Joint optimization and planning of vehicle routing and maintenance operations

Lamiaa Dahite

Ph.D. Thesis

Major: Information and Communication Sciences and Technologies
Specialty: Computer Science and their applications

presented at the ***Ecole Doctorale en Sciences Technologie et Santé (ED 585)*** **of the**
Université du Littoral Côte d'Opale
and
at the ***Centre d'Etudes Doctorales - Sciences, Ingénierie et Développement Durable (CEDoc-SIDD)*** ***of the***
Institut National de Statistique et d'Economie Appliquée

by

Lamiaa Dahite

to obtain the degree of Doctor from the Université du Littoral Côte d'Opale

Joint optimization and planning of vehicle routing and maintenance operations

Defended on December 1st, 2022, after the approval of the reviewers, in front of the examining board:

Mr. Adnan Yassine, Professor, the Université du Havre, France	**Reviewer**
Mr. Said Achchab, Professor, the Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes, Morocco	**Reviewer**
Mr. Sidi Mohamed Douiri, Professor, the Université Mohammed V de Rabat, Morocco	**Reviewer**
Mr. Aziz Moukrim, Professor, the Université de technologie de Compiègne, France	**Examiner (President)**
Mr. Cyril Fonlupt, Professor, the Université du Littoral Côte d'Opale, France	**Thesis director**
Mr. Abdeslam Kadrani, Professor, the Institut National de Statistique et d'Economie Appliquée, Morocco	**Thesis director**
Mrs. Rym Guibadj, Assistant Professor, the Université du Littoral Côte d'Opale, France	**Guest**
Mr. Rachid Benmansour, Associate Professor, the Institut National de Statistique et d'Economie Appliquée, Morocco	**Guest**

Thèse de Doctorat

Mention: Sciences et Technologies de l'Information et de la Communication
Spécialité: Informatique et applications

présentée à ***l'Ecole Doctorale en Sciences Technologie et Santé (ED 585)*** **de**
l'Université du Littoral Côte d'Opale
et
au Centre d'Etudes Doctorales - Sciences, Ingénierie et Développement Durable (CEDoc-SIDD) de
l'Institut National de Statistique et d'Economie Appliquée

par

Lamiaa Dahite

pour obtenir le grade de Docteur de l'Université du Littoral Côte d'Opale

Optimisation et planification conjointe de tournées de véhicules et d'opérations de maintenance

Soutenue le 01/12/2022, après avis des rapporteurs, devant le jury d'examen:

M. Adnan Yassine, Professeur des Universités, l'Université du Havre, France	**Rapporteur**
M. Said Achchab, Professeur de l'Enseignement Supérieur, l'Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes , Maroc	**Rapporteur**
M. Sidi Mohamed Douiri, Professeur Habilité, l'Université Mohammed V de Rabat, Maroc	**Rapporteur**
M. Aziz Moukrim, Professeur des Universités, l'Université de technologie de Compiègne, France	**Examinateur (Président)**
M. Cyril Fonlupt, Professeur des Universités, l'Université du Littoral Côte d'Opale, France	**Directeur de thèse**
M. Abdeslam Kadrani, Professeur de l'Enseignement Supérieur, l'Institut National de Statistique et d'Economie Appliquée, Maroc	**Directeur de thèse**
M^{me} Rym Guibadj, Maître de Conférences, l'Université du Littoral Côte d'Opale, France	**Invité**
M. Rachid Benmansour, Professeur Habilité, l'Institut National de Statistique et d'Economie Appliquée, Maroc	**Invité**

DEDICATION

I dedicate this thesis :

To my beloved mother, father, and sister,
Thank you for your support and your unconditional love.
Thank you for always being there to comfort, and encourage me.

To the people who have been there one day or another to encourage me.

I dedicate this thesis to all these people.
I hope that they will find in these modest words all my gratitude.

Lamiaa Dahite

Abstract

Systems are necessary for the functioning of our society. They are present in all sectors and aspects of our lives. However, systems degrade with age and usage. A failure of some critical complex systems can have dramatic consequences on safety and security. In addition, economic losses and environmental damage can be incurred. Due to the nature of the aging behavior of systems, a suitable maintenance strategy must be designed and implemented to reduce the risk of failures. Companies are becoming more aware of the importance of asset maintenance management. Most outsource this function to specialized companies because they understand its management and execution complexity. It is also the opportunity for them to focus on their core business. The maintenance service provider aims to provide efficient approaches to managing the maintenance service through continuous planning to achieve the best outcomes at a lower cost. He also needs to ensure transportation management to perform maintenance to its various customers. This thesis deals with the joint maintenance scheduling and workforce routing problem. This problem consists in defining the technicians' routes to perform maintenance operations at the right time on geographically distributed machines subject to random failures.

First, a mathematical model has been proposed to integrate the maintenance planning problem with technicians' routing. The model simultaneously determines the optimal maintenance and routing plan. Three objective functions have been proposed that incorporate failure, maintenance and routing considerations. Constructive heuristics based on failure and maintenance behaviors have been designed to generate initial solutions, followed by a general variable neighborhood search metaheuristic to solve the problem. Then, new multi-objective algorithms based on variable neighborhood descent, general variable neighborhood search, and Pareto dominance have been proposed to deal with the bi-objective problem where each maintenance objective is associated with the routing objective. The new mechanisms used have been detailed, including the improvement method, the neighborhood exploration method, the acceptance criterion, the stopping criterion, and the neighborhood change procedure. Third, an extension of the previous problem incorporating opportunistic maintenance

has been proposed. This new problem considers the existence of planned production stoppages as a constraint. The goal of this strategy is to make the arrival time of the technicians coincide with the optimal time of maintenance and to place the maintenance operations in the available planned production stoppages. The problem that integrates possible production losses has been modeled and solved using an adaptive large neighborhood search incorporating a dedicated heuristic and specific operators. Finally, the bi-objective variant of the problem has been modeled and solved. The objectives of minimizing the routing cost, maintenance cost, penalty cost for not respecting maintenance intervals, and production losses have been considered. New multi-objective algorithms that integrate novel mechanisms and that are based on Pareto local search, adaptive large neighborhood search, and Pareto dominance have been proposed to solve this problem. The performance of the proposed models and algorithms has been evaluated using several generated instances. As a result, the algorithms outperform the commercial solver and algorithms from the literature.

Keywords: vehicle routing problem, time-based maintenance, opportunistic maintenance, multi-objective optimization, variable neighborhood search, large neighborhood search.

Résumé

Les systèmes sont nécessaires au fonctionnement de notre société. Ils sont présents dans tous les secteurs et aspects de notre vie. Les systèmes se dégradent avec l'âge et l'utilisation. Une défaillance de certains systèmes complexes critiques peut avoir des conséquences dramatiques sur la sûreté et la sécurité. De plus, des pertes économiques et des dommages environnementaux peuvent être encourus. En raison de la nature du comportement vieillissant des systèmes, une stratégie de maintenance adaptée doit être conçue et appliquée pour réduire le risque de défaillances. Les compagnies sont de plus en plus conscientes de l'importance de la gestion de la maintenance des actifs. La plupart d'entre elles sous-traitent cette fonction à des sociétés spécialisées en raison de leurs compréhension de la complexité de sa gestion et de son exécution. C'est également l'opportunité pour eux de se concentrer sur leur cœur de métier. Le fournisseur des services de maintenance vise à fournir des approches efficientes pour gérer la maintenance grâce à une planification continue pour obtenir les meilleurs résultats au moindre coût. Il doit également s'assurer de la gestion du transport pour effectuer la maintenance pour ses différents clients. Cette thèse s'intéresse à la planification conjointe de la maintenance et des tournées des techniciens. Ce problème consiste à définir le routage des techniciens pour effectuer les opérations de maintenance au bon moment sur des installations géographiquement distribuées et sujettes à des défaillances aléatoires.

Premièrement un modèle mathématique a été proposé pour l'intégration du problème de la planification de la maintenance avec le routage des techniciens. Le modèle détermine simultanément le plan de maintenance et de routage optimal. Trois fonctions objectifs qui intègrent à la fois des considérations de défaillance, de maintenance et de routage ont été proposées. Des heuristiques constructives basées sur le comportement de la défaillance et de la maintenance ont été conçus pour générer des solutions initiales suivies d'une métaheuristique de recherche à voisinage variable pour la résolution du problème. Ensuite, des algorithmes multi-objectifs basée sur la descente à voisinage variable, la recherche générale à voisinage variable et la dominance de Pareto ont été proposés pour traiter le problème bi-objectifs où chaque objectif de maintenance

est associé avec l'objectif du routage. Les nouveaux mécanismes utilisés sont détaillés notamment la méthode d'amélioration, la méthode d'exploration de voisinage, le critère d'acceptation, le critère d'arrêt et la procédure de changement de voisinage. Troisièmement, une extension du problème précédent intégrant la maintenance opportuniste a été proposée. Ce nouveau problème considère l'existence des fenêtres d'arrêt de production programmés comme une contrainte à prendre en compte. Le but de cette stratégie de faire coïncider le temps d'arrivée des techniciens avec le temps optimal de maintenance mais aussi de placer les opérations de maintenance au sein des arrêts de production planifiés disponibles. Le problème intégrant les pertes possibles de production a été modélisé et résolu à l'aide d'une recherche adaptative à voisinage large qui intègre une heuristique dédiée et des opérateurs spécifiques. Finalement, la variante bi-objectifs du problème a été modélisée et traitée. Les objectifs de minimisation du coût de routage, de maintenance, des pénalités de non-respect des intervalles de maintenance, et de pertes en production ont été considérés. Des algorithmes multi-objectifs intégrant de nouveaux mécanismes et basés sur la recherche locale de Pareto, la recherche adaptative à voisinage large et la dominance de Pareto ont été proposés. La performance des modèles et des algorithmes proposés a été évaluée à l'aide de plusieurs instances générées. Les algorithmes surpassent le solveur commercial et des algorithmes de la littérature.

Mots clés: tournées de véhicules, maintenance basée sur le temps, maintenance opportuniste, optimisation multi-objectifs, recherche à voisinage variable, recherche adaptative à voisinage large.

Contents

List of Tables

List of Figures

Introduction

Outline of the current chapter

Context and motivation

Systems are necessary to the functioning of our society. They are present in every sector and aspect of our life: transport systems, communication systems, utilities, manufacturing plants, processing plants, hospitals, and banks [3].

Systems degrade with age and usage. A failure of some critical complex systems can have dramatic consequences on safety and security. The crash of an airplane or the collapse of a bridge are some illustrative examples [3]. Furthermore, economic losses and environmental damage can be incurred.

Due to the nature of the aging behavior of systems, a good maintenance strategy must be designed and applied to reduce the risk of failures. Maintenance aims to retain a piece of equipment operating or restore it to the previous state where it performs its required function. The maintenance management of complex systems is often complicated and requires an adapted strategy. The maintenance strategy needs to describe proper maintenance operations and how

and when to execute them. They include preventive and corrective maintenance (repair or replacement), inspection, and monitoring.

Maintenance evolved over the years from a technical concern to a become strategic management one. The evolution of technologies has also added scientific and technological dimensions to maintenance [3]. Indeed, maintenance outsourcing, real-time monitoring using sensors, and maintenance data analysis are often encountered in this field of study [3].

Companies are more aware of the importance of asset maintenance management. Most outsource this function to specialized companies because of their understanding of its management and execution complexity. It is also the opportunity for them to focus on their core business. The maintenance service provider aims to provide efficient approaches to managing the maintenance service through continuous planning to achieve the best outcomes at a lower cost.

This research investigates the maintenance scheduling of operations for a maintenance service provider. This kind of company is a service business. An essential part of the quality of service is related to the delivery of that service. Therefore, this service company must ensure good quality maintenance to its geographically distributed customers. This quality of maintenance is strongly linked to the time taken to complete this service.

Appropriate models need to be built to determine the optimum level of Preventive Maintenance (PM) and the optimal time to execute it to reduce failures and overall maintenance costs. Reliability engineering is predominant in this phase. It controls the operation of systems.

Mathematical maintenance policies are used to develop an effective preventive maintenance plan. The objective of such policies is to design a maintenance schedule including both preventive replacement and corrective replacement. The maintenance models dealing with these policies are based on the probability theory. The main reason is the uncertainty of the mechanism that causes failure [102].

The maintenance service is also tightly coupled to transportation management for a service provider. The technicians must arrive at the right time for the maintenance operations of the various customers. It is conditioned by the re-

spect of the deadlines and the efficient organization of maintenance technicians' routing. The arrival of technicians ought to coincide with the optimal time of PM operations while satisfying the service delivery constraints. Transport management is strongly involved in this phase. Sequencing the visits and determining the best routing policy for the geographically distributed machines is necessary. Vehicle Routing Problems (VRP) are essential tools for modeling transportation within a network. The basic VRP uses a few information: customers' locations and travel times or travel distances.

More sophisticated constraints have been added later on to deal with the complexity of the real-world requirements such as time windows, stochastic travel times, real-time planning, etc. The Vehicle Routing Problem with time windows (VRPTW) is the most used variant due to its theoretical and practical applications. It has been used successfully on several applications, including the scheduling of maintenance operations.

Problem statement, objectives and research target

Workforce Scheduling consists of constructing work timetables for the personnel to permit the organization to meet service demand. It aims at scheduling the workforce to perform tasks. It is a scheduling problem encountered by many organizations where the workforce, technicians, is the leading resource.

Maintenance services providers are among those organizations. They need to plan maintenance operations on geographically scattered machines in an efficient manner while satisfying operational constraints. The problem is complex, especially when more requirements that stem from real-life settings are considered, such as uncertainty.

Hereafter, the machines are subject to random failures that can consequently cause substantial economic losses and environmental damage. The equipment's manufacturer sometimes defines the maintenance times. Still, usually, the companies entrust their maintenance providers with the task of determining the maintenance planning that provides high-quality service at the lowest overall cost. As a result, the service provider tries to reduce the number of maintenance operations as much as possible, which can be realized by maximizing the periods

between two successive visits. At the same time, it seeks to avoid the enormous cost resulting from performing corrective maintenance operations when the machine suddenly breakdowns. Therefore, it means finding the best trade-off of the visits number to each machine to perform maintenance operations. The technicians are then routed to the maintenance operations to arrive in time.

The figure 1 depicts the problem. Several machines are located in dispersed customer sites. They are subject to random breakdowns because of the failure of a critical component. For each machine and at regular time intervals, preventive maintenance (PM) interventions are scheduled. Technicians must perform PM operations at the best possible times that reduce maintenance and transport costs. In case of sudden breakdowns, the technicians perform corrective maintenance (CM) operations.

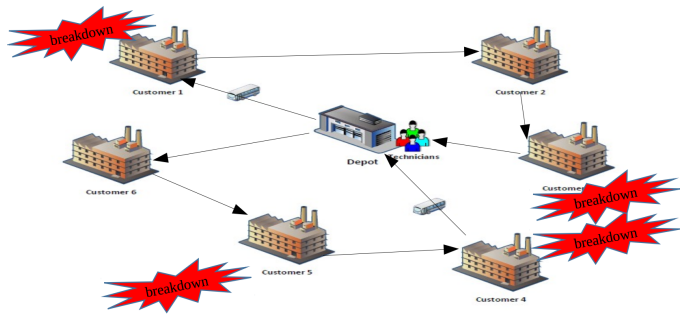

Figure 1: Schema of the problem

The decision maker in this kind of company needs to design a maintenance strategy and the transport of technicians with an overall view of the supply chain network by integrating the technical and organizational issues efficiently.

Cost, availability, reliability, and maintainability are the benchmark for evaluating maintenance decisions like cost, quality, and time objectives in logistics [62]. In supply chain management, a balance between cost and value is essential. The value is measured by the quality of a product or service and the delivery time. Maintainability is a critical concept used when the maintenance considerations are addressed in the design stages of systems [64]. Reliability engineering controls the operation of systems. Reliability is the probability that a system functions correctly and provides the desired outputs. Availability is the

probability that a system is operational and available to use. All these concepts are interconnected. A well-designed system is more reliable. A reliable system is less prone to failure. A system that is less likely to fail has more time to operate and is available. High availability, reliability, and maintainability reduce maintenance costs considerably.

The maintenance company would be interested in finding the best trade-off between service and maintenance costs on the one hand and operational and transport costs on the other hand. However, nowadays, manufacturers are moving towards integrating additional objectives to build integrated systems based on the overall view of the supply chain. This thesis aims to jointly consider maintenance technical aspects and operations management organizational aspects. The thesis aims to conceive efficient asset maintenance operations strategies and ensure their execution in the best possible way.

Determining the optimum maintenance strategy of geographically distributed assets requires building appropriate models. The transportation management of technicians is necessary to ensure this strategy is respected in time. All stakeholders need to be considered in the design of the global problem. The modelization needs to integrate the peculiarities of that latter. Reliability engineering and transport management are significant issues in designing and managing such a logistic network.

The use of optimization techniques is then essential to solving the problem. Well-performing algorithms have been developed in the literature for several combinatorial problems and VRPs. Although, there are still several challenges to be addressed to deal with these problems. Indeed, they arise from real-world scenarios due to the number of parties involved and their complex interaction, such as multiple objectives, large scale, uncertainty, limited computational time and effort, etc. Therefore, the thesis will also focus on designing efficient single-objective and multi-objective algorithms capable of solving the identified problem within a reasonable time.

Empirical and theoretical approaches

The empirical approach used so far is mixed sequential exploratory (quantitative and qualitative) with a dominance of the quantitative approach. Indeed, hypotheses are generated qualitatively based on the reality and the critical review of previous literature work. Individual cases are also studied. These hypotheses are then tested quantitatively using the simulation of a generated dataset on the built models and algorithms. The arranged epistemological position is adopted for our research design, combining an inductive research approach (constructivist and interpretative) and a hypothetical-deductive approach (positivist). The first step is used during the construction of the decision support system. We consider the complex nature of management's situations and the human dimensions in constructing the decision aid system destined for managers. The projective aim, and not only the interpretive one, of the knowledge produced contributes to this approach of engineering and management sciences and in our case. The application of our models will also be generalized to all similar industries, which supports this approach since we start from a particular point to a general case. The second approach is used when constructing mathematical methods and models. We begin with generic mathematical models of maintenance and routing to build a particular non-existent model. The same approach is adopted in the design of our solving algorithms. The combination of mathematical models inherent in our design requires this second approach generally adopted for the natural sciences. Details about designing a research methodology is available in [4].

Contributions and thesis overview

The thesis is structured as follows. The first part is dedicated to a background and literature review of the central notions and research areas explored throughout the thesis. These are divided into two chapters. This thesis expects to contribute to the literature on maintenance scheduling and workforce routing problems. The rest of the thesis is divided into several contributions summarised as follows.

Chapter 1 presents the leading strategies and orientations of maintenance and

the main methods and mathematical models used to deal with maintenance from the foundation models till now. First, maintenance orientations, maintenance concepts, and maintenance strategies are presented. The latter are reviewed with an emphasis on Reliability Centred Maintenance. Basic formalized models for maintenance optimization are then introduced. The second part of the chapter is devoted to presenting the variants of the Vehicle Routing Problem (VRP) since the original problem, the Travelling Salesman Problem (TSP). Finally, modeling paradigms are also discussed.

Chapter 2 is dedicated to the literature related to resolution methods. It starts with an overview of the different classes of optimization problems and existing heuristics and metaheuristics methods. It finally surveys multi-objective optimization techniques.

Chapter 3 proposes a new model for the maintenance scheduling and workforce routing problem. The model simultaneously determines the optimal maintenance and routing plan. More specifically, three different objective functions integrating routing, failure, and maintenance considerations are defined. In order to solve the problem, new constructive heuristics based on the failure and maintenance behavior are designed to generate initial solutions and speed up the search. The problem is then solved using a General Variable Neighborhood Search. Finally, the computational studies on generated large benchmark datasets have been realized to validate the proposed model and solution approach.

Chapter 4 deals with the bi-objective variant of the previous problem. It proposes new adaptations of Variable Neighborhood Descent (VND) and General Variable Neighborhood Search (GVNS) to tackle multi-objective optimization problems. First, it outlines the multi-objective algorithms proposed and details their components. Moreover, it describes several GVNS-based frameworks. Then, it reports the computational experiments conducted to examine the performance of the proposed algorithms in comparison with state-of-the-art multi-objective VND and GVNS algorithms.

Chapter 5 starts with a comprehensive analysis to extend the problem. It describes the development of a new model by integrating opportunistic maintenance scheduling as a new concept in the field of workforce routing problems.

The new model captures all the new characteristics incorporated in the previous problem within an opportunistic maintenance strategy. Production stoppages are already scheduled on the planning horizon. The PM operations need to be performed during those stoppages to reduce the production interruption costs. A heuristic is designed to select a production stoppage and determine the maintenance operation time. It is integrated with an Adaptive Large Neighborhood Search to solve the problem. The chapter also reports computational studies to measure the performance of the proposed model and solution approach with various parameters.

Chapter 6 is devoted to formulating and resolving a bi-objective variant of the joint opportunistic maintenance and workforce routing problem. It describes the framework of new multi-objective algorithms based on Pareto Local Search, Variable Neighborhood Descent, and Adaptive Large Neighborhood Search. It also details the experiments and discusses the results and differences between almost all the thesis algorithms.

Finally, this thesis's main contributions and results are summarized along with new directions for further works and perspectives.

Publications

To support the validation process, we communicated the research work at conferences and published it in international journals. The feedback from the reviewers and the international community improved the quality of the submitted papers and validated the proposed approaches. We presented the contributions and realized the articles under the supervision of our thesis directors (principal supervisors) and co-supervisors. The research in this thesis includes some original work that has been previously published in the following peer-reviewed articles:

International conferences papers

Dahite L., Kadrani A ., Benmansour R. ,Guibadj R. N., Fonlupt C. (2020). Optimization of Maintenance Planning and Routing Problem. In: Benmansour R., Sifaleras A., Mladenović N. (eds) Variable Neighborhood Search. ICVNS

2019. Lecture Notes in Computer Science, vol 12010. Springer, Cham. *DOI* : 10.1007/978 – 3 – 030 – 44932 – 2_7, https://doi.org/10.1007/978-3-030-44932-2_7

Dahite L., Guibadj R. N., Fonlupt C., Kadrani A ., Benmansour R. (2021, May). A Semi Adaptive Large Neighborhood Search for the Maintenance Scheduling and Routing Problem. In 2021 7th International Conference on Optimization and Applications (ICOA) (pp. 1-6). IEEE. *DOI* : 10.1109/*ICOA*51614.2021.9442627, https://ieeexplore.ieee.org/document/9442627

International journals

Dahite L., Kadrani, A., Benmansour, R., Guibadj, R.N., Fonlupt, C. Multi-Objective Model and Variable Neighborhood Search Algorithms for the Joint Maintenance Scheduling and Workforce Routing Problem. Mathematics 2022, 10, 1807. https://doi.org/10.3390/math10111807

Dahite L., Guibadj R. N., Fonlupt C., Kadrani A ., Benmansour R. The Joint Opportunistic Maintenance Scheduling and Routing Problem: Model and Solution Approach based on Adaptive Large Neighborhood Search. Journal paper to be submitted.

National conference

Dahite L., Kadrani A ., Benmansour R. , Guibadj R. N., Fonlupt C. An Adaptative Large Neighborhood Search for the Maintenance Scheduling and Routing Problem. ROADEF 2020, Montpellier, France.

Data

Dahite L., Kadrani A ., Benmansour R. , Guibadj R. N., Fonlupt C. (via Mendeley Data) (2022): Data of: Multi-Objective Model and Variable Neighborhood Search Algorithms for the Joint Maintenance Scheduling and Workforce Routing Problem. *DOI* : 10.17632/*k*84*dbvvv*4*m*.1, https://doi.org/10.17632/k84dbvvv4m.1

Part I

Literature

Chapter 1

Routing and Maintenance problems

Outline of the current chapter

1.1 Introduction

The objective of this chapter is to review the past and current research on maintenance planning optimization and Vehicle Routing problems. Such a literature review aims to understand how the two problems can be dealt with simultaneously to solve efficiently extended variants of the Workforce Routing Problem. In this chapter, we first present within a structured framework the leading strategies and orientations of maintenance and the main methods and models used to deal with maintenance planning from the foundation models till now. We then describe some variants of the Vehicle Routing Problem (VRP) since the original problem, the Travelling Salesman Problem (TSP).

1.2 Maintenance orientations

The golden triangle (cost, quality, time) gathers the main criteria that need to be accounted for in production scheduling. On the other side, the three criteria (cost, availability, reliability and maintainability) reference decision-making in the maintenance process [62].

Based on the three above criteria, three main orientations appeared to classify maintenance strategies in an organization.

Maintenance orientations have been classified by Claude Pellegrin in his book [62]. We keep the same classification he made and add more details to his definitions in this paragraph. According to Claude Pellegrin [62], the maintenance criteria must be contrasted with each other [62]:

The trade-off between technical criteria (availability, reliability, and maintainability) and the economic criterion (cost) highlights the debate between physical and economic piloting [62]. The maintenance decision-maker must guarantee the availability of resources for the production function at the least cost of maintenance and loss of production. He must also ensure their reliability and maintainability. On the other hand, the maintenance decision-maker has constraints related to the operator and the financier [62].

The opposition between availability and cost criteria on the one hand and reliability and maintainability on the other hand [62]. It illustrates the link between the choice of maintenance policies (corrective, preventive, etc.), the organization of maintenance (resources, equipment, etc.), and secondly, the efforts made to improve the reliability and maintainability of the equipment.

The classic opposition between firstly reliability, maintainability, cost, and secondly availability [62] means the opposition between what is observable in reality, the availability for which we can measure the effectiveness in the plant, and the models built (theory) to ensure the machine availability and the economic valuation models.

1.3 Maintenance strategies

The oppositions of the above section are the basis of the three maintenance strategies present in the literature and developed in the sixties. Pintelon and Parodi-Herz [63] listed many maintenance concepts. According to them, some concepts have evolved to become philosophies or even more maintenance strategies. We present in the following the central maintenance concepts that become strategies:

Life Cycle Cost (LCC) is a cost-oriented strategy that we think is supplied by the first opposition. It is based on considering maintenance costs over the equipment's entire life cycle. It is a strictly economic approach that aims to optimize the overall cost (acquisition + use + maintenance), whether direct or indirect, over the entire equipment's life cycle from the design. We seek to obtain the detailed cost of failures information over the equipment's lifetime to plan maintenance logistics [63]. In the LCC concept, several approaches exist. The most important are [63]: Terotechnology, Integrated Logistic Support/Logistic Support Analysis (ILS/LSA), and Capital asset management.

Reliability Centered Maintenance (RCM) fueled by the second opposition and based mainly on the importance of improving reliability and maintainabil-

ity through the design of the equipement and the training of operators. This approach relies on understanding each part's functioning and the impact of the failure on its functions. It is a technical approach where safety and environmental integrity are more important than cost, but the economic dimension is still there. RCM is an approach that focuses on reliability, as its name suggests. It was first developed for a high-tech/high-risk environment, but we found it now in every industry. RCM was used in the 1960s in the North American aviation industry, then was adopted by military aviation to finally be used at the high-risk industrial plants (nuclear power plants) [63]. Its main tools are statistical analysis.

The main objective of RCM is to determine the number of efficient preventive maintenance (PM) tasks and their optimal times through a detailed analysis of failure modes and failure causes. The results obtained from the analysis carried out to implement the RCM approach is often beneficial for proper planning of corrective maintenance strategies, spare part optimization as well as other logistic considerations [65], such as in the case of this thesis.

The RCM analysis process represents several sequential activities to realize. Rausand and Vatn [65] present the 12 steps of RCM process analysis:

1. The preparation of the analysis and study
2. The selection and definition of the system
3. The Functional Failure Analysis (FFA)
4. The selection of the most critical items
5. The data collection and analysis
6. The Failure Modes, Effects, and Criticality Analysis (FMECA)
7. The selection of maintenance actions (PM, CM,...)
8. The determination of maintenance intervals and optimal times of PM
9. The comparison analysis of preventive maintenance
10. Treatment of non-critical items

11. Implementation

12. Service data collection and updating

A detailed description of each step and how to implement them in the industry can be found in [65]. Step 8 is not detailed in the latter reference. It aims to determine the maintenance intervals. There are different formalized methods for optimizing maintenance intervals in the literature. However, they are not part of the RCM. To optimize maintenance intervals, we need to choose an existing model from the ones in the literature that fits our industry and adapt it to work. If the time permits, we could probably propose improvements or new models based on our understanding of our specific industry. The RCM is a valuable strategy that has evolved over the years. Moreover, RCM-based approaches have appeared to overcome some of its drawbacks, such as Gits, Business-Centred Maintenance (BCM), Risk-Based Reliability Centred Maintenance (RBCM), streamlined RCM, and so forth [65].

Our thesis is related to Step 8, where we seek to determine optimal times to perform preventive maintenance operations.

The Total Productive Maintenance (TPM) is a philosophy founded in the 1950s by Japanese engineer Seiichi Nakajima. The TPM is fed by the first opposition by favoring technical engineering over economics. It is a general approach with an overall view on both maintenance and production in manufacturing industries. It emphasizes the organization of productive resources to improve the availability of machines. It is based on the measurement of the indicator, Overall Equipment Effectiveness (OEE), that needs to be increased. It reflects the priority of physical piloting and technical engineering over economic piloting [62]. The effective use of production capacity represents its main goal by the joint integration of production, maintenance, and quality issues. The "six big losses" of useful capacity [63] are included in the OEE indicator and are taken into account in the concept, which constitutes its main strength. The OEE is calculated as follows:

$$OEE = Availability \times Performance \times Quality \tag{1.1}$$

The availability considers unplanned and planned stops. Equaling 100 % means that the machines always work during production time. The performance measures slow cycles and minor stops. When it is 100 %, the machines are as fast as possible. Finally, the quality measure considers defective produced items, including those that need reworks. This indicator equals 100 % if all items made are good. Interested readers can refer to dedicated books for detailed descriptions of the indicators.

Customized Maintenance is the most recent maintenance strategy and can include a combination of all the above maintenance strategies and maintenance concepts.

Some concepts that have not evolved to strategies include:

Ad-hoc is among the first concepts that have appeared for maintenance to implement Failure Based Maintenance (FBM) or Time/Use Based Maintenance [63]. Ad-hoc represents unplanned maintenance orders, differently from preventive maintenance (PM) planned in advance. Indeed, Ad-hoc is used in addition to the scheduled PM.

Quick and Dirty Decision Charts (Q and D) is a diagram that is established to select the most appropriate maintenance policy. It includes several questions about failure patterns, repair behaviors, business, cost structure,etc. [63] to be answered.

In our thesis, we focus on the Reliability Centered Maintenance (RCM) since the maintenance models used are extracted from this branch of the study of maintenance. It is a technical approach that focuses on understanding the reliability of machines, but the economic point of view remains present since minimizing the maintenance cost is considered.

1.4 Maintenance policies

Failure Based Maintenance (FBM) : it consists of realizing corrective maintenance actions (CM) when the failure occurs. It is reactive maintenance.

Time Based Maintenance (TBM) and Use Based Maintenance (UBM) : Preventive Maintenance (PM) is carried out at predetermined time intervals for TBM or after a use parameter (after 1000 working hours, for instance) for UBM.

Condition Based Maintenance (CBM) : PM is realized when the value of a given machine parameter, temperature, for instance, attains a predetermined value. It can be considered a predictive maintenance strategy.

Opportunity Based Maintenance (OBM) : it consists of realizing maintenance operations (CM and PM) during an opportunity time by advancing or delaying their execution from their predefined time [63] [109]. It is a prominent policy that could be approached from different points of view and perspectives. One widespread utilization in the industry is delaying maintaining some components until maintaining other more critical components simultaneously [63] [107]. Maintenance agents can perform a PM operation in advance while performing a CM action for some components [107].

1.5 Formalized models for optimization of maintenance interval

Here are some essential reliability notations and definitions that will be used in this section and throughout the thesis. Let X be a continuous random variable expressing equipment lifetime with a density function f and a cumulative distribution F. Then, R refers to the reliability function.

The realibility of an equipment can be defined as follows:

$$R(t) = 1 - F(t) \tag{1.2}$$

Where $F(t)$ is the probability of failure at instant t and also is the lifetime distribution of the density function $f(t)$.

The lifetime expectancy of the equipment until the age N is:

$$L = \int_0^N R(t)\,\mathrm{d}t \tag{1.3}$$

The instantaneous failure rate (or failure rate) function $\lambda(t)$ is equal to:

$$\lambda(t) = \frac{f(t)}{1-F(t)} = \frac{f(t)}{R(t)} = -\frac{\frac{\mathrm{d}R(t)}{\mathrm{d}t}}{R(t)} \tag{1.4}$$

The instantaneous failure rate expresses the evolution of the conditional probability of failure or the failure rate during the lifetime of the equipment.

- If $\lambda(t)$ is decreasing in an interval, then the probability of failure or the failure rate decreases in this interval. It is the infant mortality period where we have essentially early failures. At this stage, the failures are due to errors in design or manufacturing.

- If $\lambda(t)$ is constant, the probability of failure at the instant t is independent of the probability of failure at $t + \Delta t$. The exponential distribution can be used in this interval because no memory characterizes it. It is the normal life period where failures are completely random. $\lambda(t) = \lambda$ is constant if and only if $R(t) = exp(-\lambda t)$. We say that the lifetime X follows an exponential distribution $Prob(X \geq t) = exp(-\lambda t)$ and the number of failures in the interval $[0, t]$ follows a Poisson distribution of parameter λt.

- If $\lambda(t)$ increases, we are in the wear-out period or end of life period of equipment where the failure rate increases with time. Failures in this stage are due to the aging of components and materials. That may accelerate their occurrence.

These three parts constitute the well-known bathtub curve.

In this thesis, we assumed that the lifetime of any equipment follows a Weibull distribution. It can represent the three parts of the bathtub curve depending on the value of the shape parameter β. If $0 \leq \beta \leq 1$, the failure rate decreases, which corresponds to the first part of the bathtub curve. If $\beta = 1$, the Weibull distribution becomes the exponential distribution, and this value of β

corresponds to the constant part of the bathtub curve. Finally, if $\beta > 1$, the part of the bathtub curve corresponds to the wear-out period of machines.

In all the following models, we can distinguish two cases of modeling the time: the time is either continuous or discrete. We present only the case used in this thesis and that we judge to be the most realistic case, which is the consideration of a continuous time. For more description of these models, we suggest the book [62]. Moreover, for a demonstration and proof of these models in the case of discrete and continuous time, the reader can refer to the paper [102].

The formalized models for optimization of maintenance interval aim to simultaneously model the material degradation process and its impact on the related costs by building decisions based on technical parameters that characterize the material degradation and management parameters such as cost and time.

1.5.1 Age replacement model

It replaces the component with a new component at the cost of Cpm as soon as it reaches age N. A failure before age N results in a replacement under failure cost of Ccm. In this latter cost, we should include the cost of replacement by a new component (corrective maintenance CM) and the cost loss of production. The expected cost associated with an operating cycle (time interval between two replacements) is:

$$Cpm \times (1 - F(N)) + Ccm \times F(N) \tag{1.5}$$

By neglecting the duration of interventions (preventive and corrective), the expected cost CM per unit time in the steady state for the age replacement model when the PM is performed at the period N is [62] [102]:

$$CM(N) = \frac{Cpm(1-F(N))+CcmF(N)}{\int_0^N (1-F(t))\,\mathrm{d}t} \tag{1.6}$$

It is worth mentioning that the Age Preventive Replacement (APR) is only possible in the case of an increasing failure rate [62]. Therefore, the model aims to solve the equation to find N^*, the optimal age for replacement.

1.5.2 Periodic replacement models

Basic model

A component is replaced by another new component at the periodicity T, whatever its age, with the preventive cost of Cpm. If a failure occurs during period T, a replacement under failure is carried out with a corrective cost Ccm. The cost of production losses needs to be included in this cost.

The cost per unit of time is therefore [62] [102]:

$$CM(T) = \frac{Cpm+Ccm\times m(T)}{T} \tag{1.7}$$

$m(T)$ represents the average number of renewals over the interval $[0, T]$. The computation of $m(T)$ is difficult since it verifies the equation [62]:

$$m(T) = \int_0^T (1 + m(T-u))f(u)\,\mathrm{d}u \tag{1.8}$$

In the case of weibull distribution where the expression of the density function is:

$$f(t) = \beta t^{\beta-1} \times exp(-t^{\beta}) \tag{1.9}$$

The expression of $m(T)$ remain still difficult with the weibull distribution. Lomnicki (1960) proposed a polynomial expression to compute $m(T)$ [62]:

$$m(T) = \sum_{j\geq 1} c_j(\beta)\Gamma(j, bT) \tag{1.10}$$

Where $\Gamma(j, bT)$ represents the gamma incomplete and the $c_j(\beta)$ factors that need to be calculated.

Periodic replacement model with minimal repair

In this model with minimal repair of Barlow and Hunter [2], the material is replaced with the periodicity T but a failure is corrected by a "minimal repair" which does not modify the rate of degradation. The average number of failures in $[0, T]$ is equal to the number of minimum repairs, which equal the cumulative random function:

$$H(T) = \int_0^T \lambda(t)\,\mathrm{d}t \tag{1.11}$$

The average cost per unit of time is therefore [62] [102]:

$$CM(T) = \frac{Cpm+Ccm\times H(T)}{T} \tag{1.12}$$

Here we can easily obtain the optimal replacement interval T^*. For a weibull distribution for example, it can be demonstrated that assuming that $\beta > 1$ and $Ccm > Cpm$ [62]:

$$H(T) = T^{\beta} \tag{1.13}$$

The optimal period to execute maintenance in this case is:

$$T^* = \frac{(\frac{Ccm}{Cpm})\times\frac{1}{\beta-1}}{\frac{1}{\beta}} \tag{1.14}$$

1.5.3 Ordering model

In the two previous models, it is considered that a spare part is always available when the unit fails. Therefore, if the spare parts are kept, there is a whole stock management policy. The cost of storage can be very high.

If the system fails very rarely, orders are realized whenever needed to avoid this storage cost. We use the same notations in .

-If the unit does not fail until t_0, the command is carried out at t_0 and arrives after a particular regular lead time L_r. If the unit has already failed at $t = t_0 + L_r$, it is replaced. Otherwise, if it is still operating at $t = t_0 + L_r$, it is replaced anyway as a preventive measure. The optimal execution date of the PM is $t_0 + L_r$.

-If the original unit breaks down before t_0, the expedited order is carried out at this failure moment, and the spare part is replaced after its receipt, within a time limit for obtaining it. The regular order is therefore not made. We remind the reader that one cycle represents the time between two replacements. The costs used are:

- C_e: the expedited ordering cost, which is the cost of the accelerated order if the failure occurs before t_0.

- C_r: the regular cost, when the failure occurs at $[t_0, +\infty[$. We got to t_0 before it happened.
- k_f: the system down (shortage) cost per unit of time or the cost of under storage per unit of time.
- Cw: the operation cost per unit of time or the production cost per unit of time.
- s: the residual cost per unit of time (salvage cost) s<0.

The exepected cost per unit time in the steady state is [65]:

$$CM(t_0) = \frac{CeF(t_0)+C_r(1-F(t_0))+k_f[(Le-Lr)F(t_0)+\int_{t_0}^{t_0+L_r} F(t)\,\mathrm{d}t]+w\int_0^{t_0+L_r}(1-F(t))\,\mathrm{d}t+s\int_{t_0+L_r}^{+\infty}(1-F(t))\,\mathrm{d}t}{(L_e-L_r)F(t_0)+Lr+\int_0^{t_0}(1-F(t))\,\mathrm{d}t} \tag{1.15}$$

1.5.4 Inspection model

They are some systems where failures, generally not serious ones, are not detected immediately after their occurrence. Instead, the failure is discovered only at the moment of inspection (partial disassembly of the system, for instance). Two cases govern the decision-making for this kind of situation. First, the failure is detected rapidly if many inspections are executed, but we incur a high inspection cost [102]. Otherwise, if there are only a few inspections, a considerable time separates the occurrence of the failure and its detection, and a high cost of failure is incurred [102]. The goal of the model here is to find the optimal or near-optimal inspection expected cost while considering both the cost of inspection and the cost of system failure.

The most famous model presented in the literature is the BHP model by Barlow, Hunter, and Proschan proposed in 1963 [1]. The authors considered a strong hypothesis on the economic consequences of the non-detection and proposed a complex algorithm for an optimal resolution. Indeed, their algorithm requires many trials and errors to choose the time for the first inspection and the restriction on f(t) is strict. Nevertheless, the BHP model remains one of the foundation models that are now the basis of most other inspection models. Many

authors proposed improved and simpler procedures to obtain near-optimal solutions based on the BHP model. The BHP model can also be applied to condition-based maintenance if we consider the time elapsed between crossing an alert threshold and the appearance of an unacceptable state [62].

The BHP model to find an optimal inspection policy uses the following costs: the cost c_i of an inspection and the cost of failure per unit time kf. For a component that follows a lifetime distribution F(t) whose density function is f(t). We assume that the component is inspected each period t_k with (k=1,2,3,...). The policy is terminated when an inspection has led to a component's failure.

The total expected cost is [102]:

$$CM = \sum_{k=0}^{+\infty} \int_{t_k}^{t_{k+1}} [c_i(k+1) + kf(t_{k+1} - t)] \, \mathrm{d}F(t) \tag{1.16}$$

Barlow et al. [1] defined when the optimum exists and demonstrated that the inspection dates t_k form a decreasing sequence $t_{k+1} - t_k$ and verify the relation [65]:

$$t_{k+1} - t_k = \frac{F(t_k) - F(t_{k-1})}{f(t_k)} - \frac{c_i}{k_f}, k = 1, 2, 3, ... \tag{1.17}$$

Where $f(t)$ is a Pólya frequency function of order two with $f(t+\Delta)/f(t)$ is strictly decreasing for $t \geq 0$, $\Delta > 0$, and with $f(t) > 0$ for $t > 0$ and $t_0 = 0$.

1.6 Vehicle routing problems

A good organization of deliveries of finished products or pickup of raw materials or products is often a significant issue in terms of costs, quality of service, and time. The routing problem arises for distribution and collection organizations. Nowadays, solvers for this kind of problem should be embedded in an Interactive Decision Support System (IDSS) to deal with tactical issues daily or periodically and improve delivery or pickup management. An example of this thesis that illustrates this point is the routing and planning for technicians or maintenance specialists who have customers to visit and operations to execute, which vary each period. This section aims to overview the evolution of the routing problems since the original problem. In all VRP variants, a tour is defined as a set of single

visits to delivery or pickup point. The problem is to determine the sequence of visits while considering several constraints that stem from real-life settings.

1.6.1 Formulation

They are several formulations to model the VRP that can be classified into the following categories:

Vehicle flow formulations are the most used in the literature. They generally use two or three index formulations. They use a binary variable $x_{i,j}$ to count the number of times that arcs (i, j) are traversed for the two index formulation or a binary variable $x_{i,j,k}$ to store the number of times that arcs (i, j) are traversed by vehicle k. The constraints differ based on the formulations. For an example of the two-index formulations, we suggest the paper of Laporte et al. [48]. The MTZ formulation of Miller is the basis of the two-index vehicle flow formulations for the VRP. The multi-TSP is the basis of the three-index vehicle flow formulations of the VRP. Extensions to the three-index formulations can be found in the contributions chapters of the thesis. Indeed, we have chosen to formulate our models based on this first category, especially the three-index formulations.

Commodity flow formulations extend the vehicle flow formulations by adding additional constraints and a flow continuous variable f_{ij}^k expressing the amount of the commodity or the demand k that travels from node i to node j. The objective function is the same as the previous formulations. They were proposed initially for VRP by Garvin et al. [49] for the CVRP. For the TSP, we distinguish the single-commodity flow formulation (SCF) of Gavish and Graves [41] and the multi-commodity flow (MCF) formulation of Claus [54]. The difference is that in the multi-commodity flow formulation (MCF), there are k commodities instead of a unique commodity. The MCF formulation of TSP has $O(n^3)$ variables and $O(n^3)$ constraints. Therefore, it is considered the strongest among the TSP compact formulations [41].

Set partitioning formulations are very different compared to the two previous formulations. The objective here is to find a set of routes that minimize travel

cost while forming a feasible solution ($\min f = \sum_{r=1}^{R} c_r x_r$). The cost of each route r is c_r. The binary decision variable x_r equals 1 if the route r is included in the solution and 0 otherwise. An additional variable a_r^i determines whether a customer i is visited by vehicle or route r. An example of this formulation could be found in the work of Balinski, and Quandt [40] for the CVRP.

1.6.2 Traveling salesman problem

The Traveling Salesman Problem (TSP) defines the shortest tour to visit n cities, each only once, before returning to the city of departure. The solution to the problem is an optimal sequence selected from a list of n pairs "city of origin i - city of destination j". As aforementioned, each city has to be retained once and only once as the city of departure or arrival. Despite the simplicity of the problem description, it is an NP-hard optimization problem. An obvious approach to solving the TSP is enumerating all the possible routes and choosing the shortest route. This process is achievable only for small size problems since, after choosing one city arbitrarily, it remains n-1 cities to choose from for the second city. Then, after picking the second city, it remains n-2 possibilities, and so on. The number of the possible couple "origin, destination" is very high and is equal to $(n-1)!$.

The classical formulation of standard TSP has been proposed by Dantzig et al. [51]. It uses a binary decision variable x_e that equals 1 if the edge e belongs to the tour and 0 otherwise. The objective of this formulation is $\min f = \sum_{e=1}^{E} c_e x_e$ where c_e is the cost. Two sets of constraints ensure that each tour uses exactly two of the edges for sub-tour elimination. Other compact formulations appeared later, such as the MTZ formulation presented in the following paragraph, the time staged formulation (TS), and the commodity flow formulations presented in the previous subsection. We suggest the survey of Letchford et al. [41] for the different compact formulations of the TSP.

The MTZ formulation [50] [52] of the TSP is linear and utilizes a binary decision variable $x_{i,j}$ that equals 1 if the arc (i,j) is selected and 0 otherwise. The

formulation presented here [50] is nearer to the formulation of VRP used in our thesis.

$$\min f = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}x_{i,j} \tag{1.18}$$

S.t.

$$\sum_{j=1}^{n} x_{i,j} = 1, \quad \forall i = 1,2,...,n, i \neq j \tag{1.19}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1,2,...,n, i \neq j \tag{1.20}$$

$$\theta_i - \theta_j < n(1 - x_{i,j}) - 1, \quad \forall 1 < i,j \leq n, i \neq j \tag{1.21}$$

The condition $i \neq j$ ensures that it is forbidden to go from one city to itself. It could be deleted by fixing $c_{i,j} = +\infty$ in the input data whenever $i = j$. Since we seek to reduce costs, such paths would be avoided. Having only the objective (1.18) and the enter and exit constraints (1.19-1.20), we find the formulation of the assignment problem. A constraint for sub tours elimination has to be added (1.21). In the TSP and most VRPs, the hypothesis of the symmetrical matrix of time or distance is considered $c_{i,j} = c_{j,i}$. The number of variables, in this case, is divided by 2. The number of possible tours becomes $\frac{(n-1)!}{2}$. The arrival time θ_i represents here the order of the city i in the tour. The constraints (1.21) with the variables of the tour order have been added to ensure that there are no closed paths present in the solution apart from node 1.

1.6.3 Variants of vehicle routing problems

The VRP is a problem that aims to design a set of optimal routes from a central depot (or several depots, in the case of multiple depots VRP) to several geographically distributed customers while satisfying some constraints. Due to the importance of the VRP and its applications, it has always been the subject of intensive research effort. The VRP originates from the Traveling Salesman

Problem (TSP). Several extensions and variants were proposed to include real-world features and accommodate specific constraints on the delivery or pick-up process, such as time windows for customers, stochastic travel times, heterogeneous vehicles, and so on. We present herein some essential variants. The main extension from the TSP to VRP is the consideration of several vehicles that can serve the customers in parallel through multiples tours.

Capacitated Vehicle Routing Problem (CVRP)

The most basic variant of VRP that has followed the TSP is the capacitated VRP (CVRP). It considers the vehicle capacity constraints and to the classical VRP constraints. The distribution starts from one unique depot. As in the classical VRP requirements, each route begins and ends at the depot. In addition, each customer is visited only once by one of the vehicles. Finally, the demands q_i of all customers visited i by the vehicle k should not exceed the vehicle's capacity Q_k, $\sum_{i=0}^{n}\sum_{j=1}^{n+1} q_i x_{i,j,k} \leq Q_k \forall k \in \mathcal{K}$.

Periodic vehicle routing problem (PVRP)

In the PVRP, K vehicles can be used to serve the demands of a set of customers in several periods. A customer may need many visits per horizon, period one and two, for instance, giving two available required services. The period is often a day, but the model designer can divide the periods over the horizon depending on the problem at hand. In PVRP, all vehicles start from the depot and return to the depot. Such as the classical CVRP, capacity and travel duration must be respected. Some specific constraints to the periodic VRP also have to be met. In each period, up to K vehicles can be used. A customer has to be serviced once in the chosen period. This problem aims to design for each period routes that minimizes the total travel cost subject to the constraints mentioned above. The index of periods is usually added to the index of variables to deal with the PVRP. Its main advantage is keeping the problem discrete, which is generally easier to solve than a mixed program. However, its main drawback is that the number of variables and constraints increases with the index added. Both an assignment

problem and a routing problem are involved in PVRP. Therefore, the use of exact methods to solve it consumes more time. There is also a particularity in solving this kind of problem. Two main types of solving approaches have been used. The first one starts by assigning customers to periods and then solves the vehicle routing problem (VRP) for each period. This approach has been adopted in [55]. The second approach exploits the same steps but in reverse order. Customers are firstly assigned to vehicles, and then routes are built and assigned to periods [56]. For the vehicle flow formulation, particularly the three-index one, the decision variable used is $x_{i,j,k,t}$.

Heterogeneous Vehicle Routing Problem (HVRP)

In the CVRP, the vehicles are assumed to be identical and have the same capacity and cost sets. This assumption rarely holds in reality. Whereas several VRP variants consider homogeneous vehicles, real-world problems often deal with heterogeneous vehicles within a fleet. In the literature, two categories of HVRP can be found: heterogeneous fleet VRP (HFVRP) and fleet size and mix VRP (FSMVRP). In the HFVRP, the composition of the fleet is provided [9]. The fleet has a fixed number of vehicles of each type and different capacities. In FSMVRP, an infinite number of vehicles of each type is considered [10]. The heterogeneous fleet assumes M types of vehicles. Every kind of vehicle $m \in M$ is composed of a number k_m of vehicles of capacity Q_m. Sometimes in the literature, the traveling cost may depend on the type of vehicle used $c^m_{i,j}$. In the FSMVRP variant $k_m = +\infty$.

Multi-Depot Vehicle Routing Problem (MDVRP)

Most VRP variants contain only one depot from which all the vehicles leave and return. In the Multi-Depot VRP, several depots are considered. This variant may have many sub-variants. Some examples are that the same depots do not serve the vehicles, a limited number of vehicles are assigned to the depot, etc. Crevier et al. [11] proposed, for instance, an extension in which the replenishment of vehicles is realized at intermediate depots in their routes.

Vehicle Routing Problem with Time Windows (VRPTW)

In the Vehicle Routing Problem with Time Windows (VRPTW), each vehicle has to deliver or pick up an order from a customer within a specific time interval imposed by this customer called the time window. Moreover, each customer is associated with a service duration. The vehicle must arrive at a customer i before time b_i, but the customer will not be serviced before the beginning of the time window a_i. Although, the vehicle can arrive early. It corresponds to reality where generally, customers have a specific service time where they can receive or give the order (availability of agents, the opening time of the plant, etc.). A solution is feasible only if the time windows constraints are respected. This formulation of time windows is hard. In some cases, this restriction is too tight to respect in reality. Soft time windows are considered in this case where the violation of the time windows is allowed with a penalization in the objective function [15]. More complex time windows can be considered. Customers can have multiple disjoint time windows in which they can be serviced [74] [75]. In this case, a unique time window per customer needs to be picked, and a vehicle that arrives between two-time windows must wait until the following time window. Here again, if we include a cost $c(s_i)$ that depends on the start time of the service s_i of customer i, the time window is said to be soft. For solving issues, this cost is generally chosen to not increase with time. If the opposite case is needed, $c(s_i)$ can not be solved efficiently [38]. Ioachim et al. in 1998 [39] proposed an algorithm for solving the linear case. The VRPTW has been widely studied in the literature due to its important applications in reality since the 1970s.

Vehicle routing problems with profits (VRPP)

When we deal with VRP with profit, the problem is called depending on the number of vehicles used, an Orienteering Problem (OP) if we have one vehicle, and a Team Orienteering Problem (TOP) if at most K vehicles are used. The profit value associated with each location makes it more or less attractive. The OP is also called the selective traveling salesman problem [12] [13] and the maximum collection problem [14]. The difference with the VRP is that each customer or location is associated with a profit. The goal is to find a route

visiting only some customers while maximizing the total collected profit subject to the constraint to satisfy the maximum route duration. The TOP is the same with multiple routes. It is similarly called the selective VRP. Here again, the TOP originates from the OP. It is more straightforward in the formulation and resolution than VRP since it includes fewer constraints. Recently, the literature includes progressively more constraints to this variant, such as additional time window constraints (TOPTW).

Rich VRP

RVRP includes many constraints and preferences inspired by real-world applications. The constraints can be diverse to represent the real-life problem studied, making the problem complex and highly constrained.

VRP with multiple synchronization constraints

VRPMS deals with more complex temporal constraints than classical vehicle routing problems. In VRP, we seek only to find which vehicle is assigned to a given customer. VRPMS includes additional synchronization requirements. VRPMS are used whenever two or more vehicles must be used and synchronized to perform a given task [68]. It is used in multiple applications such as home health care scheduling.

Dynamic and/or stochastic vehicle routing problems

Dynamic VRP: The planning is dynamic whenever we can update the plan during the execution of this latter as new information is revealed. Dynamic VRP is also called online or real-time VRP [43] [44]. In most cases, the real-time arrival of customer requests is considered the factor that makes the problem dynamic. More recently, demands that are dynamically revealed over time for a set of customers are considered. An example inspired by dynamic VRP featuring this last aspect is treated by Pillac et al. [81]. They highlighted an adaptive neighborhood search-based approach to solve the dynamic routing and scheduling technicians' problem (DTRSP). The algorithm computed an initial solution and then re-optimized it when a new request arrived over time. The

objective is to assign the technicians with the right skills, tools and spares to the maintenance operations at the lowest overall cost. In addition, technicians were allowed to replenish tools and spares at the depot when needed to handle more requests. Indeed, an approach to solving dynamic VRPs can first solve a static problem assuming the current state information for each period. The planning change can be realized as soon as the new information is available or at fixed time intervals. We suggest the survey of Pillac et al. [45] for more details about dynamic VRP and the common scenarios of dynamism.

Stochastic VRP: When uncertainty is included in some inputs of the problem, the VRP is called a stochastic VRP. The most interesting stochastic VRP is VRP with stochastic travel time and service time (VRPSTT) [47]. The most studied variant is the VRP with stochastic demands [46]. The presence of customers is stochastic in the VRP with stochastic customers (VRPSC). Each customer requires a visit with a given probability. In the VRP with stochastic demands (VRPSD), customers demands are random variables with probability distributions. The uncertain information could be obtained from historical data generally kept in most companies. The stochastic VRP variant is essential since it enhances the robustness of a solution. It is also worth mentioning that the environment is uncertain in real-world scenarios.

Workforce scheduling

It consists of constructing work timetables for the personnel to permit the organization to meet the products and services demand. It is the variant involved in this thesis. It aims at scheduling the workforce to perform tasks. The problem is also called the Field Workforce Scheduling, the Technician Routing and Scheduling Problem (TRSP) [81], Technician and Task Scheduling Problem (TTSP) [77], Service Technician Routing and Scheduling Problem (STRSP) [78], Workforce Scheduling, and Geographically Distributed asset Maintenance problems (GDMP) [84]. The name of the variant changes with the constraints considered. These constraints can be [123] : routing, teaming, single period/ multi-period, time windows, precedence, priority, tools, spare parts, unavailability, and dynamism. We recommend the short survey of Khalfay et

al. [123] for more details about the constraints considered. Ernst et al. [67] suggested a taxonomy for classifying the literature on workforce scheduling. They presented six modules related to the workforce scheduling process. Some of the modules may be required depending on the application. They are [67]: demand modelling, shift scheduling, line of work construction, shift assignment, task assignment, and staff assignment. Each module is detailed in their survey. A complete literature review of this variant will be presented and analyzed in the contributions sections.

The core of our problem consists of three issues: vehicle routing, employee scheduling, and maintenance planning. These components are integrated with multiple constraints and requirements. The VRPTW is the variant used as the basis of our mathematical formulation. In addition, resolution methods to solve it in the literature provided us with several insights into the parameters' setting of our algorithms. Along with periodic VRP, it has constituted the reference model to approach Workforce and Technicians Scheduling.

1.7 Concluding remarks

This chapter reviewed all aspects related to the Joint Maintenance Scheduling and Workforce Routing Problem. This literature survey was divided into two main parts. The first part of the chapter surveyed maintenance from its foundation until now. Maintenance orientations have led in the literature to the definition of essential concepts in maintenance, and some of them even evolved to become maintenance strategies. These strategies were discussed with a particular focus on Reliability Centred Maintenance. Basic formalized models for optimization maintenance intervals used in Reliability engineering were then presented. Finally, the chapter's second part illustrated the difference between the main variants of the Vehicle Routing Problem (VRP), with an emphasis on Vehicle Routing Problem with Time Windows (VRPTW) being the variant involved in this thesis.

Chapter 2

Optimization and resolutions methods

Outline of the current chapter

2.1 Introduction

Optimization is the process of obtaining the best result under given circumstances. It aims at finding the maximum or minimum of an objective function or several objectives functions subject to no or several constraints. Chapter 2 is dedicated to the literature related to approximate resolution methods. The chapter starts with an overview of the different classes of optimization problems to identify the problems studied in this thesis. Then, existing heuristic and metaheuristics are discussed. The chapter finally surveys multi-objective optimization classes and techniques.

2.2 Categories of optimization problems

Whenever we have an optimization problem at hand, it is essential to identify which category this problem belongs to before solving it. Each algorithm in the literature is developed for a specific class of problems. The classification of optimization problems varies, but we can cite the following six main categories:

Continuous optimization problems, discrete optimization problems, and mixed problems : The optimization problem is continuous when the variables are real numbers, while the optimization problem is discrete when the decision variables are discrete (integer or binary). Discrete optimization problems are more difficult compared to continuous optimization problems [6]. They are classified in the category of combinatorial optimization [6]. Their difficulty lies in the fact that the search is not tractable. Among combinatorial optimization problems, we can cite: the Travelling Salesman Problem (TSP) and the knapsack problem [7]. An optimization problem combining continuous variables and discrete variables is called mixed. Mixed problems combine continuous variables and discrete variables. They also fall into the category of combinatorial optimization. Examples of such problems are Vehicle Routing Problems With Time Windows (VRPTW).

Nonlinear optimization problems and linear optimization problems : Nonlinear Programming (NLP) is characterized by an objective or at least one of the constraints that is a nonlinear function of the decision variables. On the contrary, a Linear Program (LP) has linear objectives and constraints. For example, the VRPTW is a Mixed Integer Linear Program (MILP). In the domain of operational research, there is a combination of these different categories most of the time. It could be beneficial also to determine if the objective function of the problem is convex or not convex. Convex problems are generally easier to solve.

Optimization problems with and without constraints : In an optimization problem without constraints, we minimize only the objective function (assuming we have a minimization problem) without considering any constraints on the decision variables. A problem with constraints has to consider equality and inequality constraints on decision variables. The unconstrained problems are generally easier to solve than constrained problems. It is possible to eliminate a constraint oe several by substitution in the objective function to transform a constrained problem into an unconstrained problem and solve it as this latter category of problem. This method is called the penalty method. An example of using this methodology in the VRPTW, for instance, is accepting all the solutions in the neighborhoods and then penalizing infeasible solutions with a hefty penalty in the objective function. Other methods, such as barrier and primal methods, can be applied to solve constrained optimization problems.

Single objective or multiple objective optimization problems : A single objective function defines single-objective problems. In multi-objective problems, several criteria have to be considered simultaneously. There is a compromise to choose solutions that represent the best trade-off. The objectives are, most of the time, contradictory objectives, which makes their aggregation not efficient. Mono-objective problems are simpler to solve and consume less time than multi-objective problems. It is possible to transform a multi-objective problem into a mono-objective problem by formulating one or several objectives as constraints. This approach is interesting and can considerably reduce the computational time, but it requires an effort in reflection and modeling. An example of such an

approach can be found in our previous work [103]. It is also possible to reformulate a multi-objective problem as a mono-objective problem by aggregating the different objectives. This last approach is, however, not necessarily efficient, especially if the objectives are not of the same scale or are in conflict [58].

Dynamic and static optimization problems : In static problems, we search for the values of variables that minimize or maximize the objective function. On the other hand, in dynamic problems, information can be revealed over time so that each new information is used in the optimization gradually. It is also called Intertemporal optimization. This term is defined by [37] as the category of optimization that considers the different operating conditions that a system encounters in its lifetime. The mode of operation needs then to be determined at each instant of time. The variables are functions of time that minimize or maximize the objective function in both periodic problems that are static and dynamic. Intertemporal static (or pseudo-dynamic) optimization, where the initial problem is transformed into a series of static optimization problems is used to deal with periodic problems [37]. An example of this kind of problem is provided by [37], the operation optimization of an energy system under time-varying conditions. However, dynamic problems differ from periodic problems by definition, a recursive relation between the previous and present time is present in the modeling. They have direct or indirect interdependency among the modes of operations. Indeed, dynamic programming is characterized by differential equations that we need to find in the modeling: initial conditions and the recursive relation. An example we provide for this category of problems is related to production systems where stocks are needed, making them dynamic. The flow conservation constraint is a relation that connects the value of the available stock $S_{i,t}$ of a product i at the instant t and the value $S_{i,t-1}$ of the stock for the same product at the instant t-1. The initial stock is known as $S_{i,0}$. Whenever we have information about the demand of the product i of the instant t and the supply at this instant, we can obtain $S_{i,t}$.

Deterministic or stochastic optimization problems : Deterministic optimization problems consider that the data is known. In stochastic optimization

problems, there is some uncertainty in the input data. Some components of the problem are random. Therefore, a stochastic approach may be relevant in some real-case situations. In stochastic VRPTW, the demand is usually considered stochastic, which may be appropriate because it depends on the customer. The variant Vehicle Routing Problem with Stochastic Demand (VRPSD) is studied in this case. Stochastic travel time, service time, or both can be considered in VRPTW to adjust to real case situations. It is the most interesting variant among stochastic VRP. Finally, VRP with stochastic customers is another variant studied in the literature. Here each customer has a given probability of requiring a visit. The main methods for handling uncertainty are [47]:

- Stochastic programming with recourse (1955)
- Chance-constrained programming (1959)
- Dynamic programming (1958)
- Robust optimization (recent)

Stochastic optimization includes both stochastic programming with recourse and chance-constrained programming. Uncertainties are modeled with components that follow a distribution of probability. The stochastic optimization approaches can only be used when we can obtain historical data. Stochastic programming with recourse divides the problems into different stages, while the information is revealed gradually in each stage. The simplest version is two-stage stochastic programming, in which the first stage is solved. For the second stage, penalties (recourse) are added to the objective function to account for the realization of uncertainty. Generally, a simulation approach based on Monte Carlo is used to solve the problem. Chance-constrained programming consists of authorizing constraints to be satisfied with a given probability. It is solved similarly to stochastic programming with recourse using Monte Carlo simulation. In both approaches, trying to obtain the recourse value or working on the probabilistic constraints to get an exact analytical value is the best way to solve the problem and the less time-consuming. Simulation approaches are easiest to use but are time-consuming and should be favored when the expression is

very complicated or will require a lot of reflection and time. The recourse value in the objective function for this kind of problem is an expected value.

When it is difficult to obtain exact information or any historical data on the inputs of the problem, robust optimization is generally used. This aspect is what makes it more practical. Robust optimization has appeared recently and can be a really interesting approach to dealing with stochastic problems. Robust optimization defines a set of scenarios for the possible realization of uncertain parameters. Robust optimization looks for the solution that provides the best "worst case". It uses a min (max cost) or max (min regret) as an objective function. Its main drawback, however, is that solutions can be overly pessimistic [47]. In Dynamic programming, there is a time decomposition of the problem according to stages [37].

2.3 Heuristics methods

Heuristics are problem-specific techniques (problem-dependent) used to generate good solutions for a particular problem. They try to exploit the particularities of this problem to take full advantage of them. Unfortunately, they often get trapped in a local optimum because of their greedy nature and thus fail to obtain the global optimum solution. As a result, there is no guarantee of the optimality of the solutions found. They can be divided into three categories:

2.3.1 Constructive heuristics

Constructive heuristics build a solution from scratch, step by step, according to a set of rules defined beforehand. They are divided into two categories: sequential heuristics where sub-problems are dealt with one by one, and parallel heuristics where all sub-problems are dealt with simultaneously. For the case of the VRP problem, sequential heuristics construct one route at a time while parallel heuristics construct many routes at the same time [115] [36]. In sequential heuristics, an additional vehicle is only considered if needed. For instance, in the case of Capacitated VRP (CVRP), we add another vehicle to the solution if the current one can not handle more requests. On the other hand, the number of

vehicles needs to be specified in advance for parallel heuristics. The estimated number of vehicles can be increased later on if required [36]. We suggest the paper of Potvin and Rousseau [33] for further details about sequential and parallel heuristics.

Today, constructive heuristics are generally used to generate a good initial solution for improving heuristics and metaheuristics. They are also used as part of some metaheuristics or hyperheuristics such as Adaptive Large Neighborhood Search [113] [115]. Examples of constructive heuristics for VRP are: The Clarke and Wright algorithm [29] and Insertion heuristics [36].

The Clarke and Wright algorithm is a saving greedy heuristic designed for the first time by Clarke and Wright in 1964 for CVRP [29]. It consists of merging all two available routes whenever the resulting route is feasible based on the distance saving value. The saving values are ordered in non-increasing order. The routes that are merged first have the more significant saving value. The process continues until there are no routes to be linked to form a feasible route that satisfies the load constraint [34]. The complexity of this procedure is $O(n^2 log(n))$ time, but several authors have proposed later other variants of the algorithm for its reduction.

