5, doi: 10.1109/SGRE53517.2022.9774126.

[93] H. -R. Li and Y. -L. Gao, "Particle Swarm Optimization Algorithm with Exponent Decreasing Inertia Weight and Stochastic Mutation," 2009 Second International Conference

on Information and Computing Science, Manchester, UK, 2009, pp. 66-69, doi: 10.1109/ICIC.2009.24.