TESE DE DOUTORADO Nº 285

SCHEDULING MANEUVERS FOR THE RESTORATION OF ELECTRIC POWER DISTRIBUTION NETWORKS

André Luiz Maravilha Silva

DATA DA DEFESA: 22/10/2018

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

SCHEDULING MANEUVERS FOR THE RESTORATION OF ELECTRIC POWER DISTRIBUTION NETWORKS

André Luiz Maravilha Silva

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Doutor em Engenharia Elétrica.

Orientador: Prof. Felipe Campelo França Pinto
Coorientador: Prof. Eduardo Gontijo Carrano

Belo Horizonte - MG

Outubro de 2018

Silva, André Luiz Maravilha.

S586s Scheduling maneuvers for the restoration of electric power distribution networks [manuscrito] / André Luiz Maravilha Silva. - 2018.

xxii, 74 f., enc.: il.

Orientador: Felipe Campelo França Pinto.
Coorientador: Eduardo Gontijo Carrano.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 69-74.

Bibliografia: f. 61-68.

1. Engenharia elétrica - Teses. 2. Energia elétrica - Distribuição - Teses. 3. Heurística - Teses. I. Pinto, Felipe Campelo França. II. Carrano, Eduardo Gontijo. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

Universidade Federal de Minas Gerais

School of Engineering

Graduate Program in Electrical Engineering

SCHEDULING MANEUVERS FOR THE RESTORATION OF ELECTRIC POWER DISTRIBUTION NETWORKS

André Luiz Maravilha Silva

Thesis presented to the Graduate Program in Electrical Engineering of the School of Engineering of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

Advisor: Prof. Felipe Campelo França Pinto
Co-Advisor: Prof. Eduardo Gontijo Carrano

Belo Horizonte - MG

October 2018

"Scheduling Maneuvers for the Restoration of Electric Power Distribution Networks"

André Luiz Maravilha Silva

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 22 de outubro de 2018.

Por:

[illegible]

Prof. Dr. Felipe Campelo França Pinto
DEE (UFMG) - Orientador

[illegible]

Prof. Dr. Eduardo Gontijo Carrano
DEE (UFMG) - Coorientador

[illegible]

Prof. Dr. Lucas de Souza Batista
DEE (UFMG)

[illegible]

Prof. Dr. Martín Gómez Ravetti
DEP (UFMG)

[illegible]

Prof. Dr. Elisângela Martins de Sá
DCSA (CEFET-MG)

[illegible]

Prof. Dr. Elizabeth Fialho Wanner
Computer Science Group (Aston University)

Dedicated to my friends at ORCS Lab
for all the wonderful conversations we had.

Abstract

During a fault in a power distribution network, energy utilities can change the network topology to reconnect all or at least a portion of disconnected clients, then minimizing the area affected by the fault. These changes in the network are defined by a restoration plan that specifies a set of switches to be maneuvered. The faster energy utilities reconnect disconnected clients, the lighter the penalties applied to them. Then, the energy utilities have a tight time frame to define the restoration plan before dispatching maintenance teams to perform the required maneuvers. Besides, the total time needed to perform the maneuvers has to be considered when determining the restoration plan, since the new topology that restores/minimizes the affected clients will be fully operational only after the maneuvers are completed. Although the problem of restoring power distribution networks is widely studied in the literature, no study has considered both the existence of multiple maintenance teams working in parallel and the time taken by the teams to move between locations where the maneuverable switches are located. Ignoring these characteristics results in inefficient restoration plans, taking longer than expected. In this work, we address the problem of providing a better estimation of the time to perform the restoration plan by modeling the assignment and sequencing of maneuver operations as a scheduling problem that minimizes the makespan, i.e., the total time required to complete all maneuver operations. Furthermore, we present specific heuristics for its solution that are fast enough to be incorporated into existing restoration algorithms without compromising their performance, since they already need to perform other time-consuming routines, e.g., power flow algorithms. Computational experiments with different fault scenarios showed that incorporating the proposed strategy in a restoration algorithm led to more efficient restoration plans.

Resumo

Na ocorrência de falhas em uma rede de distribuição de energia elétrica, as concessionárias de energia podem alterar a topologia da rede para reconectar clientes desconectados, minimizando a área afetada pela falha. Essas alterações na rede são definidas por um plano de restauração que especifica um conjunto de chaves a serem manobradas. Quanto mais rápido os forem clientes desconectados, menores serão as penalidades aplicadas à concessionária. Portanto, as concessionárias têm um curto período de tempo para definir um plano de restauração e enviarem equipes de manutenção para realizarem as manobras de chaveamento. Além disso, o tempo total necessário para realização das manobras deve ser considerado ao determinar o plano de restauração, uma vez que a nova topologia que restaura/minimiza os clientes afetados estará totalmente operacional somente após as manobras estarem concluídas. Embora o problema de restauração de redes de distribuição de energia elétrica seja amplamente estudado na literatura, nenhum estudo considerou, simultaneamente, a existência de múltiplas equipes de manutenção trabalhando em paralelo e o tempo demandado pelas equipes para se descolarem entre os locais onde as chaves de manobra se encontram. Ignorar essas características resulta em planos de restauração ineficientes, levando mais tempo do que o esperado. Neste trabalho, é proposta uma abordagem para fornecer melhores estimativas de tempo de execução de planos de restauração. Isso é feito através da modelagem da atribuição e sequenciamento das tarefas de chaveamento como um problema de sequenciamento de tarefas que minimiza o *makespan*, ou seja, o tempo total para conclusão de todas as operações de manobra na rede. Além disso, heurísticas específicas são apresentadas para solução desse problema de sequenciamento. As heurísticas apresentadas são rápidas o suficiente para serem incorporadas em algoritmos de restauração existentes sem que a eficiência desses algoritmos seja comprometida, uma vez que eles já devem realizar outras rotinas que consomem tempo, por exemplo, algoritmos de fluxo de potência. Experimentos computacionais considerando diferentes cenários de falhas mostraram que o uso da estrategia proposta em um algoritmo de restauração resultou em planos de restauração mais eficientes.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Electric power distribution systems are of great importance, both for the economy and society. In Brazil, there are more than 61.5 million consumer units in 99% of the cities and, from this total, 85% are residential [Aneel, 2008]. Distribution companies are subject to a number of regulations with regard to the quality of service and other technical issues. Besides, the maintenance of distribution systems can present a high cost, requiring the use of optimization techniques to support the operation of distribution systems and guarantee their correct functioning, as well as efficient use of the available resources [Carrano, 2007].

Although distribution networks are designed to meet the demands and operations constraints, these networks are subject to the occurrence of unforeseen faults which result in customers disconnected from their source. Such faults can be quite common in distribution networks with a predominance of overhead lines, particularly in urban regions with large tree canopy coverage, or in regions subject to seasonal rainstorms – all which are quite common in developing countries. In face of fault events, a quick response is required in order to minimize the penalties companies are subject, besides minimizing the negative impact on the consumers. This quick response is achieved by restoration strategies that temporarily change the topology of the distribution network to reconnect customers disconnected from their source until the problem is properly fixed.

There are many works that deal with restoration of electric power distribution networks, employing from mathematical programming techniques to heuristic algorithms as restoration strategies. However, no work has considered a proper estimation of time required to perform the restoration plan. A frequently used approach considers the number of maneuvers as a proxy of time, but this approach does not take into account the presence of remotely controlled switches nor the location of the manually maneuverable switches, which may lead to restoration plans that seem to be efficient in terms of time to be executed, when in fact they are not. Consequently, this approach results in unnecessary degradation of the service reliability indices used to monitor the quality of services offered by the distribution companies.

This work explores this issue in the restoration of electric power distribution networks by proposing an approach to estimate the actual time to perform the restoration plan. In addition, the proposed methods can return a proper sequence in which the maneuvers have to be performed by the available teams to be completed in the expected time. While this work is inserted within the broader scope of service restoration in electric power distribution networks, its focus is not on the definition of switching maneuvers of a restoration plan, but instead on assigning maneuvers to maintenance teams and sequencing their activities to minimize the time taken to perform the restoration plan, given that another approach has already determined the maneuvers to be performed. Then, the proposed approach assumes that the maneuvers to be executed are already known, and attempts to minimize the time required to implement this given restoration plan.

Although the smart grid concepts are being gradually implemented in distribution grids, these networks are far away from being completely automated, especially in developing countries. In the particular case of Brazil, except for a few recent pilot projects in specific cities, we have aged distribution networks which are being updated progressively, albeit slowly and in small incremental steps. In large cities, even for the most automated Brazilian distribution utilities, we have around 90% of the switches requiring manual operation. Up to our knowledge, this is the current scenario for most developing countries. Therefore, the consideration of manual switches remains a relevant challenge to be handled. In addition, it is important to highlight that the proposed approach can be extended to highly automated networks as well, be it as a means to deal with a smaller, but still relevant, proportion of manual switches; or as a means to assign other tasks in the network to multiple maintenance teams.

1.1 Motivation

Previous works in the literature of electric power distribution networks consider neither the displacement times nor the existence of multiple teams working in parallel when attempting to generate load restoration plans. Ignoring these characteristics may lead existing algorithms to return a restoration plan that seems to be fast to perform (e.g., when considering the number of maneuvers as a proxy of time), but in practice requiring an arbitrarily long time, since the location of switches to maneuver and teams are completely ignored.

Given this gap in the relevant technical literature on service restoration for distribution networks, this work is motivated by the need to develop a proper approach not only to estimate the completion time of restoration but an approach that can return the proper sequence in which the maneuvers have to be performed as well.

1.2 Objectives

The main objective of this work is to propose an approach capable of providing a sequence for performing the switching maneuvers that minimizes the completion time of a restoration plan and estimating the proper time of its completion. To achieve this main objective, the specific objectives are defined:

- Modeling the maneuver scheduling problem in the restoration of electric power distribution networks and formulate it as a mathematical programming problem.
- Developing specific heuristic methods that require little computational time, allowing it to be embedded into existing restoration algorithms without compromise their performance.
- Developing local search heuristics to minimize the time required by the maneuver scheduling of the final solution (or candidate solutions) returned by a restoration algorithm.
- Evaluate the proposed algorithms in a stand-alone way to validate the quality of solutions and runtime performance.
- Evaluate the proposed approach when embedded in a restoration algorithm to validate its efficacy and to compare it against the approach frequently used that consider the number of maneuvers as a proxy for time to perform a restoration plan.

1.3 Contributions

The main contributions of this work are:

- The proposal of a proper approach to estimate the time required to perform a restoration plan, considering displacement times and multiple teams working in parallel.

- The proposal of specific heuristics that return the estimation of time and the schedule of switching maneuvers, which are fast enough to be embedded into existing load restoration algorithms without compromising their efficiency.

- To allow existing load restoration algorithms that do not provide a schedule of the maneuvers to return this information with a proper estimation of time required to perform it.

The contributions mentioned above, have resulted in two publications in scientific journals specialized in the area of electrical engineering. These publications are:

Maravilha, A.L., Goulart, F., Carrano, E.G., and Campelo, F. Scheduling maneuvers for the restoration of electric power distribution networks: formulation and heuristics. *Electric Power Systems Research*, 163 Part A, 301–309, 2018. DOI: 10.1016/j.epsr.2018.06.020.

Goulart, F., Maravilha, A.L., Carrano, E.G., and Campelo, F. Permutation-based optimization for the load restoration problem with improved time estimation of maneuvers. *International Journal of Electrical Power & Energy Systems*, 101, 339–355, 2018. DOI: 10.1016/j.ijepes.2018.03.030.

In the first article, the approach proposed in this doctoral dissertation that models the estimation of time required to perform a restoration plan as a scheduling problem is described. Besides, the specific heuristics proposed to solve the scheduling problem are also described in the article.

In the second article, the restoration problem is formulated as a bi-objective optimization problem in which the energy not supplied and power not restored are minimized. For calculating the energy not supplied, it is necessary to obtain an estimation of the restoration time and, for this, the approach proposed in this doctoral dissertation is used. Besides, an algorithm based on the meta-heuristic Simulated Annealing is presented to solve the restoration problem, and therefore providing restoration plans.

1.4 Organization of the text

The current chapter has contextualized the main subject of this work and briefly described the advantages of considering the scheduling of the switching maneuvers in a proper way in the restoration of electric power distribution networks. The following chapters are briefly described:

Chapter 2 describes the load restoration in electric power distribution networks, which is important to understand the problem of scheduling the switching maneuvers that compose a restoration plan. The relevant quality indices used to quantify the quality of a restoration plan are also presented, followed by a discussion of the main works in the literature of load restoration problem.

Chapter 3 formally describes the maneuver scheduling problem and shows how it can be mapped to an identical parallel machine scheduling problem with setup times and precedence constraints. Furthermore, a mixed integer programming formulation is proposed and the literature in scheduling problems related to this work are discussed.

In Chapter 4, specific algorithms are proposed for the solution of the maneuver scheduling problem. Three algorithms are proposed: two greedy heuristics and one improving heuristic. The greedy heuristics are meant to be used collaboratively with restoration algorithms (i.e., embedded into them, although they can be used after the restoration algorithm, as a sequential strategy) and the improving heuristics to be applied at the end, as a polishing strategy, to improve the maneuver scheduling of the final restoration plan.

Chapter 5 describes the computational experiments performed to evaluate the proposed heuristics using a set of instances with different dimensions and characteristics. In addition, the heuristics are also evaluated when embedded into a load restoration algorithm and compared with another commonly used approach that considers the number of maneuvers as a proxy for time.

Finally, in Chapter 6 the conclusions are presented, ending with some ideas of continuity that can be explored in future works.

Chapter 2

Load restoration in electric power distribution networks

Electrical distribution systems are often operated in a radial configuration, which provides numerous advantages such as easier fault current protection, voltage control, lower cost, prediction and control of load flows, among others [Short, 2014]. As illustrated in Figure 2.1, a distribution system can be represented by an undirected graph $\mathcal{G}_{DS} = (\mathbb{V}, \mathbb{E})$, with edges indicating maneuverable switches and solid/dashed lines depicting the currently closed (CC)/currently open (CO) switches, respectively. Nodes indicate load sectors.

Despite the advantages of a radial configuration, the consequences of a fault in any part of the system normally propagate to a larger portion of the network, as all loads located downstream from the faulted point become *out of service* (*oos*), as illustrated in Figure 2.2. As it may be evident from Figure 2.2, a fault at node E causes the protection switch 2 to activate[1], leaving (healthy) nodes H, I, J and L disconnected as well. Such power outages can cause severe impacts on customers as well as on power distribution companies, which may be fined by regulatory agencies.

The quality of service provided by power distribution companies is usually measured by (at least) two reliability indices [Short, 2014; IEEE, 2012], one relative to the duration and the other to the frequency of outages:

- System Average Interruption Duration Index (SAIDI): it indicates the total duration of interruption for the average customer during a predefined period and usually measured in minutes or hours of interruption.

[1]Throughout this work we assume the protection system is properly coordinated [Yadav et al., 2014].

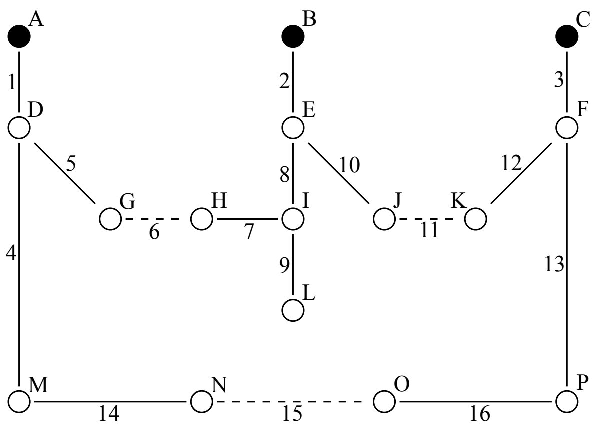


Figure 2.1: Example of a radial distribution network. Black circles (A, B, C) indicate feeder nodes, while all other nodes represent load sectors. Solid lines indicate the current configuration of the network (i.e., closed switches), while dashed ones represent currently open switches.

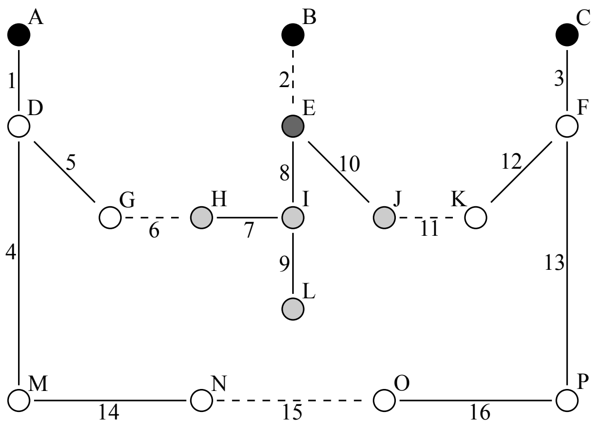
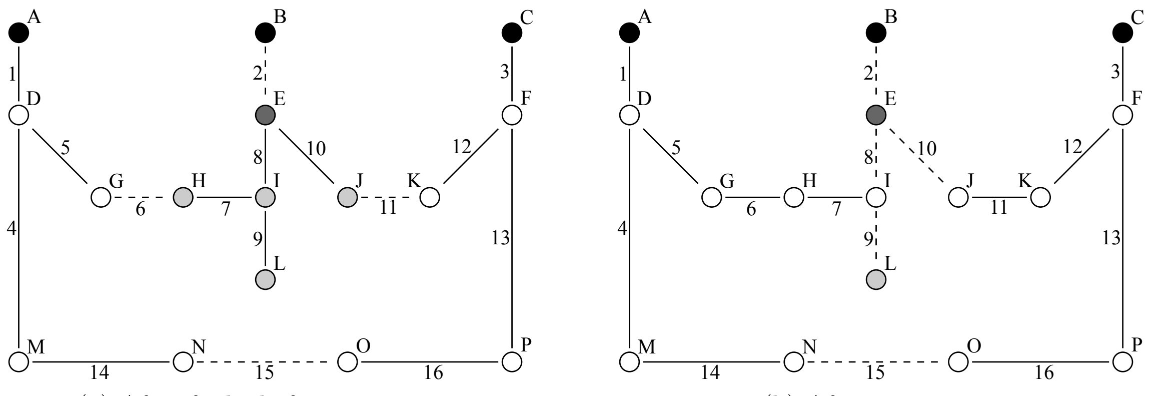


Figure 2.2: Example of the effect of a fault in a radial distribution network. Black circles (A, B, C) indicate feeder nodes, while all other nodes represent load sectors. Solid lines indicate the current configuration of the network (i.e., closed switches), while dashed ones represent currently open switches. A fault on sector E disconnects all loads downstream from it.

- System Average Interruption Frequency Index (SAIFI): it indicates how often the average customer experiences a sustained interruption over a predefined period.

In terms of SAIDI, the longer a load is disconnected, the worse this index value becomes, so there is a strong motivation for recovering healthy but *oos* loads after a power outage, while the cause of the fault is not properly fixed.

The recovering of healthy *oos* loads is performed accordingly to a restoration plan, which consists in changing the network topology (opening CC and closing CO switches) to reconnect the disconnected (healthy) load sectors to one of the feeders, subject to the network operational constraints. Let $\mathbb{E}' \subseteq \mathbb{E}$ be the set of switches



(a) After fault, before restoration

(b) After restoration

$(8, O)$ $(9, O)$ $(6, C)$ $(10, O)$ $(11, C)$

(c) Set of switching maneuvers

Figure 2.3: Example of a restoration plan performed after a fault in a radial distribution network. Black circles (A, B, C) indicate feeder nodes, while all other nodes represent load sectors. Solid lines indicate the current configuration of the network (i.e., closed switches), while dashed ones represent currently open switches. (a) A fault on sector E disconnects all loads downstream from it. (b) Resulting topology after a possible restoration plan. In this example node L was not restored due to overload constraints. (c) Switching maneuvers that compose the restoration plan applied at (a), resulting the topology illustrated in (b).

maneuvered in a restoration plan. A switch $e \in \mathbb{E}'$ can be opened or closed, depending on its current state. Thus, the notation $(e,\ \theta)$ is used to describe a restoration plan, in which $\theta \in \{O, C\}$ is the operation – O for opening and C for closing – to perform on switch $e \in \mathbb{E}'$. Figure 2.3 illustrates a restoration plan and the resulting network configuration after a fault. The restoration plan consists of the following maneuvers: (i) open switches 8 and 9 and close switch 6 in order to recover nodes H and I; (ii) open 10 and close 11 to restore J. Notice that node L remained disconnected, which may be necessary to prevent constraints violations.