The sequential insertion heuristics consist of inserting one unrouted customer between two customers in the progressing route considering the cost measure. The insertion heuristics start with a seed customer in the considered route. A new route is considered when no more customers can be added to the current one. Three sequential insertion heuristics have been proposed by Solomon [36]. They have the same operating principle but differ in the definition of the cost measure used to select the next customer to be added in the current route. The I1 heuristic is the most used and known one. Here, the customer is inserted in the best position that minimizes the first cost measure. This latter, defined by Solomon, is obtained according to Solomon's definition [36], by calculating the extra travel distance and time delay when a customer k is inserted between i and j. The I2 heuristic considers the total route distance and time when inserting a customer k. Finally, the I3 heuristic differs also in the cost

measure where the urgency of servicing the customer k is included in the cost term [36]. These heuristics can be easily tailored to deal with multiple routes.

The parallel construction heuristics build several routes at the same time. Multiple routes are initialized with customers. Routes are added later if the estimated initial number of vehicles is insufficient to construct a feasible solution. Like sequential heuristics, the customer whose insertion produces the slightest increase in the solution's cost is chosen for insertion. The Basic greedy insertion heuristics that have been used by Ropke and Pisinger for the PDPTW [113] and the VRPTW [115] are extensions of the I1 heuristic of Solomon. Potvin and Rousseau proposed in 1993 regret heuristics for the VRPTW [33]. They are based on the I1 insertion cost measure. This kind of heuristics is peculiar in considering the regret value of not inserting the customer in its best route. The regret value is computed using the cost value mentioned above. Regret heuristics have also been used by [113] [115] to solve the PDPTW and VRPTW when they defined the Adaptive Large Neighborhood Search. Trick [32] used these regret heuristics for the generalized assignment problem.

2.3.2 Improvement heuristics

Improvement heuristics start by building any feasible solution and improve it by applying successive small changes and moves. They look for better neighboring solutions by perturbing the current solution at each iteration. The initial solution can be generated randomly or using a constructive heuristic. In the second case, we are designing a compound heuristic. Compound heuristics first have a constructive phase followed by an improvement phase. The improvement phase consists of applying a neighborhood move to the current solution to yield a new solution within its neighborhood. Improvement heuristics for VRPs perform moves on either a single route (intra-route moves) or multiple routes at a time (inter-route moves).

Intra-route algorithms (single route) : The $\lambda-opt$ was proposed by Lin in 1965 for the TSP [28]. It is the most famous move. A $\lambda - opt$ move eliminates λ edges

and reconnects the λ resulting paths in a different way to obtain a new tour. First, the λ edges that produce the shorter tour are selected among all combinations of edges. This procedure is then repeated until no such combination of edges is found. The process is also called $\lambda-exchange$. The route that can not be improved more by this move is called $\lambda-optimal$. The larger the λ, the more chance the designer has to yield an optimal route [26]. According to Helsgaun [26], most of the time, the value of λ=2 and λ=3 are not exceeded in the literature to avoid huge computational time. Indeed, the procedure requires $O(n^{\lambda})$ time [36]. Ideally, we need to find a λ that would yield to a high-quality solution in a reasonable time. Lin and Kernighan [27] modify the value of λ dynamically throughout the search. Their procedure is called "variable-depth exchange" and is very efficient in solving the TSP. Another extension was proposed by Or in his Ph.D. thesis in 1976 [21] and is called the $Or-opt$ move. $Or-opt$ tries to improve a route by moving consecutive nodes in another position in the same route. It consists of moving strings of 3, 2, or 1 consecutive vertices elsewhere in the tour. It is a restricted form of 3-opt where the orientation is not changed. Or-optimality requires $O(n^2)$ time.

Inter-route algorithms (multiple routes) : VRP is an extension of TSP that include multiple routes. Hence, inter-router operators were defined to deal efficiently with the VRP in the literature. Potvin and Rousseau [24] extended the $2-opt$ to consider two routes instead of one. The resulting move is called $2-opt*$ exchange. The $2-opt*$ exchange heuristic attempts to create two new tours by removing two edges (one from each tour) and reconnecting the resulting parts without any change in the tours' orientation [36]. Similar moves are called string crosses since two strings of vertices are exchanged through the crossing of two edges in two other tours. We can also find basic moves. The swap move consists of exchanging two operations from different routes. The operator is also called 1-interchange [30]. The intra-route version of this operator is called the exchange move to specify that the exchange occurs on the same route. Different variants exist for the swap operator. The generalization is called string exchange when two strings of k vertices are exchanged between two routes. The insert operator, also called shift or relocate operator, removes a customer from its position and

inserts it in a different one in another route [30]. The operator can be used to relocate a customer within the same route. The generalization of the operator to include more than one edge is called string relocation. It consists of moving a chain of customers (k vertices) from one route to another. Worthy of note that the basic moves remain very efficient in solving the VRP. They are therefore used frequently.

2.3.3 Hyperheuristics

A hyperheuristic is a heuristic that selects heuristics [5]. In hyperheuristics, several rules are defined to choose the most suitable Low-Level Heuristic (LLH) for a given instance, problem by either scheduling the order of the LLH or using adaptive or machine learning mechanisms. The latter approach is interesting since the design of the algorithm does not require experts' intervention [31]. The definition of hyperheuristics is close to the meaning of metaheuristics. They also share the same primary goal. The main objective of hyperheuristics, similarly to metaheuristics, is solving many different problems rather than one specific problem. They are generic methods. They differ notably in the search phase. Indeed, metaheuristics explore the search space of the solutions to a given problem, while hyperheuristics exploit the search space of heuristics [35].

2.4 Metaheuristics methods

A metaheuristic is an optimization algorithm that aims to solve difficult optimization problems in several fields, such as operations research, engineering, or even artificial intelligence. There is generally no more efficient classical method known for these problems. Like classical improvement heuristics, metaheuristics aim to search in the solution space widely and thoroughly. They are often stochastic since they are based on probabilistic sampling and are inspired by natural systems. They offer the advantage of finding optimal or near-optimal solutions for complex problems in a significantly reduced computational time compared to exact methods.

Moreover, some metaheuristics employ heuristics methods by guiding them

over the search space. An example of such metaheuristics is Adaptive Large Neighborhood Search. Their perhaps only drawback is that there is no guarantee of optimum. On the other hand, metaheuristics are independent of the problem and can be applied to solve any optimization problem. Therefore, we can classify metaheuristics as local search-based methods and population-based methods.

2.4.1 Local search or trajectory-based methods

Local search-based meta-heuristics are similar to improvement heuristics. They iteratively explore the neighborhoods of a single incumbent solution to improve it. However, meta-heuristics have the particularity to include mechanisms to avoid local optima traps. Among the local search metaheuristics, we can cite the following:

Iterated local search (ILS) (Lourenco et al. (2003) [18]) : is a clever extension of stochastic hill-climbing with random starts. The only difference compared to this latter is that the selection of the point for each restart is not random but based on a modification of the best point found so far during the execution of the search.

Simulated annealing (SA)(Kirkpatrick et al. (1983) [61]) : The SA method was described originally by Metropolis et al. [57], but Kirkpatrick et al. [61] adopted it the first time for optimization. The SA algorithm always accepts better solutions, although worse solutions are also accepted with a certain probability. This criterion tries to reproduce the physical annealing process, especially in its second phase when the metal cools down slowly so that the atoms build a solid. The solution s' is accepted even if it is worse than the current solution s with a probability $exp(-(\frac{(f(s')-f(s))}{T}))$ where T is the temperature that decreases with a factor each iteration. This probability decreases at each iteration of the algorithm so that worse solutions are less likely to be accepted late in the process. Simulated annealing can be considered an extension of the hill-climbing algorithm. It is a sequence of transitions around solutions to improve the energetic objective function [121].

Tabu search (TS) (Glover (1986) [16]) : TS is a research technique whose principles were proposed for the first time by Fred Glover in 1986 [16]. It was then formalised in 1989 [17], and it became very classic in combinatorial optimization. Tabu search is based on exploring the search space while constantly seeking to improve the best current solution and keeping in memory the list of previous solutions or moves, thus guiding the search outside previously explored areas. It differs from simple local search methods by using a history of the solutions visited to guide the search towards promising directions. Tabu search is the only metaheuristic that integrates this memory mechanism into local search strategies. It, therefore, becomes possible to escape from a local minimum. In order to avoid cycling, the last visited solutions are banned and are stored in a tabu list for a certain number of iterations. The tabu list can be static, having the same length throughout the search, or dynamic, in which length changes during the search. The basic idea is inspired by research techniques used in artificial intelligence. It consists of keeping track of the past progress of the research process in one or several memories and using this information to orient the future progress of the method. The memory can be short term, intermediate-term and long-term [8]. The categories of memory in tabu search are:

- Short term memory: the tabu list includes only recently visited solutions to prohibit them and avoid cycling. It can also return to good components or solutions to intensify the search. This memory is used as an intensification mechanism.

- Intermediate-term memory: permits such as the short term memory to prevent cycling and exploit the neighboring areas of the best solutions found recently. This memory is used as an intensification mechanism. It can include diversification rules through an aspiration criterion.

- Long-term memory: gives the ability to go beyond the search space and explore other areas to diversify the search. Its main role is diversification through avoiding explored areas.

- Frequency-based memory: is used to tabu and not tabu solutions based on

the frequency of times they have been visited (the frequency of time each solution has been visited). It is used as a diversification mechanism with long-term memory.

The tabu search is a deterministic metaheuristic, but aspiration criteria can be introduced to make it stochastic. The aspiration criteria aim to determine when tabu restrictions can be overridden. They are several common methods for aspiration criteria.

Variable neighborhood search (VNS) (Mladenovic and Hansen (1997) [87]) : is a metaheuristic first proposed by Mladenovic and Hansen [87] in 1997. The main idea of VNS is the systematic change of neighborhood structures. Its goal is to find an optimal or near-optimal solution. The idea of VNS is motivated by the three following facts [88]:

- A local optimum of a neighborhood structure is not necessarily the same local optimum of a different neighborhood structure;
- A global optimum of one neighborhood structure is a local optimum considering all neighborhood structures,
- The local optima are relatively close to each other.

It starts from an incumbent solution s and applies two successive and essential mechanisms in each iteration. The first mechanism is the perturbation (or shaking) procedure essential for the VNS schema. It is used to escape from local minima and therefore ensures diversification. A local search follows it to improve the current solution. The local search procedure (intensification) aims to efficiently explore each incumbent solution's neighborhoods. Both first or best improvement acceptance strategies could be adopted in VNS local search. The same authors proposed several variants of VNS from 1997 to 2003. They are presented in [88]. The variants are:

- Variable Neighborhood Descent (VND): is a steepest descent heuristic (best improvement local search) that integrates the idea of changing neighborhoods during the search. It is based on the first fact mentioned above and is a deterministic procedure.

- Reduced Variable Neighborhood Search (RVNS): is a reduced version of the VNS procedure that does not include a local search procedure but only the shaking mechanism and the neighborhood change depending on whether the shaked solution improves the current solution or not. It is used generally when the speed to obtain the solution is more critical than solution quality.

- Variable Neighborhood Search (VNS): The RVNS that includes a local search after the shaking mechanism and before the systematic changes of neighborhoods around the local optimum found is called a Basic VNS.

- Skewed variable neighborhood search (SVNS): Some instances can have far away valleys containing optimal or near-optimal solutions breaking the fact three. Exploring large neighborhoods is important in this case. Skewed VNS is a modified VNS scheme that aims to explore valleys that are far apart from the current solution. SVNS recenters the search when a solution close to the best solution is obtained. This solution needs to be far from the current solution and maybe less good.

- General variable neighborhood search (GVNS): has the same schema as the basic VNS; however, it uses a variable neighborhood descent (VND) as a local search to explore several neighborhood structures at once. It can lead to an optimal or good near-optimal solution, but it generally consumes more time than the other VNS variants.

- Variable Neighborhood Decomposition Search (VNDS): extends the basic VNS into a two-level VNS by decomposing the problem. Compared to the VNS schema, the only difference is in the local search, where we explore a subspace to solve a subproblem instead of the whole solution space.

Adaptive Large Neighborhood Search (ALNS) (Ropke and Pisinger (2006) [113]) : Large Neighborhood Search (LNS) proposed by Shaw in 1998 is a method conceived to deal with a large-scale neighborhood or, in other words, a neighborhood of exponential size [120]. Searching in a large neighborhood may lead to better solutions [120]. It uses one destroy and one repair procedures

throughout all iterations of the search [120]. The downside of this method is the difficulty of guessing what the best destroy and repair operator is in advance. Another limitation is that the search might require different operators in the different phases of the algorithm's execution. For these reasons, Røpke and Pisinger proposed an Adaptive Large Neighborhood Search for the first time for the pickup and delivery problem [113]. The heuristics are selected using an adaptive roulette wheel mechanism from the set of destroy and repair methods that work effectively. The same authors propose some adaptations of the ALNS for several variants of the VRP [115].

2.4.2 Population-based methods

Local search-based methods generate a single solution at each iteration. On the other hand, population-based methods generate several new solutions from combinations of existing ones.

Genetic algorithm (GA) (Holland (1962) [19]) : belongs to a larger class of metaheuristics called Evolutionary Algorithms. This class of algorithms reproduces the process of natural selection. Parents are selected from the randomly generated population at each iteration using a roulette wheel or a tournament selection mechanism and create children using the crossover mechanism. This latter mechanism aims to make children share similar genetic information with their parents as in nature. The crossover can be a single point or multipoint. The children mutate with a given probability in order to diversify the search. The elitism mechanism is then applied if applicable. The best solution for each generation is picked to go to the next generation. This procedure mimics the natural selection process when only species which can adapt to their environment survive, reproduce and move to the next generation. The generation starts from parents to mutated children. Each individual of the population represents a solution in the search space. The solution is represented as a string of bits called the chromosome. The genes of the chromosome crossover and mutate. Vidal et al., in their survey [104] concluded that the Genetic algorithm shows slow convergence when applied to VRPs and does not provide good results. GA can provide out-

standing results for VRPs, but only if hybridized with enhancement mechanisms such as local search. To the best of our knowledge, a successful hybrid algorithm was proposed later to deal with VRPs. It is called a Memetic algorithm. GA has been successfully applied to many other optimizations problems and fields, such as artificial intelligence.

Ant Colony Optimization (ACO) (Dorigo (1992) [20]) : ACO is a social network method that is also inspired by a natural system, the behavior of actual ant colonies. It was first proposed to solve the Traveling Salesman Problem (TSP) by Marc Dorigo in his Ph.D. thesis in 1992 [20]. Ants secrete pheromone to communicate with each other and mark their paths when searching for food. The goal of these traces is to find the shortest path from nest to food or from food to nest. The more ants follow a route, the more attractive that route becomes. Details about the method can be found here [22].

2.5 Multi-objective optimization

Multi-objective problems are generally classified based on the role played by the decision-maker (DM) in the decision process. This classification focuses on how and when preferences of the decision-maker are integrated into the search process [101]. The four classes include no-preference, a priori, a posteriori, and interactive methods [100]. This section presents an overview of the different classes and the reference methods in the literature to solve multi-objective optimization problems.

2.5.1 No-preference methods

In no-preference methods, there is no decision-maker, the multi-objective problem (MOP) is solved, and a unique Pareto optimal solution is returned. It is the closest to the ideal point in terms of euclidian distance. In this class, the MOP is transformed into a mono-objective problem. These methods are simple and consume very few time compared to posteriori or intercative methods where the multi-objective formulation of the problem is maintained. The method of global

criterion is included in this category of methods. The objective of this method is [100]:

$$\min \sum_{i=1}^{k} (|f_i(x) - z_i^{ideal}|^p)^{\frac{1}{p}} \tag{2.1}$$

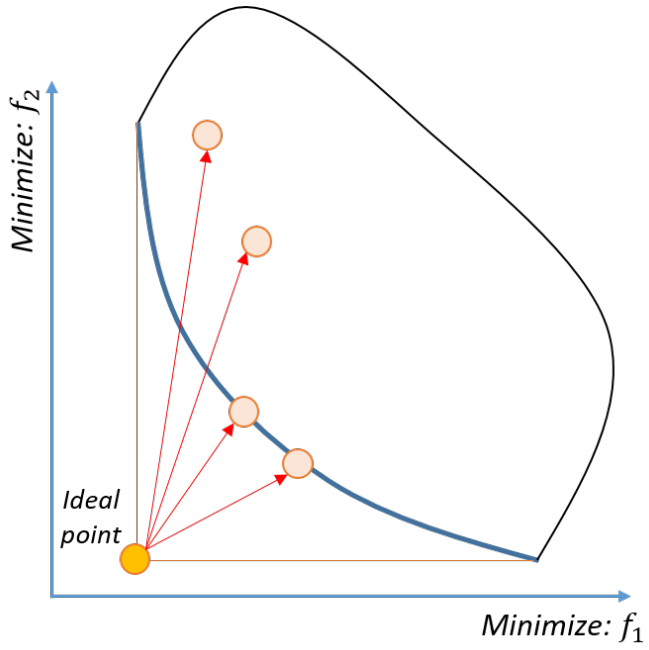

Figure 2.1: No preference method [23]

2.5.2 A priori methods

In a priori methods, a DM expresses his preference before optimization. In most a priori methods, the objectives are combined into a single objective. The most popular a priori method is called the weighted sum approach. Other methods included in this category are lexicographic ordering, goal programming [100], and the goal-attainment method [101]. In this class, the mono-objective formulation of the problem is maintained.

The weighted sum method: is a linear combination of weights. This method is simple and consumes very little time than a posteriori or interactive method. The reason is mainly that no front is stored at each iteration of the algorithm's execution. There are, however, many downsides of the weighted sum method [23]. Some of them are presented in [23]. It is possible to obtain the Pareto front, but weights must be varied, and multiple runs are required to create the Pareto Front (PF). Experts in the field should provide weights and their variations, and working with experts is not always possible. It is difficult to get a uniformly distributed Pareto front even if the weights are uniformly distributed. The objectives need to have the same scale. In the case of objectives with different scales, normalization of those objectives is required; otherwise, the solution obtained is influenced only by the higher-scale objective. Realizing normalization requires finding in advance the maximum and minimum of each objective. Two possible ways are conceivable: solving the mono-objective problem for each objective separately [90] or choosing the maximum and minimum value for each objective in the Pareto Front generated by the multi-objective algorithm [119]. The weighted sum method is formulated as follows [101]:

$$\begin{cases} \min \sum_{i=1}^{k} w_i f_i(x) \\ s.t. \quad x \in X \\ \qquad \sum_{i=1}^{k} w_i = 1 \\ \qquad w_i \geq 0, \quad \forall i = 1, 2, ..., k \end{cases} \tag{2.2}$$

2.5.3 A posteriori methods

The third class of MO methods includes the posteriori methods. In this class, the goal is to find all the Pareto optimal solutions. The decision-maker chooses one out of them in the second step. In this class, a multi-objective formulation of the problem is maintained. Examples of methods in this category are the epsilon constraint method, weighted sum approach with weights variation, Local

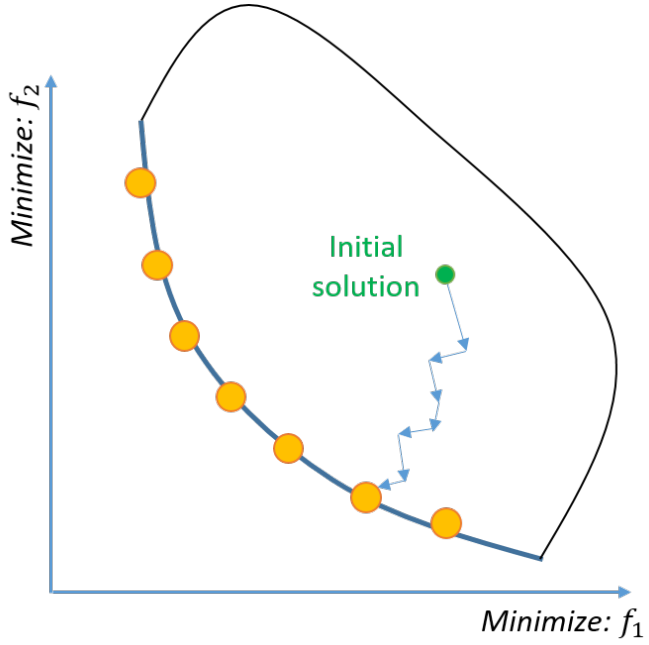

Figure 2.2: Weithed sum method [23]

search metaheuristics, and Evolutionary Multi-Objective algorithms (EMO). The posteriori methods are the most used. Among all these posteriori methods, Evolutionary Multi-Objective algorithms (EMO) are the most used and studied in multi-objective optimization. We generally seek two main goals: accurate convergence and uniform coverage. Indeed , we seek closer solutions to the optimal Pareto front that are as diverse as possible [23] [100]. The benefits of posteriori methods are the ability to obtain one Pareto front in a single run, and there is possibility of information exchange between solutions [23]. The most crucial benefit is obtaining any kind of front (convex or not) contrary to the weighted sum methods [23]. However, they are very time-consuming and can only address conflicting objectives [23]. Figure 2.3 illustrates the convergence and coverage for a Pareto front in the case of a bi-objective problem.

The formulation of the problem is:

$$\begin{cases} \min f_1(x), f_2(x), f_3(x), ..., f_k(x) \\ s.t. \quad x \in X \\ \qquad k \geq 2 \end{cases} \tag{2.3}$$

Initially, every individual of the population in the EMO algorithm is randomly generated. At each iteration, called generation of Evolutionary Multi-Objective algorithms (EMO), new solutions are generated from the population using reproduction operators (crossover and mutation operators) followed by a selection operator to select good solutions. As a result, EMO algorithms can generate diverse solutions, but there is no proof of convergence for EMO algorithms [100]. Figure 2.3 illustrates the functioning of such algorithms.

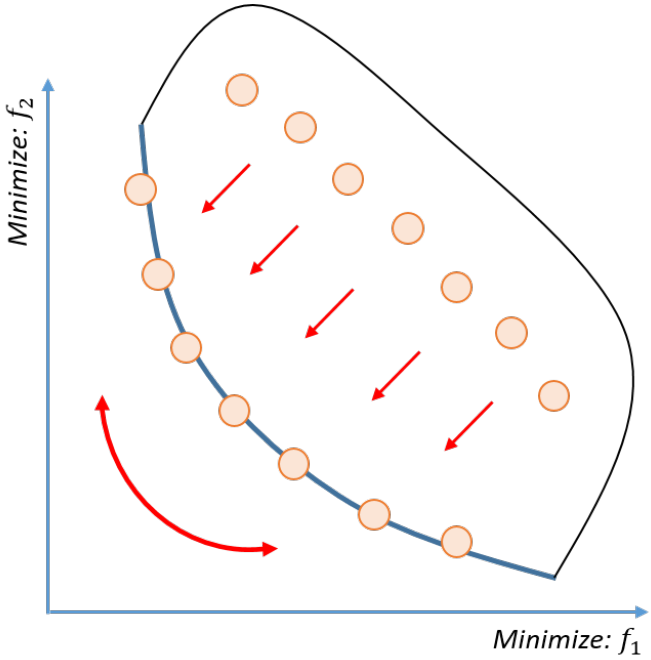

Figure 2.3: A posteriori method [23]

EMO algorithms are divided into three main groups [100]: aggregation, dominance, and performance indicator-based algorithms. Aggregation-based algorithms decompose the problem into several scalar optimization subproblems

and then optimize them simultaneously. For instance, MOEA/D is a known algorithm in this category. Most EMO algorithms are dominance-based. Indeed, they use the Pareto dominance concept as an evaluation procedure. NSGA-II is the most studied algorithm in this category. Indicators-based algorithms use an indicator during the execution of the algorithm to measure the quality of the population [100]. An example of such a measure can be the hypervolume. EMO algorithms have, however, many drawbacks, which makes them very criticized in the literature [100]. They are very time-consuming, and they lack a convergence proof [100]. However, the main drawback remains that they require a vast number of parameters settings to be efficient. This procedure can take extensive time in such algorithms and require several experiments. Failing to set the rights parameters can induce bad results for these algorithms. The EA classification can be applied to multi-objective local search algorithms that have received particular attention recently in the field of multi-objective optimization. Pareto Local Search (PLS) proposed by Paquette et al. in [69] is, however, the only method that has succeeded in establishing itself in the field. Ben Mansour et al. [71], Cota et al. [90] and Cornu et al. [73] presented all aggregation based multi-objective local search. They use local search metaheuristics with structure, component, or decomposition methodology of Evolutionary Multi-Objective algorithms (EMO) to achieve the best performances. Dubois-Lacoste et al. [70] proposed an indicator-based Pareto Local Search (PLS) in their article. The above categories can be found even with multi-objective local search-based methods. Here again, dominance-based algorithms are the most used.

Non-dominated Sorting Genetic Algorithm (NSGA-II) : The Genetic Algorithm (GA) is a well-known algorithm inspired by Darwin's theory of evolution. Several multi-objective versions of this algorithm were proposed in the literature. The most known and successful approaches are NSGA and NSGA-II. NSGA and NSGA-II are an adaption of the GA to consider multiple objectives by ranking individuals according to their domination level. NSGA-II, proposed by Deb et al. [60] is an improved version of NSGA. The improvements are a fast non-dominated sorting procedure, an elitist-keeping technique, and a novel niching operator. A detailed explanation of the NSGA-II algorithm can be found

in [60]. We suggest the paper [58] for the description of the improvements procedures and a study of the influence of the different NSGA-II parameters on the performance of the algorithm.

A Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) : The MOEA/D was proposed by Zhang and Li in 2007 [72]. It is a method that decomposes the problem into several scalar optimization subproblems and optimizes them simultaneously. The MOEA/D starts by computing the euclidean distances between all two weight vectors. The designer of the algorithm needs then to choose T closest weight vectors $\lambda^{i_1},...,\lambda^{i_T}$ in the neighborhood of every weight vector λ^i of the subproblem $i = 1,2,...,N$. $B(i) = i_1,...,i_T$ includes the indexes of the closest vectors of λ^i. It also starts with an initial population and initialization of the vector $z = (z_1,...,z_m)$ where each z_i takes during the algorithm's execution, the best value found so far for the objective f_i. For each subproblem, the following steps are realized. Two random indexes are selected from $B(i)$ in the reproduction phase. Their corresponding solutions in the population are then chosen to generate a solution y using genetic operators. An optional improvement phase is used to generate y', the improvement of the solution y. The z is then updated with the new best solutions for the objectives. The following step is the improvement of the neighboring solutions using the Tchebycheff approach as an evaluation procedure. Finally, the population of solutions is updated. The algorithm is executed until meeting the stopping criterion. Zhang and Li find out [72] that MOEA/D outperforms or performs as well as MOGLS and NSGA-II on both multi-objective 0–1 knapsack problems and continuous optimization problems. Moreover, the normalization of objectives makes MOEA/D capable of performing well even with disparately scaled objectives.

Pareto Local Search(PLS) : The first reference multi-objective local search-based method was proposed in 2004 by Paquette et al. in [69] for the bi-objective Traveling Salesman Problem. The authors extend the single objective local search algorithms. The PLS algorithm starts with an archive. The new solution is compared with the archive if it is not dominated by s. It is a speed-up technique

used by the authors. The solution picked from the archive at each iteration should not be flagged as visited. All the neighborhood is explored, and the new solution is accepted if it is not dominated by any solution in the archive. This general local search approach to multi-objective problems is called Pareto local search (PLS). The PLS with the 3-opt local search obtained great results in terms of solution quality with the best-performing metaheuristics. The PLS has the main important advantages over most local search-based methods. It is quite simple since no ideal points nor aggregation of objectives are required [69]. In addition, no parameter settings are necessary for the algorithm [69]. The method's main drawback is that its computation times for large instances are extensive. In this case, the algorithm designer needs to think of strategies that reduce the number of solutions returned while maintaining good coverage of the solutions obtained. An example of such strategies could be the study of the acceptance of some efficient solutions while taking into account the number of objectives [69]. We have proposed an improved version of PLS with interesting alternative strategies in the fourth chapter of the contributions part of this manuscript, Chapter 6.

2.5.4 The interactive methods or progressive methods

The interactive or progressive methods are similar in the resolution methods and formulation to posterior methods. In both categories, the multi-objective formulation of the problem is maintained. However, in the interactive methods, the DM guides the optimization process by giving preference during optimization [100]. The DM guides the search process to the regions that interest him. The process starts by showing the decision-maker a Pareto optimal solution. Subsequently, the decision-maker provides his preference information to generate another Pareto optimal solution. The process continues with the objective of finding desirable solutions for the decision-maker until he wishes to stop [58]. Interactive methods are generally preferable and more efficient compared to priori methods [58]. The reason is the difficulty of adjusting preferences during optimization when preferences are specified at the beginning of optimization. Interactive methods are better than posteriori methods also. The number of

Pareto optimal solutions obtained with an interactive method is less than the number of Pareto optimal solutions obtained with a posteriori method. The algorithm avoids searching in no promising regions and consumes, therefore, less time to provide directly desirable Pareto optimal solutions [58]. The most known interactive method in the literature is the NIMBUS method for nondifferentiable multi-objective optimization problems proposed by Miettinen and Mäkelä in 1995. Details about the method can be found in [59].

2.6 Concluding remarks

Chapter 2 recalled the key optimization concepts and resolution approaches related to the thesis. Nevertheless, again, the focus is on approximate approaches being the only ones that can provide a solution within a reasonable time for large-size instances. The chapter presented the different classes of optimization problems. Constructive and improvement heuristics, as well as hyperheuristics, have then been discussed. The fundamental existing metaheuristics are divided into local search-based and population-based methods. Most of the time, they are the basis of all works published in the literature regarding the development of new metaheuristics. Therefore, the last two parts of the chapter are dedicated to them. Next, existing single objective metaheuristics have been presented, followed by multi-objective metaheuristics. The relation of these latter to the single objective ones has been shown. The chapter has surveyed multi-objective optimization classes and techniques included in these classes. The population-based methods are the most used and successful approaches to date.

New metaheuristics have been published in the literature over time. However, most of the papers in the literature are based on existing metaheuristics. They try to propose a change in their framework that improves the performance of the basic metaheuristics. Most of the time, this change is minor and regards the parameters tunning. In the following chapters, we provide formalized models of studied problems and detail novel-designed metaheuristics to solve them.

Part II

Contributions

Chapter 3

The Joint Maintenance Scheduling and Workforce Routing Problem: Model and Solution Approaches based on Variable Neighborhood Search

Outline of the current chapter

3.1 Introduction

Large-scale systems are essential to today's industry, such as aeronautics, railways, telecommunications, etc. The management of such systems is often complicated and requires an adapted maintenance strategy. Maintenance is indeed a vital primary service in industries, especially when failures are likely to impact personnel safety and cause important environmental damages. It can also have a major impact on profitability.

Maintenance aims to retain equipment in its operating condition or to restore it to a previous state enabling its required functions. To perform maintenance activities, it is crucial to effectively manage a crew of operators and sequence their visits over a planning horizon. Two main categories of maintenance can be distinguished [102]:

(1) Corrective Maintenance (CM) that is performed after the failure. It includes actions such as repair or replacement.

(2) Preventive Maintenance (PM), which occurs before the failure and intends to reduce the risks of unforeseen breakdowns. It can itself be divided into two categories:

1. Time-Based Maintenance (TBM) that includes the periodic replacement and the age replacement policy (ARP).

2. Condition-Based Maintenance (CBM) that is based on the inspection of the unit or system before the intervention.

In the first case of TBM, the preventive replacement is performed at regular time intervals. For the ARP, the unit is replaced depending on its age or its remaining useful life [102]. CBM is based on the inspection of the unit or system before the intervention. It focuses on predicting the health condition of the system using information collected from sensors.

The companies mainly outsource their maintenance operations to a service provider. This maintenance provider is required to ensure good quality maintenance services at the lowest overall cost. Figuring out the right time and manner to carry out the maintenance are therefore major concerns. The planning of maintenance operations on geographically dispersed machines subjected to random failures is, therefore, a complex problem faced by service providers. In the industry, machines are exposed to random failures that can lead to significant penalties. The opportune maintenance times are sometimes defined by the equipment manufacturer, but more often, the companies entrust their maintenance providers to define the maintenance schedule. The service provider tries to reduce the number of maintenance operations as much as possible, which can be accomplished by maximizing the periods between two successive visits to the same machine. At the same time, he seeks to avoid the huge costs resulting from carrying out corrective maintenance operations on broken machines. The challenge is therefore finding the optimal number of visits to perform maintenance operations. Mathematical maintenance policies are used to develop an effective preventive maintenance plan. The objective of such policies is to design a maintenance schedule including both preventive replacement and corrective replacement. The maintenance models dealing with these policies are probabilistic due to the uncertainty of the mechanism that causes failure. When machines are geographically distributed, it is necessary to sequence the visits and determine the best routing–maintenance policy. Cost, availability, reliability, and maintainability are the benchmarks for evaluating maintenance

decisions, similar to cost, quality, and time objectives in logistics. The maintenance company would be interested in finding the best compromise between services and maintenance costs on one hand and operational and transport costs on the other hand. The objective is to jointly consider the technical aspects of maintenance and the organizational aspects of operations management. To consider these two aspects, many works in the literature consider both maintenance and routing problems [82–84, 86]. To handle this optimization problem, it is necessary to simultaneously investigate the maintenance scheduling and vehicle routing problem (VRP). The vehicle routing problem with time windows (VRPTW) has been widely used in the literature as the main variant of VRP for planning and scheduling maintenance operations [77, 78, 82, 86]. The problem addressed in this chapter consists of determining the best routing maintenance policy when planning operators' schedules to perform maintenance operations on geographically distributed machines subjected to random failures.

Herein, the main contributions of this work are summarized as follows.

(1) A mathematical model is proposed, aiming to minimize failure, maintenance and transport costs in the case of time-based preventive maintenance. The first objective minimizes the total travel cost related to technicians' routing. The second objective minimizes the failure cost whereas the third objective minimizes the total preventive and corrective maintenance cost. A penalty cost is incurred when the maintenance activities are performed after the optimal time interval. The present chapter comes with the novelty of introducing a nonlinear and uncertain failure cost that uses information from equipment degradation. Moreover, we also include a continuous-time for the last restoration in the failure and maintenance objective of the model.

(2) A General Variable Neighborhood Search algorithm has been implemented to solve the mono-objective problem. Furthermore, novel greedy constructive heuristics are proposed to generate initial solutions. They are problem-specific and construct a solution for the problem whose objective function is the failure cost (respectively, the maintenance cost). They are followed by using the best insertion heuristic as an initial solution.

(3) Extensive experiments demonstrated that the GVNS algorithm outperformed

the commercial solver. We also test the improvement heuristics BILS and VND for comparison reasons. An analysis is conducted to clearly show the differences between the performance of the algorithms when using a random initial solution and the proposed constructed initial solutions. The results show the influence of a dedicated initial solution to achieve the desired solutions within a considerably reduced CPU time.

This study will be presented as follows: Section 3.2 sets a summary of the corresponding literature. Section 3.3 delineates the problem and sets out its mathematical formulation. A description of the proposed solution approaches and their implementation details are given in Section 3.4. Section 3.5 then presents experimental results. At last, Section 3.6 lays out concluding observations and perspectives for upcoming research.

3.2 Literature review

The majority of existing studies tackling routing and maintenance problems dealt separately with each of them. Two principal research streams can be distinguished in the literature:

First stream: The routing is used to schedule maintenance operations.

Second stream: Maintenance and routing are viewed as an integrated problem by considering both their specific features.

The first stream of research includes workforce scheduling problems that are particularly useful in reality and affect many organizations. Indeed, the workforce cost constitutes one of the highest costs in any organization. The major problems addressed in the first stream are Technician Routing and Scheduling Problem (TRSP) [81], Technician and Task Scheduling Problem (TTSP) [77], Service Technician Routing and Scheduling Problem (STRSP) [78], Workforce Scheduling and Geographically Distributed asset Maintenance Problems (GDMP) [84]. Cordeau et al. [77] proposed a mixed-integer linear program and a solution approach for the ROADEF 2007 challenge. This challenge tackled the specific features of the TTSP in a large telecommunications company. A constructive

heuristic was proposed to generate a feasible solution by defining teams and assigning tasks. An adaptive large neighborhood search metaheuristic is then used to solve the problem in the improvement phase. Each technician is specialized in different tasks with different skill levels. He is able to execute tasks requiring lower levels than his as well. Skills requirements and a time window characterize each maintenance task. An outsourcing budget, as well as task priorities, were considered. Kovacs et al. [78] introduced the service technician routing and scheduling problem. Their model minimized routing and outsourcing costs by considering skills and team-building constraints. They used an adaptive large neighborhood search algorithm to solve the problem. They also proposed to select destroy–repair operator pairs and adapted the adaptive mechanism consequently. Zamorano and Stolletz [79] proposed a mixed-integer program for the multi-period Technician Routing and Scheduling Problem. The model aimed to minimize total travel, waiting, and overtime costs. Technicians' proficiency in skill domains and the level of proficiency were considered to build teams of technicians and for task assignments. In addition, each operation was associated with a specific time window that could last several periods and had skills requirements. Decomposition schemes are implemented within a branch-and-price algorithm to solve this problem optimally. Mathlouthi et al. [80] presented a mixed-integer linear programming model for the multi-attribute Technician Routing and Scheduling Problem for an application in the electronic transactions equipment domain. Their model considered several attributes: task priorities, multiple time windows, technicians' skills, spare parts inventory, breaks, and overtime. The problem is solved by a commercial solver. However, small instances could only be carried out optimally, demonstrating the problem's complexity. Pillac et al. [81] dealt with the dynamic technicians routing and scheduling problem (DTRSP). The assignment of technicians with adequate skills, spare parts, and tools to tasks was realized to minimize the overall cost. To handle more requests, technicians can replenish tools and spares at the depot anytime. The authors proposed a re-optimization approach for the periodic problem. This approach relies on a parallel adaptive neighborhood search (RpALNS) algorithm, which generates a new route each time a request arrives. The suggested parallelization scheme distributes the computational

effort among the different processors. Their metaheuristic has achieved the same performances as the state-of-the-art results for the dynamic vehicle routing problem with time windows. Raknes et al. [91] proposed a mathematical model to schedule maintenance tasks performed by technicians transported using a fleet of dedicated vessels. Many vessels stayed offshore for several shifts in their model while executing maintenance tasks. The objective was to minimize the overall cost, including transportation costs, downtime costs, costs to stay offshore between shifts, and penalty costs for not completing a task during a shift during the planning period. Blakeley et al. [85] presented an automated route-scheduling and planning system developed for Schindler Elevator Corporation. It is a geographic-information-system-integrated application that utilizes operations research to optimize preventive maintenance operations. Algorithms assigned maintenance tasks to technicians and created daily routes using the periodic vehicle routing problem. Moreover, several features were considered. They include geographic proximity, technician workload, necessary skills, and customer relations. The objective was to minimize the travel cost, overtime cost, cost for violating time windows, cost of idle or free time within each route, and finally, a penalty for imbalanced workloads. Lesain et al. [92] proposed an information system that automates work management and field communications. This latter was strengthened with a dynamic scheduler based on heuristic search, simulated annealing, and constraint-based reasoning. In addition, several constraints were enumerated, making the system more complex. Among them, we find off-hours and breaks, predefined areas of the operation, overtime time permitted for the last task, task's time window, sequenced tasks, tasks parallelism, and best utilization of skills. Çakırgil et al. [129] studied an electricity company's multi-skill workforce scheduling and routing problem. Different skills requirements and priorities characterized the maintenance operations, and teams of technicians needed to be formed to perform them. A mixed-integer programming model was proposed to complete higher-priority tasks earlier and minimize total costs. In addition, the authors proposed a two-stage matheuristic based on the variable neighborhood search to solve the bi-objective problem.

The second research stream is concerned with the combination of maintenance and routing characteristics. Workforce scheduling is not the only problem

dealt with in this case since other maintenance problems are taken into account. Reliability analysis can be included to design solutions that assign technicians at the right time to perform maintenance operations. Workforce costs and maintenance costs are both dealt with in this case. Lopez-Santana et al. [82] proposed a mathematical model called the combined maintenance and routing (CMR) and a two-phase procedure to solve it. In the first phase, a maintenance model was solved to determine the optimal times to perform preventive maintenance operations, their frequency, and their time windows while minimizing the total expected maintenance cost. The output data of the maintenance model was then considered in a second phase as the input data of the routing model that scheduled maintenance operations for each technician. The maintenance model was again solved using the updated start times of preventive operations obtained by the routing model to connect the two problems. This procedure was repeated until meeting the stopping criterion. The objective function's nonlinear and convex maintenance cost was approximated using a piecewise linear function, and the problem was solved for small instances using a commercial solver. Jbili et al. [83] modeled an integrated vehicle routing and maintenance strategy. They considered vehicles used in transcontinental transportation that were subject to random failures on the road. These vehicle failures needed to be repaired, which might take random durations that induce delays. A policy consisting of replacing the critical component when arriving at selected customers was adopted. A mathematical model was proposed to simultaneously determine the optimal routing and sequence of PM actions. The objective was to minimize the total expected cost per unit time. It considered the reliability of the vehicle, maintenance (PM operations and minimal repairs), transportation costs, maintenance durations and penalties incurred by late arrivals. A genetic algorithm was then proposed to solve large instances. Chen et al. [84] studied the maintenance of gully pots or storm drains. They modeled a multi-period VRP, which considered the risk impact of gully pot failure and its daily failure behavior. The risk impact was estimated using meteorological information. A risk-driven analysis was adopted to evaluate maintenance actions. They focused on two factors: parked cars and gully pots status information. The latter has been proven to be the dominant factor that may negatively affect the scheduling

of maintenance actions. Rashidnejad et al. [86] presented a bi-objective multi-period model of preventive maintenance planning in geographically dispersed systems through prognostic information and remaining useful life (RUL) called the integrated vehicle routing problem with time windows and maintenance scheduling (IVRPTW-MS). The first objective minimized the total cost composed of the performing maintenance cost, the expected failure cost, and the travel cost, whereas the second objective minimized the unavailability of the assets. The authors used fixed costs and did not include corrective maintenance. They used a non-dominated sorting genetic algorithm II (NSGA-II) to solve this NP-hard problem.

There is a relatively small amount of research combining preventive and corrective maintenance strategies since most of the papers examined dealt only with preventive scheduling, apart from [82, 84]. Moreover, they did not consider the uncertainty of the breakdowns and did not deal with a multi-objective perspective. Papers that considered random breakdowns among those examined are [82,83]. The multi-objective formulation is considered in [86,129]. Analyzing the previous problems emphasizes the need to propose solutions to the real problem that includes the following realistic features: large scale, uncertainty, the combination of both corrective and preventive maintenance, and finally, routing of technicians in a multi-objective perspective. To the best of our knowledge, no previous study investigated these features together. These aspects make the NP-hard optimization problem even more challenging. Therefore, heuristic-based approaches are necessary for large instances since exact methods are practically limited.

This chapter proposes a mathematical model with different objective functions for the joint maintenance and routing problem. The first objective minimizes the total travel cost and the penalty cost. The second objective minimizes both the failure cost and the penalty cost. It includes the failure cost that is nonlinear and uncertain. It is introduced with the routing model to use information from equipment degradation. The third objective minimizes the total maintenance cost and the penalty cost for late arrivals. The latter is an extension of the cost proposed by Lopez-Santana et al. [82] in their single objective model. The maintenance hypothesis of renewal theory is considered with the failure and

maintenance costs through a continuous time for the last restoration. Moreover, the failure cost has not previously been used as an objective of a routing problem. Several previous research gaps of the problem in the literature are therefore filled. We first designed novel greedy constructive heuristics to find an initial solution to solve the problem. They are based on the failure and maintenance cost and considerably reduce the CPU time when used compared to random initial solutions. They are also far faster in comparison with constructive heuristics in the literature. The proposed General Variable Neighborhood Search results are shown for each mono-objective problem. Variable Neighborhood Descent and Best Improvement Local Search results are also provided for comparison reasons.

3.3 Problem definition and formulation

We consider a set $\mathcal{M}$ of machines that are located in dispersed customers' sites. They are subjected to random breakdowns due to the failure of a critical component. For each machine and at regular time intervals, preventive maintenance (PM) interventions are scheduled. A PM operation has a cost Cpm_i and lasts a duration Tpm_i. Each preventive operation must be performed within its corresponding time window. However, if it is not possible due to the high workload of technicians, late arrivals are then allowed. Hence, technicians can arrive at an operation i following the end of its time window b_i. A penalty cost p_{ik} is incurred in this case. On the other hand, a team of technicians has to wait until the earliest time a_i to start an operation i. Teams of technicians $\mathcal{K}$ are available to perform both preventive and corrective maintenance. They perform corrective maintenance (CM) operations when machines unexpectedly break down. In this case, the machine stays in a failure state until the arrival of the team of technicians for a certain time W_i. The unit waiting cost for each operation is Cw_i. A CM operation has a cost Ccm_i and lasts a duration Tcm_i. A minimized number of vehicles among the available ones must be determined and used for all operations to reduce transport costs. The Node 0 is considered the departure point of maintenance teams, and the Node $n+1$ is considered the final destination. We denote by $\mathcal{O}$ the set of all nodes of the PM operations related to all machines,

and by $\mathcal{V}$, the set of all locations (operations nodes plus the depots). Each team of technicians has to perform the maintenance operations at the customers' sites. Thus, the model could be defined as a directed complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{A} = \{(i, j), i \in \mathcal{V}^o, j \in \mathcal{V}^d, i \neq j\}$ with $\mathcal{V}^o$ and $\mathcal{V}^d$ as the sets of origin and destination vertices. The following assumptions are considered:

- All maintenance teams have the same skills and qualifications to do the maintenance operations.
- It is considered that the random variable of the time to failure for each machine follows a Weibull distribution whose shape and scale parameters are, respectively, β_m and σ_m, $m \in \mathcal{M}$. Any other distribution resulting from the historical data of machines failures can be applied.
- We suppose $\beta_m > 1, m \in \mathcal{M}$ to deal with the third part of the bathtub curve. This part features the wear-out life of the machine when the failure rate is an increasing function of age or usage. Indeed, the wear-out is the phenomenon accelerating the risk of failure over time.
- All costs related to the maintenance and penalties are known and constant.
- The duration of PM and CM tasks are known and constant.
- After each PM or CM operation, a machine is considered in a state similar to a new one.
- The mean time of the CM operation is superior to the mean time of the PM operation.
- The cost of a CM operation is superior to the cost of a PM operation.
- Each PM operation must be performed in its associated time window. However, if it is not possible, we allow a team of technicians to arrive following the end of its time window. In this case, a penalty cost is added to the total cost.
- The failure of a machine is generally due to the failure of its critical components.

- Failures of individual machines are statistically independent.
- A machine can have several maintenance operations over the planning horizon.
- In case of failure, before the beginning of the next PM operation, the customer must wait for the team of technicians to perform a CM operation instead. The team is not rescheduled based on this new situation.
- The travel times are deterministic and satisfy the triangle inequality.

3.3.1 Notation of the joint maintenance scheduling and workforce routing problem

The notations below are employed throughout the paper.

Sets

- $\mathcal{M} = \{1, \ldots, l\}$: set of machines.
- $\mathcal{O} = \{1, \ldots, n\}$: set of PM operations ($n = \sum_{m \in \mathcal{M}} n_m$).
- $\mathcal{V} = \{0, \ldots, n+1\}$: set of all vertices including the PM operations, departure depot 0 and destination depot $n+1$.
- $\mathcal{V}^o = \{0, \ldots, n\}$: set of origin nodes.
- $\mathcal{V}^d = \{1, \ldots, n+1\}$: set of destination nodes.
- $\mathcal{K} = \{1, \ldots, K\}$: set of technicians' teams or vehicles.
- $\mathcal{A} = \{(i, j), i \in \mathcal{V}^o, j \in \mathcal{V}^d, i \neq j\}$: set of arcs linking nodes.

Maintenance parameters

- Cpm_m: total preventive maintenance operation (PM) cost of machine m.
- Ccm_m: total corrective maintenance operation (CM) cost of machine m.

- Cw_m: unit waiting cost. This cost can be interpreted as the production loss cost per unit time incurred by the customer for this machine m.
- Tpm_m: service time of PM operation of machine m.
- Tcm_m: service time of CM operation of machine m.
- $M_m(\delta_m)$: mean time to failure of the machine m given that the failure occurred before the optimal PM period for machine m, δ_m.
- W_m: waiting time before starting a CM operation on machine m . It is the time waited before the arrival of the technicians to perform a CM operation when the machine m suddenly breaks down.
- $CM_m(\delta_m)$: the total cost per unit time if the PM operation is performed when the machine m is of age δ_m.
- T_m: random variable of the time to failure of machine m.
- β_m: shape parameter of the Weibull distribution of machine m.
- σ_m: scale parameter of the Weibull distribution of machine m.
- $F_m(t)$: cumulative distribution function of t. It represents also the probability of failure of machine m in the interval $[0, t]$.
- $f_m(t)$: density function of the Weibull distribution of t of machine m.
- tol_m: the percentage of time of delaying or advancing a PM operation.
- H: length of the planning horizon.

Routing Parameters

- $t_{i,j}$: travel time associated to the arc $(i, j) \in \mathcal{A}$.
- $c_{i,j}$: routing cost associated to the arc $(i, j) \in \mathcal{A}$.
- a_i: earliest time to start a PM operation i.

- b_i: latest time to start a PM operation i.
- c: penalty cost per unit time for arriving after the deadline b_i associated to PM operation.
- K: number of teams of technicians.
- B: large number.

Variables

- $x_{i,j,k}$: binary decision variable that takes the value one if the technicians' teams k travels through arc $(i,j) \in \mathcal{A}$ and 0 otherwise.
- $\theta_{i,k}$: arrival time of the team of technicians k to the maintenance operation i.
- $p_{i,k}$: penalty variable that measures the total time in excess of the latest permitted time to start the service b_i by the team of technicians k. $p_{i,k} = max(0, \theta_{i,k} - b_i)$.
- $\rho_{i,k}$: time of the last restoration (renewal) previous to the i-th PM operation performed by the team of technicians k. It is the previous start time of the operation on the same machine.

The variables of the maintenance model that are constant input parameters for the routing one are defined as follows:

- δ_m: optimal PM period for machine m.
- n_m: frequency of PM operations of machine m. It is the number of PM operations of the machine m on the planning horizon.
- ϕ_i: execution time of operation i.
- n: number of PM operations to perform in the planning horizon.

3.3.2 The maintenance model

The decision model to determine the optimal interval between PM operations is referred to as the maintenance model. It is used when the aim of performing maintenance activities is to minimize the total related maintenance costs of preventive and corrective maintenance operations. The optimal period δ_m to perform a PM operation is determined for each machine m with the frequency n_m of PM operations in the planning horizon.

In the following, we define the main terms employed in the maintenance decision model. The probability of failure in the interval $[0, \delta_m]$ of machine m, where P_m is the probability's notation, is:

$$F_m(\delta_m) = P_m(T_m \leq \delta_m) = \int_0^{\delta_m} f_m(t)\,\mathrm{d}t \tag{3.1}$$

The model generally applied by researchers in the literature to settle optimal times to carry out PM operations in the case of periodic preventive maintenance is defined in [89]. Given a machine m, we seek the optimal time δ_m^* to execute PM operations that minimizes the total maintenance cost $CM_m(\delta_m)$:

$$CM_m(\delta_m) = \frac{E[CM_m(\delta_m)]}{E[T_m(\delta_m)]} = \frac{Cpm_m(1-F_m(\delta_m))+Ccm_mF_m(\delta_m)}{\delta_m(1-F_m(\delta_m))+M_m(\delta_m)F_m(\delta_m)} \tag{3.2}$$

The terms $E[CM_m(\delta_m)]$ and $E[T_m(\delta_m)]$ refer, respectively, to the total expected cost of a cycle and the expected cycle length for a machine m.

Lopez-Santana et al. [82] propose an extended maintenance cost which includes the waiting cost expression in addition to the PM and CM durations. These service times cannot be negligible when the systems under study are large-scale.

$$CM_m(\delta_m) = \frac{Cpm_m(1-F_m(\delta_m))+(Ccm_m+W_m*Cw_m)F_m(\delta_m)}{(\delta_m+Tpm_m)(1-F_m(\delta_m))+(M_m(\delta_m)+W_m+Tcm_m)F_m(\delta_m)} \tag{3.3}$$

$M_m(\delta_m)$ represents the expected time to failure of machine m assuming the

failure happens before δ_m.

$$M_m(\delta_m) = \int_0^{\delta_m} \frac{t f_m(t)}{F_m(\delta_m)} \, \mathrm{d}t \tag{3.4}$$

Figure 3.1 shows the maintenance cycles. One cycle can be either preventive or corrective. This figure clearly explains the equation 3.3.

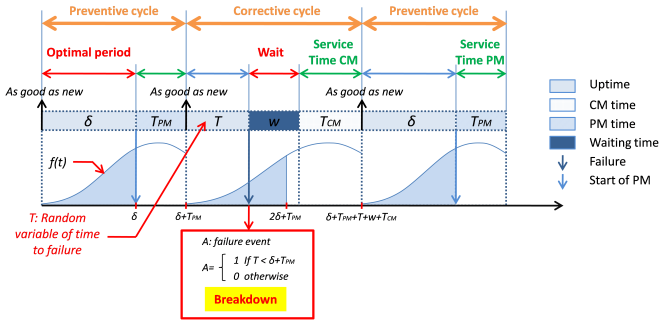

Figure 3.1: Example of maintenance cycles with preventive and corrective maintenance operations [82]

According to Lopez-Santana [82], the waiting time W_i of an operation i undertaken by the teams of technicians k is the difference between the technicians arrival time to the PM task $\theta_{i,k}$ and the mean time to failure $M_m(\theta_{i,k})$:

$$W_i = \theta_{i,k} - M_m(\theta_{i,k}), \quad i \in \{1, \ldots, n_m\}, k \in \mathcal{K} \tag{3.5}$$

The values of $\theta_{i,k}$ are needed to compute the waiting times. They are also the outputs of the routing model. Consequently, the number of W_i obtained is equal to the number of PM operations to be performed for the machine m in the planning horizon. Since the maintenance model takes only one value, Lopez-Santana et al. [82] consider the average value of all W_i, where i refers to the PM operations related to the machine m to update the waiting time W_m and iterate the procedure. However, the values of $\theta_{i,k}$ are unknown before solving the routing model. Therefore, we suppose a PM operation will be scheduled at δ_m. This approximation has also been used by [128].

The waiting time is therefore defined as follows:

$$W_m = \delta_m - M_m(\delta_m) \tag{3.6}$$

The maintenance cost per time unit for a machine m at the time δ_m is finally equal to:

$$CM_m(\delta_m) = \frac{Cpm_m(1-F_m(\delta_m))+(Ccm_m+(\delta_m-M_m(\delta_m))Cw_m)F_m(\delta_m)}{(\delta_m+Tpm_m)(1-F_m(\delta_m))+(\delta_m+Tcm_m)F_m(\delta_m)} \tag{3.7}$$

This nonlinear equation without constraints is solved to obtain the optimal period for each machine m, $\delta_m^* = \text{argmin } CM_m(\delta_m)$.

The frequency of a PM operation on the planning horizon H can be obtained as follows:

$$n_m = \frac{H}{E[T_m(\delta_m^*)]}, \quad m \in \mathcal{M} \tag{3.8}$$

The frequency of a PM task is the number of times it is realized in the horizon. Considering a frequency n_m of a machine m, the PM operations need to be performed at times $\{\delta_m^*, \delta_m^* + E[T_m(\delta_m^*)], \delta_m^* + 2 * E[T_m(\delta_m^*)],\ldots, \delta_m^* + (n_m - 1) * E[T_m(\delta_m^*)]\}$. The execution date ϕ_{mo} of an operation o corresponding to machine m is equal to ϕ_i which represents the execution date of an operation i and is obtained as follows:

$$\phi_i = \phi_{mo} = \delta_m^* + (o-1) \times E[T_m(\delta_m^*)], \quad o \in \{1,\ldots,n_m\}, m \in \mathcal{M}, i \in \mathcal{O} \tag{3.9}$$

The time window $[a_i, b_i]$ of PM tasks associated with a machine are obtained using $[a^m, b^m]$ the time window of the PM period of machine m on the first cycle. $[a^m, b^m]$ is determined using the percentage of time of postponing or preempting a PM task that we can tolerate. The cost stays relatively low and close to the minimal value $CM_m(\delta_m^*)$ in this interval. $[a_i, b_i]$ can be expressed as follows:

$$a_i = a_o^m = a^m + (o-1) \times E[T_m(\delta_m^*)], \quad o \in \{1,\ldots,n_m\}, m \in \mathcal{M}, i \in \mathcal{O} \tag{3.10}$$

$$b_i = b_o^m = b^m + (o-1) \times E[T_m(\delta_m^*)], \quad o \in \{1,\ldots,n_m\}, m \in \mathcal{M}, i \in \mathcal{O} \tag{3.11}$$

where a_o^m and b_o^m correspond to the lower and upper bound of the time window of machine m and its associated operation o. Each machine m can have

n_m operations in the horizon, $o \in \{1,\dots,n_m\}$. All operations for all machines are indexed by i. For each machine m, we can associate an operation i with $i \in \{1,\dots,n\}$ and $(n = \sum_{m\in\mathcal{M}} n_m)$. The interval $[a^m, b^m]$ is determined as follows:

$$a^m = \delta^*_m - tol_m \times \delta^*_m, \quad m \in \mathcal{M} \tag{3.12}$$

$$b^m = \delta^*_m + tol_m \times \delta^*_m, \quad m \in \mathcal{M} \tag{3.13}$$

3.3.3 The joint maintenance scheduling and workforce routing model

As mentioned previously, a machine has to undergo a number n_m of operations within the time horizon. They are indexed by $i \in \{1,\dots,n_m\}$. In this case, the parameters related to operations i related to the same machine m are equal to the parameters of this machine m. F_i, σ_i and β_i are, respectively, equal to F_m, σ_m and β_m. This way, the values of the operations' parameters in the routing model are obtained from the machines' parameters. The mathematical model of the integrated maintenance scheduling and routing problem can be formulated as follows.

$$\min(f_l(x_{i,j,k}, \theta_{i,k}, \rho_{i,k}, p_{i,k}), l = 1,2,3) \tag{3.14}$$

s.t.

$$\sum_{j=1}^{n+1}\sum_{k=1}^{K} x_{i,j,k} = 1, \quad \forall i \in \mathcal{O}, i \neq j \tag{3.15}$$

$$\sum_{i=0}^{n}\sum_{k=1}^{K} x_{i,j,k} = 1, \quad \forall j \in \mathcal{O}, i \neq j \tag{3.16}$$

$$\sum_{i=0}^{n} x_{i,j,k} = \sum_{i=1}^{n+1} x_{j,i,k}, \quad \forall j \in \mathcal{O},\ \forall k \in \mathcal{K} \tag{3.17}$$

$$\theta_{i,k} + Tpm_i(1 - F_i(\delta^*_i)) + Tcm_i F_i(\delta^*_i) + t_{i,j} \leq \theta_{j,k} + B(1 - x_{i,j,k}), \quad \forall i \in \mathcal{V}^o,\ \forall j \in \mathcal{V}^d,\ \forall k \in \mathcal{K},\ i \neq j \tag{3.18}$$

$$a_i \leq \theta_{i,k} \leq b_i + p_{i,k}, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{K} \tag{3.19}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{K} x_{0,j,k} = \sum_{i=1}^{n}\sum_{k=1}^{K} x_{i,n+1,k} \tag{3.20}$$

$$\sum_{j=1}^{n} x_{0,j,k} \leq 1, \quad \forall k \in \mathcal{K} \tag{3.21}$$

$$\theta_{i,k}, p_{i,k}, \rho_{i,k} \geq 0, \quad x_{i,j,k} \in \{0,1\} \tag{3.22}$$

The first objective function f_1 regards the transport of technicians. The second objective function f_l deals with maintenance. It equals either f_2 or f_3:

$$f_1 = \sum_{i=0}^{n}\sum_{j=1}^{n+1}\sum_{k=1}^{K} c_{i,j} x_{i,j,k} + \sum_{i=0}^{n+1}\sum_{k=1}^{K} c \times p_{i,k} \tag{3.23}$$

$$f_2 = \sum_{i=1}^{n}\sum_{k=1}^{K} (Ccm_i + W_i * Cw_i) F_i(\theta_{i,k} - \rho_{i,k}) + \sum_{i=0}^{n+1}\sum_{k=1}^{K} c \times p_{i,k} \tag{3.24}$$

$$f_3 = \sum_{i=1}^{n}\sum_{k=1}^{K} CM_i(\theta_{i,k} - \rho_{i,k}) + \sum_{i=0}^{n+1}\sum_{k=1}^{K} c \times p_{i,k} \tag{3.25}$$

The objective function f_1 (3.23) minimizes the total travel cost related to technicians routing as well as the penalty cost of not respecting the operations time windows upper bounds. The objective function f_2 (3.24) minimizes the failure cost, the waiting cost, and the penalty cost. Minimizing the machines probability of failure is equivalent to maximizing machines reliability. The objective function f_3 (3.25) minimizes the total expected maintenance cost and the penalty cost incurred for arriving after the deadline b_i associated to PM operations. The penalty term is intended to ensure that the time windows are respected. Therefore, it is considerably small compared to the first term of the objectives whether it is the routing, failure, or maintenance cost and does not increase the correlation between the objectives. The constraints (3.15) and (3.16) indicate that each PM operation has to be executed exactly once by one team of technicians. Constraints (3.17) ensure that the entry of a team of technicians

to node i is mandatorily followed by their leaving. Constraints (3.18) make sub-tours impossible. The purpose of constraints (3.19) is ensuring that each PM operation is carried out within its time window. Note that we assume $a_0 = 0$ and b_{n+1} represents the maximum time of arrival to the depot. It is permitted to arrive after the deadline b_i. A penalty p_i is then calculated. Constraints (3.20) determine the number of vehicles needed and that minimizes the total costs. Constraints (3.21) ensure that there are different teams of technicians in the different tours. They also ensure that every tour starts at the depot. Finally, the constraints (3.22) impose domain conditions on the variables.

3.3.4 The novelty of the proposed model

Several models have been proposed for the maintenance scheduling and workforce routing problem. However, most of them are included in the first stream we identified in Section 3.2. We have incorporated technical reliability requirements within the model that is included in the second stream we defined in Section 3.2. The present chapter comes with the novelty of exploring the combination of vehicle routing and maintenance problems in a unique model. The definition of the failure cost (3.24) itself and its possible use is the first novelty of the proposed model. It is a risk-based cost, particularly useful for industries where failures have serious damages that influence the safety of both agents and users. The maintenance cost (**??**) extends the cost proposed by Lopez-Santana et al. [82] that minimizes the total preventive and corrective maintenance cost. In both costs related to maintenance, the time of the last restoration previous to the i-th PM operation performed by the team of technicians k, $\rho_{i,k}$ is used to verify the hypothesis of the renewal theory. Including this continuous time in the objective functions is another novel model's aspect. Penalties for late arrivals are also associated with each objective considered.

The travel Cost: The first objective of the model aims to minimize the total travel cost and the penalty cost of violating the time window upper bounds. When considering this cost as an objective, the model is still included in the second stream of research thanks to the workforce routing constraints that incorporate maintenance requirements (maintenance time windows, task

durations). Maintenance time windows and task durations are obtained by the maintenance model as in the work of Lopez-Santana et al. [82].

The Failure Cost: The second objective function f_2 minimizes the failure and penalty costs. The aim is to maximize machines' reliability. The uncertainty aspect is integrated into the failure cost function by incorporating the probability of failure and the reliability of each machine. In the second objective function, we have used the probability of failure, $F_i(\theta_{i,k} - \rho_{i,k})$, for each operation $i \in \mathcal{O}$ to utilize information from equipment degradation and hence consider failure risks in technicians' assignment to tasks. It includes direct costs (failure cost) and indirect costs (production losses), illustrated by the waiting time. During that time, the failed machines were out of order and were not used by the organization for production activities. Losses are, therefore, supported by the company. When we multiply it by the waiting cost per unit time, we obtain a cost that can be interpreted as the production loss incurred by the organization due to this machine failure. The failure cost is beneficial in industries where breakdowns are hazardous and influence safety. This term is nonlinear and includes a random variable. It is equal, in the case of the Weibull distribution, to:

$$F_i(\theta_{i,k} - \rho_{i,k}) = 1 - e^{-((\theta_{i,k}-\rho_{i,k})/\sigma_i)^{\beta_i}}, \quad i \in \mathcal{O}, \forall k \in \mathcal{K} \tag{3.26}$$

The Maintenance Cost: The objective function f_3 minimizes the total expected maintenance cost with the penalty cost. The maintenance cost balances both preventive and corrective maintenance to find the least cost considering the two strategies.

3.3.5 Details about the model

The Task Duration: The duration of the maintenance operation i is probabilistic. It is simplified in the model as proposed by [82] to be deterministic. Its real expression proposed by [82] is:

$$d_i = Tpm_i(1 - F_i(\theta_{i,k} - \rho_{i,k})) + Tcm_i F_i(\theta_{i,k} - \rho_{i,k}), \quad i \in \mathcal{O}, k \in \mathcal{K} \tag{3.27}$$

The constraints (3.18) are therefore probabilistic since they include $F_i(\theta_{i,k} - \rho_{i,k})$ the probability of failure in the last cycle. This set of constraints makes the problem a stochastic non-linear combinatorial optimization problem. The stochastic variables make the problem more complex. The service time of the PM task is simplified and obtained using δ_i^* the optimal PM time for operation i [82]:

$$d_i = Tpm_i(1 - F_i(\delta_i^*)) + Tcm_i F_i(\delta_i^*), \quad i \in \mathcal{O} \tag{3.28}$$

Note that $\delta_i^* = \delta_m^*$ if i are operations corresponding to the same machine m. The operations for the same machine have, therefore, the same duration. The simplified duration in equation (3.28) is considered in the model instead of the expression of duration in (3.27). The objective is to make the problem a deterministic non-linear combinatorial optimization problem.

