Heuristic Methods and Machine Learning for Distribution Network Reconfiguration

Auteur : Kaci Touati, Melissa
Promoteur(s) : Louveaux, Quentin
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil électricien, à finalité spécialisée "Smart grids"
Année académique : 2024-2025
URI/URL : http://hdl.handle.net/2268.2/23274

UNIVERSITE DE LIEGE
FACULTE DES SCIENCES APPLIQUEES

Heuristic Methods and Machine Learning for Distribution Network Reconfiguration

Kaci Touati Melissa

Thesis presented to obtain the degree of :
Master of Science in Electrical Engineering

Thesis supervisor :
Louveaux Quentin

Academic year: **2024 - 2025**

Acknowledgments

This thesis concludes my studies in electrical engineering. First of all, I would like to thank my thesis supervisor, Professor Quentin Louveaux, for supervising the entire thesis and for all his advice, guidance, and feedbacks. I would also like to thank Dr. Mevludin Glavic for his research suggestions and expertise in electrical networks.

I want to deeply thank my mother for her presence and unconditional support throughout all these years of study, as well as my little sister and my big brother.

Of course, I would like to thank all the people who made these years truly amazing. Thank you, Florine, for all the time we have spent together, for these last five intense months at university, and for our never-ending conversations. I also want to thank Chloé for her kindness, her support, and all the unforgettable parties we shared. Finally, I'd like to thank my close group of friends: Charles, Tom, Anaïs, Brice, Rayane and the team Peach: Lucie, Maria, and Clément, for the incredible entrepreneurial experience we lived together during our studies.

Last but not least, I would like to thank my boyfriend Adrien, not only for his support, but also for his very professional proofreading.

Summary

The problem of distribution network reconfiguration (DNR) has been addressed for some fourty years in scientific literature and industry. The aim of reconfiguration is to find the best radial operational configuration in a given electrical state. The main objectives are to reduce active losses and improve the profile. The reconfiguration problem is combinatorial and non-linear, making it practically impossible to solve for real-size electrical networks. To counter this problem, numerous heuristics and metaheuristics have been proposed in the literature.

In this thesis, we explore and compare different strategies of reconfiguration. The first method used is based on a minimum spanning tree algorithm to generate an initial solution. This solution is then refined using two heuristics: local search (LS) and tabu search (TS). Finally, a machine learning model is trained to mimic the behavior of these heuristics. This model uses various features to predict whether or not a line is part of the optimal configuration.

All methods are tested on different scenarios with and without DGs from the IEEE 33 and IEEE 69 networks. Results show that combining classical heuristics with learning-based approaches provides a balance between performance and computational cost, especially for applications requiring fast decision-making. This work contributes to ongoing research into the reconfiguration of constantly evolving electrical distribution networks.

Résumé

Le problème de la reconfiguration des réseaux de distribution (DNR) est abordé depuis une quarantaine d'années dans la littérature scientifique et dans l'industrie. L'objectif de la reconfiguration est de trouver la meilleure configuration opérationnelle radiale dans un état électrique donné. Les principaux objectifs sont de réduire les pertes actives et d'améliorer le profil de tension. Le problème de la reconfiguration est combinatoire et non linéaire, ce qui le rend pratiquement impossible à résoudre pour des réseaux électriques à taille réelle. Pour contrer ce problème, de nombreuses heuristiques et métaheuristiques ont été proposées dans la littérature.

Dans cette thèse, nous explorons et comparons différentes stratégies de reconfiguration. La première méthode utilisée est basée sur un algorithme d'arbre couvrant minimal pour générer une solution initiale. Cette solution est ensuite affinée à l'aide de deux heuristiques : la recherche locale (LS) et la recherche tabou (TS). Enfin, un modèle machine learning est entraîné pour reproduire le comportement de ces heuristiques. Ce modèle utilise diverses caractéristiques pour prédire si une ligne fait partie ou non de la configuration optimale.

Toutes les méthodes sont testées sur différents scénarios avec et sans DG à partir des réseaux IEEE 33 et IEEE 69. Les résultats montrent que la combinaison d'heuristiques classiques et d'approches machine learning offre un équilibre entre performances et coût de calcul, en particulier pour les applications nécessitant une prise de décision rapide. Ce travail contribue à la recherche en cours sur la reconfiguration de réseaux électriques de distribution, en constante évolution.

Declaration on the use of automatic tools for writing the manuscript

I hereby certify that I have not used any generative intelligence tool in the writing of text, graphics, images, or data reproduced in this manuscript.

The content of this thesis was mainly written in French. The entire thesis was translated into English using three translators: DeepL, Reverso, and ChatGPT. When an image is not sourced, it means that I created it directly. The figures representing electrical networks were created using the Ipe extensible drawing editor, compatible with LateX, and the graphs were generated with Python. All figures contained in this thesis were generated in a style that allows colorblind people to see them correctly.

List of Abbreviations

Abbreviation	Meaning
AC	Alternating Current
ACO	Ant Colony Optimization
ANN	Artificial Neural Network
AUC-PR	Area Under the Precision-Recall Curve
CNN	Convolutional Neural Network
CUI	Current Unbalance Index
DC	Direct Current
DNN	Deep Neural Network
DNR	Distribution Network Reconfiguration
DPSO	Discrete Particle Swarm Optimization
ENS	Energy Not Supplied
GA	Genetic Algorithm
HHO	Harris Hawks Optimization
LDBAS	Lévy Flight Beetle Antennae Search
LS	Local Search
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
MILP	Mixed Integer Linear Program
MINLP	Mixed Integer Non-Linear Program
MIQP	Mixed Integer Quadratic Program
ML	Machine Learning
MST	Minimum Spanning Tree
MSFLA	Modified Shuffled Frog Leaping Algorithm
NP-hard	Non-deterministic Polynomial-time hard
OLTC	On Load Tap Changer
OPF	Optimal Power Flow
PSO	Particle Swarm Optimization
RCS	Remotely Controlled Switch
RL	Reinforcement Learning
SA	Simulated Annealing
SBAT	Selective Bat Algorithm
SDP	Semidefinite Programming
SOCP	Second Order Cone Programming
STVS	Short-Term Voltage Stability
SVM	Support Vector Machine
TS	Tabu Search
VUI	Voltage Unbalance Index
WCA	Water Cycle Algorithm

Contents

List of Figures

List of Tables

Introduction

Distribution networks, also known as low-voltage networks, are the part of the electrical grid that distributes power directly to end users. Unlike high-voltage networks, i.e., transmission networks (which operate above 70 kV), distribution networks operate between 220 V and 36 kV. The difference between these two networks is not only in terms of voltage magnitude but also in terms of structure. [7].

Most distribution networks operate radially. This means that there is only one possible path between two nodes in the network and therefore no loops. The main advantage of this type of structure is that maintenance is easier. In addition, operating costs are lower. The disadvantage is that if there is a fault, it affects the following nodes. When a network is not radial, it is meshed (such as the transport network, for example). This means that there are multiple possible paths between two nodes in the network. Distribution networks, even if they operate radially, are built with a meshed structure [8].

In distribution networks there are two types of switches: sectionalizing switches and tie switches. The sectionalizing switches are normally closed and the tie switches are normally open. If a tie switch is closed, then the topology must be reviewed and a sectionalizing switch opened to maintain radiality. The reconfiguration of distribution networks consists in this. Naturally, the more switches a network contains, the more possibilities of reconfiguration are numerous [9]. An example of a network is shown in Figure 1. Four configurations are presented. The first corresponds to the basic topology and the following ones are obtained through the activation and deactivation of different switches. The power flows are different for each of the configurations and the voltage profile is dependent on this topology.

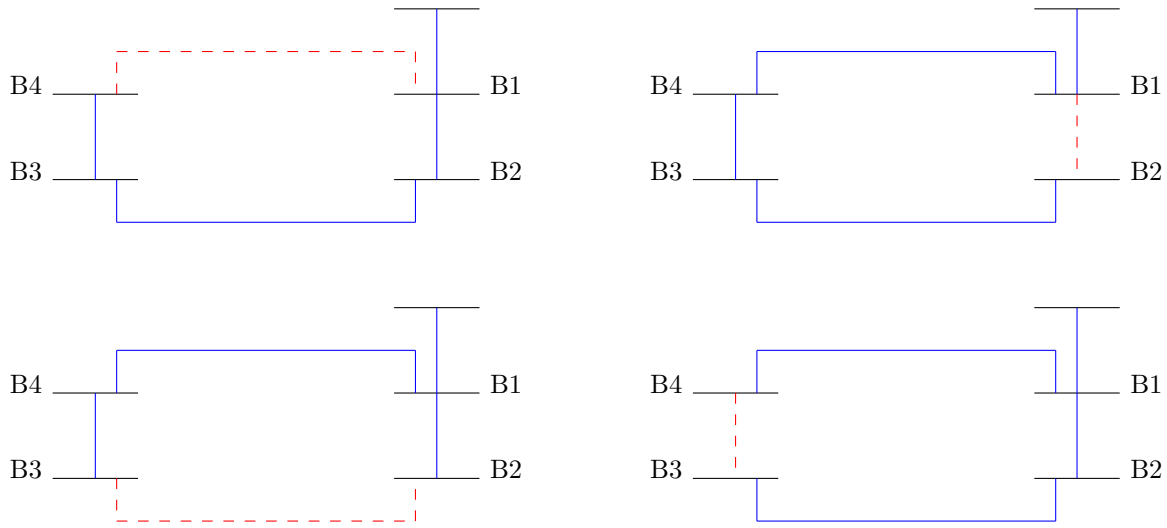


Figure 1: Possible topologies for a 4-nodes network - Figure from [1]

The reconfiguration of electrical distribution networks refers to modifying the topology of a network by opening and closing switches. There are several objectives for reconfiguration, but the most common is to minimize active losses in the network, often coupled with improving the voltage profile. From a mathematical point of view, the reconfiguration problem is complex. It is a combinatorial optimization problem, where each switch represents a binary variable (open/closed) combined with non-linear equations of power flows in electrical networks. The problem is NP-hard.

Since the problem is NP-hard, no exact approach is feasible on real-world networks within a reasonable time frame. This is why numerous approaches have been proposed in the literature, ranging from classic heuristics to more recent artificial intelligence methods, including complex metaheuristics such as genetic algorithms.

The aim of this thesis is to contribute to research on the problem of distribution network reconfiguration. To this end, several approaches are implemented and compared with each other. Before discussing the different methods proposed, chapters 1 and 2 present theoretical concepts related to the subject and a literature review focused on the last five years of research, respectively. Chapter 3 presents the first proposed method, a method based on the minimum spanning tree (MST) algorithm, which is used to generate a first feasible initial topology. In Chapter 4, the initial solution proposed is refined using two heuristics: a local search (LS) and a tabu search (TS) to improve the quality of the first solution. Finally, in Chapter 5, a machine learning model is proposed to reproduce the behaviors of the heuristics presented in the previous chapters.

Chapter 1

Theoretical background

1.1 Electricity networks

1.1.1 Power flows

Distribution networks run on alternating current (AC). In alternating current, complex power can be written as follows:

$$S = P + jQ$$

with P the active power (in Watts) and Q the reactive power (in Var). The active power is the one which is consumed by the loads but also the one that represents the useful work. Reactive power is the part that does not produce work. Even though it cannot be directly consumed by the loads, it has an important effect on voltage and voltage management is an important aspect in distribution networks. It is important to keep the voltage in an acceptable range (typically 5-10% around the nominal value) and for this purpose many equipment can be used: on load tap changers (OLTC), capacitor banks or voltage regulators. The reconfiguration can also have an effect on the voltage, this point is addressed when explaining different objectives [10] [11].

The power flow study consists in determining the electrical state of a network. This is a type of analysis used in the planning but also in the operational framework. This analysis is based on the Kirchhoff laws and the nonlinear power flow equations. At each node k of the network, the total contribution of the powers injected must correspond to the total contribution of the powers taken. For each bus k, the power flow equations are written as follows:

$$P_k = G_{kk}V_k^2 + V_k \sum_{m \in \mathcal{N} \setminus \{k\}} V_m \left(G_{km}\cos\theta_{km} + B_{km}\sin\theta_{km}\right) \tag{1.1}$$

$$Q_k = -B_{kk}V_k^2 + V_k \sum_{m \in \mathcal{N} \setminus \{k\}} V_m \left(G_{km}\sin\theta_{km} - B_{km}\cos\theta_{km}\right) \tag{1.2}$$

with:

- $V_k = |V_k|e^{j\theta_k}$ the voltage at node k, where $|V_k|$ is the magnitude and θ_k the phase angle;
- $Y_{km} = G_{km} + jB_{km}$ the elements of the admittance matrix (Y-bus matrix);
- $Y_{kk} = G_{kk} + jB_{kk}$ the sum of admittances between the k node and the mass (self-admittance);
- $\theta_{km} = \theta_k - \theta_m$ the phase shift between the voltages at the nodes k and m.

Power flow equations are solved by iterative methods and the most used method is that of Newton-Raphson [10]. Power flow equations can be simplified in radial networks. The simplified equations are called DistFlow equations and were introduced in 1989 by Mesut E. Baran and Felix F.Wu [11] [12].

Considering a branch connecting the parent node $p(i)$ to the child node i [13] [14]:

$$P_{p(i),i} = P_i^{\text{load}} + \sum_{k \in \text{child}(i)} P_{i,k} + r_{p(i),i} \frac{P_{p(i),i}^2 + Q_{p(i),i}^2}{|V_{p(i)}|^2} \tag{1.3}$$

$$Q_{p(i),i} = Q_i^{\text{load}} + \sum_{k \in \text{child}(i)} Q_{i,k} + x_{p(i),i} \frac{P_{p(i),i}^2 + Q_{p(i),i}^2}{|V_{p(i)}|^2} \tag{1.4}$$

$$|V_i|^2 = |V_{p(i)}|^2 - 2\left(r_{p(i),i} P_{p(i),i} + x_{p(i),i} Q_{p(i),i}\right) + \left(r_{p(i),i}^2 + x_{p(i),i}^2\right) \frac{P_{p(i),i}^2 + Q_{p(i),i}^2}{|V_{p(i)}|^2} \tag{1.5}$$

where :

- $P_{p(i),i}$ and $Q_{p(i),i}$ are respectively the active and reactive powers circulating from $p(i)$ to i.
- P_i^{load} and Q_i^{load} are the active and reactive powers consumed at the i node.
- $r_{p(i),i}$ and $x_{p(i),i}$are respectively the resistance and the reactance of the branch connecting the nodes $p(i)$ and i.
- $|V_i|$ et $|V_{p(i)}|$ are the voltages magnitudes at nodes i and $p(i)$ respectively.

DistFlow equations are usually solved using the Backward Forward Sweep Method [15].

1.1.2 Optimal Power Flow

The purpose of power flow is therefore to determine a feasible electrical state of the network (i.e., voltages, phase angles, and power flows). Optimal power flow (OPF) is an optimization calculation used to determine the best operating point of the network [16].

The OPF problem can be expressed as follows :

$$\begin{aligned} \min_{\{P_k, Q_k, |V_k|, \theta_k\}} \quad & \sum_{k \in \mathcal{G}} C_k(P_k) \\ \text{subject to:} \quad & \text{Power flow equations (1.1), (1.2) (or (1.3)–(1.5) in radial networks)} \\ & P_k^{\min} \le P_k \le P_k^{\max}, \quad \forall k \in \mathcal{G} \\ & Q_k^{\min} \le Q_k \le Q_k^{\max}, \quad \forall k \in \mathcal{G} \\ & |V_k^{\min}| \le |V_k| \le |V_k^{\max}|, \quad \forall k \in \mathcal{N} \\ & |S_{km}| \le S_{km}^{\max}, \quad \forall (k,m) \in \mathcal{E} \end{aligned} \tag{1.6}$$

With:

- $\mathcal{G}$ is the set of generator buses,
- $\mathcal{N}$ is the set of all buses in the network,

- $\mathcal{E}$ is the set of lines (edges) in the network,
- $C_k(P_k)$ is the generation cost function at generator k-usually convex, e.g., quadratic: $C_k(P_k) = a_k P_k^2 + b_k P_k + c_k$.

The most common problem is the minimization of an objective function corresponding to total generation costs, while respecting power flow equations, voltage and generation operational limits and power system topology. Sometimes other objective functions are used. The optimization problem can take the form of a power system loss minimization problem, or include other objectives such as emissions minimization.

The OPF problem is a non-convex optimization problem, especially when dealing with a highly meshed network, which makes its solution method more computationally demanding. The most successful methods for solving this problem are interior point methods. In the case of radial networks, convex relaxations can be considered: Second Order Connic Programming (SOCP) and Semidefinite Programming (SDP) facilitate calculations and still ensure global optimality in certain cases.

Finally, it should be noted that the use of OPF makes as much sense in planning (often in problems of sizing or locating elements in the network) as in real-time operations (dispatching or congestion management, for example).

1.2 Reconfiguration problem and optimization concepts

1.2.1 Formulation and constraints

The problem of distribution network reconfiguration (DNR) is a combinatorial optimization problem. The combinatorial aspect lies in the choice of switch states, which can be open or closed. The objective function varies according to the formulations and objectives pursued but generally includes minimization of total network losses. Let $x_l \in \{0, 1\}$ be a binary variable for each branch or switch of the network. Value 1 is a closed switch and value 0 is an open switch. The problem could be formulated as:

$$
\begin{aligned}
\min_{x} \quad & P_{\text{losses}} = \sum_{l \in \mathcal{L}} R_l I_l^2 \\
\text{s.t.} \quad & \text{Power flow equations are satisfied} \\
& V_i^{\min} \leq V_i \leq V_i^{\max}, \quad \forall i \in \mathcal{N} \\
& |I_l| \leq I_l^{\max}, \quad \forall l \in \mathcal{L} \\
& \text{Network remains connected and radial} \\
& x_l \in \{0, 1\}, \quad \forall l \in \mathcal{L}
\end{aligned}
$$

where $\mathcal{L}$ is the set of all branches and $\mathcal{N}$ the set of buses in the network [1].

Other objectives frequently discussed include minimising the number of switch-on/off operations, improving the voltage profile, reducing undistributed energy, improving various reliability indices,etc. These objectives are discussed in more detail in the section 2.1.

As Capitanescu and al. explain in their paper [17], the reconfiguration can be static or dynamic. Static reconfiguration of distribution networks (SDNR) is the optimization on a given state. This type of reconfiguration is used in a planning framework for example. Dynamic reconfiguration of distribution networks (DDNR) is the optimization in real time or near real time using remotely controlled switches (RCS). It is used to deal with the unpredictable events of the network (a fault, a peak load...)

1.2.2 Type of problem and resolution methods

As mentioned in 1.2.1 the reconfiguration problem is combinatorial. In theory, a network with N switches can have 2^N possible configurations. However, not all of them satisfy radiality and connectivity. For example, for a network of 33 nodes and 37 lines, there are 50751 radial configurations possible [18]. This number of configurations increases significantly with the size of the network and the number of branches and switches.

The problem contains nonlinear behaviors imposed by AC power flow equations as well as the combinatorial aspect of switch states, which introduce discrete variables. It is therefore formulated as a Mixed Integer Non Linear Program (MINLP). The problem can be proven to be NP-hard [19].

To address the reconfiguration problem, several models proposed in the literature are based on simplifying assumptions that allow the problem to be solved as a Mixed Integer Linear Program (MILP) or Mixed Integer Quadratic Program (MIQP). Among the most commonly used simplifications are the use of DC power flow models instead of AC models, for example, or setting all network voltages to 1 p.u. to circumvent linearities. The biggest advantage is that MILP and MIQP problems can be solved using conventional solvers. However, the assumptions chosen can lead to suboptimal solutions. Some researchers address this by using MILP or MIQP to obtain an initial solution, then refining it through more detailed electrical analyses [1].

1.2.3 Multi-objective optimization

As already mentioned, the reconfiguration problem is no longer a mono-objective problem. The problem is often formulated as a multi-objective problem. Several approaches are possible to formulate an optimization problem with several objectives.

- Weighted cost function: All objectives are summed and coefficients added to weight each objective. This is an effective method as it ensures that there is only one cost function but the difficulty lies in choosing the weight.
- Pareto optimum: Pareto solutions are configurations that include a trade-off. A solution is called a Pareto solution if the improvement of one of the objectives leads to the regression of another. All of these solutions are called the Pareto front. A point on the Pareto front is an optimal Pareto solution. It should be noted that finding the Pareto front is often computationally demanding. There are algorithms such as the Non-Dominated Sorting Genetic Algorithm (NGSA) or Multi-Objective Particle Swarm Optimization (MOPSO) that approximate the Pareto front. Put differently, a solution is called Pareto-optimal if there is no better alternative for all the objectives simultaneously [1].

1.3 Algorithms

This section provides theoretical concepts regarding the different algorithms used in the context of the reconfiguration of electrical networks.

1.3.1 Heuristics and metaheuristics

One way to address the problem of reconfiguration is to use heuristics and metaheuristics. Heuristics are algorithms specific to a problem that allow an acceptable, althought not optimal, solution to be found within a reasonable amount of time. Metaheuristics, on the other hand, do not depend on the problem

but rather address a class of problems. They do not guarantee optimality but are widely used in this field to address the scalability of the problem [20, 21].

1.3.1.1 Heuristics

Figure 1.1 shows a simplified 4-node example of the branch exchange algorithm, one popular heuristic techniques for reconfiguring distribution networks. The following is a description of this heuristic:

- The algorithm creates a loop by closing an open switch.
- Radiality must be restored by opening another switch, which is typically closed. The decision about which switch to open is not made at random; rather, it is based on how opening it affect the network.
- The process is repeated iteratively.

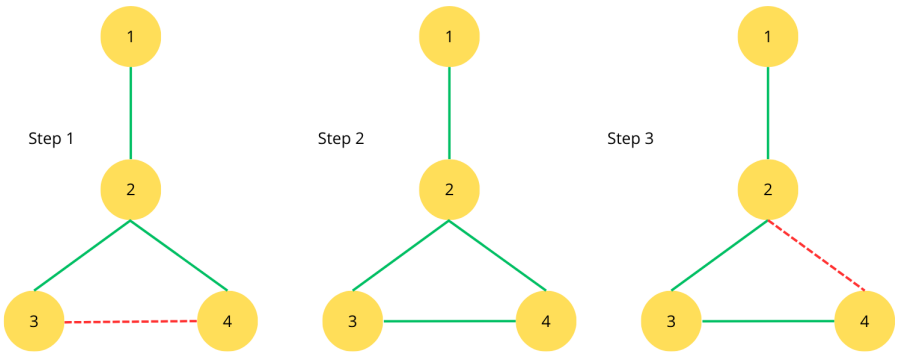


Figure 1.1: Branch-exchange heuristic

There are other basic heuristics. One method, for instance, is to open each branch, beginning with the one with the least amount of current. Another approach is to give priority to open lines with the highest impedance or based on other criteria.

The heuristics mentioned are capable of providing a high-quality solution within a reasonable time frame, but they are not always reliable [1].

1.3.1.2 Meta-heuristics

When the cost of considering an exact method is too high, metaheuristic algorithms are employed. They provide a good solution and are frequently employed in the context of distribution network reconfiguration, but they do not ensure an overall optimum.

The most widely used metaheuristic algorithms include:

- Genetic Algorithm (GA)
- Particle Swarm Optimization (PSO)
- Simulated Annealing (SA)
- Ant Colony Optimization (ACO)

Genetic Algorithm: It is an algorithm inspired by the theory of evolution of Charles Darwin. First, genes are the decision variables and chromosomes are groups of genes. Each chromosome represents a potential solution and therefore a group of chromosomes is a solution group.

The steps are shown in the Figure 1.2.

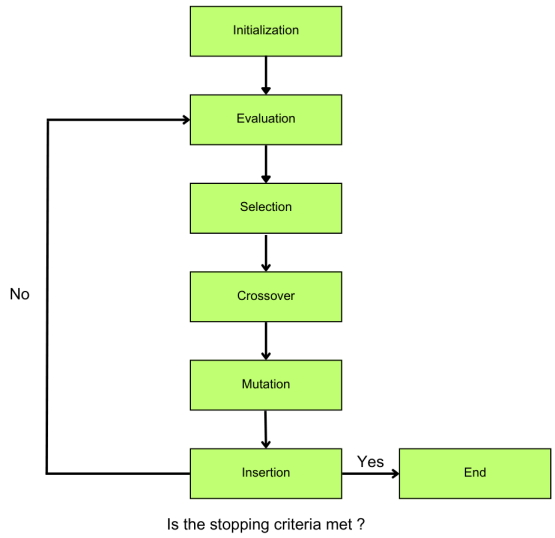


Figure 1.2: Flowchart for GA [2]

Initialization consists in the random generation of a population. For the evaluation stage, a function called fitness function is used to assess the quality of each response. Greater fitness is a better solution and implies that the chromosome is more likely to be selected as a parent of the next generation at the selection stage. Solutions retained after the selection stage are crossover and mutated. When crossover the goal is to mix the best aspects of several different solutions in order to create better solutions. The mutation, on the other hand, makes random changes to genes. The algorithm iterates until convergence. The advantage of genetic algorithms is that they are robust and the disadvantage is that they do not guarantee an overall optimum and may have high computation times depending on the size of the population [2].

Particle Swarm Optimization: This algorithm is inspired by group bird flights and fish shoals. Instead of a set of birds, we treat a set of particles (from the name). Each particle represents a solution (a network configuration, in the case of reconfiguration). Each of the particles goes through several positions but keeps in memory the best position found and the best overall position. This is an iterative algorithm where at each iteration, the positions are updated taking into account three factors:

- The force that causes it to keep going in the same direction;
- The knowledge of the best solution;
- A certain attraction to the best solution of the swarm.

The PSO is an algorithm easy to implement and with a fast-convergence quality. The disadvantage is that it can also converge fast to a local optimum. Discrete versions of the PSO are available to handle binary variables, like network switch activation and deactivation [22] [1].

Simulated Annealing: Annealing is a metallurgical process that served as the model for the algorithm. The process consists of a warming stage and a cooling stage. Each iteration corresponds to a change in the current configuration. When this modification is made, we evaluate it by calculating the variation Δ of our target (e.g., network losses). The new solution is approved if it enhances the goal (i.e., $\Delta < 0$). If not, it can still be accepted with a probability $p = \exp(-\Delta/T)$, where T is a temperature parameter that decreases with the number of iterations. At the beginning, the value of T is high and corresponds to a stage where exploration is emphasized; these are the first iterations. The algorithm aims more for exploitation as it approaches a final response and T tends toward 0 toward the end. Metal cooling is being imitated here. The SA algorithm offers solutions that are close to the global optimum. Its disadvantage remains the computation time for networks with many nodes [23] [1] [4].

Figure 1.3 shows the different steps of the SA algorithm.

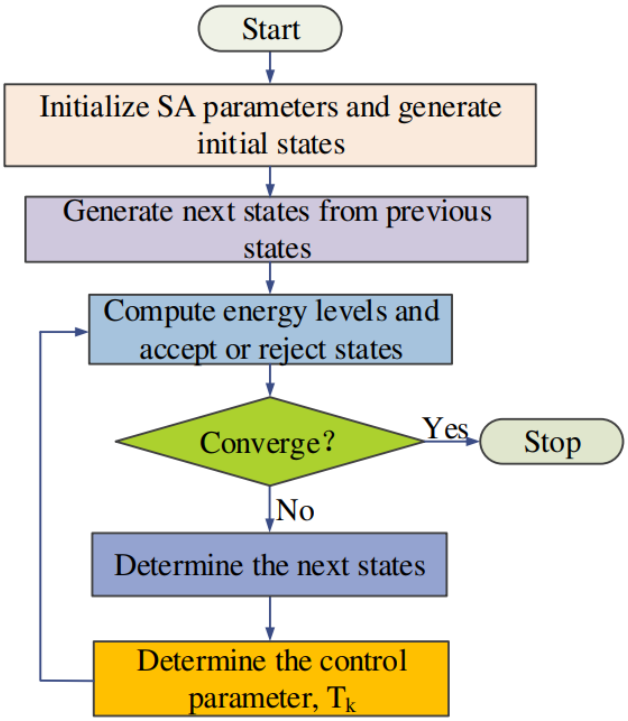


Figure 1.3: Flowchart for SA [3]

Ant Colony Optimization: Ants' feeding habits served as the model for this algorithm. Ants in real life take random paths in search of food. They leave behind pheromones that let them follow their path back. The quality of the food affects the amount of pheromones deposited. In the context of the reconfiguration problem, we could imagine an ant that starts with the initial configuration and makes changes (by closing and opening switches). It evaluates each configuration with a quantity of pheromones. If the quality of a configuration is high, the number of pheromones is large, and conversely, if the quality of a configuration is low, the number of pheromones is small. One or more solutions are reached by the algorithm.Because of pheromones, this algorithm has the advantage of combining memory and random exploration [22] [4]. The procedure is described in Figure 1.4. N is nest and F is food. The ant explores paths to the food and then returns to the nest by releasing pheromones on the way, the shorter it is the more it releases.

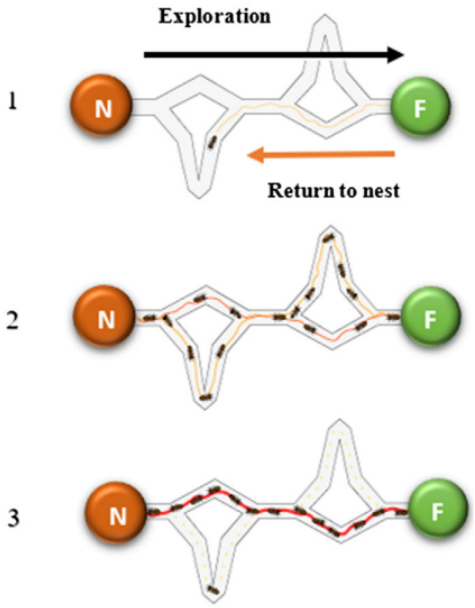


Figure 1.4: Ant colony optimization [4]

Metaheuristics are numerous and each of them can make different contributions. For more information on how these algorithms work, the support [4] provides an overview of many of them, along with their mathematical formulations. Finally, metaheuristics do not guarantee an overall optimum but provide good quality solutions in a reasonable time for networks up to a few hundred nodes

1.3.2 Machine Learning

Artificial intelligence is increasingly present in literature as a means of solving different types of problems. It is a set of algorithms that enable a model to learn based on statistics. It is possible to identify several categories [5] depending on the form of the data sets:

- **Supervised learning** : The data used for training is labelled. The best known supervised learning methods are linear methods, decision trees, support vector machines (SVM) and artificial neural networks (ANN).
- **Unsupervised learning** : Unlike supervised learning, data is not tagged. In this section we find clustering methods (k-means, hierarchical clustering...).
- **Semisupervised learning** : This is a combination of tagged and untagged data used to train the desired model.
- **Reinforcement learning (RL)** : This specific type of learning does not require pre-calculated data. An agent is implemented to interact with a certain environment. The goal is to learn a sequence of actions (policy) that maximizes the sum of rewards from each action.