It is important to highlight that some precedence rules must be satisfied when performing the maneuvers to avoid reconnecting faulted sectors and other constraints violations. For example, if the operation $(6, C)$ is performed before $(8, O)$, the faulted sector E will be energized, which cannot happen; or a situation in which performing $(6, C)$ before $(9, O)$ overloads feeder A. Therefore, it is important in a restoration plan to inform the sequence in which the maneuvers should be performed, in addition to the final network configuration.

The goal of load restoration is usually to restore as much *oos* loads as possible in the shortest time, without violating network operational constraints such as minimum voltage in loads, maximum current in lines, feeder capacity, and radiality of the system. Notice that the resulting topology after a restoration plan is temporary and the original topology is restored after the fault is fixed.

There are many studies in the literature that address the load restoration problem in electrical power distribution networks. However, there is a lack of a common formulation. Then each work solves this problem using different quality indices discussed in the following section.

2.1 Quality indices

The quality of a restoration plan is usually estimated by a variety of indices, three of which are directly relevant to this work: power not restored ($\mathcal{I}_{S_{NR}}$), number of maneuvers ($\mathcal{I}_N$) and time of maneuvers ($\mathcal{I}_T$). These three indices are summarized in Table 2.1.

Table 2.1: Relevant indices used for measuring quality of a restoration plan.

Quality Index	**Description**
Power not restored ($\mathcal{I}_{S_{NR}}$)	The sum of apparent power in each node that remains *oos* after a restoration plan. The smaller this index, the less customers are disconnected.
Number of maneuvers ($\mathcal{I}_N$)	Number of switches to be operated in a sequence of maneuvers. It does not take into account the distinction between *remotely operated* and *manually operated* switches, and not even the possible differences in time among these last ones.
Time of maneuvers ($\mathcal{I}_T$)	The actual time taken to perform all operations in a sequence of maneuvers. It depends on conditions such as traffic, weather, initial position of the dispatch team(s) etc.

In a perfect scenario, all *oos* healthy loads are reconnected at the end of the restoration process. However, such situation is not always possible, and some loads may remain disconnected from a source at the end of the process. Then, $\mathcal{I}_{S_{NR}}$ is an important index to be considered when evaluating the quality of a restoration plan. Even in a situation in which it is possible to restore all the *oos* healthy loads, this can take excessively long that it is preferable to restore just a subset of the loads, while the others remain disconnected until the fault is properly fixed. Then, the time required to perform a restoration plan is important as well, and therefore the index $\mathcal{I}_T$ should be considered when determining the restoration plan.

Although the index $\mathcal{I}_N$ does not necessarily measure the time, it is frequently used as a proxy for the time taken to perform a restoration plan. In this case, a greater number of maneuvers is translated as a long time to perform a restoration plan. Despite being quickly calculated, $\mathcal{I}_N$ does not considers the displacement times required for the teams to reach the switches location nor multiple teams working in parallel. Even more importantly in the context of smart grids, the existence of remotely controlled switches means that some maneuvers may require virtually no time to be performed.

2.2 Related works

There is a large number of works that deal with the load restoration in electric power distribution networks, and therefore a large diversity of algorithms designed for this problem. Some works approach the problem using mathematical programming [Hijazi and Thiébaux, 2015; Borges et al., 2016; Romero et al., 2016], which may be prohibitive due to the time constraints demanded by the problem. Other works propose heuristics based on rules usually adopted by engineers when solving the restoration problem or constructive heuristics to define the new configuration of the network [Aoki et al., 1989; Hsu et al., 1992; Hsu and Huang, 1995; Chen et al., 2002; Dimitrijevic and Rajakovic, 2011; Gholami et al., 2015b,a]. These heuristic strategies run fast, but may result in solutions of poor quality due to their limited capacity to explore the space of solutions.

To overcome the drawbacks presented by mathematical programming and simple heuristic techniques, metaheuristic strategies and local search techniques have been widely used for the load restoration problem [Watanabe, 2005; Garcia and França, 2008; Camillo et al., 2016; Kumar et al., 2006b,a; Sanches et al., 2014; Carrano et al., 2016; Marques et al., 2018]. Such strategies run faster than those based on mathematical programming techniques and are able to explore the space of solutions broadly.

Despite the advantages brought by the use of metaheuristics and local search techniques, most studies ignore the time of restoration or use approaches that do not model the reality. For example, the works of Hsu and Huang [1995]; Garcia and França [2008]; Dimitrijevic and Rajakovic [2011]; Mohammadi and Afrakhteh [2012]; Kumar et al. [2006b,a]; Carvalho et al. [2007]; Sanches et al. [2014]; Gholami et al. [2015a]; Camillo et al. [2016]; Marques et al. [2018] use $\mathcal{I}_N$ as a proxy for the time taken to perform a restoration plan, despite the disadvantages discussed in Section 2.1.

Some works [Carrano et al., 2016; Watanabe, 2005] recognize that $\mathcal{I}_T$ is actually a more realistic index, and try to estimate it by defining a constant time required for each switch to be operated. Watanabe [2005] arbitrarily sets this value to 1 for all switches, while Carrano et al. [2016] assume it to be provided by the engineer. In both cases the total time would be the sum of the individual times for each operated switch, which results in two fundamental issues: (i) it assumes the switches are operated sequentially, ignoring the availability of more than one maintenance team whose coordinated actions can considerably reduce the time; and (ii) the time to execute a maneuver depends mostly on the relative position between the team and the switch, thus modeling it with a constant value is not realistic. In effect, this index becomes a weighted $\mathcal{I}_N$, with the weights bearing no physical interpretation of time, and thus provides little advance with respect to what is customary in the literature.

Furthermore, not all algorithms are able to provide a proper sequence of maneuvers that can be performed on the time estimated by them. Table 2.2, summarizes the strategies used by these works, showing the quality indices used by them and the capability of providing a proper sequence of the maneuvers.

Table 2.2: Summary with some features (F1–F3, described at the bottom of the table) the studies cited in this work should possess. Inside each cell, Y means 'yes' and N stands for 'no'.

Method	Reference	F1	F2	F3
Mathematical programming	Hijazi and Thiébaux [2015]	N	N	N
	Romero et al. [2016]	Y	N	N
	Borges et al. [2016]	N	Y	N
Constructive Heuristics	Aoki et al. [1989]	N	Y	N
	Hsu et al. [1992]	N	Y	N
	Chen et al. [2002]	N	Y	N
	Dimitrijevic and Rajakovic [2011]	Y	N	N
	Gholami et al. [2015b]	Y	N	N
	Gholami et al. [2015a]	Y	N	N
Metaheuristics-based Strategies	Watanabe [2005]	Y	Y	N
	Kumar et al. [2006b]	Y	N	N
	Kumar et al. [2006a]	Y	N	N
	Camillo et al. [2016]	Y	N	N
	Carrano et al. [2016]	Y	Y	N
	Marques et al. [2018]	Y	Y	N

F1: The strategy differentiates remotely from manually maneuvered switches.
F2: The strategy returns a sequence of maneuvers or it is evident from the algorithm.
F3: The strategy considers the actual time to perform the load restoration (i.e., the time of maneuvers).

It is important to keep in mind that the information provided in Table 2.2 does not aim at comparing the strategies proposed by those works, since they consider different quality indices and formulations for the load restoration problem. Nonetheless, it is useful for contrasting these works with regard to the approach used to estimate the time taken to perform the restoration plan and the capability of returning a proper sequence of maneuvers. For instance, none of the studies cited so far presents a proper estimation of the actual time to implement a restoration plan, taking the number of maintenance teams and their displacement times into consideration. This is probably due to the necessity of assigning switching maneuvers to teams and sequencing them to estimate the time of restoration. Then, to calculate $\mathcal{I}_T$ is not as simple as to calculate $\mathcal{I}_N$. Moreover, it is necessary a fast approach to calculate $\mathcal{I}_T$ to make its use feasible in practice.

The maneuver scheduling proposed in this work, and published in [Maravilha et al., 2018], allows future works for the restoration of electric power distribution networks to provide a proper estimation of time to perform the load restoration. In this sense, the work of Goulart et al. [2018] is the first study to embed the approach proposed in this work into a restoration algorithm. In Goulart et al. [2018], the restoration problem is modeled as a bi-objective optimization problem in which the energy not supplied and power not restored are minimized. The estimation of the time taken to perform a restoration plan is used to calculate the energy not supplied more accurately. For solving the problem, an algorithm based on the metaheuristic Simulated Annealing was also proposed, which resulted in significantly better results compared to other approaches.

Chapter 3

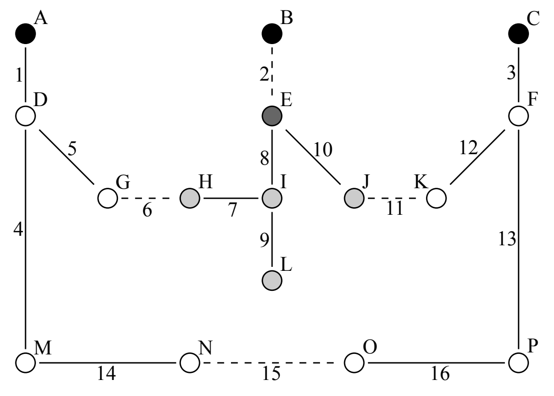
Maneuver scheduling problem

As mentioned in Chapter 2, the quality of restoration plans is measured by quality indices. While the power not restored $\mathcal{I}_{S_{NR}}$ and the number of maneuvers $\mathcal{I}_N$ can be readily computed given the new configuration of the network, for the determination of the actual time $\mathcal{I}_T$ it is necessary to assign the tasks of operating (manually controlled) switches to maintenance teams, and to determine the sequence in which the maneuvers (manual and remote ones) are performed. Only after that, it is possible to have a reliable estimation of the total time required to run the restoration plan.

For assigning and sequencing maneuvers, it is necessary to consider only the switches that have to be operated. The remaining switches and the charges they connect are irrelevant to this process. Then, the problem of assigning and sequencing maneuvers in the restoration of an electric power distribution network can be modeled over a directed graph $\mathcal{G} = (\mathbb{N}, \mathbb{A})$. The nodes $\mathbb{N} = \{0, 1, 2, \ldots, n\}$ represent the locations of the teams and switches. Node 0 is the initial position of the maintenance teams, and nodes $1, 2, \ldots, n$ are the location of switches to maneuver. The switches are divided into manually and remotely controlled ones. Thus, the set of nodes can be decomposed into three disjoint sets $\mathbb{N} = \{0\} \cup \mathbb{N}' \cup \mathbb{N}''$, in which $\mathbb{N}'$ is the set of manually controlled switches and $\mathbb{N}''$ is the set of the remotely controlled ones.

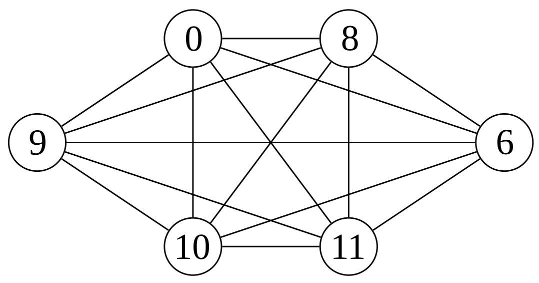
The arcs in $\mathbb{A}$ represent the paths between two locations. Since it is not necessary to dispatch any team to locations where remotely controlled switches are installed, the set of arcs is defined as $\mathbb{A} = \{(i, j) \ : \ i \in (\mathbb{N}' \cup \{0\}), j \in \mathbb{N}', \ i \neq j\}$. Note that there are no arcs entering or leaving the nodes in $\mathbb{N}''$, and also no arcs entering node 0.

Let $\mathbb{M} = \{1, 2, \ldots, m\}$ denote the set of available maintenance teams, and let $s_{i,j,\ell} \geq 0$ denote the time it takes for team $\ell \in \mathbb{M}$ to transverse $(i, j) \in \mathbb{A}$. The time required to maneuver a switch $i \in (\mathbb{N} \setminus \{0\})$ is given by $p_i \geq 0$. These parameters account for possible practical delays, like checking if a team is cleared for operating



$$\underbrace{(8,O)\quad(9,O)\quad(6,C)\quad(10,O)\quad(11,C)}_{\text{Switching Maneuvers}}$$

(a) Network after fault, before restoration; and the set of maneuvers of the restoration plan.



(b) Graph $\mathcal{G} = (\mathbb{N}, \mathbb{A})$ of the maneuver scheduling problem resulting from the restoration plan in (a).

Figure 3.1: Example of an instance of the maneuver scheduling problem obtained from the restoration plan in Figure 2.3. In (a), the network after a fault at node E and the set of maneuvers that compose the restoration plan is presented. In (b) the graph used to model the resulting instance of the maneuver scheduling problem is presented. The node 0 represent the initial position of the maintenance teams.

the switch, and is usually known. To illustrate the maneuver scheduling problem, Figure 3.1 shows the graph obtained from the restoration plan presented in Figure 2.3.

In addition, let $\mathcal{G}_{\prec}(\mathbb{N}' \cup \mathbb{N}'', \mathbb{P})$ be a precedence graph representing the precedence relations between switch operations. An arc $(i, j) \in \mathbb{P}$ means the existence of a relation $i \prec j$, i.e., switch j can be maneuvered only after switch i is maneuvered. Notice that the precedence graph is obtained from the precedence rules imposed by a restoration plan, which are important to avoid the violation of constraints. For example, if maneuver $(11, C)$ is performed before $(10, O)$, the fault in load E is re-energized, which cannot happen, then $(10, O)$ must precede $(11, C)$. Figure 3.2 shows the precedence graph obtained by the precedence rules imposed by restoration plan in Figure 3.1b.

3.1 Mixed integer programming formulation

The MIP formulation presented in this section is based on binary linear ordering variables [Dyer and Wolsey, 1990]. Linear ordering variables are also referred to as sequencing variables [Pinedo, 2016], as they allow the determination of the relative ordering

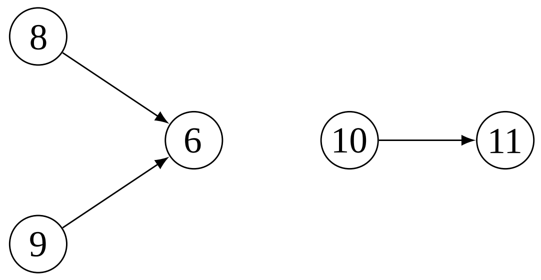


Figure 3.2: Precedence graph associated to the restoration plan from Figure 3.1. An edge (i, j) means that a switch (node in the graph) j can be maneuvered only after switch i is maneuvered. For example, switches 8 and 9 cannot be maneuvered before 6, since there is paths that connect 8 and 9 to 6. However, nodes 8, 9 and 10 can be operated in any sequence among them, since there are no paths in the graph that connect them.

between pairs of maneuvers assigned to the same team. This formulation assumes that the weak triangular inequality, $s_{i,j,\ell} \leq s_{i,k,\ell} + p_k + s_{k,j,\ell}$, $\forall \ell \in \mathbb{M}$ and $i, k, j \in \mathbb{N}'$, with $i \neq k \neq j$, is satisfied.

Given the decision variables:

$$y_{i,\ell} = \begin{cases} 1, & \text{if the maneuver of switch } i \in \mathbb{N}' \text{ is assigned to team } \ell \in \mathbb{M}; \\ 0, & \text{otherwise.} \end{cases}$$

$$z_{i,j} = \begin{cases} 1, & \text{if } i \in \mathbb{N}' \text{ is maneuvered before } j \in \mathbb{N}'; \\ 0, & \text{otherwise.} \end{cases}$$

$t_i \geq 0$, the moment in which the maneuver of $i \in \mathbb{N} \setminus \{0\}$ is performed;

$C_{max} \geq 0$, the moment in which all maneuvers are completed (makespan);

the maneuver scheduling problem in the restoration of electric power distribution networks can be formulated as the following MIP problem:

$$\text{Min. } C_{max} \tag{3.1}$$

$$\text{s.t.: } \sum_{\ell \in \mathbb{M}} y_{i,\ell} = 1 \qquad \forall i \in \mathbb{N}' \tag{3.2}$$

$$z_{i,j} + z_{j,i} \geq y_{i,\ell} + y_{j,\ell} - 1 \qquad \forall \ell \in \mathbb{M};\ \forall i,\ j \in \mathbb{N}';\ i < j \tag{3.3}$$

$$z_{i,j} + z_{j,i} \leq 1 \qquad \forall i,\ j \in \mathbb{N}';\ i < j \tag{3.4}$$

$$z_{i,k} + z_{k,j} + z_{j,i} \leq 2 \qquad \forall i,\ k,\ j \in \mathbb{N}';\ i \neq k \neq j \tag{3.5}$$

$$t_i \geq \sum_{\ell \in \mathbb{M}} s_{0,i,\ell}\ y_{i,\ell} \qquad \forall i \in \mathbb{N}' \tag{3.6}$$

$$t_j \geq t_i + p_i + \sum_{\ell \in \mathbb{M}} s_{i,j,\ell}\ y_{j,\ell} - \mathcal{M}(1 - z_{i,j}) \qquad \forall i,\ j \in \mathbb{N}';\ j \neq i \tag{3.7}$$

$$t_j \geq t_i + p_i \qquad \forall (i,j) \in \mathbb{P} \tag{3.8}$$

$$C_{max} \geq t_i + p_i \qquad \forall i \in \mathbb{N} \setminus \{0\} \tag{3.9}$$

$$y_{i,\ell} \in \{0,1\} \qquad \forall i \in \mathbb{N}';\ \forall \ell \in \mathbb{M} \tag{3.10}$$

$$z_{i,j} \in \{0,1\} \qquad \forall i,\ j \in \mathbb{N}';\ i \neq j \tag{3.11}$$

$$t_i \geq 0 \qquad \forall i \in \mathbb{N} \setminus \{0\} \tag{3.12}$$

$$C_{max} \geq 0 \tag{3.13}$$

in which the objective (3.1) is the minimization of the makespan, subject to the constraints:

- Each maneuver of a manually controlled switch has to be assigned to a single maintenance team (3.2): for each $i \in \mathbb{N}'$ the sum of all $y_{i,\ell}$ for $\ell \in \mathbb{M}$ has to be equal to 1, which means that one and only one $y_{i,\ell} = 1$.

- If two switches are maneuvered by the same team, then there is an ordering relation between them. The right side of (3.3) will be equal to one if switches $i \in \mathbb{N}'$ and $j \in \mathbb{N}'$ are assigned to the same team $\ell \in \mathbb{M}$, forcing at least one of $z_{i,j}$ or $z_{j,i}$ to be one, defining the ordering.

- If there is an ordering relation between $i \in \mathbb{N}'$ and $j \in \mathbb{N}'$, either i is maneuvered before j, or j is maneuvered before i. Constraints (3.4) ensure that no more than one of $z_{i,j}$ and $z_{j,i}$ assumes the value 1.

- Given three distinct switches $i,\ k,\ j \in \mathbb{N}'$, if i is maneuvered before k and k before j, then j cannot be maneuvered before i, i.e., the sum of the linear ordering variables $z_{i,k}$, $z_{k,j}$, $z_{j,i}$ (3.5) cannot be greater than 2.

- The moment a switch $i \in \mathbb{N}'$ is maneuvered, even if it is the first switch maneuvered by the team to which it is assigned, cannot be earlier than the time the team takes to arrive at the switches. Constraints (3.6) ensure that t_i is at least equal to the time the team takes to arrive at the location of switch i.