The Weibull distribution: The density function of the Weibull distribution is defined hereafter. Here, we assume that the position parameter of this distribution γ_m=0. Failures can happen in the interval $[0, \gamma_m]$ if $\gamma_m \geq 0$. We also consider $\beta_m > 1, m \in$ to deal with old machines in their wear-out period. The latter is characterized by an increasing failure rate of age or usage.

$$f_m(t) = \frac{\beta_m}{\sigma_m}\left(\frac{t-\gamma_m}{\sigma_m}\right)^{\beta_m - 1} e^{-\left(\frac{t-\gamma_m}{\sigma_m}\right)^{\beta_m}} \tag{3.29}$$

Fixing the number of tours: In the above model, the number of tours can be fixed to a predefined value $r \leq K$ using constraints (3.30, 3.31) instead of (3.20). Constraints (3.30, 3.31) fix the number of tours by specifying that each team should go from and arrive at the depot exactly r times.

$$\sum_{j=1}^{n+1} \sum_{k=1}^{K} x_{0,j,k} = r, \quad r \leq K \tag{3.30}$$

$$\sum_{i=0}^{n} \sum_{k=1}^{K} x_{i,n+1,k} = r, \quad r \leq K \tag{3.31}$$

3.4 Proposed solution approach

The proposed mathematical model is a mixed-integer linear program (MILP) when considering the routing objective. Furthermore, it is a Mixed Integer Non-Linear program (MINLP) when considering the failure and maintenance objectives. Commercial solvers cannot solve the problem in a reasonable time when applied to large-scale instances. VRPTW is NP-hard [86], and the classical maintenance scheduling problem using operations research techniques is NP-hard as well [86]. Naturally, the combined problem is an NP-hard problem.

The following section presents novel constructive heuristics, improvement heuristics, and General Variable Neighborhood Search (GVNS) algorithm to deal with the mono-objective combined maintenance and routing problem. At first, the method adopted is explained, then the solution representation is shown. Next, the functions used by the algorithms and neighborhood structures adopted are presented. Finally, the algorithms are described.

3.4.1 Solution representation and constraints handling

A solution is represented as a set of K tours, where each tour is a permutation of PM operations. All the constraints considered are hard, apart from verifying the upper bounds of the time windows. We allow the arrival of a team to an operation after the latest permitted time defined by its time window. We, therefore, accept only feasible solutions. Figure 3.2 shows the solution representation that was used in our implementation and the results for one instance when minimizing only the first objective (the routing cost). The solution that minimizes this cost comprises two routes (two vehicles are therefore needed). Each route is composed of three tasks in the specified order. Section 3.3.3 explicitly indicates how the three objective costs are evaluated and how the start time is calculated $\theta_{ik}, i \in \mathcal{O}, k \in \mathcal{K}$. For instance, five machines are being considered. Each machine has one operation, apart from machine three, which has two operations: three and four. The renewal time ρ_{ik} for all operations apart from four is, therefore, zero. The renewal time ρ_{42} for task four related to machine three is in route two. It is equal to the start time of task three associated with the same machine, and

that precedes task four. The data of this specified instance is detailed in [98] and Section 3.5.1 of this chapter.

Routing cost	Failure cost	Maintenance cost	Penalty
110.88	935.206	43.663	0

Route 1	Task	1	5	6
	Start time	36.2588	41.7832	45.5967
	Renewal time	0	0	0

Route 2	Task	3	2	4
	Start time	29.4947	44.109	62.657
	Renewal time	0	0	29.4947

Figure 3.2: Solution representation of the result minimizing the routing cost for the instance Re/6/2/80/80/0.07/2 (time in hours).

3.4.2 Greedy constructive heuristics for the initial solutions

We propose a fast heuristic to construct a good quality solution in the case where the objective function of the model is (3.24). This objective aims to minimize the probability that a machine fails before the start time of the preventive maintenance operation. Therefore, we call the heuristic the Greedy Constructive Heuristic Failure (GCHF). The heuristic is based on the maintenance operations being carried out as soon as possible in case of minimization of equipment failures. Therefore, the best start time an operation tends towards is the lower bound of its time window. We have also proposed a second fast heuristic to construct a good quality solution in the case where the objective function of the model is (3.25), which consists of the minimization of the total maintenance cost that we call the Greedy Constructive Heuristic Maintenance (GCHM). It is based on the principle that the best execution time ϕ_i of maintenance operations is the optimal value obtained with the maintenance model. The heuristic seeks, therefore, to coincide the start time $\theta_{ik}, i \in \mathcal{O}, k \in \mathcal{K}$ with the optimal execution time $\phi_i, i \in \mathcal{O}$. Since the routing model has the same objective, with some flexibility allowed by the time windows and more constraints due to many maintenance operations, it can be a good heuristic for the problem. When

using these heuristics to build an initial solution, the computational time is considerably reduced for the VND, BILS, and GVNS. These greedy heuristics can also lead to optimal solutions in small instances and when the penalty's cost equals 0. GCHM produces better results than GCHF for the third objective since it is dedicated to it. The heuristics are presented in the algorithm below.

Algorithm 1 : Greedy Constructive Heuristics Failure (GCHF) and Maintenance (GCHM).

Data : r: the initial number of empty routes ;
a_i: the lower bounds of the time windows ;
ϕ_i: the optimal execution times;
Result : s: the final solution
1 Sort a_i for GCHF (resp. ϕ_i for GCHM) from least to highest;
2 $OS \leftarrow$sortOperationsFromLeastToHighest(a_i) for GCHF (resp. sortOperationsFromLeastToHighest(ϕ_i)for GCHM) ;
3 **if** $r = 1$ **then**
4 Insert first and last depot and this is the final solution ;
5 **else**
6 **repeat**
7 Select r operations with the least a_i for GCHF (resp. ϕ_i for GCHM) from OS not previously assigned;
8 Insert them in the following empty positions of the r routes from first route to last route (1 to r);
9 **until** $OS = \varnothing$;
10 Insert first and last depot in all routes to obtain the final solution;

3.4.3 Neighborhood structures

The VNS and the VND procedures depend on the set of neighborhood structures chosen and their execution order. The neighborhood structures should be applied from the best to the least performing [88]. The sequence of moves operators that we considered is the swap, the insert, the 2-opt*, and the 2-opt procedure. The same order is usually used for VRP problems in the literature [105, 106]. We used a Steepest Descent Heuristic to validate this order, also known as Best Improvement Local Search. The best order to apply in local search may depend on the problem addressed but also on the characteristics of the

instances. We, therefore, have tested randomization and semi-randomization of neighborhood structures in both shaking and VND phases. We noticed that the semi-randomization (randomization of the three first neighborhood structures) with the first improvement strategy and the order with the best improvement strategy have equivalent performance in VND. However, the order is critical in the GVNS scheme, including the shaking phase. Local search procedures have two well-known acceptance strategies: First-Accept (FA) and Best-Accept (BA). In the FA, the first solution that satisfies the acceptance criterion is chosen. In contrast, the BA strategy checks all the neighbors that meet the acceptance criterion and chooses the best of them. We choose the best improvement strategy and apply an order for our GVNS algorithm. Moves can be performed on a single route (intra-route moves) or multiple routes (inter-route moves). For each operator used, all possible positions are examined.

The Swap move consists of exchanging two operations from the same route or different routes.

The Insert move generates a neighbor of a solution by removing an operation from its position and inserting it in a different one, within the same route, or in another route.

The 2-opt* operator removes arcs $(i, i+1)$ and $(j, j+1)$ from two different routes and reconnects arcs $(i, j+1)$ and $(j, i+1)$. This operator is inter-route.

The 2-opt operator removes arcs $(i, i+1)$ and $(j, j+1)$ from the same route and reconnects arcs (i, j) and $(i+1, j+1)$. This operator is equivalent to reversing the elements between i and $j+1$. This operator is intra-route.

The insert and 2-opt* operators change the number of operations in the routes and the order of operations in the routes.

3.4.4 Best improvement local search

The best improvement local search is used to classify the neighborhoods in the VND and as a heuristic approach. The results of this algorithm are presented for

these two main reasons. The BILS clearly shows the influence of a well-chosen initial solution on the rapidity of obtaining the results.

Algorithm 2 : Local Search Best Improvement (LSBI).

Data : k_{max}: number of neighborhood structures ;
s_0: initial solution ;
Result : s : solution ;

repeat
  Choose $s^{'}$ in a neighborhood N of S, such that $f(s^{'}) \leftarrow \underset{S \in V(s)}{\text{argmin}} f(s)$;
  if $f(s^{'}) < f(s)$ **then**
    $s \leftarrow s^{'}$;
until *s is a local optimum*;

3.4.5 Variable neighborhood descent and shaking

The Variable Neighborhood Descent is a search procedure that uses multiple neighborhoods, unlike a local search that explores only one neighborhood at a time.

The shaking procedure consists of selecting a random solution $s^{'}$ from the current k^{th} neighborhood of the current solution s ($s^{'} \in N_k(S)$) to diversify the search process. It is applied nS times.

3.4.6 General variable neighborhood search (GVNS)

Variable Neighborhood Search (VNS) is a simple and powerful metaheuristic proposed by Mladenovic and Hansen [87] in 1997 to solve single-objective problems. VNS systematically changes neighborhood structures during the search process. Its goal is to find an optimal or near-optimal solution. It starts from an incumbent solution s and applies two successive and essential mechanisms in each iteration. The first mechanism is the perturbation (or shaking) procedure necessary for the VNS schema. It is used to escape from local minima and therefore ensures diversification. Then, a local search follows it to improve the current solution. The local search procedure (intensification) aims to explore

Algorithm 3 : Variable Neighborhood Descent (VND).

Data : l_{max}: number of neighborhood structures ;
s_0: initial solution ;
Result : s : solution ;

$s \leftarrow s_0$;
repeat
  $l \leftarrow 1$;
  while $l \leq l_{max}$ **do**
    Choose $s^{'}$ in a neighborhood N_l, such that $f(s^{'}) \leftarrow \underset{s \in N_l(s)}{\text{argmin}} f(s)$;
    if $f(s^{'}) < f(s)$ **then**
      $s \leftarrow s^{'}$;
      $l \leftarrow 1$;
    else
      $l \leftarrow l + 1$;
    end
  end
until *no improvement is obtained*;

each incumbent solution's neighborhoods efficiently. Details about the VNS method can be found in [88]. VNS has been successfully applied to various optimization problems, especially in the mono-objective case. We use a general variable neighborhood search (GVNS) to solve the problem. It has the same schema as the basic VNS; however, it uses a variable neighborhood descent (VND) as a local search to explore several neighborhood structures at once. As a result, it has a better chance of leading to an optimal or good near-optimal solution, but it generally consumes more time than the other VNS variants. The pseudo-code of the proposed GVNS algorithm 4 is presented below. In order to reduce the computational time, dedicated heuristics were used to generate initial solutions. In addition, several experiments were conducted to select suitable neighborhood structures and to decide on their order. They are detailed in the Section 3.4.3, Section 3.4.4 and with more details in the following Section 3.5.

Algorithm 4 : General Variable Neighborhood Search (GVNS).

Data : k_{max}: number of neighborhood structures in GVNS ;
l_{max}: number of neighborhood structures in VND ;
nS : diversification parameter representing the number of times shaking is applied to the solution;
Result : s: the final solution

```
1  s_0 ← constructionHeuristic() ;
2  repeat
3      k ← 1 ;
4      while k ≤ k_max do
5          s' ← S;
6          for p = 1   to   nS do
7              s' ← Shaking(s', k);
8          s'' ← VND(s', l_max);
9          if f(s'') < f(s) then
10             s ← s'' ;
11             k ← 1;
12         else
13             k ← k + 1;
14 until stopping criterion (maximum number of iterations iter_max;
```

3.5 Computational experiments

This section presents computational experiments to measure the proposed algorithms' performance. First, we present the results of the mono-objective problem with three different objectives (the routing, the failure, and the maintenance cost) using the GVNS algorithm. CPLEX results are reported for the mono-objective problem when the objective is to minimize the routing cost since it is the only linear objective. The maintenance model without constraints was solved using Python 3.6. The joint maintenance scheduling and workforce routing model was then solved using C++. Finally, the CPLEX 12.10.0 version was used to solve the MILP using an exact method. This work has been compiled with GCC 7.4 in a Linux environment of a personal computer. Experiments were conducted using the CALCULCO computing platform in an AMD EPYC 7702 with 2CPU, two gigahertz, and one core was dedicated to each instance. The section 3.5.3

presents the results for tests executed on personal computer windows 7, 64-bit machine, with an intel i7-4510U processor (2*2.60 GHz) and 8 GB of RAM.

3.5.1 Instances Description

We use two sets of instances. The first one, denoted by Re, is inspired by industrial reality. In the Re instances, large and short maintenance durations are considered (Tcm and Tpm go from 0.5 to 48 h). We set the shape parameter of the Weibull distribution $\beta > 1$ to consider the wear-out period of the machines' life. These machines are, therefore, old. Short (27 km) and long distances (up to 500 km) are considered. The speed is fixed to 60 km/h, as adopted in real-world scenarios. The horizon is set to H = 101 h for a number of operations n = 12 and H = 80 h for n = 6. The penalty cost is fixed to c = 10. This cost is high enough to produce the desired effect of penalizing the objective functions whenever multiplied by the penalty variables. The depot time window is the interval between the lower bound $a_0 = 0$ and the upper bound b_{n+1}. For this class of instances, $b_{n+1} = H$. The Re dataset is shown in our work [98] and in Table 3.1. There are six machines that may need more than one maintenance task in the planning horizon in the class of instances Re. The travel time and distance between the same machine operations are naturally zero. The number of available vehicles is initially fixed to 4 for n = 12 and 2 for n = 6. We note that in some obtained solutions, we can use fewer vehicles than those available. The second set is derived from the well-known Solomon benchmark. These benchmark problems are available on Solomon's web page `http://web.cba.neu.edu/~msolomon/problems.htm`. We used the first 10, 25 and 50 machines of Solomon's classes of instances. The number of operations used in these instances varies from 23 to 91 operations, and the number of vehicles ranges from 5 to 35. This latter can also be reduced during the search.

Solomon's instances are classified into three groups:

- R: randomly distributed locations.
- C: geographically clustered locations.
- RC: partially randomly distributed and partially clustered locations.

There are six Solomon's classes with different locations coordinates: C101, C201, R101, R201, RC101 and RC201. The unitary speed is adopted. For each of these six classes, we consider the horizon $H = 100$ h with the latest arrival time at the depot $b_{n+1} = 200$ h and $H = 200$ h with $b_{n+1} = 400$ h. The opening time is $a_0 = 0$. The maintenance parameters were generated as described in [82]: $Tpm_i \sim U_c[5,10]$, $Tcm_i \sim U_c[15,30]$, $Cpm_i \sim U_c[100,200]$, $Ccm_i \sim U_c[400,800]$, $Cw_i \sim U_c[10,20]$, and $\sigma \sim t_{0i} * U_c[2,5]$. Only the parameter $\beta \sim U_c[2,6]$ is generated differently compared to the one described in [82] so that it can be closer to reality. The continuous uniform distribution U_c has been used to generate the random information. In the instances sets, the percentage of time windows tolerance is set to $tol = 7\%$. It is inspired by the values usually used in large-scale industries that consider a restricted time window. Another significant time windows tolerance is considered $tol = 30\%$.

t_{ij}	0	1	2	3	4	5	6	Cpm_i	Ccm_i	Cw_i	Tpm_i	Tcm_i	$f_i(t)$
0	0	0.45	0.83	0.67	1	0.67	7.5	-	-	-	0	0	-
1	0.45	0	1	1.33	0.67	1.17	7.5	193	425	19	1	2	$W(66,2)$
2	0.83	1	0	0.33	1.67	1.17	6	156	561	14	1	2	$W(100,2)$
3	0.67	1.33	0.33	0	0.83	1.17	5.5	138	561	13	1	3	$W(63,2)$
4	1	0.67	1.67	0.83	0	0.83	5	163	462	15	0.5	3	$W(84,2)$
5	0.67	1.17	1.17	1.17	0.83	0	4	200	400	20	0.5	3	$W(90,2)$
6	7.5	7.5	6	5.5	5	4	0	600	1000	30	24	48	$W(110,2)$

Table 3.1: Travel time (hours) between operations and data of the illustrative case study Re

3.5.2 Parameter setting and performance metrics

In Section (3.5.3), we test the influence of the use of the proposed greedy heuristics on the improvement of the GVNS algorithm results and instances with a number of operations that varies from 6 to 23. We have set for the subsection (3.5.3) as a stopping condition for the GVNS a maximum number of iterations $Iter_{max}$. For the two first objectives (3.23) and (3.24), $Iter_{max}$ equals 8 for 23 operations and 5 routes and 1 for all the other instances. For the last objective (**??**), $Iter_{max}$ equals 8 for 23 operations and 5 routes and 2 for all the other instances. The diversification parameter nS in the shaking phase is set to 3. We also use the value Best, which is the CPLEX value for the linear objective function (3.23).

In the case of the non-linear objectives, the Best Value of GVNS represents the minimum value of the GVNS algorithm in 5 runs for all the initial solutions when we fix the $Iter_{max}$ to 500 iterations. The percent decrease of CPU time when using the constructive initial solution CS compared to the same random feasible initial solution RS is given by:

$$TimeDev(RS,CS) = (\frac{Time(RS) - Time(CS)}{Time(RS)}) * 100\% \tag{3.32}$$

The gap between the objective value of the heuristic and the optimal solution or best solution is calculated as follows:

$$ObjDev(Best,Heuristic) = (\frac{Value(Heuristic) - Value(Best)}{Value(Best)}) * 100\% \tag{3.33}$$

The best insertion heuristic was used to construct an initial solution when testing all instances in subsection (3.5.4). In order to set the algorithm parameters, we have run several preliminary tests. We noticed that a high value of the shaking parameter nS and a high number of iterations have a negative impact on the computational time. The retained parameters for the maximum number of iterations is $iter_{max} = max(1, E[n/8])$ for the routing objective (3.23) and the maintenance objective (3.25), where the symbol E stands for the whole experiments parts of the integer portion of the number. For the failure objective (3.24), this parameter is $iter_{max} = max(1, E[n/4])$. The diversification parameter nS in the shaking phase is set to three. We compare the results of the CPLEX solver and our mono-objective GVNS algorithm for the routing objective function (3.23). In the CPLEX columns, we report the objective value, the CPU time, and whether the solution is optimal, feasible, or not found after 96 h of execution. In the GVNS algorithm, we indicate, respectively, the maximum, minimum, and average values for five runs. We also provide the corresponding maximum, minimum, and CPU time.

The pseudo-code of mono-objective GVNS and VND are shown in our work [98] and in Section 3.4. The mono-objective GVNS can directly be obtained with the MOGVNS/D algorithm presented in Chapter 4 when the initial popula-

tion is reduced to one individual. The gap between the objective value of our metaheuristic and the CPLEX solution is calculated as follows:

$$Gap(CPLEX, GVNS) = \left(\frac{Value(CPLEX) - Value(GVNS)}{Value(CPLEX)}\right) \times 100\% \qquad (3.34)$$

This gap represents the percent decrease of the cost objective when the GVNS algorithm is used in comparison to the use of the CPLEX solver. It is the case whenever the CPLEX solver only finds a feasible penalized solution, and the gap is different from zero.

The percent decrease of CPU time when using the GVNS algorithm compared to the CPLEX solver is given by:

$$ICPU(CPLEX, GVNS) = \left(\frac{Time(CPLEX) - Time(GVNS)}{Time(CPLEX)}\right) \times 100\% \qquad (3.35)$$

3.5.3 Numerical results and comparative study using the problem-specific constructive heuristics

We tested several instances to compare the different objective functions. The results are reported in tables 3.2 and 3.3. In Table 3.3, VND, GVNS*, and GVNS represent the results of VND, the best value of GVNS, and the average value of GVNS in 5 runs, respectively. We also indicate a value Best, which is the CPLEX value for the linear objective function (3.23) and the Best value of GVNS* for all the initial solutions when we fix the $Iter_{max}$ to 500 iterations for the non-linear two others objectives. An asterisk is used to mark the optimal solutions obtained by CPLEX. In Table 3.2, LS(F), LS(M), and LS(R) refer respectively to BILS applied with the GCHF, GCHM, and the random initial solution.

The greedy heuristics find a very near-optimal solution for instances with small values of penalties in the case of minimizing the failure risk as the cost objective (3.24) and for the maintenance cost objective (3.25). They are faster than the VND and GVNS, even when using them as initial solutions with VND and GVNS. The BILS provides a good solution when using the best neighborhood in the search procedure and the constructed initial solutions and is faster than VND and GVNS. The algorithms for the two first objectives are faster than the

algorithms considering the third objective. The time search of VND and BILS is inferior to the GVNS time for the model with the three objectives. We notice that for the two last objectives (3.24) and (**??**) for which initial solutions were proposed, VND is sufficient for finding the optimal solutions. For BILS, VND, and GVNS, the quality of the final solutions increases with the proposed initial solutions in comparison to using the random initial solution. The execution times with the constructed initial solutions are also less than the execution times when starting from random solutions, as one can notice in Table 3.3. Choosing a good initial solution is very important for reducing the execution time and improving the quality of the final solution rapidly. When using the greedy initial solutions, the execution time of our search algorithms considerably decreases.

Note that the tests for the actual subsection are realized for (3.24) considering only the cost of failure and penalty's cost without the expression of the waiting cost in the objective. The waiting time is still considered in the maintenance model and the maintenance cost of the routing model. The failure objective (3.24) and the maintenance objective (3.25) consider in this subsection 3.5.3 only that $\rho_{i,k} = 0$. The number of vehicles is also fixed to a predefined number $r \leq K$.

Inst	*n*	*r*		Routing OF				Failure OF				Maintenance OF			
				Best	LS(F)	LS(M)	LS(R)	Best	LS(F)	LS(M)	LS(R)	Best	LS(F)	LS(M)	LS(R)
Re	6	2	Value	110.88*	**110.88**	**110.88**	**110.88**	869.471	**869.471**	**869.471**	**869.471**	45.0156	**45.0156**	**45.0156**	45.0352
			Time	-	0.02	0.017	0.032	-	0.017	0.018	0.021	-	0.021	0.02	0.024
			ObjDev	-	0	0	0	-	0	0	0	-	0	0	0.04
			TimeDev	-	37.5	46.88	-	-	19.05	14.29	-	-	12.50	16.67	-
Re	12	4	Value	353.52*	382.8	378.96	397.68	2841.77	**2841.77**	**2841.77**	**2841.77**	100	100.02	100.02	100.064
			Time	-	0.027	0.026	0.039	-	0.025	0.026	0.035	-	0.043	0.045	0.057
			ObjDev	-	8.28	7.20	12.49	-	0	0	0	-	0.02	0.02	0.06
			TimeDev	-	30.77	33.33	-	-	28.57	25.71	-	-	24.56	21.05	-
*C*101	23	5	Value	225.078*	228.811	231.835	304.42	8097.44	**8097.44**	8100.5	8272.59	350.039	350.076	351.501	426.979
			Time	-	0.065	0.081	0.133	-	0.067	0.079	0.156	-	0.19	0.15	0.57
			ObjDev	-	1.226	2.564	34.67	-	0	0.037	2.163	-	0.010	0.417	21.98
			TimeDev	-	51.13	39.10	-	-	57.05	49.36	-	-	66.67	73.68	-
*C*101	23	8	Value	128.359*	132.027	131.671	137.131	7707.42	**7707.42**	**7707.42**	**7707.42**	235.46	**235.46**	235.472	235.464
			Time	-	0.098	0.095	0.129	-	0.051	0.056	0.134	-	0.15	0.153	0.34
			ObjDev	-	2.857	2.58	6.833	-	0	0	0	-	0	0.005	0.001
			TimeDev	-	24.03	26.36	-	-	61.94	58.21	-	-	55.88	55	-

Table 3.2: Results of the BILS with the first neighborhood

3.5.4 Numerical results of all instances

The results of the GVNS and CPLEX solver for the routing objective are represented in Table 3.4. Bold values are the minimum objective values obtained

Inst	*n*	*r*			CIS with GCHF			CIS with GCHM			Random IS			GCH	
				Best	VND	GVNS*	GVNS	VND	GVNS*	GVNS	VND	GVNS*	GVNS	GCHF	GCHM
The routing cost objective function															
Re	6	2	Value	110.88*	**110.88**	**110.88**	**110.88**	**110.88**	**110.88**	**110.88**	**110.88**	**110.88**	**110.88**	130.8	130.8
			Time	-	0.018	0.02	0.0408	0.019	0.019	0.0378	0.021	0.02	0.041	0.017	0.016
			ObjDev	-	0	0	0	0	0	0	0	0	0	17.97	17.97
			TimeDev	-	14.29	0	0.49	9.52	5	7.80	-	-	-	-	-
Re	12	4	Value	353.52*	**353.52**	**353.52**	**353.52**	378.96	**353.52**	**353.52**	377.76	**353.52**	**353.52**	622.8	590.88
			Time	-	0.03	0.052	0.0558	0.026	0.055	0.0738	0.034	0.055	0.0786	0.025	0.025
			ObjDev	-	0	0	0	7.20	0	0	6.86	0	0	76.17	67.14
			TimeDev	-	11.76	5.45	29.01	23.53	0	6.11	-	-	-	-	-
*C*101	23	5	Value	225.078*	**225.078**	**225.078**	**225.078**	**225.078**	**225.078**	**225.078**	228.432	**225.078**	226.517	661.591	630.193
			Time	-	0.089	1.31041	1.72849	0.106	1.57561	1.63801	0.172	1.71601	1.89385	0.039	0.043
			ObjDev	-	0	0	0	0	0	0	1.059	0	0.211	192.69	178.79
			TimeDev	-	48.26	23.64	8.73	38.37	8.18	13.51	-	-	-	-	-
*C*101	23	8	Value	128.359*	129.251	**128.359**	**128.359**	**128.359**	**128.359**	**128.359**	**128.359**	**128.359**	**128.359**	149.815	150.179
			Time	-	0.085	0.249602	0.443043	0.098	0.327602	0.361922	0.168	0.405603	0.508563	0.038	0.046
			ObjDev	-	0.694	0	0	0	0	0	0	0	0	16.715	16.999
			TimeDev	-	49.4	38.46	12.88	41.67	19.23	28.83	-	-	-	-	-
The failure cost objective function															
Re	6	2	Value	869.471	**869.471**	**869.471**	**869.471**	**869.471**	**869.471**	**869.471**	**869.471**	**869.471**	**869.471**	**869.471**	**869.471**
			Time	-	0.018	0.019	0.0446	0.023	0.02	0.038	0.036	0.024	0.0484	0.017	0.018
			ObjDev	-	0	0	0	0	0	0	0	0	0	0	0
			TimeDev	-	50	20.83	7.85	36.11	16.67	21.49	-	-	-	-	-
Re	12	4	Value	2841.77	**2841.77**	**2841.77**	**2841.77**	**2841.77**	**2841.77**	**2841.77**	**2841.77**	**2841.77**	**2841.77**	2851.67	2851.67
			Time	-	0.034	0.052	0.045	0.056	0.045	0.0608	0.08	0.055	0.0724	0.024	0.026
			ObjDev	-	0	0	0	0	0	0	0	0	0	0.35	0.35
			TimeDev	-	57.5	5.45	37.85	29.50	18.18	16.02	-	-	-	-	-
*C*101	23	5	Value	8097.44	**8097.44**	**8097.44**	**8097.44**	8100.5	**8097.44**	**8097.44**	8272.59	**8097.44**	8106.71	8755.39	8664.85
			Time	-	0.093	2.82362	3.31034	0.097	2.58962	2.86106	0.195	2.83922	3.48506	0.044	0.043
			ObjDev	-	0	0	0	0.037	0	0	2.163	0	0.114	8.125	7.007
			TimeDev	-	52.31	0.55	5.01	50.26	8.79	17.91	-	-	-	-	-
*C*101	23	8	Value	7707.42	**7707.42**	**7707.42**	**7707.42**	**7707.42**	**7707.42**	**7707.42**	**7707.42**	**7707.42**	**7707.42**	**7707.42**	**7707.42**
			Time	-	0.071	0.202801	0.252722	0.064	0.187201	0.287042	0.191	0.249601	0.386882	0.037	0.042
			ObjDev	-	0	0	0	0	0	0	0	0	0	0	0
			TimeDev	-	62.83	18.75	34.68	66.49	25	25.81	-	-	-	-	-
The maintenance cost objective function															
Re	6	2	Value	45.0156	**45.0156**	**45.0156**	**45.0156**	**45.0156**	**45.0156**	**45.0156**	45.0352	**45.0156**	**45.0156**	45.0521	45.0521
			Time	-	0.021	0.035	0.0382	0.021	0.031	0.048	0.096	0.043	0.0794	0.02	0.018
			ObjDev	-	0	0	0	0	0	0	0.04	0	0	0.08	0.08
			TimeDev	-	78.13	18.60	51.89	78.13	27.91	39.55	-	-	-	-	-
Re	12	4	Value	100	100.02	**100**	**100**	100.02	**100**	**100**	**100**	**100**	**100**	100.132	100.132
			Time	-	0.056	0.227	0.2404	0.059	0.237	0.2702	0.068	0.279	0.2858	0.023	0.025
			ObjDev	-	0.02	0	0	0.02	0	0	0	0	0	0.13	0.13
			TimeDev	-	17.65	18.64	15.89	13.24	15.05	5.46	-	-	-	-	-
*C*101	23	5	Value	350.039	350.076	**350.039**	350.054	350.076	**350.039**	350.054	**350.039**	**350.039**	350.054	780.088	747.736
			Time	-	0.28	5.05443	5.68156	0.31	8.81406	10.0309	0.86	9.36006	11.9497	0.04	0.04
			ObjDev	-	0.01	0	0.042	0.01	0	0.042	0	0	0.042	122.857	113.615
			TimeDev	-	67.44	46	52.45	63.95	5.83	16.06	-	-	-	-	-
*C*101	23	8	Value	235.46	**235.46**	**235.46**	**235.46**	**235.46**	**235.46**	**235.46**	**235.46**	**235.46**	**235.46**	235.505	235.505
			Time	-	0.2	1.23241	1.49137	0.25	1.66921	1.93753	0.45	1.79401	2.12161	0.04	0.04
			ObjDev	-	0	0	0	0	0	0	0	0	0	0.019	0.019
			TimeDev	-	55.56	31.30	29.71	44.44	6.96	8.68	-	-	-	-	-

Table 3.3: Results of the GVNS and the VND

by either the GVNS or the solver. Whenever a number in the improvement column is bold, there is an improvement. The quality of the initial solution obtained with the best insertion heuristic considerably reduces the CPU time since we start from a reduced objective compared to random initial solutions. The impact of good initial solutions on the CPU time is illustrated in our work [98] and in Section 3.5.3. Indeed, providing a well-chosen starting solution positively influences the execution time and helps to improve the final solution quality rapidly. Optimizing the failure cost requires more iterations to reach high-quality solutions than optimizing the routing cost or the maintenance cost.

For small instances (less than 26 operations) of Solomon's class C, both CPLEX and GVNS were able to solve the problem optimally. However, the GVNS algorithm consumed less CPU time than CPLEX for most of the instances. When the number of operations increases to more than 34, some optimums are obtained for class C, R, or RC instances. For these instances, CPLEX returns mostly either feasible solutions or optimal solutions but after a very long CPU time (more than 48 h). In many instances, CPLEX fails to find optimal solutions after one week. In most cases, the GVNS algorithm improves the objective value of feasible solutions obtained by CPLEX. The gap of GVNS for most instances is 0%. The algorithm is also very robust since the difference between the maximum, minimum, and average values is very small, almost negligible.

For the 47 tested instances, CPLEX returned either feasible or optimal values for 29 instances and failed to obtain any feasible solution for the remaining 18 instances, even after a week of execution. For the 29 instances for which we obtained either feasible or optimal solutions using CPLEX, the average gap or objective improvement of GVNS over CPLEX is 18.47%. The average improvement of CPU is 32.81%. When removing the four Re instances for which the number of operations is inferior to 12, we obtain an average objective improvement in favor of GVNS of 21.42 % and an average improvement of the CPU of 77.96%. The gap value is 0 for 17 instances among 29 instances solved. Therefore, the proposed GVNS found the optimums in these instances. The GVNS algorithm improves the CPLEX results on 11 instances out of the 29 instances for which CPLEX returns either feasible or optimal solutions. However, it obtains a slightly worse value for only one instance. The average gap for the 12 instances where the

gap is different from 0 is 44.61%. This means that when the gap is different from zero (meaning only a feasible solution is found), GVNS finds a better objective value than CPLEX, with a percentage of 44.61%. The average percent decrease of CPU time when using GVNS compared to CPLEX is 85.73% in this case.

Our GVNS algorithm outperforms CPLEX in terms of running time, the longest CPU time being only 3 h (10906.1 s) compared to the maximum CPU time reached by CPLEX of 95.53 h (343932 s).

Instance	*n*	CPLEX			GVNS						Improvement	
		Objective	Status	CPU(s)	max	min	avg	max cpu	min cpu	avg cpu	Gap	ICPU
RE/6/2/80/80/0.07/2	6	**110.88**	optimal	0.03	**110.88**	**110.88**	**110.88**	0.244	0.233	0.239	**0**	−676.67
RE/6/4/101/101/0.07/2	12	**353.52**	optimal	0.92	362.64	**353.52**	355.344	3.66	2.64	3.01	**0**	−186.96
RE/6/2/80/80/0.30/2	6	**98.64**	optimal	0.09	105.36	**98.64**	92.784	0.18	0.12	0.146	**0**	−33.33
RE/6/4/101/101/0.30/2	12	**325.92**	optimal	1.74	349.44	**325.92**	336.768	5.94	3.49	4.224	**0**	−100.57
*C*101/10/8/100/200/0.07/2	23	**119.699**	optimal	131.61	121.451	**119.699**	120.133	131.56	94.83	113.878	**0**	**27.95**
*C*101/10/5/100/200/0.07/2	23	**225.078**	optimal	2186.44	226.826	**225.078**	225.588	161.95	56.89	107.518	**0**	**97.40**
*C*101/10/7/100/200/0.07	23	**82.536**	optimal	55.28	82.996	**82.536**	82.628	211.3	156.28	179.042	**0**	−78.36
*C*101/25/18/100/200/0.07	54	**192.184**	optimal	114985	196.632	**192.184**	193.963	5586.44	4218.19	4868.12	**0**	**96.33**
*C*201/10/7/100/200/0.07	24	**77.168**	optimal	44.19	**77.168**	**77.168**	**77.168**	148.5	81.96	116.002	**0**	−85.47
*C*201/25/18/100/200/0.07	52	223.01	feasible	83458	228.691	**222.451**	224.947	3566.31	2902.28	3245.33	**0.25**	**96.52**
*R*101/10/7/100/200/0.07	34	507.37	feasible	51587.8	544.467	**450.746**	492.615	649.18	540.83	611.382	**11.16**	**98.95**
*R*101/25/18/100/200/0.07	68	244302	feasible	67536.8	316.103	**299.882**	307.842	13044.3	10906.1	12133.2	**99.88**	**83.85**
*R*201/10/7/100/200/0.07	34	**341.741**	feasible	236032	354.432	**341.741**	348.367	716.37	487.04	578.662	**0**	**99.79**
*R*201/25/18/100/200/0.07	65	-	-	-	246.648	**237.633**	239.436	10748.3	7093.85	8823.51	-	-
*RC*101/10/7/100/200/0.07	33	752.961	feasible	51859	777.745	**716.459**	747.226	752.87	541.27	659.03	**4.85**	**98.96**
*RC*101/25/18/100/200/0.07	62	18763.7	feasible	14382.1	262.036	**261.552**	261.746	7660.66	5238.13	6368.32	**98.61**	**63.58**
*RC*101/50/35/100/200/0.07	89	-	-	-	**560.06**	**560.06**	**560.06**	35389.4	31967.6	33868.1	-	-
*RC*201/10/7/100/200/0.07	33	678.987	feasible	241174	676.96	**612.94**	637.187	750.19	320.74	552.822	**9.73**	**99.87**
*RC*201/25/18/100/200/0.07	60	281.088	feasible	343932	321.337	**277.384**	287.31	6173.46	4781.44	5537.82	**1.32**	**98.61**
*RC*201/50/35/100/200/0.07	91	-	-	-	713.324	**706.072**	708.94	34343.8	26443.5	29329.9	-	-
*C*101/10/8/100/200/0.30/2	23	**82.544**	feasible	225506	83.344	**82.544**	82.704	159.31	77.38	121.358	**0**	**99.97**
*C*101/10/5/100/200/0.30/2	23	**82.544**	optimal	18631.6	83.736	**82.544**	82.9424	147.76	78.27	119.06	**0**	**99.58**
*C*101/10/7/100/200/0.30	26	**65.044**	optimal	2288.14	**65.044**	**65.044**	**65.044**	251.2	152.27	190.812	**0**	**93.35**
*C*101/25/18/100/200/0.30	54	42379.5	feasible	78964.4	166.876	**159.068**	163.753	6284.01	3802.55	4849.62	**99.62**	**95.18**
*C*201/10/7/100/200/0.30	24	**70.116**	optimal	356.48	**70.116**	**70.116**	**70.116**	109.47	84.34	99.466	**0**	**76.34**
*C*201/25/18/100/200/0.30	52	116913	feasible	80200.3	180.7	**179.144**	179.758	4440.74	3303.94	4050.43	**99.85**	**95.88**
*R*101/10/7/100/200/0.30	34	94.96	feasible	236317	107.644	95.528	101.242	805.61	431.89	558.634	−0.60	**99.82**
*R*101/25/18/100/200/0.30	68	-	-	-	208.588	**203.616**	205.714	13602.9	11030.5	12438.2	-	-
*R*201/10/7/100/200/0.30	34	**83.28**	optimal	24458.8	87.18	**83.28**	84.06	791.51	694.42	752.572	**0**	**97.16**
*R*201/25/18/100/200/0.30	65	-	-	-	195.636	**190.732**	192.26	13517.8	7052.6	9646.78	-	-
*RC*101/10/7/100/200/0.30	33	**173.116**	feasible	241311	207.477	**173.116**	189.96	747.57	425.65	571.79	**0**	**99.82**
*RC*101/25/18/100/200/0.30	62	-	-	-	232.152	**229.78**	230.761	7774.12	6244.38	7005.41	-	-
*RC*201/10/7/100/200/0.30	33	**117.011**	feasible	258832	128.91	**117.011**	121.77	638.44	439.24	555.072	**0**	**99.83**
*RC*201/25/18/100/200/0.30	60	-	-	-	224.492	**219.724**	221.649	7803.89	5486.56	6723.49	-	-
*C*101/10/18/200/400/0.07/2	52	148.156	feasible	342141	**130.743**	**130.743**	**130.743**	10770.5	7695.34	8634.23	**11.75**	**97.75**
*C*101/10/18/200/400/0.07	58	-	-	-	103.8	**95.972**	98.6456	19472.2	12096.7	15370.3	-	-
*C*201/10/18/200/400/0.07	54	-	-	-	96.216	**96.16**	96.1712	7471.94	6153	6749.16	-	-
*R*101/10/18/200/400/0.07	73	-	-	-	**71.892**	**71.892**	**71.892**	32985.8	25402.7	28932	-	-
*R*201/10/18/200/400/0.07	72	-	-	-	**71.896**	**71.896**	**71.896**	30988.5	22842.1	25594.4	-	-
*RC*101/10/18/200/400/0.07	71	-	-	-	**86.104**	**86.104**	**86.104**	24206.5	17693.7	19926	-	-
*RC*101/10/18/200/400/0.07	72	-	-	-	**86.104**	**86.104**	**86.104**	30123	21641.7	24153.9	-	-
*C*101/10/18/200/400/0.30/2	52	11986.3	feasible	252973	102.328	**100.588**	101.198	13018.1	9452.88	10463.7	**99.16**	**96.26**
*C*101/10/18/200/400/0.30	58	-	-	-	84.864	**84.624**	84.72	18539.5	9654.02	13218.7	-	-
*C*201/10/18/200/400/0.30	54	-	-	-	94.72	**88.104**	90.6752	11208.8	6835.28	9404.82	-	-
*R*101/10/18/200/400/0.30	73	-	-	-	82.712	**71.888**	77.608	31452	25678.5	29740	-	-
*R*201/10/18/200/400/0.30	72	-	-	-	81.136	**71.896**	79.288	32522.2	25003.3	28856.2	-	-
*RC*101/10/18/200/400/0.30	71	-	-	-	94.792	**86.104**	87.8416	26959.8	19969.3	24013.2	-	-
*RC*201/10/18/200/400/0.30	72	-	-	-	92.192	**86.104**	87.3216	29509.4	19741.8	23851.1	-	-

Table 3.4: Results of mono-objective GVNS for the routing objective

3.6 Concluding remarks

In this chapter, we addressed a joint maintenance and workforce routing problem. We proposed a novel mathematical model that aims to minimize either maintenance, failure, or transport costs in the case of time-based preventive maintenance. The set of machines is supposed to be geographically distributed and subject to non-deterministic failures. The joint maintenance and routing problem's objective is to simultaneously determine the optimal times to perform preventive maintenance operations on each machine and find the optimal sequence of these operations that simultaneously minimizes the maintenance and routing cost.

There are many contributions to this chapter. Firstly, we proposed a nonlinear stochastic failure cost that uses information from equipment degradation. It includes direct costs (failure cost) and indirect costs (production losses), illustrated by the waiting time. This objective is precious for industries where failures seriously influence personnel safety and cause environmental damage. We also investigate another maintenance cost previously proposed in the literature that aims to balance preventive and corrective maintenance operations costs. Both costs also consider the time of the last restoration. The proposed model considers maintenance operations time windows, penalties related to late arrival, and maintenance costs under uncertainty which are interesting features of real industrial problems.

Novel constructive heuristics based on the failure and maintenance behavior have been proposed to generate initial solutions. Three heuristic approaches were then developed and compared: Best Improvement Local Search (BILS), Variable Neighborhood Descent (VND) et General Variable Neighborhood Search (GVNS). The mathematical model and the proposed algorithms have been evaluated on randomly generated instances. We had first shown that the quality of the initial solution, obtained with the greedy constructive heuristics, considerably reduces the CPU time and improves the objective value when using BILS, VND, and GVNS. Furthermore, for the single objective variant of the problem, the computational results for the linear routing objective demonstrate that the GVNS significantly outperforms the results of the commercial solver CPLEX in

terms of solution quality and CPU time.

The next chapter is dedicated to the bi-objective variant of the problem. It also extends the proposed VND and GVNS algorithms to deal with multi-objective combinatorial problems.

Chapter 4

The Joint Maintenance Scheduling and Workforce Routing Problem: Multi-objective Variable Neighborhood Descent and Search algorithms

Outline of the current chapter

4.1 Introduction

Workforce Scheduling consists of constructing work timetables for the personnel to permit the organization to meet the products and services demand. This NP-hard problem is encountered in many organizations and businesses: service maintenance, home healthcare, call centres, etc [123]. This research is motivated by an extended situation often faced by maintenance service providers. In this situation, the latter has to define the maintenance schedule that includes the opportune maintenance times and the workforce routing. The service provider tries to reduce the number of maintenance operations as much as possible. At the same time, he seeks to avoid the huge costs resulting from carrying out corrective maintenance operations on broken machines. The challenge is therefore finding the optimal number of visits to perform maintenance operations and scheduling the technicians' visits for these maintenance operations.

This chapter addresses the following problem: performing maintenance ac-

tivities at the right time on geographically distributed machines subjected to random failures. This problem requires determining the sequence of maintenance operations for each technician to minimize the total expected costs while ensuring a high level of machine availability. To date, research in this area has dealt with routing and maintenance schedules separately. The purpose of this chapter is to determine the optimal maintenance and routing plan simultaneously. The bi-objective variant of the problem modeled in Chapter 3 is proposed. The first objective is to minimize the travel cost related to technicians' routing. The second objective can either minimize the total preventive and corrective maintenance or failure costs.

This problem considered in this chapter is bi-objective. To solve it, we focused on designing multi-objective optimization algorithms. Multi-objective optimization problems are ubiquitous nowadays since most relevant situations involve multiple criteria. Two or more objectives must be included in a multi-objective optimization problem (MOP) to achieve an optimal decision. Optimizing one objective with respect to the effect of all the other objectives is essential. The objectives are often conflicting, and there is no unique best solution for all of them. Specifically, we instead seek to determine the set of efficient solutions, also called Pareto optimal solutions, non-dominated solutions, or Pareto front. These solutions are non-dominated by any other feasible solution. The decision-maker then chooses his preferred solution from them. Indeed, the decision maker's preferences are generally considered compatible with the dominance relation.

Variable Neighborhood Search (VNS) is a metaheuristic that dates back to the work of Mladenovic and Hansen [87] in 1997. It has been proposed to solve single-objective problems. It is based on the systematic change of neighborhood structures in both the descent and the perturbation phase [93]. VNS has been successfully applied to numerous mono-objective optimization problems. Its adaptation to multi-objective problems was, however, ignored by the research community until 2008. During that year, the first multi-objective VNS algorithm was proposed by Geiger [25]. Multi-objective VNS is still rarely studied in the literature [93]. The main objective of this chapter is to propose adaptations of the VND and GVNS frameworks to solve this problem and multi-objective combinatorial optimization problems in general.

Herein, the main contributions of this chapter are summarized as follows.

(1) The bi-objective variant of the Joint Maintenance Scheduling and Workforce Routing Problem proposed in Chapter 3 is studied. The first objective minimizes the total travel cost related to technicians' routing, and the second objective can either minimize the total preventive and corrective maintenance cost or the failure cost. A penalty cost is incurred when the maintenance activities are performed after the optimal time intervals. To the best of our knowledge, the failure and maintenance costs adopted have not been used in a multi-objective study with the routing objective.

(2) Variable Neighborhood Descent and General Variable Neighborhood Search variants have been designed and implemented to solve the bi-objective problems. The proposed multi-objective VND and GVNS algorithms are based on the Pareto dominance strategy and start with a unique initial solution. The algorithm MOVND/P is designed to be an intensification local search component of the GVNS algorithms, whereas MOVND/PI and MOGVNS/P are intended to solve multi-objective problems. They use a proposed multi-objective best improvement strategy called MOBI/P and new adaptations of VNS mechanisms in the multi-objective context.

(3) Extensive experiments demonstrated that the proposed algorithms outperformed the literature algorithms. We also test some other GVNS variants for comparison reasons. An analysis is conducted to clearly show the differences between the proposed algorithms' performance and the literature algorithms. It permits measuring the influence of the proposed mechanisms to improve the results.

This chapter is presented as follows: Section 4.2 surveys multi-objective optimization literature's algorithms. Next, section 4.3 recalls the general approach adopted to tackle and solve the problem. Then, section 6.4 describes the proposed algorithms and delineates some variants. Subsequently, the computational experiments are conducted in Section 6.5. Finally, Section 4.6 summarizes and concludes the work and findings.

4.2 Literature review

Multi-objective problems are classified based on the role played by the decision-maker (DM) in the solution process. Four classes can support the DM in finding his preferred optimal solution. The four classes include no-preference, a priori, a posteriori, and interactive methods [100]. In addition, it is possible to formulate some objectives as constraints in some problems, like in [103]. The third class of MO methods is composed of the a posteriori methods. In this class, the multi-objective formulation of the problem is maintained, and the purpose is to find the Pareto optimal front. The DM then chooses a non-dominated solution from the obtained Pareto front according to his preferences. This category includes methods such as the Epsilon constraint method, local search metaheuristics, Evolutionary Multi-Objective algorithms (EMO), including population-based algorithms. They are divided into three main groups: aggregation, dominance, and performance indicator-based algorithms.

Local search metaheuristics (VNS, Tabu Search, Simulated Annealing, etc.) were proven to solve single-objective VRP variants effectively. These methods are, however, less used compared to population-based metaheuristics (Genetic Algorithms, Particle Swarm Optimization, etc.) in the multi-objective context. The main reason is that population-based metaheuristics operate with a set of initial solutions (population), which can be identified as the Pareto front and ensure diversification. In contrast, local search metaheuristics traditionally use only one initial solution. Duarte et al. [93] explored solving multi-objective combinatorial optimization problems using the Variable Neighborhood Search (VNS) metaheuristic. They proposed an adaptation of the shaking procedure, the improvement method, and the acceptance criterion within different VNS schemas (Reduced VNS, VND, and General VNS). They used the approximate Pareto front found during the search process as the incumbent solution to a multi-objective problem (a population of the solution instead of only one solution). A local search procedure based on the single objective VND, named VND-i, improves each objective i separately and returns a set of non-dominated points E. Then, the Multi-Objective VND starts with a random non-exploited point in E and explores its neighborhoods again with the VND-i procedure.

An improvement is considered when at least one new point has been added to the archive of non-dominated points. They tested these frameworks over two multi-objective combinatorial optimization problems (The Knapsack problem and the Antibandwidth-Cutwidth problem). Their best multi-objective VNS variant outperformed the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and the Strength Pareto Evolutionary Algorithm 2 (SPEA-2). Queiroz et al. [94] presented a Multi-Objective Variable Neighborhood Descent (MO-VND) to solve a bi-criteria parallel machine scheduling problem. The objectives are the minimization of the makespan and the flow time. The method is similar to the one proposed by Duarte et al. [93]. Compared to [93], the local search in the VND was implemented as a sequential search through the neighborhoods to reduce the execution time. Test results highlighted that this heuristic has a better set of non-dominated solutions compared to the well-known (NSGA-II). Ke et Zhai [95] proposed a Multi-Objective Adaptive Large Neighborhood Search (MOALNS) for a Vehicle Routing Problem (VRP), whose objectives are to minimize the total travel time and the cumulative time: the total arrival time to all customers. Their MOALNS involves a population of solutions over time and employs an external archive to store the non-dominated solutions. An adaptive probabilistic rule based on Pareto dominance was proposed with the roulette-wheel selection method to select a combination of destroy-repair operators. A weighted sum objective function was used to evaluate the quality of a move in destroy-repair operators or local search moves. Rifai et al. [96] presented a multi-objective ALNS based on the Pareto dominance for the distributed permutation flow shop scheduling problem. Three objectives were considered: minimizing makespan, total cost, and average tardiness. The algorithm starts with an initial solution and returns a unique best solution. Its performance exceeded NSGA-II.

Population-based metaheuristics and Evolutionary multi-objective algorithms have been very effective for multi-objective problems. Their main advantages are evolving a population of solutions simultaneously and having specific operators to ensure a good diversification in the Pareto front approximation. However, population-based methods such as Genetic algorithms show slow convergence when applied to single objective VRPs [104]. Researchers have proved that embedding local search methods in evolutionary algorithms significantly impacts

their performance in combinatorial optimization problems. Genetic Algorithms hybridized with other search techniques can achieve outstanding results on VRPs. Cota et al. [90] introduced two MO-ALNS algorithms for an unrelated parallel machine scheduling problem with setup times to minimize the makespan and the total energy consumption. The first algorithm is a direct extension of single-objective ALNS with Learning Automata by using multi-objective local search. The second algorithm employs the decomposition approach similar to the MOEA/D algorithm and uses the Tchebycheff aggregation function to evaluate solutions. They concluded that the MO-ALNS/D algorithm found better results than MO-ALNS. The reason is that MO-ALNS does not offer diversification control in the Pareto front approximation, where diversification is ensured in MO-ALNS/D since it uses a population of solutions.

The weighted sum approach does not usually work well with the multi-objective problem when the Pareto front is not convex since it provides only supported (convex) non-dominated solutions. This method is, however, used in some well-known multi-objective algorithms such as MOEA/D. Reference methods in multi-objective optimization are posteriori methods based on population and use the Pareto dominance concept. However, population-based methods show slow convergence when applied to single objective VRP [104]. Trajectory-based metaheuristics are very efficient for the single objective VRP but are less used in the multi-objective context since they traditionally use one unique solution. To the best of our knowledge, there is no reference multi-objective local search-based method apart from Pareto Local Search (PLS) [69]. Multi-objective trajectory-based metaheuristics efficient for this problem and VRP, in general, need to be proposed. These methods must be adapted to ensure the usual sufficient diversification obtained using a population of solutions and genetic operators.

A relatively small amount of research deals with the multi-objective variant of the problems defined in the literature section of Chapter 3, especially those of the second stream of research. The multi-objective formulation is considered in [86] [129] among all papers examined. Furthermore, the problem we defined in Chapter 3 is NP-hard. The NP-hardness remains when multiple objectives are considered. Therefore, heuristic-based approaches are necessary

for large instances since exact methods are practically limited. Due to their success in solving many mono-objective combinatorial problems, VND and GVNS algorithms were extended in several studies to deal with multi-objective optimization problems. However, most of these extensions do not question the utility of using or not important properties working well for evolutionary algorithms such as dealing with a population of solutions, aggregation, etc. They also do not propose a specific local search strategy for multi-objective optimization apart from Paquette et al. [69]. Instead, most literature algorithms apply mono-objective local search by dealing with each objective separately [93] or use a previously proposed improvement local search strategy [70] [90] such as Pareto Local Search [70].

The present work comes with the novelty of exploring the combination of vehicle routing and maintenance problems in a unique bi-objective model and proposes novel VND and GVNS algorithms to solve it. We consider the previous model with two objectives to be minimized simultaneously. The first objective minimizes the total travel cost and the penalty cost. The second objective can be minimizing the total preventive and corrective maintenance cost or the failure cost. Penalties for late arrivals are also associated with the second considered objective. The failure and maintenance costs adopted have not been used in a multi-objective study with the routing objective. The workforce and maintenance costs represent the highest costs in a plant. Failing to optimize these costs together can lead to a serious loss for the manufacturing company and reduce its profitability. The maintenance and transport are support processes often outsourced due to their importance. They are directly linked for a service provider to the production of its service. This model simultaneously optimizes these costs in an organization, making it appropriate for real-world situations. Maintenance service providers can use this model to provide the best services to their customers whenever the maintenance strategy adopted by those customers is time-based maintenance, including preventive and corrective maintenance. To efficiently solve the integrated maintenance and routing problem, multi-objective new adaptations of the variable neighborhood descent and the general variable neighborhood search algorithms are proposed. This chapter describes the design of the improvement method, the new best improvement

strategy proposed MOBI/P, the acceptance criterion, the stopping criterion, and the approach adopted to reduce the computational time. The chapter finally presents an analysis to demonstrate the efficiency and novelty of the proposed mechanisms compared to the literature. This work differs from the literature since it proposes novel adaptations of VND and GVNS algorithms to solve multi-objective problems. Indeed, it does not use the existing local search strategies but a new one called MOBI/P. Moreover, the VND algorithm, which is to be used as an intensification algorithm, is designed to be far faster than the GVNS algorithm. In addition, these algorithms include more design features to lead to better solutions than the literature.

4.3 The description of the general approach

The maintenance model is solved to determine the optimal time ϕ_i for each maintenance operation i that minimizes the total maintenance cost. The optimal date ϕ_i to perform a PM operation i is used to determine the durations d_i for the PM operations, the total number of operations n and a time window interval $[a_i, b_i]$ that minimizes the total maintenance cost. The above steps have been adopted by Lopez-Santana et al. [82]. We then solve the defined joint maintenance and routing model that minimizes the routing and the maintenance costs or the routing and the failure costs. Penalties for late arrival are considered in both cases. In the first case, the maintenance cost is minimized considering the workforce routing constraints that incorporate maintenance requirements (maintenance time windows, etc.). Integrating technical maintenance requirements with transport management is the first merit of the approach. In the second case, the failure cost is minimized to reduce the risk of failures. However, the optimal time chosen must be within a time window that minimizes the total maintenance cost. This latter case considers three objective requirements simultaneously while the minimization of maintenance cost is included in time windows constraints. Expressing some objectives of optimization problems as constraints reduces the problem complexity as adopted in [103]. The second merit of our approach is integrating several maintenance strategies. Indeed, the chosen optimal time reduces the risk of failures, the maintenance cost, and

the travel cost (risk-based maintenance, etc.). The outputs of the problem are shown in Figure 4.1. The workforce routing constraints influence the time to perform maintenance operations, the time of the last restoration that depends on that latter in the combined model, and the waiting time. All the variables are interconnected. Therefore, it is essential to model an integrated problem and to solve it using multi-objective optimization.

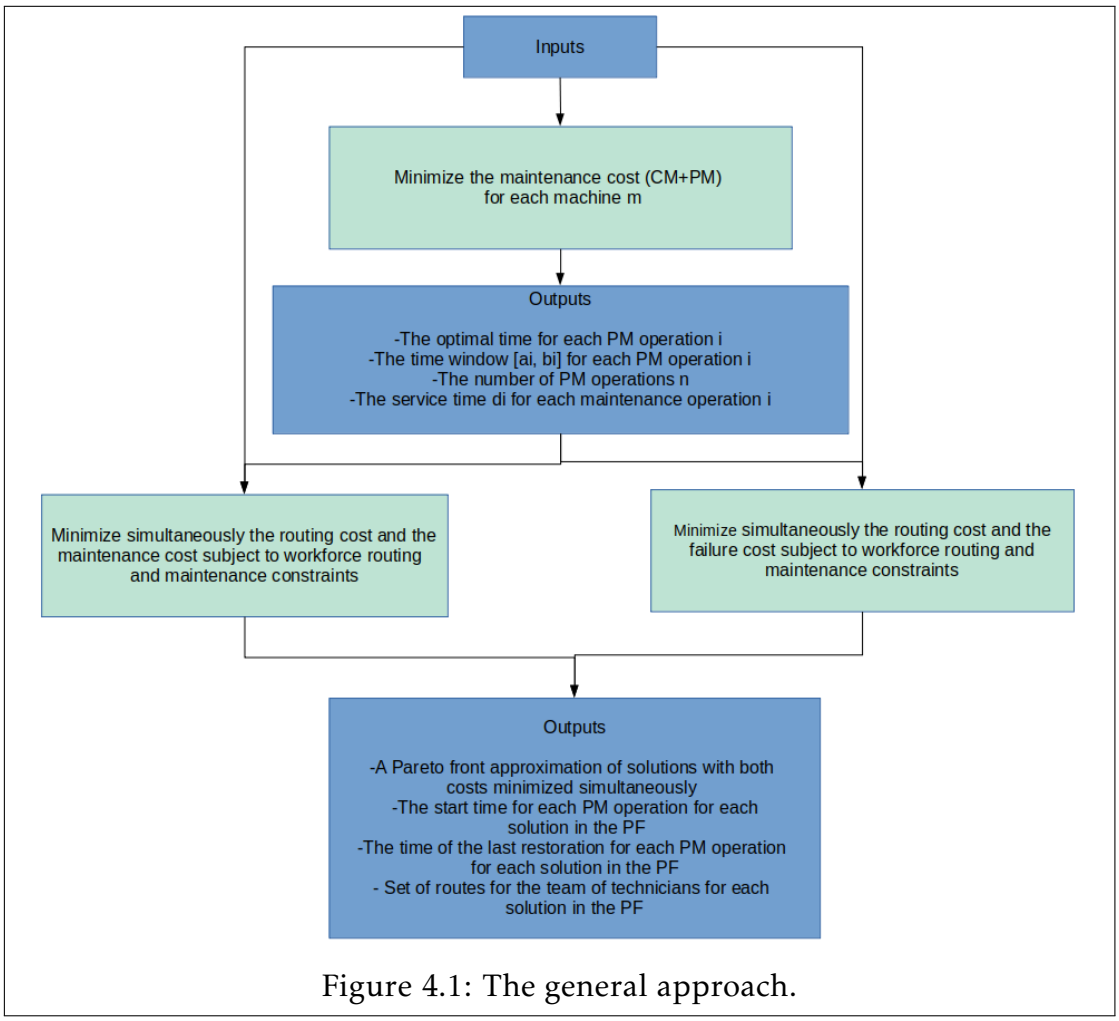

Figure 4.1: The general approach.

4.4 Proposed multi-objective algorithms

The proposed bi-objective mathematical model is a mixed integer nonlinear program (MINLP) which cannot be solved by commercial solvers in a reasonable

time when applied to large-scale instances. VRPTW is NP-hard [86], and the classical maintenance scheduling problem using operations research techniques is NP-hard as well [86]. Naturally, the combined problem is an NP-hard problem.

The following section presents an adaptation of variable neighborhood descent (VND) and general variable neighborhood search (GVNS) to the multi-objective context to solve the proposed bi-objective combined maintenance and routing problem. At first, multi-objective notions used in this chapter are defined; then the solution representation is shown. Next, the functions used by the algorithms and neighborhood structures are presented. Finally, the proposed multi-objective algorithms are described, as well as their novelty compared to the literature. The proposed algorithms extend single objective variable neighborhood descent (VND) and general variable neighborhood search (GVNS) proposed by Mladenovic and Hansen [88] to solve multi-objective problems. The algorithm MOVND/P is designed to be an intensification local search component of the GVNS algorithms, whereas MOVND/PI and MOGVNS/P are intended to solve the multi-objective problem. They use the Pareto dominance strategy and start with a unique initial solution. They also incorporate novel proposed strategies. Additionally, they all use a novel multi-objective best improvement strategy called MOBI/P. The other GVNS variants aim to measure the impact of the inclusion of a decomposition strategy on the algorithms. MOGVNS/D uses a weighted sum method instead of the Pareto dominance concept and a population of solutions, whereas MOGVNS/CDP tests the use of a population of solutions with MOGVNS/P.

4.4.1 General notions about multi-objective optimization

Pareto optimality

> **Pareto Optimal Dominance**: "The solution s dominates another solution $s^{'}$" is denoted by $f_i(s) \prec f_i(s^{'})$ that is ensured if these both conditions are verified $\forall i \in \{1, 2, \ldots, k\}, f_i(s) \leq f_i(s^{'})$ and $\exists j \in \{1, 2, \ldots, k\}, f_j(s) < f_j(s^{'})$ where k is the number of objectives.
>
> **Pareto Optimal Dominance Negation**: "The solution s does not dominate

another solution $s^{'}$" is denoted by $f_i(s) \nprec f_i(s^{'})$.

Pareto Weakly Dominance: "The solution s weakly dominates another solution $s^{'}$" is denoted by $f_i(s) \preceq f_i(s^{'})$ if $f_i(s) \leq f_i(s^{'}), \forall i \in \{1, 2, \ldots, k\}$.

Incomparable Solutions: Two solutions s and $s^{'}$ are incomparable if they are equally good. This means neither solution dominates the other and is denoted by $f_i(s) \nprec f_i(s^{'})$ and $f_i(s^{'}) \nprec f_i(s)$.

Pareto solutions

Efficient solution: a feasible solution is efficient if there does not exist any other feasible solution that dominates it [97].

Non-dominated point: represents the image in the objective space of an efficient solution [97].

Supported efficient solutions: are optimal solutions of a weighted sum method. The weights used have to be positive. The image of supported efficient solutions (supported non-dominated points) in the objective space are located on the convex part of the front [97].

Non supported efficient solutions: are efficient solutions that are not optimal solutions of any weighted sum method considering that the weights are positive [97].

4.4.2 Initial population generation

We use a weighted aggregation of the two objectives when generating an initial solution. The weighted sum method is used in MOGVNS/D to evaluate and compare solutions. In contrast, the other proposed algorithms (MOVND/P, MOVND/PI, MOGVNS/P, and MOGVNS/CDP) use a multi-objective evaluation and can therefore start with any weightless solution. The weighted sum method is a linear combination of weights. The evaluated function using this method is:

$$f(s) = w_1 * f_1(s) + w_l * f_l(s), \quad w_1 + w_l = 1, \quad l = 2, 3 \tag{4.1}$$

The algorithms start with initial solutions constructed using the best insertion heuristic. This latter calculates the minimum insertion cost for each operation that has to be inserted in the solution. The insertion cost of an operation i represents the difference between the cost solution with and without operation i. At each iteration of the heuristic, the operation with the minimum insertion cost is inserted at its best position in the current solution. The process stops when all the operations are inserted. The proposed algorithms, MOGVNS/CDP and MOGVNS/D deal with a population of solutions. Each population individual is generated using the best insertion heuristic and specific weighted sum method. The detailed procedure is described in Algorithm 5. In decomposition approaches, each population individual is associated with a subproblem. It is necessary to decompose the multi-objective problem effectively to reach the maximum parts of the Pareto front. Subproblems are defined in our case by firstly generating the weights. These weights (or search directions) are chosen to explore different directions. The best insertion heuristic is then used to construct each population's individual using these weights.

Algorithm 5 : GenerateInitialPopulation.

Data : M : population parameter
Result : pop : a population of initial solutions

```
1 pop ← vector of M + 1 empty solutions ;
2 for i = 0  to  M do
3 |   (w1, wl) ← (i/M, (M − i)/M);
4 |   pop[i] ← bestInsertionHeuristic(w1, wl) ;
5 end
```

4.4.3 Local search procedure and neighborhood structures

Our VNS and VND algorithms rely on the set of the neighborhood structures used and particularly on the sequence of their execution. The neighborhood structures are classified and then applied from the best to the least performing [88]. This work adopts the following sequence of neighborhood structures: the swap, the insert, the 2-opt*, and the 2-opt operators. We use approximately

the same order as the literature for VRPs [105,106]. The results of our previous work [98] and of Chapter 3 show that with well-chosen operators, the GVNS algorithm is an effective method to solve the single objective version of our problem. Its general structure can be applied to improve other methods performance [66]. Preliminary tests were realized using a steepest descent heuristic (best improvement local search) to verify this order. The semi-randomization and the randomization of the operators have been tested in both shaking and VND phases. They worsen the solutions obtained. We have used in Chapter 3, the classical BA strategy for the mono-objective GVNS and in this study, a novel proposed BA strategy, the MOBI/P, for the multi-objective algorithms to obtain a complete Pareto front. In our Pareto-based algorithms, a solution $s^{'}$ is considered better than an other solution s if it dominates it or if it is incomparable to it. In other words, if after applying a move operator $f(s^{'}) \prec f(s)$ or $f(s^{'}) \nprec f(s)$ and $f(s) \nprec f(s^{'})$, the solution $s^{'}$ is considered as a new efficient solution and the set A is updated by means of the $addSolution$ method. The intra-route moves are used on a single route, while inter-route moves intervene on multiple routes. We examine in the neighborhood exploration all possible positions for each operator. The three first neighborhood structures are inter-routes, and the 2-opt is an intra-route operator. The insert and 2-opt* procedure may modify the number of operations in each route as well as their order. Here's a description of the different possible moves:

The swap move exchanges two operations either in the same route or between different routes.

The insert move deletes an operation from its position and inserts it in another position that can be in the same route or in a different route.

The 2-opt* operator generates a neighboring solution by removing arcs $(i, i+1)$ and $(j, j+1)$ belonging to two distinct routes and reconnecting arcs $(i, j+1)$ and $(j, i+1)$. It withdraws two edges from a route and replaces them with two other edges to form new routes. This operator is therefore inter-route.

The 2-opt operator removes arcs $(i, i+1)$ and $(j, j+1)$ from a route and

links arcs (i, j) and $(i+1, j+1)$ in the same route. This operator is the same as a reverse operator that reverses the elements between i and $j+1$. This operator is intra-route since it intervenes on arcs belonging to the same route.