In the context of the study of the reconfiguration of electrical distribution networks, the most used machine learning approaches are deep neural networks and reinforcement learning.

1.3.2.1 Deep learning

Deep learning is a branch of machine learning whose algorithms are inspired by the functioning of neural networks. Neural networks are composed of several layers:

- An input layer that receives the input data;
- Multiple hidden layers for working with data;
- An output layer that provides the final prediction.

Among the many neural network structures in deep learning are multilayer perceptrons (MLP). This is a model used in deep learning where each neuron of each layer is connected to all the neurons of the next layer.

The connections between neurons all have a weight that is adjusted as learning takes place. The more layers there are, the deeper the learning is said to be and therefore the more complex relationships it can learn. To learn, the steps are as follows:

- Forward pass : The network receives input data and transmits it from layer to layer. A calculation $z = \sum x_i w_i + b$ is performed at the level of each neuron and a non-linear activation function is used to predict an output;
- Error calculation: The network prediction is compared to the correct answer. To quantify the error, a loss function is used (typically an average quadratic error or crossover entropy);
- Backpropagation: the error is propagated in the opposite direction to calculate the contribution of each weight on the final error, thanks to the chain rule;
- Weight update: a gradient descent allows us to modify weights in order to minimize the loss function.

Mathematical models can be found in [5] and [24], or in more detail in [25].

In the case of network reconfiguration, neural networks can be used, once the model trained to determine optimal configurations based on various inputs describing the state of the system (loads, topology,...). The advantage of machine learning algorithms is that once trained, they can provide instant answers.

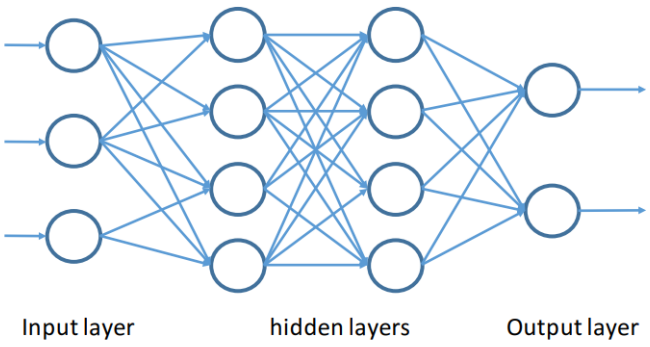


Figure 1.5: Multilayer perceptron with 3 inputs, 2 hidden layers of 4 neurons each and 2 outputs [5]

1.3.2.2 Reinforcement learning

Reinforcement learning is a slightly different discipline from machine learning. It does not use supervised learning. Instead, an agent is implemented to learn how to interact with an environment. Its goal is to

find a policy (that is, a sequence of actions) that maximizes the sum of total rewards.

A typical RL problem is formalized by a Markovian decision process (MDP), defined by:

- S : the set of all possible environmental states;
- A : the set of all possible actions of the agent;
- $P(s'|s,a)$: the probability of transition to the state s' after taking the action a in state s ;
- $R(s,a)$: the reward obtained after taking the action a in the state s.

The agent's objective is to find the optimal policy $\pi*(s)$ and for this the goal is to maximize the value function $V(s)$ or the action value function $Q(s,a)$, which respectively represent the quality of a state or an action in a given state [24].

Various algorithms are available in machine learning: Q-learning, deep Q-Network, SARSA, etc. Explanations can be found in the following supports: [25] [6].

As part of the problem of reconfiguring electrical networks, Reinforcement Learning approaches are beginning to become quite common and are quite attractive by their ability to adapt to uncertain environments [19].

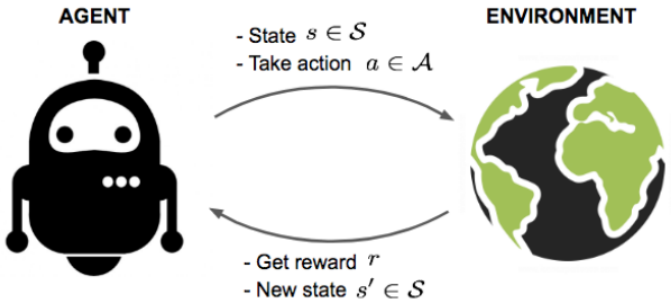


Figure 1.6: Representation of RL principle - Figure from [6]

Chapter 2

State of the art

The reconfiguration of electrical distribution networks consists in the modification of the topology of a network by means of actions of opening and closing switches, in order to optimize certain variables of the network. It was first introduced in the 1970s and among the first people to talk about it are A. Merlin and H. Back, authors of the scientific paper [26]. They introduce reconfiguration as a means to minimize the power losses of an urban distribution network.

The first reconfiguration methods were deterministic. Among these methods, there are classic branch-and-bound algorithms that identify the tree with the least losses of a graph [26]. It should be noted that until 1989, the different formulations of the reconfiguration problem did not take into account the complete equations of the AC power flow. They often proposed simplified models, based on a DC approximation. In 1989, Dariush Shirmohammadi and H. Wayne Hong were among the first to consider this aspect. They formulated the reconfiguration problem by taking into account the complete equations of the AC power flow and proposing a branch exchange heuristic. [27]

Minimization of losses in distribution networks was the main objective of the reconfiguration, however this objective has expanded. Over the past two decades, reconfiguration has been studied as a means not only to minimize losses but also to increase the capacity of distributed energy sources, improve the voltage profile or balance loads [19] [17].

From a mathematical point of view, the reconfiguration problem is formulated as a large mixed integer non linear programm (MINLP). It is NP-hard [19]. The difficulty in solving the problem is due to the combinatorial aspect (the variables defining the switches are binary) and the nonlinear aspect of the AC power flow equations. It is therefore difficult to find a solution to the problem for real-world networks. The research turned to optimization methods such as heuristics and metaheuristics [28] and recently, artificial intelligence and machine learning methods have been proposed in the literature.

This literature review presents the state of the art of recent (2020 and later) scientific contributions on the reconfiguration of distribution networks.

2.1 Problematic and objectives

The main purpose of reconfiguring electrical distribution systems is to determine the ideal switch condition. The reconfiguration problem, already defined in the section 1.2.1 consists of minimizing one or more cost functions, while respecting the physical constraints

The most basic form of reconfiguration involves minimizing active power losses but with the different advances in electrical networks, new constraints have appeared. These constraints have increased the complexity of the problem.

The minimization of active losses and the improvement of voltage profiles are no longer the only objectives. The reconfiguration of distribution grids also seeks to balance loads [29] and reduce energy non supplied (ENS) [30].

An additional objective of the reconfiguration is the integration of distributed energy sources. Renewable sources such as wind or photovoltaic power are intermittent and cause significant voltage fluctuations. Some papers explore reconfiguration as a way to increase the hosting capacity of renewable energy sources in distribution networks [17] [27].

Finally, from an economic and environmental point of view, reconfiguration can also play a role. It reduces operational costs and environmental impacts. Operating costs can be reduced by dynamically adjusting the topology [31]. Other articles in the scientific literature such as [32] include CO2 emission reduction in their targets.

In short, the problem of reconfiguration of distribution networks is a multi-constraint and multi-objective optimization problem. The current objective is to find a solution that can be used in real time, subject to physical and operational constraints.

2.2 Classic reconfiguration methods

The first methods used to solve the reconfiguration problem were either deterministic or fairly simple heuristic methods. Often these methods had a single purpose, such as minimizing losses. As mentioned in section 2, A. Merlin and H. Back were pioneers in 1975. They formulated the problem as finding a tree covering minimal losses and proposed a solution to this problem through an algorithm of branch-and-bound [26].

Some years later, in 1988, S.Civanlar and al. proposed a branch exchange algorithm [33]. This heuristic is explained in the 1.3.1.1 section.

In 1989, Mesut E. Baran and Felix F. Wu formulated the problem as a nonlinear integer optimization program. They proposed an iterative method to approach the solution [29].

All these works have shown that the reconfiguration could not only reduce losses in a significant way but also allows to improve the voltage profile and line loading. These classical approaches all show the same limitation: the combinatorial aspect of the problem. A radial network of N branches with k normally closed switches can in theory have $\binom{N}{k}$ possible configurations. For real networks, the problem can become costly in terms of resolution.

In addition, as mentioned in [28], in the majority of papers presenting these classical approaches, the networks are considered as balanced three-phase networks and do not take into account real constraints such as load imbalance or voltage and current fluctuations.

In 1990, new methods were introduced. Among these methods it is possible to find genetic algorithms [34] applied to network reconfiguration. This approach allows to find configurations close to the optimal solution, with a reasonable calculation cost. Other classical metaheuristic methods have been proposed, such as the simulated annealing algorithm [35]. This method does not guarantee an global optimum.

However, it provides a good quality solution in a reasonable time, which is not possible with exact methods.

2.3 Metaheuristic and evolutionary methods

Scientific literature is constantly being updated with numerous algorithms proposing solutions to the problem of reconfiguration. In this section, work using evolutionary algorithms and other techniques inspired by nature are presented. The focus is on post-2020 research. An introduction to metaheuristic algorithms is presented in the section 1.3.1.2.

Genetic algorithms are popular in reconfiguration and many variants of these algorithms are possible. An example of a variant is presented in the paper [36]. This is a motion for a resolution combining genetic algorithm and mathematical optimization whose purpose is to minimize economic losses due to fault in the network. Hossein Lotfi and al. combine, in 2020, an improved version of the particle swarm algorithm (IPSO) with a modified shuffled frog leaping algorithm (MSFLA) [31]. Their goal is to work and optimize simultaneously on two practices: network reconfiguration and capacitor allocation. This allows, among other things, to improve the voltage stability. The multi-objective optimization proposed in this paper is based on the Pareto principle and shows better results than a conventional genetic algorithm.

The problem of reconfiguration has been approached with genetic algorithms but also with different approaches inspired by nature. In 2020, Milad Rahimi Pour Behbahani and al. addressed the reconfiguration problem using a Discrete Particle Swarm Optimization (DPSO) algorithm. The algorithm in question was improved and then used to reduce voltage fluctuations caused by a welder, considered as a fluctuating load. It was tested on a 69-node network and a second 95-node network. It shows encouraging results on both networks as losses decrease and the voltage profile is better [37]. Another example of an algorithm inspired by nature is the one used by Surender Reddy Salkuti in 2021, named algorithm ant lion optimization. In this case, the reconfiguration is approached for an unbalanced network with a presence of distributed energy sources, whose uncertainties are treated through probabilistic analysis. The algorithm has been tested on a 17-node network and, like the previous one, it shows positive results as it reduces losses, improves the voltage profile and minimizes operating costs [38]. A third illustrative example is that proposed by Cassio Gerez and al., who in 2022 addressed the reconfiguration problem through an algorithm called Selective Bat Algorithm (SBAT). The objective function takes into account the voltage imbalance index (VUI) and current imbalance index (CUI). The modelled loads are variable. Tests were performed on a 33-node network with imbalances below 3% and losses were reduced by 31% compared to the initial configuration [28].

The range of nature-inspired algorithms used is large. There are approaches using the Harris Hawks algorithm (HHO), the water cycle algorithm (WCA), particle swarm search and a whole bunch of evolutionary methods based on Bayesian learning [19]. There are also hybrid methods for using chaotic perturbations. The Lévy Flight Beetle Antennae Search (LDBAS) algorithm is part of the methods. It was tested for dynamic reconfiguration and shows satisfactory results on a network of 33 and 118 buses [39].

The reconfiguration of electrical distribution networks has been addressed many times using metaheuristic or evolutionary methods. Each of the works makes a different and unique contribution. Some contributions are more relevant than others depending on the objective pursued and the topology concerned. What is important to remember is that whatever the metaheuristic method used, these methods are stochastic and therefore do not guarantee that the solution found is the overall optimum. In addition, the computation time of these methods increases drastically with the size of the networks, which leaves room for advanced optimization and machine learning methods, which are studied in the next section.

2.4 Advanced optimization methods and machine learning

Electricity grids are increasingly complex and with the penetration of renewable energy sources, responses must be almost instantaneous. To address this problem, hybrid approaches combining classical optimization and artificial intelligence have emerged. In this section, three types of methods are explored: advanced mathematical optimization methods, machine learning methods that use neural networks and reinforcement learning methods.

2.4.1 Advanced optimization methods

Advanced optimization methods are designed to ensure an optimal solution in a reasonable time. To do this, they usually rely on simplified models of the networks studied.

The various advances in numerical solvers have made it possible to rework deterministic formulations that had been proposed in the past. Simpler (linear) models of the DistFlow equations have been developed. Models formulated in this way can be solved using solvers such as CPLEX, by formulating a Mixed Integer Linear Program (MILP) or a Mixed Integer Quadratic Program (MIQP). However, it should be noted that the linearization of power equations can lead to sub-optimal solutions [19].

The advantage of these approaches is that they guarantee optimitality and the disadvantage is that they guarantee it within a linearized model. The answer may therefore contain errors when applied to a real network.

2.4.2 Machine learning methods - neural networks

Machine learning methods, and more concretely the methods using neural networks, allow to predict the network configuration from collected data.

In 2020, Weiye Zheng and al. proposed a method combining deep learning and robust optimization. The goal was to manage the uncertainty of distributed loads and generations. This model is represented as a deep neural network (DNN) and is coupled to an LSTM model. The long-short-term memory (LSTM) model analyses past changes and helps to anticipate future fluctuations by calculating an estimate of uncertainties. These predictions are then integrated into a robust optimization model that help select the best network configuration [40].

In 2021, Huang and al. worked on a model based on deep learning and therefore the goal was to improve short-term voltage stability (STVS) as part of reconfiguring distribution networks. The method is based on a convolutif neural network (CNN) which allows to give an estimate of the voltage stability for different topology of the network, all without having to use complex simulations. The method has been tested on a network of 69 nodes and makes it possible to accelerate the selection of optimal configurations. This method facilitates dynamic realtime reconfiguration [41].

These two approaches are not isolated, other papers in the literature use machine learning for the reconfiguration of electrical networks. The advantage is that once the model is trained, the prediction of the ideal configuration is almost instantaneous. However, a significant amount of data is required to train the model. If the model is not properly trained, the proposed configuration may be sub-optimal.

2.4.3 Machine learning methods - reinforcement learning (RL)

Among the machine learning methods, we distinguish a category that is methods using reinforcement learning (RL). These methods go even further as they learn a reconfiguration strategy. In other words,

reinforcement learning approaches do not require pre-calculated optimal solutions because they teach an agent how to behave optimally. These methods are mainly used in the context of dynamic reconfigurations.

In 2023, Gholizadeh and Musilek proposed a reconfiguration approach based on deep Q-learning and four of its variants: deep Q-learning, dueling deep Q-learning, deep Q-learning with prioritized experience replay, soft actor-critic, and proximal policy optimization. They first introduced a method of reducing the space for action. To do this, they use a first DQN to identify the most relevant configurations. This method reduces the calculation time and simultaneously guarantees solutions close to optimal [42]. In 2021, Wang and al. sought to improve the stability as well as the training speed of the DQN. To do this, they added noise layers (NoisyNet) for exploration. With this addition, the agent is more efficient in learning and there is no need to set the exploration rate ϵ. The method converges to a solution close to the optimum [43].

It is important to note that the purpose of these methods is not to completely replace conventional methods. Artificial intelligence and machine learning are complementary to the latter. Some papers explore hybrid methods, combining for example a heuristic method and a neural network approach. The literature shows an advanced stage of artificial intelligence approaches for reconfiguring distribution networks but they are still at the research stage. The challenges lie mainly in the robustness of the agents in the face of less typical events in the network. In conclusion, artificial intelligence and machine learning methods are emerging and promising. They facilitate dynamic reconfiguration [19] [43].

2.5 Perspectives of literature

The problem of reconfiguring distribution networks has evolved significantly in recent years. This was a single-purpose problem and is now completely different in view of the evolution of electricity networks. The integration of renewable energy sources and electric cars, for example, is one reason for the reformulation of the reconfiguration problem. Currently, the criteria can be many and varied: minimization of losses, improvement of the voltage profile, minimization of operating costs, minimization of indicators such as energy not supplied (ENS) [30], reduction of emissions CO_2 [32], etc. The current challenge is to find a compromise that meets many of these criteria. Several researchers have addressed this problem and proposed methods of multi-objective reconfiguration [31] [44]

Although there are many technological advances, several challenges remain and are the subject of research.

The most important challenge is related to the scalability of reconfiguration algorithms. Tests on small distribution networks are often successful, but most of the proposed algorithms see their computation time explode when the test networks are represented at a real scale [19]. This means that for a static reconfiguration, the majority of the proposed algorithms are efficient. However, for real-time or near-real-time reconfiguration, the majority of methods are not suitable. The hope for dynamic reconfiguration is mainly based on machine learning and reinforcement learning techniques [19] which would provide real-time estimates of the effect of a switch as well as advanced optimization formulations [45].

A second issue is the fact that reconfiguration increasingly requires a real-time or near-real-time response. This leads research to delve deeper into the field of dynamic reconfiguration rather than static reconfiguration [46]. The dynamic reconfiguration of electricity grids, coupled with other sources of flexibility, can be expected to become a highly discussed topic. These linkages could take the form of reconfiguration based on distributed production or the use of energy storage systems, for example. We also find in the literature the pair reconfiguration with intelligent charging of electric vehicles [46].

Third, it is important to talk about the resilience of distribution networks. There are more and more natural disasters, and reconfiguration would make a network more resilient to these events. The purpose of this type of reconfiguration is to restore the affected points by diverting power flows through other lines in the network. The reconfiguration in these situations improves network resilience and reliability and minimizes unsupplied power [47]. One way to address the problem would be to add new switches at critical locations. Optimizing the placement of these switches corresponds to long-term planning while dynamic reconfiguration corresponds to real-time optimization. Both can be coupled and can reduce cutoff in case of failure [46]. An objective not studied in the current literature is the minimization of load shedding. Integrating this variable can be a lead for future contributions [46].

In summary, the reconfiguration of distribution networks is evolving towards so-called intelligent systems. In the future, the algorithms used will have to exploit real-time data and be multi-objective. Current progress is optimistic and gives hope that artificial intelligence or stochastic optimization methods will be feasible.

2.6 Conclusion

The reconfiguration of electricity distribution networks has become a wide and well-nourished field of research. The post-2020 literature shows an increase and interest in the subject, especially with the evolution of electrical networks.

The initial one-objective problem, treated with simple heuristics, has become a multi-objective problem, much more complex, addressed by a range of algorithms. Metaheuristic methods (evolutionary algorithms, particular swarms and techniques inspired by nature) occupy a large place thanks to their management of combinatorial and non-linear problems. In parallel, the use of artificial intelligence techniques (deep learning and reinforcement learning in particular) is emerging and opens up a new and vast research topic.

There is no shortage of challenges: scalability, reliability, resilience and time horizon. However, there is clear agreement in the literature on the use of reconfiguration in future networks.

Chapter 3

Minimum spanning tree as a mean of reconfiguration

3.1 Introduction

As mentioned several times, the distribution network reconfiguration (DNR) involves modifying the topology of a network with one (or more) goal (or goals) in mind: reducing losses, improving the voltage profile or reducing overloads, for example. In this chapter, a first approach based on the Minimum Spanning Tree (MST) is presented. The idea is as follows:

- Closure of all tie-lines to create a mesh network;
- Calculate power flows and losses using Power Flow or Optimal Power Flow;
- Creation of a graph whose weights are data retrieved in the previous step (power flow, active losses or current magnitude);
- Application of a Minimum Spanning Tree algorithm (Kruskal, Prim, etc.);
- Opening of lines that are not part of the Minimum Spanning Tree to recover the reconfigured final radial network.

This initial method is fairly simple to implement. However, it does have its limitations, which are presented at the end of the chapter. The approach is tested on two networks: the *IEEE 33* network and the *IEEE 69* network, both of which are widely used in the scientific literature.

3.2 Minimum Spanning Tree algorithm (Kruskal)

This section is dedicated to the explanation of the Minimum Spanning Tree (MST) algorithm which is the first algorithm used in this study.

The principle of MST is to find, for a connected graph and whose weights are assigned to the edges, the tree whose sum of the weights is the smallest. This tree is called the minimum spanning tree. Among the most used algorithms to find this tree are the Kruskal algorithm and the Prim algorithm. In this study, the focus is on the Kruskal algorithm. The principle of this algorithm is quite simple and is represented in Figure 3.1. The steps are the following ones:

1. First of all, we sort all the edges in ascending weight order. That is, the first edge is the one with the lowest weight and the last edge is the one with the highest weight;

2. Once the edges are sorted, we add the first edge of the list to the new graph. Then, we add the second edge to the graph. From the third, we check if cycles are created. If a cycle is created, we do not add the edge and we go to the next one. If not, we add it.

3. Repeat the previous step until there are V-1 edges, where V is the number of nodes in the graph.

The pseudo code of the algorithm can be written as follows :

Algorithm 1: Kruskal Algorithm

Input: Graph $G = (V, E)$ with edge weights
Result: Minimum Spanning Tree T

```
Sort all edges E in increasing order of weight;
Initialize T ← ∅;
for each edge e = (u, v) in sorted E do
    if adding e to T does not create a cycle then
        Add e to T;
    if T contains |V| − 1 edges then
        Stop the loop
return T
```

3.3 Presentation of test networks

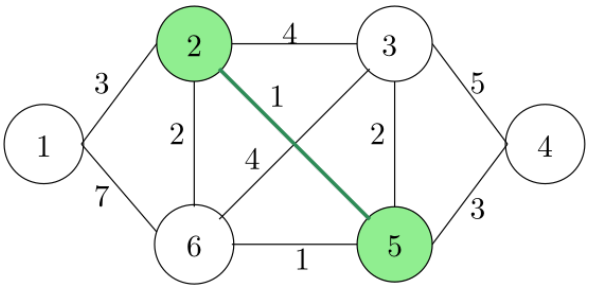
As previously mentioned, two common test networks from the scientific literature were used in this chapter: the IEEE 33 network and the IEEE 69 network.

3.3.1 IEEE 33 Network

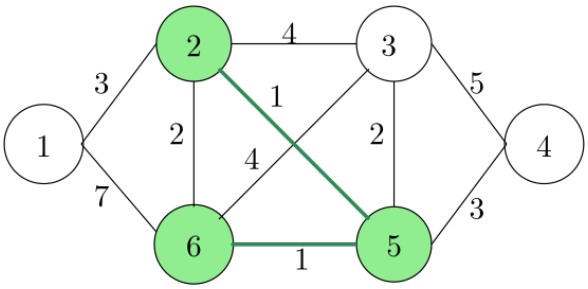
The IEEE 33 network is illustrated in Figure 3.2 in its operational configuration. It has 33 nodes and 32 lines when operating radially and is provided with 5 tie-lines represented with red dotted lines in the figure connecting the following pairs of nodes: *(8,21), (9,15), (12,22),(18,33), (25,29)*. Each line is characterized by a resistance R and a reactance X. Data were extracted from [48] and [49] for tie lines. They are shown in Appendix A.

The IEEE 33 network operates at a voltage level of 12.66 kV and is fed by a slack at node 1. The voltage profile of the network is shown in Figure 3.3. The figure shows a decrease in voltage magnitude between bus 1 and bus 18, and then between bus 19 and bus 33. The further a bus is from the slack, the greater the voltage drop and the lower the voltage magnitude of the bus. In its radial version without any distributed generation as presented, the minimum voltage is 0.904 pu and few buses are in the acceptable voltage range, i.e. between 0.95 and 1.05 pu.

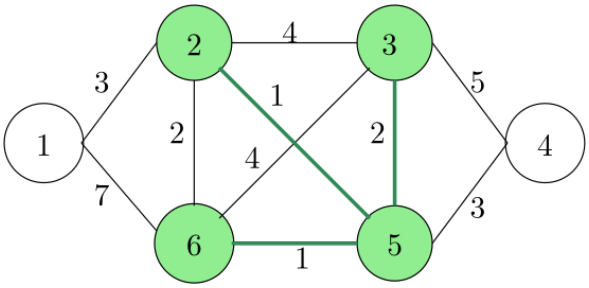
Total demand on the IEEE 33 network is 3.7 MW active power and 2.3 MVAr reactive power. The load distribution by bus is shown in Figure 3.4. The majority of buses feed loads below 100 kW and 100 kVAr, but a few exceed this value. Buses with the highest loads include 7,8,24,25 and 29 to 32.



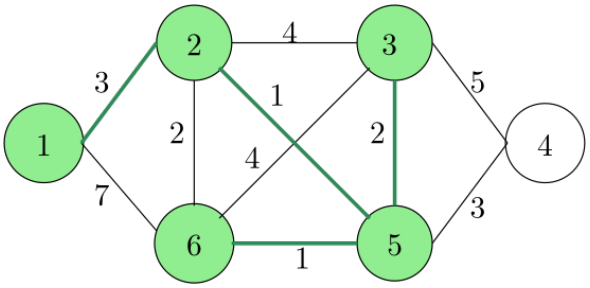
(a) Step 1



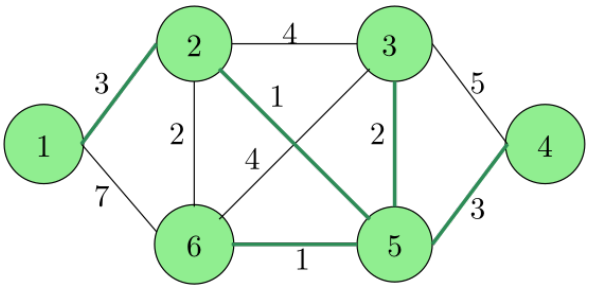
(b) Step 2



(c) Step 3



(d) Step 4



(e) Step 5

Figure 3.1: Example of the Kruskal algorithm on a 6-nodes graph

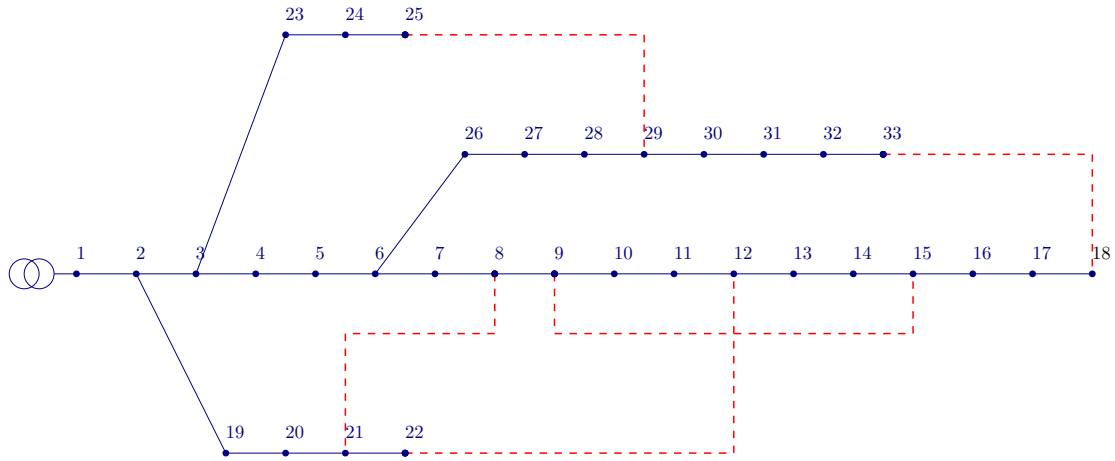


Figure 3.2: IEEE 33 network

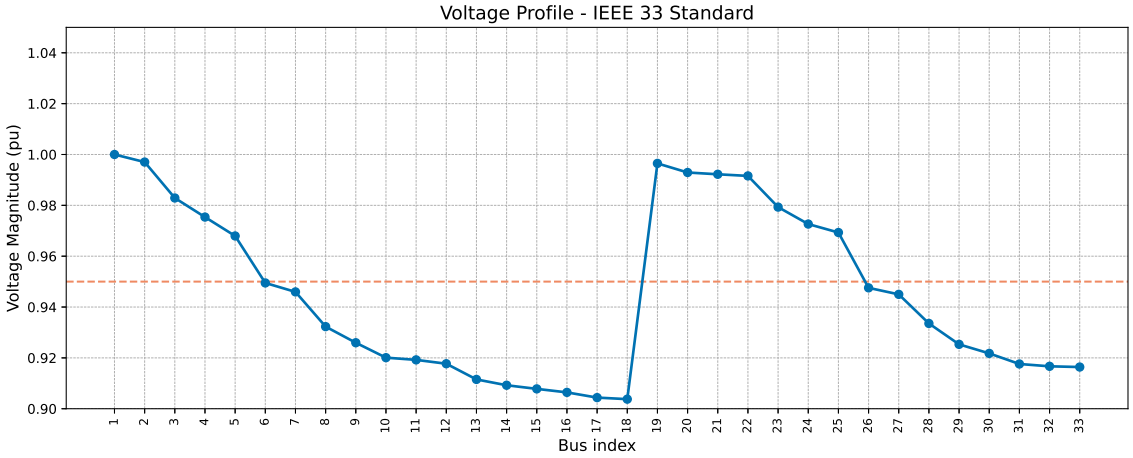


Figure 3.3: IEEE 33 network - Voltage profile

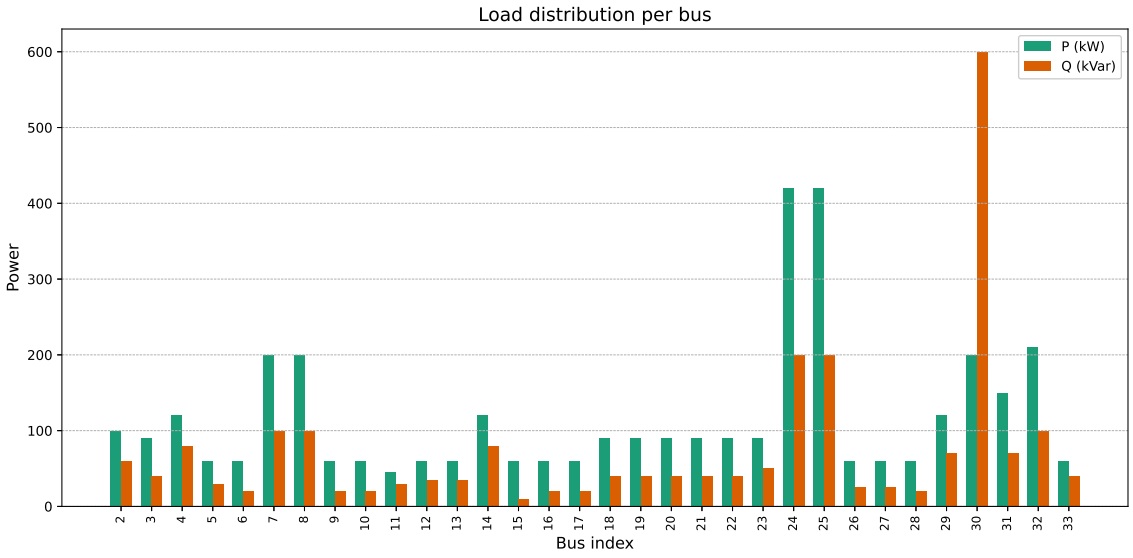


Figure 3.4: IEEE 33 - Load distribution per bus

3.3.1.1 Scenarios description

In order to study the network and its reconfiguration, 5 different scenarios are presented. The scenarios are characterized by the activation (or non-activation) of distributed generators (DG). The network has 8 distributed generators. Their capacities and locations are shown in Table 3.1.