- If $j \in \mathbb{N}'$ is maneuvered after $i \in \mathbb{N}'$, then the moment j is maneuvered cannot be earlier than the moment i is maneuvered plus the processing time p_i and the displacement time $s_{i,j,\ell}$. Disjunctive constraints (3.7) are defined to ensure these requirements. Given a sufficiently large number $\mathcal{M}$, if $z_{i,j} = 1$ then the $\mathcal{M}(1-z_{i,j}) = 0$ and the inequality $t_j \geq t_i + p_i + \sum_{\ell \in \mathbb{M}} s_{i,j,\ell}\, y_{j,\ell}$ has to be satisfied. When $z_{i,j} = 0$, $\mathcal{M}$ must be larger than $t_i + p_i + \sum_{\ell \in \mathbb{M}} s_{i,j,\ell}\, y_{j,\ell}$, so that t_i does not limit the values t_j can assume. Thus, $\mathcal{M}$ must be larger than the largest possible value for $t_i + p_i + \sum_{\ell \in \mathbb{M}} s_{i,j,\ell}\, y_{j,\ell}$, which occurs when all switching maneuvers are assigned to the same team. Then, a value for $\mathcal{M}$ can be calculated as:

$$\mathcal{M} = \sum_{j \in \mathbb{N}'} \left(p_j + \max_{(i,\ell)} \{ s_{i,j,\ell} : i \in \mathbb{N}',\ \ell \in \mathbb{M},\ i \neq j \} \right). \tag{3.14}$$

- The moment a switch (manually or remotely operated) is maneuvered must satisfy the precedence constraints given by the precedence graph $\mathcal{G}_{\prec}$. Constraints (3.8) ensure that all precedence constraints are satisfied.

- Constraints (3.9) are used to compute the makespan C_{max}.

- Finally, constraints (3.10)–(3.13) define the domain of the decision variables: $y_{i,\ell}$ and $z_{i,j}$ are binary variables, and t_i and C_{max} are non-negative continuous variables.

The MIP formulation presented above uses a big-M formulation in the disjunctive constraints (3.7), which usually leads to significant gaps between the integer solution and that of the linear relaxation [Wolsey and Nemhauser, 1999]. Methods based on linear relaxation may, therefore, require a large number of iterations until they can find the proven optimal solution. However, small to medium-sized instances can be solved with the formulation proposed above in a short amount of time, since the time required to resolve its linear relaxation is smaller than the time required by tighter formulations.

Different approaches can be used to formulate scheduling problem as a MIP problem. They mainly differ on the constraints used to calculate the moment in which each task starts. Then, two other alternative MIP formulations were implemented and compared against the formulation based on linear ordering variables. A full description of these alternative formulations can be seen in Appendix A.

The first alternative formulation evaluated is based on binary precedence variables [Manne, 1960], in which the sequence of maneuver operations performed by each team is modeled as a flow starting at node 0 (the team's initial position) that goes through nodes of switching maneuvers. These flows determine the ordering in which each team must perform the switching maneuvers assigned to them. This formulation uses a big-M constant also.

The second alternative MIP formulation evaluated is based on binary arc-time-indexed variables [Pessoa et al., 2010]. Formulations based on arc-time-indexed variables do not use big-M constants and usually provides better linear relaxations. However, they lead to a rapid increase in the number of variables and constraints and therefore may require the use of decomposition techniques to be solved.

3.1.1 Comparison of formulations

The MIP formulations were implemented in C++17 with Gurobi 8.0.1 [Gurobi Optimization, Inc., 2017] through its API for C++. The code was compiled using GNU Compiler Collection (GCC) version 8.1.1 with compiler optimization flag set to "-O2". It was used the default settings of Gurobi, but limited to a single thread and time limit of 1 hour. All experiments were performed on a dual 2.10 GHz Intel(R) Xeon(R) Silver 4116 machine with 156 GiB of main memory running Fedora 28 (64-bits).

It was considered a set of random instances with different number of switching maneuvers and teams available. Each instance was solved once with each formulation, in which were registered: the value of objective function of the incumbent solution when it stopped, the objective function of the linear relaxation, the status of the optimization (if the solution returned is the optimal or not), the runtime, the number of MIP nodes explored, and the gap between the upper and lower bound when the solver stopped.

Figure 3.3 shows, for each formulation, the proportion of instances in which at least one feasible solution was found and the proportion of the instances optimally solved within the time limit of 1 hour. Figures 3.4 and 3.5 show the same results stratified by problem size. Notice that for instances with a number of maneuvers $n \geq 50$ the formulation based on arc-time-indexed variables did not return any solution. In fact, this formulation was not able to run on the available machine for instances of these sizes. The formulation based on precedence variables was not able to find any feasible solutions to some instances with $n \geq 50$ within the time limit of 1 hour. However, different from the arc-time-indexed formulation, it was able to run. For all instances, the formulation based on linear ordering variables was able to find at least one feasible solution.

Table 3.1 shows the results obtained with Gurobi for each formulation. These results consider only the runs in which a feasible solution was found. Columns "n" and "m" are, respectively, the number of maneuvers and the number of teams; column *"Obj."* contains the mean values of objective function of the incumbent solution; column

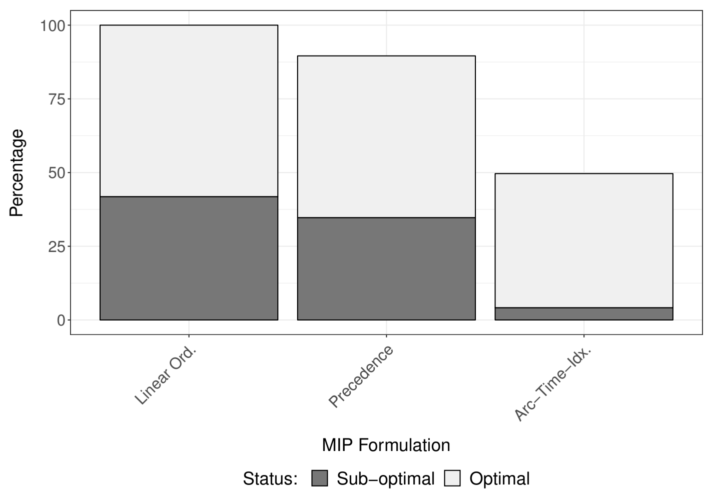


Figure 3.3: Proportion of instances solved by each MIP formulation. The dark gray part of the bars is the proportion of instances in which the respective formulation was able to find at least one feasible solution but did not find the proven optimal solution in the time limit of 1 hour. The light gray part of the bars is the proportion of instances optimally solved.

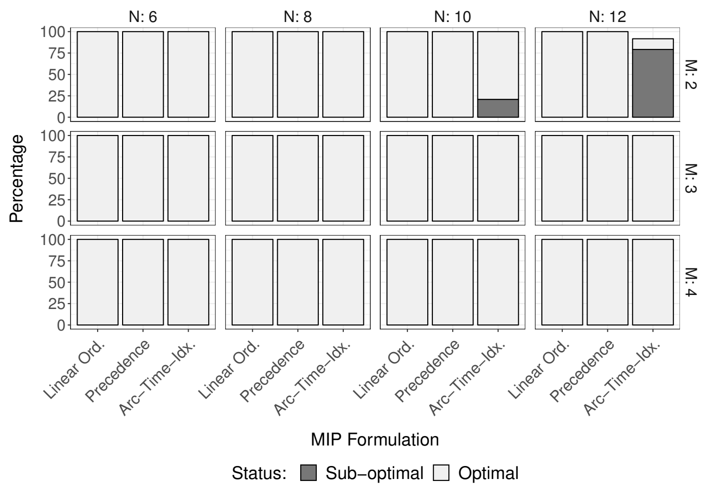


Figure 3.4: Proportion of small instances solved by each MIP formulation stratified by instance size. The dark gray part of the bars is the proportion of instances in which the respective formulation was able to find at least one feasible solution but did not find the proven optimal solution in the time limit of 1 hour. The light gray part of the bars is the proportion of instances optimally solved.

"Gap (%)" is the mean value of gap between the upper and lower bounds when Gurobi stopped; column *"RL"* is the mean value of the linear relaxation; column *"Nodes"* contains the mean number of nodes explored by the solver; and column *"Time (s)"* is the runtime in seconds. Notice that the time was limited to 1h, i.e., 3600 seconds.

From the results presented at Figures 3.3, 3.4 and 3.5 together with the results

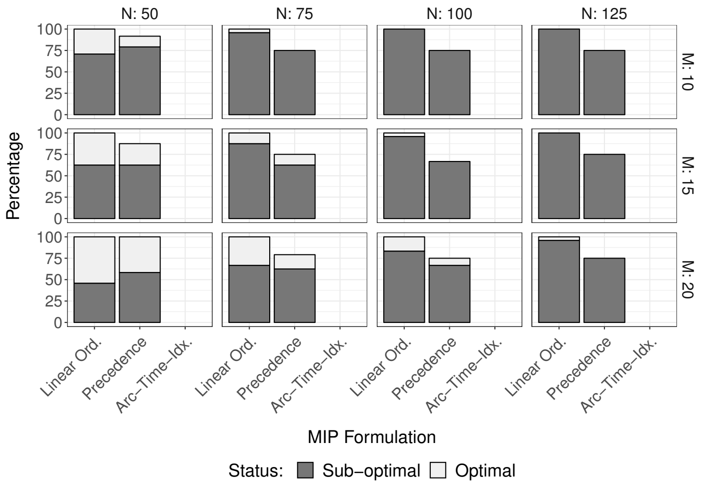


Figure 3.5: Proportion of large instances solved by each MIP formulation stratified by instance size. The dark gray part of the bars is the proportion of instances in which the respective formulation was able to find at least one feasible solution but did not find the proven optimal solution in the time limit of 1 hour. The light gray part of the bars is the proportion of instances optimally solved.

from the Table 3.1, the MIP formulation based on linear ordering variables presents a better trade-off among the number of solutions solved and strength of the formulation. With this formulation, it was possible to find at least one feasible solution for all instances.

The formulation based on precedence variables is dominated by the linear ordering formulation since it solved fewer instances to optimality within the time limit and demanded a longer time when both were able to find the optimal solution. Moreover, the formulation based on precedence variables presented the worst values of linear relaxation. The formulation based on arc-time-indexed variables, despite presenting a better linear relaxation and demand less number of MIP nodes, demanded a long time and was not able to run on large instances.

Since the context of load restoration requires a quick response to the maneuver scheduling problem to make the use of the proposed approach feasible in practice, exact methods to solve MIP formulations may be prohibitive as solution strategy. However, their results can be used as a reference to evaluate the performance of heuristics.

Table 3.1: Results obtained by solution of the MIP formulation using Gurobi solver and greedy heuristics. The results presented in this table are the mean values for each instance size.

		Linear Ordering Variables					Precedence Variables					Arc-Time-Indexed Variables				
n	m	Obj.	Gap	LR	Nodes	Time (s)	Obj.	Gap	LR	Nodes	Time (s)	Obj.	Gap	LR	Nodes	Time (s)
6	2	33.42	0.00	19.29	49.58	0.05	33.42	0.00	8.33	75.04	0.08	33.42	0.00	22.00	28.33	0.18
	3	25.67	0.00	17.58	19.71	0.04	25.67	0.00	7.71	1.00	0.06	25.67	0.00	19.41	0.79	0.04
	4	21.29	0.00	17.50	21.33	0.04	21.29	0.00	8.17	1.00	0.05	21.29	0.00	18.55	0.46	0.03
8	2	43.42	0.00	20.04	643.08	0.28	43.42	0.00	9.25	1383.50	0.52	43.42	0.00	23.92	1589.50	11.69
	3	31.21	0.00	18.08	998.88	0.37	31.21	0.00	8.71	994.50	0.60	31.21	0.00	20.49	607.25	0.99
	4	23.62	0.00	16.92	21.25	0.11	23.62	0.00	8.75	21.83	0.13	23.62	0.00	18.33	0.71	0.06
10	2	51.79	0.00	20.50	1886.96	1.54	51.79	0.00	9.83	17166.25	31.83	51.83	5.64	23.94	92988.00	1868.21
	3	35.29	0.00	17.98	2527.33	1.89	35.29	0.00	9.62	1965.08	15.33	35.29	0.00	21.31	1603.25	18.63
	4	29.25	0.00	18.08	2394.92	1.80	29.25	0.00	9.29	2377.71	5.29	29.25	0.00	19.94	772.62	1.86
12	2	55.88	0.00	19.71	3444.17	4.34	55.88	0.00	9.62	59612.83	168.95	57.82	33.02	24.28	26135.32	3244.34
	3	39.17	0.00	20.21	4452.08	5.52	39.17	0.00	10.25	15717.79	102.00	39.17	0.00	22.63	38081.04	482.61
	4	30.71	0.00	19.33	3610.50	3.80	30.71	0.00	10.83	3075.50	25.91	30.71	0.00	21.14	565.08	5.41
50	10	61.46	36.06	41.54	70273.62	2643.10	80.14	54.91	35.00	3994.95	3118.13	N/A	N/A	N/A	N/A	N/A
	15	49.04	26.20	39.25	53675.42	2264.92	63.05	45.26	34.19	1292.76	2747.84	N/A	N/A	N/A	N/A	N/A
	20	45.25	15.84	40.67	37965.42	1658.38	48.92	31.06	33.67	616.50	2260.36	N/A	N/A	N/A	N/A	N/A
75	10	92.33	52.83	45.33	54753.04	3474.95	116.89	63.98	41.61	1410.06	3607.95	N/A	N/A	N/A	N/A	N/A
	15	66.79	41.16	42.38	33962.46	3159.79	86.83	54.78	40.33	820.50	3223.57	N/A	N/A	N/A	N/A	N/A
	20	56.42	29.04	43.25	15910.29	2513.90	75.95	49.20	39.68	275.11	2978.10	N/A	N/A	N/A	N/A	N/A
100	10	130.58	64.23	47.62	24401.04	3607.09	155.67	71.10	44.67	495.11	3622.08	N/A	N/A	N/A	N/A	N/A
	15	95.79	53.10	47.17	22271.46	3492.85	116.06	59.15	49.38	242.00	3633.06	N/A	N/A	N/A	N/A	N/A
	20	77.42	42.79	47.21	8172.75	3043.78	102.61	56.38	46.06	62.61	3333.47	N/A	N/A	N/A	N/A	N/A
125	10	177.50	73.01	48.08	7084.33	3615.36	190.00	75.94	45.11	25.00	3651.34	N/A	N/A	N/A	N/A	N/A
	15	125.42	64.74	45.00	7167.12	3614.85	142.28	70.86	41.28	1.39	3677.52	N/A	N/A	N/A	N/A	N/A
	20	102.71	55.71	47.46	7909.33	3468.06	121.94	64.86	44.89	2.56	3724.13	N/A	N/A	N/A	N/A	N/A

3.2 Problem's complexity

The maneuver scheduling problem in the restoration of electric power distribution networks can be seen as a scheduling problem of identical parallel machines with setup times and precedence constraints. The correspondence is as follows: the teams $\ell \in \mathbb{M}$ are the machines; each switching maneuver $i \in (\mathbb{N}' \cup \mathbb{N}'')$ is a job that must be processed by some machine. The displacement times $s_{i,j,\ell}$ are the setup times required if the machine ℓ processes the job j immediately after the job i; the parameters p_i are the processing times required by each job i. Machine 0 is responsible for processing the jobs related to remotely controlled switches, with setup times $s_{i,j,0} = 0$ for all $(i, j) \in (\mathbb{N}'' \cup \{0\}) \times \mathbb{N}''$.

Using the notation usually employed for describing scheduling problems [Graham et al., 1979; Lawler et al., 1993], the problem addressed in this work can be stated as $P \mid prec, s_{i,j,k} \mid C_{max}$, in which: P indicates that there are multiple identical (in terms of processing times) parallel machines available; *prec* indicates the presence of general precedence relations among jobs; $s_{i,j,k}$ indicates the presence of setup times dependent on the machine and sequence; and C_{max} indicates that the objective is to minimize the makespan, i.e., the completion time of the last job to leave the system. This problem is known to be strongly NP-hard [Hurink and Knust, 2001; Garey and Johnson, 1979].

3.3 Related works on scheduling literature

The following subsections review some works in the scheduling literature that are related to the maneuver scheduling problem. The works are divided into two groups: the first group includes works that deal with setup times without precedence constraints, while the second group includes works that deal with both characteristics simultaneously. This review is not intended to be exhaustive, but to present how these scheduling problems are usually handled in the literature and the difficulty to solve the scheduling problems when setup times and precedence constraints are considered simultaneously.

3.3.1 Setup times only

While the study of scheduling problems dates from the mid-1950s [Allahverdi et al., 1999], works considering parallel machines with sequence-dependent setup times have appeared only in the beginning of the 1990s [Guinet, 1993; Ovacik and Uzhoy, 1993; França et al., 1996]. Most scheduling literature ignores setup times [Allahverdi, 2015]. In fact, there are contexts in which setup times can be ignored or aggregated into the processing times – e.g., setup times non-dependent on the sequence. However, there are many industrial and service applications in which ignoring setup times impacts negatively into solution quality [Allahverdi et al., 2008]. This is what happens when considering only the number of maneuvers performed in a restoration plan or its weighted variants.

Comprehensive surveys on scheduling problems with setup times can be found in the works of Allahverdi et al. [1999], Allahverdi et al. [2008] and Allahverdi [2015]. The majority of the works addressing scheduling problems in an environment with sequence-dependent setup times proposes heuristic methods since the presence of sequence-dependent setup times leads to NP-hard problems.

In the context of identical parallel machines with sequence-dependent setup times, Guinet [1993] proposes a constructive heuristic based on the Hungarian method to minimize the makespan whereas Gendreau et al. [2001] propose a heuristic based on a divide and merge method to construct feasible solutions. Although constructive heuristics usually demands little time to return a solution, they have a limited potential in finding solutions of good quality when the size of the problem increases. In this case, heuristics based on local search are usually employed to find better solutions. França et al. [1996] employ a heuristic based on Tabu Search [Glover, 1986; Gendreau and Potvin, 2010] to solve this problem. In the work of Tahar et al. [2006], the makespan is also minimized, and a heuristic based on linear programming is used as solution strategy. Lee and Pinedo [1997] also solve this problem, but they minimize the sum of the weighted tardiness of the jobs through a heuristic based on Simulated Annealing [Kirkpatrick et al., 1983; Nikolaev and Jacobson, 2010].

On unrelated parallel machines environments with sequence-dependent setup times, Logendran et al. [2007] minimize the total weighted tardiness through heuristics based on Tabu Search and evaluate four constructive heuristics based on different scheduling rules. Kim et al. [2003] propose a heuristic based on Simulated Annealing with six different neighborhood movements to the same problem. In the work of Kim and Shin [2003], the authors also propose a Tabu Search-based heuristics, but to minimize the maximum lateness. In Chen [2005], Chen [2006] and Chen and Wu [2006], the authors take into account some additional resource constraints and minimize, respectively, the makespan, maximum tardiness and total tardiness. These three works use a hybrid heuristic that combines the Guided Local Search heuristic [Voudouris et al., 2010] with Simulated Annealing and tabu lists.

The works above consider setup times dependent on the sequence of the jobs only. However, in some contexts, it is necessary to consider the dependency on the machines as well. For this context, Rabadi et al. [2006] solve the unrelated parallel machines problem with setup times in which they use a Greedy Randomized Adaptive Search Procedure (GRASP) [Feo and Resende, 1989, 1995] to minimize the makespan. de Paula et al. [2007] also solve a very similar problem in which the objective is to minimize the makespan plus the sum of the total weighted tardiness. As solution strategy, the authors propose a Variable Neighborhood Search (VNS) [Hansen et al., 2010] and compare its performance against three versions of GRASP. In the work of Vallada and Ruiz [2011], the authors use a Genetic Algorithm [Holland, 1992] combined with local search procedures to solve the problem, which outperformed the GRASP proposed by Rabadi et al. [2006]. Avalos-Rosales et al. [2015] propose a hybrid heuristic in which a GRASP rules the main framework, and a Variable Neighborhood Descent (VND) [Hansen et al., 2010] algorithm is used as improving procedure, outperforming the Genetic Algorithm of Vallada and Ruiz [2011]. The authors also introduced new makespan linearizations for MIP formulations which allowed to solve instances larger than those previously optimally solved. Muller et al. [2014] also use MIP to solve the problem, but different from Avalos-Rosales et al. [2015], they use MIP to define and explore neighborhoods. Recently, Santos et al. [2019] evaluate different heuristics and propose a heuristic based on Simulated Annealing with pruned neighborhoods, which performed better than all previous algorithms.