4.4.4 Updating non-dominated solutions set

The addSolution method described in Algorithm 6 is used to update the set of non-dominated solutions in the archive A [90, 97]. This method uses Pareto dominance to evaluate solutions.

Algorithm 6 : addSolution.

Data : A: a set of non-dominated solutions;
s: a starting solution;
Result : A: updated archive;
$added$: a boolean equal to true if s is added to the archive A;

```
1  added ← true;
2  foreach x ∈ A do
3      if f(x) ≤ f(s) then
4          added ← false;
5          break;
6      end
7      if f(s) ≺ f(x) then
8          A ← A − {x};
9      end
10 end
11 if added = true then
12     A ← A ∪ {s};
13 end
```

4.4.5 Multi-Objective Variable Neighborhood Descent Based on Pareto Dominance (MOVND/P)

The multi-objective variable neighborhood descent based on Pareto dominance is an adaptation of the variable neighborhood descent algorithm (VND) to solve our multi-objective problems. It is called MOVND/P. This algorithm has been designed primarily as a multi-objective local search component and an intensification algorithm. Its goal is to improve the performance of other algorithms. This algorithm is based on the general principle of exploring several neighborhoods when there is still an improvement and is inspired by the single objective VND.

Algorithm 7 describes the proposed MOVND/P algorithm. At each iteration and while the archive is still improving (steps 6–28), the current neighborhood of s is entirely explored with a novel multi-objective best improvement strategy called MOBI/P. This $MOBI/P$ procedure tests if each neighbor of s is non-dominated with the best solution found so far in the neighborhood of s. This best solution changes during the search. The MOBI/P strategy is very impactful since it allows the search to be more efficient and diversified and enables the algorithm to converge rapidly. All non-dominated solutions by the best solution found so far in the neighborhood of s are then stored in the set P (step 11).

Each point in P generated with the MOBI/P procedure that is not in the archive A is evaluated to enter this latter or not (steps 13–19). The solution x is included in the archive A if it is non-dominated by s and replaces the current solution s. This replacement of the current solution s by a solution that dominates it or that is incomparable to it is another new feature used. It can be noticed in line 15 and is directly inspired by a single-objective local search.

The counterAISecond measures if some solutions have been added to the archive from the previous iteration to the next iteration. It is incremented in line 17. The aim is to continue running the algorithm while there is still an improvement. The counterAISecond precisely measures if there is an improvement compared to the counter of the previous iteration counterAISecondPrevIt. There is, however, a stopping condition on the number of iterations $iter_{max}$ to avoid large computational time.

We could compare the current archive to the previous one to measure improvement in the latter. However, using the counters here constitutes a speed-up technique compared to using the entire archive for comparison.

We define a new multi-objective neighborhood change procedure. In our case, we say that a neighborhood N_l improves the archive A if all the solutions returned in P are non-dominated by the solutions in A. An improvement means that all new points have been added to the archive A. This is recorded using the counter $counterAI$, which is incremented at each improvement.

A new characteristic was also incorporated in the neighborhood change procedure. In our algorithm, we stay in the improving neighborhood if all the solutions in P are non-dominated by s but also when $counter$ did not reach $iterC_{max}$. The last part of the condition is required to avoid not exploring the other neighborhoods when the first neighborhood is constantly improving. We can obtain an efficient mix between staying in the best neighborhood and exploring the different neighborhoods.

4.4.6 Multi-Objective Variable Neighborhood Descent Based on Pareto Dominance Improved (MOVND/PI)

The multi-objective variable neighborhood descent improved based on Pareto dominance (MOVND/PI) presented in Algorithm 8 is an improvement of our Algorithm 7 presented in Section 4.4.5. The improved algorithm can be used to solve multi-objective problems but is not intended to be a local search component such as the previous algorithm. We allow it, therefore, to consume more computational time through a more sophisticated selection criterion. The new solution s for the following iteration is randomly chosen from the set of non-dominated solutions A among solutions that have not been previously explored in the search. The aim of this criterion is to avoid the intensification of a point already visited and exploited. The algorithm, therefore, does not include the line 26 of the Algorithm 7 to select a solution for the next iteration. The only criteria here to continue running the algorithm is the maximum number of iterations.

Algorithm 7 : Multi-objective VND based on Pareto Dominance (MOVND/P).

Data : l_{max}: number of neighborhood structures;
s_0: initial solution;
Result : A: a set of potentially efficient solutions;

```
1  s ← s_0 ;
2  addSolution(s_0, A) ;
3  P ← ∅ ;
4  iteration ← 1;
5  counterAISecond ← 0;
6  while (counterAISecondPrevIt ≠ counterAISecond ∧ (iteration ≤ iter_max)) do
7      counterAISecondPrevIt ← counterAISecond ;
8      counter ← 1;
9      l ← 1;
10     while l ≤ l_max do
11         P ← MOBI/P(s, l);
12         counterAI ← 0;
13         foreach x ∈ P do
14             if x ∉ A ∧ addSolution(x, A) then
15                 s ← x ;
16                 counterAI ← counterAI + 1;
17                 counterAISecond ← counterAISecond + 1;
18             end
19         end
20         counter ← counter + 1;
21         if (counterAI = size(P)) ∧ (counter ≤ iterC_max) then
22             l ← 1 ;
23         else
24             l ← l + 1 ;
25         end
26     end
27     iteration ← iteration + 1;
28 end
```

4.4.7 Multi-Objective General Variable Neighborhood Search Based on Pareto Dominance (MOGVNS/P)

Algorithm 9 describes the GVNS-based algorithm proposed. At each iteration of MOGVNS/P and while the archive is still improving (steps 7–33), a shaking procedure is applied on the current solution s to obtain $s^{'}$. The shaking procedure

Algorithm 8 : Multi-objective VND based on Pareto Dominance Improved (MOVND/PI).

Data : l_{max}: number of neighborhood structures;
s_0: initial solution;
Result : A: a set of potentially efficient solutions;

```
1  s_0 ← bestInsertionHeuristic(0.5, 0.5) ;
2  s ← s_0 ;
3  addSolution(s_0, A) ;
4  P ← ∅ ;
5  iteration ← 1;
6  while (s ∈ notExploredSolution(A)) ∧ (iteration ≤ iter_max) do
7      counter ← 1;
8      l ← 1;
9      while l ≤ l_max do
10         P ← MOBI/P(s, l);
11         counterAI ← 0;
12         foreach x ∈ P do
13             if x ∉ A ∧ addSolution(x, A) then
14                 counterAI ← counterAI + 1;
15             end
16         end
17         counter ← counter + 1;
18         if (counterAI = size(P)) ∧ (counter ≤ iterC_max) then
19             l ← 1 ;
20         else
21             l ← l + 1 ;
22         end
23     end
24     s ← selectRandomNotExploredSolution(A) ;
25     iteration ← iteration + 1;
26 end
```

selects a random solution $s^{'}$ from the current k^{th} neighborhood of the current solution s ($s^{'} \in N_k(s)$). The aim of this phase is to diversify the search process. This perturbation is applied nS times.

An intensification step is then applied to the perturbed solution $s^{'}$ using the MOVND/P described in Section 4.4.5. The non-dominated solutions obtained by MOVND/P are stored and returned in the set P_{VND} (step 16). All the solutions of the P_{VND} set are compared to s to update the archive A using the function addSo-

lution. The improvement of the archive A while exploring P_{VND} is recorded using $counterAI$ and $counterAISecond$ from one iteration to the other such as in the MOVND/P algorithm.

We stay in the same neighborhood if all solutions in P_{VND} are added to the archive (steps 25–29). The multi-objective neighborhood change procedure is the same as defined in MOVND/P and MOVND/PI.

In this algorithm, and differently from MOVND/P, the non-dominated solutions in the archive A are used to restart the current solution s. Indeed, the new solution s for the next iteration is randomly chosen from the archive A among solutions that have not been already explored. This criterion is similar to the one adopted in Pareto Local Search and [93] and is used to avoid the intensification of a point already exploited. However, it is time-consuming and could be replaced by randomly choosing the new solution s for the next iteration from the archive, even if it has been explored previously.

The step-by-step procedure of MOGVNS/P can be summarized as follows. First, an initial solution is constructed using the best insertion heuristic and is added to the archive of efficient solutions A. The main procedure is then repeated while there is still an improvement of the archive A and while not reaching a maximum number of iterations $iter_{max}$. It is composed of the following steps: the exploration of the neighborhood structures while $k \leq k_{max}$ and a random selection in the archive A for the next iteration of the algorithm. The neighborhood exploration phase is composed of four steps. It starts with a shaking procedure to perturb the current solution, followed by an intensification phase using the MOVND/P algorithm to generate the Pareto front P_{VND}. Next, a test is applied to verify that the points in P_{VND} are non-dominated by s to enter the set of efficient solutions A or not. Finally, the decision to stay in the improving neighborhood or explore other neighborhoods is made. Counters are used throughout the algorithms to define stopping criteria, including counters that record the improvement of the archive A.

Algorithm 9 : Multi-objective GVNS based on Pareto Dominance (MOGVNS/P).

Data : k_{max}: number of neighborhood structures in MOGVNS/P ;
l_{max}: number of neighborhood structures in MOVND/P ;
Result : A : a set of potentially efficient solutions

$s_0 \leftarrow bestInsertionHeuristic(0.5, 0.5)$;
$s \leftarrow s_0$;
$addSolution(s, A)$;
$P_{VND} \leftarrow \emptyset$;
$iteration \leftarrow 1$;
$counterAISecond \leftarrow 0$;
while $(counterAISecondPrevIt \neq counterAISecond \wedge (iteration \leq iter_{max})$ **do**
  $counterAISecondPrevIt \leftarrow counterAISecond$;
  $counter \leftarrow 1$;
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    $s^{'} \leftarrow s$;
    for $p = 1 \quad to \quad nS$ **do**
      $s^{'} \leftarrow$ **Shaking**(s', k);
    end
    $P_{VND} \leftarrow$ **MOVND/P**$(s^{'}, l_{max})$;
    $counterAI \leftarrow 0$;
    foreach $(x \in P_{VND})$ **do**
      if $x \notin A \wedge addSolution(x, A)$ **then**
        $counterAI \leftarrow counterAI + 1$;
        $counterAISecond \leftarrow counterAISecond + 1$;
      end
    end
    $counter \leftarrow counter + 1$;
    if $(counterAI = size(P_{VND})) \wedge (counter \leq iterC_{max})$ **then**
      $k \leftarrow 1$;
    else
      $k \leftarrow k + 1$;
    end
  end
  $s \leftarrow selectRandomNotExploredSolution(A)$;
  $iteration \leftarrow iteration + 1$;
end

4.4.8 Multi-Objective General Variable Neighborhood Search Based on Pareto Dominance and Decomposition (MOGVNS/CDP)

Algorithm 10 extends the MOGVNS/P by starting with a population of initial solutions rather than a single solution. The population is generated using the population generation procedure previously presented in Algorithm 5. The chosen weights in each solution aim to guide the search process in different directions. For each individual in the population, the MOGVNS/P algorithm is used to obtain the non-dominated solutions. The algorithm is therefore based on Pareto dominance. The population of solutions aims to introduce more diversity in the search process by exploring different regions. The drawback of this method is that it can be more time-consuming since we start from a population of solutions.

Algorithm 10 : Multi-objective GVNS based on Pareto Dominance and Decomposition (MOGVNS/CDP).

Data : k_{max}: number of neighborhood structures in MOGVNS/P;
l_{max}: number of neighborhood structures in MOVND/P;
$pop \leftarrow$ vector of $M+1$ empty solutions;
M : population parameter;
Result : A : a set of potentially efficient solutions;

```
1 pop ← GenerateInitialPopulation(M);
2 for i = 0 to M do
3 |   addSolution(pop[i], A) ;
4 end
5 for i = 0 to M do
6 |   s ← pop[i] ;
7 |   MOGVNS/P(s, A, kmax, lmax);
8 end
```

4.4.9 Multi-Objective General Variable Neighborhood Search Based on Decomposition (MOGVNS/D)

Algorithm 11 combines the general structure of the classical aggregation-based method, decomposition algorithms, and GVNS algorithm. It divides the multi-objective problem into $M + 1$ mono-objective problems based on aggregation through different weight vectors. The proposed algorithm here is called MOGVNS/D. It uses the single objective local search VND to solve the scalar subproblems.

The single objective VND applies a weighted sum function to evaluate and compare solutions. Given a solution s, the weighted sum function associated to a solution s and weight vector (w_1, w_l) is calculated as follows: $g(s, w_1, w_l) = w_1 \times f_1(s) + w_l \times f_l(s)$ with l=2,3. The initial population is generated using the procedure described in Algorithm 5. Compared to the previous ones based on Pareto dominance, the main drawback of this method is that the Pareto front returned is an approximation of the supported efficient solutions. It can also be time-consuming since we start from a population of solutions. On the other hand, this algorithm is easily convertible to the single objective GVNS algorithm since it is based on aggregation.

4.4.10 Novelty in the proposed algorithms

Novelty among acceptance strategies in multi-objective pptimization

Several authors have proposed and used several adaptations for the first-accept (FA) and best-accept (BA) in the multi-objective optimization. Therefore, we analyze and review the existing ones and the proposed method hereafter.

First-Accept based on Pareto dominance: This procedure consists of accepting the first solution $s^{'}$ that is non-dominated by s. It has been used by Cota et al. [90] in their multi-objective Adaptive Large Neighborhood Search based on Pareto dominance (MO-ALNS). The local search procedure in MO-ALNS is Randomized Variable Neighborhood Descent (RVND).

Best-Accept based on Pareto dominance PLS: Paquete et al. [69] proposed

Algorithm 11 : Multi-objective General Variable Neighborhood Search based on Decomposition (MOGVNS/D).

Data : k_{max}: number of neighborhood structures in MOGVNS/P;
l_{max}: number of neighborhood structures in MOVND/P;
pop: vector of $M+1$ empty solutions ;
M: population parameter;
Result : A : a set of potentially efficient solutions;

```
1  k ← array of M + 1 integers ;
2  pop ← GenerateInitialPopulation(M);
3  repeat
4      for i = 0 to M do
5          repeat
6              k[i] ← 1 ;
7              while k[i] ≤ k_max do
8                  s' ← pop[i];
9                  for p = 1 to nS do
10                     s' ← Shaking(s', k);
11                 end
12                 s'' ← VND(s', l_max);
13                 if g(s'', w_1, w_l) < g(pop[i], w_1, w_l) then
14                     pop[i] ← s'' ;
15                     k[i] ← 1 ;
16                 else
17                     k[i] ← k[i] + 1;
18                 end
19             end
20         until no improvement is obtained;
21         iteration ← iteration + 1;
22     end
23 until iteration ≤ iter_max;
24 for i = 0 to M do
25     addSolution(pop[i], A) ;
26 end
```

Pareto local search to apply the BA strategy in the multi-objective context. All the neighborhood of s is explored, and the new solution is accepted if it is not dominated by any solution in the set of efficient solutions.

Best-Accept based on Pareto dominance MOBI/P: We use in this chapter a novel best-accept strategy for multi-objective optimization called Multi-objective Best Improvement strategy based on Pareto dominance ($MOBI/P$). This procedure tests if each neighbor of s is non-dominated with the best solution found so far in the neighborhood of s. This best solution changes during the search. If the new solution $s^{'}$ dominates this best solution, it replaces it, and so on until exploring all the neighborhoods and retaining a unique last best solution that is the most converging. Choosing the last best solution permits a translation in the Pareto front, ensuring diversification if the new solution $s^{'}$ is incomparable with the best solution or more convergence if it dominates it. Moreover, it is faster compared to the PLS strategy since we avoid comparing each solution in the neighborhood to each solution in the archive as in PLS. This best solution changes during the search, which permitted us to define the last best solution as the new starting current solution in MOVND/P, the main component of our MOGVNS/P algorithm.

Figure 4.2 illustrates the MOBI/P mechanism. Whenever the new solution dominates the best solution, it replaces it, and the algorithm converges. Here the replacement permits the comparison with the best solution each time and the rapid convergence. On the other hand, when the new solution and the best solution are incomparable, replacing the best solution with the new one permits the translation in the Pareto front and, therefore, diversification.

Best-Accept for each objective separately: This procedure consists of applying a single objective local search but for each objective separately. Duarte et al. [93] used VND-1 (respectively, VND-2) to improve the value of f1 (respectively, f2) regardless of the value of f2 (respectively, f1) in a bi-objective problem. The final local optimum is then tested to enter the archive or not. They then proposed a template for multi-objective VND named MOVND, alternating improvements in each objective until no further improvements are found. This methodology was adopted later by Queiroz et al. [94].

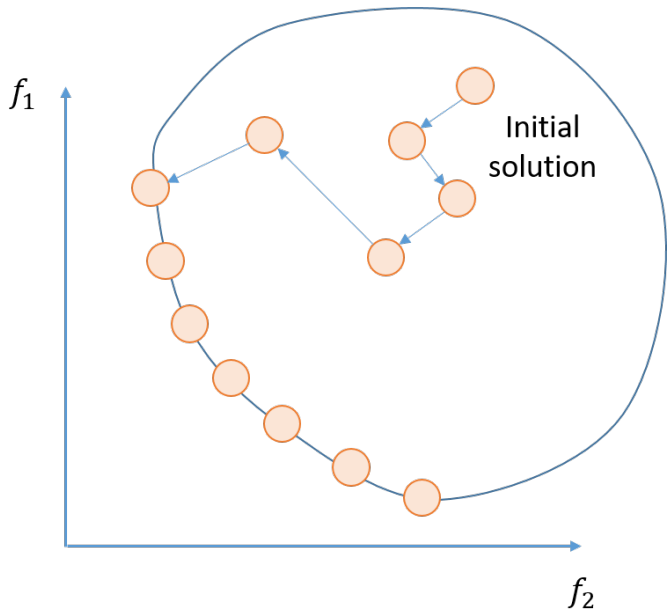

Figure 4.2: Representation of the MOBI/P mechanism

Combination between the FA and BA strategy in PLS: Dubois-Lacoste et al. [70] proposed an alternative to accept only dominating solutions to speed up the search. They suggested switching to the criteria adopted in PLS of accepting non-dominated solutions if such solutions are no longer found. The authors also proposed an alternative strategy. It consists of using the first-improvement technique to converge first to a good approximation of the Pareto front until all solutions in the archive are flagged as visited. Afterward, one can move to the PLS best improvement strategy to complete the archive with the remaining neighbors.

Novelty in the design of the multi-objective algorithms

We consider an improvement if all the solutions explored have been added to the archive. This procedure is different than the definition of improvement proposed by [93], where improving means at least one new point has been added to the archive A.

Likewise, we stay in the improving neighborhood if all the solutions explored are non-dominated by s but also when a defined counter does not reach a maximum number of iterations. This last condition is required to avoid not exploring

the other neighborhoods when the first neighborhood is constantly improving. In the literature, if there is an improvement, the exploration continues in the same neighborhood structure [93]. Queiroz and Mundim [94] propose to change the neighborhood systematically at each iteration in an adapted template from [93] to reduce the computational time.

All the proposed algorithms use the MOBI/P procedure above as a multi-objective best improvement strategy, whereas there are several different FA and BA accept strategies in the literature.

4.5 Computational experiments

This section presents computational experiments to measure the proposed algorithms' performance. We present the results of the multi-objective problem using the proposed algorithms MOVND/P, MOVND/PI, and MOGVNS/P. The results of the other proposed variants, MOGVNS/CDP and MOGVNS/D, are also shown to demonstrate the performance of the three aforementioned algorithms. Heuristic Pareto fronts are also provided for comparison to our multi-objective problem. The maintenance model without constraints was solved using Python 3.6. The joint maintenance scheduling and workforce routing model was then solved using C++. This work has been compiled with GCC 7.4 in a Linux environment of a personal computer. Experiments were conducted using the CALCULCO computing platform in an AMD EPYC 7702 with 2CPU, 2 gigahertz, and 1 core was dedicated to each instance. All methods are run using the same machine to avoid bias.

4.5.1 Instances description

We use two sets of instances. The first one, denoted by Re. The second one is inspired from Solomon's benchmark. The second set is adapted as explained in Chapter 3. Detailed explanations about the instances used and their generation can be found in the instances description Section of Chapter 3.

4.5.2 Parameter setting and performance metrics

There are two stopping conditions for the MOVND/P, MOVND/PI and MOGVNS/P: the maximum number of iterations $iter_{max}$ and the stopping criterion related to the neighborhood change $iterC_{max}$. The value of these parameters has been fixed according to the number of operations n. For the MOVND/P algorithm, the maximum number of iterations is $iter_{max} = max(1, E[n/2])$. The second defined stopping criterion related to the neighborhood change is $iterC_{max} = max(1, E[n/16])$. Similarly, for MOGVNS/P these parameters are $iter_{max} = max(1, E[n/2])$ and $iterC_{max} = max(1, E[n/16])$. For MOVND/PI, these values are $iter_{max} = 2n$ and $iterC_{max} = max(1, E[n/16])$. For the algorithm MOGVNS/D, $iter_{max} = 1$ in the multi-objective case. The algorithm also repeats the instructions until there is no improvement for each individual of the population. For the mono-objective case, as mentioned previously, the stopping criterion is a maximum number of iterations that equals $iter_{max} = max(1, E[n/8])$ for the routing objective and the maintenance objective and $iter_{max} = max(1, E[n/4])$ for the failure objective. MOGVNS/CDP calls MOGVNS/P and has, therefore, the same values as this latter for $iter_{max}$ and $iterC_{max}$. The population size parameter equals $M = 10$.

In mono-objective optimization, it is simple to evaluate the quality of a solution. It is a more difficult task in multi-objective optimization since the output is represented by sets of trade-off solutions, potentially incomparable in terms of Pareto dominance. Consequently, we use several indicators to measure the quality of an approximation of the Pareto front involving several criteria such as solution quality, computational effort, robustness, and other factors. The indicators assess solution quality and computational effort. The CPU indicator evaluates the time needed for the algorithm to return a final Pareto front. The quality indicators that we used are the number of non-dominated solutions and the hypervolume indicator to measure coverage and assess the front returned convergence.

The hypervolume (HV) indicator is a metric that measures the size of the space covered. It assesses the space enclosed by all the solutions of the objective space. The HV of an estimated Pareto front is the sum of the hypercubes that

each set of solutions contains. It shows the distribution of solutions along the Pareto front. The larger the HV indicator is, the better the Pareto front is.

The reference point (f_1^{rp}, f_l^{rp}) is chosen to be dominated by all solutions of the Pareto front. In this section, the reference point rp used is (f_1^{max}, f_l^{max}) since we seek to minimize costs. f_1^{max} and f_l^{max} are, respectively, the maximum values of the first objective (routing cost) and the second objective, either failure or maintenance cost in the Pareto front identified by the mono-objective GVNS. Figure 4.3 illustrates an example of the HV indicator for an instance in a minimization problem and a reference point rp. The size of a rectangular area a_i enclosed by a solution s_i is $hv_i = (f_1^{rp} - f_1(s_i)) * (f_l^{rp} - f_l(s_i))$ where l = 2,3. The hypervolume is the sum of the areas formulated as follows:

$$hv = \sum_{i=1}^{p} (f_1^{max} - f_1(s_i)) * (f_l^{max} - f_l(s_i)), \quad \forall s_i \in \mathcal{A} \tag{4.2}$$

where l = 2,3, and p is the number of solutions s_i in the Pareto front A.

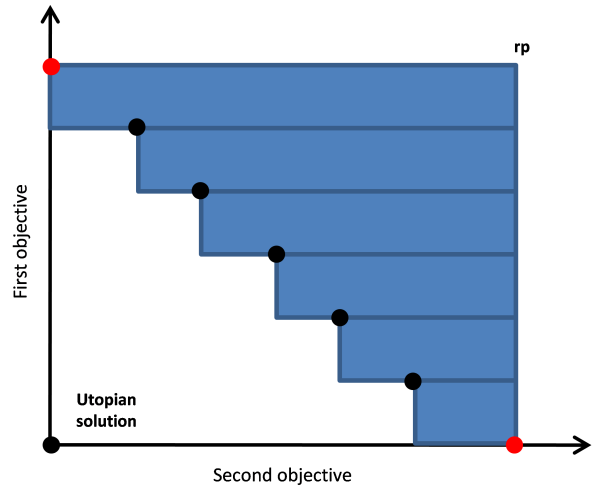

Figure 4.3: Example of the hypervolume indicator

Our bi-objective algorithms based on VND and GVNS frameworks were compared to those of [93]. Algorithms in [93] were run for a longer time to have a complete Pareto front and meaningful results. For a fair comparison, all methods were run on the same machine in the computing platform.

The improvement of our algorithms over the literature algorithm is evaluated using a percent increase of the HV indicator and the number of non-dominated points when using the proposed algorithms PA compared to the comparison algorithms CA of [93].

$$IHV(CA,PA) = (\frac{HV(PA) - HV(CA)}{HV(CA)}) \times 100\% \tag{4.3}$$

$$INDP(CA,PA) = (\frac{NDP(PA) - NDP(CA)}{NDP(CA)}) \times 100\% \tag{4.4}$$

It is also evaluated using the percent decrease of the CPU time of the proposed algorithms PA over the comparison algorithms CA.

$$ICPU(CA,PA) = (\frac{CPU(CA) - CPU(PA)}{CPU(CA)}) \times 100\% \tag{4.5}$$

4.5.3 Computational comparisons

In order to evaluate the performance of our proposed algorithms MOVND/P and MOGVNS/P, we compare their results with those of the state-of-the-art multi-objective algorithms VND and GVNS. The literature algorithms used in our comparison are MOVND and MOGVNS proposed by [93]. They also adapt the single objective VNS framework to deal with multi-objective combinatorial optimization problems. Duarte et al. [93] propose to use a different single objective VND-i to improve each objective separately. In a bi-objective problem, VND-1 and VND-2 are used. VND-1 (respectively VND-2) improves the value of f_1 (respectively f_2) regardless the value of f_2 (respectively f_1). They then check if the final local optimum should be added to the set of non-dominated points or not. The only difference between VND-i and a single objective VND is when they test whether the local optimum obtained qualifies to enter the archive E or not, and the final step is when the archive is returned. They then propose a template for multi-objective VND named MOVND, alternating improvements in each objective until no further improvements are found. It starts with a random non-exploited point in E and explores its neighborhoods again with the VND-i procedure. An improvement is considered when at least one new

point has been added to the set of non-dominated points E. Duarte et al. [93] propose a MOGVNS that starts with an initial set E of non-dominated points. It applies a multi-objective shaking, followed by the MOVND described above, and finally, a neighborhood change procedure that consists of moving to the next neighborhood only if the same neighborhood is not improving anymore.

In our comparison, we seek to compare the [93] templates with our templates and not the specific choices related to the problem. MOVND/P and MOGVNS/P start with one constructed initial solution. We use the same initial solution in the comparison algorithms of Duarte et al. [93]. The same neighborhood structures are chosen, and the same order, effective for this particular problem, is also applied in the algorithms of [93]. The best improvement strategy to explore neighborhoods is chosen for all templates. This way allows us to compare only the frameworks of the algorithms and not other considerations related to the problem. In order to have meaningful results, algorithms in [93] were run for a longer time to have a complete Pareto front.

Even if our algorithms and the literature algorithms are adaptations from the single objective well-known VND and GVNS, they are entirely different. Our MOVND/P and MOVND of the literature [93] are different. The MOBI/P function uses a proposed multi-objective best improvement local search strategy that is impactful in improving solutions' quality and reducing computational time. The VND used by [93] tries first to improve each objective independently from the other using a single objective best improvement evaluation. The comparison algorithms are not entirely based on Pareto dominance since they include a single objective evaluation. Our MOVND/P replaces the current solution with the best solution found so far in the neighborhoods with multi-objective evaluation. At the same time, in [93], there is a random selection among the non-exploited points. The neighborhood change procedure in MOVND of the literature uses the single objective VND, while we use a specific multi-objective neighborhood change procedure in MOVND/P. The stopping criteria used are also different and related to the improvement obtained. MOGVNS proposed by [93] uses a shaking procedure, the previous MOVND, and a multi-objective neighborhood change procedure adapted from the mono-objective neighborhood change procedure. Duarte et al. [93] stay in the same neighborhood when there

is still an improvement, while in our MOVND/P and MOGVNS/P, we force the exploration of all the neighborhoods since we stay in the best neighborhood only for a certain number of iteration. Our algorithms, most of the time, explore neighborhood structures sequentially. In our MOGVNS/P, the new solution for the next iteration is randomly chosen from the archive among solutions that have not already been explored. An improvement is considered in [93] when at least one new point has been added to the set of non-dominated points. In contrast, in our algorithms, the improvement implies that all the points returned after exploring the neighborhood or in the intensification step are added to the archive.

4.5.4 Test results

Tests are realized over small, medium, and large instances. Small instances include 6 and 12 PM operations. The number of operations varies between 23 and 34 for medium-sized instances and between 52 and 73 for large-sized instances. The indicators assessed are the hypervolume (HV) to measure the convergence and coverage of the solutions in the Pareto front obtained, the number of non-dominated points (NDP), and the CPU time in seconds.

Comparison with the literature

Tables 4.2 and 4.3 show the results of the proposed variable neighborhood descent algorithms, MOVND/P, MOVND/PI, and the results of the MOVND suggested by [93]. Furthermore, Table 4.2 details the results for the maintenance cost as a second objective, whereas Table 4.3 shows the results when the failure cost is the second objective of the problem.

Some results that we display in detail in Table 4.5 are also reported here for the tested instances: the average percent improvement of HV (IHV), NDP (INDP), and CPU (ICPU). They are summarized in Table 4.1. There is an improvement for bold numbers. When the maintenance cost is the second objective considered, both MOVND/P and MOVND/PI have better average values than the MOVND of [93] on all the considered quality indicators. Indeed, the average percent improvement of MOVND/P compared to MOVND [93] of the

hypervolume (IHV) and the number of non-dominated points (INDP) is, respectively, 7.82% and 2.79%. These two indicators improved slightly; however, the CPU time decreased considerably by 81.56%. The second proposed algorithm, MOVND/PI, considerably outperforms the MOVND of the literature [93] for the 46 instances tested. Indeed, the hypervolume and the number of non-dominated points increased, respectively, by an average of $IHV = 104.12\%$ and $INDP = 108.45\%$, which is a considerable improvement. However, the CPU decreased by only 4.73%. Despite the minor improvement for the CPU time, the other coverage and convergence indicators are considerably improved. We also measure this improvement for the proposed VND algorithms over MOVND of the literature [93] for the instances without the four small instances Re. For the 42 derived from Solomon's instances, the improvement of the indicators HV, NDP, and CPU of MOVND/P is, respectively, $IHV = 2.32\%$, $INDP = -3.26\%$, and $ICPU = 88.81\%$. The improvement of the same indicators for MOVND/PI becomes $IHV = 106.38\%$, $INDP = 105.20\%$, and $ICPU = 28.87\%$.

Improvement	The maintenance objective			The failure objective		
	IHV(%)	INDP(%)	ICPU(%)	IHV(%)	INDP(%)	ICPU(%)
MOVND/P over MOVND of [93]	**7.82**	**2.79**	**81.56**	**85.71**	**67.74**	**80.79**
MOVND/PI over MOVND of [93]	**104.12**	**108.45**	**4.73**	**303.78**	**304.90**	−10.21
MOGVNS/P over MOGVNS of [93]	**52.29**	**52.15**	**41.69**	**91.47**	**92.11**	**47.74**
MOGVNS/P over MOGVNS/CDP	**2.42**	**1.92**	**24.73**	−2.91	−1.68	**18.24**
MOGVNS/P over MOGVNS/D	**574.41**	**573.14**	**19.92**	**792.71**	**679.11**	**13.03**
MOGVNS/P over MOVND/PI	**25.87**	**26.90**	−706.20	**22.02**	**21.75**	−387.50
MOVND/PI over MOGVNS/P	−0.38	−0.23	**65.74**	−7.55	−7.41	**70.87**

Table 4.1: Comparison between the proposed algorithms and the literature algorithms

We consider the failure cost as a second objective with the routing one. The average percent improvement of the indicators of MOVND/P in comparison to MOVND of the literature [93] is considerable, and the values of HV, NDP, and CPU indicators equal, respectively, 85.71%, 67.74%, and 80.79%. The average percent improvement of HV for MOVND/PI compared to MOVND of the literature is 303.78%. The number of non-dominated points NDP increased by 304.90%. However, the improvement of the CPU is −10.21%. When the four small instances Re are deleted to consider only instances generated from Solomon's set, the improvement of the indicators stays considerable for both

algorithms. The values of HV, NDP, and CPU indicators improved by, respectively, 67.16%, 67.11%, and 88,81% for MOVND/P and by 324.52%, 325.54%, and 14.01% for MOVND/PI. The quality indicators are improved, and the CPU time of our algorithms is considerably less.

We can conclude that all the indicators were improved for both failure cost and maintenance cost as the second objective. There is a slight improvement for the HV and NDP indicator for MOVND/P for the maintenance cost and a considerable improvement in the CPU time that has decreased. On the other hand, MOVND/PI improves the HV and NDP indicators considerably with only a tiny improvement in the CPU time. The results of the failure cost as a second objective demonstrated a considerable improvement for all the quality and time indicators for both MOVND/P and MOVND/PI algorithms.

Table 4.4 illustrates the results of the comparison between our proposed MOGVNS/P and MOGVNS of the literature suggested by [93]. When the maintenance cost is the second objective, the average percent improvement of HV when using MOVGVNS/P is 52.29%. The number of non-dominated points NDP increased by an average of 52.15%. The computational time CPU of MOVGVNS/P decreased by $ICPU = 41.69\%$. When the failure cost is the second objective, the average percent improvement of HV of MOVGVNS compared to MOGVNS of the literature is 91.47%. The number of non-dominated points NDP increased by 92.11%. The running time is reduced by 47.74%. All the indicators have been improved. We can conclude that the MOGVNS/P algorithm considerably outperforms the literature algorithm in all the assessed indicators for both second objectives. We fixed a time limit to one week to have all the instances. We have therefore tested the MOGVNS of the literature for fewer instances since it consumes more time.

Comparison between the proposed algorithms

Tables 4.6 and 4.7 present the results of the three indicators measuring the quality of the solutions obtained and the time to get those solutions for the proposed MOGVNS/P, MOGVNS/CDP, and MOGVNS/D algorithms. Tables 4.8 and 4.9 aim to compare the performance of our MOGVNS/P with our three

Instance	*n*	MOVND/P			MOVND/PI			MOVND literature		
		HV($\times 10^{10}$)	NDP	CPU(s)	HV ($\times 10^{10}$)	NDP	CPU(s)	HV($\times 10^{10}$)	NDP	CPU(s)
RE/6/2/80/80/0.30/2	6	0.0003	5	0.05	0.0003	5	0.2	0.0003	4	0.04
RE/6/4/101/101/0.30/2	12	0.0485	8	1.22	0.0666	11	4.47	0.0843	14	5.36
RE/6/2/80/80/0.07/2	6	0.0007	4	0.06	0.0007	7	0.21	0.0002	1	0.03
RE/6/4/101/101/0.07/2	12	0.0479	5	1.22	0.0670	4	4.48	0.0572	6	4.01
*C*101/10/8/100/200/0.07/2	23	1.24	4	19.47	2.49	8	133.15	3.11	10	116.31
*C*101/10/5/100/200/0.07/2	23	1.55	5	23.72	0.31	1	5.87	0.93	3	86.28
*C*101/10/7/100/200/0.07	26	12.97	9	23.83	12.97	9	209.43	5.77	4	130.84
*C*101/25/18/100/200/0.07	54	562.34	16	533.78	1581.59	45	6428.67	667.78	19	8932.8
*C*201/10/7/100/200/0.07	24	4.10	4	15.71	5.77	4	142.89	3.07	3	40.64
*C*201/25/18/100/200/0.07	52	386.17	11	609.31	1097.39	32	5450.49	702.11	20	9680.37
*R*101/10/7/100/200/0.07	34	6.39	2	55.92	3.19	1	26.76	6.40	2	281.41
*R*101/25/18/100/200/0.07	68	951.51	12	1091.67	2140.90	27	17624.5	475.75	6	21268.8
*R*201/10/7/100/200/0.07	34	17.17	2	91.98	6.17	2	26.74	9.27	3	417.44
*R*201/25/18/100/200/0.07	65	897.63	14	1242.88	1987.66	31	14485.9	769.37	12	19628.2
*RC*101/10/7/100/200/0.07	33	17.17	3	55.66	7.22	2	24.26	7.24	2	276.07
*RC*101/25/18/100/200/0.07	62	1011.35	15	677.4	2292.50	34	10612.2	1348.52	20	13304
*RC*201/10/7/100/200/0.07	33	6.59	2	68.92	6.57	2	24.24	3.30	1	121.77
*RC*201/10/7/100/200/0.07	60	1227.72	21	1032.7	2104.66	36	8961.95	876.78	15	9027.58
*C*101/10/8/100/200/0.30/2	23	5.41	19	22.12	9.97	35	132.91	4.56	16	147.47
*C*101/10/5/100/200/0.30/2	23	1.42	5	20.67	5.41	19	126.2	2.85	10	99.42
*C*101/10/7/100/200/0.30	26	29.05	17	35.08	42.94	31	209.26	18.01	13	150.65
*C*101/25/18/100/200/0.30	54	548.34	16	590.27	2296.17	67	6034.47	274.17	8	8805.36
*C*201/10/7/100/200/0.30	24	3.93	4	9.73	19.66	20	154	12.78	13	92.96
*C*201/25/18/100/200/0.30	52	445.82	13	357.9	3223.72	94	5141.7	1097.37	32	17867.8
*R*101/10/7/100/200/0.30	34	15.71	5	61.06	6.28	2	26.77	18.85	6	340.59
*R*101/25/18/100/200/0.30	68	938.94	12	743.79	4616.59	59	18565.4	2660.35	34	54171.9
*R*201/10/7/100/200/0.30	34	8.99	3	52.73	35.96	12	564.07	29.97	10	425.77
*R*201/25/18/100/200/0.30	65	1008.25	16	904.88	3150.85	50	15022.9	441.12	7	32633.30
*RC*101/10/7/100/200/0.30	33	10.66	3	96.33	3.55	1	24.27	21.32	6	454.34
*RC*101/25/18/100/200/0.30	62	863.89	13	609.68	3322.72	50	11081.2	1727.76	26	22870.20
*RC*201/10/7/100/200/0.30	33	25.77	8	73.56	22.55	7	502.72	22.55	7	429.36
*RC*201/25/18/100/200/0.30	60	345.92	6	464.33	2882.67	50	9104.31	1787.19	31	20760.4
*C*101/10/18/200/400/0.07/2	52	294.75	17	1242.3	346.77	20	11082	294.75	17	18586.1
*C*101/10/18/200/400/0.07	58	464.93	13	1868.44	434.22	12	14898.9	289.48	8	20384.3
*C*201/10/18/200/400/0.07	54	158.07	6	1092.76	447.88	17	11487.4	342.48	13	23556.1
*R*101/10/18/200/400/0.07	73	690.09	10	1395.12	3795.37	55	49525.1	1587.10	23	64858.6
*R*201/10/18/200/400/0.07	72	871.90	14	1719.02	2366.54	38	47512.1	1556.88	25	86544.6
*RC*101/10/18/200/400/0.07	71	860.73	11	1957.61	2112.70	27	43173.4	1095.45	14	39987.6
*RC*201/10/18/200/400/0.07	72	604.69	8	3235.91	982.63	13	45042.6	755.85	10	86450.6
*C*101/10/18/200/400/0.30/2	52	260.40	32	2103.68	398.74	49	10837.7	195.29	24	34088.3
*C*101/10/18/200/400/0.30	58	569.45	16	1129.3	1530.41	43	15580.8	284.73	8	19984.4
*C*201/10/18/200/400/0.30	54	234.59	9	616.88	1172.96	45	12336.6	781.95	30	23290.3
*R*101/10/18/200/400/0.30	73	1570.12	23	2435.37	3618.16	53	49447	2457.47	36	149979
*R*201/10/18/200/400/0.30	72	430.94	7	1366.78	4063.01	66	46973.4	1785.15	29	150895
*RC*101/10/18/200/400/0.30	71	1547.50	20	1715.48	5493.56	71	44989.6	1934.23	25	80095.2
*RC*101/10/18/200/400/0.30	72	820.64	11	2227.89	2909.57	39	46574.3	2685.58	36	130648

Table 4.2: Results of MOVND/P, MOVND/PI, and MOVND literature for the maintenance cost as a second objective

Instance	*n*	MOVND/P			MOVND/PI			MOVND literature		
		HV($\times 10^{10}$)	NDP	CPU(s)	HV ($\times 10^{10}$)	NDP	CPU(s)	HV($\times 10^{10}$)	NDP	CPU(s)
RE/6/2/80/80/0.30/2	6	0.0034	5	0.06	0.0004	5	0.21	0.0004	4	0.04
RE/6/4/101/101/0.30/2	12	0.0909	8	1.22	0.1117	10	4.45	0.1008	9	4.05
RE/6/2/80/80/0.07/2	6	0.0015	4	0.06	0.0015	4	0.21	0.0004	1	0.03
RE/6/4/101/101/0.07/2	12	0.0671	5	1.22	0.1174	7	4.48	0.1001	6	3.64
*C*101/10/8/100/200/0.07/2	23	5.31	9	15.1	5.31	9	135.19	5.30	9	64.05
*C*101/10/5/100/200/0.07/2	23	1.76	3	14	6.45	11	128.08	3.52	6	43.86
*C*101/10/7/100/200/0.07	26	18.79	7	31.58	18.79	7	213.73	10.74	4	103.54
*C*101/25/18/100/200/0.07	54	1231.33	18	559.09	3899.40	57	6678.62	1983.79	29	8206.33
*C*201/10/7/100/200/0.07	24	5.77	3	16.77	5.77	3	13.77	5.77	3	37.4
*C*201/25/18/100/200/0.07	52	865.32	13	551.99	2196.74	33	5780.61	532.46	8	10127.5
*R*101/10/7/100/200/0.07	34	42.19	8	86.95	116.43	22	657.07	52.88	10	1166.9
*R*101/25/18/100/200/0.07	68	4066.99	29	2002.37	21038.07	150	18851.9	3786.48	27	33391.7
*R*201/10/7/100/200/0.07	34	63.14	12	94.28	52.66	10	647.3	21.05	4	452.07
*R*201/25/18/100/200/0.07	65	4361.90	38	1034.57	15726.16	137	15669.4	918.13	8	33337.7
*RC*101/10/7/100/200/0.07	33	55.46	9	127.03	67.90	11	565.16	43.13	7	881.1
*RC*101/25/18/100/200/0.07	62	3657.44	30	707.27	9510.84	78	11472.4	4023.34	33	15830.2
*RC*201/10/7/100/200/0.07	33	32.66	6	59.1	10.86	2	65.87	27.21	5	453.44
*RC*201/25/18/100/200/0.07	60	1566.20	15	685.66	3445.64	33	9295.29	1252.91	12	18024.7
*C*101/10/8/100/200/0.30/2	23	13.77	25	29.48	18.73	34	132.67	8.81	16	86.21
*C*101/10/5/100/200/0.30/2	23	5.49	10	16.25	10.44	19	125.59	8.22	15	116.21
*C*101/10/7/100/200/0.30	26	7.82	3	31.54	49.50	19	210.47	23.42	9	155.62
*C*101/25/18/100/200/0.30	54	806.57	12	384.27	4571.50	68	6545.2	537.50	8	10506
*C*201/10/7/100/200/0.30	24	24.09	13	32.81	27.79	15	154.49	12.95	7	160.28
*C*201/25/18/100/200/0.30	52	1308.52	20	641.11	5038.12	77	5401.41	261.65	4	5892.46
*R*101/10/7/100/200/0.30	34	72.90	14	147.27	156.51	30	642.17	104.19	20	906.05
*R*101/25/18/100/200/0.30	68	2914.36	21	1139.92	9306.64	67	19275.6	6387.17	46	55910.8
*R*201/10/7/100/200/0.30	34	55.75	10	77.69	117.96	23	658.76	77.96	14	911.04
*R*201/25/18/100/200/0.30	65	2934.54	25	880.81	12567.23	111	15819.6	1173.17	10	26295.3
*RC*101/10/7/100/200/0.30	33	54.65	9	95.43	158.08	26	579.12	30.29	5	445.19
*RC*101/25/18/100/200/0.30	62	3137.26	26	976.83	9533.74	79	10722.8	3378.24	28	25547.6
*RC*101/10/7/100/200/0.07	33	32.19	6	114.38	32.13	6	543.16	16.07	3	348.52
*RC*101/25/18/100/200/0.07	60	1654.22	16	917.79	5377.30	52	8604.74	929.90	9	17975.6
*C*101/10/18/200/400/0.07/2	52	247.51	10	519.16	544.52	22	11468.7	321.76	13	9274.35
*C*101/10/18/200/400/0.07	58	1011.13	20	1644.54	1769.47	35	16796.7	859.45	17	22559.4
*C*201/10/18/200/400/0.07	54	300.01	8	489.9	450.04	12	11983.4	375.00	10	13677
*R*101/10/18/200/400/0.07	73	5556.85	61	2423.77	24150.22	265	48796.8	4281.01	47	187668
*R*201/10/18/200/400/0.07	72	4314.09	51	2484.2	20134.71	238	45927.8	5412.29	64	154388
*RC*101/10/18/200/400/0.07	71	6007.16	57	1913.23	14228.34	135	44609.6	2107.10	20	28488
*RC*201/10/18/200/400/0.07	72	1199.78	12	1645.21	1199.79	12	45741	899.84	9	42006.1
*C*101/10/18/200/400/0.30/2	52	350.15	30	1882.84	735.40	63	11110	46.67	4	13236
*C*101/10/18/200/400/0.30	58	548.26	11	876.11	3489.58	70	16929.4	1246.10	25	24077.3
*C*201/10/18/200/400/0.30	54	409.33	11	630.19	1228.07	33	12909.9	669.76	18	20636.9
*R*101/10/18/200/400/0.30	73	4336.84	48	1873.73	13106.60	145	48922.1	4876.02	54	230071
*R*201/10/18/200/400/0.30	72	6785.69	81	3001.99	13665.38	163	46528.3	921.47	11	209551
*RC*101/10/18/200/400/0.30	71	5012.31	48	2224.24	12745.50	122	44825	3446.74	33	103688
*RC*201/10/18/200/400/0.30	72	1089.28	11	1673.89	5447.25	55	47334.7	3169.18	32	95318.4

Table 4.3: Results of MOVND/P, MOVND/PI, and MOVND literature for the failure cost as a second objective

Instance	*n*	The maintenance objective						The failure objective					
		MOGVNS literature			Improvement of MOGVNS/P over MOGVNS literature (%)			MOGVNS literature			Improvement of MOGVNS/P over MOGVNS literature (%)		
		HV($\times 10^{10}$)	NDP	CPU(s)	IHV	INDP	ICPU	HV($\times 10^{10}$)	NDP	CPU(s)	IHV	INDP	ICPU
*C*101/10/8/100/200/0.07/2	23	2.49	8	292.92	**62.52**	**62.50**	−93.80	7.66	13	291.07	−30.76	−30.77	**25.46**
*C*101/10/5/100/200/0.07/2	23	0.93	3	158.81	−33.08	−33.33	−35.97	3.52	6	146.89	**100.04**	**100.00**	−87.10
*C*101/10/7/100/200/0.07	26	10.09	7	638.24	**42.86**	**42.86**	**45.60**	18.79	7	762.43	−14.29	−14.29	**53.43**
*C*101/25/18/100/200/0.07	54	843.52	24	253456	**108.33**	**108.33**	**85.62**	2736.29	40	328945	**62.51**	**62.50**	**89.62**
*C*201/10/7/100/200/0.07	24	3.07	3	438.08	**33.34**	**33.33**	**68.18**	5.77	3	270.76	**33.33**	**33.33**	−24.74
*C*201/25/18/100/200/0.07	52	912.74	26	288684	**7.70**	**7.69**	**94.81**	1464.39	22	285426	**145.47**	**145.45**	**89.27**
*R*101/10/7/100/200/0.07	34	3.20	1	1294.75	**199.70**	**200.00**	**59.75**	95.23	18	8911.06	**72.29**	72.22	**69.82**
*R*201/10/7/100/200/0.07	34	6.18	2	2827.07	**0.09**	**0.00**	**68.17**	63.15	12	8289.31	**150.18**	**150.00**	**62.83**
*RC*101/10/7/100/200/0.07	33	7.24	2	1846.61	**0.09**	**0.00**	**71.40**	61.75	10	8152.01	**90.10**	**90.00**	**66.10**
*RC*101/25/18/100/200/0.07	62	1618.23	24	521083	**20.83**	**20.83**	**94.71**	5242.74	43	682100	**79.09**	**79.07**	**90.13**
*RC*201/10/7/100/200/0.07	33	3.30	1	1174.53	**200.25**	**200.00**	**59.59**	21.78	4	5799.4	**0.04**	**0.00**	**83.40**
*RC*201/25/18/100/200/0.07	60	1520.17	26	501547	**215.39**	**215.38**	**89.74**	2928.86	24	484991	**14.10**	**33.33**	**95.00**
*C*101/10/8/100/200/0.30/2	23	4.84	18	401.75	**147.06**	**133.33**	−156.04	7.15	13	418.21	**277.19**	**276.92**	−99.83
*C*101/10/5/100/200/0.30/2	23	5.13	18	365.8	**33.34**	**33.33**	−145.15	7.68	14	350.3	**43.09**	**42.86**	−5.53
*C*101/10/7/100/200/0.30	26	24.93	18	856.62	**55.56**	**55.56**	−78.38	23.42	9	1009.15	**355.96**	**355.56**	−75.78
*C*101/25/18/100/200/0.30	54	1268.02	37	401784	**86.49**	**86.49**	**90.30**	2621.15	39	351411	**97.49**	**97.44**	**90.52**
*C*201/10/7/100/200/0.30	24	16.71	17	712.62	**52.94**	**52.94**	−15.28	16.66	9	790.65	**122.38**	**122.22**	−17.34
*C*201/25/18/100/200/0.30	52	1474.66	43	351203	**42.93**	**53.49**	**95.26**	2355.18	36	317681	**75.02**	**75.00**	**92.68**
*R*101/10/7/100/200/0.30	34	15.71	5	5142.73	**80.02**	**80.00**	**63.00**	78.19	15	13043.5	**260.37**	**260.00**	**59.06**
*R*201/10/7/100/200/0.30	34	23.97	8	7247.38	**25.00**	**25.00**	**78.59**	66.84	12	10749.3	**175.16**	**175.00**	**79.46**
*RC*101/10/7/100/200/0.30	33	17.77	5	5546.07	−20.03	−20.00	**88.04**	97.18	16	7944.16	**25.13**	**25.00**	**74.25**
*RC*201/10/7/100/200/0.30	33	22.55	7	5212.09	**57.14**	**57.14**	**79.06**	37.51	7	5000.94	**57.41**	**57.14**	**63.73**
*C*101/10/18/200/400/0.07/2	52	329.43	19	351203	**10.53**	**10.53**	**91.65**	396.01	16	336779	**93.75**	**93.75**	**90.98**
*C*101/10/18/200/400/0.07	58	361.85	10	891106	−50.00	−50.00	**97.02**	1213.34	24	801339	**62.50**	**62.50**	**93.07**
*C*201/10/18/200/400/0.07	54	342.49	13	466326	−23.08	−23.08	**93.54**	712.53	19	567144	**10.53**	**10.53**	**91.68**
*C*201/10/18/200/400/0.30	54	729.82	28	701594	**3.58**	**3.57**	**94.44**	1116.37	30	833222	**20.01**	**20.00**	**91.00**

Table 4.4: Results of MOGVNS literature and improvement of MOGVNS/P over the MOGVNS of the literature [93]

others proposed algorithms: MOGVNS/CDP, MOVND/PI and MOGVNS/D. The comparison is based on the three quality indicators INDP, ICPU, and IHV. The last study aims to compare the performance of the algorithms presented and discuss their difference. When the maintenance cost is the second objective, and for the 46 tested instances, MOGVNS/P outperforms in average MOGVNS/CDP in the three indicators HV, number of NDP, and CPU with, respectively, 2.42%, 1.92%, and 24.73%. MOGVNS/P is better on average than the decomposition-based algorithm MOGVNS/D in the indicators HV, the number of NDP, and CPU with, respectively, 574.41%, 573.14%, and 19.92%. The MOGVNS/P improves MOVND/PI on the indicators HV and NDP by about 25.87% and 26.90%. However, MOGVNS/P consumes more time (−706.20%) than MOVND/PI. In the opposite direction, the MOVND/PI improves MOGVNS/P by −0.38%, −0.23%, and 65.74%. Therefore, we can conclude that MOVND/PI has similar performance for the HV and NDP indicators and is far faster than MOGVNS/P. When the failure cost is the second objective and for the 46 instances tested, MOGVNS/P outperforms on average MOGVNS/CDP in the three indicators HV, the number of NDP, and CPU by, respectively, −2.91%, −1.68%, and 18.24%. The

Instance	*n*	The maintenance objective						The failure objective					
		Improvement of MOVND/P over MOVND literature (%)			Improvement of MOVND/PI over MOVND literature (%)			Improvement of MOVND/P over MOVND literature (%)			Improvement of MOVND/PI over MOVND literature (%)		
		IHV	INDP	ICPU	IHV	INDP	ICPU	IHV	INDP	ICPU	IHV	INDP	ICPU
RE/6/2/80/80/0.30/2	6	23.69	25.00	-25.00	26.17	25.00	-400.00	867.65	25.00	-50.00	16.49	25.00	-425.00
RE/6/4/101/101/0.30/2	12	-42.45	-42.86	77.24	-20.95	-21.43	16.60	-9.77	-11.11	69.88	10.88	11.11	-9.88
RE/6/2/80/80/0.07/2	6	297.26	300.00	-100.00	299.57	600.00	-600.00	297.11	300.00	-100.00	299.38	300.00	-600.00
RE/6/4/101/101/0.07/2	12	-16.27	-16.67	69.58	17.08	-33.33	-11.72	-33.00	-16.67	66.48	17.22	16.67	-23.08
*C*101/10/8/100/200/0.07/2	23	-59.99	-60.00	83.26	-19.99	-20.00	-14.48	0.01	0.00	76.42	0.03	0.00	-111.07
*C*101/10/5/100/200/0.07/2	23	66.61	66.67	72.51	-66.94	-66.67	93.20	-50.00	-50.00	68.08	83.36	83.33	-192.02
*C*101/10/7/100/200/0.07	26	125.00	125.00	81.79	125.00	125.00	-60.07	75.00	75.00	69.50	75.01	75.00	-106.42
*C*101/25/18/100/200/0.07	54	-15.79	-15.79	94.02	136.84	136.84	28.03	-37.93	-37.93	93.19	96.56	96.55	18.62
*C*201/10/7/100/200/0.07	24	33.33	33.33	61.34	87.60	33.33	-251.60	0.00	0.00	55.16	-0.01	0.00	63.18
*C*201/25/18/100/200/0.07	52	-45.00	-45.00	93.71	56.30	60.00	43.70	62.51	62.50	94.55	312.56	312.50	42.92
*R*101/10/7/100/200/0.07	34	-0.04	0.00	80.13	-50.14	-50.00	90.49	-20.21	-20.00	92.55	120.18	120.00	43.69
*R*101/25/18/100/200/0.07	68	100.00	100.00	94.87	350.00	350.00	17.13	7.41	7.41	94.00	455.61	455.56	43.54
*R*201/10/7/100/200/0.07	34	85.15	-33.33	77.97	-33.43	-33.33	93.59	199.96	200.00	79.14	150.17	150.00	-43.19
*R*201/25/18/100/200/0.07	65	16.67	16.67	93.67	158.35	158.33	26.20	375.09	375.00	96.90	1612.85	1612.50	53.00
*RC*101/10/7/100/200/0.07	33	137.22	50.00	79.84	-0.32	0.00	91.21	28.58	28.57	85.58	57.41	57.14	35.86
*RC*101/25/18/100/200/0.07	62	-25.00	-25.00	94.91	70.00	70.00	20.23	-9.09	-9.09	95.53	136.39	136.36	27.53
*RC*201/10/7/100/200/0.07	33	99.99	100.00	43.40	99.45	100.00	80.09	20.01	20.00	86.97	-60.08	-60.00	85.47
*RC*201/25/18/100/200/0.07	60	40.03	40.0	88.56	140.05	140.00	0.73	25.01	25.00	96.20	175.01	175.00	48.43
*C*101/10/8/100/200/0.30/2	23	18.75	18.75	85.00	118.76	118.75	9.87	56.25	56.25	65.80	112.63	112.50	-53.89
*C*101/10/5/100/200/0.30/2	23	-50.00	-50.00	79.21	90.00	90.00	-26.94	-33.15	-33.33	86.02	27.08	26.67	-8.07
*C*101/10/7/100/200/0.30	26	61.35	30.77	76.71	138.46	138.46	-38.90	-66.63	-66.67	79.73	111.35	111.11	-35.25
*C*101/25/18/100/200/0.30	54	100.00	100.00	93.30	737.50	737.50	31.47	50.06	50.00	96.34	750.51	750.00	37.70
*C*201/10/7/100/200/0.30	24	-69.23	-69.23	89.53	53.85	53.85	-65.66	86.02	85.71	79.53	114.61	114.29	3.61
*C*201/25/18/100/200/0.30	52	-59.37	-59.38	98.00	193.77	193.75	71.22	400.11	400.00	89.12	1825.55	1825.00	8.33
*R*101/10/7/100/200/0.30	34	-16.67	-16.67	82.07	-66.69	-66.67	92.14	-30.03	-30.00	83.75	50.21	50.00	29.12
*R*101/25/18/100/200/0.30	68	-64.71	-64.71	98.63	73.53	73.53	65.73	-54.37	-54.35	97.96	45.71	45.65	65.52
*R*201/10/7/100/200/0.30	34	-70.00	-70.00	87.62	20.00	20.00	-32.48	-28.49	-28.57	91.47	51.32	64.29	27.69
*R*201/25/18/100/200/0.30	65	128.57	128.57	97.23	614.29	614.29	53.96	150.14	150.00	96.65	971.22	1010.00	39.84
*RC*101/10/7/100/200/0.30	33	-50.00	-50.00	78.80	-83.35	-83.33	94.66	80.42	80.00	78.56	421.87	420.00	-30.08
*RC*101/25/18/100/200/0.30	62	-50.00	-50.00	97.33	92.31	92.31	51.55	-7.13	-7.14	96.18	182.21	182.14	58.03
*RC*201/10/7/100/200/0.30	33	14.29	14.29	82.87	0.00	0.00	-17.09	100.31	100.00	67.18	99.98	100.00	-55.85
*RC*201/25/18/100/200/0.30	60	-80.64	-80.65	97.76	61.30	61.29	56.15	77.89	77.78	94.89	478.27	477.78	52.13
*C*101/10/18/200/400/0.07/2	52	0.00	0.00	93.32	17.65	17.65	40.37	-23.08	-23.08	94.40	69.23	69.23	-23.66
*C*101/10/18/200/400/0.07	58	60.61	62.50	90.83	50.00	50.00	26.91	17.65	17.65	92.71	105.89	105.88	25.54
*C*201/10/18/200/400/0.07	54	-53.85	-53.85	95.36	30.78	30.77	51.23	-20.00	-20.00	96.42	20.01	20.00	12.38
*R*101/10/18/200/400/0.07	73	-56.52	-56.52	97.85	139.14	139.13	23.64	29.80	29.79	98.71	464.12	463.83	74.00
*R*201/10/18/200/400/0.07	72	-44.00	-44.00	98.01	52.01	52.00	45.10	-20.29	-20.31	98.39	272.02	271.88	70.25
*RC*101/10/18/200/400/0.07	71	-21.43	-21.43	95.10	92.86	92.86	-7.97	185.09	185.00	93.28	575.26	575.00	-56.59
*RC*201/10/18/200/400/0.07	72	-20.00	-20.00	96.26	30.00	30.00	47.90	33.33	33.33	96.08	33.33	33.33	-8.89
*C*101/10/18/200/400/0.30/2	52	33.34	33.33	93.83	104.18	104.17	68.21	650.21	650.00	85.77	1475.61	1475.00	16.06
*C*101/10/18/200/400/0.30	58	100.00	100.00	94.35	437.50	437.50	22.04	-56.00	-56.00	96.36	180.04	180.00	29.69
*C*201/10/18/200/400/0.30	54	-70.00	-70.00	97.35	50.00	50.00	47.03	-38.89	-38.89	96.95	83.36	83.33	37.44
*R*101/10/18/200/400/0.30	73	-36.11	-36.11	98.38	47.23	47.22	67.03	-11.06	-11.11	99.19	168.80	168.52	78.74
*R*201/10/18/200/400/0.30	72	-75.86	-75.86	99.09	127.60	127.59	68.87	636.39	636.36	98.57	1382.99	1381.82	77.80
*RC*101/10/18/200/400/0.30	71	-19.99	-20.00	97.86	184.02	184.00	43.83	45.42	45.45	97.85	269.78	269.70	56.77
*RC*201/10/18/200/400/0.30	72	-69.44	-69.44	98.29	8.34	8.33	64.35	-65.63	-65.63	98.24	71.88	71.88	50.34

Table 4.5: Improvement of the proposed algorithms MOVND/P and MOVND/PI over the MOVND of the literature [93]

MOGVNS/P is better than MOGVNS/D in the indicators HV, NDP, and CPU by 792.71%, 679.11%, and 13.03%. The MOGVNS/P improves MOVND/PI on the three indicators HV, NDP, and CPU by about 22.02%, 21.75%, and −387.50%. The HV and NDP indicators are improved; however, here again, MOGVNS/P consumes more time than MOVND/PI when the failure cost is considered. In the opposite sense, the MOVND/PI improves MOGVNS/P by −7.55%, −7.41%, and 70.87%. We can conclude from this study that MOGVNS/P is better in all indicators compared to MOGVNS/CDP and MOGVNS/D. Moreover, MOVND/PI has approximately the same performance as MOGVNS/P, but it is considerably faster.

The convergence of the proposed algorithms for all the instances tested can be discussed from the HV results of Table 4.1. The improvement for the HV indicator of MOGVNS/P over MOGVNS/CDP is slight. The two algorithms therefore have similar convergence and coverage performance. They also have approximately the same number of non-dominated points NDP. The mono-objective GVNS, whose results are presented in Chapter 3 obtains a solution with the optimal cost found by the solver or a solution with a better objective value than the feasible solution returned by the solver. Therefore, the GVNS algorithm converges. The algorithm MOGVNS/D uses the same components (neighborhood structures, etc.) as the GVNS algorithm and an aggregated function to evaluate solutions. It therefore has the same convergence as the other GVNS algorithms. However, the set of efficient solutions returned includes only supported solutions, which explains its weak performance in the HV and NDP indicators compared to MOGVNS/P. Figure 4.4 illustrates for a given instance of 52 operations that MOGVNS/D has a good convergence but returned an incomplete Pareto front. MOVND/P is designed to be an intensification algorithm. It is a main component of MOGVNS/P and MOGVNS/CDP. It therefore has less convergence than the latter two. MOGVNS/P outperforms MOVND/PI in the HV and NDP indicators. MOGVNS/P also takes more time. Therefore, the convergence and the coverage of MOGVNS/P are better than those of MOVND/PI. The MOGVNS/P algorithm also returns more non-dominated points than MOVND/PI. This is shown in Figure 4.4. We can conclude that all the algorithms converge; however, the convergence of the GVNS algorithms is

better than the convergence of VND-based algorithms. All the GVNS algorithms proposed in this chapter have the same convergence as MOGVNS/P but differ only in terms of coverage.

Discussion of the results

In conclusion, the proposed algorithms MOVND/P, MOVND/PI, and MOGVNS/P have a better performance compared to the MOVND, and MOGVNS of [93]. One reason can be that the VND used by [93] first tries to improve each objective separately independently from the other using a single objective evaluation. As a result, the algorithms are not totally based on Pareto dominance. The method we used to fully explore the neighborhoods MOBI/P is also very impactful in improving solution quality and reducing computational time. We also force the exploration of all the neighborhoods with specific stopping criteria, while in [93], staying in the same neighborhood is necessary when it still improves. MOGVNS/P has a similar performance as MOGVNS/CDP. However, the latter consumes more time since MOGVNS/CDP is the same as MOGVNS/P applied to a population of solutions instead of one initial solution. The shaking procedure is also a better perturbation than the method we used to generate the initial population and is sufficient to diversify the search. The importance of the shaking procedure also appears when comparing the performance between MOGVNS/P and MOVND/PI. MOGVNS/P outperforms for the quality indicators MOVND/PI. MOGVNS/P is far better than MOGVNS/D for all quality indicators since the latter algorithm returns only supported solutions. All methods have a good convergence towards a good Pareto front. The methods differ in the coverage and convergence indicator, the number of non-dominated solutions, and the computational time. Our best algorithms are MOGVNS/P and MOVND/PI.

Figure 4.4 illustrates the Pareto front obtained on an instance with 52 operations named, $C201/25/18/100/200/0.30$. For this specific instance, MOGVNS/P and MOGVNS/CDP have good performance over all other algorithms. MOVND/P and MOVND/PI are better than the MOVND proposed in the literature. Moreover, MOVND/PI is also better than MOGVNS in the literature. MOGVNS/D returns an incomplete Pareto front but has a good convergence.