Table 3.1: Characteristics of IEEE 33 network DGs

Generator	Capacity P_{max} [MW]	Location (Bus)
G1	2.395	6
G2	0.236	7
G3	1.176	13
G4	0.115	18
G5	2.181	22
G6	1.751	25
G7	1.290	28
G8	2.381	33

The different scenarios are presented in Table 3.2. Only colored boxes correspond to activated generators. Boxes marked with a 0 imply that the generator in question is not in use in this scenario. The first scenario corresponds to case 3.2 and the next four are illustrated in Figure 3.5. Scenarios 2, 3 and 4 and the generator data are taken from the scientific paper [17] and scenario 5 corresponds to the activation of all generators.

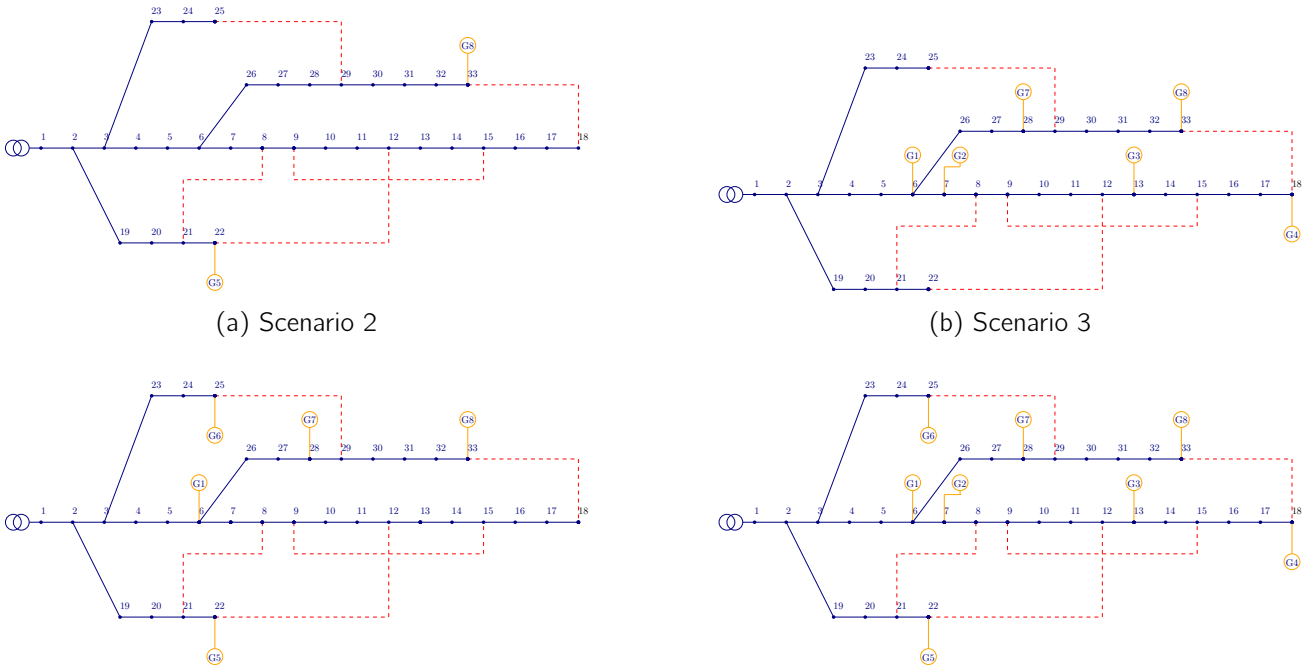
Table 3.2: Maximum capacity for generators by scenario (in MW)

Generator	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**	**Scenario 5**
G1	0	0	2,395	2,395	2,395
G2	0	0	0,236	0	0,236
G3	0	0	1,176	0	1,176
G4	0	0	0,115	0	0,115
G5	0	2,181	2,181	2,181	2,181
G6	0	0	0	1,751	1,751
G7	0	0	1,290	1,290	1,290
G8	0	2,381	2,381	2,381	2,381
Total [MW]	0	4,562	9,774	9,998	11,525

The tests carried out on the different scenarios are described later in the section 3.4.

3.3.2 IEEE 69 network

The IEEE 69 network is illustrated in Figure 3.6 in its operational configuration. It has 69 nodes and 68 lines when operating radially and is provided with 5 tie-lines represented with red dotted lines in the figure connecting the following pairs of nodes: *(11,43), (13,21), (15,46),(27,65), (50,59)*. Each line is characterized by a resistance R and a reactance X. Data were extracted from [48] and [50] for tie lines.



(a) Scenario 2

(b) Scenario 3

(c) Scenario 4

(d) Scenario 5

Figure 3.5: IEEE 33 network - Scenarios

They are shown in Appendix B.

The IEEE 69 network operates at a voltage level of 12.66 kV and is fed by a slack at node 1. The voltage profile of the network is shown in Figure 3.7. The figure shows a decrease in voltage magnitude between 1 and 27, and then between 48 and 65. As explained before, the further a bus is from the slack, the greater the voltage drop and the lower the voltage magnitude of the bus. In its radial version without any distributed generation as presented, the minimum voltage is 0.909 pu.

Total demand on the IEEE 69 network is 3.8 MW active power and 2.7 MVAr reactive power. The load distribution by bus is shown in Figure 3.8. The load distribution is not at all balanced. Buses 1 to 6 feed no load, and the majority of buses feed a load of less than 100 kW. However, buses 49, 50 and 64 feed loads of between 200 and 400 kW. Bus 61 tops the list with an active load of 1200 kW and a reactive load of 900 kVAr.

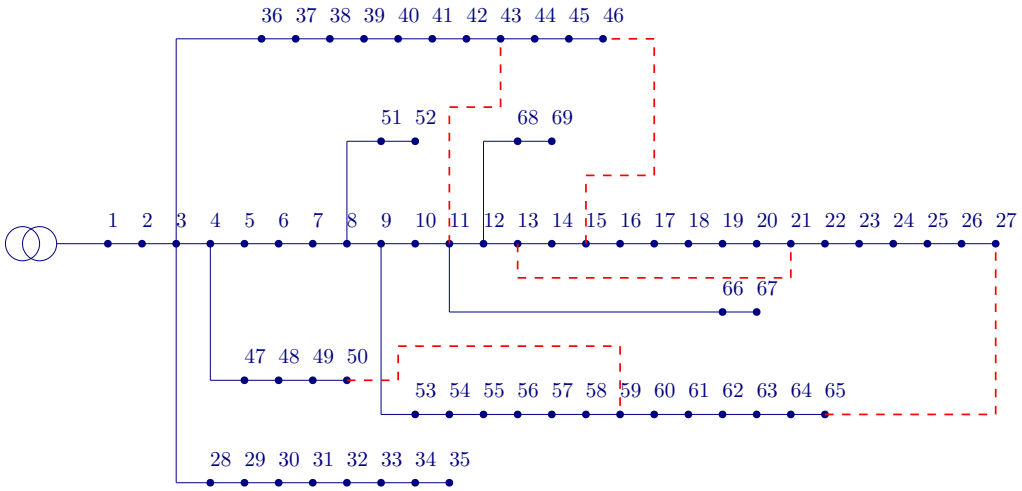


Figure 3.6: IEEE 69 network

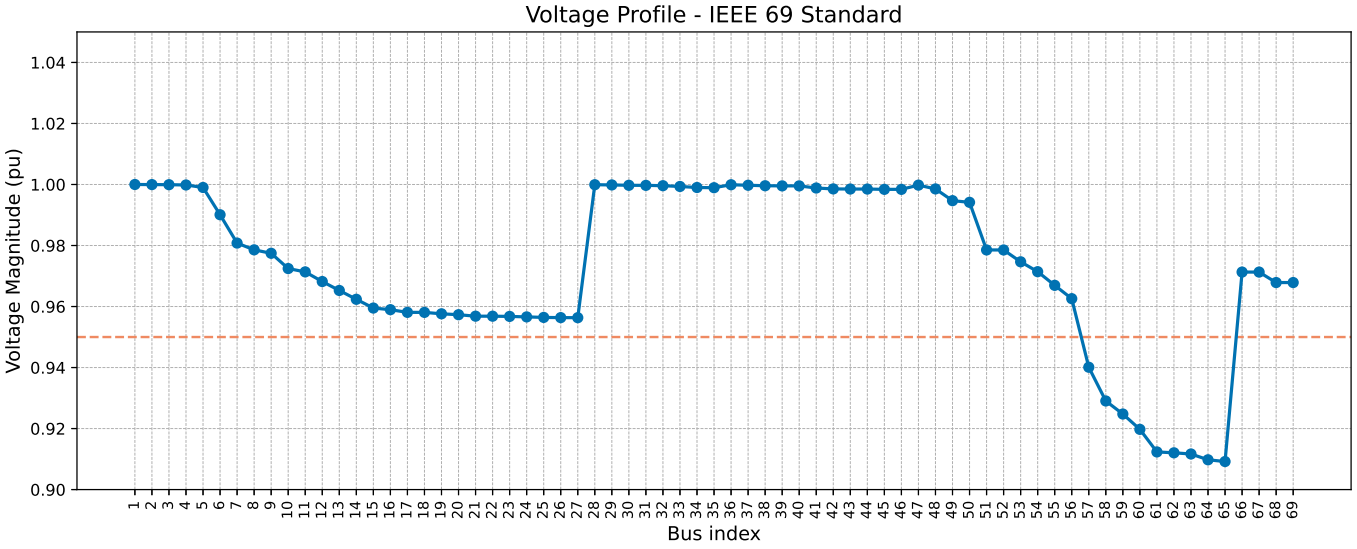


Figure 3.7: IEEE 69 network - Voltage profile

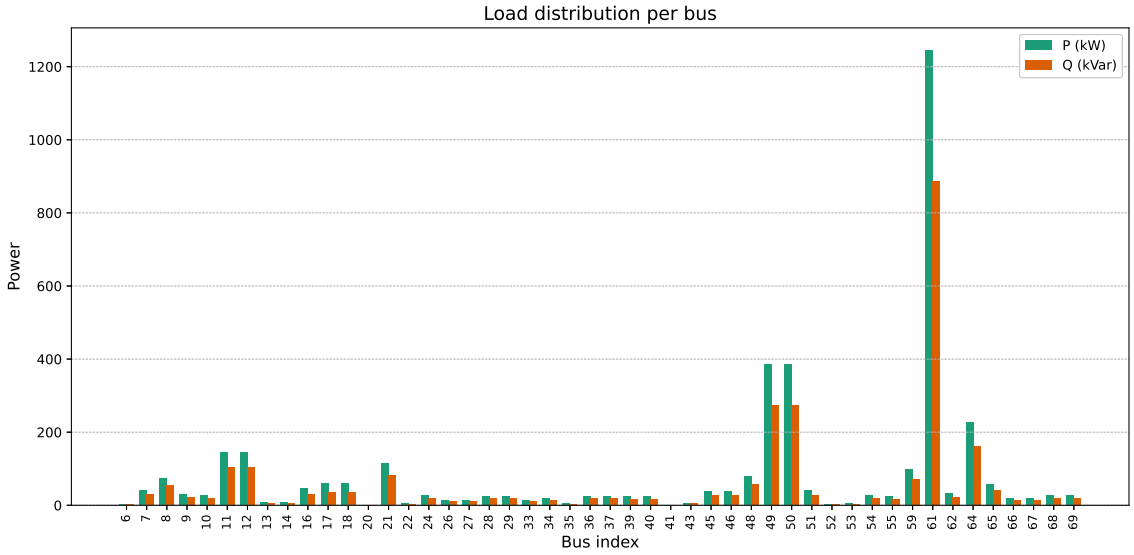


Figure 3.8: IEEE 69 - Load distribution per bus

3.3.2.1 Scenarios description

In the case of the IEEE 69 network, 6 scenarios are presented. As with the IEEE 33 network, the scenarios are characterized by whether or not the distributed generators are commissioned. For this network, 3 generators are used. Their capacities and locations are listed in Table 3.3.

Table 3.3: Characteristics of IEEE 69 network DGs

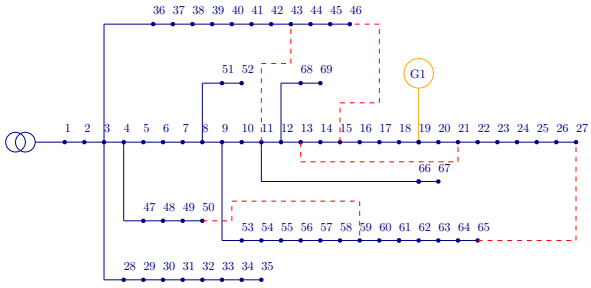
Generator	Capacity P_{max} [MW]	Location (Bus)
G1	0.45159	19
G2	1.72778	61
G3	0.51016	66

The different scenarios are presented in Table 3.4 and, as for the previous network, only the colored cells correspond to generators in service. The first scenario corresponds to the network without distributed generation as illustrated in Figure 3.6 and the other five are shown in Figure 3.9.

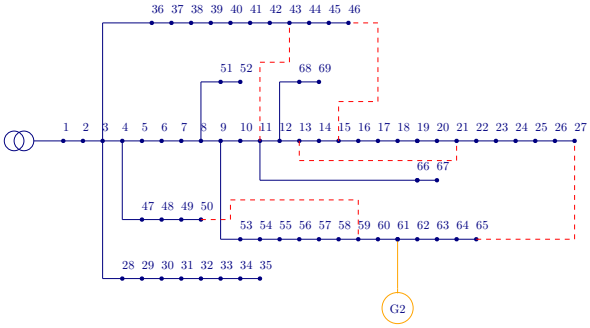
Table 3.4: Maximum capacity for generators by scenario (in MW)

Generator	**Scenario 1**	**Scenario 2**	**Scenario 3**	**Scenario 4**	**Scenario 5**	**Scenario 6**
G1	0	0,45159	0	0,45159	0,45159	0,45159
G2	0	0	1,72778	1,72778	0	1,72778
G3	0	0	0	0	0,51016	0,51016
Total [MW]	0	0,45159	1,72778	2,17937	0,96175	2,68953

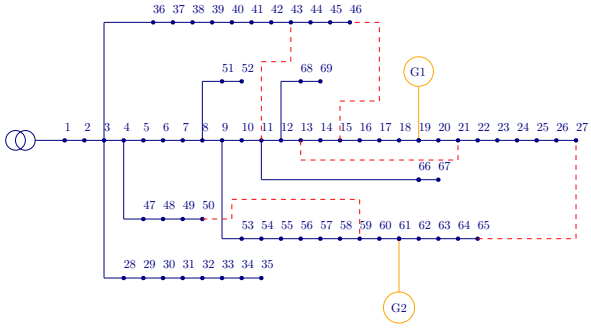
The tests carried out on the different scenarios ar described in the section 3.4.



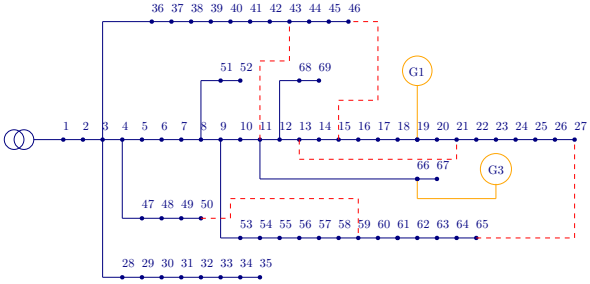
(a) Scenario 2



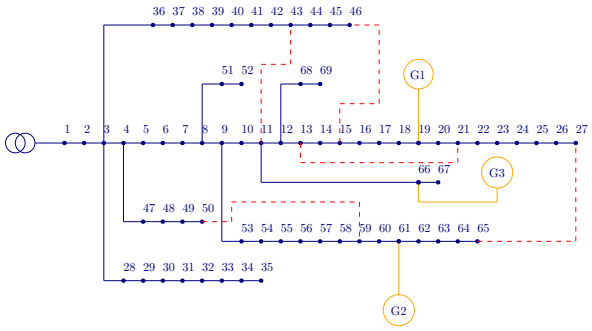
(b) Scenario 3



(c) Scenario 4



(d) Scenario 5



(e) Scenario 6

Figure 3.9: IEEE 69 network - Scenarios

3.4 Description of tests carried out

The testing methodology is that described in section 3.1. First of all, the first part of the tests is dedicated to the testing of different weights and the choice of the latter. Then, in the second part of this section, the power flow and optimal power flow methods are compared in order to choose the one that gives the best results.

3.4.1 Choice of weight

The methodology used to obtain the results of the various tests is described in the flowchart 3.10. As mentioned, three different weights are used for the minimum spanning tree:

- The negative active power losses of each line;
- The negative maximum current magnitude flowing through each line;
- The negative mean absolute active power flow through each line.

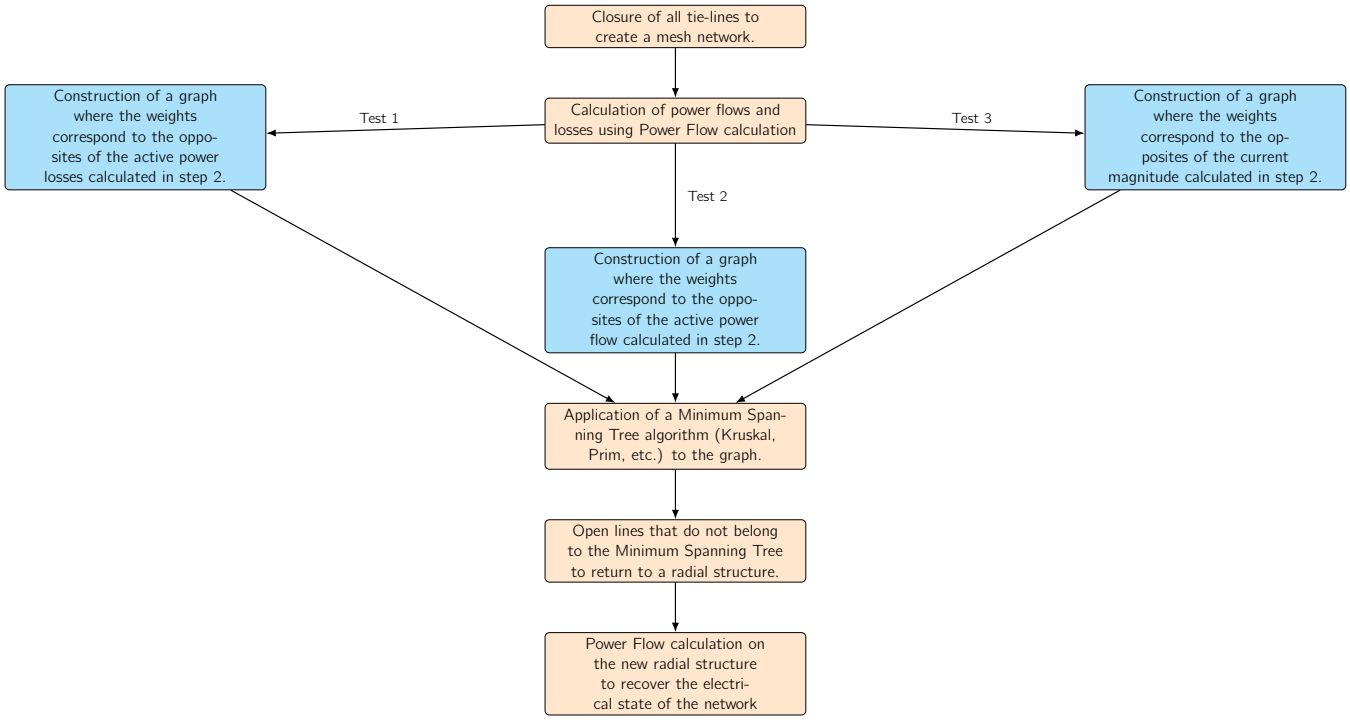


Figure 3.10: Description of weight tests

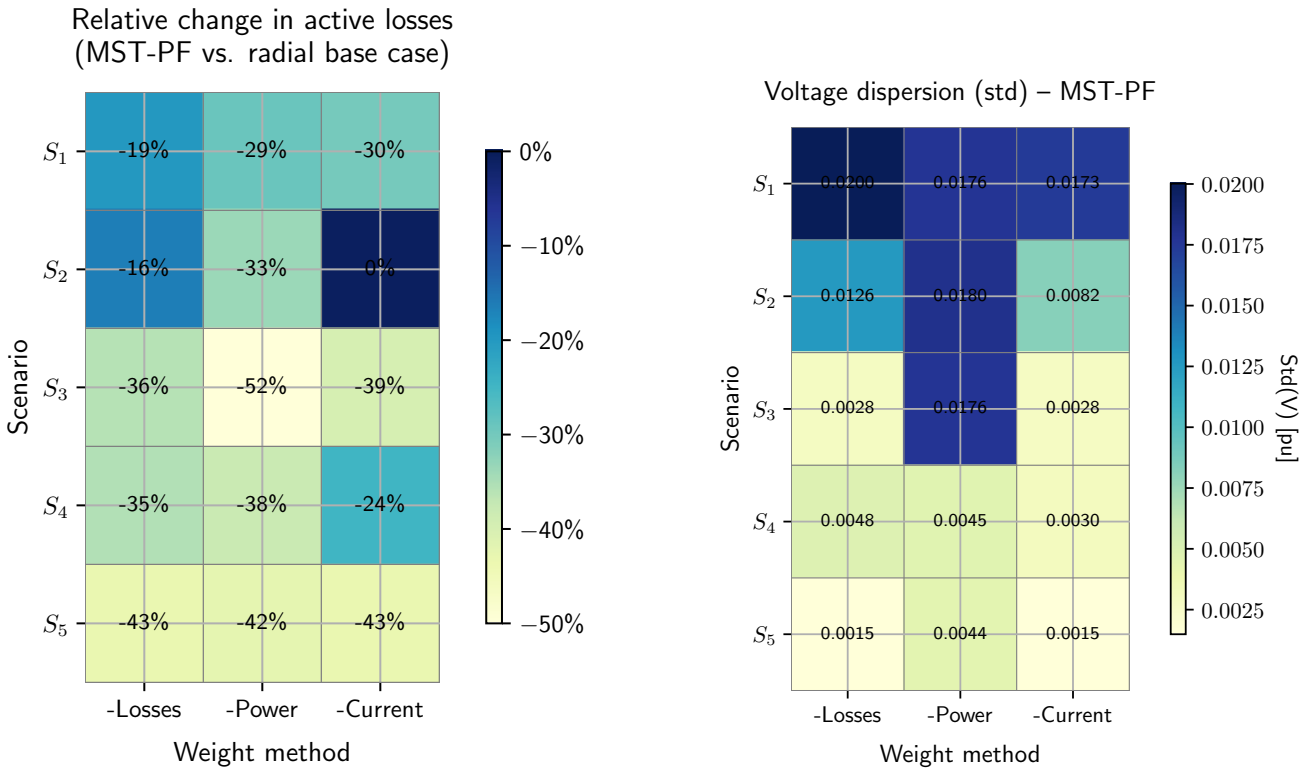
In order to compare the results obtained for each of the three weight choices, three measures are taken into account:

- The percentage decrease or increase in losses compared to the initial radial network: the more losses decrease, the better;
- The standard deviation of the voltages. A smaller standard deviation often indicates a better voltage profile;
- The minimum voltage: the higher the latter, the better.

The reason for using negative weights lies in the nature of the minimum spanning tree algorithm. In fact, the algorithm is designed to select the edges with the lowest cost, so using negative values allows us to prioritize the lines with the highest loads.

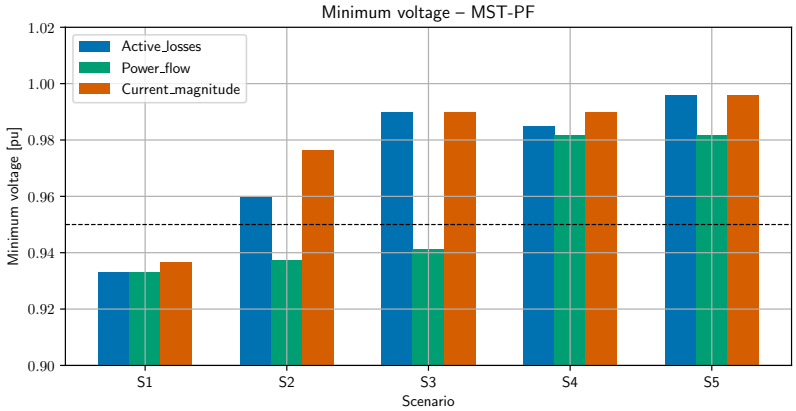
3.4.1.1 IEEE 33 network

The different results for the three metrics mentioned are represented in the set of Figures 3.11 for the IEEE 33 network.



(a) Relative change in losses

(b) Voltage dispersion



(c) Minimum voltage per weight method

Figure 3.11: Performance metrics per weight method - IEEE 33 network

The first figure, Figure 3.11a is a heatmap representing for each scenario (y-axis) and each weight type

(x-axis) the percentage increase or decrease in losses compared to the respective initial radial case. For example, the box corresponding to scenario 2 and the loss type -*Power* represents the percentage decrease between the basic radial network corresponding to scenario 2 and its reconfiguration by minimum spanning tree using the negative active power flow as weight. The first thing to notice on this figure is that for all weights and all scenarios, it is possible to observe a decrease in losses. This decrease is generally more pronounced for the weight corresponding to the negative active power flow.

The second figure, Figure 3.11b, is also a heatmap of scenarios and weights, but this time the metric observed is the voltage standard deviation (in pu). We can see that the weights -*Current* and -*Losses* often have lower voltage standard deviations than the cases where the weight used is the -*Power*. A lower voltage standard deviation generally represents a better voltage profile. However, this measurement cannot be considered individually, which is why in Figure 3.11c, we show a bar graph of the minimum voltage for each scenario and weight. It is visible that the weight corresponding to negative power flow often has the lowest minimum voltage. This is important, since in scenarios 2 and 3, while the other two weights manage to be within acceptable voltage ranges, the -*Power* weight fails. However, it is not possible to establish a true hierarchy in relation to the other two weights concerning minimum voltage.

3.4.1.2 IEEE 69 network

The metrics used to evaluate the effect of the weights taken into account on the IEEE 69 network are represented in the set of Figures 3.12.

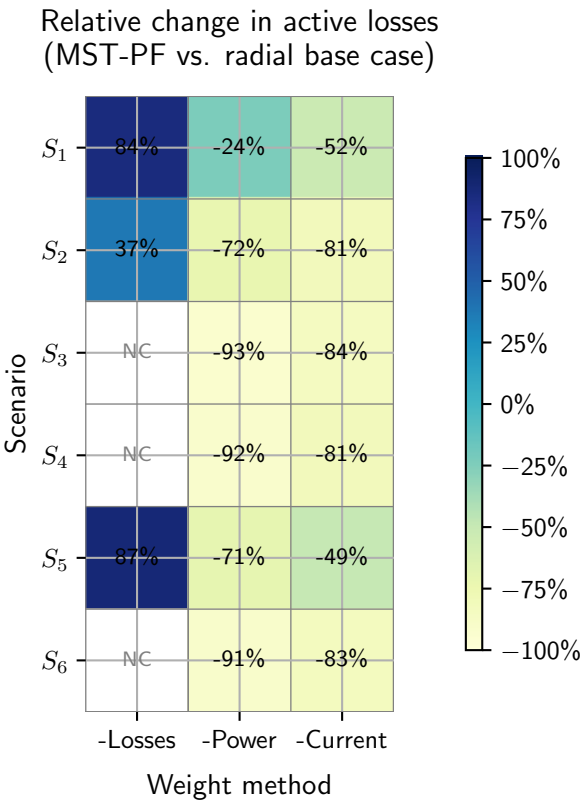
As with the IEEE 33 network, the first heatmap 3.12a shows the increase or decrease in losses after reconfiguration and compared with the basic radial case of each scenario. The first thing to notice about this heatmap is that using the -*Losses* weight completely breaks the algorithm. Indeed, for scenarios 3, 4 and 6, the power flow on the minimum spanning tree does not converge. For scenarios 1, 2 and 5, it does converge, but losses increase with reconfiguration.