3.3.2 Setup times and precedence constraints simultaneously

Scheduling problems that present setup times or precedence constraints, but not both simultaneously, can be solved by list scheduling algorithms, i.e., there is an ordering among the jobs such that when an assignment rule is applied, the optimal solution is found [Schutten, 1996]. However, it does not hold when both setup times and precedence constraints are considered simultaneously [Hurink and Knust, 2001].

Regarding scheduling problems that consider sequence-dependent setup times and precedence constraints simultaneously, the literature is more restricted. One of the first works to jointly consider sequence-dependent setup times and precedence constraints is due to Kusiak and Finke [1987], in which the authors proposed a flow-based MIP formulation and branch-and-bound algorithm to minimize the makespan, but the authors considered a single machine environment. He and Kusiak [1992] also propose a MIP formulation for the problem with a reduced number of variables and constraints. Besides, a constructive heuristic is proposed by the authors. Uzsoy et al. [1991] also consider the single machine environment but, unlike Kusiak and Finke [1987] and He and Kusiak [1992], they aim at minimizing the maximum lateness. For solving the problem, the authors proposed a branch-and-bound algorithm.

Hurink and Knust [2001] addressed the identical parallel machine with sequence-dependent setup times and precedence constraints with the objective of minimizing the makespan. They show that this problem is strongly NP-hard and a dominant set of schedules, i.e., a set of schedules containing at least one optimal schedule, is very unlikely to be calculated efficiently with list scheduling techniques. Driessel and Mönch [2009] and Driessel and Mönch [2011] also addressed the parallel machine scheduling with sequence-dependent setup times and precedence constraints but, instead of minimizing the makespan, their work focus on the minimization of the total weighted tardiness. The authors evaluate different versions of the VNS as a solution strategy. Gacias et al. [2010] solves a similar problem, but consider two versions of the problem. In one the maximum lateness is minimized, while the other minimizes the sum of completion times. For both versions, the authors employ a branch-and-bound algorithm and a heuristic based on the Climbing Discrepancy Search [Milano and Roli, 2002].

3.3.3 Overview of the related works

Tables 3.2 and Table 3.3 summarize the works cited above. They present the works in terms of machine environment, constraints, objective function and solution strategy and/or main contribution of the work. Although the works consider different objective functions and some additional constraints, they share relevant features to the present work, which is the sequence-dependent setup times and precedence constraints.

Notice that the majority of the works employ heuristic algorithms to solve the problems since exact approaches become prohibitive to large-scale (and even for some medium-scale) instances. This is due not only to the fact that the problems belong to the NP-hard class, but exact approaches usually tend to return poor quality solutions for large and medium instances when runtime is limited, even if allowed to run for a few hours.

Table 3.2: Summary of the cited works that deal with setup times only.

Machines*	Reference	Constraints**	Objective	Solution strategy / Contribution
P	Guinet [1993]	s_{ij}	C_{max}	Heuristic based on the Hungarian method
	Gendreau et al. [2001]	s_{ij}	C_{max}	Divide and Merge heuristic
	França et al. [1996]	s_{ij}	C_{max}	Tabu Search
	Tahar et al. [2006]	s_{ij}	C_{max}	Linear programming based heuristic
	Lee and Pinedo [1997]	s_{ij}	$\sum w_i T_i$	Simulated Annealing
R	Logendran et al. [2007]	s_{ij}	$\sum w_i T_i$	Tabu Search, Scheduling Rules
	Kim et al. [2003]	s_{ij}	$\sum w_i T_i$	Simulated Annealing
	Kim and Shin [2003]	s_{ij}	$\max L_i$	Tabu Search
	Chen [2005][1]	s_{ij}	C_{max}	GLS + SA + Tabu List
	Chen [2006][2]	s_{ij}	$\max T_i$	GLS + SA + Tabu List
	Chen and Wu [2006][2]	s_{ij}	$\sum T_i$	GLS + SA + Tabu List
	Rabadi et al. [2006]	s_{ijk}	C_{max}	GRASP
	de Paula et al. [2007]	s_{ijk}	$C_{max} + \sum w_i T_i$	VNS, GRASP
	Vallada and Ruiz [2011]	s_{ijk}	C_{max}	Genetic Algorithm
	Avalos-Rosales et al. [2015]	s_{ijk}	C_{max}	GRASP + VND, MIP
	Muller et al. [2014]	s_{ijk}	C_{max}	Local Search + sub-MIP
	Santos et al. [2019]	s_{ijk}	C_{max}	Simulated Annealing

* P means identical parallel machines; and R means unrelated parallel machines.
** s_{ij} means sequence-dependent setup times; and s_{ijk} means machine and sequence-dependent setup times.
[1] Consider additional resource constraints and machine eligibility.
[2] Consider additional resource constraints.

Table 3.3: Summary of the cited works that deal with setup times and precedence constrains simultaneously.

Machines*	Reference	Constraints**	Objective	Solution strategy / Contribution
1	Kusiak and Finke [1987]	s_{ij}, prec	C_{max}	MIP + B&B
	He and Kusiak [1992]	s_{ij}, prec	C_{max}	MIP, Constructive heuristic
	Uzsoy et al. [1991]	s_{ij}, prec	$\max L_i$	B&B
P	Hurink and Knust [2001]	s_{ij}, prec	C_{max}	Theoretical results, NP-completeness
	Driessel and Mönch [2009]	s_{ij}, prec	$\sum w_i T_i$	VNS
	Driessel and Mönch [2011]	s_{ij}, prec	$\sum w_i T_i$	VNS
	Gacias et al. [2010]	s_{ij}, prec	$\max L_i$, $\sum C_i$	B&B, Climbing Discrepancy Search

* P means identical parallel machines; and R means unrelated parallel machines.

** s_{ij} means sequence-dependent setup times; and "prec" indicates the presence of precedence constraints.

Chapter 4

Proposed algorithms

Before presenting the proposed heuristics to the maneuver scheduling problem, we introduce the structure used to encode candidate solutions to this problem as well as the method used to evaluate the objective function and feasibility of the solutions. After that, the greedy heuristics are detailed, followed by the ILS-based heuristic.

4.1 Solution encoding

A candidate solution is encoded as a set of schedules, $\Pi = \{\Pi_0, \Pi_1, \dots, \Pi_m\}$, in which a schedule $\Pi_\ell = \langle \pi_\ell^1, \pi_\ell^2, \dots, \pi_\ell^{|\Pi_\ell|} \rangle$ is a sequence of switches assigned to team ℓ and $\pi_\ell^r \in (\mathbb{N} \setminus \{0\})$ is the r-th switch maneuvered by team ℓ. An example of a set of switching maneuvers, a set of precedence relations, and a possible candidate solution under this encoding is shown in Figure 4.1. Jointly with maintenance teams 1 to m, an additional sequence Π_0 is used for assigning the remotely controlled switches, e.g., switch 5 in Figure 4.1.

$$\underbrace{(1,O)\ \ (2,O)\ \ (3,C)\ \ (4,O)\ \ (5,O)\ \ (6,C)}_{\text{Set of switching maneuvers}}$$

$$\underbrace{\mathbb{P} = \{(1,3),\ (2,3),\ (3,6),\ (4,6),\ (5,6)\}}_{\text{Set of precedence relations}}$$

(a) Set of maneuvers to perform of a restoration plan and its precedence relations.

$$\begin{aligned}
\Pi_0 &= \langle\ \ 5\ \ \rangle \\
\Pi_1 &= \langle\ \ 2,\ \ 3,\ \ 4\ \ \rangle \\
\Pi_2 &= \langle\ \ 1,\ \ 6\ \ \rangle
\end{aligned}$$

(b) A candidate solution.

Figure 4.1: Example of a solution with the proposed encoding. Team 1 maneuvers switches 2, 3, and 4, and team 2 maneuvers 1 and 6. The maneuver of the remotely controlled switch (switch 5) is assigned to sequence Π_0.

4.2 Objective function and feasibility evaluation

To evaluate a solution, i.e., calculate its makespan, the moments t_i at which the maneuvers are performed must be calculated. Let $\mathbb{P}_j = \{i \ : \ (i,j) \in \mathbb{P}\}$ be the set of switches that have to be maneuvered before j, and $\overline{\mathbb{P}}_j = \{i \ : \ (j,i) \in \mathbb{P}\}$ the set of switches that can be maneuvered only after j is maneuvered. Algorithm 4.1 describes the evaluation procedure of a given solution.

First, the data used by this procedure is initialized (lines 1–9). The flag *feasible* (used to track the feasibility of the solution) is initialized with *true*; Δ is a counter for the number of switches with maneuvering time already calculated, initialized as 0; t_0 is the initial moment in which the work of the teams start, then it is initialized as 0; t_j is the maneuvering moment of switch j, initially set as infinity; γ_j is the number of switches with undefined maneuvering moments that must be maneuvered before j; φ_ℓ is the current position of team ℓ, initially set as 0, i.e., its initial position; and σ_ℓ is the index of the next switch to calculate the maneuvering time in schedule Π_ℓ, starting at 1.

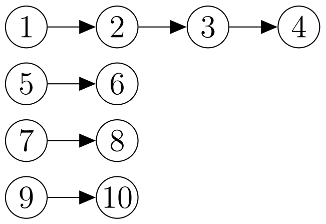
After initializing the data, the schedule of each team is evaluated switch by switch until all maneuvering times t_j are calculated or an infeasibility is detected (lines 10–26). The variable δ is initialized as 0 (line 11), which counts the number of maneuvering moments (t_j) calculated at the current iteration. The inner loop (lines 12–23) tries, for each team $\ell \in \mathbb{M}$, to calculate the maneuvering time (lines 15–18) of the σ-th switch in Π_ℓ, referenced as j, which is done only if $\gamma_j = 0$, i.e., the maneuvering moments of all switches that must precede j were already calculated. After that, the counters of pending precedents are updated (line 19), the makespan is updated (line 20), the current position of the team is updated (line 21), the index of the next switch to calculate the maneuvering time is incremented (line 22), and the counter of maneuvering times calculated at the current iteration is incremented. At the end of the iteration, the counter of maneuvering times calculated so far is updated (line 24). Besides, the feasibility is also by verifying if at least one maneuvering time was calculated at the iteration, i.e., $\delta > 0$. If the solution is unfeasible (*delta* is equal to zero), then the flag *feasible* is set to *false* (line 26) and the makespan is set to infinity (line 27).

The procedure presented at Algorithm 4.1 has a computational complexity of $O(n^2)$, as the precedence constraints have to be verified for each maneuver. Besides calculating the makespan, it also checks the solution feasibility, returning a makespan equal to infinity for unfeasible solutions. Unfeasible solutions are those in which (i) a sequence of maneuvers assigned to a team does not satisfy the precedence constraints;

Algorithm 4.1: Procedure for solution evaluation and feasibility check.

```
    Input: $\Pi$, $\mathbb{N}$, $\mathbb{N}'$, $\mathbb{N}''$, $\mathbb{M}$, $\mathbb{P}_j$, $\overline{\mathbb{P}}_j$, $p_i$, $c_{i,j,\ell}$
    Output: $C_{max}$, $t_i$

    // Initialization
 1  $feasible \leftarrow true$;
 2  $\Delta \leftarrow 0$;
 3  $t_0 \leftarrow 0$;
 4  foreach $j \in \mathbb{N}' \cup \mathbb{N}''$ do
 5      $t_j \leftarrow \infty$;
 6      $\gamma_j \leftarrow |\mathbb{P}_j|$;
 7  foreach $\ell \in \mathbb{M}$ do
 8      $\varphi_\ell \leftarrow 0$;
 9      $\sigma_\ell \leftarrow 1$;

    // Compute the operation moments and makespan
10  while $\Delta < |\mathbb{N}'| + |\mathbb{N}''| \wedge feasible$ do
11      $\delta \leftarrow 0$;
12      for $\ell \in \mathbb{M} : \sigma_\ell \leq |\Pi_\ell|$ do
13          Let $j$ be the $\sigma_\ell$-th switch in $\Pi_\ell$;
14          if $\gamma_j = 0$ then
                // Moment in which $j$ is operated
15              if $\ell = 0$ then
16                  $t_j \leftarrow \max_{k \in \mathbb{P}_j}\{t_k + p_k\}$;
17              else
18                  $t_j \leftarrow \max\left\{t_{\varphi_\ell} + p_{\varphi_\ell} + c_{\varphi_\ell,j,\ell},\ \max_{k \in \mathbb{P}_j}\{t_k + p_k\}\right\}$;

                // Update the counters of pending precedents
19              foreach $k \in \overline{\mathbb{P}}_j$ do $\gamma_k \leftarrow \gamma_k - 1$ ;

                // Update the makespan
20              $C_{max} \leftarrow \max\{C_{max},\ t_j + p_j\}$;

                // Update the other algorithm's data
21              $\varphi_\ell \leftarrow j$;
22              $\sigma_\ell \leftarrow \sigma_\ell + 1$;
23              $\delta \leftarrow \delta + 1$;

        // Check feasibility
24      $\Delta \leftarrow \Delta + \delta$;
25      if $\delta = 0$ then
26          $feasible \leftarrow false$;
27          $C_{max} \leftarrow \infty$;
```

or (ii) the maneuver of a switch i assigned to a team ℓ_1 depends on the completion of the maneuver of a switch j assigned to another team ℓ_2, but to maneuver j, team ℓ_2 has to wait for the maneuvering of k assigned to team ℓ_1 that appearing after i in the sequence Π_{ℓ_1}, which results in a deadlock. An example of an unfeasible solution is shown in Figure 4.2.


(a) Precedence graph

$$\begin{aligned} \Pi_0 &= \langle \quad \rangle \\ \Pi_1 &= \langle\ 1,\ 5,\ \mathbf{10},\ 6,\ \mathbf{7}\ \rangle \\ \Pi_2 &= \langle\ 2,\ 3,\ \mathbf{8},\ 4,\ \mathbf{9}\ \rangle \end{aligned}$$

(b) Infeasible solution

Figure 4.2: Example of an unfeasible solution. Team 1 must wait for team 2 to maneuver switch 9, before it can operate switch 10. However, team 2 has to maneuver switches 8 and 4 before 9, and for maneuvering switch 8 it has to wait for team 1 to maneuver 7, which results in a deadlock.

In scheduling problems with precedence constraints, the start of some tasks can be postponed without increasing the makespan. These tasks are referred to as the "slack" tasks. The tasks that cannot be postponed without increase the makespan are referred to as the "critical" tasks and the set of critical tasks is referred to as "critical path" [Pinedo, 2016]. It is noteworthy that in a local search procedure, only changes that affect tasks in the "critical path" may improve the makespan. However, changes that minimize the total completion time, i.e., the sum of the completion time of the schedule of all teams, makes the solution more flexible to changes which may result in a makespan minimization, since the teams will be less overloaded. Therefore, throughout this work, when the values of objective function of two solutions are compared, a lexicographic comparison is performed in which the makespan is considered first, then the total completion time is considered.

4.3 Greedy heuristics

4.3.1 A simple greedy heuristic

The proposed greedy heuristic builds a solution iteratively. At each iteration, all remotely controlled switches with precedence constraints satisfied have their maneuvering moments calculated, and then the maneuver of a manually controlled switch with precedence constraints satisfied is assigned to a maintenance team, and its maneuvering moment is calculated. The choice of the manually controlled switch and maintenance team to assign it to follows the earliest start time rule. Once a switch is chosen, its maneuvering moment is set as the largest value between the moment the team reaches the switch location and the moment of conclusion of all switches that have to precede it. For describing the greedy heuristic in more detail, the following notations are introduced:

- $\langle \Pi_\ell,\ i \rangle$ is the concatenation of the maneuvering of switch i at the end of the sequence of maneuvers performed by the maintenance team ℓ, which results in $\langle \pi_\ell^1, \pi_\ell^2, \dots, \pi_\ell^{|\Pi_\ell|}, i \rangle$.
- $\langle\ \rangle$ is an empty sequence.

Algorithm 4.2 details the proposed greedy heuristic. First, the data used by this heuristic is initialized (lines 1–8). Sets $\mathbb{S}'$ and $\mathbb{S}''$ contain, respectively, the manually controlled and remotely controlled switches with maneuvering moments still undefined; t_j is the maneuvering moment of switch j, initially set as 0; γ_j is the number of switches with undefined maneuvering moments that must be maneuvered before j; φ_ℓ is the current position of team ℓ, initially set as 0, i.e., its initial position; and Π_ℓ is the schedule of team ℓ.

After initializing the data, the assignment of manually controlled switches and the definition of maneuvering moments, including for remotely controlled switches, are performed (lines 9–21). First, as long as $\mathbb{S}''$ contains remotely controlled switches with precedence constraints satisfied, their maneuvering moments are calculated (line 11). After determining the maneuvering moment of a switch j, the values γ_k of switches preceded by j are updated (line 12). Then, the maneuver of switch j is included at the end of the schedule Π_0 (line 13), and it is removed from $\mathbb{S}''$ (line 14).

When no more remotely controlled switches with precedence constraints satisfied are available in $\mathbb{S}''$, a manually controlled switch $j \in \mathbb{S}'$ is chosen, and its maneuvering is assigned to a maintenance team ℓ. The choice of j and ℓ (line 16) is performed according to the earliest start time rule,

$$\underset{(j,\ell)\in\mathbb{S}'\times\mathbb{M}}{\arg\min} \left\{ t_{\varphi_\ell} + p_{\varphi_\ell} + c_{\varphi_\ell,j,\ell} \ :\ \gamma_j = 0 \right\}. \tag{4.1}$$

Once j and ℓ have been chosen, the maneuvering moment t_j is defined as the maximum between the moment in which team ℓ reaches switch j and the moment in

Algorithm 4.2: Simple greedy heuristic.

```
   Input: $\mathbb{N}$, $\mathbb{N}'$, $\mathbb{N}''$, $\mathbb{M}$, $\mathbb{P}_j$, $\overline{\mathbb{P}}_j$, $p_i$, $c_{i,j,\ell}$
   Output: $\Pi$, $C_{max}$, $t_i$

   // Initialization
1  $\mathbb{S}' \leftarrow \mathbb{N}'$;
2  $\mathbb{S}'' \leftarrow \mathbb{N}''$;
3  foreach $j \in \mathbb{N}$ do
4      $t_j \leftarrow 0$;
5      $\gamma_j \leftarrow |\mathbb{P}_j|$;
6  foreach $\ell \in \mathbb{M}$ do
7      $\varphi_\ell \leftarrow 0$;
8      $\Pi_\ell \leftarrow \langle\ \rangle$;

   // Assignment and sequencing
9  while $|\mathbb{S}'| + |\mathbb{S}''| > 0$ do
       // Remotely controlled switches
10     while $\exists j \in \mathbb{S}'' : \gamma_j = 0$ do
11         $t_j \leftarrow \max\left\{0,\ \max_{i \in \mathbb{P}_j}\{t_i + p_i\}\right\}$;
12         foreach $k \in \overline{\mathbb{P}}_j$ do $\gamma_k \leftarrow \gamma_k - 1$ ;
13         $\Pi_0 \leftarrow \langle\Pi_0,\ j\rangle$;
14         $\mathbb{S}'' \leftarrow \mathbb{S}'' \setminus \{j\}$;

       // Manually controlled switch
15     if $|\mathbb{S}'| > 0$ then
16         Choose some $j \in \mathbb{S}'$ and $\ell \in \mathbb{M}$ according to Eq. (4.1);
17         $t_j \leftarrow \max\left\{t_{\varphi_\ell} + p_{\varphi_\ell} + c_{\varphi_\ell,j,\ell},\ \max_{i \in \mathbb{P}_j}\{t_i + p_i\}\right\}$;
18         foreach $k \in \overline{\mathbb{P}}_j$ do $\gamma_k \leftarrow \gamma_k - 1$ ;
19         $\Pi_\ell \leftarrow \langle\Pi_\ell,\ j\rangle$;
20         $\varphi_\ell \leftarrow j$;
21         $\mathbb{S}' \leftarrow \mathbb{S}' \setminus \{j\}$;

   // Compute the makespan
22 $C_{max} \leftarrow \max_{i \in \mathbb{N}' \cup \mathbb{N}''}\{t_i + p_i\}$;
```

which all switch operations that must precede it are completed (line 17). After that, the values γ_k of all k preceded by j are updated (line 18). The maneuver is appended at the end of Π_ℓ (line 19), the current position of team ℓ is updated (line 20) and j is removed from $\mathbb{S}'$ (line 21). After all switch operations are assigned, the makespan is computed (line 22).