Instance	*n*	MOGVNS/P			MOGVNS/CDP			MOGVNS/D		
		HV($\times 10^{10}$)	NDP	CPU(s)	HV ($\times 10^{10}$)	NDP	CPU(s)	HV($\times 10^{10}$)	NDP	CPU(s)
RE/6/2/80/80/0.30/2	6	0.0003	5	0.61	0.0003	5	0.69	0.0002	3	2.17
RE/6/4/101/101/0.30/2	12	0.0909	15	22.72	0.0784	13	23.76	0.0181	3	64.39
RE/6/2/80/80/0.07/2	6	0.0007	4	0.58	0.0010	6	0.69	0.0005	3	2.37
RE/6/4/101/101/0.07/2	12	0.0862	9	19.37	0.0766	8	21.64	0.0190	2	49.14
*C*101/10/8/100/200/0.07/2	23	4.04	13	567.69	4.35	14	540.53	1.55	5	963.5
*C*101/10/5/100/200/0.07/2	23	0.62	2	215.94	1.86	6	301.86	1.24	4	995.92
*C*101/10/7/100/200/0.07	26	14.41	10	347.19	14.41	10	543.16	2.88	2	1540.26
*C*101/25/18/100/200/0.07	54	1757.32	50	36453.2	1476.15	42	32099.8	140.58	4	21119.6
*C*201/10/7/100/200/0.07	24	4.10	4	139.39	4.10	4	451.88	1.02	1	814.02
*C*201/25/18/100/200/0.07	52	983.00	28	14984.4	1954.68	57	29964.8	140.42	4	14866
*R*101/10/7/100/200/0.07	34	9.60	3	521.14	3.20	1	1425.67	3.20	1	4394.02
*R*101/25/18/100/200/0.07	68	1506.65	19	20893	1982.43	25	38759.2	158.59	2	55887.3
*R*201/10/7/100/200/0.07	34	6.19	2	899.84	12.38	4	1100.96	3.10	1	4236.49
*R*201/25/18/100/200/0.07	65	2821.33	44	47265.5	1538.86	24	39607.3	320.60	5	50983.9
*RC*101/10/7/100/200/0.07	33	7.25	2	528.11	7.25	2	1435.14	3.62	1	2839.68
*RC*101/25/18/100/200/0.07	62	1955.35	29	27539.3	3169.04	47	46969.1	337.13	5	33062.7
*RC*201/10/7/100/200/0.07	33	9.89	3	474.6	6.60	2	1363.49	6.60	2	4309.18
*RC*201/25/18/100/200/0.07	60	4794.41	82	51475.7	1871.00	32	32881.3	116.93	2	29448.2
*C*101/10/8/100/200/0.30/2	23	11.97	42	1028.66	12.25	43	1012.73	1.71	6	1255.38
*C*101/10/5/100/200/0.30/2	23	6.84	24	896.75	10.55	37	916.97	1.99	7	1230.12
*C*101/10/7/100/200/0.30	26	38.78	28	1528.05	40.17	29	1676.91	5.54	4	1407.87
*C*101/25/18/100/200/0.30	54	2364.74	69	38991.2	2707.43	79	41306.1	274.17	8	33123.8
*C*201/10/7/100/200/0.30	24	25.56	26	821.53	31.46	32	1169.56	5.90	6	872.48
*C*201/25/18/100/200/0.30	52	2107.70	66	16633	1920.48	56	32819.2	240.06	7	18837.8
*R*101/10/7/100/200/0.30	34	28.28	9	1902.74	21.99	7	3817.55	12.57	4	4957.75
*R*101/25/18/100/200/0.30	68	3599.37	46	33302.6	5242.55	67	78662.6	391.23	5	65550.5
*R*201/10/7/100/200/0.30	34	29.97	10	1551.54	56.94	19	3758.07	14.98	5	4040.9
*R*201/25/18/100/200/0.30	65	3339.90	53	80219.7	3217.40	52	41885.8	315.09	5	42049.4
*RC*101/10/7/100/200/0.30	33	14.21	4	663.56	28.44	8	1989.76	7.11	2	4935.93
*RC*101/25/18/100/200/0.30	62	2694.57	51	53688.2	2923.98	44	61284.6	398.72	6	40465.1
*RC*201/10/7/100/200/0.30	33	35.43	11	1091.48	25.77	8	3174.47	9.67	3	4613.12
*RC*201/25/18/100/200/0.30	60	4669.82	81	31113.3	2479.05	43	59595.3	461.22	8	27920.8
*C*101/10/18/200/400/0.07/2	52	364.11	21	29319.5	260.08	15	28688.7	86.69	5	56155.4
*C*101/10/18/200/400/0.07	58	180.92	5	26547.2	578.96	16	55461.7	108.55	3	57331.8
*C*201/10/18/200/400/0.07	54	263.46	10	30138.6	289.80	11	30350.5	131.73	5	42786.6
*R*101/10/18/200/400/0.07	73	2691.29	39	143386	3312.35	48	160197	138.00	2	74408.6
*R*201/10/18/200/400/0.07	72	2491.10	40	156440	1992.88	32	247398	186.83	3	79235.2
*RC*101/10/18/200/400/0.07	71	1486.71	19	19687.1	1799.71	23	106481	234.74	3	68235.6
*RC*201/10/18/200/400/0.07	72	1133.81	15	89683	1133.80	15	55479.2	302.33	4	81787
*C*101/10/18/200/400/0.30/2	52	598.30	47	48933.3	415.00	51	78190.2	56.96	7	43388.3
*C*101/10/18/200/400/0.30	58	1921.90	54	103540	1743.90	49	103414	177.95	5	58642.1
*C*201/10/18/200/400/0.30	54	755.93	29	39023.7	1251.16	48	45827.1	156.39	6	56019.9
*R*101/10/18/200/400/0.30	73	4573.85	67	193702	6485.25	95	258245	477.87	7	143966
*R*201/10/18/200/400/0.30	72	3016.52	49	138179	3570.61	58	253926	345.06	5	88666.5
*RC*101/10/18/200/400/0.30	71	3404.46	44	62050.9	4487.63	58	220287	524.99	7	83348.4
*RC*201/10/18/200/400/0.30	72	2238.12	30	98068	2536.52	34	149601	447.63	6	77137.2

Table 4.6: Results of MOGVNS/P, MOGVNS/CDP and MOGVNS/D for the maintenance cost as a second objective

Instance	n	MOGVNS/P			MOGVNS/CDP			MOGVNS/D		
		HV($\times 10^{10}$)	NDP	CPU(s)	HV ($\times 10^{10}$)	NDP	CPU(s)	HV($\times 10^{10}$)	NDP	CPU(s)
RE/6/2/80/80/0.30/2	6	0.0005	6	0.59	0.0004	5	0.65	0.0002	3	2.43
RE/6/4/101/101/0.30/2	12	0.1698	15	15.71	0.1907	17	24.67	0.0339	3	60.23
RE/6/2/80/80/0.07/2	6	0.0018	5	0.59	0.0018	5	0.71	0.0011	3	2.32
RE/6/4/101/101/0.30/2	12	0.1172	7	19.9	0.1005	6	18.95	0.0333	2	45.58
*C*101/10/8/100/200/0.07/2	23	5.31	9	216.96	10.61	18	981.69	2.36	4	1085.69
*C*101/10/5/100/200/0.07/2	23	7.04	12	274.83	2.93	5	573.82	2.93	5	1293.41
*C*101/10/7/100/200/0.07	26	16.10	6	355.05	24.15	9	1021.91	13.42	5	1627.55
*C*101/25/18/100/200/0.07	54	4446.77	65	34139.1	5092.48	74	34088.1	478.86	7	25013.6
*C*201/10/7/100/200/0.07	24	7.69	4	337.74	3.89	2	475.76	5.77	3	993.2
*C*201/25/18/100/200/0.07	52	3594.67	54	30615.3	2942.53	44	29954.2	399.40	6	20716.7
*R*101/10/7/100/200/0.07	34	164.07	31	2689.1	217.78	40	5828.58	31.76	6	5327.16
*R*101/25/18/100/200/0.07	68	18654.84	133	93925.7	15409.97	109	90059.7	1262.27	9	64565
*R*201/10/7/100/200/0.07	34	158.00	30	3081.38	200.01	37	4998.82	36.84	7	5065.3
*R*201/25/18/100/200/0.07	65	15381.74	134	74284	21163.04	183	87084	1033.06	9	60050.9
*RC*101/10/7/100/200/0.07	33	117.39	19	2763.77	120.11	19	4822.82	43.24	7	4516.16
*RC*101/25/18/100/200/0.07	62	9389.07	77	67295.3	10924.15	89	63118.8	853.54	7	29063.3
*RC*201/10/7/100/200/0.07	33	21.79	4	962.57	55.62	10	3642.36	16.33	3	4058.1
*RC*201/25/18/100/200/0.07	60	3341.87	32	24243.6	4940.53	47	47436.3	313.30	3	36816.5
*C*101/10/8/100/200/0.30/2	23	26.99	49	835.69	31.05	54	1032.43	3.86	7	1339.66
*C*101/10/5/100/200/0.30/2	23	10.99	20	369.67	16.10	28	838.58	2.75	5	1098.4
*C*101/10/7/100/200/0.30	26	106.78	41	1773.90	95.24	36	1755.06	13.02	5	2057.67
*C*101/25/18/100/200/0.30	54	5176.53	77	33325.1	7161.20	106	42890.5	201.54	6	31026.5
*C*201/10/7/100/200/0.30	24	37.05	20	927.78	39.34	21	1098.54	12.97	7	1200.75
*C*201/25/18/100/200/0.30	52	4122.09	63	23267.9	4268.45	65	24749.7	196.19	6	20101
*R*101/10/7/100/200/0.30	34	281.77	54	5339.7	203.74	38	5483.74	36.52	7	4909.06
*R*101/25/18/100/200/0.30	68	8193.86	59	31630.7	10913.77	78	67300.9	1111.17	8	72624
*R*201/10/7/100/200/0.30	34	183.93	33	2207.54	250.89	44	5704.28	39.03	7	6199.63
*R*201/25/18/100/200/0.30	65	11274.87	96	80442.8	11462.14	97	82898.8	822.02	7	53859.1
*RC*101/10/7/100/200/0.30	33	121.61	20	2045.54	161.47	26	2161.36	30.40	5	4634.17
*RC*101/25/18/100/200/0.30	62	14242.86	118	70866.5	11046.54	91	75571.3	361.91	6	43626.3
*RC*201/10/7/100/200/0.07	33	59.04	11	1814.02	38.31	7	2147.76	21.47	4	4423.04
*RC*201/25/18/100/200/0.07	60	5687.61	55	24037.7	7904.99	76	52829.2	310.14	6	37858.9
*C*101/10/18/200/400/0.07/2	52	767.28	31	30374.4	691.19	27	65021.6	173.26	7	36495
*C*101/10/18/200/400/0.07	58	1971.71	39	55556.1	1996.60	39	114965	353.90	7	58048.8
*C*201/10/18/200/400/0.07	54	787.57	21	47186.4	416.12	11	21047.7	187.51	5	45972.9
*R*101/10/18/200/400/0.07	73	17862.57	196	218379	21549.32	232	291184	820.01	9	157945
*R*201/10/18/200/400/0.07	72	16413.05	194	212096	17552.03	204	287891	761.31	9	152872
*RC*101/10/18/200/400/0.07	71	9064.57	86	139083	14973.80	140	254233	737.76	7	101806
*RC*201/10/18/200/400/0.07	72	1999.65	20	124109	2228.21	22	108438	499.90	5	77240.8
*C*101/10/18/200/400/0.30/2	52	548.62	47	66374.2	916.12	76	81434.6	70.02	6	45062.4
*C*101/10/18/200/400/0.30	58	2193.46	44	101037	3329.42	66	125092	249.18	5	68694.4
*C*201/10/18/200/400/0.30	54	1339.74	36	75004.4	1838.22	49	77837.6	297.70	8	59965.1
*R*101/10/18/200/400/0.30	73	13650.66	151	304359	13525.98	147	295760	813.41	9	221425
*R*201/10/18/200/400/0.30	72	8549.00	102	98438.4	13032.20	153	301157	838.35	10	176422
*RC*101/10/18/200/400/0.30	71	11283.23	108	197350	9114.36	86	231112	940.41	9	176085
*RC*201/10/18/200/400/0.30	72	4456.75	45	3502.3	5715.83	57	14995.39	594.30	6	132318

Table 4.7: Results of MOGVNS/P, MOGVNS/CDP and MOGVNS/D for the failure cost as a second objective

Instance	*n*	Improvement of MOGVNS/P over MOGVNS/CDP (%)			Improvement of MOGVNS/P over MOGVNS/D (%)			Improvement of MOGVNS/P over MOVND/PI (%)			Improvement of MOVND/PI over MOGVNS/P (%)		
		HV	NDP	CPU	HV	NDP	CPU	HV	NDP	CPU	HV	NDP	CPU
RE/6/2/80/80/0.30/2	6	-0.66	0.00	11.59	64.09	66.67	71.89	-0.45	0.00	-205.00	0.45	0.00	67.21
RE/6/4/101/101/0.30/2	12	15.85	15.38	4.38	401.16	400.00	64.72	36.36	36.36	-408.28	-26.67	-26.67	80.33
RE/6/2/80/80/0.07/2	6	-33.12	-33.33	15.94	34.30	33.33	75.53	0.26	-42.86	-176.19	-0.26	75.00	63.79
RE/6/4/101/101/0.30/2	12	12.53	12.50	10.49	354.59	350.00	60.58	28.71	125.00	-332.37	-22.31	-55.56	76.87
*C*101/10/8/100/200/0.07/2	23	-7.14	-7.14	-5.02	160.02	160.00	41.08	62.50	62.50	-326.35	-38.46	-38.46	76.55
*C*101/10/5/100/200/0.07/2	23	-66.67	-66.67	28.46	-50.00	-50.00	78.32	101.71	100.00	-3578.71	-50.42	-50.00	97.28
*C*101/10/7/100/200/0.07	26	0.00	0.00	36.08	400.00	400.00	77.46	11.11	11.11	-65.78	-10.00	-10.00	39.68
*C*101/25/18/100/200/0.07	54	19.05	19.05	-13.56	1150.04	1150.00	-72.60	11.11	11.11	-467.04	-10.00	-10.00	82.36
*C*201/10/7/100/200/0.07	24	0.00	0.00	69.15	299.99	300.00	82.88	-28.92	0.00	2.45	40.70	0.00	-2.51
*C*201/25/18/100/200/0.07	52	-49.71	-50.88	49.99	600.02	600.00	-0.80	-10.42	-12.50	-174.92	11.64	14.29	63.63
*R*101/10/7/100/200/0.07	34	199.97	200.00	63.45	199.82	200.00	88.14	200.96	200.00	-1847.46	-66.77	-66.67	94.87
*R*101/25/18/100/200/0.07	68	-24.00	-24.00	46.10	850.00	850.00	62.62	-29.63	-29.63	-18.55	42.10	42.11	15.64
*R*201/10/7/100/200/0.07	34	-49.99	-50.00	18.27	99.98	100.00	78.76	0.28	0.00	-3265.15	-0.28	0.00	97.03
*R*201/25/18/100/200/0.07	65	83.34	83.33	-19.34	780.00	780.00	7.29	41.94	41.94	-226.29	-29.55	-29.55	69.35
*RC*101/10/7/100/200/0.07	33	-0.04	0.00	63.20	99.91	100.00	81.40	0.43	0.00	-2076.88	-0.43	0.00	95.41
*RC*101/25/18/100/200/0.07	62	-38.30	-38.30	41.37	480.00	480.00	16.71	-14.71	-14.71	-159.51	17.24	17.24	61.47
*RC*101/10/7/100/200/0.07	33	49.95	50.00	65.19	49.95	50.00	88.99	50.54	50.00	-1857.92	-33.57	-33.33	94.89
*RC*101/25/18/100/200/0.07	60	156.25	156.25	-56.55	4000.11	4000.00	-74.80	127.80	127.78	-474.38	-56.10	-56.10	82.59
*C*101/10/8/100/200/0.30/2	23	-2.32	-2.33	-1.57	599.96	600.00	18.06	20.00	20.00	-673.95	-16.67	-16.67	87.08
*C*101/10/5/100/200/0.30/2	23	-35.13	-35.14	2.21	242.87	242.86	27.10	26.32	26.32	-610.58	-20.84	-20.83	85.93
*C*101/10/7/100/200/0.30	26	-3.45	-3.45	8.88	600.01	600.00	-8.54	-9.68	-9.68	-630.22	10.71	10.71	86.31
*C*101/25/18/100/200/0.30	54	-12.66	-12.66	5.60	762.52	762.50	-17.71	2.99	2.99	-546.14	-2.90	-2.90	84.52
*C*201/10/7/100/200/0.30	24	-18.75	-18.75	29.76	333.35	333.33	5.84	30.00	30.00	-433.46	-23.08	-23.08	81.25
*C*201/25/18/100/200/0.30	52	9.75	17.86	49.32	777.99	842.86	11.70	-34.62	-29.79	-223.49	52.95	42.42	69.09
*R*101/10/7/100/200/0.30	34	28.57	28.57	50.16	125.00	125.00	61.62	350.27	350.00	-7007.73	-77.79	-77.78	98.59
*R*101/25/18/100/200/0.30	68	-31.34	-31.34	57.66	820.01	820.00	49.20	-22.03	-22.03	-79.38	28.26	28.26	44.25
*R*201/10/7/100/200/0.30	34	-47.37	-47.37	58.71	100.00	100.00	61.60	-16.67	-16.67	-175.06	20.00	20.00	63.64
*R*201/25/18/100/200/0.30	65	3.81	1.92	-91.52	959.99	960.00	-90.77	6.00	6.00	-433.98	-5.66	-5.66	81.27
*RC*101/10/7/100/200/0.30	33	-50.01	-50.00	66.65	99.97	100.00	86.56	300.46	300.00	-2634.07	-75.03	-75.00	96.34
*RC*101/25/18/100/200/0.30	62	-7.85	15.91	12.40	575.80	750.00	-32.68	-18.90	2.00	-384.50	23.31	-1.96	79.36
*RC*201/10/7/100/200/0.30	33	37.48	37.50	65.62	266.62	266.67	76.34	57.15	57.14	-117.11	-36.36	-36.36	53.94
*RC*201/25/18/100/200/0.30	60	88.37	88.37	47.79	912.49	912.50	-11.43	62.00	62.00	-241.74	-38.27	-38.27	70.74
*C*101/10/18/200/400/0.07/2	52	40.00	40.00	-2.20	320.01	320.00	47.79	5.00	5.00	-164.57	-4.76	-4.76	62.20
*C*101/10/18/200/400/0.07	58	-68.75	-68.75	52.13	66.67	66.67	53.70	-58.33	-58.33	-78.18	140.00	140.00	43.88
*C*201/10/18/200/400/0.07	54	-9.09	-9.09	0.70	100.00	100.00	29.56	-41.18	-41.18	-162.36	70.00	70.00	61.88
*R*101/10/18/200/400/0.07	73	-18.75	-18.75	10.49	1850.17	1850.00	-92.70	-29.09	-29.09	-189.52	41.02	41.03	65.46
*R*201/10/18/200/400/0.07	72	25.00	25.00	36.77	1233.35	1233.33	-97.44	5.26	5.26	-229.26	-5.00	-5.00	69.63
*RC*101/10/18/200/400/0.07	71	-17.39	-17.39	81.51	533.35	533.33	71.15	-29.63	-29.63	54.40	42.11	42.11	-119.30
*RC*201/10/18/200/400/0.07	72	0.00	0.00	-61.65	275.02	275.00	-9.65	15.38	15.38	-99.11	-13.33	-13.33	49.78
*C*101/10/18/200/400/0.30/2	52	44.17	-7.84	37.42	950.35	571.43	-12.78	50.05	-4.08	-351.51	-33.36	4.26	77.85
*C*101/10/18/200/400/0.30	58	10.21	10.20	-0.12	980.04	980.00	-76.56	25.58	25.58	-564.54	-20.37	-20.37	84.95
*C*201/10/18/200/400/0.30	54	-39.58	-39.58	14.85	383.36	383.33	30.34	-35.55	-35.56	-216.32	55.17	55.17	68.39
*R*101/10/18/200/400/0.30	73	-29.47	-29.47	24.99	857.13	857.14	-34.55	26.41	26.42	-291.73	-20.89	-20.90	74.47
*R*201/10/18/200/400/0.30	72	-15.52	-15.52	45.58	774.20	880.00	-55.84	-25.76	-25.76	-194.16	34.69	34.69	66.01
*RC*101/10/18/200/400/0.30	71	-24.14	-24.14	71.83	548.48	528.57	25.55	-38.03	-38.03	-37.92	61.36	61.36	27.50
*RC*201/10/18/200/400/0.30	72	-11.76	-11.76	34.45	400.00	400.00	-27.13	-23.08	-23.08	-110.56	30.00	30.00	52.51

Table 4.8: Comparison of the performance of the proposed algorithms MOGVNS/P, MOVND/PI, MOGVNS/CDP, and MOGVNS/D for the maintenance cost as a second objective

Instance	*n*	Improvement of MOGVNS/P over MOGVNS/CDP (%)			Improvement of MOGVNS/P over MOGVNS/D (%)			Improvement of MOGVNS/P over MOVND/PI (%)			Improvement of MOVND/PI over MOGVNS/P (%)		
		HV	NDP	CPU	HV	NDP	CPU	HV	NDP	CPU	HV	NDP	CPU(s)
RE/6/2/80/80/0.30/2	6	19.17	20.00	9.23	97.21	100.00	75.72	13.79	20.00	-180.95	-12.12	-16.67	64.41
RE/6/4/101/101/0.30/2	12	-10.93	-11.76	36.32	400.47	400.00	73.92	51.99	50.00	-253.03	-34.21	-33.33	71.67
RE/6/2/80/80/0.07/2	6	0.63	0.00	16.90	67.74	66.67	74.57	25.04	25.00	-180.95	-20.02	-20.00	64.41
RE/6/4/101/101/0.30/2	12	16.63	16.67	-5.01	252.25	250.00	56.34	-0.17	0.00	-344.20	0.17	0.00	77.49
*C*101/10/8/100/200/0.07/2	23	-50.00	-50.00	77.90	125.03	125.00	80.02	0.00	0.00	-60.49	0.00	0.00	37.69
*C*101/10/5/100/200/0.07/2	23	139.95	140.00	52.11	139.97	140.00	78.75	9.09	9.09	-114.58	-8.33	-8.33	53.40
*C*101/10/7/100/200/0.07	26	-33.33	-33.33	65.26	20.00	20.00	78.19	-14.29	-14.29	-66.12	16.67	16.67	39.80
*C*101/25/18/100/200/0.07	54	-12.68	-12.16	-0.15	828.61	828.57	-36.48	14.04	14.04	-411.17	-12.31	-12.31	80.44
*C*201/10/7/100/200/0.07	24	97.60	100.00	29.01	33.33	33.33	65.99	33.34	33.33	-2352.72	-25.00	-25.00	95.92
*C*201/25/18/100/200/0.07	52	22.16	22.73	-2.21	800.02	800.00	-47.78	63.64	63.64	-429.62	-38.89	-38.89	81.12
*R*101/10/7/100/200/0.07	34	-24.66	-22.50	53.86	416.61	416.67	49.52	40.91	40.91	-309.26	-29.03	-29.03	75.57
*R*101/25/18/100/200/0.07	68	21.06	22.02	-4.29	1377.88	1377.78	-45.47	-11.33	-11.33	-398.23	12.78	12.78	79.93
*R*201/10/7/100/200/0.07	34	-21.00	-18.92	38.36	328.83	328.57	39.17	200.03	200.00	-376.04	-66.67	-66.67	78.99
*R*201/25/18/100/200/0.07	65	-27.32	-26.78	14.70	1388.95	1388.89	-23.70	-2.19	-2.19	-374.07	2.24	2.24	78.91
*RC*101/10/7/100/200/0.07	33	-2.26	0.00	42.69	171.52	171.43	38.80	72.90	72.73	-389.02	-42.16	-42.11	79.55
*RC*101/25/18/100/200/0.07	62	-14.05	-13.48	-6.62	1000.02	1000.00	-131.55	-1.28	-1.28	-486.58	1.30	1.30	82.95
*RC*201/10/7/100/200/0.07	33	-60.82	-60.00	73.57	33.39	33.33	76.28	100.57	100.00	-1361.32	-50.14	-50.00	93.16
*RC*201/25/18/100/200/0.07	60	-32.36	-31.91	48.89	966.68	966.67	34.15	-3.01	-3.03	-160.82	3.11	3.13	61.66
*C*101/10/8/100/200/0.30/2	23	-13.09	-9.26	19.06	600.03	600.00	37.62	44.05	44.12	-529.90	-30.58	-30.61	84.12
*C*101/10/5/100/200/0.30/2	23	-31.75	-28.57	55.92	300.01	300.00	66.34	5.25	5.26	-194.35	-4.98	-5.00	66.03
*C*101/10/7/100/200/0.30	26	12.12	13.89	-1.07	720.34	720.00	13.79	115.70	115.79	-742.83	-53.64	-53.66	88.14
*C*101/25/18/100/200/0.30	54	-27.71	-27.36	22.30	2468.43	1183.33	-7.41	13.23	13.24	-409.15	-11.69	-11.69	80.36
*C*201/10/7/100/200/0.30	24	-5.80	-4.76	15.54	185.72	185.71	22.73	33.35	33.33	-500.54	-25.01	-25.00	83.35
*C*201/25/18/100/200/0.30	52	-3.43	-3.08	5.99	2001.02	950.00	-15.75	-18.18	-18.18	-330.77	22.22	22.22	76.79
*R*101/10/7/100/200/0.30	34	38.30	42.11	2.63	671.56	671.43	-8.77	80.04	80.00	-731.51	-44.46	-44.44	87.97
*R*101/25/18/100/200/0.30	68	-24.92	-24.36	-27.80	637.41	637.50	56.45	-11.96	-11.94	-64.10	13.58	13.56	39.06
*R*201/10/7/100/200/0.30	34	-26.69	-25.00	61.30	371.25	371.43	64.39	55.92	43.48	-235.11	-35.86	-30.30	70.16
*R*201/25/18/100/200/0.30	65	-1.63	-1.03	2.96	1271.61	1271.43	-49.36	-10.28	-13.51	-408.50	11.46	15.63	80.33
*RC*101/10/7/100/200/0.30	33	-24.69	-23.08	5.36	300.04	300.00	55.86	-23.07	-23.08	-253.22	30.00	30.00	71.69
*RC*101/25/18/100/200/0.30	62	28.94	29.67	6.23	3835.47	1866.67	-62.44	49.39	49.37	-560.90	-33.06	-33.05	84.87
*RC*201/10/7/100/200/0.30	33	54.11	57.14	15.54	174.94	175.00	58.99	83.74	83.33	-233.98	-45.57	-45.45	70.06
*RC*201/25/18/100/200/0.30	60	-28.05	-27.63	54.50	1733.87	816.67	36.51	5.77	5.77	-179.35	-5.46	-5.45	64.20
*C*101/10/18/200/400/0.07/2	52	11.01	14.81	53.29	342.86	342.86	16.77	40.91	40.91	-164.85	-29.03	-29.03	62.24
*C*101/10/18/200/400/0.07	58	-1.25	0.00	51.68	457.14	457.14	4.29	11.43	11.43	-230.76	-10.26	-10.26	69.77
*C*201/10/18/200/400/0.07	54	89.27	90.91	-124.19	320.02	320.00	-2.64	75.00	75.00	-293.76	-42.86	-42.86	74.60
*R*101/10/18/200/400/0.07	73	-17.11	-15.52	25.00	2078.35	2077.78	-38.26	-26.04	-26.04	-347.53	35.20	35.20	77.65
*R*201/10/18/200/400/0.07	72	-6.49	-4.90	26.33	2055.89	2055.56	-38.74	-18.48	-18.49	-361.80	22.67	22.68	78.35
*RC*101/10/18/200/400/0.07	71	-39.46	-38.57	45.29	1128.67	1128.57	-36.62	-36.29	-36.30	-211.78	56.97	56.98	67.93
*RC*201/10/18/200/400/0.07	72	-10.26	-9.09	-14.45	300.01	300.00	-60.68	66.67	66.67	-171.33	-40.00	-40.00	63.14
*C*101/10/18/200/400/0.30/2	52	-40.11	-38.16	18.49	683.54	683.33	-47.29	-25.40	-25.40	-497.43	34.05	34.04	83.26
*C*101/10/18/200/400/0.30	58	-34.12	-33.33	19.23	780.25	780.00	-47.08	-37.14	-37.14	-496.81	59.09	59.09	83.24
*C*201/10/18/200/400/0.30	54	-27.12	-26.53	3.64	350.03	350.00	-25.08	9.09	9.09	-480.98	-8.34	-8.33	82.79
*R*101/10/18/200/400/0.30	73	0.92	2.72	-2.91	1578.21	1577.78	-37.45	4.15	4.14	-522.13	-3.99	-3.97	83.93
*R*201/10/18/200/400/0.30	72	-34.40	-33.33	67.31	919.74	920.00	44.20	-37.44	-37.42	-111.57	59.85	59.80	52.73
*RC*101/10/18/200/400/0.30	71	23.80	25.58	14.61	1099.82	1100.00	-12.08	-11.47	-11.48	-340.27	12.96	12.96	77.29
*RC*201/10/18/200/400/0.30	72	-22.03	-21.05	-123.42	649.92	650.00	74.68	-18.18	-18.18	29.22	22.22	22.22	-41.29

Table 4.9: Comparison of the performance of the proposed algorithms MOGVNS/P, MOVND/PI, MOGVNS/CDP and MOGVNS/D for the failure cost as a second objective

This instance can represent the results of the performance of all the algorithms on average. The proposed MOGVNS/P algorithm has better convergence compared to MOVND/PI. The average percent results obtained in Table 4.1 show the same performance. However, the results of MOVND/PI, on average, are close to those of MOGVNS/P, although it is less converging.

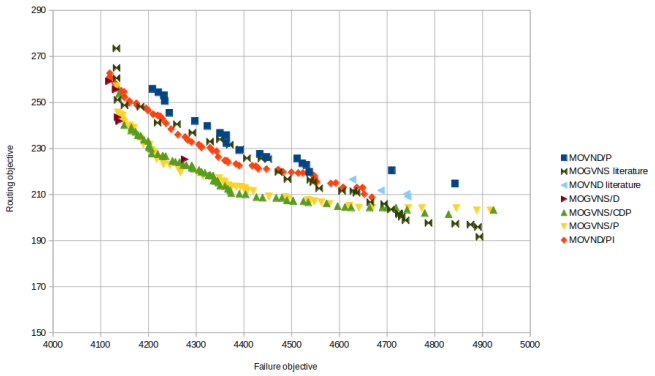

Figure 4.4: Pareto fronts of the different algorithms for the instance with 52 operations, C201/25/18/100/200/0.30, and the failure cost as second objective.

4.6 Concluding remarks

In this chapter, we addressed the joint maintenance and workforce routing problem. The problem was modeled in Chapter 3. The bi-objective variant is studied in this chapter. We seek to minimize simultaneously both maintenance and transport costs. For the first time, a bi-objective approach is proposed to deal with this problem, where we associate either the failure cost or the maintenance cost with the routing cost. New adaptations of variable neighborhood descent and general variable neighborhood search frameworks called respectively MOVND/P, MOVND/PI, and MOGVNS/P are proposed to deal with combinatorial multi-objective optimization problems, and this problem, particularly. To obtain high-quality solutions, we described how to design the improvement method, the acceptance criterion, the stopping criterion, and the neighborhood change procedure. These algorithms are based on the Pareto dominance concept and use a new multi-objective best improvement strategy

called MOBI/P. We test a pure decomposition approach. The resultant algorithm, MOGVNS/D, decomposes the problem into several scalar subproblems and uses the weighted sum method to evaluate the solutions. This algorithm is easily convertible to a mono-objective general variable neighborhood search. Finally, we test the use of a population of solutions with MOGVNS/P. The resulting variant is called MOVNS/CDP. Several numerical experiments have been performed to validate our proposals. The proposed algorithms were evaluated on randomly generated instances. The obtained results also demonstrate that MOVND/P, MOVND/PI, and MOGVNS/P outperform existing MOVND and MOGVNS in the literature for all indicators measuring convergence and coverage in less computational time. Compared to the other proposed MOGVNS variants, MOGVNS/P is significantly better than MOGVNS/D for all the quality and time indicators since the latter algorithm returns only supported solutions. MOGVNS/P slightly outperforms MOGVNS/CDP in less computational time. Moreover, MOVND/PI and MOGVNS/P have almost similar performances with a time advantage in favor of MOVND/PI.

In the next chapter, we integrate another maintenance strategy to reduce maintenance costs and improve the quality of maintenance services in a plant. We also propose another approximate method to solve the defined problem.

Chapter 5

The Joint Opportunistic Maintenance Scheduling and Workforce Routing Problem: Model and Solution Approach based on Adaptive Large Neighborhood Search

Outline of the current chapter

5.1 Introduction

Maintenance is crucial to ensure the operation of transport systems (ships, buses, trains, airplanes, etc.), communication systems (televisions, phones, etc.), manufacturing and processing plants, hospitals, and banks [3]. Moreover, it increases the sound functioning of industrial activities such as transport [103].

The maintenance process impacts both the availability of the production tools, and the quality of the manufactured products [108]. Furthermore, the importance of the maintenance function is closely related to the growing demand for system productivity, availability, safety, and customer satisfaction. Therefore, it is necessary to synchronize maintenance decisions with the production demands to improve the company's global performance. Maintenance functions should be well planned and globally optimized to meet product quality in a reasonable time and at the lowest overall cost. A well-defined maintenance strategy increases the residual lifetime of equipment and the performance of the manufacturing system. A well-defined maintenance plan is critical for a company to profitably and efficiently operate. Corrective and preventive maintenance strategies are the most adopted strategies. In Preventive Maintenance (PM), operations are performed at regular time intervals, while in the Corrective Maintenance (CM) strategy, the operation is performed when a failure occurs. The traditional optimization problem consists of determining the best trade-off between preventive and corrective maintenance [89]. Preventive maintenance is cheaper than corrective maintenance. Indeed, when equipment suddenly breaks down, the company incurs very high costs related to the loss of production, environmental damage, and safety consequences [109]. However, moving from traditional maintenance strategies towards more adapted maintenance strategies such as Condition Based Maintenance (CBM) or Opportunistic Maintenance (OM) is now necessary. Condition Based Maintenance focuses on predicting the health condition of the system using information collected from sensors. It is only performed when a threshold of equipment deterioration is exceeded. Opportunistic Maintenance (OM) is the right instant to perform Preventive Maintenance. This term was first mentioned in the 1960s by researchers from the RAND Project, now called RAND Corporation [107]. There are many OM

definitions in the literature. From the economic point of view, an opportunity is defined as any period where a maintenance operation can be realized without any loss or negative impact on production [107]. Opportunities are events that occasionally happen and are, therefore, difficult to predict, or they are planned if they are associated with already known production stoppages. According to [107], OM tries to answer the following questions: what is the appropriate time to perform a maintenance operation? What components need this maintenance? What components are prior? Which opportunity to choose among several?

Maintenance is intended to maintain production systems. These latter need to be stopped most of the time for maintenance operations. Therefore, this negative effect must be considered in maintenance planning and optimization. PM should be performed when the production is impacted the least. Consequently, the production stoppages need to be considered as opportunities to use for maintenance actions. Moreover, when the machines are geographically distributed, sequencing the technicians' visits and determining the best routing plan are additional issues that complicate the problem. The combined maintenance and routing problem was studied in several works [82] [83] [117] and in our previous works [98], [99] and chapters 3 and 4. In this work, we consider planned production stoppages as opportunities. We then look for the best opportunities to perform PM operations. The start times chosen for PM operations have to respect maintenance time windows and ensure the functioning of the equipment at the least possible overall cost.

To the best of our knowledge, no study considers the integrated maintenance, production, and routing problem. This chapter addressed a new variant of the combined maintenance and routing problem that considers production stoppages as opportunities to schedule preventive maintenance operations.

The problem addressed in this paper is faced by service providers that should plan maintenance operations on a set of geographically distributed machines subject to random failures. The objective of the problem is to determine the best technicians' routing plan to do maintenance operations while considering off-peak production hours as advantageous periods.

The contributions of this chapter are presented hereafter:

(1) The new problem is first defined. Then, the related research streams are introduced. Furthermore, the problem's potential stream of research is highlighted. Next, a mathematical formulation is proposed, aiming to minimize maintenance, transport, and production losses costs in the case of time-based opportunistic preventive maintenance. The first objective minimizes the total technicians' transport costs, the penalty cost, and the production losses cost. The second objective minimizes the total maintenance cost with the penalty cost and production losses cost. In both objectives, the penalty cost is related to technicians' late arrivals and the production losses incurred when the planned production stoppages are not used for maintenance activities. The present chapter comes with the novelty of introducing production requirements with the problem adressed in Chapter 3. To the best of our knowledge, no previous work in the literature considers the routing and maintenance scheduling with production constraints.

(2) A dedicated heuristic is then proposed. It inputs a sequence of maintenance operations, selects the right production stoppage, and computes for each maintenance operation its start time considering the selected production stoppage. This heuristic is then applied to evaluate the solutions explored. Finally, it is integrated with a Semi-Adaptive Large Neighborhood Search (SALNS), specially designed to solve the problem. The SALNS includes a new removal operator based on failure risks and a novel adaptation of the adaptive mechanism to achieve the best results.

(3) Extensive experiments demonstrated that the SALNS with the integrated heuristic outperformed the commercial solver. Moreover, the utility of the heuristic is shown through a sensitivity analysis of the predominant parameter that directly influences the results. The SALNS is finally tested on the problem of Chapter 3 to highlight the similarities with the GVNS metaheuristic.

The remainder of this chapter is organized as follows: Section 5.2 gives an overview of related problems in the literature. For these latter, Section 5.3 surveys the resolution methods. Section 5.4 describes the combined maintenance and routing problem with production stoppages considerations and provides its

mathematical formulation. A description of the proposed solution approach and details on its implementation is described in Section 5.5, followed by experimental results and discussion in Section 5.6. Section 5.7 provides the computational results of the method adopted for the problem of Chapter 3. Finally, concluding remarks are presented in Section 5.8.

5.2 Literature on related problems

5.2.1 The integration of vehicle routing problem and maintenance

Vehicle routing and maintenance scheduling problems have been widely studied in the literature. Most of the existing studies that tackle routing and maintenance problems separately deal with routing and maintenance. In [98], [99] and Chapter 3, we identified two main literature research streams. The first uses the routing model to plan maintenance operations, and the second combines maintenance and routing considerations in the same modeling. In the first stream, Cordeau et al. [77] proposed a construction heuristic and a basic adaptive large neighborhood search heuristic to solve the technician and task scheduling problem. They considered technicians' skills with different proficiency levels, teaming, task priorities, and a time window associated with each task. In the second stream, Lopez-Santana et al. [82] proposed a mathematical model called the Combined Maintenance and Routing (CMR) and a two-phase procedure to solve it.

5.2.2 The integration of vehicle routing problem and production

The Production Routing Problem (PRP) combines production scheduling and vehicle routing problems. Given a plant that produces and distributes one product to several customers, the objective of the PRP is planning the production, inventory, and routing while minimizing the overall cost. Li et al. [112] modeled the PRP with multiple products and outsourcing options in a mixed-integer

linear program and proposed a Three-Level Heuristic (TLH) to solve it and solve the classical PRP. TLH includes a two-phase iterative method, a repairing strategy, and a fix-and-optimize procedure. The heuristic is tested on newly generated instances and exiting benchmark instances for the classical PRP.

5.2.3 The integration of maintenance and production

Policies that separately plan the production and the maintenance operations are not efficient [109]. Production and maintenance functions are interconnected since the equipment's degradation is related to machine usage. The failure of machines involves stopping production. This fact can generate conflicts between these two departments in a company. Therefore, researchers have understood that setting up policies that satisfy maintenance and production considerations is very important. Integrating production into maintenance planning is a common problem studied in the literature. Several perspectives and points of view can be distinguished to integrate maintenance and production. Budai et al. [109] distinguished the following streams of research: planning maintenance while considering production, planning production taking into account maintenance, and finally, planning maintenance as if it is a production process.

In the first perspective, maintenance can be planned while considering production. Indeed, production systems must be stopped most of the time for maintenance actions. This negative effect has therefore to be considered in maintenance planning and optimization. This research stream is composed of the following substreams [109]: costing of downtime, opportunistic maintenance, and maintenance scheduling in line with production. Costing of downtime consists of determining an accurate estimation of maintenance costs, including even the unforeseen consequences caused by the corrective maintenance. Opportunistic maintenance occurs at an opportune moment when the units are less needed for production. Opportunities are events that may occasionally occur and are, therefore, in this particular case, difficult to predict. The reasons behind opportunistic maintenance can be that a failure of one component is often an opportunity to perform preventive maintenance on the other components or because of other outside interruptions of production (market interruptions,

etc.). Opportunistic maintenance was also studied by Thomas et al. [107]. They defined OM as the grouping of maintenance operations, associating a preventive maintenance operation with corrective maintenance operations to benefit from the proximity between components and finally performing a preventive maintenance operation during an opportunity. This latter category is called synchronization maintenance-production. The grouping of maintenance operations can be achieved in two main ways [107]. The first one is associating several maintenance operations in one planned production stoppage even if the PM operations need to be realized in other different future times. The second one usually used in industry is performing a non-planned preventive maintenance operation while doing another planned preventive maintenance operation. The latter practice should be avoided since it could influence the safety of maintenance agents. According to Thomas et al. [107], these two OM strategies have not been studied in the literature. Associating a preventive maintenance operation with corrective maintenance operations is an OM strategy that uses the proximity between components. The technicians inspect the state of components close to a piece that requires a CM maintenance operation. However, very few studies propose an investigation or are dedicated to this notion of proximity [107]. Synchronization maintenance-production supposes that future production stoppages are imposed and already planned by the production. The causes of these stoppages differ (environment, legal recommendations, market, lack of raw material, etc.). Tan and Kramer [111] introduced a framework for preventive maintenance optimization that combines Monte Carlo simulation with a Genetic Algorithm. The methodology proposed includes opportunistic maintenance with corrective and preventive maintenance operations. Their method outperformed Markov techniques in solution accuracy. Cheung et al. [110] proposed a multi-period Mixed Integer Linear Programming (MILP) model to optimize short-term site-wide maintenance plans. Each plant had several shutdown, overhaul, and startup periods to maintain. The problem was allocating plant maintenance to these periods with the least effect on production. Maintenance was incorporated in the proposed MILP model through its frequency and shutdown periods considered with overhauling and startups. In addition to the previous constraints, inspection, material, and utility balance equations, plant bounds, and inventory

constraints were considered. Thomas et al. [108] proposed a novel approach based on the "odds algorithm" to select among the planned production stoppages the ones that would be optimal for performing maintenance operations. The prognosis process provided the set of maintenance operations that needed to be performed on the horizon. The method was proved optimal. However, the authors hypothesized that the maintenance operation should necessarily start at the beginning of the production stoppage. Planning of maintenance while considering production also includes the last substream: maintenance scheduling in line with production. The production here is included in the modeling. However, the model's objective is to address maintenance decisions without determining how to plan the production in a better way [109].

The second stream [109] treats the planning of production with maintenance considerations. Maintenance operations need to be simultaneously planned with production. Maintenance has to be done either because of a failure or because the quality of the produced items is not good enough. This kind of problem is called the integrated planning of production and maintenance.

In the last research stream, maintenance is seen as a production process that needs to be planned. We seek to determine appropriate levels of capacity concerning the maintenance demand. Details about substreams of each stream can be consulted in [109].

5.2.4 The integration of vehicle routing problem, maintenance and production

To the best of our knowledge, the problem addressed in this chapter is a new problem that integrates Vehicle Routing Problem with opportunistic time-based maintenance scheduling. Indeed, we consider production stoppages as opportunities to select when planning maintenance operations. It extends the joint maintenance scheduling and workforce routing problem defined in our previous work of [98], [99], and chapters 3 and 4. Integrating production with the joint maintenance scheduling and workforce routing problem generates new issues in the literature. The problem presented in this chapter studies maintenance planning and considers production stoppage intervals as constraints imposed by

the production department. It can be positioned in the opportunity maintenance stream, especially the synchronization of production and maintenance. Through this literature review and analysis, we defined a new problem that deals with three aspects: routing, maintenance, and production. The problem addressed in this chapter integrates routing and maintenance scheduling while considering production.

Figure 5.2 sums up this literature section and clearly shows the positioning of the problem. The existing streams are blue and green colored. The problem defined is positioned in the research streams green highlighted.

5.3 Literature on related solution methods

5.3.1 Resolution methods applied to related problems

This section focuses on existing solution methods appiled to related problems. For a long time, problems that integrate maintenance planning and technician routing have been addressed in the literature using vehicle routing to schedule maintenance tasks. Several solving methods have been proposed in the ROADEF challenge of 2017 [123]. We find approximate mixed-integer programming and branch and price among the exact methods. The metaheuristics used are Adaptive Large Neighbourhood Search (ALNS), the matheuristic parallel ALNS, Greedy Randomized Adaptive Search Procedure, and Intelligent Decision heuristic. The most used metaheuristics are generally based on local search techniques such as Iterative Local Search, Hill Climbing, and Simulated Annealing [123]. Kovacs et al. [78] proposed an ALNS algorithm to solve the service technician routing and scheduling problem. The problem considered the following features: outsourcing costs, team building, and technicians' skills and their level in each skill. Several works dealt with the previous problem by including technical aspects of maintenance. We addressed in [98] and Chapter 3 the combined maintenance scheduling and routing problem and proposed an optimization model that extends the work of [82] with three objectives: the routing, the failure, and the maintenance cost. To solve the problem, constructive heuristics based on failure and maintenance behavior and a General Variable Neighborhood

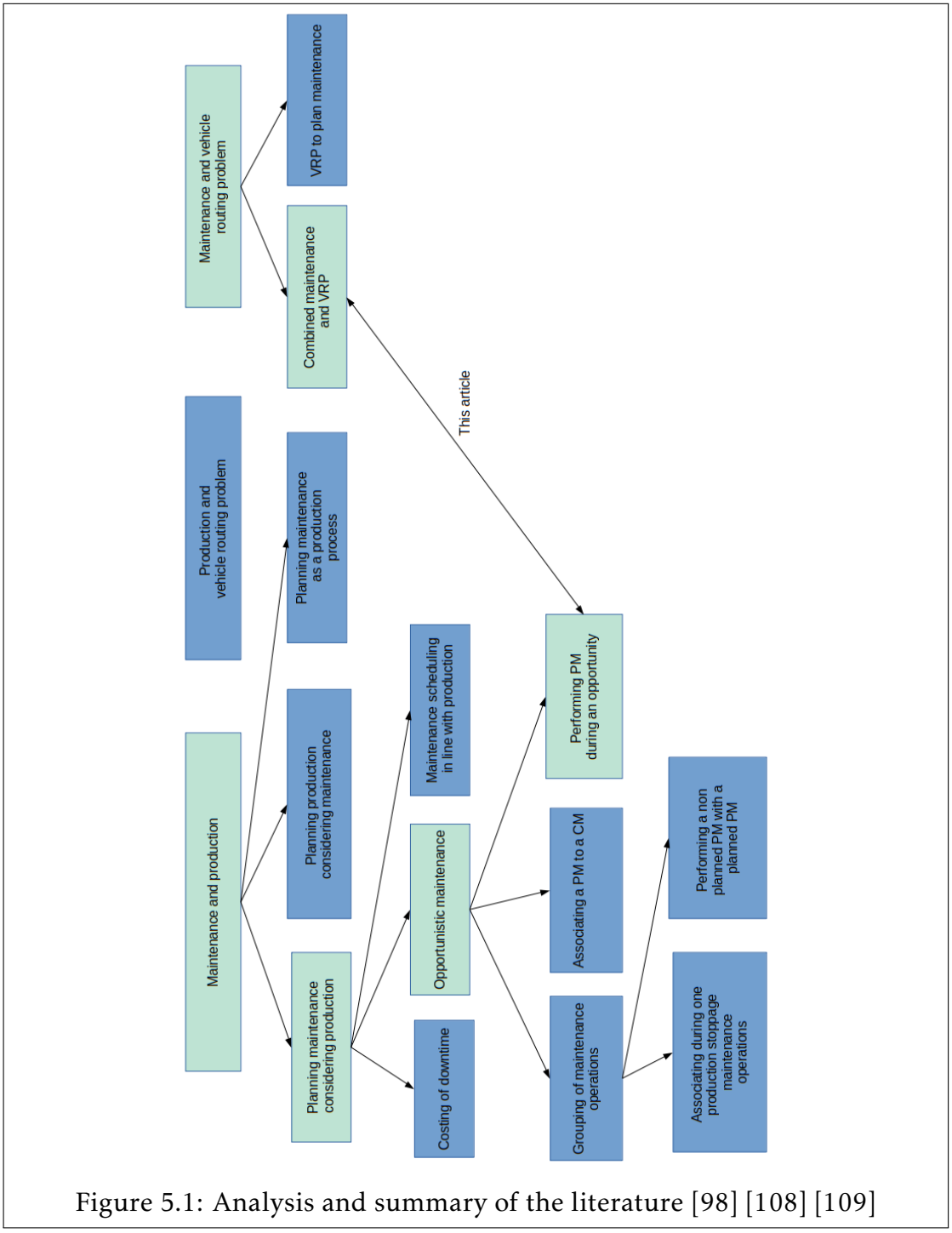

Figure 5.1: Analysis and summary of the literature [98] [108] [109]

Search (GVNS) were proposed. Results have highlighted that using dedicated constructive heuristics to generate initial solutions allow us to reach the optimum solutions in a short computational time. Moreover, the GVNS algorithm outperformed the commercial solver. We then considered the bi-objective variant in [99] and Chapter 4. It has been solved using novel multi-objective VND and GVNS. The algorithms used a novel best improvement strategy and adaptations of VNS mechanisms to the multi-objective case. The proposed algorithms outperformed the literature VND and GVNS methods. Finally, Chen et al. [117] proposed a dynamic period vehicle routing problem with profit (DPVRPP) to maximize the drains' daily changing conditions. They then proposed an adaptive planning heuristic with both routing and adaptive planning stages to solve the problem. The search method adopted is VNS with many local moves.

Adaptive Large Neighborhood Search (ALNS) is the most used variant for maintenance and routing scheduling problems that use routing to plan maintenance operations, especially if skilled technicians are considered. Variable Neighborhood Search (VNS) has been recently successfully applied for these problems. The ALNS and VNS methods have been recently hybridized in the literature to solve the VRP more efficiently. Sze et al. [126] proposed an Adaptive Variable Neighbourhood Search (AVNS) to solve the Capacitated Vehicle Routing Problem in which Large Neighborhood Search (LNS) was used as a diversification procedure. The proposed AVNS algorithm comprises two stages: a learning phase and a multi-level VNS with a local guided search.To speed up the algorithm, they integrated a flexible data structure and a neighborhood reduction scheme. As a result, the AVNS algorithm produced good results compared to existing methods in the literature.

5.3.2 Analyzing the state of the art of selection mechanisms used in ALNS

ALNS is an effective method but has many limits under study. One major limitation is the design of the destroy and repair heuristics. First, these operators should be particular to cope with the problem's peculiarities. Secondly, the selection strategy of those operators should be efficient and well-designed. That's

why most research on this method focuses on these two aspects. Hereafter, we analyze some of the existing adaptive selection mechanisms.

Roulette wheel mechanism : The roulette wheel is a probabilistic operator selection method. It has been mainly used to select chromosomes for crossover in the Genetic Algorithm and ALNS metaheuristic to select a couple of destroying and repair operators. At each iteration of the ALNS algorithm, operator scores are updated. They are the basis of weight determination. Probabilities to select the different operators are then computed using those weights depending on whether the best solution is improved, the current solution is the only one improved, or neither solution is improved. However, the new solution is accepted anyway by a given probability. The roulette wheel is used later in our ALNS algorithm. We suggest consulting the detailed procedure and equations used in Section 5.5.4. Ropke and Pisinger [113] proposed the ALNS metaheuristic for solving the Pickup and Delivery Problem with Time Windows as an extension of the LNS that considers several heuristics instead of one ruin and one create operators. They used the roulette wheel mechanism to select the heuristic based on the search history.

Cost improvement-based adaptation Laborie et Godard [118] proposed an improved roulette wheel mechanism called cost improvement-based adaptation. The scores are obtained differently compared to the roulette wheel method. The cost improvement of the operator when the new solution is better than the best solution and the computational cycle time of the destroy or repair operator are used in this selection mechanism. The ruin and create operators are proportionally rewarded to the ratio of the cost improvement on their running time. This approach favors operators that improve the solution in less computational time. The operators chosen are the ones that achieve a good solution faster.

Learning Automata Learning Automata is a machine learning approach in which one action among several actions is selected from the random environment while repeatedly interacting with this latter. The chosen action is then considered again as an input for the random environment to learn. Detailed description of

this strategy can be consulted in [114] [122]. This mechanism has been recently used in ALNS by Cota et al. [114] for the Unrelated Parallel Machine Scheduling Problem with Setup Times to minimize the makespan. The proposed algorithm is called Adaptive Large Neighborhood Search with Learning Automata (ALNS-LA).

5.4 Problem definition and formulation

We consider a set of geographically distributed machines $\mathcal{M}$ in multiple customer sites subject to non-deterministic failures, which can cause sudden failures and production losses. Preventive Maintenance (PM) operations are scheduled for each machine regularly to reduce the risk of unexpected breakdowns. A PM operation i costs Cpm_i and has a duration Tpm_i. Each preventive maintenance operation i needs to be performed within its time window $[a_i, b_i]$. Nevertheless, if it is not realizable, because of the high workload of technicians, a team of technicians k can start an operation i after its latest allowed start time b_i. In the latter case, a penalty of $p_{i,k}$ is incurred. The lower bound of the time window a_i is a hard constraint that needs to be respected, meaning that if a technician's team arrives earlier, it should wait until a_i to start the PM operation. Teams of technicians $\mathcal{K}$ with the same skills are assigned to realize preventive and corrective maintenance operations at opportune moments. Corrective Maintenance (CM) occurs when a machine suddenly breaks down. In this case, the machine remains unusable for a time length of W_i until the technicians repair it. Each operation's waiting cost per unit time is Cw_i. A CM operation i costs Ccm_i and lasts a time period Tcm_i. The equipment is shut down during the waiting time for a CM operation W_i. The cost of this waiting is $Cw_i \times W_i$. It can represent the production loss incurred by the customer for this operation i before repairing the associated failed machine. The maintenance scheduling process provides all the maintenance operations with their characteristics (optimal timing ϕ_i, intervention interval $[a_i, b_i]$, frequency of operations, the number of all operations n) that should be performed on the planning horizon to ensure the well functioning of the machines in each production site. The objective of the problem is to assign the teams of technicians to the maintenance operations while trying to execute

the maintenance operations at their optimal times ϕ_i.The optimal execution timing ϕ_i of each PM operation and its allowed time interval $[a_i, b_i]$ are obtained with a formalized cost model for optimizing the maintenance intervals. During the horizon, some production stoppages $[e_p, l_p]$ are known and already planned. We look to schedule each PM operation during one of the production stoppages p to reduce the production losses incurred because of the shutdown of the machines in work periods while respecting the maintenance interval $[a_i, b_i]$. Off-peak hours of production are considered. The decision-maker has at his disposal the production stoppages $[e_p, l_p]$ before maintenance planning. A production stoppage is a time window for which the beginning e_p and the duration d_p are known and therefore permits us to obtain the upper bound l_p. The production stoppages are soft time windows. They can be violated with a penalty pe_{ip} or pl_{ip}. We seek to determine what is the most appropriate production stoppage, if any, that allows performing the PM operation i with fewer production losses. The node 0 is considered the departure point of maintenance teams, and the node $n+1$ is considered the same arrival point for the teams. We denote by $\mathcal{O}$ the set of all nodes associated with PM operations and by $\mathcal{V}$, the set of operations nodes plus the departure and the arrival nodes. The model can be represented as a directed complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{A} = \{(i,j), i \in \mathcal{V}^o, j \in \mathcal{V}^d, i \neq j\}$ with $\mathcal{V}^o$ and $\mathcal{V}^d$ are the sets of origin and destination nodes. All the hypothesis of the problem of Chapter 3 holds for this chapter. In addition, the following assumptions are considered regarding the available production stoppages:

- During the whole duration of production stops time windows, there are no production activities, and the machines are stopped.
- Production stops time windows are opportunity periods to do maintenance operations.
- During the production stoppages, there are no production losses applied when doing maintenance.
- The production stops do not overlap and are characterized by a start time, an end time, and a duration. All these values are deterministic.

- Two consecutive production stoppages intervals are far enough apart so that the PM operations can not start in a production stoppage interval and terminate in another consecutive production stoppage interval.

- The duration of the production stoppages is considered superior to the duration of the maintenance operations.

5.4.1 Notation of the opportunistic maintenance and routing problem

The following notations are used throughout the paper.

Sets

- $\mathcal{M} = \{1, \ldots, M\}$: the indexes of machines.

- $\mathcal{O} = \{1, \ldots, n\}$: the indexes of preventive maintenance operations ($n = \sum_{m \in \mathcal{M}} n_m$). The parameter n_m is the number of operations related to machine m.

- $\mathcal{V} = \{0, \ldots, n+1\}$: set of nodes (departure depot 0, indexes of maintenance operations, arrival depot $n+1$).

- $\mathcal{V}^o = \{0, \ldots, n\}$: set of origin nodes.

- $\mathcal{V}^d = \{1, \ldots, n+1\}$: set of destination nodes.

- $\mathcal{K} = \{1, \ldots, K\}$: set of technicians teams.

- $\mathcal{A} = \{(i, j), i \in \mathcal{V}^o, j \in \mathcal{V}^d, i \neq j\}$: set of arcs between nodes.

- $\mathcal{P} = \{1, \ldots, P\}$: set of production stoppages time windows.

Maintenance parameters

- Cpm_i, Ccm_i, Cw_i: cost of preventive and corrective maintenance operation (PM and CM) of operation i, and the unit waiting costof the arrival of a technicians team to operation i for repairing a sudden failure on its associated machine.
- Tpm_i, Tcm_i: duration of PM and CM task of operation i.
- $M_i(t)$: mean time to failure when the failure occurs before t concerning operation i.
- W_i: waiting time before starting a CM operation i when there is sudden failure on the associated machine.
- $CM_i(t)$: the total expected cost of maintenance per unit time of the operation i when performed at the age t.
- β_i, σ_i: shape and scale parameters of the Weibull distribution of an operation i corresponding to a machine where $\beta_i > 1$.
- $T_i, F_i(t), f_i(t)$: random variable representing the time to failure, cumulative distribution and density function of the Weibull distribution.

Routing parameters

- $t_{i,j}$: travel time of the arc $(i,j) \in \mathcal{A}$.
- $c_{i,j}$: routing cost of the arc $(i,j) \in \mathcal{A}$.
- a_i: lower bound of the time window to perform a PM operation i.
- b_i: upper bound of the time window to perform a PM operation i.
- d_i: The duration of the maintenance operation i. It is equal to $d_i = Tpm_i(1 - F_i(\delta_i^*)) + Tcm_i F_i(\delta_i^*), \quad i \in \mathcal{O}$. The optimal PM time δ_i^* for operation i is the solution of the equation of the cost model $\min(C_m(\delta_m))$ (5.19) where m is the machine that corresponds to the operation i.

- c: fixed penalty cost per unit time for not respecting the latest time b_i of the time window.
- K: total number of technicians teams or vehicles.
- B: big number at least superior to the maximum duration of the tours.
- e_p: lower bound of the time window of production stop $p \in \mathcal{P}$.
- dp_p: the duration of the time window of production stop $p \in \mathcal{P}$.
- l_p: upper bound of the time window of production stop $p \in \mathcal{P}$ where $l_p = e_p + dp_p$.

Variables

- $x_{i,j,k}$: binary decision variable which is set to 1 if the team of technicians k traverses the arc $(i, j) \in \mathcal{A}$ and 0 otherwise.
- $\theta_{i,k}$: the start time of the maintenance operation i by the team of technicians k.
- $p_{i,k}$: the length of the delay when the team of technicians k performs a PM operation i after the closing of the time window b_i. It is equal to $max(0, \theta_{i,k} - b_i)$.
- $\rho_{i,k}$: time of the last intervention on the same machine previous to the i-th PM operation when the team of technicians k performs it. If no previous operation exists, this time is set to 0.
- $\lambda_{i,p}$: binary variable that equal 1 if the production stoppage interval p is selected for PM operation i.
- $pe_{i,p}$: a variable expressing the amount of violation of the lower bound of the production stop time window e_p when the instant of the beginning of the maintenance operation is $\theta_{i,k}$. It is equal to $max(0, e_p - \theta_{i,k})$.

- $pl_{i,p}$: a variable expressing the amount of violation of the upper bound of the production stop, l_p when the instant considered is $\theta_{i,k} + d_i$, the end of the operation. It is equal to $max(0, \theta_{i,k} + d_i - l_p)$.

  The variables of the maintenance model used here that are constant input parameters for the routing one are defined as follows:

- δ_m: optimal PM period for machine m.
- ϕ_i: execution time of operation i obtained using δ_m.
- n: number of PM operations to perform in the planning horizon.

5.4.2 The maintenance model

The optimal time of each maintenance operation is obtained by solving a formalized cost model to optimize the maintenance intervals. The maintenance model aims to determine the optimal PM time δ_m of machine m that minimizes the total maintenance cost of preventive and corrective maintenance operations.

The cost model presented in [82] is an extension of the classical cost model used in reliability-centered maintenance presented in [89]. This extension includes the consideration of the maintenance durations, the waiting time until technicians arrive in the case of corrective maintenance operations, and an iterative procedure to connect the VRP with the maintenance cost model.

$$CM_m(\delta_m) = \frac{Cpm_m(1-F_m(\delta_m))+(Ccm_m+W_m*Cw_m)F_m(\delta_m)}{(\delta_m+Tpm_m)(1-F_m(\delta_m))+(M_m(\delta_m)+W_m+Tcm_m)F_m(\delta_m)} \tag{5.1}$$

Where $M_m(\delta_m)$ is:

$$M_m(\delta_m) = \int_0^{\delta_m} \frac{t f_m(t)}{F_m(\delta_m)} \, \mathrm{d}t \tag{5.2}$$

We propose an approximation to avoid this iterative procedure in Chapter 3 illustrated in equation (5.3). This approximation has also been adopted by [128]. Details about this approximation can be found in our work [98], [99] and Chapter 3. The waiting time is equal to:

$$W_m = \delta_m - M_m(\delta_m) \tag{5.3}$$

The maintenance cost per unit time with this waiting time expression for a machine m at the time δ_m is finally equal to:

$$CM_m(\delta_m) = \frac{Cpm_m(1-F_m(\delta_m))+(Ccm_m+(\delta_m-M_m(\delta_m))Cw_m)F_m(\delta_m)}{(\delta_m+Tpm_m)(1-F_m(\delta_m))+(\delta_m+Tcm_m)F_m(\delta_m)} \tag{5.4}$$

This non linear equation without constraints is solved to obtain the optimal period for each machine m, $\delta_m^* = \text{argmin } CM_m(\delta_m)$.

A detailed description of how to obtain the frequency of PM operations in the horizon and the time windows used as data for the routing model could be found in our work [98], [99] and Chapter 3.

5.4.3 The integrated opportunistic maintenance and routing model with production constraints

The mathematical model can be formulated as follows:

$$\min(f_l(x_{i,j,k}, \theta_{i,k}, \rho_{i,k}, p_{i,k}, pe_{i,p}, pl_{i,p}, \lambda_{i,p}), l = 1, 2) \tag{5.5}$$

S.t.

$$\sum_{j=1}^{n+1}\sum_{k=1}^{K} x_{i,j,k} = 1, \quad \forall i \in \mathcal{O}, i \neq j \tag{5.6}$$

$$\sum_{i=0}^{n}\sum_{k=1}^{K} x_{i,j,k} = 1, \quad \forall j \in \mathcal{O}, i \neq j \tag{5.7}$$

$$\sum_{j=1}^{n+1} x_{0,j,k} = 1, \quad \forall k \in \mathcal{K} \tag{5.8}$$

$$\sum_{i=0}^{n} x_{i,n+1,k} = 1, \quad \forall k \in \mathcal{K} \tag{5.9}$$

$$\sum_{i=0}^{n} x_{i,j,k} = \sum_{i=1}^{n+1} x_{j,i,k}, \quad \forall j \in \mathcal{O}, \forall k \in \mathcal{K} \tag{5.10}$$

$$\theta_{i,k} + d_i + t_{i,j} \leq \theta_{j,k} + B(1 - x_{i,j,k}), \quad \forall i \in \mathcal{V}^o, \forall j \in \mathcal{V}^d, \forall k \in \mathcal{K}, i \neq j \tag{5.11}$$

$$a_i \leq \theta_{i,k} \leq b_i + p_{i,k}, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{K} \tag{5.12}$$

$$e_p - pe_{i,p} \leq \theta_{i,k} + B(1 - \lambda_{i,p}), \quad \forall i \in \mathcal{O}, \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{5.13}$$

$$\theta_{i,k} + d_i \leq l_p + pl_{i,p} + B(1 - \lambda_{i,p}), \quad \forall i \in \mathcal{O}, \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{5.14}$$

$$\sum_{p=1}^{P} \lambda_{i,p} = 1, \quad \forall i \in \mathcal{O} \tag{5.15}$$

$$\theta_{i,k}, p_{i,k}, \rho_{i,k}, pe_{i,p}, pl_{i,p} \geq 0, \quad x_{i,j,k}, \lambda_{ip} \in \{0,1\} \tag{5.16}$$

The objective functions

$$f_1 = \sum_{i=0}^{n} \sum_{j=1}^{n+1} \sum_{k=1}^{K} c_{i,j} x_{i,j,k} + \sum_{i=0}^{n+1} \sum_{k=1}^{K} c \times p_{i,k} + \sum_{i=1}^{n} \sum_{p=1}^{P} (min(d_i, pe_{i,p}) + min(d_i, pl_{i,p})) \tag{5.17}$$

$$f_2 = \sum_{i=1}^{n} \sum_{k=1}^{K} CM_{i,k}(\theta_{i,k} - \rho_{i,k}) + \sum_{i=0}^{n+1} \sum_{k=1}^{K} c \times p_{i,k} + \sum_{i=1}^{n} \sum_{p=1}^{P} (min(d_i, pe_{i,p}) + min(d_i, pl_{i,p})) \tag{5.18}$$

Where:

$$CM_i(\theta_{i,k} - \rho_{i,k}) = \frac{Cpm_i(1 - F_i(\theta_{i,k} - \rho_{i,k})) + (Ccm_i + W_i \times Cw_i) F_i(\theta_{i,k} - \rho_{i,k})}{(\theta_{i,k} - \rho_{i,k} + Tpm_i)(1 - F_i(\theta_{i,k} - \rho_{i,k})) + (\theta_{i,k} - \rho_{i,k} + Tcm_i) F_i(\theta_{i,k} - \rho_{i,k})}, \quad \forall i \in \mathcal{O}, \forall k \in \mathcal{K} \tag{5.19}$$

With:

$$M_i(\theta_{i,k} - \rho_{i,k}) = \int_0^{\theta_{i,k}-\rho_{i,k}} \frac{t f_i(t)}{F_i(\theta_{i,k} - \rho_{i,k})} \mathrm{d}t, \quad \forall i \in \mathcal{O}, \forall k \in \mathcal{K} \tag{5.20}$$

$$F_i(\theta_{i,k} - \rho_{i,k}) = 1 - e^{-((\theta_{i,k}-\rho_{i,k})/\sigma_i)^{\beta_i}}, \quad \forall i \in \mathcal{O}, \forall k \in \mathcal{K} \tag{5.21}$$

$$W_i = \theta_{i,k} - \rho_{ik} - M_i(\theta_{i,k} - \rho_{i,k}), \quad \forall i \in \mathcal{O}, \forall k \in \mathcal{K} \tag{5.22}$$

The objective function f_1 (5.17) minimizes the total travel cost related to technicians routing, the production losses incurred when not using a production stoppage interval, and the penalty's cost of violating the upper bound of PM operations time windows. Likewise, the objective function f_2 (5.18) minimizes the total expected maintenance cost, the production losses, and the penalty cost of not respecting the PM time windows. Constraints (5.6, 5.7) indicate that each operation must be performed only once. Constraints (5.8,5.9,5.10) ensure that each team of technicians has to leave the departure node 0; if a team of technicians arrives on a vertex i to do a maintenance operation, it should leave it; and finally, teams of technicians must arrive after executing the last operation in the tour at the arrival node $n+1$. The purpose of the constraints (5.11) is to prevent the creation of sub-tours. Constraints (5.12) assure that each operation is performed within its time window. Note that we assume that $a_0 = 0$ and b_{n+1} represents the latest time for technicians to return to the arrival depot. The violation of the upper bound of the time window is allowed. The amount of violation is measured in the variable $p_{i,k}$. Constraints (5.13, 5.14) favor the execution of a PM operation within one of the production stoppage time windows. The earliness penalty $pe_{i,p}$ and tardiness penalty $pl_{i,p}$ related to production losses are calculated if PM operations are fully or partially performed outside of production stoppages intervals. Constraints (5.15) ensure that one appropriate production stoppage interval is selected for each maintenance operation. Finally, the constraints (5.16) specify the domain values of decision variables.