The second heatmap, in Figure 3.12b, shows us that the standard deviation is rather high for cases where the weight is -*Losses* but is relatively close for the other two methods. Once again, we can look at Figure 3.12c to complete the analysis concerning voltage. Indeed, as in the IEEE 33 case, it is not possible to conclude from a hierarchization of performance at voltage level between the -*Power* and -*Current* methods. In some scenarios, the -*Power* method performs better, while in others it is-*Current* that takes the upper hand.

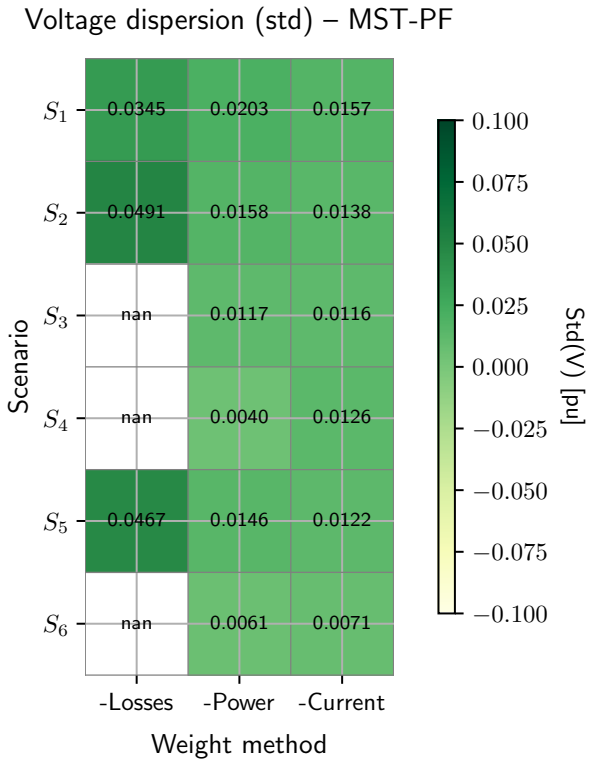
3.4.1.3 Conclusion on the choice of weight

On the basis of the results obtained and presented in the two previous sections, it is possible first of all to rule out the use of the -*Losses* weight. Indeed, it fails completely on the IEEE 69 network, leading to configurations that are either impractical or suboptimal.

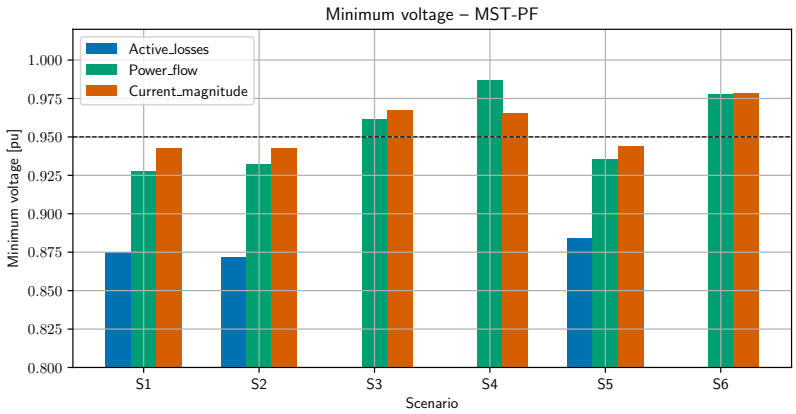
The -*Power* weight often gives correct results in terms of losses, making it possible to reduce them remarkably. However, it is also associated with a poorer voltage profile and a minimum voltage outside the acceptable range of values. In addition, the -*Power* weight approach is not reliable. Indeed, in the tests carried out, we used the average of the absolute values of the power flows at both ends of each of the lines. This represents the average power carried by each line. However, based on the tests carried out, we found that the difference between sending-end and receiving-end active powers is negligible. This means that using the average of the absolute values provides no additional information compared to using just the sending-end or just the receiving-end active power. This suggests that the -*Power* weight does



(a) Relative change in losses



(b) Voltage dispersion



(c) Minimum voltage per weight method

Figure 3.12: Performance metrics per weight method - IEEE 69 network

not reflect line conditions, which are better represented by the -*Current* weight.

Consequently, the weight chosen for the remainder of this study is the maximum current magnitude of the lines, i.e. the -*Current* weight. This approach offers a good compromise between lowering losses and improving the voltage profile in the network. Moreover, it aligns perfectly with the paper [51] where the use of this weight leads to successful results. This paper is discussed in more detail in the following sections.

3.4.2 Choice of power flow calculation method

In this section, the aim is to take the reasoning a step further and ask whether it would be possible to use optimal power flow (OPF) instead of power flow (PF), and whether the latter could generate better results. Reminders of the power and optimal power flow methods can be found in sections 1.1.1 and 1.1.2 respectively. The methodology used to obtain the results of the various tests is described in the flowchart 3.13.

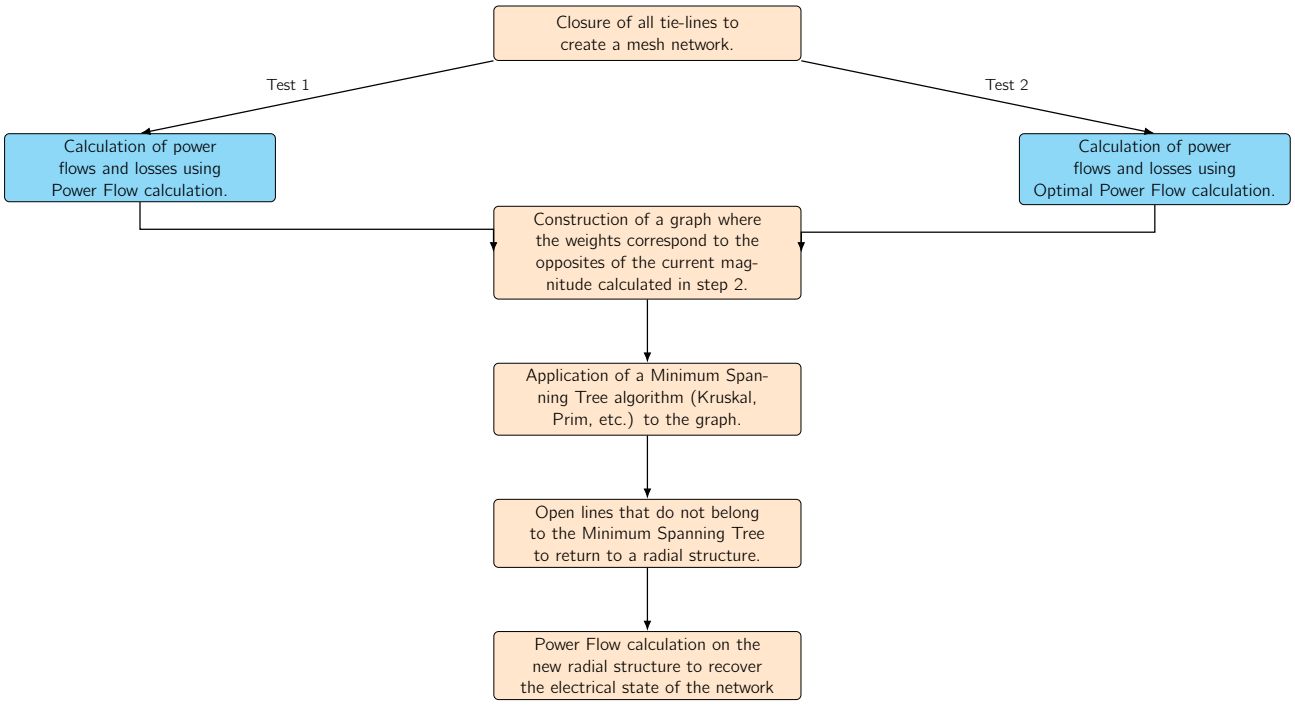


Figure 3.13: Description of PF-OPF tests

3.4.2.1 IEEE 33 network

The aim of this section is to compare the performance of the two tests presented on the IEEE 33 network. To do this, two criteria are taken into account: total active network losses and voltage profiles. The results of this comparison between methods are shown in Figure 3.14 and in the set of Figures 3.15.

Three networks are presented for each scenario (described in section 3.3.1.1). The first is the basic radial case, i.e. the configuration in which IEEE 33 operates, shown in Figure 3.2. The MST-PF configuration corresponds to the configuration obtained following application of the first test, and the MST-OPF configuration corresponds to the configuration obtained following application of the second test.

The first thing to discuss are the losses. These are shown in Figure 3.14. It is visible from the figure that the calculation from the first test (i.e. using the MST with the data obtained from the power flow)

results in a drop in losses in all scenarios. These reductions range from -20% to -43%. In contrast, the second test (i.e. using OPF data as the weight for the MST) does not always perform well. Sometimes it leads to a reduction in losses, as can be seen in scenarios 1, 3 and 5, but sometimes it leads to an increase in losses, which is particularly marked in scenario 4. It should be noted that in the first scenario, the application of MST-PF or MST-OPF amounts to exactly the same thing, since the latter contains no distributed generation.

The voltage, which should normally be measured in the range [0.95 pu, 1.05 pu] to ensure proper operation of the power network, sometimes falls below the minimum value. These few occasions occur in scenario 1, as well as in the base case of scenario 2. This is because the voltage, often regulated by the presence of reactive power, is first improved by the presence of distributed generators. This improvement in the voltage profile can be seen in Figure 3.15 as the scenarios progress. In the case of the IEEE 33 network, the test carried out has no impact on the voltage profile. It is noticeable that the curves corresponding to voltage vs. bus for the MST-PF and MST-OPF methods have relatively the same shape and are almost superimposable.

In the case of the IEEE 33 test network, the use of the minimum spanning tree results in some improvements. The MST-PF test reduces losses and improves the voltage profile in all cases presented. The MST-OPF test, on the other hand, does not always reduce losses. On the contrary, losses sometimes increase. However, like the MST-PF test, it ensures a remarkable improvement in the voltage profile in all scenarios. In conclusion, the minimum spanning tree, and more concretely the one using as weight the opposite of the maximum line voltage, obtained by power flow calculation, helps to improve the voltage profile and reduce losses in the case of IEEE 33.

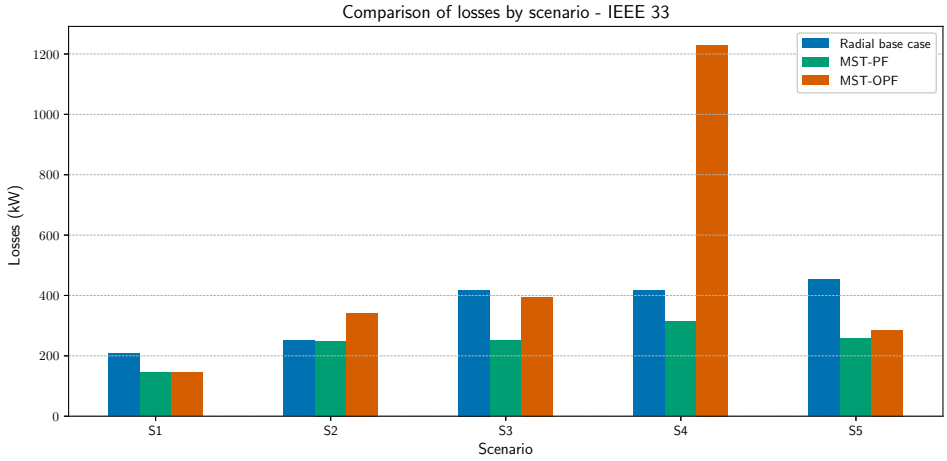
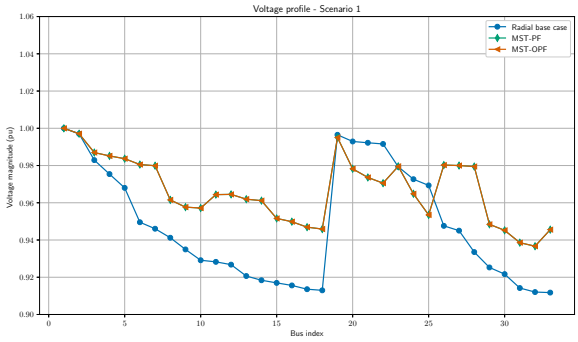


Figure 3.14: Losses (kW) by scenario - IEEE33

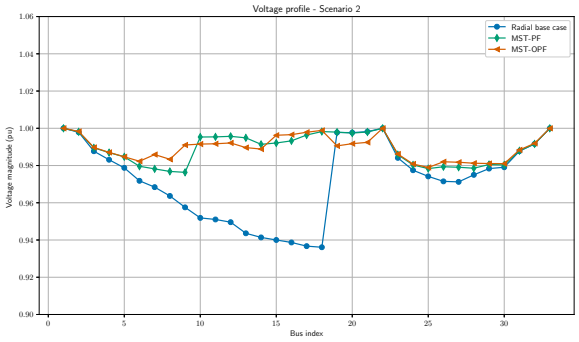
3.4.2.2 IEEE 69 network

The results of experiments carried out on the IEEE 69 network can be found in Figure 3.16 as well as in the Figure set 3.17. As in the previous case, each scenario has three networks. The first network is the IEEE 69 as it operates from the ground up and as represented in Figure 3.6. The second and third networks are derived from the tests explained above. The results are the same as for IEEE 33, for the 6 scenarios presented in section 3.3.2.1.

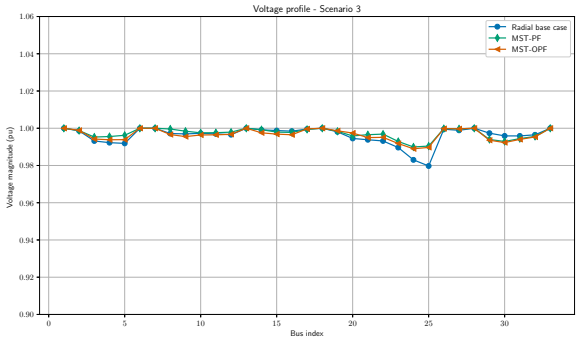
Losses are shown in Figure 3.16. As with the IEEE 33 network, the MST-PF method reduces losses



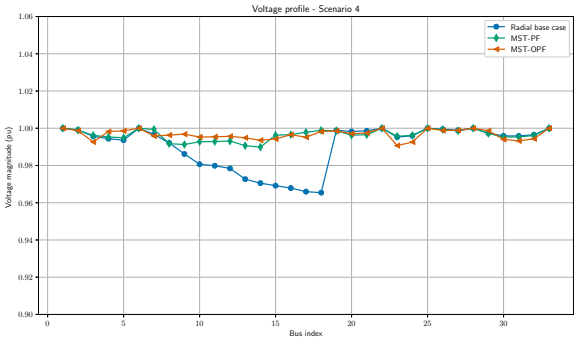
(a) Scenario 1



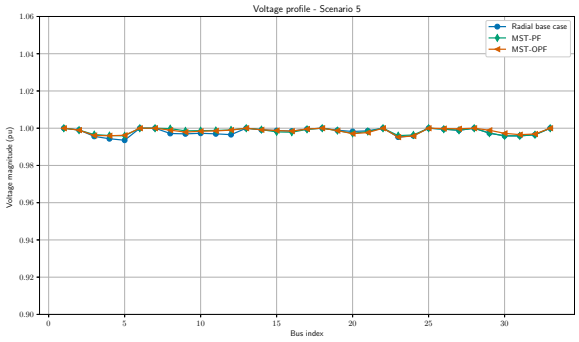
(b) Scenario 2



(c) Scenario 3



(d) Scenario 4



(e) Scenario 5

Figure 3.15: Voltage profile comparison - IEEE 33 scenarios

in all scenarios. However, as in the IEEE 33 case, using the MST-OPF method does not offer stable results. It can be seen that for some scenarios, such as 1, 2 and 4, the method reduces losses (in one case, scenario 4, to a greater extent than the MST-PF method). However, in the other scenarios, losses increase.

Regarding the voltage profile, Figure 3.17 shows that the MST-OPF method improves the voltage profile in all scenarios. The MST-PF method, on the other hand, is not always the best. Particularly in scenarios 4 and 6, we can see that it diverges from the MST-OPF method. In addition, as in the IEEE 33 case, the addition of distributed generators improves the voltage profile compared with the basic scenario 1, which has no generators.

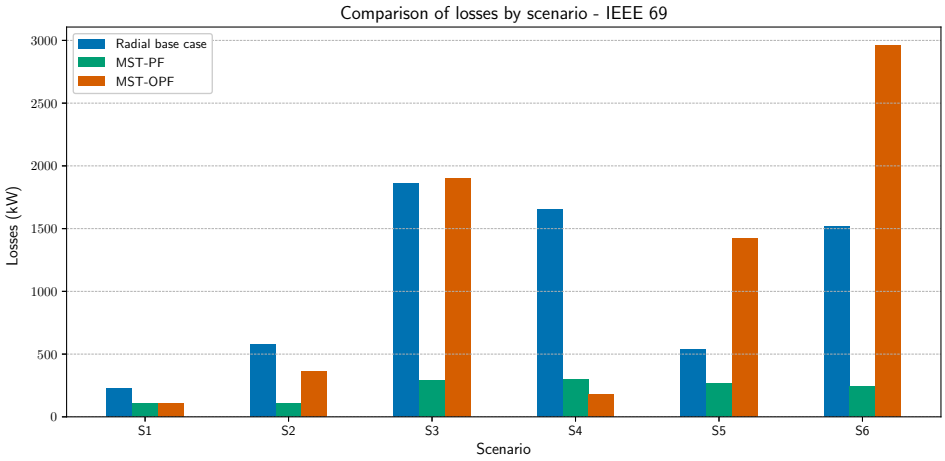
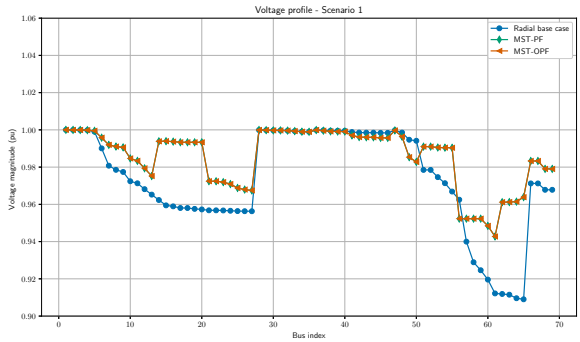
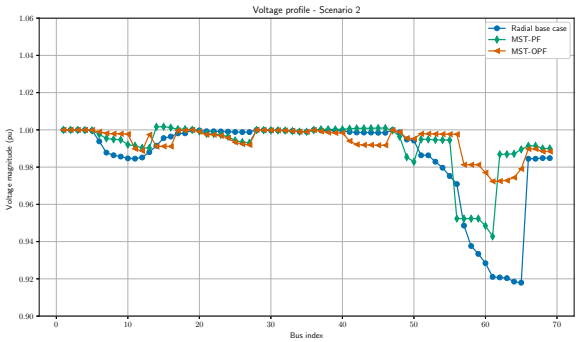


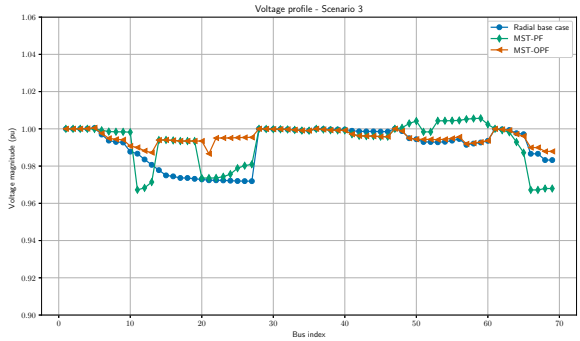
Figure 3.16: Losses (kW) by scenario - IEEE69



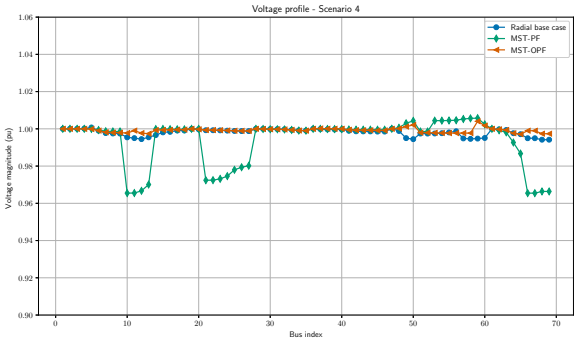
(a) Scenario 1



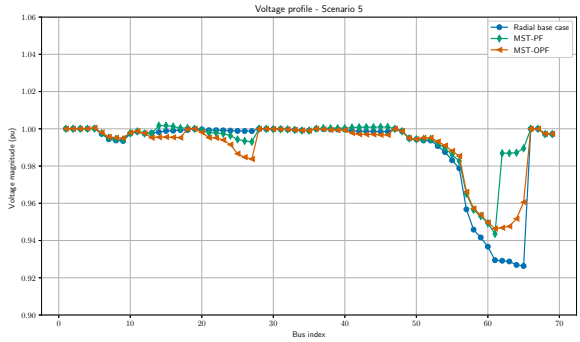
(b) Scenario 2



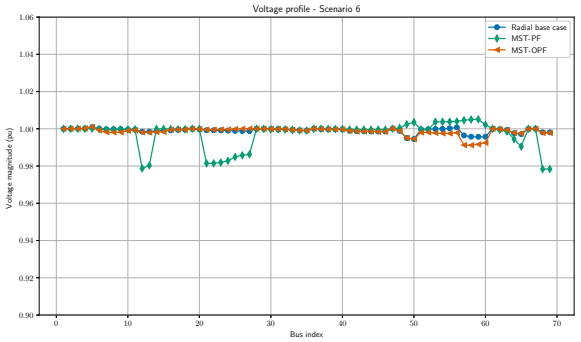
(c) Scenario 3



(d) Scenario 4



(e) Scenario 5



(f) Scenario 6

Figure 3.17: Voltage profile comparison - IEEE 69 scenarios

3.4.2.3 Conclusion on power flow calculation method

In view of the results obtained for the IEEE 33 and IEEE 69 networks, we can easily confirm that the MST-PF method is better suited to solving the problem of reconfiguring distribution networks. Indeed, in all the scenarios presented, whether for the 33-node network or the 69-node network, this method reduces total active losses. What is more, in the majority of cases, although not systematically, it improves the voltage profile compared to the base radial configuration. On the other hand, the MST-OPF method, although capable of offering better voltage profile performance in the IEEE 69 network, is highly unstable in its active loss results and does not ensure consistently positive behavior.

In conclusion, the MST-PF method are retained for the remainder of the study.

3.5 Results and discussion

The aim of this section is to evaluate the performance of the method based on the minimum spanning tree, which uses the maximum current of each line obtained by calculating power flow as the weight of the Kruskal algorithm. Results are presented on IEEE 33 and IEEE 69 test networks.

3.5.1 Results on the IEEE 33 network

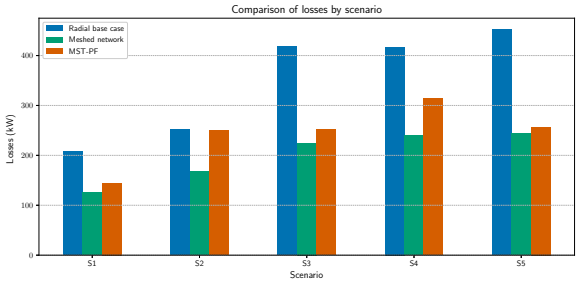
The results for each method and scenario are presented in Table 3.5. Two comparative graphs are also available in Figure 3.18. The first represents the losses for the three cases studied, and the second the value of the voltage standard deviation for each scenario and each test. The MST-PF method significantly reduces losses compared with the basic radial configuration. Scenario 5 shows the greatest reduction, i.e. minus 43.25% in losses. On the other hand, we can see that scenario 2 does not see its losses reduced significantly, since they only drop by 0.76%.

With regard to voltage profiles, these are systematically improved by the MST-PF method when compared with the basic radial case. In particular, minimum voltages are higher. For example, for scenario 1, we go from a minimum voltage of 0.91 pu to a minimum voltage of 0.94 pu. Although the MST-PF method improves voltage profiles remarkably, it never exceeds the results of the fully meshed network, i.e. with all tie lines closed. In fact, the mesh network always has the highest minimum voltages and the lowest voltage standard deviations. In fact, this network is not usable since distribution networks operate in a radial configuration. However, it serves as a reference point for comparison. In the scientific paper [51], the authors assume that the optimal solution is close to the results offered by the mesh network. The voltage standard deviation is also improved by the MST-PF method, especially in scenarios 1 and 2.

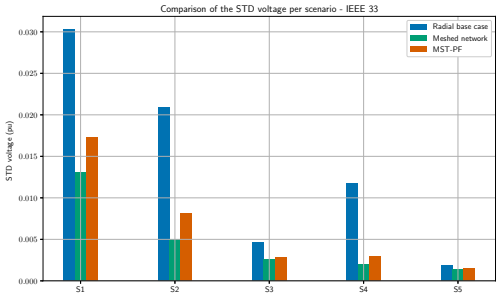
The results obtained with the MST-PF method show a good compromise between optimality, represented by the mesh network, and the constraints linked to operation under radial configuration. In fact, they confirm that maximum current is a relevant and reliable criterion for weighting a graph, as it is representative of real network conditions.

Table 3.5: Comparison of performances by scenario and method - IEEE 33

Scenario	Test	Losses (kW)	% losses change vs. base case	Vmin (pu)	Vmax (pu)	STD voltage
S1	Radial base	207,9	–	0,9118	1	0,0303
	Meshed network	126,0	-39,39	0,9501	1	0,0131
	MST-PF	144,9	-30,30	0,9367	1	0,0173
S2	Radial base	251,6	–	0,9361	1	0,0209
	Meshed network	167,7	-33,35	0,9826	1	0,0049
	MST-PF	249,7	-0,76	0,9763	1	0,0082
S3	Radial base	418,5	–	0,9797	1	0,0047
	Meshed network	225,0	-46,24	0,9905	1	0,0026
	MST-PF	251,2	-39,98	0,9899	1	0,0028
S4	Radial base	417,1	–	0,9654	1	0,0117
	Meshed network	239,7	-42,53	0,9940	1	0,0019
	MST-PF	313,3	-24,89	0,9899	1	0,0030
S5	Radial base	452,3	–	0,9935	1	0,0019
	Meshed network	243,6	-46,14	0,9957	1	0,0014
	MST-PF	256,7	-43,25	0,9959	1	0,0015



(a) Losses per scenario and per test - IEEE 33



(b) STD voltage per scenario and per test - IEEE 33

Figure 3.18: IEEE 33 network - Results

3.5.2 Results on the IEEE 69 network

Table 3.6 shows all the results for the IEEE 69 network. Figures 3.19 support the results shown in the table. In the case of the IEEE 69 network, the MST-PF method delivers even greater improvements in terms of loss reduction. These range from 49% less losses than in the radial case to -83% in scenario 6. This could indicate a high performance of the MST-PF method in complex and complete networks.

In terms of voltage profile, the results are not always as good. Scenarios 1, 2 and 5 see their minimum voltages increase. However, the voltages do not reach the minimum value of 0.95 pu. This is one of the points highlighting the limitations of the MST-PF method on its own. Indeed, local optimizations may be possible. Scenarios 3, 4 and 6 show favorable results, with voltages always above the 0.95 pu threshold.

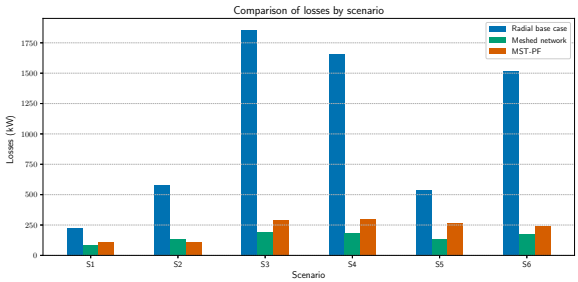
The method is therefore effective even with DG penetration in the network.

When we compare the results obtained with the MST-PF method with those obtained with the mesh network, we can see that the MST-PF manages to reduce losses considerably. These losses are almost the same order of magnitude as those of the mesh configuration. This applies to both losses and voltages.

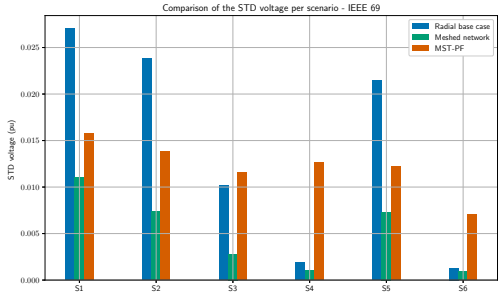
Overall, the results obtained on the IEEE 69 network show that the MST-PF method presented in this chapter performs rather well.

Table 3.6: Comparison of performances by scenario and method - IEEE 69

Scenario	Test	Losses (kW)	% losses change vs. base case	Vmin (pu)	Vmax (pu)	STD voltage
S1	Radial base	226,1	–	0,9090	1	0,0271
	Meshed network	86,4	-61,79	0,9524	1	0,0110
	MST-PF	106,5	-52,90	0,9427	1	0,0157
S2	Radial base	575,5	–	0,9179	1	0,0238
	Meshed network	130,8	-77,27	0,9723	1,0001	0,0074
	MST-PF	107,4	-81,34	0,9428	1,0017	0,0138
S3	Radial base	1858,1	–	0,9719	1,0004	0,0102
	Meshed network	190,3	-89,76	0,9917	1,0008	0,0028
	MST-PF	290,2	-84,38	0,9672	1,0057	0,0116
S4	Radial base	1656,6	–	0,9942	1,0007	0,0019
	Meshed network	182,5	-88,98	0,9963	1,0010	0,0010
	MST-PF	299,7	-81,91	0,9655	1,0057	0,0126
S5	Radial base	534,7	–	0,9263	1,0002	0,0215
	Meshed network	130,4	-75,61	0,9734	1,0004	0,0073
	MST-PF	269,6	-49,58	0,9436	1,0017	0,0122
S6	Radial base	1514,5	–	0,9945	1,0009	0,0011
	Meshed network	179,0	-88,18	0,9965	1,0012	0,0009
	MST-PF	243,8	-83,90	0,9784	1,0051	0,0071



(a) Losses per scenario and per test - IEEE 69



(b) STD voltage per scenario and per test - IEEE 69

Figure 3.19: IEEE 69 network - Results

3.6 Exhaustive exploration of the spanning trees of the IEEE 33 network

In the previous section, we made the assumption that the mesh network was always the one with the fewest losses. In this section, we naively analyze all possible trees from the IEEE 33 mesh network. It is important to point out that this type of analysis is not feasible for a network with a significant number of nodes or tie-lines, and even less so in real time. Computing all trees exhaustively is a computationally demanding operation. However, we do it for the IEEE 33 network since it is a network with a reasonable number of nodes and a reasonable number of tie-lines, too, in order to assess whether the MST as defined in this chapter is close to optimal and to have a set of exhaustive results. The IEEE 33 network has 32 operational lines and 5 tie-lines. The number of trees from this data is 50 751.