The worst case complexity of this heuristic happens when no remotely controlled switches are present, i.e., when $\mathbb{N}''$ is empty. In this case, the *while* loop (lines 9–21) will loop n times, and all switch/team pairs have to be evaluated at each iteration for deciding the next assignment. This results in a worst-case complexity of $O(n^2m)$ for this heuristic.

4.3.2 NEH-based heuristic

The Nawaz-Enscore-Ham (NEH) heuristic [Nawaz et al., 1983] initially proposed for solving the Permutation Flow-Shop Scheduling Problem is one of the most efficient constructive heuristics to minimize the makespan for that problem [Kalczynski and Kamburowski, 2007]. Since then, many works have proposed adaptations of this strategy to solve other variants of scheduling problems due to its good performance as constructive heuristics.

In this section, we employ the insertion criterion of the NEH heuristic in the construction of a solution. Different from the simple greedy heuristic presented in the previous section, the insertion criterion of NEH allows switching maneuvers to be placed in positions of the schedule other than the last, on the condition that the relative order of the maneuvers already scheduled does not change. The criterion adopted to choose the next switching maneuver to schedule is similar to the one used in the previous heuristic. However, instead of evaluating the insertion at the end of the schedule only, it evaluates the insertion at all positions of the schedule of each maintenance team. Algorithm 4.3 describes the NEH-based heuristic in more details.

Algorithm 4.3: NEH-based heuristic.

```
Input: $\mathbb{N}$, $\mathbb{N}'$, $\mathbb{N}''$, $\mathbb{M}$, $\mathbb{P}_j$, $\overline{\mathbb{P}}_j$, $p_i$, $s_{i,j,\ell}$
Output: $\Pi$, $C_{max}$, $t_i$

// Initialization
1  $\mathbb{S}' \leftarrow \mathbb{N}'$;
2  $\mathbb{S}'' \leftarrow \mathbb{N}''$;
3  foreach $j \in \mathbb{N}$ do
4      $t_j \leftarrow 0$;
5      $\gamma_j \leftarrow |\mathbb{P}_j|$;

// Assignment and sequencing
6  while $|\mathbb{S}'| + |\mathbb{S}''| > 0$ do
       // Remotely controlled switches
7      while $\exists j \in \mathbb{S}'' : \gamma_j = 0$ do
8          foreach $k \in \overline{\mathbb{P}}_j$ do $\gamma_k \leftarrow \gamma_k - 1$ ;
9          $\Pi_0 \leftarrow \langle \Pi_0,\ j \rangle$;
10         $\mathbb{S}'' \leftarrow \mathbb{S}'' \setminus \{j\}$;

       // Manually controlled switch
11     if $|\mathbb{S}'| > 0$ then
12         Find $j \in \mathbb{S}'$, $\ell \in \mathbb{M}$ and $r$ with lowest value of $\Phi(\Pi, j, \ell, r)$;
13         Insert $j$ at the $r$-th position of $\Pi_\ell$;
14         foreach $k \in \overline{\mathbb{P}}_j$ do $\gamma_k \leftarrow \gamma_k - 1$ ;
15         $\mathbb{S}' \leftarrow \mathbb{S}' \setminus \{j\}$;

16 Compute $C_{max}$ and $t_j$ for all $j \in \mathbb{N}$ according to Algorithm 4.1;
```

The evaluation of the increase in the objective function with the insertion of a switching maneuver j in the r-th position of scheduling of the ℓ-th team of a partial solution Π is given by $\Phi(\Pi, j, \ell, r)$ (line 12). It can be calculated using the Algorithm 4.1 but considering the partial solution after the insertion of j at the r-th position of Π_ℓ, and the inputs $\mathbb{N}$ and $\mathbb{N}'$ containing only those switching maneuvers that are present in the partial solution Π.

This NEH-based heuristic presents a worst-case complexity of $O(n^4)$, since it has to perform the Algorithm 4.1 at each iteration for all unscheduled switch operations in all possible positions. Despite a greater complexity compared to the simple greedy heuristic of the previous section, the NEH-base heuristic is still fast enough to be used inside a load restoration method to estimate the restoration time.

4.4 ILS-based heuristics

As the size of the problems increases, the performance of constructive heuristics may deteriorate in terms of solution quality. In this case, an improving heuristic can be applied on the schedule of the final restoration plan to find a better one. For this, we present an improving heuristics based on the Iterated Local Search (ILS) metaheuristic [Baxter, 1981; Lourenço et al., 2010]. The ILS explores the space of solutions by repeatedly applying a local search method to a solution obtained by a perturbation on the incumbent one. Despite its simple structure, the ILS has been shown to be effective in solving combinatorial optimization problems, including scheduling ones [Allahverdi, 2015].

The general structure of the proposed ILS-based heuristic is presented at the flow chart in Figure 4.3 and its pseudo-code is presented in Algorithm 4.4. The ILS has as input parameters: an initial solution x (e.g. obtained by a greedy heuristic); a function $f : \mathcal{X} \mapsto \mathbb{R}$ that evaluates solutions; a set $\mathcal{N} = \{\eta_1, \ldots, \eta_{|\mathcal{N}|}\}$ of neighborhood functions; a perturbation function $\rho : \mathcal{X} \to \mathcal{X}$; and the maximum number of passes of the perturbation function, $\overline{\delta} \geq 1$.

Initially, a locally optimal solution x^* is found (line 1) and the counter δ is initialized (line 2). At any given iteration (lines 3–12), the perturbation function is applied δ times (lines 4–6). The larger the δ, the greater the perturbation of the incumbent solution tends to be. The Ejection Chain strategy [Glover, 1996] is employed as perturbation. For the problem addressed in this work, Ejection Chain transfers a maneuver assigned to a team ℓ_1 to another team ℓ_2, then a maneuver assigned to team ℓ_2 to a team ℓ_3 and so on, until all teams have transferred and received a

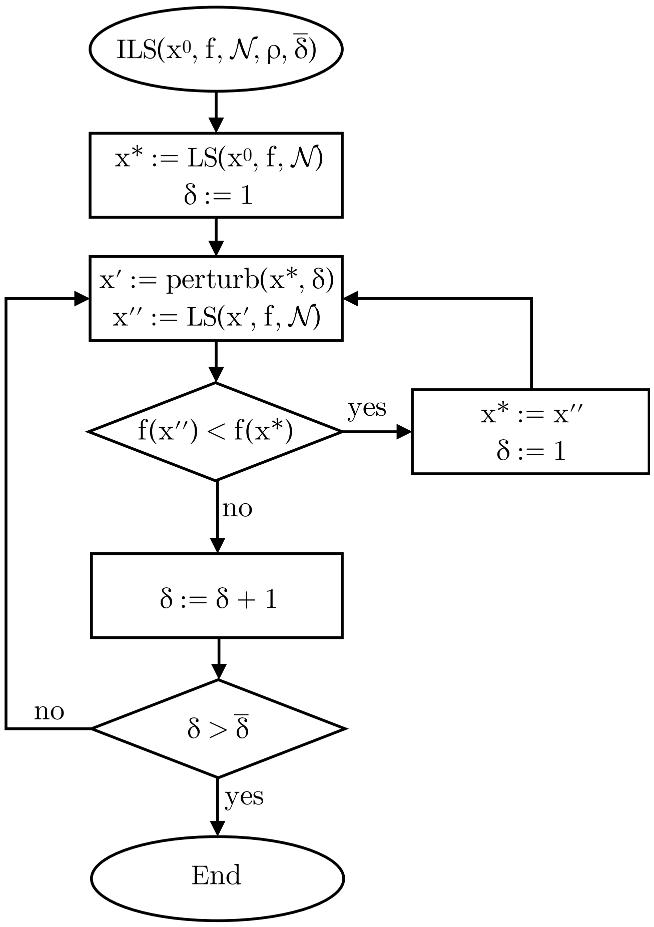

ILS(x^0, f, $\mathcal{N}$, ρ, $\bar{\delta}$)
x* := LS(x^0, f, $\mathcal{N}$)
δ := 1
x′ := perturb(x*, δ)
x″ := LS(x′, f, $\mathcal{N}$)
f(x″) < f(x*)
yes
x* := x″
δ := 1
no
δ := δ + 1
no
δ > $\bar{\delta}$
yes
End

Figure 4.3: General structure of the ILS-based heuristic.

maneuver to/from another. If a team has no switching maneuver assigned to it, then it is ignored and the process just continues from the next team. When moving a switching maneuver between teams, only moves that result in feasible solutions are considered, with Algorithm 4.1 used for checking feasibility. Once the perturbed solution x' is obtained, the local search is performed, resulting in a new local optimal solution x'' (line 7). If $f(x'') < f(x^*)$ then x'' becomes the incumbent solution (line 9) and δ is reset (line 10). Otherwise, δ is incremented (line 12). It is important to highlight that the comparison between $f(x'')$ and $f(x^*)$ is the lexicographic comparison, in which the makespan is considered first, and then the sum of completion times of all teams. This secondary criterion is important because a reduction may allow improvements in the makespan at next iterations of the ILS.

Algorithm 4.4: General structure of the proposed ILS-based heuristic.

```
Input: $x$, $f$, $\mathcal{N}$, $\rho$, $\overline{\delta}$
Output: $x^*$
// Find a first local optimal solution
1  $x^* \leftarrow$ local_search($x, \mathcal{N}$);
2  $\delta \leftarrow 1$;
3  while $\delta \leq \overline{\delta}$ do
       // Perform a perturbation in solution $x^*$
4      $x' \leftarrow x^*$;
5      for $i$ in 1 to $\delta$ do
6          $x' \leftarrow \rho(x')$
       // Perform a local search
7      $x'' \leftarrow$ local_search($x', f, \mathcal{N}$);
       // Update the best solution found
8      if $f(x'') < f(x^*)$ then
9          $x^* \leftarrow x''$;
10         $\delta \leftarrow 1$;
11     else
12         $\delta \leftarrow \delta + 1$;
```

4.4.1 Neighborhood functions and local search strategy

Let $\mathcal{X}$ be the set of all feasible solutions for the problem and $\mathcal{P}(\mathcal{X})$ the set of all possible subsets of $\mathcal{X}$, a neighborhood function is a function $\eta : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{X})$ that, for any feasible solution $x \in \mathcal{X}$, maps a set of other feasible solutions, denoted as *neighborhood*.

Five neighborhood functions were used in this work: Shift, Exchange, Reassignment, Swap, and Direct-Swap. These neighborhood functions are described below and Figure 4.4 illustrates them.

- *Shift*: the maneuver of a switch j is shifted to another position within the same schedule, as shown in Figure 4.4b, where switch 2 is shifted to the second position of the schedule. Notice that this is the only possible shift for that maneuver, since it has to be operated before the switch 6.

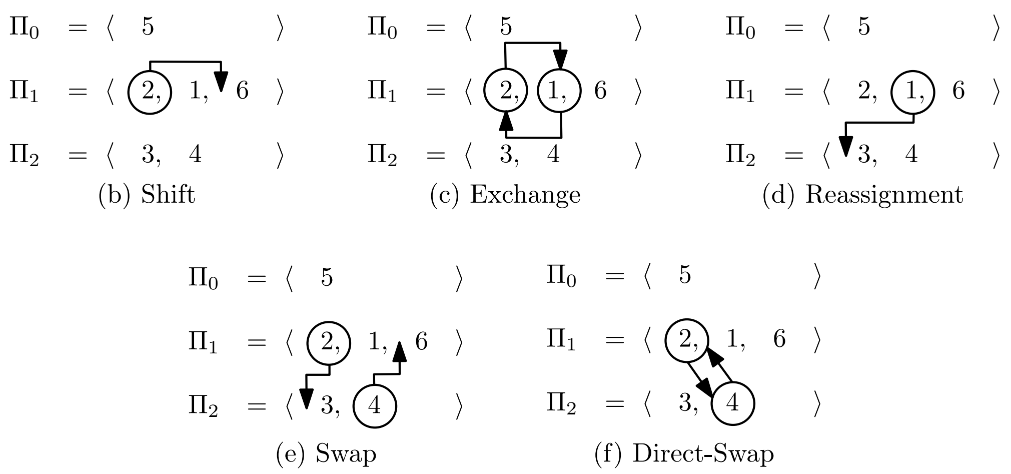
- *Exchange*: two switches i and j maneuvered by the same team ℓ have their position exchanged in the schedule. In Figure 4.4c switches 1 and 2 have their position exchanged in the schedule assigned to team 1.

- *Reassignment*: a maneuver is assigned to a different team, as shown in Figure 4.4d, where switch 1 is reassigned from team 1 to team 2. The only feasible position to place it on the schedule of the team 2 is before the maneuvering of switch 3, since the maneuvering of switch 1 must precede it.

$$\underbrace{(1,O)\ \ (2,O)\ \ (3,C)\ \ (4,O)\ \ (5,O)\ \ (6,C)}_{\text{Set of switching maneuvers}}$$

$$\underbrace{\mathbb{P} = \{(1,3),\ (2,3),\ (3,6),\ (4,6),\ (5,6)\}}_{\text{Set of precedence relations}}$$

(a) Restoration plan (with switch 5 remotely controlled) and the set of precedence rules.


$\Pi_0 = \langle\ 5\ \rangle$
$\Pi_1 = \langle\ 2,\ 1,\ 6\ \rangle$
$\Pi_2 = \langle\ 3,\ 4\ \rangle$

(b) Shift (c) Exchange (d) Reassignment (e) Swap (f) Direct-Swap

Figure 4.4: Example of movements performed by the neighborhood functions implemented for the maneuvers scheduling problem.

- *Swap*: a maneuver currently assigned to a team ℓ_1 is transferred to a team ℓ_2, with $\ell_1 \neq \ell_2$, and a maneuver currently assigned to team ℓ_2 is transferred to team ℓ_1. In Figure 4.4e, the maneuvering of switch 2, currently assigned to team 1, is transferred to team 2, and switch 4, currently assigned to team 2, is transferred to team 1.

- *Direct-Swap*: it is similar to the *Swap* function. However, the positions changed in the teams are maintained. In Figure 4.4f, the maneuvering of switches 2 and 4 currently assigned to teams 1 and 2, respectively, are swapped. The switch 2 is placed at the previous position of the switch 4 at team 2 and the switch 4 is placed at the previous position of the switch 2 at team 1.

Some characteristics of the neighborhoods are summarized in Table 4.1. The second column (*Teams*) contains the number of teams involved in the movement, the third column (*Cardinality*) shows the number of neighbor solutions depending on the number of maneuvers (n) and teams (m), and the fourth column (*Complexity*) shows the complexity of growth of the neighborhood. The values presented in columns *Cardi-*

nality and *Complexy* consider a neighborhood from a solution with maneuvers equally distributed among the teams.

Table 4.1: Summary of some characteristics of the neighborhood functions

Neighborhood	Teams	Cardinality[†]	Complexity[†]
Shift	1	$\frac{n^2 - nm}{m}$	$O(n^2)$
Exchange	1	$\frac{n^2}{m}$	$O(n^2)$
Reassignment	2	$\frac{-n^2}{m} + n^2 + nm - n$	$O(n^2)$
Swap	2	$\left(\frac{n}{m}\right)^4$	$O(n^4)$
Direct-Swap	2	$\left(\frac{n}{m}\right)^2$	$O(n^2)$

[†] n is the number of switching maneuvers and m is the number of teams available.

Regarding the maneuvering of remotely controlled switches, only *Shift* and *Exchange* functions can be applied. The other ones would result in unfeasible solutions, since they would assign a remote maneuver to a maintenance team or a manual maneuver to Π_0 (no team).

Some movements performed by the neighborhood functions may result in unfeasible solutions. However, since the resulting solutions are evaluated using Algorithm 4.1, any unfeasible solution is guaranteed to be attributed a value of infinity for its makespan. Then, the objective value of unfeasible solutions will be equal to *infinity*.

In the proposed ILS-based heuristic, local search is performed according to Variable Neighborhood Descent (VND) strategy [Hansen et al., 2010], which allows the use of different neighborhood functions. The key idea of the VND search is to escape from local minimum from a given neighborhood by iteratively alternating the neighborhood functions. Algorithm 4.5 shows the general structure of the VND search. The neighborhood functions are considered in the following sequence on the VND implemented: *Shift*, *Exchange*, *Reassignment*, *Direct-Swap* and *Swap*.

Initially, the start solution x^0 is set as the current best solution x^* (line 1), and the index i that defines which current neighborhood function is initialized as 1 (line 2). At any given iteration (lines 3–9), the best neighbor solution x' from $\eta_i(x^*)$ is found (line 4). If x' is better than the current best solution x^*, then x^* is updated (line 6) and the index i is reset (line 7). Otherwise, the index i is incremented, then the next neighborhood function will be used in the next iteration of the VND.

The VND stops when none of the neighborhood functions in $\mathcal{N}$ can produce a solution better than the current best solution x^*. Then, the solution x^* is a local minimum for all neighborhoods defined by the functions $\eta_i \in \mathcal{N}$.

Algorithm 4.5: General structure of the VND search.

```
Input: $x^0$, $f$, $\mathcal{N} = \{\eta_1, \eta_2, \ldots, \eta_k\}$
Output: $x^*$

1 $x^* \leftarrow x^0$;
2 $i \leftarrow 1$;
3 while $i < |\mathcal{N}|$ do
4     $x' \leftarrow \arg\min_{x \in \eta_i(x^*)} f(x)$;
5     if $f(x') < f(x^*)$ then
6         $x^* \leftarrow x'$;
7         $i \leftarrow 1$;
8     else
9         $i \leftarrow i + 1$;
```

Chapter 5

Computational experiments

5.1 Experimental design

The computational experiments are divided into two parts. The first evaluates the performance of the proposed algorithms in terms of solution quality and running time. The running time is important due to the time constraints imposed by the context of load restoration problems, as discussed earlier in Chapter 2. The second part of the experiment evaluates the contribution in the final restoration plan (or set of restoration plans) when considering the proposed approach incorporated into a restoration strategy to obtain a proper estimation of time.

For the first part of the experiment, a set of random instances with different dimensions and characteristics were considered. Initially, each instance was modeled as a MIP problem (using the MIP formulation proposed in Section 3.1) and solved with Gurobi, limited to 1 hour of runtime. The values of linear relaxation, objective function of the incumbent solution, runtime, number of MIP nodes explored and gap of duality were recorded. These values were used as reference to compare the quality of solutions returned by the proposed heuristics.

After that, each instance was solved with the two greedy heuristics proposed in Section 4.3. Their performance in terms of value of objective function and runtime were compared. The greedy heuristic with the best performance was used to generate the initial solution for the ILS-based heuristic. Then, each instance was solved 10 times with the ILS-based heuristic considering different random seeds for each run. The running time and value of the objective function of the solution returned were recorded.