Figure 5.2 shows how a production stoppage is selected based on penalties. The production stoppages are distributed over the horizon and a possible choice between the two nearest production stoppage time windows has to be made. The production stoppage that will be chosen for operation i is the one that has the less penalty term. If we suppose that both penalties are inferior to the duration $pe_{i1} < d_i$ and $pl_{i0} < d_i$, we will have $min(d_i, pe_{i1}) = pe_{i1}$ and $min(d_i, pl_{i0}) = pl_{i0}$. Since $pe_{i1} > pl_{i0}$, the production stoppage $p = 0$ will be chosen.

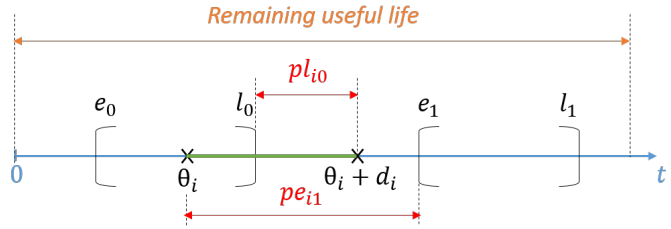

Figure 5.2: The production stop time windows over the horizon

Figure 5.3 shows some possible cases that can occur during the emplacement of the start times and the duration of the operations. In addition, it illustrates clearly the expression used in the objective function for the production stops.

5.4.4 The novelty of the proposed model

Several models have been proposed for the maintenance scheduling and workforce routing problem. Some integrate technical maintenance considerations, but they remain scarce compared to studies about the workforce routing problem. Most studies deal with preventive maintenance scheduling, and only a few include corrective maintenance [82, 84]. Corrective maintenance is rarely considered, although it is crucial and influences the final output of the model considerably. These aspects have been treated in the model of the previous chapters 3, 4 and in our work [98], [99]. Therefore, all the previous novel elements are also included in this chapter's model. Furthermore, we noticed that integrating other maintenance strategies adopted nowadays is rare. In this model, we integrated an opportunistic maintenance strategy with time-based preventive maintenance

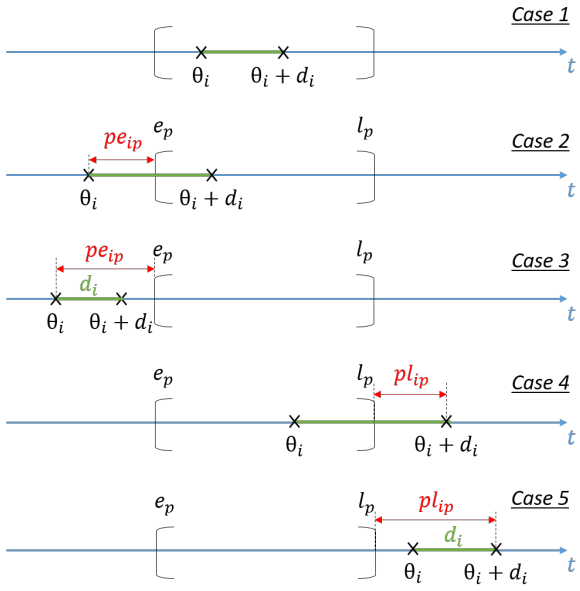

Figure 5.3: Possible cases of the difference between the start times and the production stop time windows

and corrective maintenance. The purpose is to jointly determine the routing and maintenance plan while penalizing the more minor production activities of the customers' service provider. To the best of our knowledge, no paper surveyed or modeled a similar problem. Therefore, the first novelty of this model compared to Chapter 3 and Chapter 4 is the integration of opportunistic maintenance within the problem. The VRP with multiple time windows (VRPMTW) is an essential variant of the VRP rarely studied in the literature [74] [75]. Unlike this variant, we dispose of a unique time window interval per operation to visit. The multiple time windows are instead considered for production stoppages. There are, therefore, two types of intervals, which make the problem far more different than the classical VRPTW and VRPMTW. The latter characteristic is the second novelty of the model. In addition, another main difference from VRPMTW is that these production intervals can or not be considered. We use multiple time windows of production stoppages that need to be favored to reduce the negative impact on production. The third novelty of the model lies within this last aspect. The routing or maintenance objective of the model integrates both a penalty term such as in [98], [99] and Chapter 3 to force the respect of the maintenance

intervals. The last novelty of the model is the inclusion in the objectives of a third term related to production losses incurred whenever the production stoppages available on the horizon are not selected for maintenance operations. During these production stoppages, production tools are stopped, and maintenance operations have no impact on production activities and the company's profit. These time windows are soft, meaning that the maintenance operations can be realized entirely, partially, or not at all in these intervals. In each case, the loss resulting from stopping the machines in their work period is considered. The start times need to coincide with the optimal times obtained by the maintenance model or be at least within the maintenance intervals. At the same time, a start time needs to be determined in the maintenance interval so that the duration of the maintenance operation is placed within one of the production stoppage intervals.

5.5 Proposed solution approach

The problem addressed is a Mixed Integer Program (MIP). Commercial solvers can not solve large-sized instances in a reasonable computational time. VRPTW and maintenance scheduling problems using OR techniques are NP-hard [86]. Adding production features to the objectives and constraints adds more complexity to the problem. As a result, it is more difficult to solve, and its NP-hardness remains. In the rest of the section, we propose a solution approach composed of a heuristic that deals with the optimal choice of start times and the selection of the best opportunity to execute maintenance. We then propose a Semi Adaptive Large Neighborhood Search (SALNS). Large Neighborhood Search (LNS) was initially proposed by Shaw [120] to deal with a neighborhood of exponential size. Searching in a large neighborhood may lead to better solutions [120]. The particularity of this method is that the neighborhood is sampled rather than explored, and the search is defined by destroy and repair operators [120]. It uses one destroy and one repair procedure throughout all iterations of the search [120]. This leads to the following drawbacks: it is difficult to know which destroy and repair operators are suitable for a given instance in advance. Moreover, some operators may be suitable for the first iterations while others are in later iterations. Røpke

and Pisinger [113] proposed ALNS with several destroy and repair methods for the first time to solve the Pickup and Delivery Problem. They then proposed some adaptations of the ALNS for several variants of the VRP problem [115]. ALNS can effectively deal with large neighborhoods and tightly constrained problems. Moreover, it dynamically selects destroy and repair operators based on their performance history to balance intensification and diversification in the search process. Finally, the selected operators are applied to improve the current solution.

5.5.1 Solution representation, local search phase, and constraints handling

A solution is represented with K tours where each tour is a sequence of PM operations. Only feasible solutions are accepted. The constraints are hard apart from verifying the upper bounds of the maintenance time windows and respecting both the upper and lower bounds of the production stoppages time windows. We use classical VRP moves to improve the current solution. Preliminary experiments conducted in [98], [99] and Chapter 3 showed that it is more effective to explore the neighborhoods in the following order: swap, insert, 2-opt*, and 2-opt. Approximately, the same order is used for VRP in the literature [105] [106]. The detailed description of the moves can be found in [98], [99] and Chapter 3. The Figure 5.4 represents the evaluation of the optimal solution obtained, for one instance. In this example, we have six machines. Each machine can have several operations. The start times are computed with the SALNS algorithm and the heuristic that considers the number=23 of interval's discretization. The start times for operations, as shown, can be in the beginning, middle, or after the time windows considered depending on the combination that minimizes the cost evaluated. Both the heuristic and the SLANS algorithm are presented below. The renewal time $\rho_{i,k}$ for all operations is illustrated.

Routing cost	Maintenance cost	Earliness penalty	Tardiness penalty
384.961	120.244	15.511	15.9302

Route 1	Operation	10	12	11
	Start time	50.0732	59.489	95.7673
	Renewal time	0	0	50.0732
	Time window	[45.5967,52.4607]	[58.3434,67.1263]	[95.7673,102.631]

Route 2	Operation	5	3	6	7	4
	Start time	29.4947	50.1716	62.657	95.8194	97.5972
	Renewal time	0	0	29.4947	62.657	50.1716
	Time window	[29.4947,33.9347]	[44.190,50.749]	[62.657,67.0971]	[95.8194,100.259]	[92.7394,99.3795]

Route 3	Operation	8	9
	Start time	41.7832	87.8334
	Renewal time	0	41.7832
	Time window	[41.7832,48.0732]	[87.8384,94.1234]

Route 4	Operation	1	2
	Start time	36.2538	76.5413
	Renewal time	0	36.2538
	Time window	[36.2538,41.7171]	[76.5413,81.9996]

Figure 5.4: Representation of the solution for the illustrative instance with 12 operations, Re/6/4/101/101/0.07/2.

5.5.2 Heuristic for determining the best start time with multiple production stoppages time windows constraints

A solution s is a set of K tours. Each tour is represented with an ordered list of PM operations. In order to evaluate the solution s, we should set the start time of each PM operation while respecting the order of operations and minimizing the value of the objective function. We propose a dedicated heuristic to evaluate a given solution called Choice of the Start Time with Multiple Production Stoppages Time Windows CSTMPSTW. It finds the best start times in the maintenance intervals to perform PM operations. Furthermore, it selects the most convenient production stoppage time window among the multiple production stoppages time windows available. First, the heuristic calculates the start time of operations considering the sequence order and the maintenance time windows. This continuous start time is then translated into the maintenance interval to have as much as possible the duration of a maintenance operation during one of the production stoppages. The aim is to totally or partially include the maintenance operation in one of the production stop intervals that reduce the production losses cost the least.

We recall that the optimal execution time of each PM operation is obtained by solving the maintenance cost model. This latter is then used to obtain the maintenance time windows used in the routing. It is explained in the thesis work in [98], [99] and Chapter 3. These optimal execution times change when considering the routing constraints. Moreover, it undergoes more changes when production stoppages intervals are considered since they guide start time translation in the maintenance intervals. This translation is, however, limited in the case of the flexible upper bound time windows.

Algorithm 12 describes the proposed heuristic. For each tour and each operation in this tour, an operation start time is computed in the line 4. With this initial start time, we suppose that each technician team starts the maintenance operations as soon as it arrives. Two cases are then distinguished. In the first case, a penalty is incurred because of the non-respect for the maintenance interval's upper bound. This penalty is computed in lines (6-7) and then the production stoppage that reduces the production losses cost is selected. The selection of the most appropriate production stoppage is detailed in lines (8-22). In the second case, the start time of the operation is translated into the remaining part of its maintenance time window to find a new start time that reduces the production losses. We test a reduced number of candidates points in the interval $[\theta_{j,k}, b_j]$. This is realized using discretization of the maintenance interval in steps (23- 35). For each tested point in the maintenance interval, we select a most convenient production stoppage using the same procedure detailed in steps (8-22). The best coordinate $\theta_{j,k} i \in \mathcal{O}, k \in \mathcal{K}$ is the one that minimizes the total production losses cost. The aim of choosing the best start times using discretization is to fully consider the length of the maintenance time window among the possibilities that could reduce the total production losses cost. This discretization is only applied when the initial start time does not violate the upper bound of the maintenance time window as we assume that for the steps (5-22) there is no need to translate more outside the interval to not increase the penalty cost of violating the upper bound of the time windows. This strategy reduces computational time.

At the end of the heuristic, we obtain the most convenient production stop and the best start time $\theta_{j,k}$ for each operation in the current solution.

The step-by-step procedure of $CSTMPSTW$ can be summarized as follows.

For each tour, for each operation in the tour: $\theta_{j,k}$=max($\theta_{i,k}$+d_j+$t_{i,j}$, a_i),
If the maintenance time window is not respected ($\theta_{j,k} > b_j$)**:**

- Calculate the penalty for not respecting the maintenance interval,
- Calculate the production losses penalties for each production stoppage time window p,
- Choose the production stoppage time window p which minimizes the cost of total or partial non-respect of the placement of the maintenance operation in this window,

If the maintenance time window is met ($a_j <= \theta_{j,k} <= b_j$)**:**

- Discrete the interval $[\theta_{j,k}, b_j]$ in $number + 1$ points,
- Compute, for each point in the interval, the production losses penalties for each production time window p,
- Choose, for each point, the production stoppage time window p that minimizes the cost of total or partial non-respect of placing the maintenance operation in that window,
- Choose the point $\theta_{j,k}$ which minimizes the cost of the production losses,

5.5.3 ALNS operators

Destroy operators

They destroy part of the solution by a predefined degree of destruction d. ALNS has a high degree of destruction (40 percent of the solution). Removed tasks are saved in a vector of unrouted operations during the running. In our case, this degree is equal to 40 percent of the total number of operations n.

Highest cost removal operator removes d operations with the highest cost. The operator aims to remove tasks that worsen the objective function more so that in the recreating phase, the repair operators can reinsert them in other positions to reduce the huge costs. For the routing objective function, we calculate the

Algorithm 12 : Heuristic for the choice of the best start time with multiple production stoppages time windows constraints ($CSTMPSTW$).

Data : $[e_1, l_1], ..., [e_p, l_p]$: a set of production stoppages ;
$[a_i, b_i]$: maintenance time windows for each $i \in \mathcal{O}$;
d_j: the duration of each operation j ;
$t_{i,j}$: travel time between operations i and j;
s: a solution ;
Result : $pe_{j,p}, pl_{j,p}$: The earliness and the tardiness penalties related to PM operation execution on the production stoppage p ;
$\theta_{j,k}$: the best start time of each operation j in a tour k;
$sumPenalty$: total maintenance penalty costs;
$totalProductionLosses$: total productions losses ;

Initialize vectors, productionLossesChosen, indexTWChosen, productionLossesChosenCoordinate and indexTWChosenCoordinate to 0 ;
foreach $k \in s.Tours$ **do**
  foreach $j \in s.Tours[k].Operations$ **do**
    $\theta_{jk} \leftarrow max(\theta_{ik}+d_j+t_{ij}, a_i), j = successor(i)$;
    if $\theta_{j,k} > b_j$ **then**
      $p_{jk} \leftarrow \theta_{jk} - b_j$;
      $sumPenalty \leftarrow sumPenalty + c \times p_{j,k}$;
      Initialize vectors, tardinessPenalty, earlinessPenalty, sumProductionLosses to 0 ;
      foreach $p \in \mathcal{P}$ **do**
        if $\theta_{j,k} + d_j > l_p$ **then**
          $pl_{j,p} \leftarrow \theta_{j,k} + d_j - l_p$;
          $tardinessPenalty[p] \leftarrow min(pl_{j,p}, d_j)$;
        end
        if $\theta_{j,k} < e_p$ **then**
          $pe_{j,p} \leftarrow e_p - \theta_{j,k}$;
          $earlinessPenalty[p] \leftarrow min(pe_{j,p}, d_j)$;
        end
        $sumProductionLosses[p] = tardinessPenalty[p] + earlinessPenalty[p]$;
      end
      /* The index of the best selected production stoppage and its corresponding production losses */ ;
      $indexTWChosen[j] \leftarrow \left\{ p \in \mathcal{P}, p = \underset{p \in \mathcal{P}}{\text{argmin}}(sumProductionLosses) \right\}$;
      $productionLossesChosen[j] \leftarrow \underset{p \in \mathcal{P}}{\text{argmin}}(sumProductionLosses)$
    else
      /* Interval discretization $[\theta_{j,k}; b_j]$ */;
      $step \leftarrow \frac{b_j - \theta_{j,k}}{number}$;
      Initialize discreteCoordinate to 0 ;
      $discreteCoordinate[0] \leftarrow \theta_{j,k}$;
      for $c = 1, c \leq number, c++$ **do**
        $discreteCoordinate[c] \leftarrow discreteCoordinate[c-1] + step$;
      end
      for $c = 1, c \leq number, c++$ **do**
        Realize the steps (8- 22) to select $indexTWChosenCoordinate[j][c]$ and to compute $productionLossesChosenCoordinate[j][c]$
      end
      $productionLossesChosen[j] \leftarrow \underset{c \in \{1,2,...,number\}}{\text{argmin}} (productionLossesChosenCoordinate[j])$;
      $\theta_{j,k} \leftarrow \left\{ discreteCoordinate[c], c \in \{1,2,...,number\}, \underset{c \in \{1,2,...,number\}}{\text{argmin}} (productionLossesChosenCoordinate[j]) \right\}$;
    end
  end
end
/* sum of the elements of the table productionLossesChosen*/
$totalProductionLosses \leftarrow \underset{j \in \mathcal{O}}{sum}(productionLossesChosen)$;

difference between the cost when the task is included in the solution and the cost when it is removed. Precisely, for each task k located between task i and task j, the saving value $c_{ik} + c_{kj} - c_{ij}$ is calculated. For the maintenance objective, the cost of each operation is obtained using the start times. The penalties for both objectives are incurred when maintenance intervals are not respected and when the production stoppages intervals are not efficiently used. The array of unrouted operations is sorted in increasing order of the total cost values, and the d last tasks are removed from the solution.

Highest Risk removal operator removes a predefined number of tasks with the highest risk of breakdown. Indeed, it removes tasks with the highest probability of failure. This new proposed operator is based on the probability of failure designed for the failure cost objective we proposed in [99], [98], and Chapter 3 and performs very well for the three objectives.

Related removal is based on similarity. It has been first proposed by Shaw [127]. It consists of removing similar parts so that the repair operator reinserts them in a better way. The reason is that similar parts are easy to interchange. Therefore, the lower $R(i,j)$, the more related are the two tasks. The relatedness measure used here comprises two terms: a time term and a travel time (or distance term). Therefore, we defined relatedness as follows:

$$R(i,j) = \alpha \left|\theta_{ik} - \theta_{jk}\right| + \beta t_{ij}, \quad \forall i,j \in \mathcal{O}\, \forall k \in \mathcal{K} \tag{5.23}$$

Where (α,β)=(0.5,0.5), θ_{ik} is the start time of operation i performed by the team of technicians k, and t_{ij} is the travel time between i and j.

Time oriented removal is another variant of related removal that integrates only the time part. Operations that are served at approximately the same time are removed. It is similar to the previous operator with (α,β)=(1,0).

Distance oriented removal is another variant of related removal that integrates only the distance part. Operations that are geographically close to the chosen request are removed. It is the related removal operator with (α,β)=(0,1).

Repair Heuristics

They insert the operations that were destroyed. They are divided into two categories for VRP problems. Sequential heuristics construct one route at a time, while parallel heuristics construct several routes simultaneously. The heuristics presented hereafter are parallel.

Basic greedy insertion calculates the minimum insertion cost for every solution's operation that must be inserted. First, we determine the insertion cost as the difference between the total cost of the solution without the inserted maintenance operation and the cost of the solution with the inserted operation. Then, we insert the one with the minimal cost difference until there are no operations to be inserted.

Best insertion is the same as the basic greedy insertion procedure but improves it by considering the order of insertion of the unrouted operations. It consists, therefore, of inserting the operations in their best positions considering all operations of all machines and their order of insertion.

Two regret insertion inserts maintenance operations based on the regret value instead of the cost value. The regret value can be obtained by calculating the difference in total cost between the best insertion position and the second best. After ordering tasks, those with a high regret value are inserted first. For example, among a set of unrouted tasks $\mathcal{U}$, the heuristic inserts a task i such that:

$$i = argmax(\Delta f_i^2 - \Delta f_i^1), \quad \forall i \in \mathcal{U} \tag{5.24}$$

Differently from Ropke and Pisinger [115] and similarly to [125], the regret value is calculated based on all different positions that can be in the same route or different routes. Ropke and Pisinger [115] considered only positions in different routes. When a task is inserted, the insertion positions of the remaining unrouted tasks are recalculated by considering the change caused by inserting this task at this position.

5.5.4 Update of weights

The following definitions are needed to describe the weight adjustment. The weight w_h denotes the weight of either a destroy and repair heuristic h. The weights update mechanism is similar for all the destroy and repair operators. Furthermore, the number of times the heuristic h has been used during the iterations is called u_h. The success of the heuristic h is denoted s_h. Both s_h and u_h are initialized to zero at the beginning. After using an operator h in an iteration, s_h is increased by δ_i with $i = 1, 2, 3$ according to the following cases related to solution quality:

- δ_1: the new solution is the best one found so far.
- δ_2: the new solution improves the current solution.
- δ_3: the new solution does not improve the current solution, but is accepted.

Moreover, for reasonable adjustments we have to ensure the inequality $\delta_1 > \delta_2 > \delta_3$. It is not, however, a general rule. Ropke and Pisinger have chosen $\delta_1 > \delta_3 > \delta_2$ in [113]. Finally, the reaction factor $0 \leq \rho \leq 1$ controls the influence of the recent success of a heuristic on its weight. Therefore, we calculate the weights for the next iterations by:

$$w_h = \begin{cases} (1-\rho)w_h + \rho * \frac{s_h}{u_h} & \text{if } u_h > 0 \\ (1-\rho)w_h & \text{if } u_h = 0 \end{cases} \tag{5.25}$$

As a result, the parameter ρ is decisive for the weight adjustment. In the case of $\rho = 0$, the weights remain constant at their initial level, and the probabilities never change. If $\rho = 1$, only the recent success is accounted for in the heuristic's selection. Typically, the recent success and the heuristic's performance before the last iteration are relevant and should be considered, which means $0 < \rho < 1$. Therefore, we have chosen $\rho = 0.7$ as it is usually done in the literature.

The operator h can be either a repair r or a destroy operator d. To compute the probabilities we use the following formulas:

$$p_r = \frac{w_r}{\sum_{r=1}^{R} w_r}, \quad p_d = \frac{w_d}{\sum_{d=1}^{D} w_d} \tag{5.26}$$

5.5.5 Acceptance criterion

The main advantage of metaheuristics compared to local search is allowing the search to move to worse solutions to introduce some diversification and avoid the local optima trap. The Simulated Annealing metaheuristic uses a strategy inspired by the physical annealing process. Better solutions are accepted but also worse solutions with a certain probability. The solution s' is accepted even if it does not improve the current solution s to ensure diversification according to the probability $exp(-(\frac{(f(s')-f(s))}{T}))$. The temperature T also decreases with a factor α at each iteration, ensuring that the probability of accepting worse solutions decreases as well when the algorithm is executed, ensuring that late in the process worse solutions are less likely to be accepted. At the beginning, T is initialized with the value of the total cost $f(s_0)$ of the initial solution s_0 as follows:

$$T = \frac{w_T}{-ln(0.5)} \times f_i(s_0) \tag{5.27}$$

Similarly to Ropke and Pisinger [113], we set $w_T = 0.05$ so that a solution that is 5% percent worse than $f_i(s_0)$ is accepted with a probability of 0.5. The cooling factor of the temperature T is fixed to $\alpha = 0.99975$.

5.5.6 Proposed Semi-Adaptive Large Neighborhood Search (SALNS)

The framework of the proposed semi-ALNS approach is illustrated in Algorithm 13. The same framework has been proposed in our previous work [66] to solve the joint maintenance scheduling and routing problem presented in our thesis work [98] and Chapter 3. The SALNS algorithm starts with an initial solution constructed using the Best Insertion Heuristic. Operators are selected using a semi-adaptive mechanism (steps 11-15) that consists of using the roulette wheel to select operators according to their past success but switching to a random selection as soon as there is no improvement of the best solution. This random selection assures diversification in addition to applying the simulated annealing criterion. The diversification time is the opportunity for the learning

mechanism to have enough time to learn from random selection in the environment. The heuristics' probabilities are updated at each iteration by the new information revealed regarding the heuristics' contribution to the improvement of the current solution and the best solution. Learning occurs each iteration, but the selection according to the probabilities happens only if the new obtained solution improves the best solution found so far in the neighborhoods. The proposed mechanism seeks to alternate between diversification and learning. A destroy operator $d \in D$ removes operations from the solution that are afterward reinserted using a repair operator $r \in R$. The local search procedure VND is then applied to the new solution yielded s^{dr} (Step 17). Three cases are distinguished (Steps 20-38). If the resulting solution s' improves the best solution found so far, it replaces s_{best} and the current solution s. The score is increased by δ_1, and $counterI$ is set to 1. The variable $booleanI$ indicates if the best solution is improved.

If the new solution s' improves only the current solution s but is worse than s_{best}, it replaces it and the score is increased by an amount $\delta_2 < \delta_1$. If the solution yielded s' is worse than to s but satisfies the simulated annealing acceptance criteria, the resulting solution s' replaces the current solution s and the score is increased by an amount $\delta_3 < \delta_2 < \delta_1$. In these two latter cases, $counterI$ is set to 0 to specify that there is no improvement of the best solution. In each iteration, temperature and probabilities are updated.

5.5.7 Novelty of the proposed solution approach

Novelty of the proposed heuristic

A heuristic is proposed to select the adequate production stoppage and determine the best maintenance operation start times. It is detailed in Section 5.5.2. This heuristic deals with, therefore, two aspects. In VRPTW's literature, the start times can be placed anywhere in the interval but with respect to the values obtained using constraints (5.11) and (5.12). This procedure has been explained in detail in the work of Kallehauge et al. [38] and in [76]. The problems featured in these papers deal with VRP with usual objectives.

This procedure can not be applied in our convex maintenance objective

Algorithm 13 : Semi Adaptive Large Neighborhood Search (SALNS).

Data : l_{max}: number of neighborhood structures in VND ;
R : set of repair operators D: set of remove operators $iter_{max}$: maximum number of Iterations α : cool parameter
Result : s_{best} : best solution found

$s_0 \leftarrow bestInsertionHeuristic()$;
$s \leftarrow s_0$;
$s_{best} \leftarrow s_0$;
$T \leftarrow \frac{-w_T}{ln(0.5)} \times s_0$;
$s^{dr} \leftarrow s_0$;
$u_d, u_r \leftarrow 0$;
$w_r, w_d \leftarrow initializeWeights()$;
$p_r, p_d \leftarrow initializeProbabilities(w_r, w_d)$;
$counterI \leftarrow 0$;
repeat
  if $counterI = 0$ **then**
    $r, d \leftarrow selectRandomOperators(R, D)$;
  else
    $r, d \leftarrow selectOperatorsWithRouletteWheel(R, D)$
  end
  $s^{dr} \leftarrow r(d(s))$;
  $s^{'} \leftarrow VND(s^{dr}, l_{max})$;
  $u_d \leftarrow u_d + 1$;
  $u_r \leftarrow u_r + 1$;
  $random \sim U_c[0, 1]$;
  if $f(s') < f(s_{best})$ **then**
    $s_{best} \leftarrow s'$;
    $s \leftarrow s'$;
    $s_d, s_r \leftarrow s_d, s_r + \delta_1$;
    $counterI \leftarrow 1$;
  else
    if $f(s') < f(s)$ **then**
      $s \leftarrow s'$;
      $s_d, s_r \leftarrow s_d, s_r + \delta_2$;
      $counterI \leftarrow 0$;
    else
      if $random < exp(-(\frac{(f(s')-f(s))}{T}))$ **then**
        $s \leftarrow s^{'}$;
        $s_d, s_r \leftarrow s_d, s_r + \delta_3$;
      end
      $counterI \leftarrow 0$;
    end
  end
  $T \leftarrow T \times \alpha$;
  $w_r, w_d \leftarrow adjustWeights(w_r, w_d)$;
  $p_r, p_d \leftarrow adjustProbabilities(w_r, w_d)$;
until *maximum number of iterations* $iter_{max}$;

since the best start time will be in the middle of the maintenance interval if the constraints permit it. The first and second objectives also include the last term to measure production losses incurred. This term forces the continuous start times to translate in the interval until finding the best value of that latter that satisfies the problem's constraints. The heuristic tests all the possibilities using the discretization of a chosen partition from the maintenance intervals for both objectives. We try, however, a reduced number of candidate points in the intervals, as explained in 5.5.2. Indeed, only the remaining feasible partition of the maintenance interval is tested.

Novelty of the proposed SALNS algorithm

The novelty of the proposed metaheuristic is illustrated in these aspects. The first novelty of our proposed ALNS is the semi-adaptive mechanism that effectively alternates between diversification and learning to select the operators used in the search. The second novelty of this algorithm is using a new removal operator based on the breakdown risks to cope with the peculiarities of the problem. The metaheuristic is very similar to the General Variable Neighborhood Search (GVNS) metaheuristic [88]. Steps 11 to 15 are considered as a guided shaking mechanism inspired from the GVNS metaheuristic. Using dedicated operators and selecting them according to their past success leads to better solutions and improve the computational time. Furthermore, the random selection of operators enhances learning with new information and, at the same time, ensures diversification while selecting according to past success reduces the computational time. Steps 23, 28 and 33 improves the results and is added similarly to the GVNS metaheuristic. This line was considered also by [78] [114]. The VND (line 17) is added to the classical ALNS algorithm in [77] [113] [115] and reinforces the similarity with the GVNS metaheuristic. The same local search procedure has also been used by [114].

5.6 Computational experiments

This section describes the methodology of instances' generation, the parameter values used in the proposed algorithm, the performance metrics used, and finally, provides test results for the problem using the solver and the proposed SALNS algorithm. Results are reported for small, medium, and large size instances. The maintenance model was implemented with Python 3.6 in a Linux environment to get the expected start time, frequency, and time windows for each PM operation. The combined maintenance and routing problem with production stoppages considerations was implemented with C++ and compiled with GCC7.4 in a Linux environment. The Concert Technology Library of CPLEX 12.10.0 version with default settings in C++ has been used to solve the MIP using the exact method. Experiments were conducted using the CALCULCO computing platform in an AMD EPYC 7702 with 2CPU, 2 gigahertz, and 1 core was dedicated to each instance. All methods are run using the same equipment to avoid bias.

5.6.1 Generation of instances

We generated two sets of instances to validate the model and evaluate our SALNS. The instances of the first set related to routing data are generated based on Solomon benchmark instances available on Solomon's web page `http://web.cba.neu.edu/~msolomon/problems.htm` as described in [82]. The first 10 and 25 customers of Solomon's classes of instances have been used to stand for the machines used. The customers' coordinates are considered as the machines' coordinates. Two horizon values are considered, H=100 and H=200. This leads to many PM operations that vary from 23 to 73 operations. The technician teams vary from 5 to 18 for this set of instances. The unitary speed is considered. The horizon is set to H=100 hours when the latest arrival time at the depot is b_{n+1}=200 hours and to H=200 hours when b_{n+1}=400 hours. Six Solomon's classes (C101, C201, R101, R201, RC101, RC201) are divided into three main types (R, C, RC), categorizing the geographical dispersion of machines. The types R, C and RC respectively stand for randomly distributed locations, geographically clustered locations, and finally, the mix of both. It is worth mentioning that

only instances until 10 machines, 3 vehicles, and H=125 have been studied [82]. Maintenance parameters were randomly generated as described in [82] according to the continuous uniform distribution U_c distributions as follows: $Tpm_i \sim U_c[5,10]$, $Tcm_i \sim U_c[15,30]$, $Cpm_i \sim U_c[100,200]$, $Ccm_i \sim U_c[400,800]$, $Cw_i \sim U_c[10,20]$ and $\sigma \sim t_{0i} * U_c[2,5]$. Only the parameter $\beta \sim U_c[2,6]$ has a larger interval compared to the one described in [82]. We consider that larger values of β are closer to reality. The shape parameter of the Weibull distribution $\beta > 1$ ensures that the machines are in the wear-out period of their life. Each instance name has the following format $Class/M/K/H/b_{n+1}/tol$. The class represents the geographic distribution of machine locations, M is the number of machines, and K is the available number of technicians' teams that can be reduced during the search. In addition, H is the horizon's length, b_{n+1} the latest arrival time at the arrival node, and tol represents the percentage of time windows tolerance. An optional argument is sometimes added at the end of the instance's name if the value of β is mentioned. The instance C101/25/18/100/200/0.07 is an instance based on the data of the first 25 customers of class C101. The 25 machines are located in the coordinates of Solomon's class C101. The initial number of technicians teams is 18, the horizon $H = 100$, $b_{n+1} = 200$, and $tol = 7\%$.

The second set of instances denoted by Re is proposed. Large and short maintenance service times as well as short and long distances are considered in this set to address reality. The number of machines considered is 6. The horizon equals H=101 hours for K=4 (n=12) and H=80 hours for K=2 (n=6). The speed equals 60km/h. Data about this set can be found in our work [98] and in 3.

For both sets, production parameters are obtained as follows. The number of production stops time windows P for all the instances corresponds to the number of days in the horizon H, assuming that the customer's factory stops within a period corresponding to one shift for each day. It is determined as follows: $P = E[H/24]$. For example, a horizon with $H = 100$ hours corresponds to approximately 4 days, then 4 production stoppages. Only for the specific instance Re with $H = 101$, 3 production stoppages are considered. The start of the first production stoppage time window is $e_p = \frac{H}{P}$. The following production stoppages are then obtained through the horizon. The duration of the production stoppage is one shift length $dp_p = 8$ hours. The opening time equals a_0=0. The

penalty cost per unit time for violating the maintenance intervals is set to c=10. The percentage of time windows tolerance is either narrow tol = 7% as adopted in heavy industries or large tol = 30%.

5.6.2 Parameter settings and performance metrics

Extensive computational experiments are conducted to set the parameter values and determine the most efficient destroy and repair operators. Since the ALNS algorithm is composed of many operators and each has its own parameters, parameter tuning was mainly performed considering the solution quality indicator. We have measured the contribution of each operator to solution quality. Preliminary experiments on a subset of representative dataset instances showed that the random and sequential operators that remove random tasks often worsen the solution quality. The highest cost removal slightly improves the quality of the solution. It is worth mentioning that the highest risk removal followed by the related removal is the most efficient operator, increasing the solution quality considerably. Furthermore, their removal causes more solution degradation. The time and distance-oriented removal improve the solution quality slightly. The best insertion heuristic is the most efficient repair operator, followed by the greedy best insertion and the regret heuristic. The regret heuristic makes the difference by improving the solution in only some instances. Using the combination chosen ensures a trade-off between good quality and the least CPU time.

The stopping condition for our SALNS algorithm is the maximum number of iterations $iter_{max} = max(1, E[7n/8])$ where the symbol E stands for the integer portion of the number. The degree of destruction for the SALNS is $d = E[0.4 \times n] + 1$. As mentioned above, $w_T = 0.05$ and the cooling rate $\alpha = 0.99975$. The values of the rewards are δ_1=50, $\delta_2 = 30$ and $\delta_3 = 20$. The weights of all the ruin and create operators are initialized at the beginning of the search to 20. Meanwhile, the scores s_h and the number of times the heuristics are used u_h are set to 0. The reactive factor ρ is important for the weight adjustment. We have set $\rho = 0.7$ as it is the most used value in the literature.

The MILP is tested on the CPLEX solver for the objective routing function.

The maximum time limit is fixed at 24 hours for up to 73 operations. The ample preset time aims to obtain at least one feasible solution as a comparison indicator with our SALNS algorithm. However, most of the time, the solver failed to achieve this goal. Therefore, in the CPLEX columns of Table 5.1, we report the objective value followed by whether the solution is optimal, feasible, or not found after 24 hours of execution and finally, the CPU time.

The SALNS algorithm integrates the heuristic to deal with the problem's constraints. We report the maximum, minimum, and average value in 5 runs and the corresponding maximum, minimum, and average CPU time. The pseudo-code of SALNS is shown in Algorithm 13.

Two key indicators have been used to evaluate the performance of SALNS:

- The average percent decrease of the minimum objective value obtained by the proposed SALNS metaheuristic in comparison with the CPLEX solution:

$$Gap(CPLEX, SALNS) = \left(\frac{Value(CPLEX) - Value(SALNS)}{Value(CPLEX)}\right) \times 100\% \tag{5.28}$$

- The percent decrease of the CPU time when using SALNS compared to the CPLEX solver:

$$ICPU(CPLEX, SALNS) = \left(\frac{Time(CPLEX) - Time(SALNS)}{Time(CPLEX)}\right) \times 100\% \tag{5.29}$$

5.6.3 Numerical test results

Tables 5.1 gives the comparison of our SALNS with a discretization of 23 of the maintenance's time windows. This value was retained since it provides the best values for the objective functions. For the small instances when the number of operations is less or equal to 12, SALNS performs well as all the instances achieve the optimum. No proven optima were obtained for the medium and large instances between 23 and 73 operations. In addition, CPLEX only returned feasible solutions for 10 instances among 14 medium-size instances for which

the number of operations is inferior to 34. For the remaining 26 instances with a number of operations between 52 to 73 (large instances), only 3 feasible solutions were returned, and the solver could not provide any other feasible solutions for the remaining 23 large instances. When the maintenance interval is segmented into 23 intervals (number=23), the improvement of the average objective in favor of SALNS is equal to 17.65%. Moreover, the proposed SALNS algorithm is better than CPLEX regarding running time. The minimum and the average CPU times of SALNS are better than CPLEX's CPU time by 65.94 % and 62.37 % respectively. If we only consider the generated set from Solomon (medium and large instances), the improvement of the average objective in favor of SALNS is equal to 21.73 %. The minimum and the average CPU times are improved by 89.36% and 88.27% respectively.

Table 5.2 displayed the results of the SALNS algorithm with small numbers of discretization 2 and 8 and compared them to CPLEX results. If number=2, only three points are tested in the maintenance interval, the lower bound, the middle, and the upper bound. To achieve a reduced objective value, and since the objective value is composed of both continuous and discrete terms, many possibilities in the maintenance interval have to be tested. We can notice that the improvement of the SALNS objective value increases quickly with the number of points tested in the maintenance interval since the objective value considerably decreases. The objective value decreases very fast between the number=2 and the numbers 8 and 23. There is no significant improvement in the objective function when the discretization number varies from 8 to 23. For the 16 instances for which CPLEX returned either an optimal or feasible solution, SALNS with number=2 improves the objective value compared to CPLEX up to 16.10 %. The minimum CPU time of SALNS outperforms CPLEX's time by 79.48% and the average CPU time by 77.41%. For medium and large-size instances derived from Solomon set, the improvement of the objective is equal to 19.81%. The minimum CPU time and the average CPU time are improved by 93.09% and 92.18% respectively. When number=8, the gap or percent decrease of objective value when using SALNS compared to CPLEX is 17.37%, for the minimum CPU time is 74.30%, and for the average CPU time is 71.77%. If only medium and large-size instances are considered (excluding the 3 instances Re), the improvement of the

objective for all remaining instances in favor of SALNS is equal to 21.37 %. The minimum and the average CPU times are improved by respectively 92.34% and 91.67%. The computational time of the SALNS increases with the instance size. It also increases when the number of segmentation of the maintenance interval is significant during the evaluation of a solution. However, in all cases, it only represents a small fraction of the CPU time consumed by the commercial solver CPLEX.

Table 5.3 gives the results for all the instances when the evaluation heuristic used in SALNS uses discretizations with values for the parameter *number* of 2 and 8 for the maintenance objective. The improvement of the objective value on average slowly increases with the number of intervals. For example, the improvement of the objective value in average when using number=8 in the segmentation of maintenance intervals instead of number=2 is 0.81 %. The CPU time grows when these numbers grow as the improvement of the minimum and average CPU times are negative and respectively equal -22.44 % and -23.35 %.

5.7 Computational results for the maintenance scheduling and workforce routing problem

This section presents the results of the SALNS algorithm to solve the problem defined in Chapter 3. Computational experiments are presented hereafter. The algorithms have been implemented in C++ compiled with GCC 7.4 in a Linux environment. The solver used is CPLEX 12.10.0. Tests were executed using the CALCULCO computing platform.

5.7.1 Parameters settings and performance metrics

Instances were generated as described in our work [98] and Chapter 3. In addition to Solomon's class C, we have included the remaining classes, R and RC. We run the tests for the problem with the routing objective described in [98], [99] and Chapter 3. The number of vehicles here is minimized during the search. Extensive experiments have been done to decide which and how many destroy and repair operators to use. The random and sequential removals

Instance	*n*	CPLEX			ALNS number=23						Improvement (%)		
		Objective	Status	CPU	max	min	avg	max cpu	min cpu	avg cpu	Gap	ICPU min	ICPU avg
RE/6/2/80/80/0.30/2	6	115.946	optimal	0.13	115.946	115.946	115.946	0.28	0.24	0.262	0.00	-84.62	-101.54
RE/6/2/80/80/0.07/2	6	115.946	optimal	0.13	115.946	115.946	115.946	0.29	0.24	0.262	0.00	-84.62	-101.54
RE/6/4/101/101/0.07/2	12	384.961	optimal	15.32	394.081	384.961	390.433	8.21	5.73	7.114	0.00	62.60	53.56
*C*101/10/8/100/200/0.07/2	23	241.943	Feasible	86391.3	243.667	243.667	243.667	288.26	208.96	252.028	-0.71	99.76	99.71
*C*101/10/7/100/200/0.07	26	210.732	Feasible	86397.4	210.732	210.732	210.732	534.87	432.44	499.234	0.00	99.50	99.42
*C*101/25/18/100/200/0.07	54	-	-	-	461.004	459.944	460.18	13441.1	11324.1	12507	-	-	-
*C*201/10/7/100/200/0.07	24	198.957	Feasible	86257	198.957	198.957	198.957	196.4	180.53	188.738	0.00	99.79	99.78
*C*201/25/18/100/200/0.07	52	-	-	-	513.907	513.907	513.907	12158.7	10531.2	11133.3	-	-	-
*R*101/10/7/100/200/0.07	34	978.253	Feasible	86372.4	924.793	860.626	879.82	1798.85	1558.5	1670.07	12.02	98.20	98.07
*R*101/25/18/100/200/0.07	68	-	-	-	803.366	770.794	789.58	53802.2	48964.3	51583.2	-	-	-
*R*201/10/7/100/200/0.07	34	705.883	Feasible	86305.2	652.662	644.133	650.758	1949.09	1607.39	1798.09	8.75	98.14	97.92
*R*201/25/18/100/200/0.07	65	-	-	-	627.759	627.547	627.717	45909.2	37381.9	39954.3	-	-	-
*RC*101/10/7/100/200/0.07	33	931.284	Feasible	86352.5	1080.71	1042.89	1066.76	1632.4	1260.49	1444.96	-11.98	98.54	98.33
*RC*101/25/18/100/200/0.07	62	-	-	-	618.836	598.923	602.906	27617.2	24793.9	26053.7	-	-	-
*RC*201/10/7/100/200/0.07	33	1965.339	Feasible	86343.5	1026.03	1006.15	1010.13	1440.49	1110.89	1325.86	-4.23	98.71	98.46
*RC*201/25/18/100/200/0.07	60	-	-	-	599.545	594.432	596.588	28479.1	21124.2	24824.2			
*C*101/10/8/100/200/0.30/2	23	186.897	Feasible	86279.3	187.589	187.589	187.589	326.59	239.24	298.012	-0.37	99.72	99.65
*C*101/10/5/100/200/0.30/2	23	-	-	-	310.751	310.485	310.538	404.63	320.38	351.27	-	-	-
*C*101/10/7/100/200/0.30	26	159.555	Feasible	86338.3	166.986	166.986	166.986	556.94	361.42	440.198	-4.66	99.58	99.49
*C*101/25/18/100/200/0.30	54	-	-	-	347.711	347.711	347.711	16538.8	14209.2	15314.4	-	-	-
*C*201/10/7/100/200/0.30	24	135.563	Feasible	86270.5	141.527	141.527	141.527	283.75	256.87	267.158	-4.40	99.70	99.69
*C*201/25/18/100/200/0.30	52	-	-	-	346.899	346.899	346.899	11415.5	9121.84	10068.2	-	-	-
*R*101/25/18/100/200/0.30	68	120380	Feasible	86359.8	519.816	509.722	514.237	76899.1	62604.8	70801.7	99.58	27.51	18.02
*R*201/10/7/100/200/0.30	34	-	-	-	424.657	412.128	419.801	2073.16	1587.26	1912.16	-	-	-
*R*201/25/18/100/200/0.30	65	-	-	-	408.217	408.217	408.217	41208.7	30520.4	33986.3	-	-	-
*RC*101/10/7/100/200/0.30	33	-	-	-	738.668	719.319	729.517	1918.62	1423.33	1610.91	-	-	-
*RC*101/25/18/100/200/0.30	62	-	-	-	437.668	437.668	437.668	22963.9	20409.8	21149.4	-	-	-
*RC*201/10/7/100/200/0.30	33	-	-	-	770.432	755.323	764.175	1830.76	1226.12	1490.95	-	-	-
*RC*201/25/18/100/200/0.30	60	-	-	-	445.177	439.346	440.889	24148.8	20607.3	22269.7	-	-	-
*C*101/10/18/200/400/0.07/2	52	-	-	-	421.432	421.028	421.237	26020.9	20472.6	23480.9	-	-	-
*C*101/10/18/200/400/0.07	58	-	-	-	472.007	472.007	472.007	27029.6	22243.5	24222.3	-	-	-
*C*201/10/18/200/400/0.07	54	4339.23	Feasible	86345.4	311.323	311.323	311.323	17945.2	13837.2	15654.7	92.83	83.97	81.87
*R*101/10/18/200/400/0.07	73	-	-	-	443.228	443.228	443.228	50802.7	48652.1	49914.6	-	-	-
*R*201/10/18/200/400/0.07	72	-	-	-	461.168	461.168	461.168	49667.6	41901.5	45313.5	-	-	-
*RC*101/10/18/200/400/0.07	71		-	-	459.822	459.822	459.822	50159	38062.1	41173.1	-	-	-
*RC*201/10/18/200/400/0.07	72	-	-	-	444.904	444.904	444.904	49077.5	37914.6	41977.7	-	-	-
*C*101/10/18/200/400/0.30/2	52	-	-	-	288.184	285.636	286.219	31930.7	25903	28957	-	-	-
*C*101/10/18/200/400/0.30	58	-	-	-	222.324	217.57	219.472	34229.7	25333.5	30033.8	-	-	-
*C*201/10/18/200/400/0.30	54	4310.33	Feasible	43190.3	186.42	185.58	186.084	19310.4	17849.5	18517.8	95.69	58.67	57.13
*R*101/10/18/200/400/0.30	73	-	-	-	313.309	313.309	313.309	53773.1	43808.5	47330.4	-	-	-
*R*201/10/18/200/400/0.30	72	-	-	-	312.881	312.881	312.881	53016.6	40425.3	46936.1	-	-	-
*RC*101/10/18/200/400/0.30	71	-	-	-	315.02	315.02	315.02	49636.8	37041.3	41757.4	-	-	-
*RC*201/10/18/200/400/0.30	72	-	-	-	287.462	287.462	287.462	44136.4	36619.6	40644	-	-	-

Table 5.1: Results for the mono-objective problem for the routing cost as an objective function

that remove random operations have worsened the results. We have tried the combination without adding time and distance-oriented removal and two regret heuristic. It seems that using the combination chosen is the one that ensures good quality of solutions in less computational time. We have set as a stopping condition for all the ALNS a maximum number of iterations $iter_{max} = E[3n/4]$ where the symbol E stands for the integer portion of the number. The update period in the comparison algorithms is set to $iter_{max} = E[n/4]$. The degree of destruction for all ALNS algorithms is $d = E[0.4 \times n] + 1$. We fix $w_T = 0.05$ and the cooling rate $\alpha = 0.99975$ as explained. In our proposed ALNS and in

Instance	*n*	SALNS number=2						SALNS number=8						Improvement SALNS number=2 (%)			Improvement SALNS number=8 (%)		
		max	min	avg	max cpu	min cpu	avg cpu	max	min	avg	max cpu	min cpu	avg cpu	Gap	ICPU min	ICPU avg	Gap	ICPU min	ICPU avg
RE/6/2/80/80/0.30/2	6	115.946	115.946	115.946	0.17	0.14	0.152	115.946	115.946	115.946	0.22	0.19	0.204	0.00	-7.69	-16.92	0.00	-46.15	-56.92
RE/6/2/80/80/0.07/2	6	115.946	115.946	115.946	0.17	0.14	0.152	115.946	115.946	115.946	0.22	0.18	0.202	0.00	-7.69	-16.92	0.00	-38.46	-55.38
RE/6/4/101/101/0.07/2	12	406.321	384.961	389.233	4.41	3.55	3.958	394.081	384.961	388.609	5.74	4.13	4.76	0.00	76.83	74.16	0.00	73.04	68.93
*C*101/10/8/100/200/0.07/2	23	244.27	244.27	244.27	196.42	155.48	172.214	243.722	243.722	243.722	177.28	143.02	157.252	-0.96	99.82	99.80	-0.74	99.83	99.82
*C*101/10/7/100/200/0.07	26	211.202	211.202	211.202	350.53	274.55	304.54	210.824	210.824	210.824	383.89	284.06	342.018	-0.22	99.68	99.65	-0.04	99.67	99.60
*C*101/25/18/100/200/0.07	54	463.819	461.963	462.712	8014.66	7241.65	7556.54	462.132	459.944	460.381	9391.83	7729.25	8844.9	-	-	-	-	-	-
*C*201/10/7/100/200/0.07	24	199.525	199.525	199.525	183.43	135.19	159.246	198.957	198.957	198.957	150.45	129.06	139.278	-0.29	99.84	99.82	0.00	99.85	99.84
*C*201/25/18/100/200/0.07	52	514.96	514.96	514.96	8286.11	7647.74	7821.64	513.907	513.907	513.907	8543.69	7734.67	8154.37	-	-	-	-	-	-
*R*101/10/7/100/200/0.07	34	864.489	864.489	864.489	1447.3	1222.56	1315.1	937.501	861.712	889.592	1318.67	1148.54	1245.96	11.63	98.58	98.48	11.91	98.67	98.56
*R*101/25/18/100/200/0.07	68	807.798	784.233	793.564	41162.9	34234.1	37564.3	797.383	775.653	788.254	41784.8	38052.1	40006.1	-	-	-	-	-	-
*R*201/10/7/100/200/0.07	34	652.662	644.842	650.701	1418.59	1221.54	1300.51	657.605	651.67	653.254	1482.6	1219.99	1323.07	8.65	98.58	98.49	7.68	98.59	98.47
*R*201/25/18/100/200/0.07	65	633.923	633.639	633.771	27489.6	23706.1	25692	628.886	628.674	628.802	30021.8	26723.9	28513.1	-	-	-	-	-	-
*RC*101/10/7/100/200/0.07	33	1136.65	1042.89	1077.2	1252.19	872.67	1062.68	1103.08	1042.89	1082.3	1045.9	974.07	1015.38	-11.98	98.99	98.77	-11.98	98.87	98.82
*RC*101/25/18/100/200/0.07	62	601.422	599.122	599.582	18903.2	16828.1	17543.4	598.923	598.923	598.923	20861.6	16730.9	18115.1	-	-	-	-	-	-
*RC*201/10/7/100/200/0.07	33	1059.03	1009.34	1036.15	1001.65	805.5	901.09	1052.42	1021.05	1031.99	1020.69	912.71	960.874	-4.56	99.07	98.96	-5.77	98.94	98.89
*RC*201/25/18/100/200/0.07	60	608.323	595.644	598.308	19610.9	16046.6	17540.1	595.069	594.432	594.559	18430.3	14439.1	17222	-	-	-	-	-	-
*C*101/10/8/100/200/0.30/2	23	200.894	200.818	200.845	259.55	196.36	227.46	189.825	189.825	189.825	231.49	197.14	220.544	-7.45	99.77	99.74	-1.57	99.77	99.74
*C*101/10/5/100/200/0.30/2	23	303.888	302.733	302.964	299.64	214.97	260.2	322.678	306.322	309.593	268.92	206.08	244.818						
*C*101/10/7/100/200/0.30	26	176.622	176.622	176.622	344.11	279.91	310.336	167.605	167.605	167.605	383.89	304.51	331.792	-10.70	99.68	99.64	-5.05	99.65	99.62
*C*101/25/18/100/200/0.30	54	391.331	388.466	389.655	11534.9	11043.7	11301.6	357.662	351.823	353.126	11929	9754.13	10511.1	-	-	-	-	-	-
*C*201/10/7/100/200/0.30	24	153.738	153.738	153.738	163.87	128.68	146.166	141.624	141.624	141.624	175.93	154.86	165.976	-13.41	99.85	99.83	-4.47	99.82	99.81
*C*201/25/18/100/200/0.30	52	376.91	376.91	376.91	7881.28	6283.33	7071.07	347.61	347.61	347.61	7348.49	6634.33	6900.45	-	-	-	-	-	-
*R*101/25/18/100/200/0.30	68	520.892	520.872	520.876	45399	34878.4	40244.8	519.472	514.241	517.271	53308.8	47975.9	51296.2	99.57	59.61	53.40	99.57	44.45	40.60
*R*201/10/7/100/200/0.30	34	409.543	405.599	407.799	1643.78	1377.5	1504.88	434.346	411.703	421.078	1503.24	1382.41	1446.17	-	-	-	-	-	-
*R*201/25/18/100/200/0.30	65	438.189	438.189	438.189	24726.4	21537.2	22547.5	412.681	412.681	412.681	26526.8	21388.5	24159.9	-	-	-	-	-	-
*RC*101/10/7/100/200/0.30	33	707.585	697.991	702.925	1248.9	986.16	1108.49	734.374	731.477	733.215	1217.74	989.54	1146.94	-	-	-	-	-	-
*RC*101/25/18/100/200/0.30	62	471.935	471.008	471.386	17987.4	15692.5	16892	440.761	438.822	439.21	16104.7	14317.8	15264.2	-	-	-	-	-	-
*RC*201/10/7/100/200/0.30	33	756.489	750.338	753.987	1146.46	982.25	1063.85	754.933	748.534	752.029	1181.12	1001.63	1108.46	-	-	-	-	-	-
*RC*201/25/18/100/200/0.30	60	474.44	464.566	468.516	16544.4	14713.8	15717.8	454.032	445.157	449.552	17734.3	16583.8	17101.6	-	-	-	-	-	-
*C*101/10/18/200/400/0.07/2	52	421.783	421.379	421.51	18176.9	15404.7	16691.9	421.302	420.83	421.15	18719.5	16850.8	17547.1	-	-	-	-	-	-
*C*101/10/18/200/400/0.07	58	472.502	472.477	472.492	23051.4	15447.3	19111.9	472.124	472.099	472.109	22000	18115.5	20011.5	-	-	-	-	-	-
*C*201/10/18/200/400/0.07	54	314.142	314.142	314.142	13153	11270.8	12307.4	311.396	311.396	311.396	14752	12649.5	13538.5	92.76	86.95	85.75	92.82	85.35	84.32
*R*101/10/18/200/400/0.07	73	443.436	443.436	443.436	42682.7	31994.4	35677.4	443.228	443.228	443.228	44770	37767.1	40769.5	-	-	-	-	-	-
*R*201/10/18/200/400/0.07	72	461.196	461.196	461.196	34153.1	29822.9	32245.6	461.168	461.168	461.168	39074.6	32539.7	36178.6	-	-	-	-	-	-
*RC*101/10/18/200/400/0.07	71	460.034	460.034	460.034	31277.4	24758.8	27663.7	459.822	459.822	459.822	35780.1	27410.5	31624.8	-	-	-	-	-	-
*RC*201/10/18/200/400/0.07	72	445.142	445.142	445.142	28460.8	26386.6	27401.8	444.904	444.904	444.904	33168.8	30460.9	31723.7	-	-	-	-	-	-
*C*101/10/18/200/400/0.30/2	52	313.074	311.43	312.234	23589.6	17031.4	20129.4	288.606	286.285	287.303	24631.6	21234.1	22874	-	-	-	-	-	-
*C*101/10/18/200/400/0.30	58	266.532	253.737	261.535	22673.9	19135.4	21060.1	224.255	219.501	221.403	26461.2	19654.1	22797.3	-	-	-	-	-	-
*C*201/10/18/200/400/0.30	54	234.399	234.399	234.399	15910	13022.6	14649.9	202.211	191.642	196.108	13398.8	9908.11	11386.8	94.56	69.85	66.08	95.55	77.06	73.64
*R*101/10/18/200/400/0.30	73	338.412	338.412	338.412	34740.9	28630.6	31390.9	314.749	314.749	314.749	37355.9	32423.9	35048.2	-	-	-	-	-	-
*R*201/10/18/200/400/0.30	72	337.327	337.327	337.327	41341.6	31493.6	36313.7	314.626	314.626	314.626	38962.5	33539.5	36778.9	-	-	-	-	-	-
*RC*101/10/18/200/400/0.30	71	342.399	342.399	342.399	28952.3	24815.2	26601.4	316.102	316.102	316.102	32255.5	26446	30496.7	-	-	-	-	-	-
*RC*201/10/18/200/400/0.30	72	316.273	316.273	316.273	35313.2	29798.4	32599.8	289.567	289.567	289.567	33729.2	29943.3	32416.7	-	-	-	-	-	-

Table 5.2: Results of SALNS with the routing cost as an objective function for small numbers of maintenance intervals' discretization

the classical ALNS algorithms, the values of the rewards are δ_1=50, $\delta_2 = 30$ and $\delta_3 = 20$. The weights of both destroy and repair operators are equal at the beginning to 20. The scores s_h and the number of times the heuristics are used u_h are fixed to 0 at the beginning. The reactive factor is fixed to $\rho = 0.7$. For the learning automata-based algorithms and like in [114], we set the rewards to $a_1 = 0.2$, $a_2 = 0.1$, $a_3 = 0.05$ and the penalty to $b_1 = 0.02$ when we do not accepted the new solution with the simulated annealing criteria. The proposed comparison algorithms use the same neighborhood structures, removal, and insertion operators. They also have been tuned to the same values in all

Instance	*n*	SALNS number=2						SALNS number=8						Impr. of SALNS nb=8 over nb=2 (%)		
		max	min	avg	max cpu	min cpu	avg cpu	max	min	avg	max cpu	min cpu	avg cpu	Gap	ICPU min	ICPU avg
RE/6/2/80/80/0.30/2	6	48.7134	48.7134	48.7134	0.14	0.13	0.136	48.7084	48.7084	48.7084	0.26	0.16	0.208	0.01	-23.08	-52.94
RE/6/2/80/80/0.07/2	6	48.7134	48.7134	48.7134	0.15	0.13	0.138	48.7084	48.7084	48.7084	0.26	0.16	0.206	0.01	-23.08	-49.28
RE/6/4/101/101/0.07/2	12	120.299	120.267	120.273	3.64	2.73	3.178	120.22	120.22	120.22	4.25	3.67	3.896	0.04	-34.43	-22.59
*C*101/10/8/100/200/0.07/2	23	323.216	323.216	323.216	145.69	124.47	132.29	322.578	322.578	322.578	167.23	144.44	156.794	0.20	-16.04	-18.52
*C*101/10/7/100/200/0.07	26	318.824	314.73	315.548	293.34	212.01	257.506	314.252	314.252	314.252	302.26	269.42	276.734	0.15	-27.08	-7.47
*C*101/25/18/100/200/0.07	54	582.691	582.691	582.691	6918.72	5542.36	6190.28	580.668	580.565	580.588	8445.37	6993.85	7660.41	0.36	-26.19	-23.75
*C*201/10/7/100/200/0.07	24	257.743	257.743	257.743	253.3	187.4	212.404	257.717	257.717	257.717	265.91	200.46	233.756	0.01	-6.97	-10.05
*C*201/25/18/100/200/0.07	52	564.867	564.774	564.826	7249.4	6430.94	6857.46	564.663	564.663	564.663	9215.51	7397.53	8287.14	0.02	-15.03	-20.85
*R*101/10/7/100/200/0.07	34	1136.74	1078.14	1089.86	1567.75	1284.35	1458.79	1075.28	1075.28	1075.28	1659.19	1445.08	1589.37	0.27	-12.51	-8.95
*R*101/25/18/100/200/0.07	68	1042.71	1005.58	1025.32	44987.8	39409.8	42078.2	1049.62	1022.96	1032.55	53176.6	49554.6	51395.8	-1.73	-25.74	-22.14
*R*201/10/7/100/200/0.07	34	860.275	846.519	850.212	1723.44	1338.4	1587.26	860.275	845.737	849.58	1795.92	1481.99	1651.63	0.09	-10.73	-4.06
*R*201/25/18/100/200/0.07	65	852.827	849.117	850.206	27473.6	24501.5	25700.5	848.298	845.229	846.459	32922.5	28578.4	30212.6	0.46	-16.64	-17.56
*RC*101/10/7/100/200/0.07	33	1277.72	1231.23	1247.3	1368.04	1060.2	1187.89	1280.31	1231.27	1252.77	1330.45	1139.85	1246.06	0.00	-7.51	-4.90
*RC*101/25/18/100/200/0.07	62	783.029	774.874	778.322	21932.9	20274.8	21268.3	780.192	774.548	777.043	28576	23772.9	26340.7	0.04	-17.25	-23.85
*RC*201/10/7/100/200/0.07	33	1220.97	1187.38	1205.47	1272.15	1029.17	1102.54	1252.3	1184	1208.4	1317.13	1165.5	1248.22	0.28	-13.25	-13.21
*RC*201/25/18/100/200/0.07	60	786.024	754.504	760.907	21176.2	19846.4	20610.4	761.183	754.02	755.902	25163.8	21049.4	22437.2	0.06	-6.06	-8.86
*C*101/10/8/100/200/0.30/2	23	276.905	276.905	276.905	187.16	154.12	168.314	268.505	268.505	268.505	242.52	188.4	217.06	3.03	-22.2	-28.96
*C*101/10/5/100/200/0.30/2	23	403.473	403.473	403.473	286.15	209.16	247.358	429.011	410.065	413.854	390.3	282.26	324.76	-1.63	-34.95	-31.29
*C*101/10/7/100/200/0.30	26	281.798	281.386	281.48	468.89	356.85	396.398	272.499	272.499	272.499	445.64	375.53	402.496	3.16	-5.23	-1.54
*C*101/25/18/100/200/0.30	54	489.128	487.768	488.197	9833.02	8402.64	9285.08	458.145	457.85	458.082	14366	12119.8	13161.7	6.13	-44.24	-41.75
*C*201/10/7/100/200/0.30	24	207.536	207.536	207.536	241.7	174.5	212.578	200.97	200.97	200.97	333.29	261.61	290.172	3.16	-49.92	-36.50
*C*201/25/18/100/200/0.30	52	422.962	421.274	422.145	10282	8728.26	9306.85	404.048	403.766	403.878	12190.7	11278.4	11841.9	4.16	-29.22	-27.24
*R*101/25/18/100/200/0.30	68	826.335	822.133	823.177	63814.6	60426.6	62585.8	825.099	818.357	821.353	105824	81638.9	93010.4	0.46	-35.10	-48.61
*R*201/10/7/100/200/0.30	34	618.967	602.108	610.979	1891.25	1577.19	1698.16	613.147	610.956	612.271	2114.16	1775.9	1939.34	-1.47	-12.60	-14.20
*R*201/25/18/100/200/0.30	65	663.298	660.316	661.686	34970.8	31173.8	33390	649.278	645.558	646.462	49498.5	42531.5	45894.5	2.23	-36.43	-37.45
*RC*101/10/7/100/200/0.30	33	910.191	883.708	890.18	1551.66	1266.44	1388.54	933.105	926.703	929.79	1793.97	1474.02	1670.96	-4.87	-16.39	-20.34
*RC*101/25/18/100/200/0.30	62	653.654	651.662	652.576	30657.5	28044.7	29327.1	636.755	632.625	634.322	36492	30634.8	34054.6	2.92	-9.24	-16.12
*RC*201/10/7/100/200/0.30	33	975.222	942.922	963.886	1506.66	1173.6	1322.4	1003.41	955.51	973.516	1945.49	1431.64	1583.96	-1.33	-21.99	-19.78
*RC*201/25/18/100/200/0.30	60	633.918	632.786	633.38	24798.6	21781.4	23346.2	617.018	613.735	615.599	29591.5	27271.3	28307.4	3.01	-25.20	-21.25
*C*101/10/18/200/400/0.07/2	52	739.334	739.334	739.334	7589.29	6963.07	7283.35	738.706	738.706	738.706	10634.2	9410.65	10030.2	0.08	-35.15	-37.71
*C*101/10/18/200/400/0.07	58	798.304	798.304	798.304	17845	16445.7	17049.7	798.214	798.214	798.214	20032.3	18399.9	18974.8	0.01	-11.88	-11.29
*C*201/10/18/200/400/0.07	54	529.015	529.015	529.015	10407.8	8864.4	9654.24	527.623	527.623	527.623	12875.2	10895.1	11930.7	0.26	-22.91	-23.58
*R*101/10/18/200/400/0.07	73	1087.08	1087.08	1087.08	36335.8	32524.6	34796.7	1087.01	1087.01	1087.01	43695.8	37089.2	39302.7	0.01	-14.03	-12.95
*R*201/10/18/200/400/0.07	72	1026.64	1026.64	1026.64	34377.9	29981.8	31369.9	1026.47	1026.47	1026.47	39801.1	36685.4	37798.2	0.02	-22.36 -	20.49
*RC*101/10/18/200/400/0.07	71	1053.87	1053.87	1053.87	30661	27721.6	29339.7	1053.56	1053.56	1053.56	37597.2	31527.5	33905.2	0.03	-13.73	-15.56
*RC*201/10/18/200/400/0.07	72	1024.08	1024.08	1024.08	39749.5	33940.8	36034.4	1023.71	1023.71	1023.71	49851.9	45715.6	47708.9	0.04	-34.69	-32.40
*C*101/10/18/200/400/0.30/2	52	622.486	622.486	622.486	11469.5	10851.4	11231.9	602.092	602.092	602.092	16247.3	14314.2	15246	3.28	-31.91	-35.74
*C*101/10/18/200/400/0.30	58	581.286	581.286	581.286	17927.6	16780.7	17350	554.603	554.603	554.603	16389.1	15332.2	15890.8	4.59	8.63	8.41
*C*201/10/18/200/400/0.30	54	422.73	422.524	422.679	23722.8	20171.5	22188.3	407.752	407.744	407.745	28177.1	24033.8	26543.6	3.50	-19.15	-19.63
*R*101/10/18/200/400/0.30	73	956.952	956.951	956.952	137785	88050.8	106829	952.589	952.58	952.584	169710	136216	146329	0.46	-54.70	-36.97
*R*201/10/18/200/400/0.30	72	893.854	893.721	893.76	63630.8	57807.9	61224.6	884.697	884.683	884.687	111558	78221.6	89228.8	1.01	-35.31	-45.74
*RC*101/10/18/200/400/0.30	71	904.831	904.831	904.831	71267.5	66047.6	68398.1	897.325	897.325	897.325	95805.3	78562.8	87454.3	0.83	-18.95	-27.86
*RC*201/10/18/200/400/0.30	72	884.427	884.33	884.379	77532.2	63861	67891.8	870.942	870.942	870.942	100121	85929.8	94260.9	1.51	-34.56	-38.84

Table 5.3: Results of SALNS with the maintenance cost as an objective function for small numbers of interval's discretization

parameters. They also have the same schema. The only difference between the four algorithms is the adaptive mechanism. The proposed semi-ALNS algorithm uses the semi-adaptive mechanism proposed. ALNS RS represents the classical ALNS with a reset of scores every period. ALNS represents the classical ALNS without this reset. LA-ALNS U represents an ALNS with learning automata that are updated every period as defined in [114], LA-ALNS represents ALNS with

learning automata as an adaptive mechanism without the update every period.