To do this, we perform the following steps:

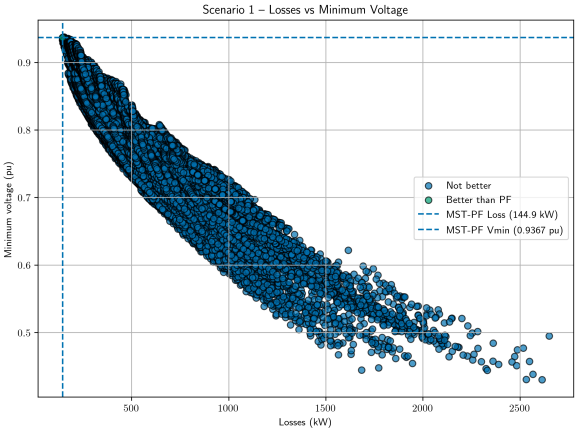
1. For each scenario, we generate all possible trees from the mesh network on which the powerflow converges,
2. For each tree, we record the minimum voltage v_{min} in pu, the total losses P_{loss} in kW and the standard deviation voltage in pu,
3. We compare each of the configurations obtained with the configuration obtained by MST and count the number of trees better than the MST. To be better, three conditions must be met: the minimum voltage must be strictly greater than that of the MST, the amount of losses must be strictly less than the MST and the standard deviation of the configuration must be strictly lower than the one of the MST,
4. A scatter plot is drawn for each scenario. The y-axis shows the minimum voltage in pu and the y-axis shows the losses in kW. The figures can be found in the Figure set 3.20.

The first thing to analyze is the number of trees better than those obtained by MST-PF per scenario. The results are shown in Table 3.7. The first column corresponds to the scenario. The second column shows the number of trees on which the power flow calculation converges out of a possible 50 751, and the third column shows the number of trees better than MST-PF. The first thing to remark is that the number of trees converging is relatively similar for all scenarios except the first, where fewer trees correspond to feasible configurations. Next, we note that in two cases the tree obtained by MST-PF is the best, and in one case it is the second best. Finally, for scenarios S2 and S4 it is not the best tree, but it is still in the top 2% of best trees.

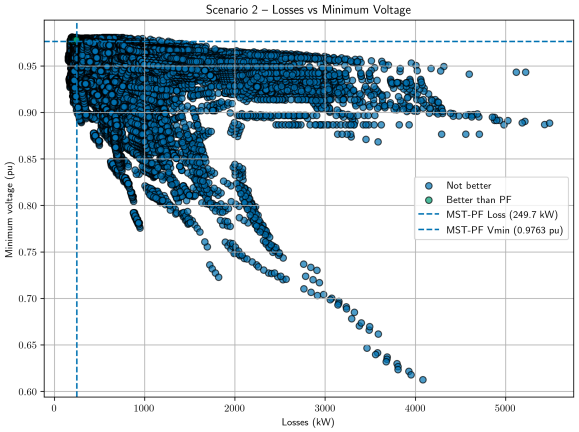
Table 3.7: Convergent trees and better trees than MST-PF

Scénario	**Convergent trees**	**Better than MST-PF**
S1	44 803	1
S2	50 424	988
S3	50 675	0
S4	50 673	229
S5	50 751	0

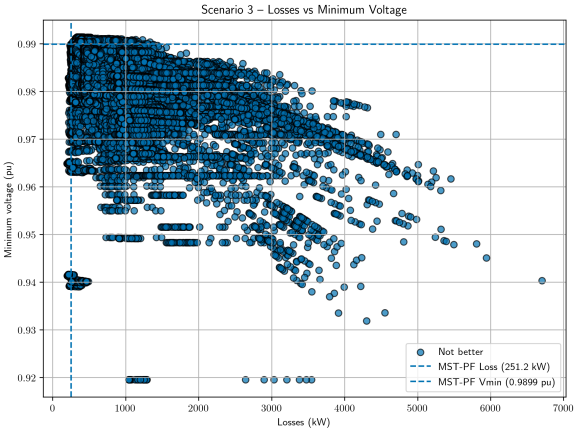
The results in the table and the figures in the set 3.20 show how, among the number of feasible trees, the minimum spanning tree stands out. We have no reason to generalize to other networks and expect this behavior to be propagated, but this naive approach gives us a point of reference and reinforces the legitimacy of using the MST algorithm as presented as a starting point for more advanced methods.



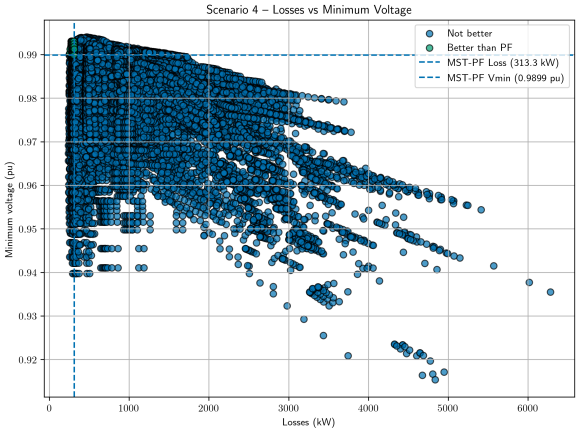
(a) Scenario 1



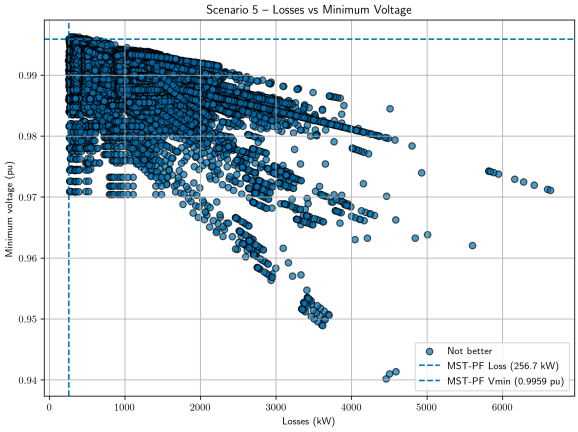
(b) Scenario 2



(c) Scenario 3



(d) Scenario 4



(e) Scenario 5

Figure 3.20: Losses vs minimum voltage for all the spanning trees of IEEE 33 network

3.7 Conclusion of the chapter

In this chapter, we have had the opportunity to study in detail an approach to distribution network reconfiguration based on the Minimum Spanning Tree. The implementation is based on the Kruskal algorithm and uses the opposite of the maximum current per line, obtained from the power flow calculation, as a weighting criterion for the Kruskal algorithm. All the results for IEEE 33 and IEEE 69 networks show that this method is particularly effective.

The Minimum Spanning Tree method offers, in addition to encouraging results in terms of reducing total active losses and improving the voltage profile, the advantage of being easy to implement. As a reminder, on the IEEE 33 network, we saw loss reductions of up to -43% compared with the basic operational case. On the IEEE 69 network, this percentage is even more encouraging, with losses reduced by up to 83% compared with the radial base case. This shows the relevance of this method for larger networks.

Despite the remarkable performance of the MST-PF method, it can be seen that the voltage profile is degraded for the IEEE 69 network. Indeed, we sometimes remain below the minimum voltage threshold required for proper network operation. These results suggest that there is still room for improvement.

The next chapter therefore present more advanced search approaches. In particular, local search, inspired by the methodology described in the scientific paper [51]. The next objective is to improve the response already obtained using the Minimum Spanning Tree method presented in this chapter.

Chapter 4

Minimum spanning tree refinement using heuristics

4.1 Introduction

In the previous chapter, we saw that the Minimum Spanning Tree reconfiguration method was already producing adequate results. In most scenarios, losses were reduced and the voltage profile improved. However, the simplistic nature of MST opens the door to the exploration of other algorithms with the objective of capturing the non linearities.

The aim of the current chapter is to refine the answer obtained by Minimum Spanning Tree by using well-known heuristics. The first heuristic is inspired by the scientific paper [51] and is called local search in the paper. The second heuristic is the tabu search. Several papers such as [52] and [53] discuss it in the context of reconfiguration, often with further improvements of the tabu search as known from the basic [54]. The principles are explained in the next section and pseudo-codes are presented.

Finally, the idea is to start from a first fast solution obtained through MST calculation and improve it using the methods mentioned. The tests are described and tested on the test networks presented in section 3.3.

4.2 Presentation of the algorithms

The aim of this first part of the chapter is to present the algorithms that are used in the tests. First, an MST calculation determine the starting configuration. This is refined firstly by local search, presented as a branch exchange as in [51]. Finally, two variants of tabu Search are be presented and used. Each step includes a theoretical reminder and pseudo-code to facilitate algorithmic understanding. The notations to be used in the various explanations are listed in Table 4.1.

Table 4.1: Notations used in the algorithms presented

Symbol	Unit	Description
Network topology		
V	-	Set of network nodes
$\mathcal{E}$	-	Set of network lines
e_{ij}	-	Line connecting the node i to j
$\mathcal{N}$	-	Complet network
Electrical values obtained after Power Flow calculation		
$\lvert I_{ij}^{\text{from}}\rvert$, $\lvert I_{ij}^{\text{to}}\rvert$	kA	Current values measured at both ends of the line (i,j)
I_{ij}	kA	Line current (i,j)
V_{min}, V_{max}	pu	Minimum/maximum voltage after PF
σ_V	pu	Standard deviation of voltage
P_{loss}	kW	Total active losses on the network
MST		
w_{ij}	-	Weight assigned to line (i,j) for the MST ($w_{ij} = -\max\lvert I_{ij}\rvert$)
Local search and tabu search		
S_{courant}	-	Current configuration
$S^\star$	-	Best configuration found
$\mathcal{N}(S)$	-	Solution neighbourhood S (*branch-exchange*)
T	-	Tabu list (FIFO queue of forbidden lines)
L	itér.	Maximum tabu list size
k_{max}	itér.	Number of iterations allowed
ε	kW	Minimum improvement threshold (TS-v1)
δ	pu	Minimum voltage improvement threshold V_{min} (TS-v2)
$patience$	-	Tabu Search stop variable

4.2.1 Starting solution by MST

As explained in the previous chapter, an initial configuration is obtained from the Kruskal algorithm. This algorithm calculates the Minimum Spanning Tree of a weighted graph, i.e. the weight with the smallest sum of edge costs. The principle of this algorithm is as follows: we start with the meshed network, i.e. the network with all tie-lines closed. We perform a power flow calculation and weight each edge e_{ij}, linking nodes i and j with

$$w_{ij} \;=\; -\max\left(\lvert I_{ij}^{\text{from}}\rvert, \lvert I_{ij}^{\text{to}}\rvert\right),$$

which corresponds to the opposite of the maximum current in a line. As a reminder, the negative sign allows us to keep the most overloaded lines as a priority. Finally, we perform an MST calculation using Kruskal's algorithm to obtain our new radial configuration.

Algorithm 2: Minimum Spanning Tree reconfiguration

Input: Meshed network $\mathcal{N} = (V, \mathcal{E})$
Output: Radial network $\mathcal{T}$ (MST)

Compute RunPF($\mathcal{N}$) ;
foreach $e_{ij} \in \mathcal{E}$ **do**
  $w_{ij} \leftarrow -\max(|I_{ij}^{\text{from}}|, |I_{ij}^{\text{to}}|)$
$\mathcal{T} \leftarrow \text{Kruskal}(V, \mathcal{E}, w)$;
return $\mathcal{T}$

The complexity of Kruskal's algorithm is $\mathcal{O}(E \log E)$ with E the number of edges in the graph [55].

4.2.2 Local search by branch exchange

Once the initial solution has been determined, the next step is to apply a local search. In the case of this study, we apply the one proposed in the paper [51]. This is based on the principle of branch exchange. The principle behind this type of algorithm is explained in section 1.3.1.1 and is described in detail here. Papers such as [33] or [56] use the Branch exchange algorithm. The main aim is to explore the neighborhood of the solution obtained by MST. To do this, solutions close to the optimal solution, i.e. differing by one line, are evaluated to see whether they constitute better configurations, i.e. whether they reduce losses more remarkably and improve the voltage profile.

The implementation used in this work applies the following principle:

1. We identify all open lines;
2. For each open line, we look at the adjacent lines that could form a cycle if they were closed;
3. We explore all possible branch exchange scenarios, i.e. all pairs (open line, closed line) leading to new radial configurations. These scenarios are called neighbors;
4. For each configuration, we perform a powerflow computation, check radiality and connectivity, and measure total active losses;
5. If the switch reduces losses by at least the ε improvement threshold (whose value is 10^{-3} in our case), we retain the configuration;
6. The previous steps are repeated as long as a better solution can be found or the maximum number of iterations is reached.

What is important to remember is that only one criterion is taken into account when using this local search. The improvement criterion is the minimization of total active losses P_{loss} in kW. However, a voltage check is added at the end to avoid proposing configurations that are not feasible in terms of voltage.

The pseudo-code of the algorithm can be written as follows:

Algorithm 3: Local search - Branch exchange

Input: Initial network S_{courant}, number of iterations allowed $k_{\max}$
Output: Improved configuration S_{current}

```
for k = 1 to k_max do
    foreach open line l_c (tie-line) do
        foreach open line l_o creating a cycle with l_c do
            Building a new configuration S' closing l_c and opening l_o;
            if S' is radial and connected then
                if P_loss(S') < P_loss(S_current) − ε then
                    S_current ← S';
                end
            end
        end
    end
end
return S_current
```

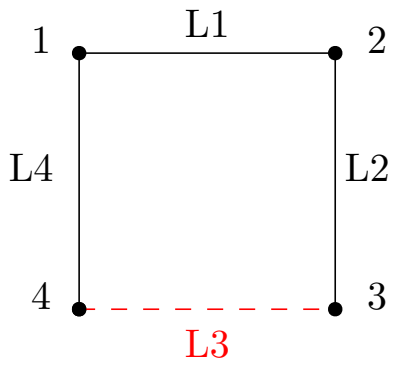


Figure 4.1: Example of a local branch-exchange search: L3 is open and closing it creates cycles. Sequentially, we try to open L1, L2 and L4. Each time a better configuration is reached, it is retained.

4.2.3 Tabu Search

The second proposed heuristic is tabu Search. It was first introduced in the 80s by an american mathematician, Fred Glover [57] [58]. In this project, it is used in addition to local search to refine the answer obtained after the second step. It is therefore the third algorithm used. Tabu search is similar to local search, but differs from the latter in that it integrates a memory called the Tabu list, which is a list of size L. This memory records the last exchanges made, i.e. the last lines opened. Moves that include these lines are forbidden unless they satisfy an aspiration criterion, such as a significative loss improvement in our case. In contrast to local search, Tabu Search allows moves that do not improve the solution, in order to cope with the greedy nature of local search, which could lead to local optima. This method is explained and used in the context of reconfiguration in the paper [53].

In this work, two variations of tabu search are explored. The difference between the two variants is the objectives.

4.2.3.1 Tabu Search - variant 1: Single objective

The first variant of tabu search is based on a single, simple objective and criterion: the reduction of total active losses P_{loss}.

The principle of the algorithm is as follows:

1. All neighbors are identified in the same way as in the local search. We scan the set of open lines and evaluate all open-closed line pairs in the cycles;
2. For each neighbor we compute PowerFlow and recover the total active losses P_{loss} of the configuration.
3. If the open line involved in the proposed exchange is part of the tabu list, the new configuration is authorized only if it meets the aspiration criterion, i.e. if the total losses are lower than those of the best known global solution.
4. At each iteration, we select the neighbor that generates the fewest losses, even if it generates more than the current solution. We then replace the current solution with this neighbor;
5. If the neighbor is better than the global solution, it replaces the current best global answer and the line used is added to the tabu list.

The difference with local search, in addition to the addition of a memory, is the selection of the neighbor that improves losses the most after evaluating them all. More formally:

$$f_1(S) = P_{\text{loss}}(S), \quad \text{accepted if } f_1(S) < f_1(S^\star) - \varepsilon$$

with :

- $f_1(S)$ is the objective function associated with the configuration S;
- $P_{\text{loss}}(S)$ represents the total active losses of the network, expressed in kW;
- $S^\star$is the best current solution;
- ε is a minimum improvement threshold, allowing negligible variations to be ignored;

This method does not take into account the voltage profiles in the objective, but performs an a posteriori check to verify the validity of the solution proposed by tabu search.

In addition, there is a variable implemented in the algorithm called *patience*. This variable corresponds to the number of iterations without any improvement. If this number exceeds the *patience* variable, the algorithm is stopped prematurely.

4.2.3.2 Tabu Search - variant 2: Multi-objective

The second variant of tabu search has a dual objective: in addition to reducing total active losses, its aim is to improve the voltage profile. To achieve this, a Pareto filter is applied. The principle of this filter is explained in section 1.2.3.

The algorithm works in exactly the same way as variant 1, the only difference being in the criteria used to select the overall solution. Indeed, $S^\star$ is only updated if the neighbor S' satisfies the following two conditions:

$$\big(P_{\text{loss}}(S'),\ \sigma_V(S')\big) \prec \big(P_{\text{loss}}(S^\star),\ \sigma_V(S^\star)\big) \quad \text{and} \quad V_{\min}(S') \geq V_{\min}(S^\star) - \delta$$

where :

- $P_{\text{loss}}(S)$ represents the total active losses of the network,
- $\sigma_V(S)$ is the standard deviation of the voltages,
- $V_{\text{min}}(S)$ is the lowest voltage magnitude,
- δ is a very small tolerance on the minimum voltage (10^{-4} p.u. in our case).

It should be noted that the condition used during exploration remains the reduction of losses, but a response is only accepted if it is filtered by our Pareto filter, which means that the best solution is only improved if both objectives are taken into account.

The notation $\prec$ alludes to strict Pareto domination [59], except for V_{min}. Indeed, V_{min} can decrease as long as it remains within the admissible range of values. Note also that σ_V is calculated in pu.

This second variant not only reduces losses but also improves the voltage profile. In the paper [51] the assumption made is that any improvement in losses generate an improvement in voltage. However, this assumption is not always valid, as can be seen in Table 3.6. In scenarios 3 and 6, for example, MST reconfiguration reduces losses, but the voltage profile is degraded. This is why a second variant of Tabu Search is proposed here.

4.2.3.3 Tabu search : pseudo-code

The two variants of the Tabu search differ in their criterion for selecting the best solution. The pseudo-code of the algorithm is the same for both, and only the improvement criterion changes. This can be written as follows:

Algorithm 4: Tabu Search

Input: Initial solution S_{current} (MST + LS), tabu list $T = \varnothing$
Result: Best global solution $S^\star$

```
Initialize S* ← S_current;
Initialize P_loss(S*), σ_V(S*), V_min(S*);
for k = 1 to k_max do
    Generate the set 𝒩 of neighbors of S_current with branch-exchange;
    Select the best eligible neighbor S' ∈ 𝒩 (tabu unless aspiration);
    S_current ← S';
    if S' meets improvement criteria (TS-v1 or TS-v2) then
        S* ← S';
    Add open line to tabu list T.Push();
    if no overall improvement since patience iterations then
        Stop the loop
return S*
```

4.3 Description of tests carried out

This section is dedicated to the description of tests carried out to reconfigure distribution networks using the algorithms described in the previous section. Three tests are carried out:

1. Application of a local search on the configuration obtained by MST;
2. Application of variant 1 of the tabu search on the configuration obtained by MST followed by the local search;

3. Application of variant 2 of the tabu search on the configuration obtained by MST followed by the local search;

The point of comparison for each test is the configuration obtained by MST as described in chapter 3. The tests is applied to each of the networks described in section 3.3, for the different scenarios described in the same section.

4.3.1 Choice of LS parameters

The local search in the form of branch exchange proposed in the previous section has two parameters: the improvement threshold ε and the maximum number of iterations k_{max}.

First, let us focus on ε. This threshold is used to determine when an answer is better. For example, if we define $\varepsilon = 10$, this means that we accept as best solutions only those that have at least 10 kW less losses than the current solution. In order to choose the value of this parameter, we choose arbitrarily scenario 3 of the IEEE 33 network. Several values ranging from 10^{-5} to 10^{3} were tested on this scenario. For each value of ε, we represent the losses of the final solution obtained by local search, as well as the number of iterations required to obtain it. Curves showing the evolution of these two values as a function of ε are shown in Figure 4.2.

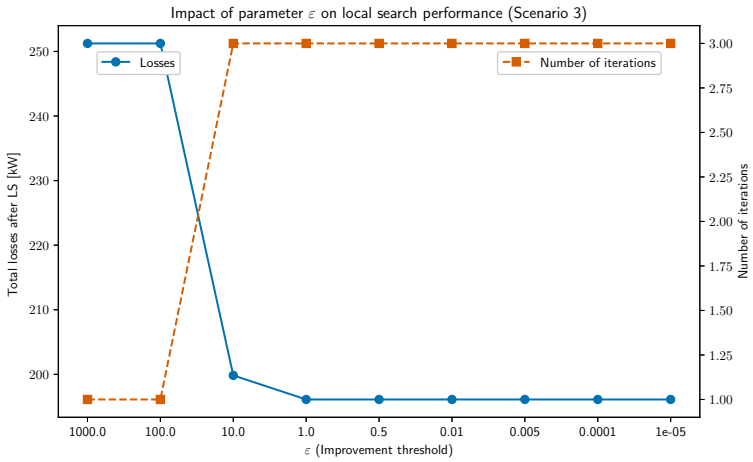
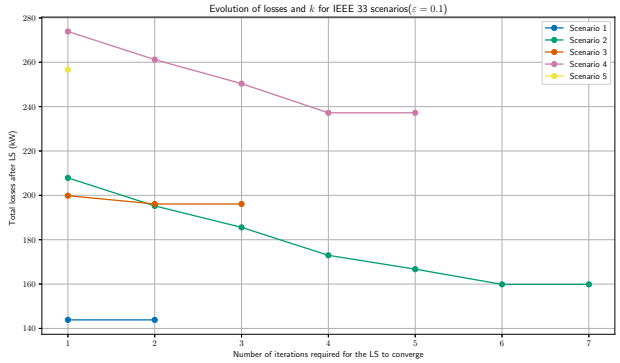


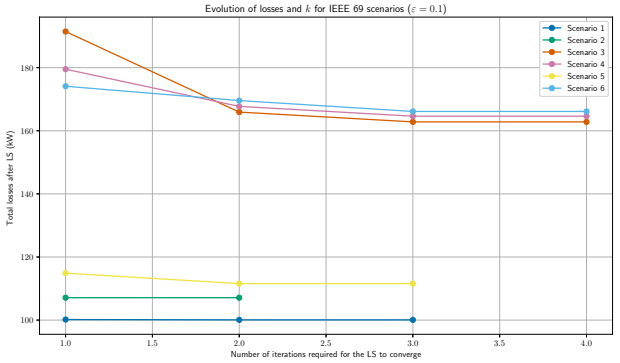
Figure 4.2: ε effect on LS

We can see from the figure that for $\varepsilon \geq 100$ the search stops after just one iteration. It finds no improvement, which seems rather logical given the large value of ε. When this parameter lies between 10 and 1, we notice a drop in losses and a greater number of iterations, up to 1. Finally, when $\varepsilon < 1$ the number of iterations remains constant and losses no longer decrease. This figure was created to show that the choice of parameters has a significant influence on the result. For our purposes, we keep $\varepsilon = 10^{-1}$, which means we accept a configuration if it decreases the current response by 0.1 kW.

In order to set the parameter k_{max}, we plot the number of iterations required to ensure convergence of the local search for each of the IEEE 33 and IEEE 69 network scenarios. The different results are shown in Figure 4.3. It can be seen that for the IEEE 33 network scenarios, the number of iterations required for the algorithm to converge ranges from 1 to 7, while for the IEEE 69 network it is between 2 and 4.



(a) Losses and number of iterations for LS - IEEE 33



(b) Losses and number of iterations for LS - IEEE 69

Figure 4.3: Losses and number of iterations for LS

All results taken into account, the choices for the parameters of the local search algorithm by branch exchange are fixed at : $\varepsilon = 10^{-1}$ kW and $k_{max} = 30$. The value of k_{max} is set much higher than what is actually needed in the scenarios we are studying, but this is a safety measure since the algorithm must be able to adapt to other scenarios, and even other networks.

4.3.2 Choice of TS parameters

The tabu search algorithm uses the following parameters:

- The improvement threshold ε;
- The length of the Tabu list L;
- The maximum number of iterations k_{max};
- The number of iterations without improvement before stopping $patience$.

By symmetry with the local search, ε is set at 10^{-1} kW.

First, let us focus on the maximum number of iterations. To do this, we set $patience = 9999$, which cancels out any influence of the $patience$ parameter on the result. First, we set the list size to 8. This value is based on the literature. In the example presented on [60] the list length is 5 and in the paper [61] the same value is 10. 8 therefore seems a reasonable choice, which is checked later. In order to determine the maximum number of iterations, we plot in Figure 4.4 for three scenarios relating to the IEEE 69 network the number of iterations required before convergence is reached, i.e. before reaching a quantity of losses in kW that no longer changes with iterations. We can see that for the three scenarios proposed, a maximum of 2 iterations are required. As with the local search, we set k_{max} to 20.

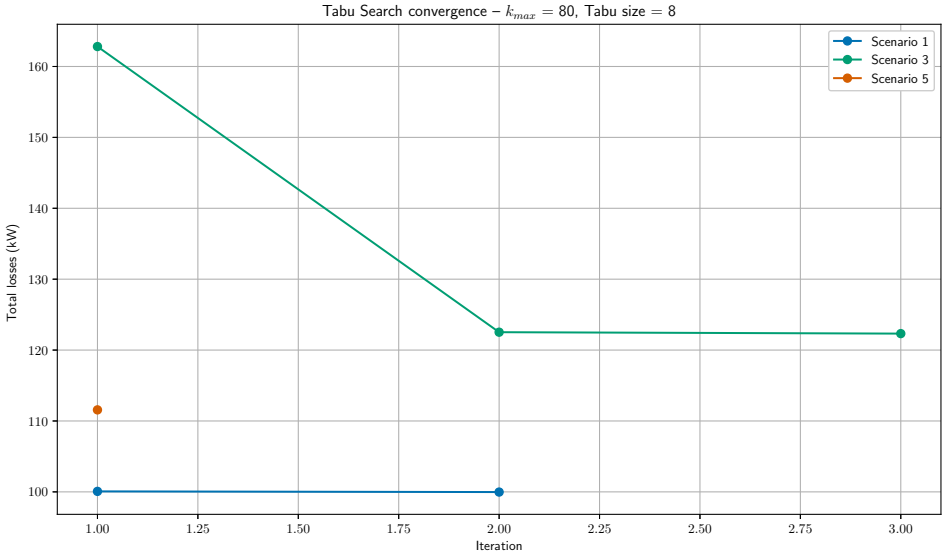


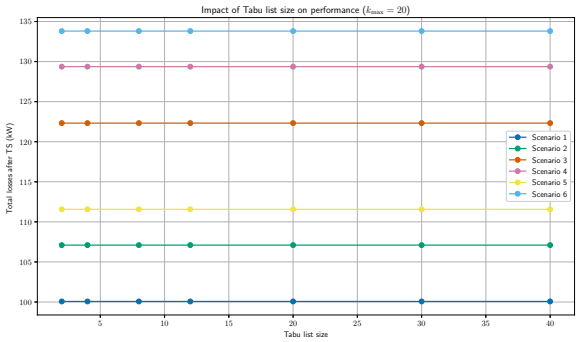
Figure 4.4: Choice of k_{max} for TS

Now that we have decided on the maximum number of iterations, we move on to choosing the size L of the tabu list. To do this, for all scenarios on the IEEE 69 network (an arbitrarily chosen network), we plot the losses after tabu search as a function of different tabu list sizes. The value of *patience* remains fixed at a very large value, 9999. The scenario chosen is scenario 3 of the IEEE 69 network, and the results are shown in Figure 4.5a. We note that the tabu list length, in the case of our optimization, has almost no influence on the response. So we leave L fixed at 8.

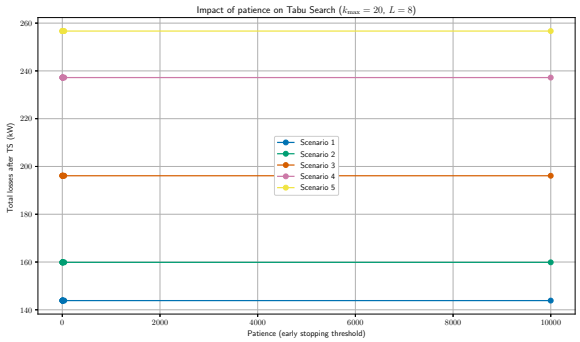
Finally, we decide on the value of the *patience* parameter. This time, we take the IEEE 33 network and try out different *patience* values, with L and k_{max} fixed at the values determined previously. The representation of this test is shown in Figure 4.5b. We can see that, as with the list size, the *patience* value has almost no influence in our study. For this reason, the value is simply set to 10.

.

All results taken into account, the parameter choices for the tabu search algorithm is be set to : $varepsilon = 10^{-1}$, $k_{max} = 20$, $L = 8$ and $patience = 10$.