For the second part of the experiment, three scenarios of fault with different levels of severeness, based on a real network, were constructed. The candidate restoration

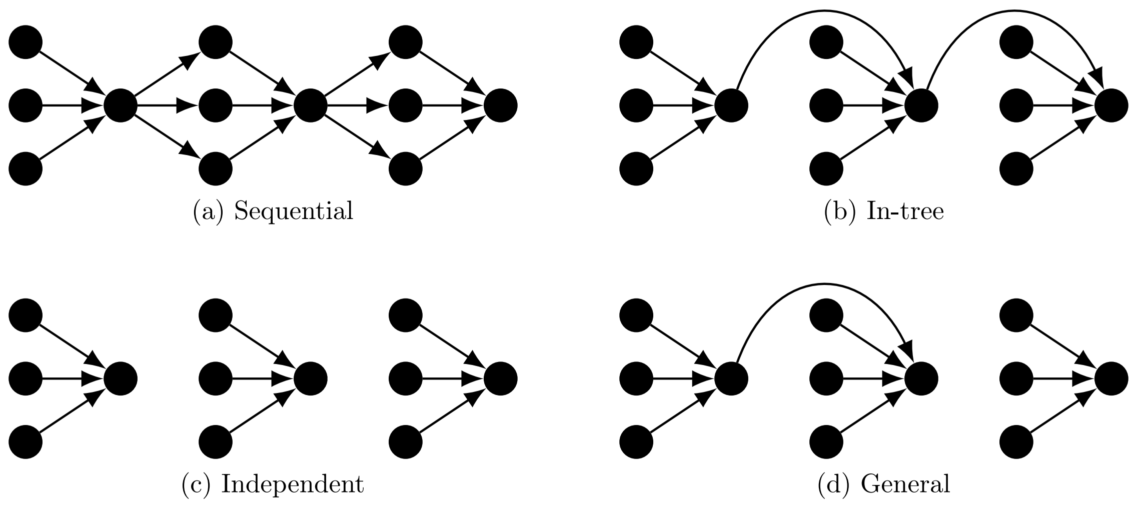


Figure 5.1: Schemes of precedence graph.

plans returned using the proposed approach were compared against those returned when considering only the number of maneuvers as proxy of time.

5.2 Test instances

A set of 576 random instances of different sizes were created to serve as a benchmark set to perform the first part of the experiment that evaluates the performance of the algorithms. The set of instances is divided into two groups: small and large instances. In the group of small instances, the number of maneuvers varied as $n \in \{6, 8, 10, 12\}$ and, for each value of n, the number of teams varied as $m \in \{2, 3, 4\}$. For the group of large instances, the number of maneuvers varied as $n \in \{50, 75, 100, 125\}$ and, for each value of n, the number of teams varied as $m \in \{10, 15, 20\}$. The times of operation p_i for the manually controlled switches were randomly generated from a uniform distribution $U(1, 4)$ and, for the remotely controlled ones, these times were set to 1. For all instances, 10% of the switches were considered as remotely controlled. The displacement times $s_{i,j,\ell}$ were randomly generated from a uniform distribution $U(7, 14)$ and then corrected to satisfy the weak triangular inequality using a modified Floyd-Warshall algorithm [Rocha et al., 2008]. For all parameters, it was considered integer values only.

For each pair (n, m), four schemes of precedence graph were considered: sequential, in-tree, independent, and general. Examples of these four schemes are given in Figure 5.1. For each configuration (n, m, precedence graph), three instances were generated using different seeds for the random number generator.

The naming convention used for the instances was

`ORCS-<n>-<m>-<prec>-<s>-<seed-id>`, in which: `<n>` is the number of maneuvers; `<m>` is the number of maintenance teams; `<prec>` is the scheme of the precedence graph (S for sequential, T for in-tree, I for independent, and G for general); `<s>` is the number of stages (for precedence graphs of type S, T or I) or the desired density, also referred as order of strength [Vanhoucke et al., 2008] (for precedence graphs of type G); and `<seed-id>` is an ID for the seed used to initialize the random number generator of the instance generator script.

5.3 Implementation details and benchmark machine

The proposed heuristics were implemented in C++17. The code was compiled using GNU Compiler Collection (GCC) version 8.1.1 with compiler optimization flag set to "-O2". The MIP formulation was solved with Gurobi 8.0.1 [Gurobi Optimization, Inc., 2017] through its API for C++ using default settings, but limited to a single thread. All experiments were performed on a dual 2.10 GHz Intel(R) Xeon(R) Silver 4116 machine with 156 GiB of main memory running Fedora 28 (64-bits).

The source code of the MIP formulation and heuristics, the instance files and the script used to create them, the scripts of the experiments performed, and the data obtained from the experiments are available for download at `https://github.com/andremaravilha/phd`.

5.4 Performance of the proposed algorithms

Table 5.1 shows the results obtained by Gurobi for the set of instances using the proposed MIP formulation (3.1)–(3.13) as well as the results obtained by the simple greedy heuristic and NEH-based greedy heuristic. The results in this table are summarized as mean values for each instance size. Columns n and m are, respectively, the number of maneuvers and the number of teams; column *"MIP Obj."* contains the mean values of objective function obtained with Gurobi (limited to 1 hour of runtime); the next columns contain the results obtained by the greedy heuristics, in which *"Obj"* and *"Time (s)"* columns contain, respectively, the mean values of objective function, and the mean values of runtime in seconds. Notice that a value equal to zero in column *"Time (s)"* means a runtime smaller than 0.0001 seconds.

For small instances, i.e., $n \in \{6, 8, 10, 12\}$, Gurobi was able to find the proven optimal solution in a few seconds for all of the instances. However, even a few seconds

Table 5.1: Results obtained by solution of the MIP formulation using Gurobi solver and greedy heuristics. The results presented in this table are the mean values for each instance size.

			NEH-based Greedy Heur.		Simple Greedy Heur.	
n	m	MIP Obj.[†]	Obj	Time(s)[‡]	Obj	Time(s)[‡]
6	2	33.42	35.21	0.0000	34.00	0.0000
	3	25.67	27.71	0.0000	27.42	0.0000
	4	21.29	23.67	0.0000	23.62	0.0000
8	2	43.42	47.67	0.0000	48.08	0.0000
	3	31.21	34.42	0.0000	34.71	0.0000
	4	23.62	25.21	0.0000	25.71	0.0000
10	2	51.79	56.62	0.0000	57.08	0.0000
	3	35.29	38.92	0.0000	40.00	0.0000
	4	29.25	33.04	0.0000	32.17	0.0000
12	2	55.88	60.25	0.0000	61.12	0.0000
	3	39.17	44.04	0.0000	44.38	0.0000
	4	30.71	35.12	0.0000	35.46	0.0000
50	10	61.46	65.38	0.0101	65.88	0.0000
	15	49.04	52.38	0.0110	52.04	0.0000
	20	45.25	49.29	0.0137	48.25	0.0000
75	10	92.33	88.21	0.0347	86.54	0.0000
	15	66.79	68.00	0.0468	66.04	0.0000
	20	56.42	59.58	0.0539	58.08	0.0000
100	10	130.58	110.46	0.1176	108.67	0.0000
	15	95.79	83.67	0.1393	83.67	0.0000
	20	77.42	72.12	0.1589	69.83	0.0000
125	10	177.50	133.71	0.2808	131.46	0.0000
	15	125.42	95.83	0.3500	93.29	0.0000
	20	102.71	83.38	0.4056	81.58	0.0000

[†] For most instances with $n \geq 50$ Gurobi was not able to find the proven optimal solution within the time limit of 1 hour. [‡] Values equal to 0.0000 mean a runtime smaller than 0.0001 seconds.

is prohibitive to a method that intends to be embedded into a restoration algorithm. For large instances, i.e, $n \in \{50, 75, 100, 125\}$, Gurobi was not able to find the proven optimal solution within the time limit of 1 hour for all instances.

Regarding the instances with $n = 50$, Gurobi performed better than the greedy heuristics in terms of mean value of objective function, but it took a much longer time than the heuristics. Despite not being suitable as solution strategy for the context of load restoration, the results obtained with Gurobi are used as reference to evaluate the performance of the proposed heuristics. Notice that the greedy heuristics, in addition to run faster than Gurobi, achieved better results in terms of objective function for the instances with $n \geq 75$. These results reinforce the choice of heuristics rather than

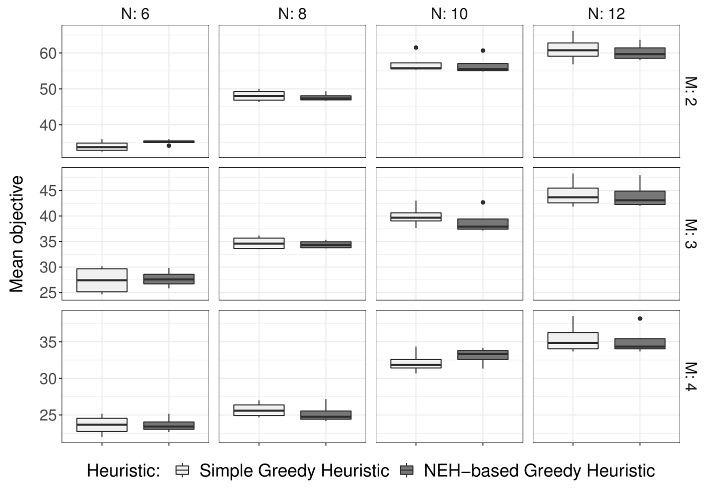


Figure 5.2: Mean value of objective function obtained for small instances. The results are stratified by heuristic and instance size.

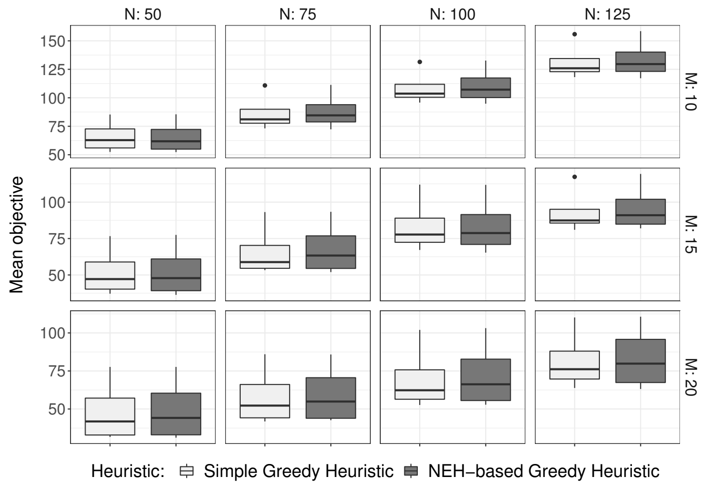


Figure 5.3: Mean value of objective function obtained for large instances. The results are stratified by heuristic and instance size.

exact approaches for this problem.

In addition to the results presented on the table above, Figures 5.2 and 5.3 show boxplot graphs for the mean values of objective function of the solutions returned by the greedy heuristics. The results are stratified by instance size.

From the results presented in the Table 5.1 and Figures 5.2 and 5.3, it can be observed that both NEH-based greedy heuristic and the simple greedy heuristic present similar performance in terms of mean value of objective function of the solutions returned by them. However, the simple greedy heuristic outperforms the NEH-based greedy heuristic in terms of runtime, mainly for the set of large instances, which is evident in the boxplot presented in Figure 5.4.

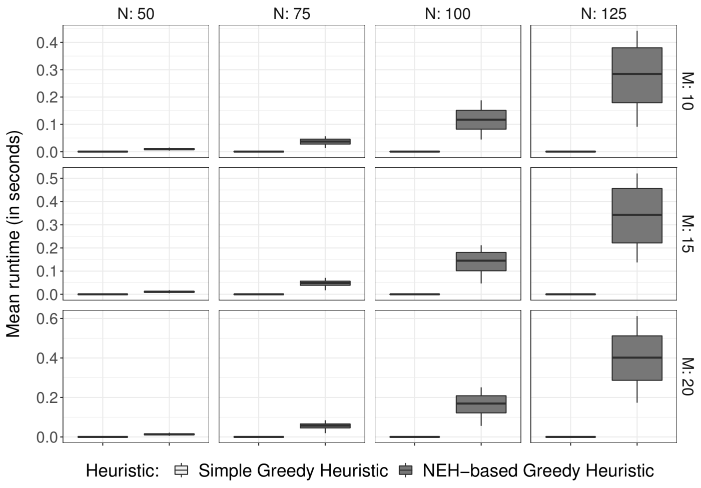


Figure 5.4: Mean value of runtime (in seconds) by heuristic and instance size. As the number of maneuvers increase, the NEH-based Greedy Heuristic presents a significant increase in runtime, while the Simple Greedy Heuristic continues to demand a low runtime (less than 0.0001s) for all problem sizes. The results for small instances were omitted in this figure because they were very small (less than 0.0001s) for both Simple Greedy Heuristic and NEH-based Greedy Heuristic.

It was already expected that the NEH-based greedy heuristic would demand a longer time than the simple greedy heuristic, since its insertion criterion leads to a greater worst-case complexity. However, this additional effort did not result in a significant improvement to the quality of the solutions. From these results, we can assume the simple greedy heuristic is the better choice as a constructive heuristic for the maneuver scheduling problem to be embedded into a restoration algorithm to estimate the time required to perform restoration plans, and therefore it was also used as strategy to generate initial solutions to the ILS-based heuristic.

Table 5.2 shows the results obtained with the ILS-based heuristic. Columns "n" and "m" contain, respectively, the number of maneuvers and the number teams available; column *"Start Obj."* contains the mean values of objective function of the start solutions used by the ILS; column *"Final Obj."* contains the mean values of objective function of final solutions returned by the ILS; column *"Improv (%)"* is the average percent improvement by the ILS over the start solution; and column *"Time (s)"* contains the mean values of runtime in seconds.

Focusing on instances in which Gurobi was not able to find the proven optimal solution (i.e., on the instances with 50 maneuvers or more), it is possible to observe that the ILS found better solutions than Gurobi for several instances. Also, even for instances in which ILS was not able to match or outperform Gurobi, the gap was generally small enough for practical purposes. Furthermore, it is important to

Table 5.2: Results obtained by the ILS-based heuristic. The results presented are the mean values for each size of problem.

n	m	MIP	Start Obj.	Final Obj.	Improv (%)	Time (s)
6	2	33.42	34.00	33.58	1.16	0.0000
	3	25.67	27.42	25.96	5.08	0.0001
	4	21.29	23.62	21.46	8.42	0.0003
8	2	43.42	48.08	44.21	7.67	0.0007
	3	31.21	34.71	31.79	7.98	0.0010
	4	23.62	25.71	24.38	4.95	0.0007
10	2	51.79	57.08	52.88	7.19	0.0026
	3	35.29	40.00	36.17	9.46	0.0026
	4	29.25	32.17	30.42	5.20	0.0019
12	2	55.88	61.12	56.58	7.32	0.0044
	3	39.17	44.38	40.71	8.12	0.0041
	4	30.71	35.46	33.21	6.09	0.0035
50	10	61.46	65.88	61.67	6.78	0.6679
	15	49.04	52.04	50.12	4.73	0.6587
	20	45.25	48.25	46.58	4.49	0.7118
75	10	92.33	86.54	82.62	4.63	4.1344
	15	66.79	66.04	62.88	5.43	2.7165
	20	56.42	58.08	56.33	3.44	2.9667
100	10	130.58	108.67	104.29	4.05	13.4255
	15	95.79	83.67	79.83	4.89	9.8188
	20	77.42	69.83	67.54	3.65	7.5887
125	10	177.50	131.46	126.17	3.98	56.5374
	15	125.42	93.29	89.96	3.59	20.9732
	20	102.71	81.58	78.62	3.76	19.1502

emphasize that the mean runtime of the ILS was from a few milliseconds to under 57 seconds, compared to the 1 hour allowed to Gurobi.

It is interesting to highlight that the proposed ILS-based heuristic reached the same solution in all runs for each instance. This behavior, combined with the gap between the objective function of the final solution returned by the ILS and Gurobi, suggests the ILS as a robust strategy in terms of the variability of quality of the solutions returned to be used in a real environment.

The results reinforce the choice of heuristics rather than exact approaches for this problem, allowing us to draw tentative conclusions regarding their performance. However, the precise advantages of using them as part of the restoration planning process remain to be investigated. In the next section three case studies, based on a real network, are used to evaluate the gains when considering the maneuver scheduling problem to estimate the time of performing a restoration plan when compared to the usual strategy of considering the number of maneuvers as a proxy of time.

5.5 Performance analysis in network restoration

As explained in Chapter 2, the SAIDI index gets worse as more loads stay out of service. The time to perform the maneuvers $\mathcal{I}_T$ should, therefore, be an indicator to be minimized. However, current literature tends to focus on minimizing only the number of maneuvers $\mathcal{I}_N$, which is easier to compute but far less accurate. In this section, we compare these two approaches in different scenarios and show how an apparently good result with respect to $\mathcal{I}_N$ could cause needless increases in SAIDI. For that, we adopt a system from a Brazilian distribution utility with 5 feeders, 703 buses, and 132 switches (see Figure 5.5). Loads were estimated according to average consumption at peak hours on a weekday, and the load flow parameters (voltage and current) for each new configuration were computed using a matrix-based Forward-Backward sweep [Lisboa et al., 2014].

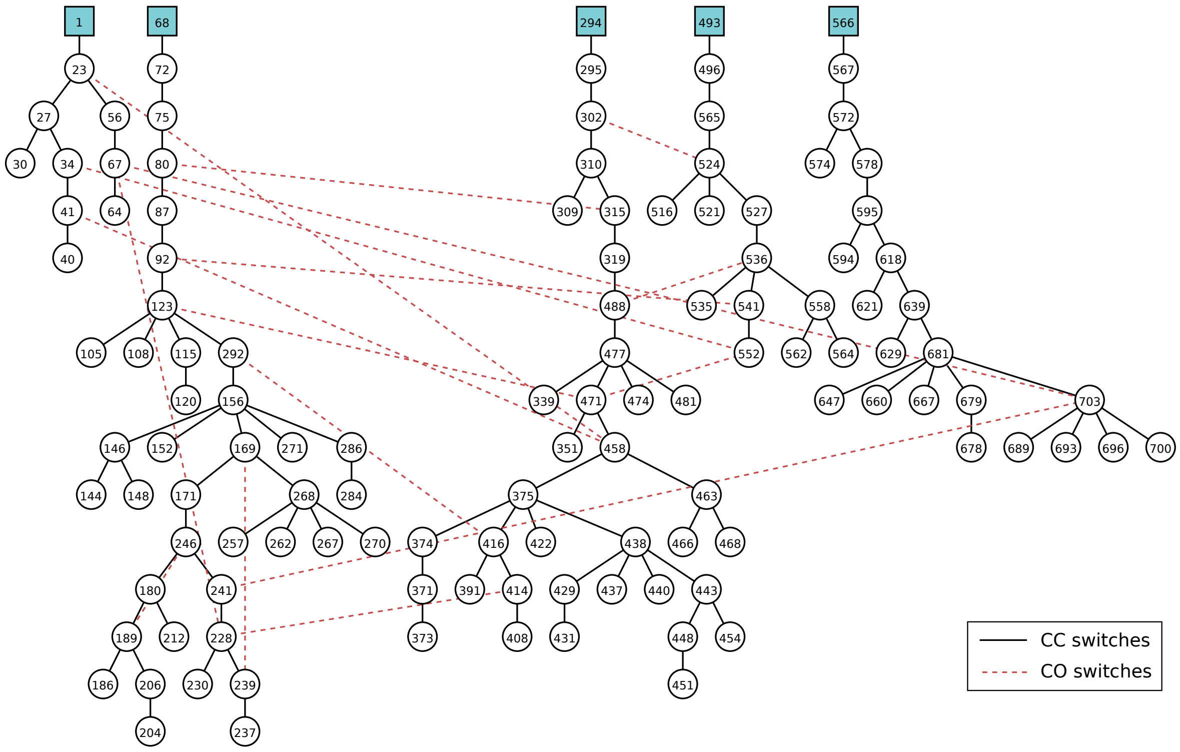


Figure 5.5: Distribution system used as basis for the computational experiments. For simplicity of presentation only the nodes are numbered, and we refer to a switch by the pair of nodes it connects.