Table 5.4 illustrates the improvement of ALNS over CPLEX for both solution quality and time for small, medium, and large instances. We report in tables 5.6 and 5.7 the improvement of solution quality (gap) and in time when using semi ALNS compared to each of the four algorithms for the minimum and average values obtained in five runs. The equations 5.30 and 5.31 represent the average percent decrease of objective value and CPU time, respectively, when using the proposed semi-ALNS algorithm compared to the comparison algorithms (CA) or CPLEX for the instances for each of the defined categories. The time used is in seconds.

$$Gap(CA, SemiALNS) = (\frac{Value(CA) - Value(SemiALNS)}{Value(CA)}) \times 100\% \tag{5.30}$$

$$ICPU(CA, SemiALNS) = (\frac{Time(CA) - Time(SemiALNS)}{Time(CA)}) \times 100\% \tag{5.31}$$

The equation 5.32 represents a measure we define for robustness that indicates the percent deviation between the average solution and the minimum solution obtained by an algorithm for a given instance.

$$RPD(f_{avg}, f_{min}) = (\frac{Value(f_{avg}) - Value(f_{min})}{Value(f_{avg})}) \times 100\% \tag{5.32}$$

5.7.2 Computational results

Comparison with the commercial solver

The detailed results of the SALNS and the CPLEX solver for the routing objective are represented in Table **??**. Bold values are the minimum objective values obtained by either the SALNS or the solver. Whenever a number in the improvement column is bold, there is an improvement. These results are summarised in Table 5.4. We can conclude that, on average, for all instances, the proposed algorithm outperforms the solver CPLEX in both solution quality and CPU time. For the 29 instances for which the solver returned either a feasible or optimal solution, the improvement of objective value on average defined is 18.26% and the percent decrease of CPU time is 62.30% on average.

Figures 5.5 and 5.6 show the improvement of SALNS over CPLEX in objective value and time for some small and medium instances for which CPLEX returned

Instances Type	n	# instances	# feasible	# optimal	Avg gap	Avg ICPU
RE	$[\![6,12]\!]$	4	0	4	-0.59	-30.23
C,R,RC	$[\![23,34]\!]$	16	8	8	1.04	70.03
	$[\![52,68]\!]$	9	8	1	56.73	89.69
	$[\![71,73]\!]$	13	Not solved by CPLEX		ALNS returned a solution	

Table 5.4: Improvement of ALNS over Cplex

Instance	*n*	CPLEX			SALNS						Improvement	
		Objective	Status	CPU(s)	max	min	avg	max cpu	min cpu	avg cpu	Gap	ICPU
RE/6/2/80/80/0.07/2	6	**110.88**	optimal	0.03	129.6	**110.88**	114.624	0.13	0.05	0.078	**0**	-66.67
RE/6/4/101/101/0.07/2	12	**353.52**	optimal	0.92	362.64	**353.52**	357.168	2.21	1.56	1.924	**0**	-69.57
RE/6/2/80/80/0.30/2	6	**98.64**	optimal	0.09	105.36	**98.64**	99.984	0.11	0.07	0.08	**0**	**22.22**
RE/6/4/101/101/0.30/2	12	**325.92**	optimal	1.74	352.502	333.6	345.3004	2.64	1.86	2.258	-2.36	-6.90
*C*101/10/8/100/200/0.07/2	23	**119.699**	optimal	131.61	**119.699**	**119.699**	**119.699**	103.34	80.27	**92.044**	**0**	39.01
*C*101/10/5/100/200/0.07/2	23	**225.078**	optimal	2186.44	229.33	**225.078**	225.929	131.38	108.9	119.396	**0**	**95.02**
*C*101/10/7/100/200/0.07	23	**82.536**	optimal	55.28	**82.536**	**82.536**	**82.536**	168.12	133.47	145.474	**0**	-141.44
*C*101/25/18/100/200/0.07	54	**192.184**	optimal	114985	**192.184**	**192.184**	**192.184**	4108.85	3611.06	3902.63	**0**	**96.86**
*C*201/10/7/100/200/0.07	24	**77.168**	optimal	44.19	**77.168**	**77.168**	**77.168**	68	64.17	65.89	**0**	-45.21
*C*201/25/18/100/200/0.07	52	223.01	feasible	83458	229.22	**222.451**	227.549	3980.68	3277.91	3656.4	**0.25**	**96.07**
*R*101/10/7/100/200/0.07	34	507.37	feasible	51587.8	490.304	**450.746**	458.658	695.65	623.84	661.456	**11.16**	**98.79**
*R*101/25/18/100/200/0.07	68	244302	feasible	67536.8	316.154	**304.602**	308.462	19699.2	14500.9	17107.1	**99.88**	**78.53**
*R*201/10/7/100/200/0.07	34	**341.741**	feasible	236032	354.432	**341.741**	350.552	716.7	630.99	679.16	**0**	**99.73**
*R*201/25/18/100/200/0.07	65	-	-	-	**237.633**	**237.633**	**237.633**	15369.8	13234	14520.7	-	-
*RC*101/10/7/100/200/0.07	33	752.961	feasible	51859	738.237	**710.88**	718.583	509.78	458.09	489.108	**5.59**	**99.12**
*RC*101/25/18/100/200/0.07	62	18763.7	feasible	14382.1	263.272	**261.552**	262.186	8255.24	6922.03	7455.85	**98.61**	**51.87**
*RC*201/10/7/100/200/0.07	33	678.987	feasible	241174	677.558	**612.94**	653.2352	440.41	298.29	358.556	**9.73**	**99.88**
*RC*201/25/18/100/200/0.07	60	281.088	feasible	343932	**277.384**	**277.384**	**277.384**	10296.6	9258.18	9585.13	**1.32**	**97.31**
*C*101/10/8/100/200/0.30/2	23	**82.544**	feasible	225506	83.344	**82.544**	82.9184	106.57	87.78	97.454	**0**	**99.96**
*C*101/10/5/100/200/0.30/2	23	**82.544**	optimal	18631.6	83.08	**82.544**	82.8656	109	84.83	96.958	**0**	**99.54**
*C*101/10/7/100/200/0.30	26	**65.044**	optimal	2288.14	**65.044**	**65.044**	**65.044**	142.42	109.39	125.484	**0**	**95.22**
*C*101/25/18/100/200/0.30	54	42379.5	feasible	78964.4	159.288	**158.432**	158.914	5070.92	3966.48	4459.98	**99.63**	**94.98**
*C*201/10/7/100/200/0.30	24	**70.116**	optimal	356.48	**70.116**	**70.116**	**70.116**	74.66	59.26	68.974	**0**	**83.38**
*C*201/25/18/100/200/0.30	52	116913	feasible	80200.3	183.128	**179.144**	180.359	4215.36	3599.58	3967.49	**99.85**	**95.51**
*R*101/10/7/100/200/0.30	34	**94.96**	feasible	236317	97.468	**94.96**	95.928	658.57	483.04	589.172	**0.00**	**99.80**
*R*101/25/18/100/200/0.30	68	-	-	-	205.024	202.556	203.202	18352.1	15026.2	16476.2	-	-
*R*201/10/7/100/200/0.30	34	**83.28**	optimal	24458.8	87.18	**83.28**	84.828	579.41	493.24	537.442	**0**	**97.98**
*R*201/25/18/100/200/0.30	65	-	-	-	191.644	**190.732**	191.042	13055.6	11468.4	12188.8	-	-
*RC*101/10/7/100/200/0.30	33	**173.116**	feasible	241311	194.481	181.86	188.629	569.07	449.7	503.768	-5.05	**99.81**
*RC*101/25/18/100/200/0.30	62	-	-	-	231.428	229.008	230.218	9536.05	8498.41	8951.62	-	-
*RC*201/10/7/100/200/0.30	33	**117.011**	feasible	258832	133.526	**117.011**	122.694	433.29	377.88	414.96	**0**	**99.85**
*RC*201/25/18/100/200/0.30	60	-	-	-	221.332	**217.413**	218.651	8417.77	6768.66	7479.31	-	-
*C*101/10/18/200/400/0.07/2	52	148.156	feasible	342141	**130.743**	**130.743**	**130.743**	9141.93	6237.36	7737.75	**11.75**	**98.18**
*C*101/10/18/200/400/0.07	58	-	-	-	**95.972**	**95.972**	**95.972**	14606.6	10655.1	12297.1	-	-
*C*201/10/18/200/400/0.07	54	-	-	-	96.104	96.104	96.104	8470.07	6785.02	7486.99	-	-
*R*101/10/18/200/400/0.07	73	-	-	-	**71.892**	**71.892**	**71.892**	21833	19692.7	20437	-	-
*R*201/10/18/200/400/0.07	72	-	-	-	**71.896**	**71.896**	**71.896**	23954.7	17682.5	20366.6	-	-
*RC*101/10/18/200/400/0.07	71	-	-	-	**86.104**	**86.104**	**86.104**	17273.1	15486.3	16225.6	-	-
*RC*101/10/18/200/400/0.07	72	-	-	-	**86.104**	**86.104**	**86.104**	20202.1	15885.3	17852.1	-	-
*C*101/10/18/200/400/0.30/2	52	11986.3	feasible	252973	100.032	**89.256**	93.8592	7087.5	5223.03	6135.14	**99.26**	**97.94**
*C*101/10/18/200/400/0.30	58	-	-	-	86.324	**84.624**	85.108	12139.9	10507.2	10980.8	-	-
*C*201/10/18/200/400/0.30	54	-	-	-	99.052	**89.624**	91.7152	8208.33	6116.23	7305.57	-	-
*R*101/10/18/200/400/0.30	73	-	-	-	82.712	**71.888**	80.5472	32585.2	29741.7	30938.9	-	-
*R*201/10/18/200/400/0.30	72	-	-	-	91.308	**71.896**	81.2096	29377	22308.5	26935.2	-	-
*RC*101/10/18/200/400/0.30	71	-	-	-	94.792	**86.104**	87.8416	18071.8	14166.9	16006.9	-	-
*RC*201/10/18/200/400/0.30	72	-	-	-	97.992	**86.104**	92.332	22546.5	20935.3	21917.2	-	-

Table 5.5: Results of SALNS for the routing objective

an optimal or near-optimal solution.

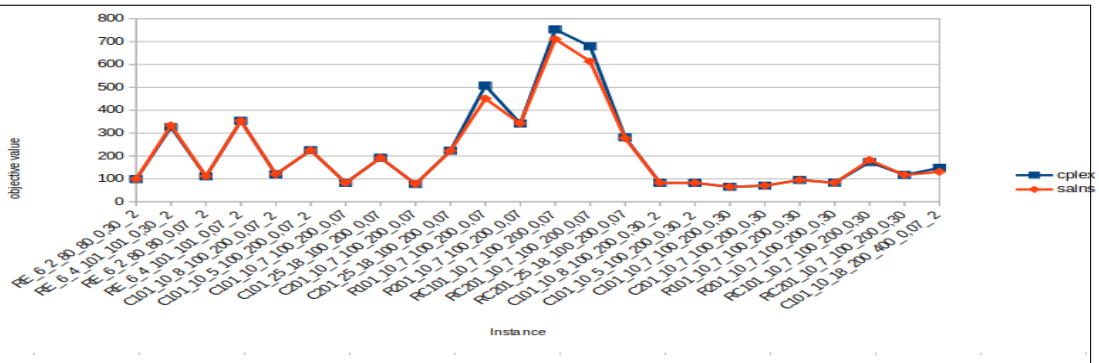

Figure 5.5: Comparison in the objective value of SALNS and CPLEX

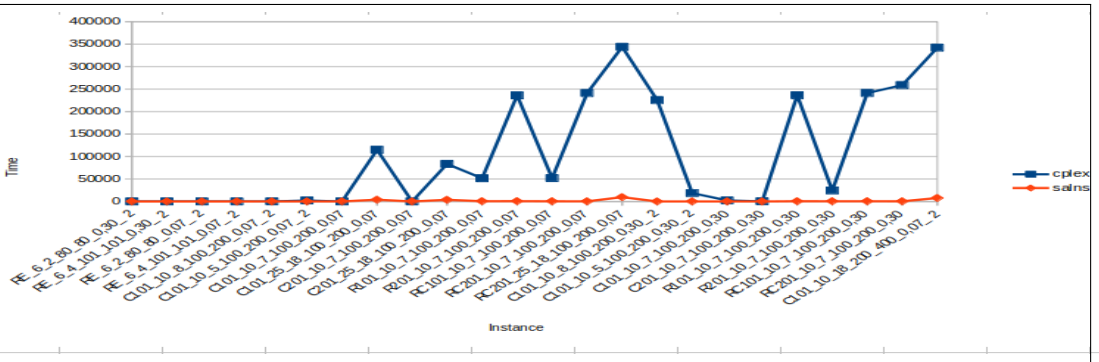

Figure 5.6: Comparison in time of SALNS and CPLEX

Comparison with the literature

The comparative algorithms used consider the same choices related to the problem's specifications that perform better for the problem. The results of this algorithms comparison suggest that the adaptive selection mechanism proposed outperformed in terms of time and solution quality, the two classical ALNS algorithms (with and without resetting scores each period) and the ALNS with learning automata (with and without the update of probabilities each period). All the total values are superior to 0 in tables 5.6 and 5.7. It is worth mentioning that when using the semi-adaptive mechanism (mix between random selection and selection according to past success) with the learning automata equations,

our algorithm and LA-ALNS with the proposed semi-adaptive selection mechanism have almost similar performance. Our algorithm is also more robust than the other classical ALNS algorithms since its average RPD is lower than theirs. The average RPD for semi-ALNS is 1.43%. The average RPD values for classical ALNS with reset scores and without are respectively equal to 1.63% and 1.78%. The average RPD values for ALNS with learning automata with and without update are respectively equal to 1.17% and 1.42%. The ALNS with LA is slightly better than our algorithm for the average RPD indicator. For all the instances, the proposed algorithm performs better on average than the four other algorithms in terms of CPU time and solution quality. Our algorithm performs better than the classical ALNS algorithms for the RPD indicator but slightly worse than the ALNS with LA algorithms.

Instances	n	ALNS RS		ALNS		LA- ALNS U		LA-ALNS	
		gap	gap(avg)	gap	gap(avg)	gap	gap(avg)	gap	gap(avg)
RE	[[6,12]]	0.56	0.80	1.97	0.33	1.34	0.29	0.63	0.18
C	[[23,34]]	0	0.57	0.08	0.23	0.16	0.39	0.28	1.10
	[[52,68]]	1.43	1.74	1.45	1.91	2.39	1.82	2.51	2.09
R	[[23,34]]	0.55	0.70	0	2.18	0.15	0.63	0.18	1.59
	[[52,68]]	-0.05	0.52	-0.37	0.22	0.80	0.90	0.61	1.71
	[[71,73]]	3.80	0.86	7.09	3.69	5.69	3.46	-7.76	5.84
RC	[[23,34]]	0.62	-0.01	-0.43	1.80	0.18	0.37	0.62	1.07
	[[52,68]]	0.06	-0.06	0.05	1.43	0.10	0.37	0.34	0.60
	[[71,73]]	0	3.28	3.06	2.77	4.58	3.67	4.58	3.67
Total		0.74	0.96	1.15	1.46	1.34	1.09	1.60	1.63

Table 5.6: Improvement in solution quality of SALNS over the other ALNS algorithms

Instances	n	ALNS RS		ALNS		LA- ALNS U		LA-ALNS	
		ICPU	ICPU(avg)	ICPU	ICPU(avg)	ICPU	ICPU(avg)	ICPU	ICPU(avg)
RE	[[6,12]]	11.26	8.65	7.91	-3.06	12.02	12.27	10.14	9.16
C	[[23,34]]	16.45	22.73	8.45	26.72	18.51	23.56	8.27	6.33
	[[52,68]]	9.87	21.6	-14.01	12.64	17.61	26.49	-10.85	8.54
R	[[23,34]]	19.86	22.08	-9.81	3.34	25.65	28.76	3.73	10.87
	[[52,68]]	5.64	14.11	11.29	20.77	26.37	27.16	19.76	19.81
	[[71,73]]	11.31	14.86	15.50	17.24	5.90	6.06	-7.76	5.84
RC	[[23,34]]	16.62	22.23	20.73	28.82	24.58	25.83	-12.14	-1.05
	[[52,68]]	11.36	19.90	-13.32	16.41	22.83	27.49	17.44	21.24
	[[71,73]]	-4.13	3.28	-3.33	28.16	29.11	28.09	18.02	27.06
Total		11.61	19.46	0.36	17.86	20	23.99	3.19	10.52

Table 5.7: Improvement in time of SALNS over the other ALNS algorithms

Figures 5.7 and 5.8 represent the improvement of the average time in five runs

of SALNS over the two classical ALNS algorithms and the two ALNS algorithms with learning automata.

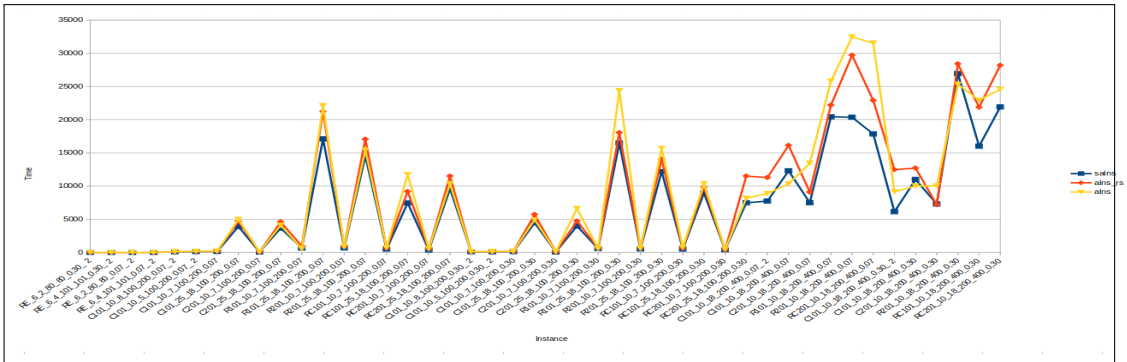

Figure 5.7: Comparison in time of SALNS and classical ALNS algorithms

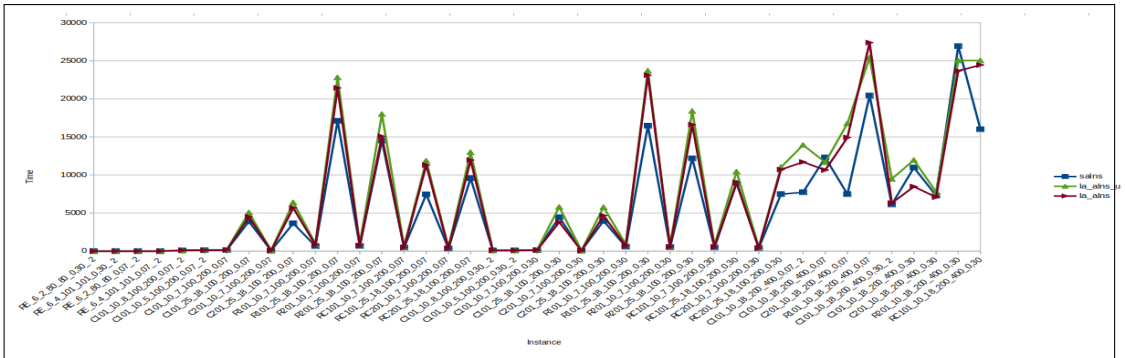

Figure 5.8: Comparison in time of SALNS and LA-ALNS algorithms

5.8 Concluding remarks

Opportunistic maintenance consists in planning maintenance at the right time on equipment while penalizing, the less, production process. However, determining this right time is a challenging problem since it includes technical considerations of the machine's lifetime and production periods to assess the less negative impact on production. We defined a new problem that extends a problem faced by maintenance service providers. A set of technicians must visit several machines geographically distributed in multiple customers' sites subject to

random failures. Production stoppages are available and already planned on the horizon and need to be favored to reduce the negative impact on production. Three interrelated aspects characterize the problem: routing, maintenance, and production, influencing the final decision and the total cost. To the best of our knowledge, no previous work has considered these three aspects together, and this research is the first of its kind to propose such an approach. In this chapter, we proposed a new problem, a mathematical model that integrates the three aspects of the presented problem, and a heuristic approach to deal with it. The contributions of this chapter are several. We first started with a rigorous identification of the positioning of the proposed problem in the literature and we defined a stream for it. We then proposed a mathematical formulation that minimizes the total travel or maintenance costs. Two types of penalties are considered in each of the objectives. The first penalty cost is incurred when not respecting the maintenance intervals. The second penalty represents the possible production losses if the opportunities of available production stoppage intervals can not be used for maintenance operations. A new heuristic based on discretization was proposed to determine the best start time in the maintenance interval and the most convenient production stoppage. It was integrated with a Semi Adaptive Large Neighborhood Search to solve the problem. SALNS integrates a new removal operator based on risk and a semi-adaptive mechanism to cope with the peculiarities of the problem. The metaheuristic selects the most suitable operators while alternating between learning and diversification to obtain high-quality solutions. Results suggest that the SALNS performs well since the objective value significantly decreases with the number of tested points in the maintenance interval. It also outperforms the commercial solver on average in all tested instances.

The SALNS algorithm is tested on the Maintenance Scheduling and Workforce Routing Problem of Chapter 3. We conducted a comparative analysis with the commercial solver CPLEX and adapted algorithms schemes from the literature: classical ALNS and ALNS with learning automata (LA-ALNS) algorithms. The algorithms consider the same choices related to the problem's specifications that perform well for the problem and use the same local search procedure. The results suggest that the proposed ALNS performs better.

In the next chapter, we propose and model a bi-objective variant of the problem. Additional assumptions are considered. Multi-Objective methods based on the SALNS, PLS, and VND algorithms are proposed to solve the problem.

Chapter 6

The Joint Opportunistic Maintenance Scheduling and Workforce Routing Problem: Multi-objective Model and algorithms based on Pareto Local Search and Adaptive Large Neighborhood Search

Outline of the current chapter

6.1 Introduction

Opportunistic Maintenance (OM) is an important maintenance strategy that significantly impacts manufacturing plants and service companies. OM is a Preventive Maintenance (PM) strategy that takes advantage of a system's shutdown to reduce production interruption costs. The planned and unplanned production stoppages are opportunity intervals where maintenance resources are available, and almost no production losses are incurred since the system is shut down. The unexpected stoppage intervals are due to the failure of the components of systems, market interruptions, etc. The planned stoppage intervals can be

numerous such as days off of a week, annual shutdown period of the plant. These stoppages can also be due to environment, legal recommendations, market, lack of raw material, etc [107].

This chapter addresses the bi-objective variant of the Joint Opportunistic Maintenance Scheduling and Workforce Routing Problem. Production stoppages are scheduled on the planning horizon. Technicians need to perform preventive maintenance during these production stoppages to impact the least the production activities of the maintenance service provider's customers. To the best of our knowledge, no previous work has considered a model that combines maintenance, production, and routing. This chapter extends the prior problem adressed in Chapter 5 to simultaneously determine the optimal maintenance and routing plan while satisfying production stoppages constraints by finding the best OM scenario. Consequently, this new problem shares similar properties with the problem of Chapter 5 but is extended to more complex levels.

Several conflicting criteria are necessary to find suitable solutions accurately. In addition, these solutions need to account for the benefits of all the objectives considered.

In scalarization techniques, we can order two feasible solutions to differentiate between them and select the best objective value. In contrast, the Pareto dominance concept permit only a partial order in the objective space [96]. Indeed, there are also several incomparable feasible solutions [96]. Therefore, we need to determine the set of efficient solutions called the Pareto front.

The considered problem is NP-hard, and finding optimal solutions is difficult even for small instance sizes. Therefore, developing multi-objective metaheuristics or heuristics for this problem is essential. Pareto Local Search (PLS) [69] is a local search heuristic method for solving NP-hard multi-objective combinatorial problems (MOCP) in the Pareto sense. Variable Neighborhood Descent (VND) is a heuristic proposed by Mladenovic and Hansen [87] for single-objective problems based on the fundamental idea of the systematic change of neighborhood structures. Adaptive Large Neighborhood Search (ALNS) is a large neighborhood improvement heuristic [115]. These methods are hybridized to form new PLS and multi-objective VND and ALNS to solve the problem and multi-objective combinatorial problems in general.

The contributions of this chapter are the following:

(1) A new bi-objective mathematical model which integrates routing, maintenance, and production considerations is proposed in the case of time-based preventive maintenance. The first objective minimizes the total travel cost related to technicians' routing and the production losses incurred if the available opportunity intervals for maintenance are not selected. The second objective minimizes the total maintenance cost and the penalty cost incurred if the maintenance intervals are not respected. To our knowledge, the multi-objective variant of the novel problem described in Chapter 5 has never been studied to date in the literature.

(2) New multi-objective heuristics have been proposed to solve the problem. These algorithms are based on the Pareto dominance concept. They also start with a unique initial solution. The Pareto Local Search Multi-objective Variable Neighborhood Descent (PLS-MOVND) hybridizes the PLS and the VND methods to achieve the best performances. It uses new adaptations of VND mechanisms in the multi-objective context. The Multi-objective Semi Adaptive Large Neighborhood Search (MOSALNS/P) is an extended framework of the ALNS heuristic to consider the multi-objective. It uses the MOVND/P algorithm and several specifically designed features in the intensification phase.

(3) Computational experiments are realized to validate the proposed model and solutions approaches. Results suggest that our methods considerably outperform an existing method of the literature. Furthermore, a comparison between the proposed algorithms PLS-MOVND, MOSALNS/P, and MOVND/PI confirms the relevance of the approach adopted.

The rest of the chapter is organized as follows. In Section 6.2 we review related multi-objective methods in the literature. Section 6.3 describes the bi-objective problem and its mathematical formulation. The multi-objective algorithms are presented in Section 6.4. They are followed by an experimental evaluation of several generated instances problems in Section 6.5. Concluding remarks in Section 6.6 close the chapter.

6.2 Literature on related solution methods

Multi-objective problems are generally classified based on the role played by the Decision-Maker (DM) in the decision process. The four classes include no-preference, a priori, a posteriori, and interactive methods [100]. In no-preference methods, there is no decision-maker, the Multi-Objective Problem (MOP) is solved, and a unique Pareto optimal solution is returned. It is the closest to the ideal point in terms of euclidian distance. In a priori methods, the DM expresses his preference before optimization. In most a priori methods, the objectives are aggregated into a single objective. The most popular a priori method is called the weighted sum approach. Other methods included in this category are lexicographic ordering and goal programming. In these two classes, the MOP is transformed into a mono-objective problem. The third class of MO methods includes a posteriori methods. In this class, the goal is to find all the Pareto optimal solutions. The decision-maker chooses one out of them in the second step. Examples of methods in this category are the epsilon constraint method, weighted sum approach with weights variation, local search metaheuristics, and Evolutionary Multi-Objective algorithms (EMO) [100]. They are divided into three main groups: aggregation, dominance, and performance indicator-based algorithms [100]. The interactive and progressive methods are similar in resolution and formulation to the a posteriori methods. In both categories, the multi-objective formulation of the problem is maintained. However, in the interactive methods, the DM guides the optimization process by giving preference during optimization.

Most research papers deal with the second and third categories and focus on Evolutionary Multi-Objective algorithms (EMO) [100]. Local search metaheuristics are less used but have recently attracted many researchers. They have been proven to be effective approaches to solving many problems, mainly single objective VRP variants, and outperform the population-based method in this latter problem [104].

The first multi-objective local search method called Pareto Local Search (PLS) was proposed by Paquete et al. [69] for the bi-objective Traveling Salesman Problem. They extended the single objective local search algorithm. The PLS

algorithm starts with a set of non-dominated solutions called an archive. The new solution is compared with the archive if it is not dominated by the current solution s. It is a speed-up technique used by the authors. Furthermore, the solution picked from the archive should not be flagged as visited at each iteration. Finally, the neighborhood is explored, and the new solution is accepted if any solution in the archive does not dominate it. This general local search approach to multi-objective problems has been called Pareto local search. The PLS obtained outstanding results in solution quality with the best performing metaheuristics. Dubois-Lacoste et al. [70] presented a study of the anytime behavior of original PLS. This algorithm can not be effective when terminated before completion. The original PLS has poor anytime behavior. The completion of the algorithm is time-consuming for large instances. The authors used alternative algorithmic components for each PLS step to improve its anytime behavior. An alternative to the selection step consists of selecting solutions that have more potential to improve the current archive. They defined the concept of the optimistic hypervolume contribution (ohvc). It is the potential contribution to the hypervolume of the archive by the local ideal point. They then proposed to accept only "dominating" solutions to speed up the search. If such solutions were no longer found, they switched to the criteria adopted in PLS for accepting non-dominated solutions. Finally, the last alternative for neighborhood exploration is using the first-improvement strategy to generate the Pareto front until all solutions in the archive are flagged as visited, then moving to the PLS best improvement strategy. The objective of this alternative is to converge first to a good approximation of the Pareto front while the PLS best-improvement phase completes the archive with the remaining neighbors. They proposed another PLS variant called Dynagrid-HV, a dynamic discretization of the objective space to converge faster, which outperformed their first PLS variant and the original PLS. They tested it on the bi-objective traveling salesman problem and the bi-objective quadratic assignment problem.

Much recent research proposes multi-objective local search metaheuristics for their effectiveness on significant variants of problems. It incorporates with them some elements of Evolutionary Multi-Objective algorithms (EMO) apart from genetic operators. This hybridization uses local search metaheuristics

with structure, component, or methodology of evolutionary multi-Objective algorithms (EMO) to achieve the best performances. In this hybridization with the local search, scalarization algorithms are the most used.

Zhang and Li [72] presented MOEA/D, a method that decomposes the problem into several scalar optimization subproblems and optimizes them simultaneously. They found that MOEA/D outperformed or performed as well as the Multi-Objective Genetic Local Search (MOGLS) and NSGA-II on multi-objective 0–1 knapsack and continuous optimization problems. Moreover, the normalization of objectives makes MOEA/D capable of performing well even with disparately-scaled objectives. Ben Mansour et al. [71] presented an iterated multi-objective local search algorithm, called ($MoLS^{AugWT}$), based on the scalarizing function known as the augmented weighted Chebyshev function. In addition, a neighborhood structure uses a ranking algorithm. This latter is based on a weighted addition ratio. Its primary purpose is to order the candidates' items to add to the solutions. This method focuses on the most promising areas of the search space to improve the quality of the obtained solutions but also speeds up the convergence of the population of the solutions. Furthermore, they proposed the gradual weight vectors generation method. It aims to generate weight vectors that gradually change during the search process. Experimental results have shown the performance of the proposed algorithm over the state-of-the-art algorithms for the multi-objective multidimensional knapsack problem. Cota et al. [90] proposed two MO-ALNS algorithms for an unrelated parallel machine scheduling problem with setup times. They considered the minimization of two objectives simultaneously: the makespan and the total energy consumption. The first algorithm, MO-ALNS, is an extension of the single objective ALNS with Learning Automata (ALNS-LA) with an adaptive mechanism proposed in their previous work [114]. It uses a multi-objective random variable neighborhood descent method (MO-RVND) that returns one solution. The acceptance criterion accounts for the Pareto dominance in the algorithm and MO-RVND. MO-ALNS is entirely based on Pareto dominance and starts with one unique solution for each iteration. The second algorithm follows the general structure of the MOEA/D algorithm. It decomposes the problem into many sub-problems and generates weights and neighbors using the Tchebycheff

approach commonly used in MOEA/D. Single objective (ALNS-LA) [114] is used to solve the subproblems instead of the genetic operators. It uses, however, the aggregation function of Tchebycheff as an acceptance criterion in local search. MO-ALNS/D is proposed since MO-ALNS offers no control of diversification. Indeed, the decomposition and aggregation in MOEA/D preserved in MO-ALNS/D are responsible for controlling the diversification in the Pareto front approximation. They concluded that MO-ALNS/D provides better results for large instances than MO-ALNS, while MO-ALNS performs better for small instances. Both algorithms outperformed the MOEA/D algorithm on generated instances for quality indicators for large instances. The comparison of time between the algorithms has not been performed. Cornu et al. [73] proposed a multi-objective metaheuristic called Perturbed Decomposition Algorithm (PDA) in two-phase. In the first phase, PDA decomposes the search into several linearly aggregated subproblems. The second phase conducts an iterative process. First, the aggregated problems are perturbed and then optimized using the ILS algorithm followed by the PLS algorithm. The algorithm is better than those used for the bi-objective and tri-objective TSP.

6.3 Problem definition and formulation

6.3.1 Additional variables

In the multi-objective case, we consider the hypothesis that the teams of technicians start the maintenance operations as soon as they arrive. Therefore, the choice of the best production stoppage time window is the only one considered. To model this hypothesis, adding the following intermediate variables is crucial to distinguish between the start and arrival times.

- A_{ik}: the arrival date to the maintenance operation i by the team of technicians k.

- W_{ik}: the waiting time between the arrival to the location of a maintenance operation i and its effective start by the team of technicians k.

6.3.2 Objective functions

In the objective functions, each of the needed terms to minimize should be assigned to one of the objective functions. We should separate only the conflicting objectives so that the design of the objectives makes sense when determining the minimal number of objectives needed. In the present case, the maintenance objective conflicts with the routing objective. The maintenance objective also conflicts with the production objective since this latter term aims to execute maintenance operations in production stoppage intervals to penalize the production less. The optimal maintenance times are not necessarily located in these opportunity intervals. Moreover, in an organization, the maintenance and production departments are always in conflict. Indeed, these departments' objectives conflict. The penalty is applied when the maintenance interval that reduces the maintenance cost is not respected. This penalty aims to force respecting the maintenance interval. Therefore, it is not in conflict with the maintenance objective. It is rather in accordance with that latter objective. The model and the objective functions considered are presented in the following section.

6.3.3 The integrated opportunistic maintenance and workforce routing model with production stoppages constraints

The mathematical model can be formulated as follows:

$$f_1 = \sum_{i=0}^{n}\sum_{j=1}^{n+1}\sum_{k=1}^{K} c_{i,j}x_{i,j,k} + \sum_{i=1}^{n}\sum_{p=1}^{P}(min(d_i, pe_{i,p}) + min(d_i, pl_{i,p})) \tag{6.1}$$

$$f_2 = \sum_{i=1}^{n}\sum_{k=1}^{K} CM_i(\theta_{i,k} - \rho_{i,k}) + \sum_{i=0}^{n+1}\sum_{k=1}^{K} c \times p_{i,k} \tag{6.2}$$

S.t.

$$\sum_{j=1}^{n+1}\sum_{k=1}^{K} x_{i,j,k} = 1, \quad \forall i \in \mathcal{O}, i \neq j \tag{6.3}$$

$$\sum_{i=0}^{n}\sum_{k=1}^{K} x_{i,j,k} = 1, \quad \forall j \in \mathcal{O}, i \neq j \tag{6.4}$$

$$\sum_{j=1}^{n+1} x_{0,j,k} = 1, \quad \forall k \in \mathcal{K} \tag{6.5}$$

$$\sum_{i=0}^{n} x_{i,n+1,k} = 1, \quad \forall k \in \mathcal{K} \tag{6.6}$$

$$\sum_{i=0}^{n} x_{i,j,k} = \sum_{i=1}^{n+1} x_{j,i,k}, \quad \forall j \in \mathcal{O}, \forall k \in \mathcal{K} \tag{6.7}$$

$$\theta_{i,k} + d_i + t_{i,j} \leq \theta_{j,k} + B(1 - x_{i,j,k}), \quad \forall i \in \mathcal{V}^o, \forall j \in \mathcal{V}^d, \forall k \in \mathcal{K}, i \neq j \tag{6.8}$$

$$\theta_{i,k} + d_i + t_{i,j} \leq A_{j,k} + B(1 - x_{i,j,k}), \quad \forall i \in \mathcal{V}^o, \forall j \in \mathcal{V}^d, \forall k \in \mathcal{K}, i \neq j \tag{6.9}$$

$$\theta_{i,k} + d_i + t_{i,j} + B(1 - x_{i,j,k}) \geq A_{j,k}, \quad \forall i \in \mathcal{V}^o, \forall j \in \mathcal{V}^d, \forall k \in \mathcal{K}, i \neq j \tag{6.10}$$

$$\theta_{i,k} = max(A_{i,k}, a_i), \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{K} \tag{6.11}$$

$$\theta_{i,k} = A_{i,k} + W_{i,k}, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{K} \tag{6.12}$$

$$a_i \leq \theta_{i,k} \leq b_i + p_{i,k}, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{K} \tag{6.13}$$

$$e_p - pe_{i,p} \leq \theta_{i,k} + B(1 - \lambda i, p), \quad \forall i \in \mathcal{O}, \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{6.14}$$

$$\theta_{i,k} + d_i \leq l_p + pl_{i,p} + B(1 - \lambda i, p), \quad \forall i \in \mathcal{O}, \forall k \in \mathcal{K}, \forall p \in \mathcal{P} \tag{6.15}$$

$$\sum_{p=1}^{P} \lambda_{i,p} = 1, \quad \forall i \in \mathcal{O} \tag{6.16}$$

$$\theta_{i,k}, p_{i,k}, \rho_{i,k}, pe_{i,p}, pl_{i,p} \geq 0, \quad x_{i,j,k}, \lambda i, p \in \{0,1\} \tag{6.17}$$

In the multi-objective case, the objective function f_1 (6.1) minimizes the total travel cost related to technicians routing and the production losses when not using a production stoppage time window. Meanwhile, the objective function f_2 (6.2) minimizes the total maintenance cost and the penalty cost of not respecting the maintenance time windows. Constraints (6.3, 6.4) indicate that each operation must be performed only once. Constraints (6.5,6.7,6.6) ensure that each team of technicians has to leave the departure node 0; if a team of technicians arrives on a vertex i to do a maintenance operation, it should leave it; and finally, teams of technicians must arrive after executing the last operation in the tour at the arrival node $n+1$. The purpose of the constraints (6.8) is to prevent the creation of sub-tours. Constraints (6.13) assure that each operation is performed within its time window. Note that we assume that $a_0 = 0$ and b_{n+1} represents the latest time for technicians to return to the arrival depot. The violation of the closing time of the time window is allowed. Furthermore, the amount of violation is measured in the variable $p_{i,k}$. Constraints (6.14, 6.15) ensure that each maintenance operation and its execution must be performed within a production stoppage time window. These constraints are soft. Penalties of earliness $pe_{i,p}$ and tardiness $pl_{i,p}$ related to production losses are calculated if PM operations are fully or partially performed outside of production stoppages intervals. Constraints (6.16) force the choice of one appropriate production stoppage interval for each maintenance operation. Constraints (6.9),(6.10) define the value of the arrival time. Constraints (6.12) calculate the value of the waiting time. Constraints (6.11) force the technicians to start the maintenance operation as soon as they arrive. Finally, the constraints (6.17) specify the domain values of decision variables.

6.4 Proposed multi-objective algorithms

The multi-objective version of the problem treated is a Mixed Integer Non-Linear program (MINLP). The problem comprises the VRPTW and the maintenance scheduling problem using OR techniques which are both NP-hard [86]. The production features in the objectives and constraints and considering the multi-objective version of the problem is more computationally expensive since the problem more extensive.

In the rest of the section, we propose two multi-objective algorithms to deal with the above problem. The first one is based on Pareto Local Search and Variable Neighborhood Descent. The second one is based on Adaptive Large Neighborhood Search and Variable Neighborhood Descent. Adaptive Large Neighborhood Search (ALNS) heuristic is an enhancement of the Large Neighborhood Search (LNS) method proposed by Shaw in 1998 [120]. ALNS was first proposed by Ropke and Pisinger for the pickup and delivery problem in 2006 [115]. In the LNS method, a single destroy and repair operator is chosen to be used throughout all iterations of the search [120]. The main drawback of LNS is that it is unknown in advance which method is suitable for a given instance. Moreover, the performance of a method varies during the search. Some methods can be suitable for the first iterations, while others are more efficient in later iterations. To overcome this problem, ALNS allows the selection of many destroy and repair operators. The algorithm then assigns a weight to each operator depending on its success in the previous iterations. Past success is the indicator of future success. During the runtime, these weights are adjusted. Pareto Local Search (PLS) is a recent method proposed in 2004 by Paquette et al. in [69] for the bi-objective Traveling Salesman Problem. The authors proposed a general local search approach to multi-objective problems. It is therefore considered the first reference multi-objective local search-based method. It is very used in the field due to its simplicity and performance.

6.4.1 Multi-objective optimization

Like many other real-world problems, this problem requires optimizing two conflicting objective. There is no optimal solution for all the objectives simultaneously, or that dominates all the objectives. In a posteriori optimization, when the decision-maker does not intervene or intervene last, a set of Pareto optimal solutions needs to be found. They are solutions that offer several optimal compromises between the objectives. The solution s is said to be a Pareto optimal or efficient solution if there is no feasible solution $s^{'}$ that dominates it. The projection of Pareto optimal solutions in the objective space is called the Pareto front. It is composed of several non-dominated points. Without loss of generality, for a problem with objectives that need to be all minimized, we define the following concept:
Pareto optimal dominance: The solution s dominates another solution $s^{'}$ or $f_i(s) \prec f_i(s^{'})$ if these both conditions are verified $\forall i \in \{1, 2, \ldots, k\}, f_i(s) \leq f_i(s^{'})$ and $\exists j \in \{1, 2, \ldots, k\}, f_j(s) < f_j(s^{'})$ where k is the number of objectives to be minimized.
Incomparable solutions: Two solutions s and $s^{'}$ are incomparable if no solution dominates the other. This can be denoted by $f_i(s) \nprec f_i(s^{'})$ and $f_i(s^{'}) \nprec f_i(s)$

6.4.2 Solution representation and constraints handling

A solution is represented with a number of K tours, each of which is a sequence of PM operations. The number of tours is reduced during the search. Only feasible solutions are accepted. The constraints are hard apart from verifying the upper bounds of the maintenance time windows and respecting both the upper and lower bounds of the production stop time windows.

6.4.3 Local search phase

We use in all these algorithms the following neighborhood structures: swap, insert, 2 opt*, and 2opt in this order. A detailed description of the moves can be found in the previous contributions chapters 3 and 4. In our local search, a solution $s^{'}$ is better than another solution s if $f(s^{'}) \prec f(s)$ or $(f(s^{'}) \nprec f(s)$ and

$f(s) \prec f(s^{'})$). If $s^{'}$ dominates s or is incomparable with it, the set A is updated to include $s^{'}$.

6.4.4 Pareto Local Search- Multi-objective Variable Neighborhood Descent (PLS-MOVND)

We propose in this section a hybridizing between the Pareto Local Search algorithm (PLS) and an adaptation of the Variable Neighborhood Descent to the multi-objective case. Indeed, it includes some of the design properties of the MOVND/PI proposed in Chapter 4. The VND is a search that explores multiple neighborhoods and defines an order of exploration depending on the neighborhood's performance.

Algorithm 14 describes the proposed PLS-MOVND algorithm. At each iteration and while the solution s is not explored (steps 6– 29), the current neighborhood P is generated and then it is entirely explored with the PLS best improvement strategy.

The new solution x is only accepted to be included in the current neighborhood P of s if it is not dominated by s. It is a speed-up technique used by the Paquete et al. [69] in step 10. Therefore, in the local search, we always test if the new solution x is non-dominated by s. Each point in P generated that is not in archive A is evaluated to enter this latter or not with the PLS best improvement strategy (steps 12–19). The PLS best improvement strategy defined for the first time by Paquete et al. [69] consists of accepting the new solution x if it is not dominated by any solution x' in the archive A (steps 12–19).

The designed multi-objective neighborhood change procedure proposed is novel. We consider that a neighborhood N_l improves the archive A if all the solutions in P are non-dominated by any solution in the archive A. Indeed, an improvement means that all new points have been added to the archive A. The counter $counterAI$ is incremented to record each improvement.

Another new characteristic was also included in the neighborhood change procedure. We stay in the first neighborhood if all the solutions in P are non-dominated by any solution in the archive A but also when $counter$ did not reach $iterC_{max}$. Otherwise, we move to the next neighborhood. The counter $counter$ is

used to control the number of iterations before moving to the next neighborhood, even though the current one is improving. Moving to the next neighborhood is compulsory if $iterC_{max}$ is attained. It aims to avoid not exploring the other neighborhoods when the first neighborhood is constantly improving. An efficient mix between staying in the improving neighborhood and moving to explore the following neighborhoods can be achieved through this procedure. However, in most cases, all the neighborhoods are explored sequentially.

This algorithm has been designed to solve multi-objective thesis problems. It is simple and parameter-free, which makes it easy to test and use on any combinatorial optimization problem. Furthermore, its simplicity and rapidity permit it to be used and included in other metaheuristics within a multi-phase framework to enhance the latter's performance.

6.4.5 Multi-objective Variable Neighborhood Descent based on Pareto dominance (MOVND/P)

This algorithm has been presented in detail in Chapter 4. It is designed to be a multi-objective local search component or intensification algorithm. Indeed, it has been used in the multi-objective General Variable Neighborhood Search algorithms proposed in Chapter 4. The algorithm is based on the general principle of exploring several neighborhoods when there is still an improvement. We have used the counter $counterAISecond$ to measure if there is an improvement compared to the counter of the previous iteration $counterAISecondPrevIt$. Using the counters here constitutes a speed-up technique compared to simply storing the archive, which would have the same effect. For each neighborhood structure, the neighbors are explored using the MOBI/P procedure proposed in Chapter 4 that checks if each neighbor of s is non-dominated with the best solution found so far in the neighborhood of s. For each element x of the set P generated with the MOBI/P procedure, solution x is included in the archive if it is non-dominated by s and replaces the current solution s. The $counterAISecond$ is used to measure if some solutions have been added to the archive from one iteration to the next iteration to continue running the algorithm while there is still an improvement. There is, however, a stopping condition on the number of

Algorithm 14 : Pareto Local Search -Multi-objective VND (PLS-MOVND).

Data : l_{max}: number of neighborhood structures;
s_0: initial solution;
Result : A: a set of potentially efficient solutions;

$s_0 \leftarrow bestInsertionHeuristic(0.5, 0.5)$;
$s \leftarrow s_0$;
$addSolution(s_0, A)$;
$P \leftarrow \emptyset$;
$iteration \leftarrow 1$;
while $(s \in NotExploredSolution(A)) \wedge (iteration \leq iter_{max})$ **do**
    $counter \leftarrow 1$;
    $l \leftarrow 1$;
    while $l \leq l_{max}$ **do**
        $P \leftarrow Neighborhood(s, l)$;
        $counterAI \leftarrow 0$;
        foreach $x \in P$ **do**
            foreach $x' \in A$ **do**
                if $x \notin A \wedge xnon - dominatedbyx'$ **then**
                    $addSolution(x, A)$;
                    $counterAI \leftarrow counterAI + 1$;
                end
            end
        end
        $counter \leftarrow counter + 1$;
        if $(counterAI = size(P)) \wedge (counter \leq iterC_{max})$ **then**
            $l \leftarrow 1$;
        else
            $l \leftarrow l + 1$;
        end
    end
    $s \leftarrow selectRandomNotExploredSolution(A)$;
    $iteration \leftarrow iteration + 1$;
end

iterations $iter_{max}$ to avoid extensive computational time. In the multi-objective neighborhood change procedure, we stay in the best neighborhood when all the solutions have been added to the archive, which is an excellent improvement. The $counterAI$ records how many solutions from the set P have been added to the archive A of efficient solutions whenever there is a new exploration of the neighborhood-generated P. Note that the content of Chapter 4 has also been published in our work [99].

6.4.6 ALNS operators

Destroy operators

They destroy part of the solution. ALNS generally uses a high degree of destruction d to explore a large neighborhood. Destroy operators in ALNS are the main components that permit new solutions evocation. They affect search process diversification.

Highest cost removal operator removes d operations with the highest cost. The operator aims to remove tasks that worsen the objective function more so that in the recreating phase, the repair operators can reinsert them in other positions to reduce the huge costs. For the routing objective function, we calculate the difference between the cost when the maintenance operation is in the solution and the cost when it is removed. Precisely, for each task k located between operation i and operation j, the saving value $c_{ik} + c_{kj} - c_{ij}$ is calculated. For the maintenance objective, the cost of each operation is obtained using the start times. The penalties, added in both cases, are incurred when maintenance intervals are not respected and when the production stoppages intervals are not efficiently used. Then, the array of unrouted operations is sorted in increasing order of the total cost values, and the d last tasks are removed from the solution.

Highest Risk removal operator is a newly defined operator designed for the specific problem we deal with, mainly when the problem includes a failure cost that we proposed in [98] and in Chapter 3. This operator removes maintenance operations with the highest probability of failure and risk of breakdown. It performs well for the routing, failure, and maintenance objectives. We suppose in that operator that tasks with the highest risk of breakdown should be removed, so

they are rescheduled at more appropriate times to avoid machines' breakdowns.

Related removal has been for the first time proposed by Shaw [127]. It removes the similar parts of the solution, admitting they are easy to interchange, so the repair operator reinserts them better. The operations are more related when the relatedness measure $R(i,j)$ is lower. The definition of relatedness depends on the problem at hand. The relatedness in our algorithm comprises two terms: a time term and a travel time term (or distance term). We define relatedness as follows:

$$R(i,j) = \alpha \left|\theta_{ik} - \theta_{jk}\right| + \beta t_{ij}, \quad \forall i,j \in \mathcal{O}\ \forall k \in \mathcal{K} \tag{6.18}$$

Where θ_{ik} is the start time of task i realized by the team of technicians k, and t_{ij} is the travel time between i and j. To consider equal contributions of the two terms, (α,β)=(0.5,0.5).

Time oriented removal operator is a variant of related removal. It includes only the first part related to time (α,β)=(1,0). Operations performed at the same time are removed.

Distance oriented removal operator is another variant of related removal that includes only the travel time part or distance part (α,β)=(0,1). Operations close to the chosen task are removed.

Repair Heuristics

They perform the insertion of the parts of the solutions destroyed by ruin operators as long as feasible insertions are possible. Repair heuristics are divided into two categories: sequential insertion heuristics and parallel insertion heuristics. In sequential heuristics, sub-problems are dealt with one by one, and in parallel heuristics, all sub-problems are dealt with at the same time [115]. For the case of the VRP problem, sequential heuristics consider one route at a time, while parallel heuristics consider many routes at the same time. All the repair operators proposed here are parallel.

Basic greedy insertion inserts a randomly selected unrouted operation in the position that minimizes the insertion cost until there are no remaining operations

to be inserted. The insertion cost represents the difference between the cost of the solution with the inserted operation and the solution's cost without it.

Best insertion is a variant of the basic greedy insertion where the order of insertion of the unrouted operations is considered to achieve better solutions. It consumes, however, more time compared to the previous operator. Therefore, all possible insertions of all unrouted operations are evaluated at each iteration. The operation with the best insertion cost is then selected to be inserted at its best position.

Two regret insertion use the regret value instead of the cost value to insert operations. The two regret value represents the total cost difference between the best insertion position and the second best. Operations with high regret value are the first inserted after ordering the operations. For example, among a set of unrouted operations $\mathcal{U}$, this repair operator inserts an operation i according to:

$$i = argmax(\Delta f_i^2 - \Delta f_i^1), \quad \forall i \in \mathcal{U} \tag{6.19}$$

The regret value uses all positions in the same route or different routes. This approach has been used also by [125]. Ropke and Pisinger [115] use only positions in different routes, which is a different approach. When a task is inserted, the insertion positions of the remaining unrouted operations are recalculated by considering the change caused by inserting this operation at this position.

6.4.7 Update of weights

A weight w_h is associated to each destroy or repair operator h to save its past performance of improving a current solution. The number of times the operator h is used during the search is denoted by u_h. The better the heuristic performs the higher its weight. We note s_h the accumulated success score of the operator h. The vectors s_h and u_h are initialized with zero at the beginning of the search. In each iteration, two heuristics are selected to create a solution s' from a current solution s. An amount δ_i where $i = 1, 2, 3$ is then added to increase the score s_h depending on the following cases:

- δ_1: $f(s') < f(s_{best})$, the new solution s' improves the best solution s_{best}.

- δ_2: $f(s_{best}) \leq f(s') < f(s)$, the new solution s' improves only the current solution s.

- δ_3: $f(s) \leq f(s')$, the new solution s' is accepted even though it is not better than the current solution s.

The inequality $\delta_1 > \delta_2 > \delta_3$ have to be ensured for the weights adjustments. This could change to ensure a different control during the search process. Ropke and Pisinger [113] have for instance chosen $\delta_1 > \delta_3 > \delta_2$.

The weights are updated for the next iteration using the following formula:

$$w_h = \begin{cases} (1-\rho)w_h + \rho * \frac{s_h}{u_h} & \text{if } u_h > 0 \\ (1-\rho)w_h & \text{if } u_h = 0 \end{cases} \tag{6.20}$$

The reaction factor $0 \leq \rho \leq 1$ indicates if the emphasis is placed on the success of the recent iterations. With $\rho = 0$, the weights remain at their initial level. In this case, the probabilities never change. Suppose $\rho = 1$, the success of the last iteration is considered. We select the reaction factor such as $0 < \rho < 1$ to consider the operators' past performance throughout the whole search process.

The values of the probabilities are obtained using the weights. Naturally, like for the weights, operators that perform well have higher probabilities. The probability of choosing a repair method p_r and a destroy method p_d is given by:

$$p_r = \frac{w_r}{\sum_{r=1}^{R} w_r}, \quad p_d = \frac{w_d}{\sum_{d=1}^{D} w_d} \tag{6.21}$$

6.4.8 Acceptance criterion: Multi-objective Simulated Annealing

The main advantage of metaheuristics compared to local search is allowing the search to move to worse solutions to introduce some diversification and avoid the local optima trap. The Simulated Annealing metaheuristic uses this strategy inspired by the physical annealing process. Better solutions are accepted but also worse solutions with a certain probability. The solution s' is accepted even if do not improve the current solution s to ensure diversification according

to the probability $min(1, exp(-(\frac{\Delta E}{T})))$. ΔE represents the energetic variation. The temperature T decreases with a factor α at each iteration, ensuring that the probability of accepting worse solutions decreases when the algorithm is executed. This process ensures that worse solutions are less likely to be accepted late in the process. This acceptance rule consists in accepting only improving solutions with certainty. Many other probabilistic acceptance strategies for multi-objective simulated annealing have been proposed in the literature [121]. In the beginning, T is initialized with the value of the total cost $f(s_0)$ of the initial solution s_0 as follows:

$$T = \frac{-w_T}{ln(0.5)} \times f(s_0) \tag{6.22}$$

The energetic variation is ΔE for multi-objective simulated annealing. For a bi-objective problem, f_1 and f_2 represents the objective function, s' the new solution and s the current solution. ΔE can be expressed as follows:

$$\Delta E = w_1 \times (f_1(s') - f_1(s)) + w_2 \times (f_2(s') - f_2(s)) \tag{6.23}$$

Similarly to Ropke and Pisinger [113], we set $w_T = 0.05$ so that a solution that is 5% percent worse than $f(s_0)$ is accepted with a probability of 0.5. The cooling factor of the temperature T is fixed to $\alpha = 0.99975$.

6.4.9 Multi-Objective Semi-Adaptive Large Search based on Pareto dominance (MOSALNS/P)

Algorithm 15 the proposed ALNS-based algorithm. At each iteration of MOSALNS/P (steps 7– 50) and in step 8, three counters recording the improvement that occurs during the search are set to their value of the previous iteration to test their increase later in the algorithm. In step 8, these counters are stored to be used in the selection of the solution s in each iteration. They are incremented according to the improvement in the algorithm. Their use is a new characteristic of that latter.

A semi adaptive mechanism is applied in steps (9-13). It is an adaptation of our semi-adaptive mechanism used in the SALNS algorithm published

in [66] and Chapter 5. It consists of using the selection based on past performances using the roulette wheel used by [115] only if there is an improvement. Otherwise, the selection is random to ensure more diversification. It is also an opportune time for the algorithm to learn. For the multi-objective case, if the new solution dominates the best solution s_{best}, $counterD$ is incremented and if the new solution is incomparable with the current solution s, the $counterInComp$ is incremented. If there is no improvement in whether the solution is accepted with a probability or not, $counterNI$ is incremented. The selection is based on the roulette wheel mechanism if $counterD \geq counterNI$ or if $counterInComp \geq counterNI$, which means if the number of times we have obtained non-dominated solutions is superior to the number of times we have not. If there is no improvement, a random selection occurs that can be assimilated to a more guided shaking mechanism since the perturbation occurs when needed. This random selection assures an additional diversification with the simulated annealing criteria. It is time for the learning to have enough time to learn from random selection in the environment. Probabilities of the heuristics are also updated every iteration by new information revealed. Learning occurs each iteration, but the selection according to the probabilities happens only if a new non-dominated solution has been added to the archive A. The proposed mechanism seeks to alternate between diversification and learning.

A destroying operator removes d operations from the solution, and a repair operator is later used to insert the removed operations into the current solution. An intensification step of the current solution using the VND algorithms follows then (steps 15-20). Steps (15 -20) aim to speed up the search while applying two categories of local search. In the first iterations of the algorithm, only a single objective VND using the weighted sum aggregation is applied to converge rapidly. The non-dominated solutions obtained through this search are stored in P. In this step, not all the solutions of the Pareto front are obtained. In the later iterations of the search, the following steps occur. The intensification is realized using a multi-objective method MOVND/P proposed in Chapter 4 and published in [99]. A single objective VND leads the search to rapidly converge toward the front. Then at this time, we explore the neighborhood of the new solution yielded to constitute a potentially more complete set P of non-dominated solutions. The

pseudo-code of the mono-objective VND can be found in [98] and Chapter 3. This new strategy uses a single objective VND to converge rapidly to a good approximation of the Pareto Front (PF) in the first iterations of the search. This approximation is constituted of convex points of the PF. The MOVND/P phase completes the archive with the remaining neighbors in the last iterations to obtain a complete PF.

Three cases are distinguished when exploring each point of the set P. If the solution x dominates the best solution found so far s_{best} (steps 25-28), it replaces s_{best} and the current solution s and is added to the archive A by means of the $addSolution$ method presented in Chapter 4 and [99]. The score is increased by δ_1, and $counterD$ is increased. This replacement of the best solution s_{best} by a solution that dominates it is another new feature used. If the new solution x is incomparable with the current solution s (steps 30-32), it is added to the archive but similarly to [90], the score is increased by an amount δ_2 with $\delta_2 < \delta_1$. The solution is added to the archive, and $counterInComp$ is increased. If the new solution x is dominated (steps 33-38), it can be accepted although with a given probability. We have defined two types of archives, the archive A that stores the non-dominated solutions obtained during the search and A' which stores the non-dominated solutions obtained by the simulated annealing from the solutions that are worse but are accepted with the defined probability. A multi-objective simulated annealing criterion is used, and the obtained solutions are stored in A' to be used later to ensure more diversification. Moreover, the score is increased by an amount δ_3 with $\delta_3 < \delta_2 < \delta_1$ and $counterNI$ is incremented. If none of these three cases hold, $counterNI$ is incremented.

In each iteration, temperature (step 42) and probabilities 43 are updated. The new procedure for selecting a solution s for the next iteration is described in the following (steps 44-48). The random selection of the following solution s in each iteration depends on whether the solution has succeeded in yielding at least one non-dominant solution in the previous iteration. In this case, the next solution is chosen from the archive A. If not, we select the next solution s among the dominated accepted solutions of the archive A' by the simulated annealing criteria. When the stopping criterion is met, the archive A contains the efficient non-dominated solutions.

Algorithm 15 : Muti-objective SALNS based on Pareto Dominance (MOSALNS/P).

Data : l_{max}: number of neighborhood structures in MOVND/P ;
Result : A : a set of potentially efficient solutions

$s_0 \leftarrow bestInsertionHeuristic(0.5, 0.5)$;
$s \leftarrow s_0$, $s_{best} \leftarrow s_0$, $s^{dr} \leftarrow s_0$;
$addSolution(s, A)$,$addSolution(s, A')$;
$P \leftarrow \emptyset$, $T \leftarrow \frac{-w_T}{ln(0.5)} \times s_0$,$u_d, u_r \leftarrow 0$, $iteration \leftarrow 1$;
$w_r, w_d \leftarrow initializeWeights()$, $p_r, p_d \leftarrow initializeProbabilities(w_r, w_d)$;
$counterD \leftarrow 0$, $counterInComp \leftarrow 0$, $counterNI \leftarrow 0$;
repeat
  $counterDPrev \leftarrow counterD$, $counterInCompPrev \leftarrow counterInComp$, $counterNIPrev \leftarrow counterNI$;
  if $counterD \geq counterNI \vee counterInComp \geq counterNI$ **then**
    Select $r \in R$, $d \in D$ using roulette wheel mechanism;
  else
    Select randomly $r \in R$, $d \in D$;
  end
  $s^{dr} \leftarrow r(d(s))$;
  if $iteration \leq F * iter_{max}$ **then**
    $s^{'} \leftarrow VND(s^{dr}, l_{max})$;
    $addSolution(s^{'}, P)$;
  else
    $P \leftarrow MOVND/P(s^{dr}, l_{max})$;
  end
  $u_d \leftarrow u_d + 1$, $u_r \leftarrow u_r + 1$,$random \sim U_c[0, 1]$;
  $counterD \leftarrow 0$, $counterInComp \leftarrow 0$, $counterNI \leftarrow 0$;
  $s_{best} \leftarrow s$;
  foreach $(x \in P)$ **do**
    if $f(x) \prec f(s_{best})$ **then**
      $s_{best} \leftarrow x$;
      $addSolution(x, A)$;
      $s_d, s_r \leftarrow s_d, s_r + \delta_1$, $counterD \leftarrow counterD + 1$;
    else
      if $f(x) \not\prec f(s) \wedge f(s) \not\prec f(x)$ **then**
        $addSolution(x, A)$;
        $s_d, s_r \leftarrow s_d, s_r + \delta_2$,$counterInComp \leftarrow counterInComp + 1$;
      else
        $\Delta E = w_1 \times (f_1(x) - f_1(s)) + w_2 \times (f_2(x) - f_2(s))$;
        if $random < min(1, exp(-(\frac{\Delta E}{T})))$ **then**
          $s_d, s_r \leftarrow s_d, s_r + \delta_3$,$counterNI \leftarrow counterNI + 1$, $addSolution(x, A')$;
        end
        $counterNI \leftarrow counterNI + 1$;
      end
    end
  end
  $T \leftarrow T \times \alpha$;
  Adjust the weights w_r and w_d and the probabilities p_r and p_d of the heuristics ;
  if $(counterDPrev \leq counterD) \vee (counterInCompPrev \leq counterInComp)$ **then**
    $s \leftarrow selectRandomNotExploredSolution(A)$;
  else
    $s \leftarrow selectRandomNotExploredSolution(A')$;
  end
  $iteration \leftarrow iteration + 1$;
until *maximum number of iterations maxIter*;

6.4.10 Novelty in the proposed algorithms

Novelty in the PLS-MOVND algorithm

The PLS-MOVND algorithm extends the PLS method proposed by Paquete et al. [69] by exploring several neighborhood structures. In addition, it is combined with some design features of MOVND/PI presented in Chapter 4. Our case considers an improvement if all the solutions explored have been added to the archive. Duarte et al. [93] for their part considered an improvement when at least one new point has been added to the archive A. The neighborhood change procedure consists of staying in the best neighborhood if all the solutions explored are non-dominated by s. We also stay in this neighborhood when the defined counter does not reach a maximum number of iterations to avoid not exploring the other neighborhoods when the current neighborhood is still improving. In the literature [93], the exploration continues in the same neighborhood structure when there is an improvement. Queiroz and Mundim [94] changed the neighborhood systematically in an adapted template of multi-objective VND from [93] to reduce the computational time.

Novelty in the MOSALNS/P algorithm

The MOSALNS/P algorithm has several new characteristics we incorporated in the search. These new features first include using counters all algorithm long to measure the muti-objective improvement. They distinguish the dominating and incomparable solutions and the case where no such solutions are found. The semi-adaptive mechanism and using a single objective local search VND directly followed in later iterations by the proposed multi-objective local search MOVND/P are also new characteristics. Such an approach aims to rapidly converge toward the PF in the first iterations of the search and then constitute a complete PF later in the search. Furthermore, the MOVND/P algorithm uses a novel proposed multi-objective best improvement strategy MOBI/P that we presented in Chapter 4. Three cases are distinguished when exploring each point of the set P. The three cases have also been adopted by Cota et al. [90]. The authors, however, compared the current solution with s at each iteration without

defining the concept of the best solution s_{best} in the multi-objective perspective in the first case. We replace the best solution s_{best} with a new solution that dominates it. It is another new feature that has not been used by Cota et al. [90]. The classical mono-objective ALNS, unlike the other mono-objective algorithm, clearly distinguishes between the best solution s_{best} and a current solution s [115]. In the last case, there is also no use of an additional archive A' of worse and accepted solutions using simulated annealing. Moreover, in every case, the authors in the literature did not use counters to measure the multi-objective improvement. Finally, another novelty of the algorithm is the selection from two archives for the subsequent iterations.