(a) Choice of Tabu list size L for TS



(b) Choice of *patience* for TS

4.4 Results and discussion

The aim of this section is to present the results obtained using the algorithms presented above. As a reminder, three tests are carried out and compared with a reference response, which is the configuration

obtained by MST as presented in chapter 3. The tests are carried out on the IEEE 33 and IEEE 69 test networks used throughout this work. For each test, and each test scenario, we evaluate the losses and voltage profile to determine the qualities of the different configurations obtained, with the aim of evaluating the performance of our algorithms in the context of power network reconfiguration.

The results are presented in tabular form for both networks. The table is organized as follows:

- Column 1: Scenarios for the network under study;
- Column 2: Tests carried out;
- Column 3: Losses in kW for the configurations obtained from the tests carried out;
- Column 4: The percentage increase or decrease in losses for the three methods compared with a reference solution;
- Column 5: Minimum voltage in pu;
- Column 6: Maximum voltage in pu;
- Column 7: Standard deviation of voltage in pu.

4.4.1 Results on the IEEE 33 network

The results for the IEEE 33 network are shown in Table 4.2. Figures 4.6 and 4.7 support these results, and respectively represent losses in kW for each scenario and each method, and standard deviation of voltage in pu for each scenario and each method.

When we focus on the table, we first see that almost all the time, whatever the method used, there is a reduction in losses. This is trivial, since the proposed algorithms do not accept a worse configuration than the one initially proposed. In scenario 1, for the test involving the second version of the tabu test, we can observe a very slight, if not negligible, increase in losses. In scenario 5, the various heuristics bring little or no change to the configuration already obtained by MST-PF. For scenarios 2 to 4, we can see that the percentage reduction in losses varies between 13% and 38%, which is a rather positive result. The question now is whether one method stands out from the rest when it comes to loss reduction. Looking at Figure 4.6, we see that local search in the form of branch exchange significantly improves three cases out of 5 and does not degrade the other two. The first variant of the tabu search performs relatively similarly to the local search. The second version of the tabu search, on the other hand, is not stable. We can see that sometimes it leads to a configuration with fewer losses than version one (as in scenarios 2 and 6) and other times it performs less well (as in scenarios 3 and 4).

Regarding voltages, it can first be seen from Table 4.2 that for scenario 1, no configuration ensures a decent minimum voltage. Indeed, this result is quite consistent with the results obtained in the previous chapter and represented in the Table 3.5. In the following scenarios, the minimum voltages are always above the acceptable threshold. What can be noted is that, except in the case of scenario 5, all the other configurations obtained by local search or tabu search slightly degrade the voltage compared with the configuration obtained by MST. The second version of tabu search often presents even lower minimum voltages, albeit within limits. In fact, the algorithm allows voltage degradation as long as it doesn not fall below the 0.95 threshold. The minimum voltage drop alone cannot confirm that there is a degradation in the voltage profile, but the evolution of the standard deviation of voltages, shown in the last column and represented in Figure 4.7 does. Indeed, we can see that for the majority of scenarios, with the exception of the first, the standard deviation of voltages is quite low for the configuration obtained by MST-PF. It is slightly higher for the configurations obtained by local search and by local search followed by the first tabu

search variant, and reaches its maximum for all scenarios when applying the second tabu search variant. In the paper [51] one of the assumptions made is that any improvement in losses (i.e. decrease in losses) is coupled with an improvement in the voltage profile. However, we see here that this assumption does not apply to any of the cases.

Table 4.2: Comparison of performances by scenario and method - IEEE 33

Scenario	**Test**	**Losses (kW)**	**% losses change vs. MST-PF**	**Vmin (pu)**	**Vmax (pu)**	**STD voltage**
S1	MST-PF	144,9	–	0,9367	1	0,0173
	MST + LS	143,9	-0,69	0,9331	1	0,0168
	MST + LS + TS (V1)	143,9	-0,69	0,9331	1	0,0168
	MST + LS + TS (V2)	145,9	0,69	0,9367	1	0,0177
S2	MST-PF	249,7	–	0,9763	1	0,0082
	MST + LS	159,9	-35,96	0,9626	1	0,0112
	MST + LS + TS (V1)	159,9	-35,96	0,9626	1	0,0112
	MST + LS + TS (V2)	153,1	-38,69	0,9596	1	0,0131
S3	MST-PF	251,2	–	0,9899	1	0,0028
	MST + LS	196,1	-21,93	0,9649	1	0,0104
	MST + LS + TS (V1)	196,1	-21,93	0,9649	1	0,0104
	MST + LS + TS (V2)	217,6	-13,38	0,9653	1	0,0116
S4	MST-PF	313,3	–	0,9899	1	0,003
	MST + LS	237,2	-24,29	0,9691	1	0,0096
	MST + LS + TS (V1)	237,2	-24,29	0,9691	1	0,0096
	MST + LS + TS (V2)	244	-22,12	0,9565	1	0,0161
S5	MST-PF	256,7	–	0,9959	1	0,0015
	MST + LS	256,7	0	0,9959	1	0,0015
	MST + LS + TS (V1)	256,7	0	0,9959	1	0,0015
	MST + LS + TS (V2)	249,9	-2,65	0,9924	1	0,0023

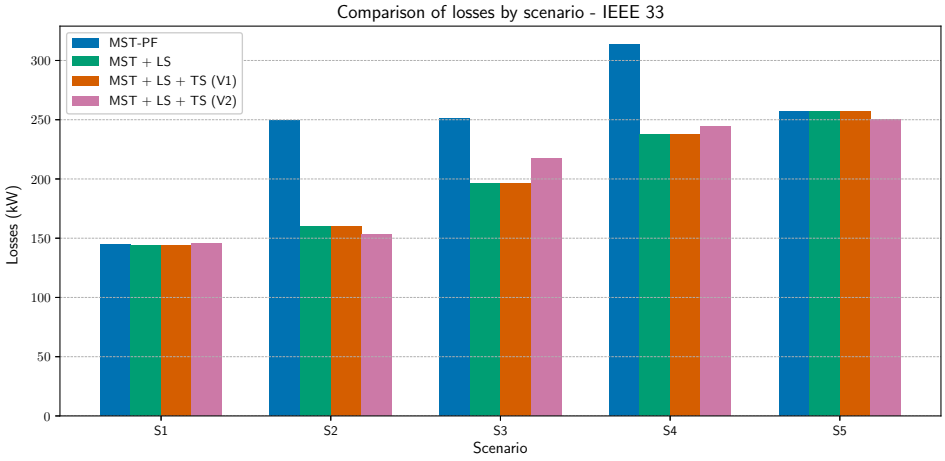


Figure 4.6: Losses (kW) per scenario and per test - IEEE33

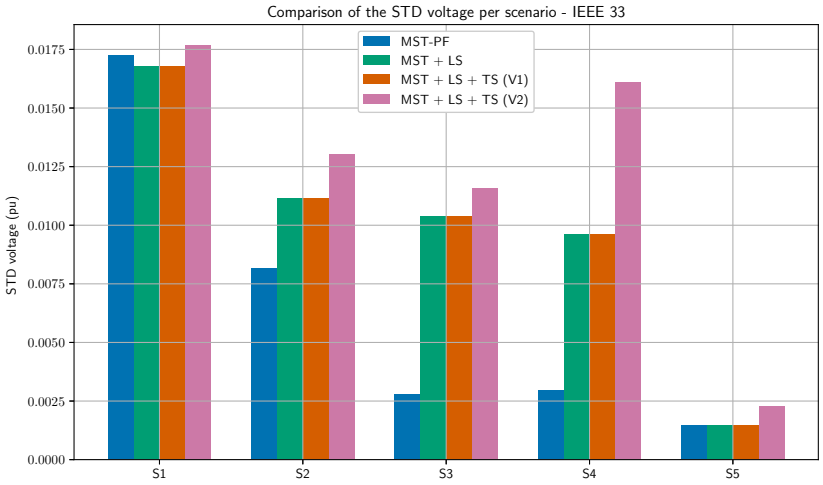


Figure 4.7: STD voltage (pu) per scenario and per test - IEEE33

The first conclusion that can be made is that the second variant of tabu search is the one that performsthe worst. It does not ensure a reduction of losses but in addition to this deteriorates in all cases the profile of voltage. The second remark that can be made is that in the case of this network, the tabu search and the local search by branch exchange perform relatively in the same way and therefore, in the specific case of this network, the Tabu search does not provide additional information or a better configuration.

4.4.2 Results on the IEEE 69 network

The results for the IEEE 69 network can be found in Table 4.3. Figures 4.8 and 4.9 support these results and represent respectively the kW losses for each of the scenarios and each of the methods and the standard deviation of the voltage in pu for each of the scenarios and each of the methods.

First of all, as for the IEEE 33 network, the first thing that we observe, in a trivial way again, is that almost all the configurations obtained through heuristics allow to reduce losses. These decreases range

from -0.28% for scenario 2 to -62.37% for scenario 5. The only configuration that does not decrease losses, but which increases them in an almost negligible way is the second version of the tabu search applied to scenario 2. This is most likely a precision error. When we look at Figure 4.8, we can see in a natural way the behaviour that we have just described. Unlike the IEEE 33 network where the first version of tabu search performed as well as local search, we see in this case that for several scenarios such as 3, 4 and 6, the tabu search brings an additional not negligible decrease in losses. Version 1 and 2 of the tabu search perform similarly in terms of losses.

Regarding the voltage profile, we can first notice in Table 4.3 that for three scenarios (S1, S2 and S5), the minimum voltages do not reach the minimum threshold of 0.95 (even if they are not far away). These results are in line with those presented in Table 3.6 of the previous chapter, where the voltage had already been improved thanks to the MST-PF method. Unlike the IEEE 33 network, the integration of an additional search does not degrade the voltages in the case of the IEEE 69 network, at least not always. In scenarios 1, 2 and 5 the minimum voltage results remain relatively the same. For other scenarios, local search can always improve the minimum voltage. However, tabu search degrades it very slightly at times. The maximum voltages are all below the maximum value of 1.05 pu and the proposed heuristics slightly decrease it compared to the value obtained by MST-PF. If we look at Figure 4.9, we can see that the search that performs best in the standard deviation of the voltage is the local search. The tabu search, whether for variant one or two, does not ensure in the case of the IEEE 33 network a systematic decrease in the standard deviation. However, except for the test of variant 2 on scenario 5, where the value of the standard deviation is greater than that of the MST-PF network, the values remain relatively similar in the case of MST-PF.

What can be said about the IEEE 69 network is that the integration of a heuristic as a means to refine the response obtained by MST is a good track. The majority of configurations offer a significant loss reduction. The Tabu search performs better in terms of losses but the local search seems to offer a good improvement in the voltage profile.

Table 4.3: Comparison of performances by scenario and method - IEEE 69

Scenario	Test	Losses (kW)	% losses change vs. MST-PF	Vmin (pu)	Vmax (pu)	STD voltage
S1	MST-PF	106,5	–	0,9427	1	0,0157
	MST + LS	100,1	-6	0,9428	1	0,0154
	MST + LS + TS (V1)	100,1	-6	0,9428	1	0,0154
	MST + LS + TS (V2)	100	-6,1	0,9428	1	0,0149
S2	MST-PF	107,4	–	0,9428	1,0017	0,0138
	MST + LS	107,1	-0,28	0,9428	1,0017	0,0138
	MST + LS + TS (V1)	107,1	-0,28	0,9428	1,0017	0,0138
	MST + LS + TS (V2)	107,6	0,19	0,9428	1,0017	0,0109
S3	MST-PF	290,2	–	0,9672	1,0057	0,0116
	MST + LS	162,8	-43,90	0,9887	1,0043	0,0042
	MST + LS + TS (V1)	122,3	-57,86	0,9624	1,0031	0,0117
	MST + LS + TS (V2)	122,3	-57,86	0,9624	1,0031	0,0117
S4	MST-PF	299,7	–	0,9655	1,0057	0,0126
	MST + LS	164,6	-45,08	0,9903	1,0043	0,0034
	MST + LS + TS (V1)	129,4	-56,82	0,9869	1,0031	0,0044
	MST + LS + TS (V2)	129,4	-56,82	0,9869	1,0031	0,0043
S5	MST-PF	296,6	–	0,9436	1,0017	0,0122
	MST + LS	111,6	-62,37	0,9428	1,0003	0,0117
	MST + LS + TS (V1)	111,6	-62,37	0,9428	1,0003	0,0117
	MST + LS + TS (V2)	111,6	-62,37	0,9428	1,0003	0,0136
S6	MST-PF	243,8	–	0,9784	1,0051	0,0071
	MST + LS	166,1	-35,87	0,9953	1,0042	0,002
	MST + LS + TS (V1)	133,8	-45,11	0,9777	1,0031	0,0063
	MST + LS + TS (V2)	133,8	-45,11	0,9777	1,0031	0,0061

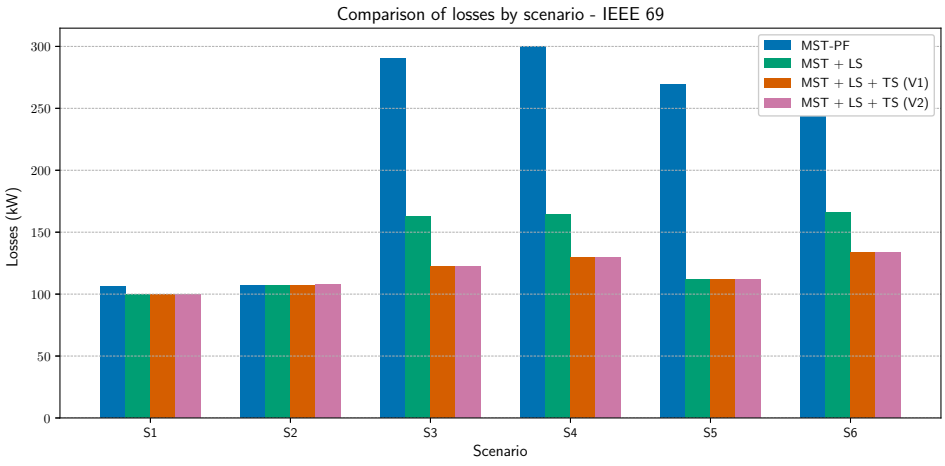


Figure 4.8: Losses (kW) per scenario and per test - IEEE69

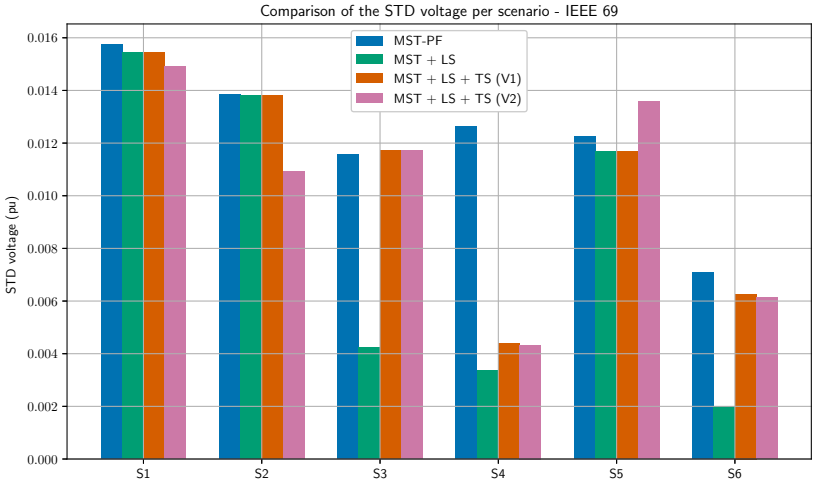


Figure 4.9: STD voltage (pu) per scenario and per test - IEEE69

4.5 Conclusion of the chapter

The different tests carried out in this chapter and the different results presented allowed us to evaluate the effectiveness of the use of heuristics in the context of the reconfiguration of distribution networks. Indeed, in most cases they provide a better answer than that proposed by the minimum spanning tree method detailed in the first chapter. In the worst case and if no improvement is found, the algorithms keep a response close to the initial.

The results are not the same on IEEE 33 and IEEE 69 networks. On the IEEE 33 network, even if the methods allow a reduction of losses, it is possible to notice that a degradation of the voltage profile is often present. The local search seems to offer in the case of IEEE 33 a good compromise between voltage profile and losses, and the tabu search does not often bring improvements compared to the local search. On the other hand, in the IEEE 69 network, the two heuristics reduce losses significantly, sometimes they decrease by 60% compared to the configuration already obtained by MST-PF, and this is often accompanied by an improvement of the voltage profile. In the case of the IEEE 69 network, the tabu search performs better than the local search in several cases, whether it is variant 1 or variant 2, and this without penalizing the voltage profile.

Thanks to all these results, it is possible to state that the heuristic methods make it possible to improve in several cases the answers obtained by MST-PF in the context of reconfiguration. However, a final dimension that has not yet been mentioned is computational time. Even if the configurations are satisfactory, the resolution in a reasonable time remains important. To do this, two figures representing the calculation times per network and per scenario are available: 4.10 and 4.11.

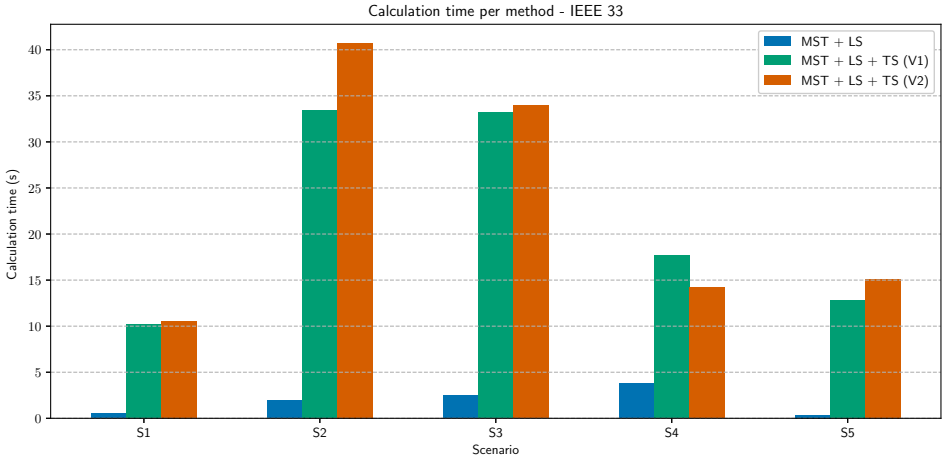


Figure 4.10: Computational time (s) per method - IEEE 33

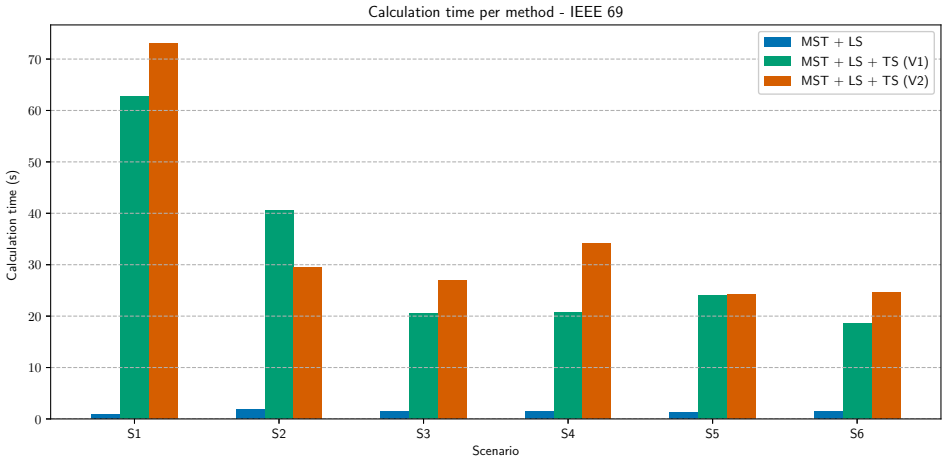


Figure 4.11: Computational time (s) per method - IEEE 69

We can easily notice that for all scenarios the refinement of a configuration by local search is much faster than by tabu search. Branch exchange local search often takes less than one second while tabu search varies between 10 and 35 seconds for IEEE 33 network and 50 to 80 seconds for IEEE 69 network. The second variant is often more expensive from a computational point of view, but both tabus have long computation times.

The calculation time is our next study factor. Indeed, the purpose of reconfiguring distribution networks is not only to get a proper configuration, but to do it in a reasonable time as well. To address this point, a machine learning approach is proposed in the following chapter.

Chapter 5

Machine learning for automating heuristic refinement

5.1 Introduction

In previous chapters, we explored different methods for the problem of distribution network reconfiguration. We started with the minimum spanning tree (MST) and had the opportunity during Chapter 4 to refine the answer obtained by MST with heuristics such as local search (LS) or tabu search (TS). These methods provide relevant configurations and approach the problem in a consistent way. However, as we see in the conclusion of the previous chapter, computation times are not negligible, especially in the case of tabu search. What is more, the computation time increases with the size of the network, the number of tie-lines and the number of distributed generators (DGs). In a planning context, the time constraint is somewhat less important. However, in an operational context, decisions sometimes need to be taken quickly, and fast algorithms are required.

The main aim of this chapter is to challenge the limits imposed by the heuristics presented, using machine learning (ML). The objective is to create a model that can quickly predict a target configuration of a network as a function of its electrical state which is defined in part by load profile and on service distributed generators. The model is trained using the results obtained from the methods studied in the previous chapter: MST combined with local search.

Beyond reducing computation time, it is important that the model has effective generalization capability. Generalization capability refers to its ability to predict the target configuration of scenarios that it has never encountered during training.

The idea of this chapter is first to present the model used and second discuss its performance in comparison with the methods that are presented in previous chapters. In the conclusion of the previous chapter, we see that the tabu method do not consistently provide an improvement over local search. For this reason, and for reasons of computation time, we continue this chapter using only local search.

5.2 Formulation of the problem

Consider an electrical distribution network represented by a graph $G = (N, L)$, where:

- N denotes the set of n nodes (or buses): $\#N = n$;
- L denotes the set of m lines: $\#L = m$.

Each network scenario can be represented by an overall electrical state S that includes:

- The active and reactive loads of the buses: P_i and Q_i for all $i \in N$.
- The state of the generators. Let
$$\mathcal{G} = \{G1, G2, \ldots, Gk\}$$
be the set of distributed generators with $\#\mathcal{G} = k$. Each scenario is represented by a binary vector according to the status (in service/off service) of the generators: $\mathbf{g} = (g_1, g_2, ...g_k) \in \{0,1\}^k$ with:
$$g_j = \begin{cases} 1 & \text{if generator } G_j \text{ is activated in the scenario,} \\ 0 & \text{otherwise.} \end{cases} \qquad \text{for } j \in \{1, \ldots, k\}.$$

The electrical network has two specific states:

- The initial state is the state in which the network is meshed, i.e., all lines are closed. It is named S_{mesh}.
- The second state corresponds to the target state, i.e., the solution provided by the MST-LS algorithm presented in the previous chapter. It is named S_{target}

For each line $l \in L$, we define a binary variable $y_l \in \{0,1\}$ which indicates whether a line is open (0) or closed (1) in the target configuration:

$$y_l = \begin{cases} 1 & \text{if line } l \text{ is in service in } S_{\text{target}}, \\ 0 & \text{if it is out of service.} \end{cases} \qquad \text{for } l \in \{1, \ldots, m\}.$$

Each line l is associated with a feature vector $X_l \in \mathbb{R}^d$. This vector is calculated from the initial state, i.e., the state where the network is meshed S_{mesh} for a specific scenario (characterized by its loads and the combination of DGs in service). There are topological, static, and power flow features. They are described in detail in section 5.4.4.

All lines are thus represented by a matrix of features:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \in \mathbb{R}^{m \times d} \quad \text{and a vector of labels} \quad \mathbf{y} = (y_1, y_2, \ldots, y_m) \in \{0,1\}^m.$$

We can thus state the learning problem as learning a decision function that approximates:

$$\mathbb{P}(y_l = 1 \mid X_l), \quad \forall l \in \{1, \ldots, m\},$$

i.e., estimating, for each line, the probability that it is part of the target configuration, given its local characteristics.

Hypothesis of conditional independence of lines

In an electrical network, the state of one line cannot be independent of the state of the other lines on the network. However, in order to make the learning problem computationally feasible, let us consider the following hypothesis:

It is assumed that, once line l's features are known, whether a line is activated or not, does not depend on any other lines in the network.

What this means is that rather than modeling all lines together, we have characterized each line independently of other lines in the network, to predict the state of each line based entirely on its own features.

$$\mathbb{P}(y_1, y_2, \ldots, y_m \mid X_1, X_2, \ldots, X_m) \approx \prod_{l=1}^{m} \mathbb{P}(y_l \mid X_l).$$

If we do not make this assumption, we would have to approximate the joint distribution $P(\mathbf{y} \mid X)$ over the 2^m possible configurations of m lines. For a network with $m = 37$ lines (as in IEEE-33), this represents $2^{37} = 1.37 \times 10^{11}$ distinct combinations. Furthermore, if we describe each line with $d = 17$ characteristics and we want to train a model on 10 000 scenarios, we expect an input space of size $10\,000 \times 37 \times 17 = 6.29 \times 10^6$ values. Such computations are, not achievable on any standard machine.

This assumption allows us to simplify the initial problem into m independent binary classification problems, each learning to predict:

$$\mathbb{P}(y_l = 1 \mid X_l), \quad \text{independently for each } l \in \{1, \ldots, m\}.$$

These classifiers can be trained independently, with models such as XGBoost. Then, an algorithm calculating the minimum spanning tree is used to reconstruct a radial network.

Final formulation of the problem

Under the explained assumption, and after concatenating the N simulated scenarios, the learning problem can be rewritten as a standard binary classification problem:

$$\begin{aligned} &\text{Data:} \quad X_{\text{total}} \in \mathbb{R}^{N \cdot m \times d}, \quad \mathbf{y}_{\text{total}} \in \{0,1\}^{N \cdot m}, \\ &\text{Objective:} \quad \text{learn a function } f_\theta : \mathbb{R}^d \to [0,1] \text{ such that } f_\theta(X_l) \approx \mathbb{P}(y_l = 1 \mid X_l). \end{aligned}$$

Each row of the dataset corresponds to an observation (X_l, y_l) from a given scenario, and is treated as an independent instance of the problem.

In summary, the proposed classification model learns to reproduce the decisions made by the MST + LS heuristic based on the characteristics of each row. The probability found is then used as a weight in an MST algorithm to ensure that the final solution is radial.

5.3 Presentation of the ML model

In this study, the machine learning algorithm used is XGBoost (eXtreme Gradient Boosting). XGBoost is an improved version of the gradient boosting algorithm. Gradient boosting decision trees are an ensemble method for combining several shallow trees to form a more efficient model. The trees are generated sequentially, one after the other, and each tree learns to correct the errors of the previous tree. To do this, it trains on the gradients of the loss function. This iterative process of minimizing a cost function reduces the bias of the model step by step. The algorithm is widely used and is well suited to data structured in tabular form. Indeed, data used in this study is tabularly structured, since each line is described by electrical properties [62] [63].

In addition to being capable of processing data in the format of tables, XGBoost is known to be capable of processing unbalanced datasets. This can be used in the cases examined since the dataset is in fact unbalanced: there are always significantly more active lines, i.e. lines belonging to the operational configuration, than tie-lines, i.e. lines that remain open. This creates an imbalance in classes.

Finally, the use of the XGBoost algorithm is justified, apart from already being used in the literature currently, by the ease of use because of implementations already proposed in Python with a scikit-learn-compatible API.

5.4 Building of the dataset

In order to train the XGBoost model efficiently, databases are artificially generated from a simulation of scenarios. The purpose of this section is to describe the process of generating the dataset used.

5.4.1 Generating a scenario

An electrical scenario corresponds to a particular state of the network. As mentioned in section 5.2, a scenario is defined by its active and reactive loads and the state of each generator. The steps for constructing a scenario are as follows:

1. The starting point is the IEEE-33 or IEEE-69 meshed network (S_{mesh}), i.e., with all lines closed.
2. The active and reactive loads of the buses ($P_i, Q_i \ \forall i \in N$) are modified by applying a random multiplicative factor between 0.6 and 1.3.
3. A combination of distributed generators (DGs) is activated, according to a different logic depending on the experiment considered (see Section 5.4.3). The binary vector $\mathbf{g} \in \{0,1\}^k$ indicates the active generators.
4. The heuristic method (Local Search) is applied to obtain a radial target solution S_{target}.
5. For each line $l \in L$, a feature vector X_l is extracted from the meshed state of the scenario. A label y_l is defined depending on whether or not the line belongs to S_{target}.

Each scenario therefore provides a feature matrix $X^{(i)} \in \mathbb{R}^{m \times d}$ and a label vector $y^{(i)} \in \{0,1\}^m$.

5.4.2 Dataset construction

The generated scenarios are then concatenated to form a single tabular dataset:

$$X_{\text{total}} = \begin{bmatrix} X^{(1)} \\ X^{(2)} \\ \vdots \\ X^{(N)} \end{bmatrix} \in \mathbb{R}^{N \cdot m \times d}, \quad \mathbf{y}_{\text{total}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} \in \{0,1\}^{N \cdot m},$$

where N is the number of scenarios and m is the number of lines in the network.

A simplified illustration is given below:

Table 5.1: Training dataset structure

Line	f_1	f_2	...	f_d	y
1	0.62	0.13	...	0.25	1
2	0.07	0.85	...	0.78	0
⋮	⋮	⋮		⋮	⋮
$N \cdot m$	0.12	0.67	...	0.42	1

5.4.3 Experiments

Two experiments are conducted in this chapter to evaluate the model's ability to generalize to unknown generator and load configurations.

- **Experiment 1: Learning with all generators (all-generator)**
  In this experiment, scenarios are generated from all possible combinations of generators in the network. For example, for the IEEE-33 network, the 8 generators G_1 to G_8 can be freely combined.
- **Experiment 2: Partial learning (partial-generator)**
  For this experiment, only certain generators are used to create the training scenarios. The others are deliberately excluded to test the model's ability to generalize to configurations it has never seen before. For example, for the IEEE 33 network, generators $G7$ and $G8$ are not included in any of the scenarios dedicated to training.

As a reminder, the generators defined for IEEE 33 and IEEE 69 networks are presented in Tables 3.1 and 3.3.