We compare two different formulations for the load restoration problem: $\mathbf{f}^{(1)} = [\mathcal{I}_{S_{NR}}, \mathcal{I}_N]$ and $\mathbf{f}^{(2)} = [\mathcal{I}_{S_{NR}}, \mathcal{I}_T]$. Both are multi-objective, with the first objective to minimize the power not restored, while the second being either the number or time of maneuvers. The outcome in both cases is a set of Pareto-optimal solutions which is computed here by enumeration[1] with each stage limited to a maximum of five maneuvers (up to four openings, including an isolation switch, followed by on closing) which we believe is a large enough number to represent real-world cases. Only restoration plans that result in feasible configurations were considered to be Pareto-optimal.

The proposed procedure of this work is to perform the optimization using the greedy heuristic (as its processing time does not hinder the process), and then employ the ILS to improve the estimation in the returned solutions. This approach is used here with the formulations $\mathbf{f}^{(1)}$ and $\mathbf{f}^{(2)}$. In what follows, we consider three different scenarios of faults to compare both methodologies.

5.5.1 Scenario 1: a single fault at a source

This first scenario consists of a single fault at a source node, but it can be considered serious since it leaves a whole feeder out of service. The source 493 was arbitrarily chosen to illustrate this scenario. A simplified view is shown in Figure 5.6. Despite its severeness, we can recover the whole *oos* region with an isolation maneuver and the closing of any CO switch shown in Figure 5.6, so for the problem considering $\mathcal{I}_N$ all of these possibilities are deemed optimal candidates, as shown in Table 5.3.

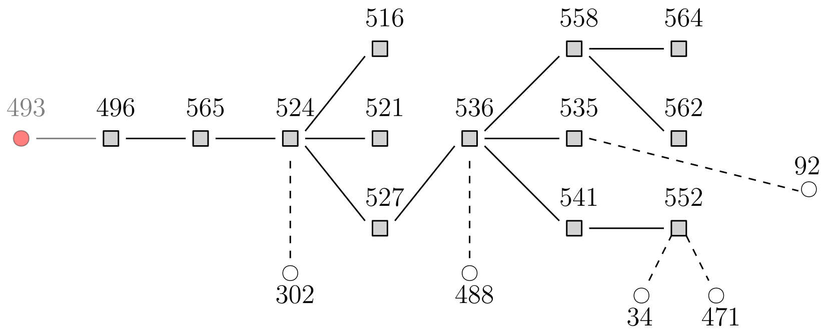


Figure 5.6: Scenario 1: Fault at the source 493.

The problem with the approach which uses $\mathcal{I}_N$ becomes clear in this case: it considers all five solutions described in Table 5.3 as equally good, when in reality some

[1]It is appropriate to highlight that this approach should not be recommended due to the elevate processing time to compute the set of Pareto-optimal solutions. Since the focus on this work is in the implications of disregarding the actual time of maneuvers, this is not of concern here. In a practical context, a heuristic would be more adequate.

Table 5.3: Optimal sequences of maneuvers obtained for Scenario 1 by each objective formulation. The post-fault conditions (equivalent to a no maneuver solution) are shown for comparison. $\mathcal{I}_T^{ILS}$ is the estimation returned by the ILS. Both time indices are given in an appropriate time unit.

f_2	**Sequence of maneuvers**	$\mathcal{I}_{S_{NR}}$ (pu)	$\mathcal{I}_N$	$\mathcal{I}_T$	$\mathcal{I}_T^{ILS}$
Post-fault	**Open:** {}; **Close:** ()	0.0185	0	0.000	0.000
$\mathcal{I}_N$	**Open:** {(493,496)}; **Close:** (302,524)	0.0000	2	25.758	25.758
	Open: {(493,496)}; **Close:** (488,536)	0.0000	2	81.607	81.607
	Open: {(493,496)}; **Close:** (92,541)	0.0000	2	148.947	148.947
	Open: {(493,496)}; **Close:** (471,552)	0.0000	2	161.563	161.563
	Open: {(493,496)}; **Close:** (34,552)	0.0000	2	164.065	164.065
$\mathcal{I}_T$	**Open:** {(493,496)}; **Close:** (302,524)	0.0000	2	25.758	25.758

maneuvers require less time to be performed, conditional on the initial positions of the teams. The utility may experience an unnecessarily large contribution to SAIDI if the decision maker chooses a plan that takes longer. In contrast, considering $\mathcal{I}_T$, only the fastest plan is returned as optimal, which minimizes economic penalties due to longer disconnections. Notice in this case that the time estimations provided by the heuristic and the ILS are equal due to the small size of the solutions.

5.5.2 Scenario 2: simultaneous faults on multiple non-source nodes

The second scenario consists of simultaneous faults at non-source nodes. Albeit less severe, each fault leaves fewer loads disconnected. Such a scenario can be quite common in distribution networks with a predominance of overhead lines, particularly in urban regions with large tree canopy coverage, or in regions subject to seasonal rainstorms – all of which are quite common in developing countries. Nodes 27, 536 and 681 were arbitrarily chosen to illustrate this scenario.

In this scenario the enumeration of possible restoration plans returned 38 non-dominated solutions for $\mathbf{f}^{(1)}$ and 27 for $\mathbf{f}^{(2)}$. Figure 5.7 shows the optimal solutions according to both approaches, with respect to $\mathcal{I}_{S_{NR}}$ and $\mathcal{I}_T^{ILS}$ (using the ILS) to allow for comparisons.

A few important points are evident from this figure. First, there are many different sequences that yield the same objective values, as shown as parentheses in the figure. Second, all possible sequences that recover the most power were found using both approaches. However, for the sub-sequences with larger $\mathcal{I}_{S_{NR}}$ (which demand less maneuvers), the total time taken to implement these plans is basically the same (notice all points returned for $\mathbf{f}^{(1)}$ define a horizontal line), meaning that the $\mathbf{f}^{(1)}$ con-

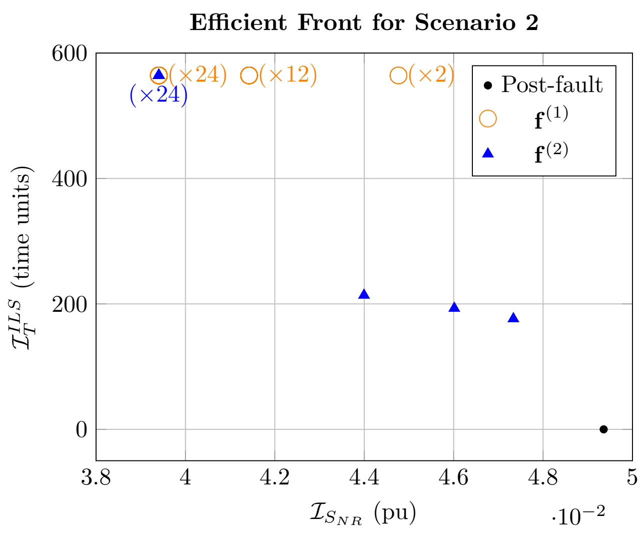


Figure 5.7: Results for scenario 2. The number of different sequences of maneuvers that recover the same $\mathcal{I}_{S_{NR}}$ and demand the same time $\mathcal{I}_T^{ILS}$ is shown in parentheses. If unspecified, this number is equal to 1.

cept cannot reach solutions with lower times. On the contrary, the approach using time directly, instead of the number of maneuvers, found three other restoration plans that may recover fewer loads but can actually be executed in a considerably shorter time, providing the decision maker with reasonable alternatives from which the desired course of action can be selected.

Finally, we would like to remark that the final decision of the plan to be executed, among the non-dominated ones, is left to the operator. This decision should be taken considering factors such as distribution utility policies, accumulated SAIDI before and after the maneuvers, expected financial penalties, current network conditions, plan risk, and operator experience. It is important to emphasize that if a decision criterion was defined a priori (e.g. accumulated SAIDI), this criterion could be adopted as the objective function of the optimization algorithms, without the need of a multiobjective approach. However, this kind of strategy is often unfeasible, since decision criteria tend to vary depending on the feeders affected by the fault and the current scenario of the network.

5.5.3 Scenario 3: simultaneous faults on multiple sources

This is the most severe case of failure, which leaves multiple feeders out of service. Keep in mind that there can be more than one feeder per substation, so sometimes a single failure can cause such a serious scenario. To illustrate this scenario, the sources 68 and 294 were arbitrarily chosen. Given its large dimension, complete enumeration would demand an unreasonably long time (a few months) to finish. We, therefore, decided to prune nodes from the enumeration tree that resulted in dominated solutions. This procedure corresponds more closely to what would be performed in practice. Even with this pruning strategy, the final number of solutions is too large to be shown in a tabular form, so a graphical representation is shown in Figure 5.8. There were 35 and 66 solutions returned by the $\mathbf{f}^{(1)}$ and $\mathbf{f}^{(2)}$ approaches, respectively.

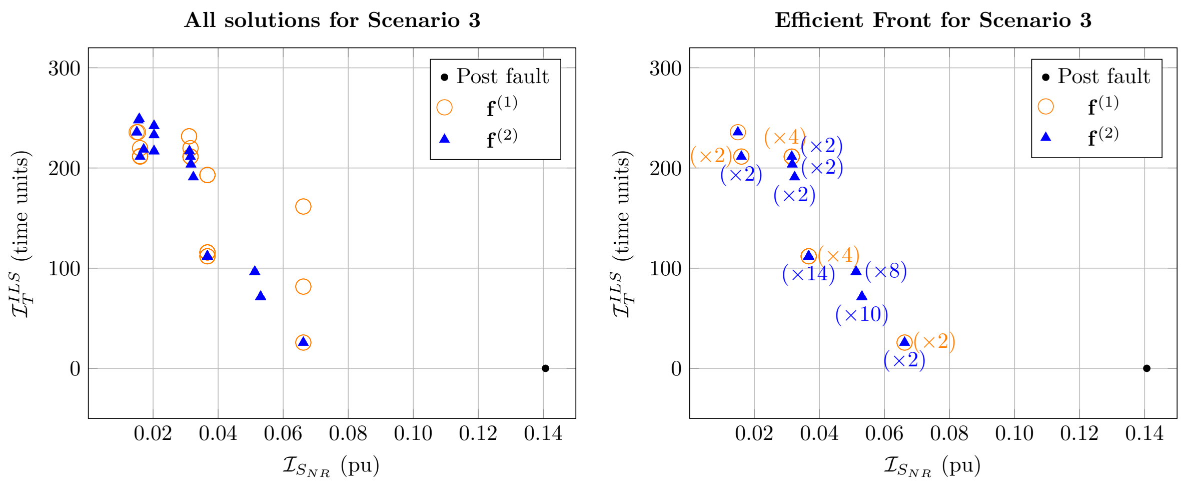


Figure 5.8: Results for scenario 3. **Left:** all solutions returned by both approaches. **Right:** only the non-dominated sequences.

As seen in the left panel, both approaches returned some dominated solutions. This was already mentioned in previous scenarios for $\mathbf{f}^{(1)}$ and, in this case, because some solutions had more maneuvers to be executed, the heuristic used in $\mathbf{f}^{(2)}$ possibly overestimated the actual total maneuver time. Thus, in this scenario, the unnecessary contribution to SAIDI could happen with both approaches. However, if the complete procedure as proposed here is followed, after the execution of the ILS in each solution and the removal of the non-dominated ones, the total number of alternatives reduces to 13 with $\mathbf{f}^{(1)}$ and 43 with $\mathbf{f}^{(2)}$, as shown in the right panel of Figure 5.8. There are two observations that can be made in this case. First, only about 37% of the sequences are not dominated when considering only the number of maneuvers, contrasting with 65% when using the time heuristic. Second, even by looking at effectively different solutions, some alternatives are only found with the $\mathbf{f}^{(2)}$, which could represent more promising trade-offs to the decision maker but which would remain unattainable if the time is not used directly as objective.

Chapter 6

Final remarks

6.1 Overall conclusions

In the literature of load restoration, some works recognize the importance of considering a more reliable estimation of the time required to perform the maneuvers of a restoration plan. However, most of them consider the number of maneuvers as a proxy of time. The works that try to improve this estimation define a fixed time for each switch to be maneuvered, which becomes nothing more than a weighted version of the number of maneuvers, with the weights bearing no physical interpretation of time. Since it is necessary to assign switching maneuvers to teams and sequence them to obtain a proper estimation of time, this is not as simple as the usual approach that considers only the number of maneuvers. Moreover, the approach adopted to estimate the time required to perform the maneuvers need to be fast, since the process of defining a restoration plan is already time-consuming due to repeatedly calculations of power flow. These may be some of the reasons why previous works have not considered a more reliable estimation so far.

In this work, we proposed the estimation of maneuvering time for the restoration of electric power distribution networks through the solution of a scheduling problem. The scheduling problem that needs to be solved is equivalent to the parallel machine scheduling problem with sequence-dependent setup times and precedence constraints, which is an NP-hard problem. To allow the proposed approach to be used in practice, some efficient heuristics, in terms of runtime and solution quality, were also proposed. The use of the proposed formulation and heuristics avoids the issues that arise when employing the usual approach in the literature, namely that of using the number of maneuvers as a proxy of time.

The proposed formulation that considers the estimation of time to perform the restoration plan was compared against the usual approach that considers the number of maneuvers when formulating a plan in different fault scenarios with different levels of severeness, based on a real network. The results of this comparison show that the main drawback of using the number of maneuvers is that it tends to regard inferior solutions (in terms of time) as if they were equally good. Thus, this procedure may deceive the dispatch engineer into executing a plan that causes additional increases in SAIDI, which could otherwise be prevented with the proposed technique. Moreover, some alternatives may be inaccessible if only the number of maneuvers is considered, as shown in Scenarios 2 and 3. This setback would be even more evident if the Pareto-front was approximated with high-level heuristics (such as multi-objective Variable Neighborhood Search), which are more common in practice. These methods usually incorporate built-in diversity mechanisms that prevent solutions that are equivalent in terms of objective values. In this case, among the sub-sequences with the same value of $\mathcal{I}_{S_{NR}}$, only one would remain in the final non-dominated set. With the $\mathbf{f}^{(2)}$ approach, this would not be a problem since solutions are equivalent if they take the same time, but with the $\mathbf{f}^{(1)}$ concept we would keep or lose the fastest plan by blind chance. These results should provide enough reason for adopting a time-based formulation to avoid these pitfalls.

6.2 Further works

Scenarios that require 12 maneuvers or more, like the ones used in the experiments, can be very large compared to what commonly happens in practice, but the experiments performed with large instances were used to evaluate how the proposed heuristics scale when the size of the problem increases.

Although scenarios with a large number of maneuvers are unlikely in the context of load restoration, other similar scheduling problems that need to consider setup times and precedence constraints can be found in the area of electrical engineering, among others. As an example, we can cite the weekly scheduling of maintenance and services in the distribution network, which can easily require tens to hundreds of jobs to be performed by maintenance teams. Thus the proposed heuristics can be extended to solve these problems as well, without much additional effort, since they scale well with problem size.

Other aspects that can be investigated in future works are: the inclusion of stochastic parameters, such as travel times, which may change due to traffic conditions or other events; and a non-fixed number of available teams, such as the possibility of requesting outsourced teams to speed up the restoration process.

Future works can also investigate the behavior of neighborhood functions for the maneuver scheduling problem. Since the objective of the problem is to minimize the makespan, this objective function has a fitness landscape with many plateaus, which may difficult local search methods escape from these regions.

Bibliography

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345–378.

Allahverdi, A., Gupta, J. N. D., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239.

Allahverdi, A., Ng, C. T., Cheng, T. C. E., and Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032.

Aneel (2008). *Atlas de energia elétrica do Brasil*. Aneel, 3 edition. Available at http://www.aneel.gov.br. Accessed on September 20, 2018.

Aoki, K., Nara, K., Itoh, M., Satoh, T., and Kuwabara, H. (1989). A new algorithm for service restoration in distribution systems. *IEEE Transactions on Power Delivery*, 4(3):1832–1839.

Avalos-Rosales, O., Angel-Bello, F., and Alvarez, A. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *International Journal of Advanced Manufacturing Technology*, 76:1705–1718.

Baxter, J. (1981). Local optima avoidance in depot location. *Journal of the Operational Research Society*, 32:815–819.

Borges, T. T., Carneiro, S., Garcia, P. A. N., and Pereira, J. L. R. (2016). A new OPF based distribution system restoration method. *International Journal of Electrical Power and Energy Systems*, 80:297–305.

Camillo, M. H. M., Fanucchi, R. Z., Romero, M. E. V., de Lima, T. W., da Silva Soares, A., Delbem, A. C. B., Marques, L. T., Maciel, C. D., and London, J. B. A. (2016). Combining exhaustive search and multi-objective evolutionary algorithm for service restoration in large-scale distribution systems. *Electric Power Systems Research*, 134:1–8.

Carrano, E. G. (2007). *Algoritmos evolucionários eficientes para otimização de redes.* PhD thesis, Universidade Federal de Minas Gerais.

Carrano, E. G., Silva, G. P., Cardoso, E. P., and Takahashi, R. H. C. (2016). Subpermutation-based evolutionary multiobjective algorithm for load restoration in power distribution networks. *IEEE Transactions on Evolutionary Computation*, 20(4):546–562.

Carvalho, P., Ferreira, L., and Barruncho, L. (2007). Optimization approach to dynamic restoration of distribution systems. *International Journal of Electrical Power & Energy Systems*, 29(3):222–229.

Chen, C. S., Lin, C. H., and Tsai, H. Y. (2002). A rule-based expert system with colored Petri net models for distribution system service restoration. *IEEE Transactions on Power Systems*, 17(4):1073–1080.

Chen, J.-F. (2005). Unrelated parallel machine scheduling with secondary resource constraints. *International Journal of Advanced Manufacturing Technology*, 26:285–292.

Chen, J.-F. (2006). Minimization of maximum tardiness on unrelated parallel machines with process restrictions and setups. *International Journal of Advanced Manufacturing Technology*, 29(5-6):557–563.

Chen, J.-F. and Wu, T.-H. (2006). Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. *Omega*, 34(1):81–89.

de Paula, M. R., Ravetti, M. G., Mateus, G. R., and Pardalos, P. M. (2007). Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighborhood search. *IMA Journal of Management Mathematics*, 18(2):101–115.

Dimitrijevic, S. and Rajakovic, N. (2011). An innovative approach for solving the restoration problem in distribution networks. *Electric Power Systems Research*, 81(10):1961–1972.

Driessel, R. and Mönch, L. (2009). Scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times using variable neighborhood search. In *International Conference on Computers & Industrial Engineering, CIE'09*, pages 273–278.

Driessel, R. and Mönch, L. (2011). Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times. *Computers & Industrial Engineering*, 61(2):336–345.

Dyer, M. E. and Wolsey, L. A. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, 26:255–270.

Feo, T. A. and Resende, M. G. C. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.

França, P. M., Gendreau, M., Laporte, G., and Müller, F. M. (1996). A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, 43(2-3):79–89.

Gacias, B., Artigues, C., and Lopez, P. (2010). Parallel machine scheduling with precedence constraints and setup times. *Computers & Operations Research*, 37(12):2141–2151.

Garcia, V. J. and França, P. M. (2008). Multiobjective service restoration in electric distribution networks using a local search based heuristic. *European Journal of Operational Research*, 189(3):694–705.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman & Co.

Gendreau, M., Laporte, G., and Guimarães, E. M. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 133(1):183–189.

Gendreau, M. and Potvin, J.-Y. (2010). Tabu search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of metaheuristics*, chapter 2, pages 41–59. Springer, 2 edition.

Gholami, M., Moshtagh, J., and Ghadernejad, N. (2015a). Service restoration in distribution networks using combination of two heuristic methods considering load shedding. *Journal of Modern Power Systems and Clean Energy*, 3(4):556–564.

Gholami, M., Moshtagh, J., and Rashidi, L. (2015b). Service restoration for unbalanced distribution networks using a combination two heuristic methods. *Electric Power and Energy Systems*, 67:222–229.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

Glover, F. (1996). Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65(1-3):223–253.

Goulart, F., Maravilha, A. L., Carrano, E. G., and Campelo, F. (2018). Permutation-based optimization for the load restoration problem with improved time estimation of maneuvers. *International Journal of Electrical Power & Energy Systems*, 101:339–355.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

Guinet, A. (1993). Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria. *International Journal of Production Research*, 31(7):1579–1594.

Gurobi Optimization, Inc. (2017). Gurobi Optimizer Reference Manual. http://www.gurobi.com.

Hansen, P., Mladenović, N., Brimberg, J., and Pérez, J. A. M. (2010). Variable neighborhood search. chapter 3, pages 61–86.

He, W. and Kusiak, A. (1992). Scheduling manufacturing systems. *Computers in Industry*, 20(2):163–175.

Hijazi, H. and Thiébaux, S. (2015). Optimal distribution systems reconfiguration for radial and meshed grids. *International Journal of Electrical Power and Energy Systems*, 72:136–143.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. A Bradford Book.

Hsu, Y. Y. and Huang, H. M. (1995). Distribution system service restoration using the artificial neural network approach and pattern recognition method. *IEE Proceedings - Generation, Transmission and Distribution*, 142(3):251–256.

Hsu, Y. Y., Huang, H. M., Kuo, H. C., Peng, S. K., Chang, C. W., Chang, K. J., Yu, H. S., Chow, C. E., and Kuo, R. T. (1992). Distribution system service restoration using a heuristic search approach. *IEEE Transactions on Power Delivery*, 7(2):734–740.

Hurink, J. and Knust, S. (2001). List scheduling in a parallel machine environment with precedence constraints and setup times. *Operations Research Letters*, 29(5):231–239.

IEEE (2012). IEEE Std 1366-2012 (Revision of IEEE Std 1366-2003) - IEEE guide for electric power distribution reliability indices.

Kalczynski, P. J. and Kamburowski, J. (2007). On the NEH heuristic for minimizing the makespan in permutation flow shops. *Omega*, 35(1):53–60.

Kim, C. O. and Shin, H. J. (2003). Scheduling jobs on parallel machines: a restricted tabu search approach. *The Intenational Journal of Advanced Manufacturing Technology*, 22(3-4):278–287.

Kim, D.-W., Na, D.-G., and Chen, F. F. (2003). Unrelated parallel machine scheduling with setup times and total weighted tardiness objective. *Robotics and Computer-Integrated Manufacturing*, 19(1-2):173–181.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kumar, Y., Das, B., and Sharma, J. (2006a). Genetic algorithm for supply restoration in distribution system with priority customers. In *International Conference on Probabilistic Methods Applied to Power Systems, PMAPS'06*.

Kumar, Y., Das, B., and Sharma, J. (2006b). Service restoration in distribution system using non-dominated sorting genetic algorithm. *Electric Power Systems Research*, 76(9-10):768–777.

Kusiak, A. and Finke, G. (1987). Modeling and solving the flexible forging module scheduling problem. *Engineering Optimization*, 12(1):1–12.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H., and Shmoys, D. B. (1993). Sequencing and scheduling: algorithms and complexity. In *Handbooks in Operations Research and Management Science*, volume 4, chapter 9, pages 445–522.

Lee, Y. H. and Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100(3):464–474.

Lisboa, A. C., Guedes, L. S. M., Vieira, D. A. G., and Saldanha, R. R. (2014). A fast power flow method for radial networks with linear storage and no matrix inversions. *International Journal of Electrical Power and Energy Systems*, 63:901–907.

Logendran, R., McDonell, B., and Smucker, B. (2007). Scheduling unrelated parallel machines with sequence-dependent setups. *Computers & Operations Research*, 34(11):3420–3438.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2010). Iterated local search: framework and applications. In *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, volume 146, chapter 12, pages 363–397. Springer.

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, 8(2):219–223.

Maravilha, A. L., Goulart, F., Carrano, E. G., and Campelo, F. (2018). Scheduling maneuvers for the restoration of electric power distribution networks: formulation and heuristics. *Electric Power Systems Research*, 163 Part A:301–309.

Marques, L. T., Delbem, A. C. B., and London, J. B. A. (2018). Service restoration with prioritization of customers and switches and determination of switching sequence. *IEEE Transactions on Smart Grid*, 9(3):2359–2370.

Milano, M. and Roli, A. (2002). On the relation between complete and incomplete search: an informal discussion. In *International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR'02*, pages 237–250.

Mohammadi, F. and Afrakhteh, H. (2012). Optimal load restoration in distribution network using intentional islanding. *Journal of Electrical Engineering*, 12(4):108–113.

Muller, F. M., Araújo, O. B., Stefanello, F., and Zanetti, M. (2014). MIP-based neighborhood search for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *Revista de Administração da UFSM*, 7(3):506–523.

Nawaz, M., Enscore, E. E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95.

Nikolaev, A. G. and Jacobson, S. H. (2010). Simulated annealing. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of metaheuristics*, chapter 1, pages 1–59. Springer, 2 edition.

Ovacik, I. M. and Uzhoy, R. (1993). Worst-case error bounds for parallel machine scheduling problems with bounded sequence-dependent setup times. *Operations Research Letters*, 14(5):251–256.

Pessoa, A., Uchoa, E., de Aragão, M. P., and Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2(3-4):259–290.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, and Systems.* Springer, 5 edition.

Rabadi, G., Morgan, R. J., and Al-Salem, A. (2006). Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, 17(1):85–97.

Rocha, P. L., Ravetti, M. G., Mateus, G. R., and Pardalos, P. M. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research*, 35(4):1250–1264.

Romero, R., Franco, J. F., Leitão, F. B., Rider, M. J., and Souza, E. S. (2016). A new mathematical model for the restoration problem in balanced radial distribution system. *IEEE Transactions on Power Systems*, 31(2):1259–1268.

Sanches, D. S., London, J. B. A., Delbem, A. C. B., Prado, R. S., Guimarães, F. G., Neto, O. M., and Lima, T. W. (2014). Multiobjective evolutionary algorithm with a discrete differential mutation operator developed for service restoration in distribution systems. *International Journal of Electrical Power and Energy Systems*, 62:700–711.

Santos, H. G., Toffolo, T. A. M., Silva, C. L. T. F., and Berghe, G. V. (2019). Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *International Transactions in Operational Research*, 26(2):707–724. First published: 20 June 2016.

Schutten, J. M. J. (1996). List scheduling revisited. *Operations Research Letters*, 18(4):167–170.

Short, T. (2014). *Electric Power Distribution Handbook*. CRC Press, 2 edition.

Tahar, D. N., Yalaoui, F., Chu, C., and Amodeo, L. (2006). A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times. *International Journal of Production Economics*, 99(1-2):63–73.

Uzsoy, R., Martin-Vega, L. A., Lee, C.-Y., and Leonard, P. A. (1991). Production scheduling algorithms for a semiconductor test facility. *IEEE Transactions on Semiconductor Manufacturing*, 4(4):270–280.

Vallada, E. and Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211(3):612–622.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., and Tavares, L. V. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187(2):511–524.

Voudouris, C., Tsang, E. P. K., and Alsheddy, A. (2010). Guided local search. chapter 11, pages 321–361.

Watanabe, I. (2005). An ACO algorithm for service restoration in power distribution systems. In *IEEE Congress on Evolutionary Computation, CEC'05*, pages 2864–2871.

Wolsey, L. A. and Nemhauser, G. L. (1999). *Integer and Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley.

Yadav, M., Gupta, C., Kumar, S. V., and Hasan, I. (2014). Optimal coordination of directional overcurrent relay: a review. *International Journal of Emerging Technology and Advanced Engineering*, 4(5):859–862.

Appendix A

Alternative mixed integer programming formulations

There are different strategies for writing scheduling problems as mixed integer programming problems. They mainly differ on the constraints used to calculate the moment in which each job starts. One of the most common strategies is based on disjunctive constraints with big-M values. However, it is well known that the use of big-M tends to lead to weak formulations, i.e., the gap between the values of objective function of the optimal solution of the linear relaxation and the complete formulation is large. On the other hand, arc-time-indexed formulations usually provide better linear relaxations, but lead to a rapid increase in the number of variables and constraints, which makes prohibitive the use of default branch-and-cut strategies to solve them.

In the next subsections, we present two alternative mixed integer formulations for the maneuver scheduling problem. The first formulation uses big-M values for the disjunctive constraints, while the second formulation is based on arc-time-indexed variables.

A.1 Precedence variables

In this formulation based on binary precedence variables [Manne, 1960], the sequence of maneuver operations performed by each team is modeled as a flow starting at node 0 (the team's initial position) that goes through nodes of switch maneuvers. These flows determine the ordering in which each team must perform the switch maneuvers assigned to them.

Given the decision variables:

$$x_{i,j,\ell} = \begin{cases} 1, & \text{if } i \in \mathbb{N}' \cup \{0\} \text{ and } j \in \mathbb{N}' \text{ are maneuvered by team } \ell \in \mathbb{M} \text{, with } i \\ & \text{maneuvered immediately before } j; \\ 0, & \text{otherwise.} \end{cases}$$

$t_i \geq 0,$ the moment in which the maneuver of $i \in \mathbb{N} \setminus \{0\}$ is performed;

$C_{max} \geq 0,$ the moment in which all maneuvers are completed (makespan);

the maneuver scheduling problem in the restoration of electric power distribution networks can be formulated as the following MIP problem:

$$\begin{aligned}
\text{Min. } & C_{max} && && \text{(A.1)} \\
\text{s.t.: } & \sum_{j \in \mathbb{N}'} x_{0,j,\ell} \leq 1 && \forall \ell \in \mathbb{M} && \text{(A.2)} \\
& \sum_{\substack{i \in \mathbb{N}' \cup \{0\}: \\ i \neq j}} \sum_{\ell \in \mathbb{M}} x_{i,j,\ell} = 1 && \forall j \in \mathbb{N}' && \text{(A.3)} \\
& \sum_{\substack{j \in \mathbb{N}': \\ j \neq i}} \sum_{\ell \in \mathbb{M}} x_{i,j,\ell} \leq 1 && \forall i \in \mathbb{N}' && \text{(A.4)} \\
& \sum_{\substack{h \in \mathbb{N}' \cup \{0\}: \\ h \neq i,\ h \neq j}} x_{h,i,\ell} \geq x_{i,j,\ell} && \forall i,j \in \mathbb{N}';\ \forall \ell \in \mathbb{M};\ i \neq j && \text{(A.5)} \\
& t_j \geq s_{0,j,\ell} - \mathcal{M}(1 - x_{0,j,\ell}) && \forall j \in \mathbb{N}';\ \forall \ell \in \mathbb{M} && \text{(A.6)} \\
& t_j \geq t_i + p_i + s_{i,j,\ell} - \mathcal{M}(1 - x_{i,j,\ell}) && \forall i,j \in \mathbb{N}';\ \forall \ell \in \mathbb{M};\ i \neq j && \text{(A.7)} \\
& t_j \geq t_i + p_i && \forall (i,j) \in \mathbb{P} && \text{(A.8)} \\
& C_{max} \geq t_i + p_i && \forall i \in \mathbb{N} \setminus \{0\} && \text{(A.9)} \\
& x_{i,j,\ell} \in \{0,1\} && \begin{gathered} \forall i \in \mathbb{N}' \cup \{0\};\ \forall j \in \mathbb{N}'; \\ \forall \ell \in \mathbb{M};\ i \neq j \end{gathered} && \text{(A.10)} \\
& t_i \geq 0 && \forall i \in \mathbb{N} \setminus \{0\} && \text{(A.11)} \\
& C_{max} \geq 0 && && \text{(A.12)}
\end{aligned}$$

in which the objective (A.1) is the minimization of the makespan, subject to the constraints:

- Each team has to start their work from its initial location. Then, constraints (A.2) ensure that for each $\ell \in \mathbb{M}$, the sum of all $x_{0,j,\ell}$, for $j \in \mathbb{N}'$, has to be 1 if the team has some maneuver assigned to it, or 0 otherwise.

- Each maneuver of a manually controlled switch has to be performed by a single maintenance team. Constraints (A.3) ensure for each $j \in \mathbb{N}'$, the sum of all $x_{i,j,\ell}$ for $i \in \mathbb{N}' \cup \{0\}$ and $\ell \in \mathbb{M}$ is equal to 1, which means that one and only one $x_{i,j,\ell} = 1$ for a given j.

- If a maneuver is the last one performed by a maneuver team, then it has no successors. Otherwise, the maneuver has only one successor maneuvers. Constraints (A.4) ensure, for each $i \in \mathbb{N}'$, the sum of all $x_{i,j,\ell}$ for $j \in \mathbb{N}'$ and $\ell \in \mathbb{M}$ is less or equal to 1.

- If team ℓ maneuvers switches i and j, with i maneuvered immediately before j, there is a switch h maneuvered immediately before i by the same team. The left side of (A.5) ensures the sum of all $x_{h,i,\ell}$, for $h \in \mathbb{N}' \cup \{0\}$, to be equal equal to 1 if $x_{i,j,\ell}$ is equal to 1.

- If a switch j is the first one maneuvered by a team ℓ, then the moment in which it is maneuvered has to be satisfy the displacement time of team ℓ from its initial location to the switch location. The disjunctive constraints (A.6) ensure $t_i \geq s_{0,j,\ell}$ if j is the first switch maneuvered by a team.

- If switches i and j are maneuvered by the same team ℓ, with i maneuvered immediately before j, then the moment in which j is maneuvered must not be before the moment in which the maneuver of i is completed increased by the displacement time from i to j. The disjunctive constraints (A.7) ensure $t_j \geq t_i + x_{i,j,\ell}$ if $x_{i,j,\ell} = 1$.

- The moment a switch (manually or remotely operated) is maneuvered must satisfy the precedence constraints given by the precedence graph $\mathcal{G}_{\prec}$. Constraints (A.8) ensure that all precedence constraints are satisfied.

- Constraints (A.9) are used to compute the makespan C_{max}.

- Finally, constraints (A.10)–(A.12) define the domain of the decision variables: $x_{i,j,\ell}$ are binary variables, and t_i and C_{max} are non-negative continuous variables.

The formulation above also uses big-M values at disjunctive constraints (A.5) and (A.6). A value for the this constant $\mathcal{M}$ can be defined as described in Eq. (3.14).

A.2 Arc-time-indexed variables

The MIP formulation presented in this section is based on binary arc-time-indexed variables [Pessoa et al., 2010]. This approach segments the time horizon in discrete units. Then, it assumes the parameters p_i and $s_{i,j,\ell}$ are non-negative integer values.

In addition to the parameters introduced so far, let $\overline{T}$ be the upper limit for the makespan, the time horizon is discretized in values 0, 1, ..., $\overline{T}$. The value of $\overline{T}$ can be computed as the big-M in Eq. (3.14), but the larger this value the larger is the number of variables and constraints in the formulation. Then, $\overline{T}$ can be set as the makespan of a solution obtained by a heuristic.

Given the decision variables:

$$w_{i,j,\ell,r} = \begin{cases} 1, & \text{if } i \text{ and } j \in \mathbb{N}' \text{ are maneuvered by team } \ell \in \mathbb{M} \text{ with } j \text{ maneuvered} \\ & \text{at moment } r; \\ 0, & \text{otherwise.} \end{cases}$$

$t_i \geq 0,$ the moment in which the maneuver of $i \in \mathbb{N} \setminus \{0\}$ is performed;

$C_{max} \geq 0,$ the moment in which all maneuvers are completed (makespan);

the maneuver scheduling problem in the restoration of electric power distribution networks can be formulated as the following MIP problem:

$$\text{Min. } C_{max} \tag{A.13}$$

$$\text{s.t.: } \sum_{j \in \mathbb{N}'} \sum_{r = s_{0,j,\ell}}^{H-p_j} w_{0,j,\ell,r} \leq 1 \qquad \forall \ell \in \mathbb{M} \tag{A.14}$$

$$\sum_{\substack{i \in \mathbb{N}' \cup \{0\}: \\ i \neq j}} \sum_{\ell \in \mathbb{M}} \sum_{\substack{r = s_{0,i,\ell} + \\ p_i + s_{i,j,\ell}}}^{H-p_j} w_{i,j,\ell,r} = 1 \qquad \forall j \in \mathbb{N}' \tag{A.15}$$

$$\sum_{\substack{j \in \mathbb{N}': \\ j \neq i}} \sum_{\ell \in \mathbb{M}} \sum_{\substack{r = s_{0,i,\ell} + \\ p_i + s_{i,j,\ell}}}^{H-p_j} w_{i,j,\ell,r} \leq 1 \qquad \forall i \in \mathbb{N}' \tag{A.16}$$

$$\sum_{\substack{h \in \mathbb{N}' \cup \{0\}: \\ h \neq i,\ h \neq j}} \sum_{\substack{v = s_{0,h,\ell} + \\ p_h + s_{h,i,\ell}}}^{r - p_i - s_{i,j,\ell}} w_{h,i,\ell} \geq w_{i,j,\ell,r} \qquad \begin{array}{c} \forall i, j \in \mathbb{N}' : i \neq j;\ \forall \ell \in \mathbb{M}; \\ r = s_{0,i,\ell} + p_i + s_{i,j,\ell}, \ldots, H - p_j; \end{array} \tag{A.17}$$

$$t_j = \sum_{\substack{i \in \mathbb{N}': \\ i \neq j}} \sum_{\ell \in \mathbb{M}} \sum_{\substack{r = s_{0,i,\ell} + \\ p_i + s_{i,j,\ell}}}^{H - p_j} r w_{i,j,\ell,r} \qquad \forall j \in \mathbb{N}' \tag{A.18}$$

$$t_j \geq t_i + p_i \qquad \forall (i,j) \in \mathbb{P} \tag{A.19}$$

$$C_{max} \geq t_i + p_i \qquad \forall i \in \mathbb{N} \setminus \{0\} \tag{A.20}$$

$$w_{i,j,\ell,r} \in \{0,1\} \qquad \begin{array}{r} i \in \mathbb{N}' \cup \{0\};\; j \in \mathbb{N}' : i \neq j;\; \ell \in \mathbb{M}; \\ r = s_{0,i,\ell} + p_i + s_{i,j,\ell}, \ldots, H - p_j; \end{array} \tag{A.21}$$

$$t_i \geq 0 \qquad \forall i \in \mathbb{N} \setminus \{0\} \tag{A.22}$$

$$C_{max} \geq 0 \tag{A.23}$$

in which the objective (A.13) is the minimization of the makespan, subject to the constraints:

- Each team has to start their work from its initial location. Then, constraints (A.14) ensure that for each team $\ell \in \mathbb{M}$, there is at most one variable $w_{0,j,\ell,r}$ equal to one, for $j \in \mathbb{N}'$ and values of r available. Notice that the values of r start from $s_{0,j,\ell}$, which ensure that if a switch j is the first one maneuvered by team ℓ, then it can not be maneuvered before the time required to team ℓ to arrive its location.

- Each maneuver of a manually controlled switch has to be performed by a single maintenance team. Constraints (A.15) ensure that for each $j \in \mathbb{N}'$ there is only exactly one variable $w_{i,j,\ell,r} = 1$ for a given j.

- Each maneuver has at most one successor maneuver if it is not the last one maneuvered by a team. Otherwise, it has no successor maneuver. The left side of constraints (A.16) are equal to 1 if switch $j \in \mathbb{N}'$ is not the last one maneuvered by a team, or equal to 0 otherwise.

- The predecessor of each maneuver must be assigned to the same team (A.17). This set of constraints, together with constraints (A.15) and (A.16) also ensure that variables $w_{i,j,\ell,r}$ are equal to one only if the respective value of r respect the displacement and processing times between consecutive maneuvers.

- The moment t_j in which a switch j is maneuvered must be the equal to the value of r in which the binary variable $w_{i,j,\ell,r}$ is active. Constraints (A.18) just ensure this condition to be satisfied.

- The moment a switch (manually or remotely operated) is maneuvered must satisfy the precedence constraints given by the precedence graph $\mathcal{G}_{\prec}$. Constraints (A.19) ensure that all precedence constraints are satisfied.

- Constraints (A.20) are used to compute the makespan C_{max}.

- Finally, constraints (A.21)–(A.23) define the domain of the decision variables: $w_{i,j,\ell,r}$ are binary variables, and t_i and C_{max} are non-negative continuous variables.