Table 5.2: Presentation of the two experiments

Experiment	**Number of combinations**	**DG excluded from training**	**Validation objective**
All-generator training	Random (0-8 among G1-G8) for IEEE 33, Random (0-3 among G1-G3) for IEEE 69	No DG	Evaluate interpolation performance
Partial-generator training	Random (0-6 among G1-G6) for IEEE 33, Random (0-2 among G1-G2)	G7 and G8 for IEEE 33 and G3 for IEEE 69	Evaluate the generalization capacity on generators never seen before

5.4.4 Feature extraction

In order to train the presented XGBoost model effectively, it needs to be provided with relevant information. In this study, each line is described using a feature vector. Three types of features are selected: static features, features obtained by calculating power flow, and topological features.

Static features represent the intrinsic properties of each line. These are the initial features describing a line: resistance per km of line, reactance per km of line, and the widely-used R/X ratio (resistance/reactance). It should be noted that when modeling the test networks, all line lengths were fixed at 1 km. This implies that line length, which could form part of the static features, are not used in this study, as it is irrelevant since it is constant over all lines.

The features obtained by power flow are those determining the electrical state of a line in a given scenario, and are highly dependent on the load profile and active DGs. For the purposes of this study three features of this type are used: the current flowing through the line in kA, the active power injected or consumed at the starting point of the line in kW, and the reactive power injected or consumed at the starting point of the line in kVAr.

Finally, topological features are considered. These indicate where a line is located in the network, and are computed from an analysis of graph structure. From both ends u and v of every line, starting node degree and ending node degree, i.e. number of branches on a node are calculated, and slack distance to both line ends (in terms of number of branches).

The features are then concatenated to form an input matrix X for each scenario.

Table 5.3: Summary of features used for each line

Type of feature	**Description**
Static	• Resistance per km R (ohm/km) • Reactance per km X (ohm/km) • Ratio R/X
Electric (by computing power flow)	• Current intensity I circulating in the line (kA) • Absolute value of active power injected at starting point P (MW) • Absolute value of reactive power injected at starting point Q (MVAr) • Voltage magnitude at start node $\|V_{from}\|$ (p.u.) • Voltage magnitude at end node $\|V_{to}\|$ (p.u.) • Voltage magnitude drop $\|\|V_{from}\| - \|V_{to}\|\|$ (p.u.) • Phase angle difference $\|\angle V_{from} - \angle V_{to}\|$ (degrees)
Topological	• Degree of start node deg(u) and end node deg(v) • Distance from u and v to the slack node (in number of branches) • Betweenness centrality of start node (bus from u). This feature corresponds to the number of shortest paths in the graph that pass through u. • Betweenness centrality of end node (bus to v). This feature corresponds to the number of shortest paths in the graph that pass through v. • Sum of betweenness centrality of both nodes. A large sum indicates that the line connects two important nodes in the network. • Absolute difference of betweenness centrality. A large difference indicates that the line connects a central node to a less central node.

5.5 Evaluation metrics

Before moving on to the explanation of the training process and the testing process, a section is devoted to explaining the different metrics used in this study. Three metrics are presented: accuracy, F1-score,

and AUC-PR (Area Under the Precision-Recall Curve) [64].

5.5.0.1 Accuracy

Accuracy is a simple measure that determines the proportion of correct predictions made by the model across all lines in the network. It can be formulated as follows:

$$\texttt{accuracy} = \frac{\texttt{Total number of correct predictions}}{\texttt{Total number of lines}}$$

Let us take the example of a network with 100 lines. If we have a model that can correctly classify 90 lines, the accuracy is 90%. This is an easy and practical measure to use, but it is not effective when the data is unbalanced. For example, if for this same network of 100 lines we know that 90 lines must be closed and we create a model that predicts all closed lines, it has an accuracy of 90%. Accuracy is therefore not representative when classes are imbalanced.

5.5.0.2 F1-score

The F1 score is a metric that involves two other metrics: precision and recall.

Precision answers the following question: among the lines that the model predicted as belonging to class 1, how many actually belong to that class? It is therefore the proportion of true positives (TP) among all positive predictions, i.e., the sum of true positives and false positives (FP). This can be written as follows:

$$\texttt{Precision} = \frac{\texttt{TP}}{\texttt{TP + FP}}$$

Recall answers the following question: Among the lines that are actually active, i.e., belonging to class 1, how many does the model manage to detect? It is therefore the proportion of true positives among the total number of positives, i.e., the sum of true positives and false negatives. This can be written as follows:

$$\texttt{Recall} = \frac{\texttt{TP}}{\texttt{TP + FN}}$$

The F1-score combines precision and recall as follows:

$$\texttt{F1-score} = 2 \times \frac{\texttt{Precision} \times \texttt{Recall}}{\texttt{Precision} + \texttt{Recall}}$$

An example to help understand this is as follows: consider a network with 10 lines that are actually active and a model that predicts 12 active lines. Of these 12 active lines, 8 are actually active. The precision is therefore $8/12 = 0.67$ and the recall $8/10 = 0.8$. Consequently, the F1-score is:

$$\texttt{F1-score} = 2 \times \frac{0.67 \times 0.8}{0.67 + 0.8} = 0.73$$

The F1-score is a reliable metric when working with unbalanced classes. However, it requires the definition of a threshold (by default, this is 0.5). In the case of this study, there is no precise threshold for deciding whether a line is active or not. Instead, probabilities associated to each line are used as weights in an MST algorithm.

5.5.0.3 AUC-PR (Area Under the Precision-Recall Curve)

AUC-PR is a metric used to measure the classification ability of a model for a specific class, in this case, the closed lines (class 1). In our case, the minority class corresponds to open lines. As indicated by its name, it measures the area under the precision-recall curve.

The difference between this metric and the two presented above is that it is not necessary to set a threshold since it considers all possible thresholds. The AUC-PR is calculated as follows:

1. First, the model creates the set of predictions for belonging to class 1 (closed line) for each line in the network.
2. We sort the probabilities in descending order;
3. For each probability value (which serves as a threshold), the lines are divided as follows:
   - All lines with a probability greater than or equal to this threshold are considered to be predicted in class 1;
   - The others are considered to be predicted in class 0.
4. For each threshold, the predictions are compared to the true classes of the lines and the precision and recall are calculated.
5. The calculation is repeated for each threshold. A set of points (recall, precision) is obtained, which can be used to plot the precision-recall curve;
6. The area under the curve is calculated and this value is called AUC-PR.

Let us imagine a small network with 5 lines. The following table shows the probabilities of belonging to class 1 predicted by the model for each of the lines, as well as the true classes:

Table 5.4: Predicted probabilities and true classes - Example

Ligne	**Predicted probability (class 1)**	**True class**
1	0.9	1
2	0.8	0
3	0.7	1
4	0.6	1
5	0.3	0

The AUC-PR can be calculated as follows:

1. Sort the lines by decreasing probability: 1,2,3,4,5
2. Calculate the precision and recall for each threshold (0.9,0.8,0.7,...)
   - For the threshold 0.9: only line 1 is predicted to be active: precision = $1/1 = 1$ and recall $1/3 = 0.33$;
   - For the threshold 0.8: lines 1 and 2 are predicted to be active: precision = $1/2 = 0.5$ and recall $1/3 = 0.33$;
   - For threshold 0.7: lines 1, 2, and 3 are predicted to be active: precision = $2/3 = 0.67$ and recall $2/3 = 0.67$;
   - For the threshold 0.6: lines 1, 2, 3, and 4 are predicted to be active: precision = $3/4 = 0.75$ and recall $3/3 = 1$;

- For the threshold 0.3: lines 1, 2, 3, 4, and 5 are predicted to be active: precision = $3/5 = 0.6$ and recall $3/3 = 1$;

3. The points (recall, precision) are then plotted on a graph and the area under the curve is calculated.

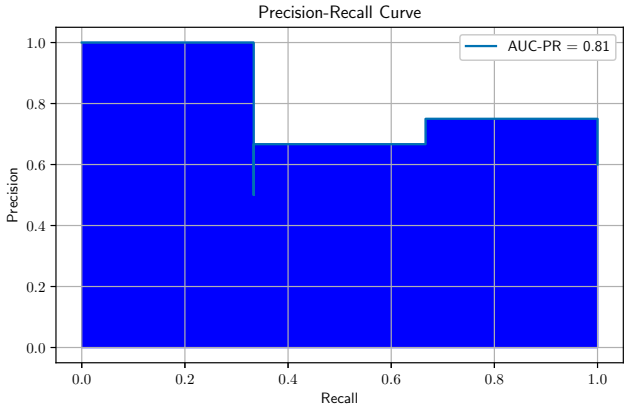


Figure 5.1: AUC-PR on an example

An area close to 1 means that the model is able to distinguish well between the minority class and the majority class. This metric is effective when dealing with unbalanced classes. It provides a more representative evaluation than accuracy, for example, and unlike the F1 score, it does not depend on a threshold.

5.6 Training process

The purpose of this section is to detail the training procedure for the XGBoost model. First, we initialize an XGBoost classifier. To do this, we must choose the values of the hyperparameters that the algorithm takes as input. First, we use the default values. These are the following hyperparameters:

- `n_estimators`: this is the number of decision trees that the model creates. The more there are, the more information the model can learn. However, the more there are, the higher the chance of overfitting. The default for XGBoost is 100;
- `learning_rate`: this is the hyperparameter that decides the step size of the gradient descent. By default, this value is set to 0.3 on XGBoost;
- `max_depth`: this is the maximum depth of each tree. The higher the depth of a tree, the more complexities it can capture. The default value is 6;
- `sub_sample` and `col_sample`: these are respectively the proportion of data (rows) and features (columns) considered for the trees construction. The default value is 1;
- The problem is formulated as a binary classification, which means that it must decide whether or not to keep a line. For the test networks used in this study (IEEE 33 and IEEE 69), the number of tie lines is small compared to the number of lines in operation. In fact, we have 5 open lines for 37 and 73 lines, respectively. The classes are therefore unbalanced, and to remedy this, we use the hyperparameter `scale_pos_weight`. This parameter allows us to give more weight to the minority class. It defaults to 1 but is set in our model to the following ratio:

$$\texttt{scale\_pos\_weight} = \frac{\#class0}{\#class1}$$

.

The default values provide an initial model. However, the default values are not necessarily the most optimal, so the goal is to determine the optimal values for the various hyperparameters. To do this, we used Bayesian optimization with the Optuna library [65]. The goal is to test different combinations of hyperparameters iteratively. Optuna chooses the combinations to test based on previous results, which allows us to avoid exhaustive testing. The ranges explored for the various hyperparameters mentioned are presented in Table 5.5.

Table 5.5: Range of hyperparameters

Hyperparameter	**Range**
`learning_rate`	0.01 – 0.4
`max_depth`	4 – 12
`subsample`	0.6 – 1.0
`colsample_bytree`	0.6 – 1.0
`n_estimators`	50 – 500

For each trial, Optuna randomly selects a combination that it adapts as it goes along based on the results observed. It works as follows:

1. First, we select randomly 20% of the scenarios from the training set;
2. For each combination of hyperparameters proposed by Optuna, a stratified cross-validation by scenario (StratifiedGroupKFold) is perfomed. This means that:
   - A scenario is either in the training set or in the validation set for the same fold;
   - The proportion of open and closed lines is preserved in each fold.
   - Cross-validation comprises 4 folds: at each iteration, 3 folds are used for training and 1 for validation.
3. For each fold, the model is trained on the scenarios in that fold and validated on the scenarios in the validation fold. At the end, the AUC-PR is calculated on the lines in the validation fold.
4. Once the 4 folds have been performed for the current Optuna combination, the average AUC-PR is calculated over the 4 folds.
5. The fixed number of iterations (and therefore hyperparameter combinations) is 150.
6. At the end of the 150 iterations, Optuna selects the hyperparameter combination with the best AUC-PR average over the 4 folds.

After the optimization step presented, the XGBoost model with the hyperparameters determined using Optuna is retrained. The results of the hyperparameters obtained for the IEEE 33 and IEEE 69 networks are shown in Table 5.6.

Table 5.6: Hyperparameters obtained with Optuna Bayesian optimization for IEEE 33 and IEEE 69 networks

Network	`learning_rate`	`max_depth`	`subsample`	`colsample_bytree`	`n_estimators`	**AUC-PR**
IEEE 33	0.023	12	0.66	0.68	469	0.9983
IEEE 69	0.035	12	0.74	0.95	403	0.9998

Table 5.7 shows the evaluation metric values before and after tuning. Tuning has almost no effect on the results. This can be explained by the fact that values before tuning are already high, the potential for gain is therefore limited, and it is possible that maximum performance is reached.

Table 5.7: Evaluation metrics before and after hyperparameter tuning

	IEEE 33			**IEEE 69**		
Tuning	**Accuracy**	**F1-score**	**AUC-PR**	**Accuracy**	**F1-score**	**AUC-PR**
Before tuning	0.927	0.957	0.995	0.956	0.976	0.996
After tuning	0.93	0.959	0.995	0.956	0.976	0.996

5.7 Testing process

We now discuss the testing procedure. The aim of testing is to evaluate the model and measure its ability to correctly predict optimal configurations for scenarios it has never learned during training. The two experiments presented in section 5.4.3 differ in their testing process.

Let us start with the first experiment, i.e. the one where the model is trained with all the generators. In order to verify that the model learns the logic of heuristic reconfiguration rather than just memorizing specific scenarios, the testing set is built with scenarios that are different from those used in training. Two elements determine a scenario: the combination of active generators and the random modification of loads. A scenario with a combination of active generators and a specific load profile cannot be found in both the training set and the test set. The model created using this approach is therefore tested on scenarios not present in the training set. However, no new generators are added in the test phase. The combinations are made from the same generators. This allows us to evaluate the model's performance in different scenarios but with known assets.

Let us move on to the second experiment. The difference with the first experiment is that for this second experiment, we add to the test set, generators that were never seen during training. This is done in order to assess the model's ability to generalize to scenarios with unknown generators. For example, for the IEEE 33 network, generators G7 and G8 are not used in the training set. However, they are included in the validation set.

5.8 Results and discussion

The aim of this section is to present the results obtained on IEEE 33 and IEEE 69 networks from the two experiments proposed in the previous sections of the chapter. To do this the performance of each experiment and an accurate analyse about the learning capabilities of the models, as well as the influence of features are discussed. The section is closed with a summary in 5.8.5.

5.8.1 General presentation

To start presenting the results, it is important to understand the two experiments that are presented in section 5.4.3.

5.8.2 Performance comparison for the two experiments

The aim of this first part is to present the values found for three metrics, for the IEEE 33 and IEEE 69 networks, each tested with the two proposed experiments. The metrics used to evaluate the model are the ones presented in section 5.5: accuracy, F1-score and AUC-PR. Table 5.8 shows the values of each metric for all experiment-network combinations.

Table 5.8: Performance comparison of the two experiments on IEEE 33 and IEEE 69 networks

Test network	Experiment	Accuracy	F1-score	AUC-PR
IEEE 33	All-generator training	0.933	0.960	0.998
IEEE 33	Partial-generator training	0.93	0.959	0.995
IEEE 69	All-generator training	0.977	0.988	1.000
IEEE 69	Partial-generator training	0.956	0.976	0.996

In the first place, we can see that the metric values obtained in the first experiment (all-generator) are always higher than those obtained in the second experiment (partial-generator). This observation applies to both test networks. This is to be expected, since partial-generator training adds the difficulty of testing the model on networks with new assets. However, all metric values are already high (no metric is below 0.93) and the increases are not significant. The fact that the metrics have high values in the case of partial-generator training, for both test networks, implies that the models created have a high generalization capacity and are therefore capable of predicting reconfiguration on scenarios with assets never seen before in a configuration close to that which could be provided by an MST-LS heuristic.

Focusing on the IEEE 69 network, it can be seen first of all that performance is slightly better than for the IEEE 33 network. This is not strange, since the IEEE 33 network has 8 generators, whereas the IEEE 69 network only has 3. As the number of combinations is much smaller, the model has the opportunity during training to see a greater percentage of possible scenarios than with the IEEE 33 network.

In conclusion, two things can be noted: first, the metrics are always above 93% regardless of the approach, which means that the models function correctly and make predictions as needed. Second, the XGBoost model's ability to generalize to cases it had never seen before is particularly noteworthy.

5.8.3 Learning capacity analysis

The purpose of this section is to evaluate the learning capacity of the models, that is, their ability to improve their performance as scenarios (data) are added. To this end, learning curves are presented for the two test networks and for the two experiments. Figure 5.2 shows the learning curves for the first experiment (all-generator training) and Figure 5.3 shows the learning curves for the second experiment (partial-generator training).

The learning curves were generated from the 10 000 scenarios created for each network and each experiment. Each point on the curve corresponds to a given size of subset of scenarios. For example, for a point at 3 000 scenarios, 30% of the scenarios are used. Then, a cross-validation is performed on these 3 000 scenarios. This means that the 3 000 scenarios are divided into five folds of 600 scenarios each. Four folds are used for training (80%) and the last fold is used for validation (20%). A rotation is performed between the folds to calculate the scores on the five 80/20 combinations. The average scores are calculated for the training score and the validation score. The metric used is AUC-PR. This process is performed for scenario sizes ranging from 1 000 to 10 000, gradually increasing the number of scenarios used by 1 000 each time. This allows us to see how AUC-PR evolves as a function of the number of

scenarios used.

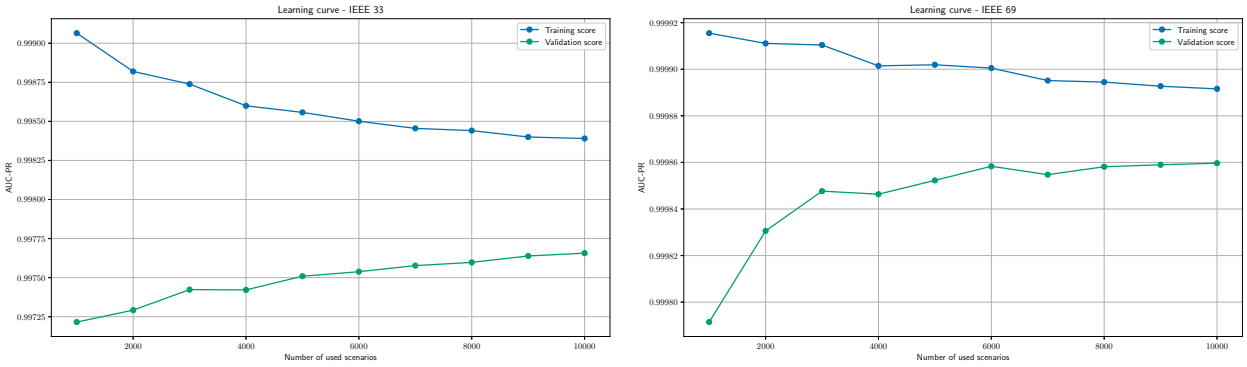


Figure 5.2: Learning curves - experiment 1 (all-generator training) - IEEE 33 and 69 networks

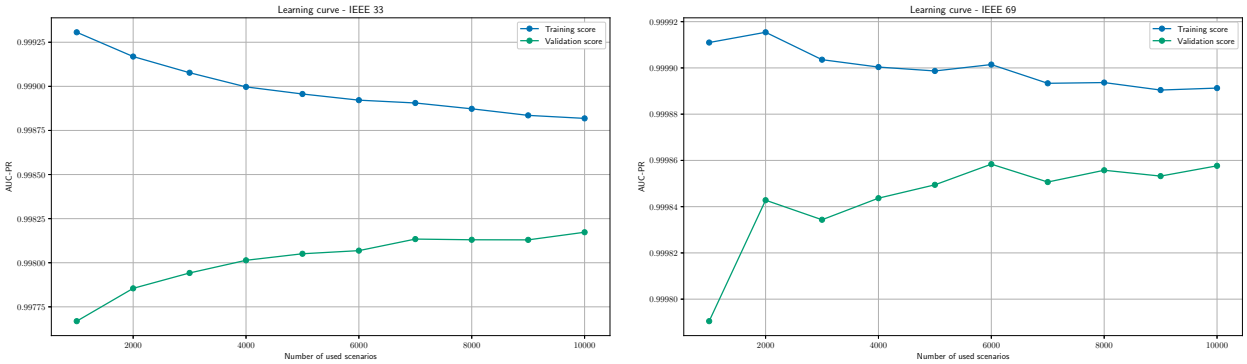


Figure 5.3: Learning curves - experiment 2 (partial-generator training) - IEEE 33 and 69 networks

The behavior of the learning curves is similar for the two test networks and for the two experiments. It can be observed that the training score curve starts quite high in all the cases presented and decreases as scenarios are added. The validation curve increases as scenarios are added. In all four cases, convergence is observed from 7 000 scenarios onwards.

It can be concluded that the model learns and continues to learn as scenarios are added, but the choice of 10 000 scenarios is not necessarily the most optimal, and it would have been possible to use fewer scenarios. Indeed, even though it can be seen that the curves converge starting at 7 000 scenarios, looking at the y-axis, we see that the AUC-PR gap between these two curves when the number of scenarios used is for example 2 000 is of the order of 10^{-3}.

5.8.4 Importance of features analysis

The purpose of this section is to analyze the importance of each feature used by the model to predict whether a line is part of the target configuration or not. The importance of each feature is measured at the end of model training, based on its contribution to tree decisions. The sum of all importances is 1. For example, if a feature has an importance of 0.3, this means that it contributed 30% to the line classification decision. The feature importances for the model created using the first experiment are shown in Figure 5.4 and those for the model created using the second experiment are shown in Figure 5.5.

First, we can see from Figures 5.4 that the features with the greatest importance for the first experiment are the topological features. On the contrary, those with the least impact are the features obtained by power flow. For both the IEEE 33 network and the IEEE 69 network, the most important feature is the distance between the bus from of the line and the slack. This importance is 0.25 for the IEEE 33 network and 0.35 for the IEEE 69 network. It makes sense that the importance of this distance feature increases with the size of the network. The second most important feature for the IEEE 33 network is the difference in betweenness, with an importance of 0.3, and for the IEEE 69 network, it is the betweenness of the bus to, named v, with an importance of 0.25. Next, for both the IEEE 33 network and the IEEE 69 network, several features have an importance between 0.05 and 0.10. These features are all either topological or static. Finally, we note that the features obtained by calculating power flow contribute little and are the least important.

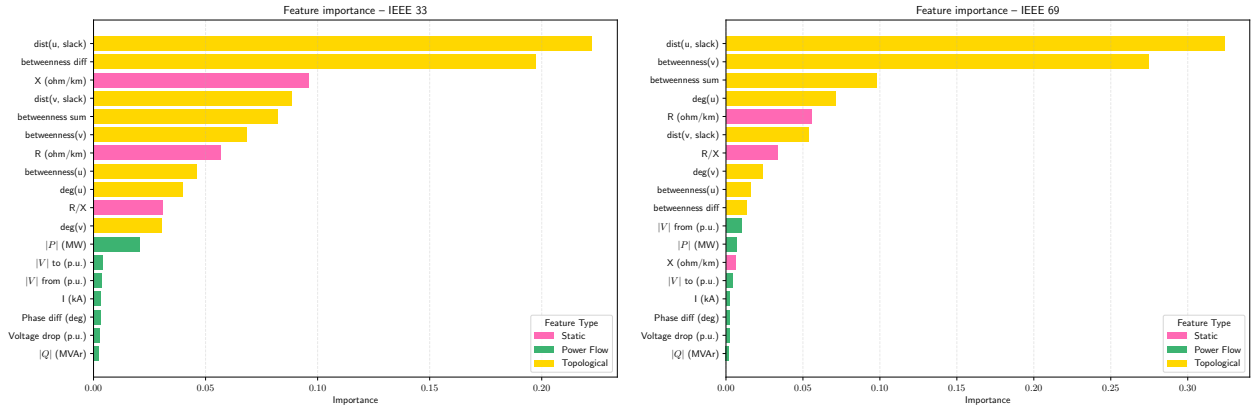


Figure 5.4: Importance of features - experiment 1 (all-generator training) - IEEE 33 and 69 networks

Regarding the Figures 5.5 representing the second experiment, exactly the same trends as for the first experiment are seen. Indeed, the importance of features depends on the network but is not dependent on the methods used that we have presented.

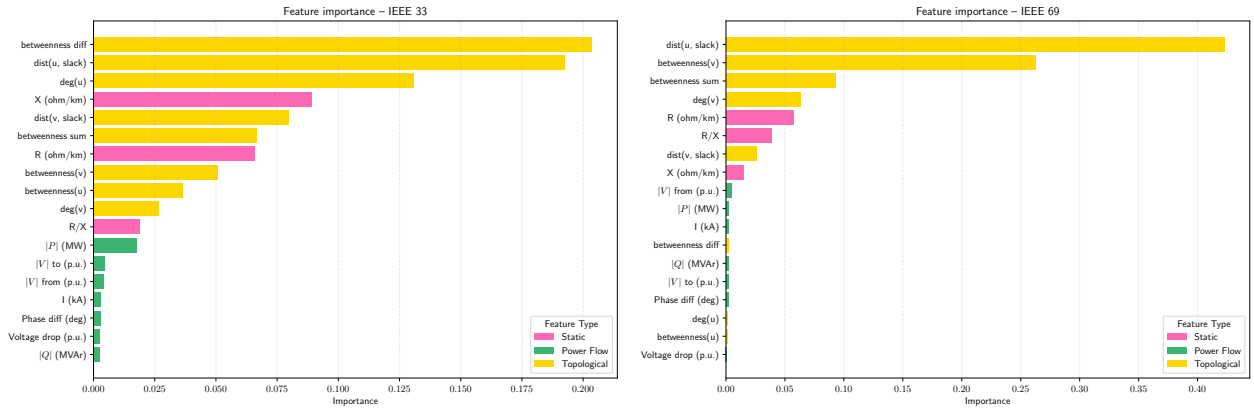


Figure 5.5: Importance of features - experiment 2 (partial-generator training) - IEEE 33 and 69 networks

In short, features that contribute the most to the model are the topological features, followed by the static ones. Features obtained by powerflow calculations have much less influence on reconfiguration decisions. This low importance of the features obtained through power flow computation may be due, in part, to the hypothesis of conditional independence of the lines. Indeed, power flow calculation is a global calculation that takes into account the interactions between all elements of the network. The model

may have more difficulty evaluating these features since they depend on the entire network. Conversely, topological or static characteristics, which describe a line in a more local way, are easier to exploit in this context.

5.8.5 Discussion and summary

Based on the results obtained and presented throughout this section, some general conclusions can be drawn regarding the behavior of the XGBoost model as implemented.

The performance of the trained model is great, demonstrating the relevance of a model such as the one proposed in the context of distribution network reconfiguration.

Although both strategies work well, the partial-generator training experiment is notable for its capacity to correctly forecast the ideal configuration in situations it has never seen. Lastly, the most useful line characteristics for the model were identified through a feature analysis. The characteristics obtained by powerflow calculations are often the most expensive to obtain and appear to be the least influential, at least for these two test networks and under the hypothesis of conditional independence. This opens the door to exploring models that do not include these features and that would be more lightweight.

In summary, all the results obtained validate the two proposed strategies. As a reminder, the goals of a machine learning approach for the DNR problem were twofold: to enable generalization and reduce computational time. The first goal can be considered as achieved when seeing the results presented, and the second is explored in the next section.

5.9 Comparison of the ML method to heuristic methods on known scenarios

In chapter 3, scenarios were defined for each of the two test networks. The descriptions can be found in section 3.3. The goal of this section is to use the machine learning models created on these scenarios and to compare the performances, compared to the other methods seen throughout this study (MST method, MST-LS method, MST-Tabu method).

5.9.1 Comparison of methods on the IEEE 33 network

For the IEEE 33 network, five scenarios were presented. For each of these five scenarios, the Table 5.9 shows the computation time for each method. On the other hand, the figures 5.7 show the representations by scenario and by method of the losses generated by the configuration proposed by method and the computation times. The method using the tabu search is not shown in the computation time figure because its computation times are high and hinder the visibility of the graph.

First of all, when looking at the losses by method and by scenario, it can be seen that the losses generated by the machine learning model obtained by all-generator training (experience one, called ML-V1 in the graph) are very close to the losses generated by the two heuristics (LS and TS) but are never lower. This makes sense since the training is done using the losses generated by the LS method as a reference. It can even be said that the model using the all-generator training perform rather well on the IEEE 33 network given how the losses are almost identical to those obtained by heuristics. However, when it comes to the model trained with the partial-generator training strategy, Figure 5.7 shows that losses are often significant except in scenario 1. This behavior may seem surprising given the performance results presented in the previous section, but the first thing to remember is that when creating the training

set for this model, generators G7 and G8 were not included. Then, during testing, they were added as follows: 3,000 scenarios were generated by randomly drawing 0 to 8 DGs from G1-G8. However, among the 5 scenarios presented for the IEEE 33 network, 4 contain G7 and G8 (scenarios S2 to S5). Even though the accuracy, F1-score, and AUC-PR are very high, this does not guarantee the reliability of the model. Indeed, there are scenarios for which the values of these metrics are low. To better understand this point, Figure 5.6 is presented. This figure shows the distributions of the three metrics used. First, we note that the minimum AUC-PR is 0.96, which is close to the calculated average value. However, looking at the accuracy, we see that the minimum value is 0.70 and, more importantly, we see that the median value is 0.95. This means that half of the test scenarios classify 95% of the lines correctly. However, the other half of the scenarios do not reach this value. This shows that despite promising overall results, there are cases where the model does not perform well. This highlights one of the limitations of this model.

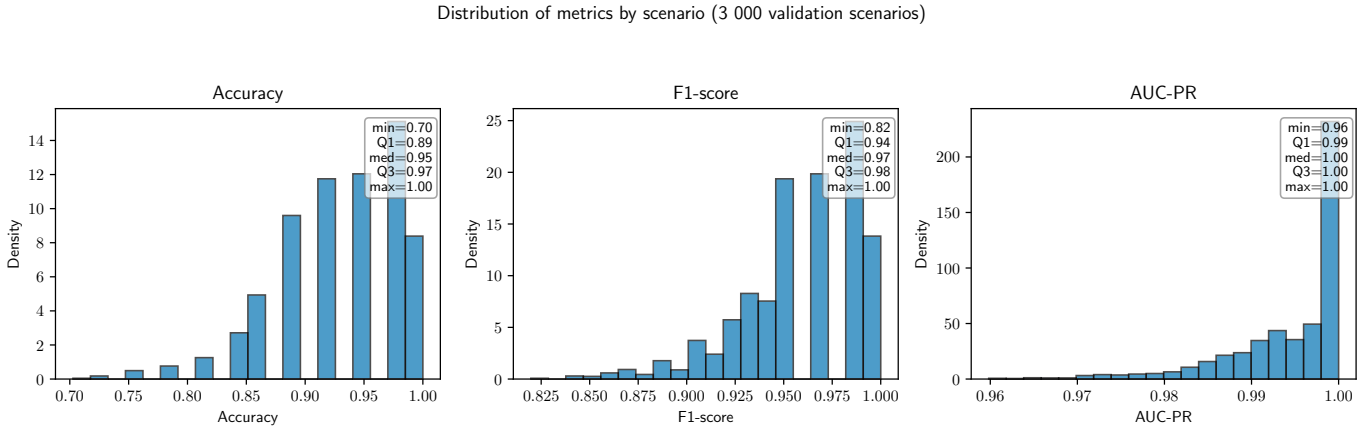


Figure 5.6: Distribution of metrics by scenario - IEEE 33

As a reminder, the second objective of machine learning methods was to reduce computation time. When looking at the Table 5.9 and the figure showing the different computation times by method and by scenario 5.7, it can be seen quite easily that the computation times of machine learning methods are the lowest. They are only comparable with the MST-PF method, but when looking at the losses,there is much better performance with machine learning models for the same computation time. Even though the times are small for almost all methods, it shows the capability of the machine learning algorithm to scale larger networks. In the case of the IEEE 33 network, for the scenarios studied, machine learning models allow for significantly reduced computation times.

Table 5.9: Computation time (in seconds) per method and scenario - IEEE 33 network

Scenario	**MST-PF**	**MST + LS**	**MST + LS + TS**	**ML_V1**	**ML_V2**
S1	0.047	0.747	19.741	0.075	0.072
S2	0.050	2.782	21.858	0.092	0.085
S3	0.046	1.446	17.019	0.080	0.079
S4	0.056	1.644	18.166	0.086	0.076
S5	0.044	0.0365	16.905	0.078	0.071

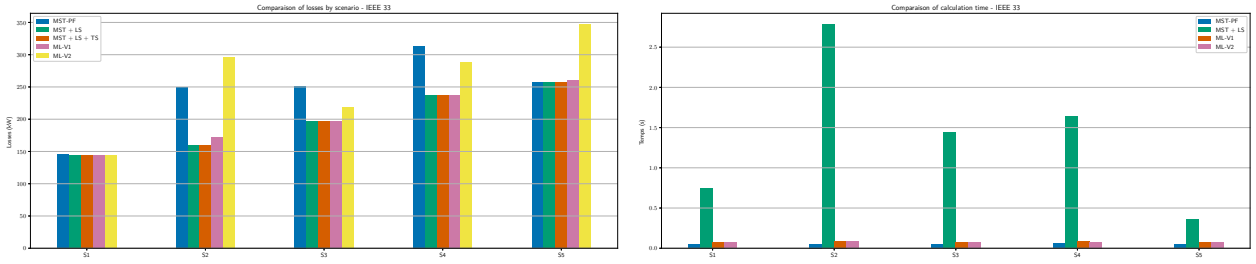


Figure 5.7: IEEE 33 network - Losses and computation time for different methods by scenario

5.9.2 Comparison of methods on the IEEE 69 network

For the IEEE 69 network, six scenarios were presented. For each of these scenarios, Table 5.10 shows the computation times for each method. On the other hand, the Figures 5.8 show, as for the IEEE 33 case, the times and losses based on the scenarios and methods. As with the IEEE 33 network, the MST + LS + TS method is not represented in the computation time graph due to its significant computation times.

Firstly, with regard to losses, we see that, as with the IEEE 33 network, in cases where the generator excluded from the training is not present (G3 in the case of the IEEE 69 network), the losses obtained using machine learning models are often of the same order as those obtained by local search. However, when this new generator comes into play, the losses obtained by the ML-V2 model are greater than those obtained by the ML-V1 model.

Regarding computation times, the conclusion is the same as for the IEEE 33 network. Machine learning models remain unbeatable when it comes to predicting a configuration in a short time.

Table 5.10: Computation time (in seconds) per method and scenario - IEEE 69 network

Scenario	MST-PF	MST + LS	MST + LS + TS	ML_V1	ML_V2
S1	0.046	1.093	36.858	0.102	0.095
S2	0.046	0.718	37.475	0.092	0.079
S3	0.045	1.680	44.519	0.205	0.209
S4	0.136	3.673	60.467	0.580	0.081
S5	0.051	1.021	25.948	0.092	0.085
S6	0.045	1.355	38.473	0.091	0.080

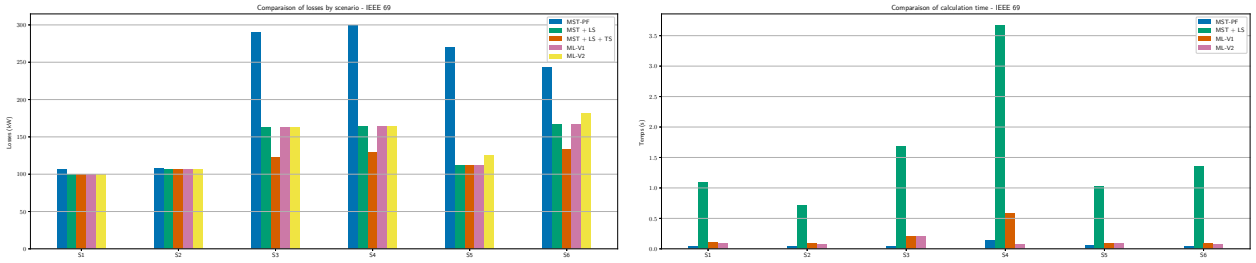


Figure 5.8: IEEE 69 network - Losses and computation time for different methods by scenario

Whether for the IEEE 33 network or the IEEE 69 network, the machine learning model obtained from the all-generator training strategy performs significantly well in predicting a configuration, as the local

search algorithm would, but in a more reasonable amount of time. When it comes to the second proposed model, we see that there are limitations and that, despite encouraging overall results, performance on more complex cases with assets that were not seen during training is not always remarkable, which reduces the reliability of the model.

5.10 Conclusion of the chapter

The results obtained and presented in this chapter confirm that the ML approach based on an XGBoost model as described is relevant and complements heuristics well in the context of the distribution network configuration problem. Several elements highlight the correct functioning of the models. First, metrics showing high values such as the F1-score, accuracy, and AUC-PR, all three of which exceed 0.9 in almost all scenarios. Computation times are also improved thanks to the machine learning model. The latter allows computation times to be divided by 10 to 100 times compared to the LS heuristic. This point opens the door to dynamic reconfiguration, and therefore in real-time or near-real-time, something that would not be possible with the computation times of heuristics.

The proposed machine learning model has its advantages, as mentioned in the previous paragraph, but it is important to highlight its limitations. The first point to mention is that the model never perform better than the LS algorithm. Indeed, it was designed to imitate it. Another important point is the machine learning model's weak generalization capacity. We saw that when new assets were added, the model had a little more difficulty proposing a network configuration that minimized losses. This is an issue that could be reworked by, for example, modifying features and introducing generators as features in their own right instead of just evaluating their effect using power flow. Thirdly, we can mention class imbalance. This problem was addressed by tuning the hyperparameters, but it is something to consider when using a classification model. Fourthly, we can mention the cost of the features obtained by PowerFlow, which are often more computationally expensive to obtain but have little effect on decisions. Finally, the artificial generation of scenarios as carried out in this study, although it provides preliminary results, may not represent the actual operational conditions of distribution networks and may not take into account aspects such as outages.

Considering the points mentioned and analyzing the proposed model, several areas could be subject to improvement. First, it would be possible to explore other more complex models (neural networks or graph neural networks) that can process inputs in multidimensional form, thereby eliminating the need to make the conditional independence assumption. Secondly, a second area for improvement could be a comprehensive analysis of the choice of features in order to determine, in cases where the conditional independence assumption is not necessary, whether power flow features become more important, for example.

Finally, to conclude, the experiments proposed allow us to evaluate the advantages of machine learning models, but also highlight the limitations of these methods and the importance of in-depth knowledge of the networks on which we work.

Conclusion

The subject of this thesis was the problem of reconfiguring distribution networks. The aim of the study was to make new contributions to the subject by proposing methods for optimizing network topology in order to minimize active losses and improve the voltage profile. As a reminder, the problem is NP-hard. Indeed, it has a combinatorial character due to the state of the switches and a non-linear aspect imposed by the power flow equations. Its exact solution on real-size networks is not feasible in a reasonable amount of time. Several approaches were studied in this thesis to counter this problem.

First, we were able to define an initial solution using the Minimum Spanning Tree algorithm. Different weighting criteria were presented for the MST, and the conclusion was that the use of currents was the most appropriate.

In the rest of the thesis, the solution obtained by MST is refined using two heuristics: a local search and a tabu search. Both proposals show encouraging results and improve the solution obtained by MST. Two variants of TS were tested: a single-objective version focused on loss reduction and a multi-objective version that also included the voltage profile. The single-objective version showed slightly better results on the test networks.

Finally, in order to make reconfiguration even faster and more suitable for potential real-time use, a machine learning approach is proposed. A model (XGBoost) was trained to reproduce the behavior of the heuristics. It shows positive results and significantly reduces computation time. However, it is important to note that despite positive overall performance, the ability to generalize to assets never seen during training can still be improved.

The entire work shows that a combination of classic heuristic methods and more recent approaches such as machine learning approaches is a promising avenue for the reconfiguration of distribution networks.

This thesis leaves room for improvement. Among the possible avenues for improvement, the first would be to test these algorithms on larger networks. Indeed, there is no open source data available, and the algorithms have all been tested only on IEEE 33 and IEEE 69 networks. It is therefore not possible to say that what is true here is also true in a different network. A second avenue for improvement would be to improve the machine learning model. Indeed, the latter still has limitations, as presented in section 5.10, particularly its ability to generalize to assets it has never seen before.

In conclusion, this contribution brings simpler algorithms such as the minimum spanning tree back into the spotlight. These algorithms are often underused, yet highly effective. Combining them with more comprehensive methods such as the heuristics presented or machine learning models is a promising avenue for further research on DNR.

Appendix A

IEEE 33 network data

Table A.1 lists all the characteristics of the IEEE 33 network. By convention, the loads are given for the to-bus in each case.

Table A.1: IEEE 33 network data

From bus	To bus	P [kW]	Q [kVar]	R [Ω]	X [Ω]
1	2	100	60	0.0922	0.0470
2	3	90	40	0.4930	0.2512
3	4	120	80	0.3661	0.1864
4	5	60	30	0.3811	0.1941
5	6	60	20	0.8190	0.7070
6	7	200	100	0.1872	0.6188
7	8	200	100	1.7117	1.2357
8	9	60	20	1.0299	0.7400
9	10	60	20	1.0440	0.7400
10	11	45	30	0.1967	0.0651
11	12	60	35	0.3744	0.1237
12	13	60	35	1.4680	1.1549
13	14	120	80	0.5416	0.7129
14	15	60	10	0.5909	0.5260
15	16	60	20	0.7462	0.5449
16	17	60	20	1.2889	1.7210
17	18	90	40	0.7320	0.5739
2	19	90	40	0.1640	0.1564
19	20	90	40	1.5042	1.3555
20	21	90	40	0.4095	0.4784
21	22	90	40	0.7089	0.9373
3	23	90	50	0.4512	0.3084
23	24	420	200	0.8980	0.7091
24	25	420	200	0.8959	0.7010
6	26	60	25	0.2031	0.1034
26	27	60	25	0.2842	0.1447
27	28	60	20	1.0589	0.9338
28	29	120	70	0.8043	0.7006
29	30	200	100	0.5074	0.2585
30	31	150	70	0.9745	0.9629
31	32	210	600	0.3105	0.3619

Table A.1 – continued

From bus	To bus	P [kW]	Q [kVar]	R [Ω]	X [Ω]
32	33	60	40	0.3411	0.5302
8	21	90	40	2.0000	2.0000
9	15	60	10	2.0000	2.0000
12	22	90	40	2.0000	2.0000
18	33	60	40	0.5000	0.5000
25	29	120	70	0.5000	0.5000

Appendix B

IEEE 69 network data

Table B.1 lists all the characteristics of the IEEE 69 network. By convention, the loads are given for the to-bus in each case.

Table B.1: IEEE 69 network data

From bus	To bus	P [kW]	Q [kVar]	R [Ω]	X [Ω]
1	2	0	0	0.0005	0.0012
2	3	0	0	0.0005	0.0012
3	4	0	0	0.0015	0.0036
4	5	0	0	0.0251	0.0294
5	6	2.6	2.2	0.3660	0.1864
6	7	40.4	30.0	0.3810	0.1941
7	8	75.0	54.0	0.0922	0.0470
8	9	30.0	22.0	0.0493	0.0251
9	10	28.0	19.0	0.8190	0.2707
10	11	145.0	104.0	0.1872	0.0619
11	12	145.0	104.0	0.7114	0.2351
12	13	8.0	5.0	1.0300	0.3400
13	14	8.0	5.5	1.0440	0.3450
14	15	0.0	0.0	1.0580	0.3496
15	16	45.5	30.0	0.1966	0.0650
16	17	60.0	35.0	0.3744	0.1238
17	18	60.0	35.0	0.0047	0.0016
18	19	0.0	0.0	0.3276	0.1083
19	20	1.0	0.6	0.2106	0.0690
20	21	114.0	81.0	0.3416	0.1129
21	22	5.0	3.5	0.0140	0.0046
22	23	0.0	0.0	0.1591	0.0526
23	24	28.0	20.0	0.3463	0.1145
24	25	0.0	0.0	0.7488	0.2475
25	26	14.0	10.0	0.3089	0.1021
26	27	14.0	10.0	0.1732	0.0572
3	28	26.0	18.6	0.0044	0.0108
28	29	26.0	18.6	0.0640	0.1565
29	30	0.0	0.0	0.3978	0.1315
30	31	0.0	0.0	0.0702	0.0232
31	32	0.0	0.0	0.3510	0.1160

Table B.1 – continued

From bus	To bus	P [kW]	Q [kVar]	R [Ω]	X [Ω]
32	33	14.0	10.0	0.8390	0.2816
33	34	19.5	14.0	1.7080	0.5646
34	35	6.0	4.0	1.4740	0.4873
3	36	26.0	18.55	0.0044	0.0108
36	37	26.0	18.55	0.0640	0.1565
37	38	0.0	0.0	0.1053	0.1230
38	39	24.0	17.0	0.0304	0.0355
39	40	24.0	17.0	0.0018	0.0021
40	41	1.2	1.0	0.7283	0.8509
41	42	0.0	0.0	0.3100	0.3623
42	43	6.0	4.3	0.0410	0.0478
43	44	0.0	0.0	0.0092	0.0116
44	45	39.22	26.3	0.1089	0.1373
45	46	39.22	26.3	0.0009	0.0012
4	47	0.0	0.0	0.0034	0.0084
47	48	79.0	56.4	0.0851	0.2083
48	49	384.7	274.5	0.2898	0.7091
49	50	384.7	274.5	0.0822	0.2011
8	51	40.5	28.3	0.0928	0.0473
51	52	3.6	2.7	0.3310	0.1114
9	53	4.35	3.5	0.1740	0.0886
53	54	26.4	19.0	0.2030	0.1034
54	55	24.0	17.2	0.2842	0.1447
55	56	0.0	0.0	0.2813	0.1433
56	57	0.0	0.0	1.5900	0.5337
57	58	0.0	0.0	0.7837	0.2630
58	59	100.0	72.0	0.3042	0.1006
59	60	0.0	0.0	0.3861	0.1172
60	61	1244.0	888.0	0.5075	0.2585
61	62	32.0	23.0	0.0974	0.0496
62	63	0.0	0.0	0.1450	0.0738
63	64	227.0	162.0	0.7105	0.3619
64	65	59.0	42.0	1.0410	0.5302
11	66	18.0	13.0	0.2012	0.0611
66	67	18.0	13.0	0.0047	0.0014
12	68	28.0	20.0	0.7394	0.2444
68	69	28.0	20.0	0.0047	0.0016
11	43	6.0	4.3	0.5000	0.5000
13	21	114.0	81.0	0.5000	0.5000
15	46	39.22	26.3	1.0000	1.0000
27	65	59.0	42.0	1.0000	1.0000
50	59	100.0	72.0	2.0000	2.0000

Bibliography

[1] Hossein Lotfi, Mohammad Ebrahim Hajiabadi, and Hossein Parsadust. Power distribution network reconfiguration techniques: A thorough review. *Sustainability*, 16(23), 2024.

[2] Yunchen Huo, Ran Yi, Yanni Xie, Changlin Huang, and Jingyao Tong. Genetic algorithm. https://optimization.cbe.cornell.edu/index.php?title=Genetic_algorithm#:~:text=The%20Genetic%20Algorithm%20,modal%20problems. Consulté le 23 mars 2025.

[3] Bo Yang, Jingbo Wang, Xiaoshun Zhang, Tao Yu, Wei Yao, Hongchun Shu, Fang Zeng, and Liming Sun. Comprehensive overview of meta-heuristic algorithm applications on pv cell parameter identification. *Energy Conversion and Management*, 2020.

[4] Elsevier. Chapter 3 : Training and optimization algorithms. *Applications of Artificial Intelligence Techniques in the Petroleum Industry*, 2020.

[5] Pierre Geurts and Louis Wehenkel. Introduction to machine learning. https://people.montefiore.uliege.be/lwh/AIA/, 2024. Consulté le 24 mars 2025.

[6] Lilian Weng. A (long) peek into reinforcement learning. https://lilianweng.github.io/posts/2018-02-19-rl-overview/, 2018. Consulté le 24 mars 2025.

[7] FEBEG. Transport et distribution de l'électricité. https://www.febeg.be/fr/transport-et-distribution-de-lelectricite#:~:text=Le%20r%C3%A9seau%20de%20transmission%3A%20C,de%2070%20kV%20et%20plus., 2025. Consulté le 22 mars 2025.

[8] Ayoub El berkaoui, Saida Bahsine, Aziz Oukennou, Fatima Ait nouh, and Bouchra Rzine. Distribution network topology planning and optimization: a brief review. *E3S Web of Conferences*, 469, 2023.

[9] Seethiah V. and Ah King R. T. F. Distribution network reconfiguration for power loss reduction using harmony search. *8th International Conference on Advances in Power System Control*, pages 1–6, 2009.

[10] Bertrand Cornélusse. ELEC0447 Analysis of Electric Power and Energy Systems. https://github.com/bcornelusse/ELEC0447-analysis-power-systems, 2024. Consulté le 22 mars 2025.

[11] a-eberle. Puissance réactive : augmenter l'efficacité et la stabilité du réseau électrique. https://www.a-eberle.de/fr/savoir/puissance-aveugle/#:~:text=Puissance%20r%C3%A9active%20(Q)%20%3A&text=Elle%20repr%C3%A9sente%20la%20puissance%20%C3%A9lectrique%20qui%20circule%20entre%20les%20conducteurs,n'effectue%20aucun%20travail%20m%C3%A9canique., 2024. Consulté le 22 mars 2025.

[12] Christopher Yeh. Simplified DistFlow Equations. https://chrisyeh96.github.io/2023/03/28/simplified-distflow.html, 2024. Consulté le 23 mars 2025.

[13] Vassilis Kekatos. Distflow and lindistflow. https://engineering.purdue.edu/ECE/Academics/Undergraduates/UGO/CourseInfo/courseInfo?courseid=841&show=true&type=grad, 2021. Consulté le 23 mars 2025.

[14] Masoud Farivar and Steven H. Low. Branch flow model: Relaxations and convexification—part i. *IEEE Transactions on Power Systems*, 28(3), August 2013.

[15] Diego Issicaba and Jorge Coelho. Evaluation of the forward-backward sweep load flow method using the contraction mapping principle. *IEEE Transactions on Power Systems*, 6(6):3229 – 3237, December 2016.

[16] Letif Mones. A gentle introduction to optimal power flow. `https://invenia.github.io/blog/2021/06/18/opf-intro/`, 2021. Consulté le 10 avril 2025.

[17] Florin Capitanescu, Luis F. Ochoa, Harag Margossian, and Nikos D. Hatziargyriou. Assessing the potential of network reconfiguration to improve distributed generation hosting capacity in active distribution systems. *IEEE Transactions of Power Systems*, 30(1), January 2015.

[18] Omar Kahouli, Haitham Alsaif, Yassine Bouteraa, Naim Ben Ali, and Mohamed Chaabene. Power system reconfiguration in distribution network for improving reliability using genetic algorithm and particle swarm optimization. *MDPI : Applied Sciences*, 11, March 2021.

[19] Nastaran Gholizadeh and Petr Musilek. A generalized deep reinforcement learning model for distribution network reconfiguration with power flow-based action-space sampling. *Energies*, 17, October 2024.

[20] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, 2009.

[21] Algorithmes. Comment choisir entre les algorithmes heuristiques et métaheuristiques ? `https://www.linkedin.com/advice/0/how-can-you-choose-between-heuristic-metaheuristic-algorithms?lang=fr&lang=fr&originalSubdomain=fr`. Consulté le 23 mars 2025.

[22] Diogo Freitas, Luiz Guerreiro Lopes, and Fernando Morgado-Dias. Particle swarm optimisation: A historical review up to the current developments. *MDPI : Entropy*, 11, March 2020.

[23] Peter Rossmanith. Le recuit simulé. `https://interstices.info/le-recuit-simule/`, 2009. Consulté le 23 mars 2025.

[24] Gilles Louppe. Introduction to artificial intelligence. `https://github.com/glouppe/info8006-introduction-to-ai`, 2025. Consulté le 24 mars 2025.

[25] Gilles Louppe. Deep learning. `https://github.com/glouppe/info8010-deep-learning`, 2025. Consulté le 24 mars 2025.

[26] A. MERLIN and H. PACK. Search for a minimal-loss operating spanning tree configuration in an urban power distribution system. *Proc. 5th Power System Computation Conf., Cambridge, UK*, 5:1–18, september 1975.

[27] Dariush Shirmohammadi and H. Wayne Hong. Reconfiguration of electric distribution networks for resistive line losses reduction. *IEEE Transactions of Power Delivery*, 7(2), April 1989.

[28] Cassio Gerez, Eduardo Coelho Marques Costa, and Alfeu J. Sguarezi Filho. Distribution network reconfiguration considering voltage and current unbalance indexes and variable demand solved through a selective bio-inspired metaheuristic. *Energies*, 15, February 2022.

[29] Mesut E. Baran and Felix F.Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power Delivery*, 4(2), April 1989.

[30] Ali Azizivahed, Hossein Narimani, Mehdi Fathi, Ehsan Naderi, and Mohammad Rasoul Narimani Hamid Reza Safarpour. Multi-objective dynamic distribution feeder reconfiguration in automated distribution systems. *ScienceDirect*, 147, 2018.

[31] Hossein Lotfi, Reza Ghazi, and Mohammad bagher Naghibi-Sistani. Multi-objective dynamic distribution feeder reconfguration along with capacitor allocation using a new hybrid evolutionary algorithm. *Energy Systems*, 11, 2020.

[32] Wei-Chen Lin, Chao-Hsien Hsiao, Wei-Tzer Huang, Kai-Chao Yao, Yih-Der Lee, Jheng-Lun Jian, and Yuan Hsieh. Network reconfiguration framework for co2 emission reduction and line loss minimization in distribution networks using swarm optimization algorithms. *Sustainability*, 16, February 2024.

[33] S. Civanlar, J.J Grainger, H. Yin, and S.S.H Lee. Distribution feeder reconfiguration for loss reduction. *IEEE Transactions on Power Delivery*, 3(3), July 1988.

[34] Koichi Nara, Atsushi Shios, Minoru Kitagawa, and Toshihisa Ishihara. Implementation of genetic algorithm for distribution systems loss minimum reconfiguration. *IEEE Transactions on Power Systems*, 7(3), August 1992.

[35] Young-Jae Jeon and Jae-Chul Kim. Network reconfiguration in radial distribution system using simulated annealing and tabu search. *IEEE Power Engineering Society Winter Meeting*, 4:2329–2333, 2000.

[36] Giovanny Andrés Diaz Vargas, Darin Jairo Mosquera, and Edwin Rivas Trujillo. Optimization of topological reconfiguration in electric power systems using genetic algorithm an nonlinear programming with discontinuous derivatives. *Electronics*, 13, February 2024.

[37] Milad Rahimi Pour Behbahani, Alireza Jalilian, and MohamadAli Amini. Reconfiguration of distribution network using discrete particle swarm optimization to reduce voltage fluctuations. *Wiley*, May 2020.

[38] Surender Reddy Salkuti. Feeder reconfiguration in unbalanced distribution system with wind and solar generation using ant lion optimization. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 12(3), 2021.

[39] JIE WANG, WEIQING WANG, and HAIYUN WANG HUIWEN ZUO. Dynamic reconfiguration of multiobjective distribution networks considering dg and evs based on a novel ldbas algorithm. *IEEE*, 2020.

[40] Weiye Zhenga, Wanjun Huanga, and David J. Hilla. A deep learning-based general robust method for network reconfiguration in three-phase unbalanced active distribution networks. *Electrical Power and Energy Systems*, 2020.

[41] Wanjun Huang, Weiye Zheng, and David J. Hill. Distribution network reconfiguration for short-term voltage stability enhancement: An efficient deep learning approach. *IEEE transactions on smart grid*, 12(6), November 2021.

[42] NASTARAN GHOLIZADEH, NAZLI KAZEMI, and PETR MUSILEK. A comparative study of reinforcement learning algorithms for distribution network reconfiguration with deep q-learning-based action sampling. *IEEE*, 11, February 2023.

[43] BEIBEI WANG, HONG ZH, HONGHUA XU, YUQING BAO, and HUIFANG DI. Distribution network reconfiguration based on noisynet deep q-learning network. *IEEE*, 9, June 2021.

[44] M.A. Tavakoli Ghazi Jahania, P. Nazariana, A. Safarib, and M.R. Haghifamc. Multi-objective optimization model for optimal reconfiguration of distribution networks with demand response services. *Sustainable Cities and Socie*, 47, March 2019.

[45] Sérgio F. Santos, Matthew Gough, Desta Z. Fitiwi, José Pogeira, Miadreza Shafie-khah, and João P. S. Catalão. Dynamic distribution system reconfiguration considering distributed renewable energy sources and energy storage systems. *IEEE Systems Journal*, 16(3), September 2022.

[46] Hossein Lotfi, Mohammad Ebrahim Hajiabadi, and Hossein Parsadust. Power distribution network reconfiguration techniques : a thorough review. *Sustainibility*, 16, November 2024.

[47] Zihao Li, Wenchuan Wu, Boming Zhang, and Xue Tai. Analytical reliability assessment method for complex distribution networks considering post-fault network reconfiguration. *IEEE TRANSACTIONS ON POWER SYSTEMS*, 35(2), March 2020.

[48] Xiaoniumang. Ieee 33 bus and 69 bus system modelling. https://github.com/xiaoniumang/pandapower, 2017.

[49] Arif Wazir and Naeem Arbab. Analysis and optimization of ieee 33 bus radial distributed system using optimization algorithm. *(JETAE) Journal of Emerging Trends in Applied Engineering*, 1(2):2518–4059, 2016.

[50] N.C. Sahoo and K. Prasad. A fuzzy genetic approach for network reconfiguration to enhance voltage stability in radial distribution systems. *Energy Conversion and Management*, 47:3288–3306, March 2006.

[51] Hamed Ahmadi and José R. Martí. Minimum-loss network reconfiguration : A minimum spanning tree problem. *Sustainable Energy, Grids and Networks*, 1(9), 2015.

[52] Dionicio Zocimo Ñaupari Huatuco, Luiz Otávio Pinheiro Filho, Franklin Jesus Simeon Pucuhuayla, and Yuri Percy Molina Rodriguez. Network reconfiguration for loss reduction using tabu search and a voltage drop. *Energies*, 17:2744, June 2024.

[53] Marcos A. N. Guimaraes and Carlos A. Castro. Reconfiguration of distribution systems for loss reduction using tabu search. *Energies*, 17(11):2744, 2024.

[54] GeeksForGeeks. What is tabu search? `https://www.geeksforgeeks.org/what-is-tabu-search/`, 2024. Consulté le 13 avril 2025.

[55] Geeksforgeeks. Time and space complexity analysis of kruskal algorithm. `https://www.geeksforgeeks.org/time-and-space-complexity-analysis-of-kruskal-algorithm/`, 2024.

[56] S.K.Goswami. Distribution system planning using branch exchange technique. *IEEE Transactions on Power Systems*, 12(2):685–694, May 1997.

[57] Wikipedia. Recherche tabou. `https://fr.wikipedia.org/wiki/Recherche_tabou`, 2017.

[58] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Energies*, 13(5):533–549, 1986.

[59] Miqing Li and Xin Yao. Dominance move: A measure of comparing solution sets in multiobjective optimization. *CERCIA, School of Computer Science, University of Birmingham*, 2017.

[60] Aniket Andhale. Tabu search | artificial intelligence. `https://medium.com/@andy08/tabu-search-artificial-intelligence-9853f44e6923`, 2024. Consulté le 2 mai 2025.

[61] Zhenkun Yang Min Jiang and Zhaohui Gan. Optimal components selection for analog active filters using clonal selection algorithms. *Department of Computer Science, Wuhan University of Science & Technology*, 2007.

[62] scikitLearn. Gradient-boosted trees. `https://scikit-learn.org/stable/modules/ensemble.html`, year = 2025, note = Consulté le 2 mai 2025.

[63] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *University of Washington*, 2016.

[64] Jakub Czakon. F1 score vs roc auc vs accuracy vs pr auc: Which evaluation metric should you choose? `https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc`, 2025. Consulté le 2 mai 2025.

[65] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.