6.5 Computational experiments

The following sections present the methodology of instances' generation, the parameter values used in all the proposed and comparative algorithms, the performance metrics used, and finally, provide test results for the multi-objective problem. This latter is solved using the proposed methods PLS-MOVND, MOVND/P, MOVND/PI, and MOSLANS/P and then the comparative algorithm. Results are reported for small, medium, and large size instances. The maintenance model was implemented with Python 3.6 in a Linux environment to get the expected start time, frequency, and time windows for each PM operation. The combined maintenance scheduling and workforce routing problem with production stoppages considerations was implemented with C++ and compiled with GCC 7.4 in a Linux environment. The Concert Technology library of CPLEX 12.10.0 version with default settings in C++ has been used to solve the MIP using an exact method. Experiments were conducted using the CALCULCO computing platform in an AMD EPYC 7702 with 2CPU, 2 gigahertz, and 1 core was dedicated to each instance. All methods are run using the same equipment to avoid bias.

6.5.1 Generation of instances

The instances were generated as described in the previous chapter 5. The same data have been used for the multi-objective variant of the problem.

6.5.2 Parameter settings, performance metrics, and comparison procedure

The parameter settings for the multi-objective problem are described below. They have been set according to the results of extensive computational experiments. The stopping criterion of the algorithms is fixed according to the number of operations n. Larger values of the number of operations will lead to a better Pareto front approximation in excessive computational time. The parameter values presented here provide a good compromise between the solution quality and computational time. The stopping condition for the MOVND/P, MOVND/PI, and PLS-MOVND is the maximum number of iterations $iter_{max}$. Another defined stopping criterion is related to the neighborhood change $iter_{Cmax}$. The parameter $iter_{max}$=$2 \times n$ for PLS-MOVND and $iter_{max}$=$1.75 \times n$ for MOVND/PI, the improved version of MOVND/P. Results of MOVND/PI with an $iter_{max}$ that equals $2 \times n$ were displayed; however, all tests and computational comparisons with the other algorithms were realized with the value of $1.75 \times n$. This value equals $iter_{max} = max(1, E[n/2])$ for MOVND/P. The second defined stopping criterion determining the time of staying in the improving neighborhood in the neighborhood change procedure is $iterC_{max} = max(1, E[n/16])$ for the three proposed algorithms MOVND/P, MOVND/PI, and PLS-MOVND. The proposed algorithm MOSALNS/P is stopped when the maximum number of iterations attains $iter_{max} = max(1, E[7n/8])$. Factor F is set to 0.25, so the VND is applied only for the $0.25 \times iter_{max}$ first iterations. All the other specific MOSALNS/P parameters, such as the rewards, the cooling factor, and the degree of destruction are set to the same values chosen for the mono-objective SALNS of Chapter 5. The operators used are also the same. The symbol E again here stands for the integer portion of the number.

The comparative algorithm MO-ALNS of Cota et al. [90] is used to assess our

multi-objective algorithms' performance for the defined problem. The maximum number of iterations is fixed for this algorithm to $iter_{max} = 15 \times n$. The algorithm uses the first improvement strategy, which is rapid in each iteration but fails to form a complete front with a small number of iterations. For each $10 \times n$ iteration, only a few points are added to the set of efficient solutions. It is a learning automata-based algorithm for which the rewards are set to $a_1 = 0.2$, $a_2 = 0.1$ and $a_3 = 0.05$ and the penalty to $b_1 = 0.02$ when the new solution is not accepted with the simulated annealing criteria. The values of the rewards and the penalty are the same as those adopted by Cota et al. in their works [114] [90]. The proposed algorithm MOSALNS/P and the comparison algorithm use the same neighborhood structures, removal, and insertion operators and start with the same initial solution. We have already shown in our previous work in Chapter 5 that mono-objective ALNS with learning automata has approximately the same performance in terms of solution quality as SALNS with a slight improvement of CPU time in favor of our SALNS algorithm. The reason is that the semi-adaptive mechanism we defined assures an additional diversification. We concluded that the more the algorithm diversifies, the less computational time. The update period of the MO-ALNS with learning automata is equal to $E[7.5 \times n]$).

These two algorithms differ entirely in the general design and the exploration strategy. However, they are based on Pareto dominance and use the same choices related to the problem's specifications (the same destroy and repair operators, local search operators, and initial solution).

In multiple objective contexts, comparing the methods can be tricky since it involves a whole Pareto Front instead of only one solution. Many criteria should be assessed to evaluate solutions fully. The solution quality in the mono-objective context can be mainly evaluated using convergence and maybe robustness indicators. The solution is a Pareto front in a multi-objective problem. The convergence and the coverage of the set of efficient solutions should be discussed. The computational effort is also a decisive factor for both types of problems, especially in heuristic methods. A series of experiments need to be carried out to find the best trade-off between CPU time and solution quality. They exist more than 20 indicators for multi-objective optimization that deal with the convergence only, the coverage only, or both at the same time. Fonseca et al. [119] have selected

four as the most discriminant indicators to compare stochastic multi-objective methods. They are the hypervolume, the unary epsilon, the r indicator based on using a set of utility functions, and the coverage indicator. Few researchers used all these indicators at the same time. Most authors pick the most representative of all aspects of solution quality. Our study chooses to assess solution quality with the most important of all these indicators, the hypervolume. The particularity of this indicator is that it is the one that is present in all papers dealing with multi-objective optimization since it jointly assesses convergence and coverage. It is also the most representative of all indicators. It also has the advantage of measuring the quality of the front without the need for another comparison method or the exact Pareto front. We also choose to display the number of non-dominated points even if it can be inferred from the results of the hypervolume indicator and, finally, the CPU time. Indeed, the HV indicator is more significant whenever the Pareto front is better in terms of convergence, coverage, and the number of non-dominated solutions.

The hypervolume (HV) indicator is a metric that measures the size of the space covered between the set of efficient solutions obtained and a reference point. The reference point (f_1^{rp}, f_2^{rp}) is a point that is dominated by all solutions of the Pareto front of all the algorithms tested. Assuming, we are dealing with a minimization problem, the reference point rp is (f_1^{max}, f_2^{max}) where f_1^{max} and f_2^{max} are respectively the maximum values of the first and second objectives. More formally, the size of a rectangular area a_i between a solution s_i and the reference point rp is $hv_i = (f_1^{rp} - f_1(s_i)) * (f_2^{rp} - f_2(s_i))$. The hypervolume represents the sum of the areas where p is the number of solutions s_i in the Pareto front A:

$$hv = \sum_{i=1}^{p} (f_1^{max} - f_1(s_i)) * (f_2^{max} - f_2(s_i)), \quad \forall s_i \in \mathcal{A} \tag{6.24}$$

The improvement of our algorithms over the literature algorithm is evaluated using the following three indicators that measure the percent increase or decrease, thus improvement when using the proposed algorithms PA compared to the comparison algorithm CA of [90].

- The percent increase of the HV indicator is:

$$IHV(CA,PA) = (\frac{HV(PA) - HV(CA)}{HV(CA)}) \times 100\% \tag{6.25}$$

- The percent increase of the number of non-dominated points is calculated as follows:

$$INDP(CA,PA) = (\frac{NDP(PA) - NDP(CA)}{NDP(CA)}) \times 100\% \tag{6.26}$$

- The percent decrease of the CPU time is given by:

$$ICPU(CA,PA) = (\frac{Time(CA) - Time(PA)}{Time(CA)}) \times 100\% \tag{6.27}$$

6.5.3 Tests results

Comparison with the literature

We compare the approximated Pareto front returned by all the methods using the two indicators mentioned above and the running time consumed by each algorithm to return a Pareto front. We tested all the instances in this study that generated less or equal to 65 operations for all multi-objective algorithms. Larges instances stop up to 65 operations. The detailed results of the PLS-MOVND and MOVND/PI algorithms are shown in Table 6.1. The results of MOVND/P, MOSALNS/P, and the result of the MO-ALNS algorithm of Cota et al. [90] are presented in detail in Table 6.2. For all these 36 tested instances, PLS-MOVND increases the hypervolume (HV) and the number of non-dominated points by respectively 845.10% and 775.02%. The CPU was reduced by 79.98% on average compared to the MO-ALNS algorithm of Cota et al. [90]. MOVND/PI, with a maximum number of iterations of $1.75 \times n$, which is the value used in all these tests, improves the HV and NDP by respectively 876.27% and 817.14% and the CPU by 79.85%. MOSALNS/P improves the MO-ALNS of [90] for respectively

the HV, the NDP, and the CPU time with an average of 567.84%, 511.48% and 49.79%. The detailed results for each instance of the percent improvement of these algorithms over the algorithm in [90] are displayed in Table 6.3.

All the proposed algorithms PLS-MOVND, MOVND/PI, and MOSALNS/P considerably outperforms MO-ALNS suggested by Cota et al. [90] in all the indicators assessed.

Comparison between the proposed algorithms

We conduct another comparative study of the three proposed algorithms. The results suggested that our PLS-MOVND and MOVND/PI outperform our MOSALNS/P. Indeed, PLS-MOVND outperforms on average in terms of the hypervolume, the number of non-dominated points, and the running time MOSALNS/P by respectively 62.30%, 65.01 % and 49.92%. The proposed algorithm MOVND/PI outperforms the MOSALNS/P on three indicators, HV, NDP, and CPU time, by respectively 68.34%, 70.10%, and 48.37%. The HV and NDP are improved more than the improvement realized by PLS-MOVND. The improvement in the CPU time for the two algorithms is approximately the same.

These remarks push us to compare our proposed PLS-MOVND and MOVND/PI algorithms. The results are the following: for all the tested instances, MOVND/PI improves the HV, NDP indicators, and CPU time compared to PLS-MOVND by 6.45%, 5.48% and a little more computational time since the improvement of time equals -9.60%. For the instance of the set RE, this improvement equals respectively for the HV, NDP, and CPU time 3.73 %, 4.52% and 6.72%. For all the instances based on the Solomon set, this improvement is 6.79%, 5.59%, and -11.64%. For the medium-size instances and small instances less than 34 operations, this improvement for the HV, NDP, and CPU equals -3.66%, -4.37%, and 14.65%. The HV and NDP slightly decrease, but the CPU time is reduced up to 14.65%. Therefore, we compare, for the same small and medium-size instances, MOVND/PI with a maximum number of iterations of 2n with PLS-MOVND that is already tuned to 2n. The improvement of MOVND/PI over PLS-MOVND becomes 2.79%, 1.42% and 1.30%. We then report the results of only the medium-size instances that include all instances less than 34 operations

without the RE set of instances (Solomon's set of instances). The improvement of MOVND/PI over PLS-MOVND both tuned to 2n iterations is 4.64%, 2.51% and 4.26%. For large instances where the number of operations is between 52 to 65, the improvement of MOVND/PI with $iter_{max}$=1.75n over PLS-MOVND is respectively 19.10%, 17.78% and -39.91%. The detailed results of the comparison between the proposed algorithms are displayed in Table 6.4.

On the other side, for all the instances tested, PLS-MOVND improves the HV, NDP indicators, and CPU time compared to MOVND/PI by -0.72%, 0.01%, and 2.23%. For the RE instances, this improvement equals respectively for the HV, NDP, and time -3.20 %, -4.08% and -7.76%. For all the instances based on the Solomon set, this improvement is -0.41%, 0.52%, and 3.48%. For the small and medium-sized instances with less than 34 operations, this improvement for the HV, NDP, and CPU equals 8.47%, 8.98%, and -18.19%. Since the HV and NDP improve but the CPU time is worse (-18.19%), we compare for the same small and medium-size instances PLS-MOVND that is tuned to $2 \times n$ to MOVND/PI with the same maximum number of iterations of 2n. The improvement of PLS-MOVND over MOVND/PI becomes -2.79%, -1.42%, and -2.32%. The results of the medium-size instances that include all Solomon's instances less than 34 operations demonstrate that PLS-MOVND improved MOVND/PI when both algorithms are tuned to 2n iterations by -4.64%, -2.51% and -5.03%. For large instances where the number of operations is between 52 to 65, the improvement of PLS-MOVND with $iter_{max}$=1.75n over MOVND/PI is respectively -12.21%, -11.21% and 27.76%.

We can conclude that MOVND/PI slightly outperforms PLS-MOVND for instances of small and medium-sized in quality indicators and CPU time. In contrast, it considerably outperforms both quality indicators but consumes more time. Hence, MOVND/PI is better for small and medium-size instances than PLS-MOVND.

Discussion of the results

The multi-objective study concludes that local search designed algorithms PLS-MOVND and MOVND/PI are the best algorithms. Indeed, they are considerably

better in all quality indicators and running time compared to MOSALNS/P and MO-ALNS of Cota et al. [90]. Moreover, they are both parameter-free since only two stopping criteria must be tuned. Furthermore, MOVND/PI offers another advantage that the PLS-MOVND could not offer. It has a local search component version which is MOVND/P, presented in Chapter 4 that is very fast and could be integrated into many multi-objective algorithms. For example, MOVND/P has been used in Chapter 4 to create a multi-objective Variable Neighborhood Search MOGVNS/P and here to create MOSALNS/P. The algorithms MOVND/P, MOVND/PI, and PLS-MOVND are very simple, rapid parameters free, and demonstrate outstanding performance. Thus, they can be easily integrated into other multi-phase frameworks.

The MOSALNS/P algorithm requires fixing some parameters and deciding on the appropriate destroy and repair operators in addition to the neighborhood structures. The MOVND/P, MOVND/PI, and PLS-MOVND require only the choice of the neighborhood structures to be used. Appropriate values of the parameters used can be determined through experimentation and literature. The MOSALNS/P algorithm is considerably better in all quality indicators and running time compared to MO-ALNS of Cota et al. [90] but is less performing than MOVND/PI and PLS-MOVND in quality and time indicators. These results confirm that the "less is more" approach is pertinent to optimization. This approach has been discussed [53]. Less is more approach (LIMA) affirms that using the least ingredients gives the best results. In optimization, fewer ingredients in the algorithms make them eager to provide better outcomes. Figures 6.1, 6.2, 6.3 and 6.4 show the Pareto fronts for the proposed algorithms PLS-MOVND and MOSALNS/P and of a comparative algorithm MO-ALNS suggested by Cota et al. [90]. The figures also illustrate the Pareto fronts of the MOVND/P and MOVND/PI of Chapter 4. The algorithms MOVND/PI, PLS-MOVND, MOSALNS/P, and MO-ALNS of Cota et al. [90] have approximately the same convergence. They differ, however, considerably in terms of coverage and the number of non-dominated points. The proposed algorithms MOVND/PI, PLS-MOVND, and MOSALNS/P considerably improved all the quality and time indicators in comparison with MO-ALNS of Cota et al. [90]. We also notice that PLS-MOVND and MOVND/PI have the same performance and are better in all indicators than MOSALNS/P.

Figures 6.1,6.2,6.3 clearly show what is reported in the comparison between the proposed algorithms section. MOVND/PI is better for small and medium-size instances compared to PLS-MOVND. The instances of the figures have respectively 23, 26 and 34 operations. In Figure 6.4 with 62 operations, the algorithms MOVND/PI and PLS-MOVND have approximately the same performance.

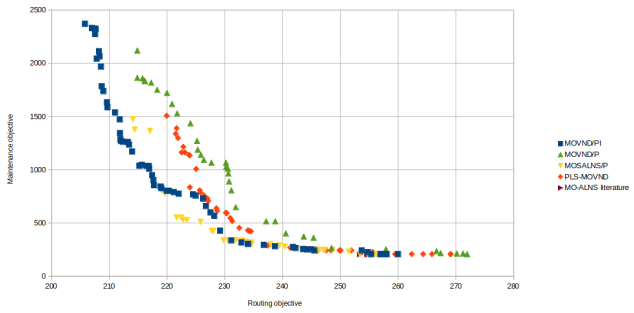

Figure 6.1: Pareto fronts of the different algorithms for the instance with 23 operations, C101/10/8/100/200/0,07/2.

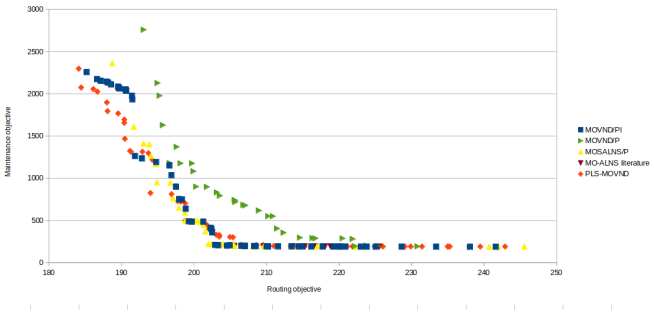

Figure 6.2: Pareto fronts of the different algorithms for the instance with 26 operations, C101/10/7/100/200/0,30.

6.6 Concluding remarks

This chapter deals with the bi-objective variant of the Joint Opportunistic Maintenance and Workforce Routing Problem defined in Chapter 5. The purpose of this chapter is to propose a mathematical model that integrates the three

Instance	*n*	PLS-MOVND			MOVND/PI 1.75n			MOVND/PI 2n		
		HV($\times 10^8$)	NDP	CPU(s)	HV ($\times 10^8$)	NDP	CPU(s)	HV($\times 10^8$)	NDP	CPU(s)
RE/6/2/80/80/0.30/2	6	0.0088	8	0.18	0.0086	8	0.18	0.0075	7	0.22
RE/6/4/101/101/0.30/2	12	0.57	21	4.55	0.62	22	3.92	0.58	21	4.49
RE/6/2/80/80/0.07/2	6	0.0088	8	0.18	0.0086	8	0.18	0.0075	7	0.22
RE/6/4/101/101/0.07/2	12	0.557	15	4.53	0.623	17	3.94	0.615	17	4.49
*C*101/10/8/100/200/0.07/2	23	3.771	43	136.7	5.98	64	103.34	4.193	47	117.33
*C*101/10/5/100/200/0.07/2	23	3.47	50	129.52	2.91	42	97.83	4.14	56	111.01
*C*101/10/7/100/200/0.07	26	17.63	58	215.51	15.90	52	168.81	25.94	83	185.44
*C*101/25/18/100/200/0.07	54	460.64	121	4127.46	437.967	115	5663.06	508.81	132	5618.51
*C*201/10/7/100/200/0.07	24	8.50	39	157.27	9.88	45	121.36	12.25	56	135.52
*C*201/25/18/100/200/0.07	52	223.87	69	3604.1	336.50	100	5070.08	310.53	94	5137.56
*R*101/10/7/100/200/0.07	34	36.88	76	619.14	23.43	47	556.2	32.62	64	600.84
*R*201/10/7/100/200/0.07	34	30.47	66	617.83	26.19	56	547.01	37.01	79	606.07
*R*201/25/18/100/200/0.07	65	526.04	102	7963.92	871.56	170	13537.8	573.35	109	14179.8
*RC*101/10/7/100/200/0.07	33	30.05	55	548.44	39.11	69	472.03	28.01	50	556.44
*RC*101/25/18/100/200/0.07	62	566.39	105	6708.53	643.23	119	9405.91	488.79	88	9444.95
*RC*201/10/7/100/200/0.07	33	32.72	62	549.26	31.70	62	511.41	40.84	78	564.71
*RC*201/25/18/100/200/0.07	60	569.67	119	5965.14	567.55	113	8442.21	578.67	118	8086.02
*C*101/10/8/100/200/0.30/2	23	4.23	51	135.89	3.01	36	103.31	4.58	54	122.54
*C*101/10/5/100/200/0.30/2	23	2.76	43	128.50	2.50	38	97.97	2.60	39	123.62
*C*101/10/7/100/200/0.30	26	17.78	57	216.55	17.97	57	168.66	12.66	40	199.8
*C*101/25/18/100/200/0.30	54	209.93	54	4111.24	349.54	90	5492.25	382.82	97	4934.17
*C*201/10/7/100/200/0.30	24	11.59	54	157.8	10.47	50	121.39	12.01	56	151.28
*C*201/25/18/100/200/0.30	52	272.34	77	3584.38	352.11	97	4137.9	320.47	87	4243.34
*R*101/10/7/100/200/0.30	34	30.29	62	620.07	30.37	63	580.17	26.43	55	671.92
*R*201/10/7/100/200/0.30	34	43.42	89	618.37	35.43	72	562.26	29.56	59	636.57
*R*201/25/18/100/200/0.30	65	506.63	93	7921.97	393.24	73	13436.7	609.81	112	14178.4
*RC*101/10/7/100/200/0.30	33	28.44	52	547.71	23.26	42	467.28	28.28	50	544.74
*RC*101/25/18/100/200/0.30	62	520.61	95	6691.48	620.80	113	9239.14	508.84	91	9240.5
*RC*201/10/7/100/200/0.30	33	38.71	79	549.89	26.74	55	511.61	35.83	71	570.02
*RC*201/25/18/100/200/0.30	60	592.56	125	5960.52	515.97	108	7908.32	453.95	95	7028.24
*C*101/10/18/200/400/0.07/2	52	106.46	82	6657.21	142.52	110	10189.3	171.98	132	10161.1
*C*101/10/18/200/400/0.07	58	328.36	96	9773.87	401.33	117	12869.1	381.71	111	14069.3
*C*201/10/18/200/400/0.07	54	258.38	84	7573.51	316.29	102	10880.3	290.60	93	10639.8
*C*101/10/18/200/400/0.30/2	52	91.08	72	6606.19	89.75	69	9703.89	118.91	91	9587.42
*C*101/10/18/200/400/0.30	58	436.76	113	9902.82	446.79	115	12371.9	414.52	106	12560.4
*C*201/10/18/200/400/0.30	54	267.97	94	7579.47	327.61	111	9042.4	351.74	119	9419.19

Table 6.1: Results of the proposed algorithms PLS-MOVND and MOVND/PI

Instance	*n*	MOVND/P			MOSALNS/P			MO-ALNS literature		
		HV($\times 10^8$)	NDP	CPU(s)	HV ($\times 10^8$)	NDP	CPU(s)	HV($\times 10^8$)	NDP	CPU(s)
RE/6/2/80/80/0.30/2	6	0.0071	7	0.06	0.0023	2	0.07	0.0033	3	0.66
RE/6/4/101/101/0.30/2	12	0.28	13	1.24	0.366	12	8.17	0.114	4	12.47
RE/6/2/80/80/0.07/2	6	0.0071	7	0.06	0.002	2	0.07	0.0033	3	0.67
RE/6/4/101/101/0.07/2	12	0.417	12	1.21	0.36	10	5.08	0.108	3	12.74
*C*101/10/8/100/200/0.07/2	23	3.10	35	30.47	3.53	40	396.41	0.47	6	529.65
*C*101/10/5/100/200/0.07/2	23	1.78	26	23.15	3.09	43	390.03	0.55	9	429.83
*C*101/10/7/100/200/0.07	26	11.44	37	35.35	17.08	55	520.46	2.62	9	852.35
*C*101/25/18/100/200/0.07	54	56.58	16	199.69	279.27	73	14415.9	79.18	22	30236.4
*C*201/10/7/100/200/0.07	24	3.06	15	16.48	7.42	35	297.23	0.59	3	615.35
*C*201/25/18/100/200/0.07	52	148.69	48	454.49	313.63	90	10763.1	39.80	13	24983.7
*R*101/10/7/100/200/0.07	34	15.24	32	111.83	22.66	48	1750.66	2.13	5	2701.96
*R*201/10/7/100/200/0.07	34	6.56	16	30.68	24.90	53	1957.42	3.80	9	2787.6
*R*201/25/18/100/200/0.07	65	62.26	14	713.03	371.97	73	30943.7	37.81	9	68007
*RC*101/10/7/100/200/0.07	33	10.85	21	100.92	26.97	47	1689.98	3.61	7	2426.32
*RC*101/25/18/100/200/0.07	62	120. 88	25	343.23	330.48	61	22995.7	71.34	14	52676.8
*RC*201/10/7/100/200/0.07	33	7.96	18	34.8	24.67	48	1762.61	1.76	5	2413.12
*RC*201/25/18/100/200/0.07	60	165.63	35	830.26	306.79	63	26586.6	49.83	11	48751.7
*C*101/10/8/100/200/0.30/2	23	1.87	23	24.71	3.30	38	205.42	0.32	4	541.22
*C*101/10/5/100/200/0.30/2	23	1.36	22	23.15	1.72	26	245.7	0.48	8	486.64
*C*101/10/7/100/200/0.30	26	9.97	32	32.53	10.04	32	405.57	2.75	9	905.24
*C*101/25/18/100/200/0.30	54	108.87	29	682.73	207.31	54	10826	26.64	7	29357.4
*C*201/10/7/100/200/0.30	24	4.53	22	28.9	9.29	44	327.98	1.19	6	592.06
*C*201/25/18/100/200/0.30	52	74.87	22	259.7	149.85	42	9591	38.31	11	23200.1
*R*101/10/7/100/200/0.30	34	14.68	32	94.82	16.48	35	1727.18	3.82	9	2674.57
*R*201/10/7/100/200/0.30	34	13.44	29	162.11	17.66	36	1969.89	3.93	9	2821.66
*R*201/25/18/100/200/0.30	65	186.21	36	2046.73	362.67	66	28000.2	48.22	9	71686.80
*RC*101/10/7/100/200/0.30	33	15.26	29	53.22	28.44	52	1453.47	4.49	9	2545.55
*RC*101/25/18/100/200/0.30	62	72.75	14	812.32	318.96	56	25771.4	74.83	14	56895.70
*RC*201/10/7/100/200/0.30	33	18.40	39	118.45	21.16	44	1588.43	3.50	8	2450.95
*RC*201/25/18/100/200/0.30	60	210.91	45	809.49	340.94	71	14313	36.41	8	48949.1
*C*101/10/18/200/400/0.07/2	52	55.25	45	1197.89	66	51	25780.1	17.07	14	49151.2
*C*101/10/18/200/400/0.07	58	171.48	50	1667.12	280.39	81	34024.9	30.18	9	82066.5
*C*201/10/18/200/400/0.07	54	85.60	29	827.01	207.49	68	23015.2	14.80	5	59135
*C*101/10/18/200/400/0.30/2	52	48.67	40	815.41	31.94	24	20285.1	18.36	14	47728.1
*C*101/10/18/200/400/0.30	58	184.45	48	834	259.41	65	36565.1	49.39	13	84385
*C*201/10/18/200/400/0.30	54	77.15	29	560.63	181.53	62	22369.5	11.15	4	58094

Table 6.2: Results of the proposed algorithms MOVND/P, MOSALNS/P and the comparative algorithm MO-ALNS of Cota et al. [90]

Instance	*n*	Improvement of PLS-MOVND over MO-ALNS literature (%)			Improvement of MOVND/PI over MO-ALNS literature (%)			Improvement of MOSALNS/P over MO-ALNS literature (%)		
		IHV	INDP	ICPU	IHV	INDP	ICPU	IHV	INDP	ICPU
RE/6/2/80/80/0.30/2	6	165.21	166.67	72.73	157.76	166.67	72.73	-30.61	-33.33	89.39
RE/6/4/101/101/0.30/2	12	404.55	425	63.51	448.32	450	68.56	221.43	200	34.48
RE/6/2/80/80/0.07/2	6	165.21	166.67	73.13	157.76	166.67	73.13	-30.61	-33.33	89.55
RE/6/4/101/101/0.07/2	12	414.19	400.00	64.44	475.20	466.67	69.07	233.98	233.33	60.13
*C*101/10/8/100/200/0.07/2	23	701.16	616.67	74.20	1172.15	966.67	80.49	651.42	566.67	25.16
*C*101/10/5/100/200/0.07/2	23	526.29	455.56	69.87	426.67	366.67	77.24	458.27	377.78	9.26
*C*101/10/7/100/200/0.07	26	570.88	544.44	74.72	505.19	477.78	80.19	550.02	511.11	38.94
*C*101/25/18/100/200/0.07	54	481.73	450.00	86.35	452.72	422.73	81.27	252.68	231.82	52.32
*C*201/10/7/100/200/0.07	24	1325.50	1200.00	74.44	1557.33	1400.00	80.28	1144.45	1066.67	51.70
*C*201/25/18/100/200/0.07	52	462.40	430.77	85.57	745.35	669.23	79.71	687.89	592.31	56.92
*R*101/10/7/100/200/0.07	34	1632.33	1420.00	77.09	1000.86	840.00	79.41	964.40	860.00	35.21
*R*201/10/7/100/200/0.07	34	700.18	633.33	77.76	587.87	522.22	80.38	554.00	488.89	29.78
*R*201/25/18/100/200/0.07	65	1291.07	1033.33	88.29	2204.63	1788.89	80.09	883.59	711.11	54.50
*RC*101/10/7/100/200/0.07	33	731.05	685.71	77.40	981.68	885.71	80.55	645.93	571.43	30.35
*RC*101/25/18/100/200/0.07	62	693.88	650.00	87.26	801.59	750.00	82.14	363.23	335.71	56.35
*RC*201/10/7/100/200/0.07	33	1753.19	1140.00	77.24	1695.45	1140.00	78.81	1297.49	860.00	26.96
*RC*201/25/18/100/200/0.07	60	1043.20	981.82	87.76	1038.95	927.27	82.68	515.66	472.73	45.47
*C*101/10/8/100/200/0.30/2	23	1226.20	1175.00	74.89	844.89	800.00	80.91	934.23	850.00	62.05
*C*101/10/5/100/200/0.30/2	23	471.57	437.50	73.59	417.55	375.00	79.87	256.15	225.00	49.51
*C*101/10/7/100/200/0.30	26	546.40	533.33	76.08	553.03	533.33	81.37	264.91	255.56	55.20
*C*101/25/18/100/200/0.30	54	687.93	671.43	86.00	1211.92	1185.71	81.29	678.12	671.43	63.12
*C*201/10/7/100/200/0.30	24	874.30	800.00	73.35	779.81	733.33	79.50	681.11	633.33	44.60
*C*201/25/18/100/200/0.30	52	610.77	600.00	84.55	818.96	781.82	82.16	291.10	281.82	58.66
*R*101/10/7/100/200/0.30	34	691.98	588.89	76.82	694.11	600.00	78.31	330.96	288.89	35.42
*R*201/10/7/100/200/0.30	34	1003.96	888.89	78.08	800.78	700.00	80.07	349.09	300.00	30.19
*R*201/25/18/100/200/0.30	65	950.56	933.33	88.95	715.44	711.11	81.26	652.06	633.33	60.94
*RC*101/10/7/100/200/0.30	33	532.42	477.78	78.48	417.37	366.67	81.64	532.43	477.78	42.90
*RC*101/25/18/100/200/0.30	62	595.72	578.57	88.24	729.60	707.14	83.76	326.25	300.00	54.70
*RC*201/10/7/100/200/0.30	33	1004.04	887.50	77.56	662.84	587.50	79.13	503.50	450.00	35.19
*RC*201/25/18/100/200/0.30	60	1527.42	1462.50	87.82	1317.06	1250.00	83.84	836.35	787.50	70.76
*C*101/10/18/200/400/0.07/2	52	523.66	485.71	86.46	734.96	685.71	79.27	286.68	264.29	47.55
*C*101/10/18/200/400/0.07	58	987.93	966.67	88.09	372.22	1200.00	84.32	829.00	800.00	58.54
*C*201/10/18/200/400/0.07	54	1645.01	1580.00	87.19	2036.73	1940.00	81.60	1301.27	1260.00	61.08
*C*101/10/18/200/400/0.30/2	52	395.86	414.29	86.16	388.66	392.86	79.67	73.89	71.43	57.50
*C*101/10/18/200/400/0.30	58	784.27	769.23	88.26	804.56	784.62	85.34	425.20	400.00	56.67
*C*201/10/18/200/400/0.30	54	2301.51	2250.00	86.95	2835.92	2675.00	84.43	1526.85	1450.00	61.49

Table 6.3: Improvement of the proposed algorithms, PLS-MOVND, MOVND/PI, MOSALNS/P over the comparative algorithm MO-ALNS of Cota et al. [90]

Instance	*n*	Improvement of MOVND/PI over PLS-MOVND (%)			Improvement of PLS-MOVND over MOSALNS/P (%)			Improvement of MOVND/PI over MOSALNS/P (%)		
		IHV	INDP	ICPU	IHV	INDP	ICPU	IHV	INDP	ICPU
RE/6/2/80/80/0.30/2	6	-2.81	0.00	0.00	282.19	300.00	-157.14	271.44	300.00	-157.14
RE/6/4/101/101/0.30/2	12	8.67	4.76	13.85	56.97	75	44.31	70.59	83.33	52.02
RE/6/2/80/80/0.07/2	6	-2.81	0.00	0.00	282.19	300.00	-157.14	271.44	300.00	-157.14
RE/6/4/101/101/0.07/2	12	11.87	13.33	13.02	53.96	50.00	10.83	72.22	70.00	22.44
*C*101/10/8/100/200/0.07/2	23	58.79	48.84	24.39	6.62	7.50	65.52	69.30	60.00	73.93
*C*101/10/5/100/200/0.07/2	23	-15.91	-16.00	24.47	12.18	16.28	66.79	-5.66	-2.33	74.92
*C*101/10/7/100/200/0.07	26	-9.79	-10.34	21.67	3.21	5.45	58.59	-6.90	-5.45	67.57
*C*101/25/18/100/200/0.07	54	-4.99	-4.96	-37.20	64.95	65.75	71.37	56.72	57.53	60.72
*C*201/10/7/100/200/0.07	24	16.26	15.38	22.83	14.55	11.43	47.09	33.18	28.57	59.17
*C*201/25/18/100/200/0.07	52	50.31	44.93	-40.68	-28.62	-23.33	66.51	7.29	11.11	52.89
*R*101/10/7/100/200/0.07	34	-36.45	-38.16	10.17	62.75	58.33	64.63	3.43	-2.08	68.23
*R*201/10/7/100/200/0.07	34	-14.04	-15.15	11.75	22.35	24.53	68.33	5.18	5.66	72.05
*R*201/25/18/100/200/0.07	65	65.67	66.67	-69.99	41.43	39.73	74.26	134.31	132.88	56.25
*RC*101/10/7/100/200/0.07	33	30.16	25.45	13.93	11.41	17.02	67.55	45.01	46.81	72.07
*RC*101/25/18/100/200/0.07	62	13.57	13.33	-40.21	71.38	72.13	70.83	94.63	95.08	59.10
*RC*201/10/7/100/200/0.07	33	-3.12	0.00	6.89	32.61	29.17	68.84	28.48	29.17	70.99
*RC*201/25/18/100/200/0.07	60	-0.37	-5.04	-41.53	85.69	88.89	77.56	85.00	79.37	68.25
*C*101/10/8/100/200/0.30/2	23	-28.75	-29.41	23.98	28.23	34.21	33.85	-8.64	-5.26	49.71
*C*101/10/5/100/200/0.30/2	23	-9.45	-11.63	23.76	60.48	65.38	47.70	45.32	46.15	60.13
*C*101/10/7/100/200/0.30	26	1.03	0.00	22.11	77.14	78.13	46.61	78.96	78.13	58.41
*C*101/25/18/100/200/0.30	54	66.50	66.67	-33.59	1.26	0.00	62.02	68.60	66.67	49.27
*C*201/10/7/100/200/0.30	24	-9.70	-7.41	23.07	24.73	22.73	51.89	12.64	13.64	62.99
*C*201/25/18/100/200/0.30	52	29.29	25.97	-15.44	81.74	83.33	62.63	134.97	130.95	56.86
*R*101/10/7/100/200/0.30	34	0.27	1.61	6.43	83.77	77.14	64.10	84.27	80.00	66.41
*R*201/10/7/100/200/0.30	34	-18.40	-19.10	9.07	145.82	147.22	68.61	100.58	100.00	71.46
*R*201/25/18/100/200/0.30	65	-22.38	-21.51	-69.61	39.69	40.91	71.71	8.43	10.61	52.01
*RC*101/10/7/100/200/0.30	33	-18.19	-19.23	14.68	0.00	0.00	62.32	-18.19	-19.23	67.85
*RC*101/25/18/100/200/0.30	62	19.24	18.95	-38.07	63.22	69.64	74.04	94.63	101.79	64.15
*RC*201/10/7/100/200/0.30	33	-30.90	-30.38	6.96	82.94	79.55	65.38	26.40	25.00	67.79
*RC*201/25/18/100/200/0.30	60	-12.93	-13.60	-32.68	73.80	76.06	58.36	51.34	52.11	44.75
*C*101/10/18/200/400/0.07/2	52	33.88	34.15	-53.06	61.29	60.78	74.18	115.93	115.69	60.48
*C*101/10/18/200/400/0.07	58	22.22	21.88	-31.67	17.11	18.52	71.27	43.13	44.44	62.18
*C*201/10/18/200/400/0.07	54	22.45	21.43	-43.66	24.53	23.53	67.09	52.48	50.00	52.73
*C*101/10/18/200/400/0.30/2	52	-1.45	-4.17	-46.89	185.15	200.00	67.43	181.02	187.50	52.16
*C*101/10/18/200/400/0.30	58	2.30	1.77	-24.93	68.37	73.85	72.92	72.23	76.92	66.16
*C*201/10/18/200/400/0.30	54	22.25	18.09	-19.30	47.62	51.61	66.12	80.47	79.03	59.58

Table 6.4: Comparison between the performance of the proposed algorithms PLS-MOVND. MOVND/PI and MOSALNS/P

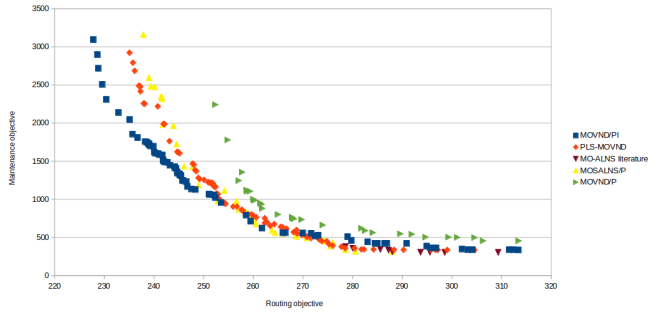

Figure 6.3: Pareto fronts of the different algorithms for the instance with 34 operations, R201/10/7/100/200/0,30.

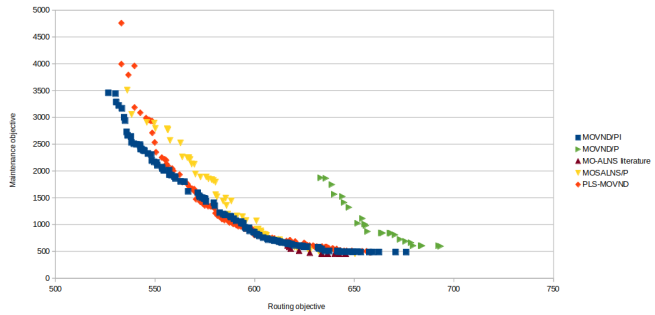

Figure 6.4: Pareto fronts of the different algorithms for the instance with 62 operations, RC101/25/18/100/200/0,07.

aspects, routing, maintenance, and production, of the presented problem and multi-objective algorithms to solve it. A bi-objective mathematical model is proposed. It minimizes simultaneouslythe total travel cost and a variable amount of production losses if the opportunities of available production stoppage intervals can not be used totally or partially for maintenance actions. It also minimizes the maintenance cost with the penalty cost of not respecting the maintenance intervals. In addition to the problem defined in Chapter 5, we suppose that the technicians start the maintenance operations as soon as they arrive. Multi-objective extensions of PLS, VND, and ALNS algorithms are proposed to deal with combinatorial multi-objective optimization problems in general and this problem especially. First, Pareto Local Search Multi-objective Variable Neighborhood Descent (PLS-MOVND), a hybridizing between the Pareto Local Search

algorithm (PLS) and an adaptation of the Variable Neighborhood Descent to the multi-objective case is proposed to solve this bi-objective variant of the problem. It is followed by a multi-objective Semi Adaptive Large Neighborhood Search (MOSALNS/P) designed for the same purpose. They are both based on Pareto dominance and start with a unique initial solution. The design of every part of the algorithms is described. Moreover, the MOSALNS/P uses MOVND/P as a multi-objective local search algorithm which, in turn, uses a new defined multi-objective best improvement strategy (MOBI/P). MOASLNS/P integrates a new removal operator based on risk and a semi-adaptive mechanism to cope with the peculiarities of the problem. The metaheuristic selects the most suitable operators while alternating between learning and diversification to obtain high-quality solutions. Several computational experiments have shown the significance of the proposed methods. The proposed methods considerably outperform in all time and quality indicators a multi-objective Adaptive Large Neighborhood Search of the literature. Moreover, the two proposed methods, PLS-MOVND and MOVND/PI outperform MOSALNS/P. They are also efficiently designed and parameters-free, which makes them easy to test on new problems.

Chapter 7

Conclusion and perspectives

Outline of the current chapter

This chapter summarizes this thesis, recalls its main contributions, and sheds light on some room for future directions.

7.1 Summary

This thesis addresses the maintenance scheduling and workforce routing problem. The problem arises for maintenance service providers. This service company must ensure a good quality of service at the lowest overall cost to geographically distributed customers. The quality of that service is mainly linked to the right time to execute maintenance operations. This right time is first related to reliability models that serve to determine it. Then, teams of qualified technicians must arrive at the right time following a designed routing plan. In this thesis, we took a closer look at how vehicle routing problems and reliability engineering can contribute to service providers' maintenance scheduling. We

investigate the problem from a technical and management perspective to understand how to integrate technical aspects of reliability with organizational aspects of transport management. Further analysis leads the study to focus on vehicle routing problems, particularly Vehicle Routing Problem with Time Windows (VRPTW). This variant is used to plan transportation within a network. More sophisticated constraints have been added to deal with the complexity of the real-world requirements. Maintenance technical considerations are integrated as additional constraints and objectives that must be considered to tackle the problem. We propose new models based on reliability and VRP modeling paradigms. In the second part of the thesis, opportunistic maintenance is considered along with preventive and corrective maintenance. The problem must satisfy both its operational constraints and production constraints. Opportunistic maintenance is executed during the planned shutdown of systems. The problems defined are too hard for contemporary commercial solvers for interesting size instances. Moreover, finding the optimum solutions for these problems is crucial and yet challenging. Therefore, we propose innovative heuristics and single and multi-objective algorithms to solve the problems and deal with the operational constraints. Each piece of equipment degrades over time. The maintenance decisions determine which time to execute maintenance operations. The routing decisions need to ensure that the delivery of the maintenance services is on time.

Several topics in the relevant literature are reviewed. The problem is analyzed from both technical engineering and management point of view in Chapter 1. The first part of the chapter presents the maintenance methods from their foundation until now. A great emphasis is devoted to the distinction between the different concepts and their interaction. Maintenance orientations are presented to understand the essence of the notions involved when defining maintenance concepts in the literature. They have led, in the literature to the definition of essential concepts in maintenance, and some of these concepts even evolved to become maintenance strategies. The latter are reviewed with an emphasis on the Reliability Centered Maintenance used in this thesis. Basic formalized models for the optimization of maintenance intervals used in Reliability engineering are then discussed. Finally, the chapter's second part presents the main variants of the Vehicle Routing Problem (VRP). The original Traveling Salesman Problem

(TSP) has extended to several VRP variants to accommodate a wide range of industrial applications. However, the further analysis leads the study to focus on Vehicle Routing Problem with Time Windows (VRPTW). VRPTW enforces the visits within given time windows.

Rapid and robust solution techniques need to be developed with the growth of computational platforms and problems faced in the industry. Chapter 2 is dedicated to the literature related to resolution methods. The focus is on approximate approaches. Indeed, exact approaches take a very long time, depending on the size of the problem at hand, to produce a solution. Chapter 2 starts with an overview of the different classes of optimization problems to identify the problems studied in this thesis. Then, existing heuristic and metaheuristics and their categories are presented and discussed. The chapter finally surveys multi-objective optimization classes and techniques.

We developed models and algorithms for several variants of the studied problem based on the aforementioned literature survey. Indeed,this manuscript gathers our investigations on the problem at hand and on several approximated methods. The contributions of this thesis are summarized below.

In chapter 3, we address a joint maintenance scheduling and workforce routing problem. The set of machines is supposed to be geographically distributed and subject to random failures. We seek to determine the optimal times to perform preventive maintenance operations on each machine and find the optimal sequence of maintenance operations to perform that minimizes each objective considered. We proposed a novel mathematical model that aims to minimize either failure, maintenance, or transport costs in the case of time-based preventive maintenance. We proposed a nonlinear stochastic failure cost that uses information from equipment degradation. It includes direct costs and indirect costs. This objective is to be considered for high-risk industries where safety is of primal concern. We also investigate the maintenance cost previously proposed in the literature. It balances both preventive and corrective expenses related to maintenance operations. The time of the last restoration is included in both costs to verify the renewal theory hypothesis. These objectives are studied along with the routing cost. Finally, the model considers several operational constraints such as maintenance operations time windows, penalties related to the non-

respect of maintenance time windows, and the uncertainty of breakdowns to accommodate real industrial problems. Novel constructive heuristics based on the failure and maintenance behavior are proposed to generate initial solutions followed by a General Variable Neighborhood Search (GVNS) metaheuristic to solve the problem. The performance of the GVNS algorithm is examined on generated instances. The computational results indicate that the dedicated initial solutions improve the quality of the final solutions and considerably reduce the CPU time of the GVNS metaheuristic. The experimental results also reveal the effectiveness of the GVNS algorithm. Indeed, the algorithm's results have far exceeded those of the commercial solver.

Chapter 4 considers the bi-objective variant of the previous problem. The bi-objective model deals with the same constraints. Besides, we associate the failure cost or maintenance cost with the routing cost in the objectives. This approach aims to determine the optimal times to perform preventive maintenance operations on each machine and find the optimal sequence of these operations that simultaneously minimizes the routing cost and the maintenance or the failure cost. Adaptations of Variable Neighborhood Descent and General Variable Neighborhood Search, which are single-objective metaheuristics, are proposed to contain multiple objective scenarios. They are called MOVND/P, MOVND/PI, and MOGVNS/P. Every component's design is detailed, including the improvement method, the multi-objective local search, the acceptance criterion, the stopping criterion, and the neighborhood change procedure. These algorithms are based on the Pareto dominance concept. The proposed multi-objective local search incorporated, MOBI/P, turned out to be a new state-of-the-art multi-objective best improvement strategy. While exploring the neighborhood, the MOBI/P permits a rapid convergence if the new solution dominates the current solution and ensures diversification in the case of incomparable solutions. The methods are simple, with few parameters to tune. This feature allowed the integration of the local search algorithm MOVND/P later in the thesis into advanced multi-phase frameworks. Several comparisons to existing multi-objective VND and GVNS metaheuristics are conducted. The experimental results reveal the effectiveness of the proposed algorithms on all quality and time indicators measured. Several GVNS-based frameworks are proposed. Their performances

are evaluated to measure the influence of state-of-art techniques such as the decomposition approach or using an initial population of solutions.

Chapter 5 extends the previous problem to deal with Opportunistic Maintenance (OM). The OM policy is a novel concept strongly linked to the optimal maintenance system. It consists of planning maintenance at the right time on equipment while penalizing the less the production process. Determining this right time is difficult since it includes not only technical considerations of the machine's lifetime and management considerations of workforce routing. Additionally, scheduled production stoppages need to be favored to reduce the negative impact on production. In this regard, the problem is characterized by three interrelated aspects: routing, maintenance, and production influencing the final decision. To our knowledge, this thesis presents the first study that includes formalizing the opportunistic maintenance concept in a workforce scheduling context. We have formulated a new model that complies with the previous formalization. Additional constraints and objective costs are considered. Production losses are incurred depending on the use of production stoppages intervals for maintenance operations. A new heuristic, CSTMPSTW, is proposed to determine the best start time in the maintenance intervals and to select a production stoppage. It is integrated with a Semi-Adaptive Large Neighborhood Search (SALNS) to solve the problem. SALNS is an adaptation of the ALNS metaheuristic that integrates a new removal operator based on the risk of failure. It destroys maintenance operations related to components with a potentially high probability of failing. It also incorporates a semi-adaptive mechanism to significantly enhance the solution approach's performance. The latter aims to balance between the diversification procedure and learning from past information while selecting operators. The computational results demonstrate the feasibility of the model. Indeed, it satisfies a significant number of additional constraints. Computational experiments confirm the effectiveness of the solution approach to solve the problem compared to the commercial solver. The SALNS has been tested on the joint maintenance scheduling and workforce routing problem of Chapter 3 and provided good results.

The purpose of Chapter 6 is to address a bi-objective variant of the previously defined problem. Several problem characteristics are identified, including

problem size and constraint levels. In addition to the previous problem, the technicians start the maintenance operations as soon as they arrive with respect to the maintenance intervals. We formulated a new bi-objective model that integrates the additional hypothesis. The first objective minimizes the routing cost and the possible production losses if the available production stoppage intervals can not be used partially or entirely for maintenance operations. The second objective minimizes the total maintenance cost with the penalty cost of not respecting the maintenance intervals. In the second part of this chapter, extensions of PLS and ALNS algorithms are proposed to deal with combinatorial multi-objective optimization problems in general and this problem particularly. First, we proposed PLS-MOVND, which can significantly enhance the performance of PLS by integrating multiple neighborhood structures and several design features. Next, a Multi-Objective Adaptive Large Neighborhood Search named (MOSALNS/P) is proposed. MOSALNS/P appropriately captures the novel components of SALNS and extends it to deal with multi-objective optimization. Furthermore, it incorporates MOVND/P as a local search procedure to help the method approximate the whole Pareto front. Finally, the proposed algorithms, MOSALNS/P, PLS-MOVND, MOVND/PI, and the state-of-the-art multi-objective ALNS, have been compared. The experimental results reveal the effectiveness of the methods in all time and quality indicators measured. Indeed, all the proposed methods considerably outperform a multi-objective Adaptive Large Neighborhood Search of the literature. Moreover, the two methods, PLS-MOVND and MOVND/PI, with almost similar performance, topped MOSALNS/P. The outstanding performance of PLS-MOVND and MOVND/PI and their frameworks encourage their application to new complex problems. They are also efficiently designed and parameters free and can be easily integrated into other multi-phase frameworks.

7.2 Future research

This research provides efficient models and algorithms that can be integrated with a decision support system to deal with tactical management problems for a maintenance service provider. The questions and the open research tracks are, in fact, numerous. Future research in this field can tackle the same problem

from other points of view or extend the contributions to meet more the industrial world requirements. The first avenue regarding future research concerns exploring the integration of different maintenance strategies with the routing problem to reduce the maintenance costs and improve the quality of maintenance services. In chapters 5 and 6, we considered opportunistic maintenance and defined a new problem. Since the stream of research regarding opportunistic maintenance and the described problem are in the early stage of development and are proposed for the first time, there is still vast room for improvements. Other production components could be integrated into the design to improve the model's efficiency. An example of such new elements is a production planner that helps decision-makers in the planning phase to construct efficient work plans. We have considered in our work production stoppages to perform maintenance operations as opportunity intervals. We could have rather included the production plan as a constraint that we must respect. Moreover, the production can be planned simultaneously with the maintenance and the routing to design a complete three-decision model.

Condition-based maintenance could also constitute an alternative to time-based maintenance as this will be representative of interesting real-world scenarios. Condition-based or predictive maintenance uses data to deliver decision models based on past performances. The maintenance can be delayed until some thresholds of measured parameters are reached. The data exploited could be related to history of maintenance failures, production demand, or past work plans. A large amount of data can also be recorded through vehicle tracking. This data offers essential insights on complementary information such as travel time, service time, disruptions, delays, etc.

Studying the dynamic version of the studied problems is also of primal interest. The planner can be conceived dynamically to interact in real-time with technicians by sharing related information and rescheduling alternative plans based on the new situation. Furthermore, incorporating uncertain information, travel time, service time, demand, etc., can improve the model realism.

Managerial insights on the practicability of the proposed approaches to other problems need to be drawn. Some future directions in the general field include extending the models to new systems currently emerging in workforce

scheduling for service providers. Some examples include home health care problems, nurse scheduling, nurse rostering, crew scheduling, and a general comprehensive list of transport problems in a supply chain. The development of these models needs to be naturally tailored to the problems to scale well in practice and to include their different scenarios.

On the other hand, this thesis proposes several local search-based multi-objective algorithms. They have been designed to ensure sufficient diversity, usually obtained with a population of solutions and genetic operators. Since most literature works focuses on population-based methods, further research is required to explore other local search-based algorithms and their interactions.

Although the present study demonstrates the effectiveness of the proposed methods over several state-of-the-art multi-objective local search methods, further studies are required to improve them regarding some mechanisms. The MOBI/P procedure can offer more control over the diversification procedure, limiting the time spent in this latter and speeding up the search. The MOVND/PI algorithm can include a more sophisticated diversification procedure to outperform MOGVNS/P in terms of solution quality. Since then, the hybridization of local search and population-based methods has demonstrated outstanding results for VRP in the literature; it would be promising to hybrid the techniques proposed with population-based methods to solve the problem. One potentially interesting research direction is integrating the proposed multi-objective algorithms, especially MOVND/P, MOVND/PI, and PLS-MOVND, into more sophisticated frameworks of multi-objective metaheuristics in the literature. Indeed, population-based methods particularly can be coupled with MOVND/P to enhance their performance further. The search can also be guided towards more promising regions of the search space by including learning mechanisms in the local search procedure. Machine learning techniques can be used to drive metaheuristics. We genuinely believe that guiding the search by learning from the past performances will speed up this latter. This procedure will escape unnecessary moves and reduce the computational time considerably, and therefore offer the possibility to solve larger instances. Supervised learning techniques such as Neural Networks or Support Vector Machines can be trained with optimal schedules. We think reinforcement learning is more suitable for the methods

since it depends on an agent interacting with the random environment to learn from past experiences using a reward and penalty system.

An essential part of the thesis was devoted to the proposition of heuristic dedicated to coping with the peculiarities of the defined problems. These heuristics can be improved and coupled to achieve better performance. The Greedy Constructive Heuristic Maintenance (GCHM) needs to be combined with a simplified version of the CSTMPSTW heuristic that only determines the optimal start times of maintenance operations to provide an efficient initial solution. This solution can outperform the initial solution provided with the well-known best insertion heuristic in terms of computational time and will be particularly powerful for large-size instances. The heuristics GCHM and GCHF integrated with CSTMPSTW can be used afterward as new insertion operators for SALNS and MOSALNS/P.

7.3 Concluding remarks

This thesis focused on developing models and algorithms for maintenance scheduling and workforce routing problems. The model proposed has incorporated several real-world features such as uncertainty and reliability considerations. In addition, the model was extended to account for critical operational constraints related to production. Such an approach aims to integrate all the key elements that influence the cost structure the most. The systemic point of view has been the main driving force behind the models' development.

The thesis was also devoted to designing dedicated heuristics and metaheuristics that scale to the sizes of real-life systems. Their goal was to find high-quality solutions in short running times while considering several aspects that stem from real-life settings. The focus has been placed on the solutions' quality and the time spent obtaining them.

The approaches proposed in this thesis apply to large and interconnected real-life networks. Service providers can offer quality services while reducing their running costs. Furthermore, the solution methods proposed throughout this work can be applied to any single or multi-objective combinatorial optimization problem with only a suitable adaptation of the operators.

Future directions were proposed in this chapter. They can focus on exploring further aspects faced by service providers in practice. More components and operational constraints that directly impact the total cost can be included to allow for the smooth functioning of the system. Alternative efficient maintenance strategies could be considered. One research direction is to benefit from real-time data to adjust plans consequently.

Besides, the proposed solution approaches are currently in an earlier phase, and there is a great margin for optimization, especially by integrating them with more sophisticated multi-phase frameworks. The proposed metaheuristics can also be driven by machine learning techniques to enhance their performance.

Bibliography

[1] Barlow, R. E., Hunter, L. C., Proschan, F. (1963). Optimum checking procedures. Journal of the society for industrial and applied mathematics, 11(4), 1078-1095.

[2] Barlow, R., and Hunter, L. (1960). Optimum preventive maintenance policies. Operations research, 8(1), 90-100.

[3] Kobbacy K.A.H., Murthy D.N.P. (2008) An Overview. In: Complex System Maintenance Handbook. Springer Series in Reliability Engineering. Springer, London.

[4] Thietart, R.A. (2014). Methodes de recherche en management (4ème éd.). Paris, France: Dunod. 656 pages.

[5] Cowling, P., Kendall, G., Soubeiga, E. (2000, August). A hyperheuristic approach to scheduling a sales summit. In International conference on the practice and theory of automated timetabling (pp. 176-190). Springer, Berlin, Heidelberg.

[6] https://nte.mines-albi.fr/Optimisation/fr/co/0Intro_Classification.html

[7] https://en.wikipedia.org/wiki/Combinatorial_optimization

[8] Algorithme de recherche avec tabou, Course INF6953, UFR, Bourgogne University. France.

[9] Tarantilis, C. D., Kiranoudis, C. T., Vassiliadis, V. S. (2004). A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research, 152(1), 148-158.

[10] Asefi, H., Shahparvari, S., Chhetri, P., Lim, S. (2019). Variable fleet size and mix VRP with fleet heterogeneity in Integrated Solid Waste Management. Journal of Cleaner Production, 230, 1376-1395.

[11] Crevier, B., Cordeau, J. F., Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. European journal of operational research, 176(2), 756-773.

[12] Kirschstein, T., Bierwirth, C. (2018). The selective Traveling Salesman Problem with emission allocation rules. Or Spectrum, 40(1), 97-124.

[13] Laporte, G., Martello, S. (1990). The selective travelling salesman problem. Discrete applied mathematics, 26(2-3), 193-207.

[14] Butt, S. E., Cavalier, T. M. (1994). A heuristic for the multiple tour maximum collection problem. Computers and Operations Research, 21(1), 101-111.

[15] Koskosidis, Y. A., Powell, W. B., Solomon, M. M. (1992). An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints. Transportation science, 26(2), 69-85.

[16] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers and operations research, 13(5), 533-549.

[17] Glover, F. (1989). Tabu search—part I. ORSA Journal on computing, 1(3), 190-206.

[18] Lourenço, H.R., Martin, O.C., Stützle, T. (2003). Iterated Local Search. In: Glover, F., Kochenberger, G.A. (eds) Handbook of Metaheuristics. International Series in Operations Research and Management Science, vol 57. Springer, Boston, MA.

[19] Holland, J. H. (1962). Outline for a logical theory of adaptive systems. Journal of the ACM (JACM), 9(3), 297-314.

[20] M. Dorigo, Optimization, Learning and Natural Algorithms. PhD thesis. Politecnico di Milano, Italy, 1992

[21] Or, I., Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Ph.D thesis. Department of Industrial Engineering and Management Science, Northwestern University, Illinois,1967

[22] M. Dorigo and G. Di Caro (1999) The Ant Colony Optimization meta-heuristic. In: D. Corne, M. Dorigo and F. Glover (eds.), New Ideas in Optimization. McGraw Hill, London, UK, pp. 11–32.

[23] Mirjalili S., Multi-objective Optimization Problems and Algorithms, Udemy Online Course.

[24] Potvin, J. Y., Rousseau, J. M. (1995). An exchange heuristic for routeing problems with time windows. Journal of the Operational Research Society, 46(12), 1433-1446.

[25] Geiger, M. J. (2008). Randomised variable neighbourhood search for multi objective optimisation. arXiv preprint arXiv:0809.0271.

[26] Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. European journal of operational research, 126(1), 106-130.

[27] Lin, S., Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. Operations research, 21(2), 498-516.

[28] Lin, S. (1965). Computer solutions of the traveling salesman problem. Bell System Technical Journal, 44(10), 2245-2269.

[29] Clarke, G., Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations research, 12(4), 568-581.

[30] Coelho, V. N., Coelho, I. M., Mladenović, N., Ramalhinho, H., Ochi, L. S., Guimarães, F. G., Souza, M. J. (2018, October). Less Is More: The Neighborhood Guided Evolution Strategies Convergence on Some Classic Neighborhood Operators. In International Conference on Variable Neighborhood Search (pp. 77-88). Springer, Cham.

[31] Chen, Y., Cowling, P. I., Polack, F. A., Mourdjis, P. J. (2016). A multi-arm bandit neighbourhood search for routing and scheduling problems.

[32] Trick, M. A. (1992). A linear relaxation heuristic for the generalized assignment problem. Naval Research Logistics (NRL), 39(2), 137-151.

[33] Potvin, J. Y., Rousseau, J. M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research, 66(3), 331-340.

[34] Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. European journal of operational research, 59(3), 345-358.

[35] https://en.wikipedia.org/wiki/Hyper-heuristic

[36] Hosny, M. (2011). Heuristic techniques for solving the vehicle routing problem with time windows. In 2011 international conference on future information technology.

[37] Frangopoulos, C. A. Static and Dynamic Optimization of Synthesis, Design and Operation of Total Energy Systems of Ships. MOSES Workshop, EPFL Sion, Switzerland

[38] Kallehauge B., Larsen J., Madsen O.B., Solomon M.M. (2005) Vehicle Routing Problem with Time Windows. In: Desaulniers G., Desrosiers J., Solomon M.M. (eds) Column Generation. Springer, Boston, MA.

[39] Ioachim, I., Gelinas, S., Soumis, F., Desrosiers, J. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. Networks: An International Journal, 31(3), 193-204.

[40] Balinski, M. L. and Quandt, R. E. (1964). On an integer program for a delivery problem.Operations Research 12(2), 300-304.

[41] Letchford, A. N., Nasiri, S. D., Theis, D. O. (2013). Compact formulations of the Steiner traveling salesman problem and related problems. European Journal of Operational Research, 228(1), 83-92.

[42] Miettinen K. (1998) A Priori Methods. In: Nonlinear Multiobjective Optimization. International Series in Operations Research and Management Science, vol 12. Springer, Boston, MA.

[43] Jaillet, P., Wagner, M. R. (2008). Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses. Operations research, 56(3), 745-757.

[44] Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. Computers and Industrial Engineering, 99, 300-313.

[45] Pillac, V., Gendreau, M., Guéret, C., Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. European Journal of Operational Research, 225(1), 1-11.

[46] Gendreau, M. (2018). Stochastic Vehicle Routing: an Overview and an Exact Solution Approach for the Vehicle Routing Problem with Stochastic Demands under an Optimal Restocking Recourse Policy.

[47] Jabali, O. (2018). Stochastic Vehicle Routing Problems.

[48] Laporte, G., Nobert, Y. and Desrochers, M. (1985). Optimal Routing under Capacity and Distance Restrictions. Operations Research 33(5), 1050-1073.

[49] Garvin, W. W., Crandall, H. W., John, J. B. and Spellman, R. A. (1957). Applications of Linear Programming in the Oil Industry. Management Science 3(4), 407-430.

[50] Miller, R., Traveling Salesman Problem: the MTZ Formulation

[51] Dantzig, G., Fulkerson, R., Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. Journal of the operations research society of America, 2(4), 393-410.

[52] Miller, C. E., Tucker, A. W., Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. Journal of the ACM (JACM), 7(4), 326-329.

[53] Mladenović, N., Pei, J., Pardalos, P. M., Urošević, D. (2022). Less is more approach in optimization: a road to artificial intelligence. Optimization Letters, 16(1), 409-420.

[54] Claus, A. (1984). A new formulation for the travelling salesman problem. SIAM Journal on Algebraic Discrete Methods, 5(1), 21-25.

[55] Alegre, J., Laguna, M., Pacheco, J. (2007). Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. European Journal of Operational Research, 179(3), 736-746.

[56] Tang, H., Miller-Hooks, E., Tomastik, R. (2007). Scheduling technicians for planned maintenance of geographically distributed equipment. Transportation Research Part E: Logistics and Transportation Review, 43(5), 591-609.

[57] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E. (1953). Equation of state calculations by fast computing machines. The journal of chemical physics, 21(6), 1087-1092.

[58] Mirjalili S., Dong J.S. (2020) What is Really Multi-objective Optimization?. In: Multi-Objective Optimization using Artificial Intelligence Techniques. SpringerBriefs in Applied Sciences and Technology. Springer, Cham.

[59] Miettinen, K., Mäkelä, M. M. (1995). Interactive bundle-based method for nondifferentiable multiobjeective optimization: NIMBUS. Optimization, 34(3), 231-246.

[60] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 6(2), 182-197.

[61] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

[62] Pellegrin, C. (1997). Fondements de la décision en maintenance, Paris, Economica, 112 pages.

[63] Pintelon L., Parodi-Herz A. (2008) Maintenance: An Evolutionary Perspective. In: Complex System Maintenance Handbook. Springer Series in Reliability Engineering. Springer, London.

[64] Kobbacy K.A.H., Murthy D.N.P. (2008) An Overview. In: Complex System Maintenance Handbook. Springer Series in Reliability Engineering. Springer, London.

[65] Rausand M., Vatn J. (2008) Reliability Centred Maintenance.In: Complex System Maintenance Handbook. Springer Series in Reliability Engineering. Springer, London.

[66] Dahite, L., Guibadj, R. N., Fonlupt, C., Kadrani, A., Benmansour, R. (2021, May). A Semi Adaptive Large Neighborhood Search for the Maintenance Scheduling and Routing Problem. In 2021 7th International Conference on Optimization and Applications (ICOA) (pp. 1-6). IEEE.

[67] Ernst, A. T., Jiang, H., Krishnamoorthy, M., Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. European journal of operational research, 153(1), 3-27.

[68] Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. Transportation Science, 46(3), 297-316.

[69] Paquete, L., Chiarandini, M., Stützle, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In

Metaheuristics for multiobjective optimisation (pp. 177-199). Springer, Berlin, Heidelberg.

[70] Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T. (2015). Anytime pareto local search. European journal of operational research, 243(2), 369-385.

[71] Mansour, I. B., Alaya, I., Tagina, M. (2017, September). Chebyshev-based iterated local search for multi-objective optimization. In 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP) (pp. 163-170). IEEE.

[72] Zhang, Q., Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on evolutionary computation, 11(6), 712-731.

[73] Cornu, M., Cazenave, T., Vanderpooten, D. (2017). Perturbed decomposition algorithm applied to the multi-objective traveling salesman problem. Computers and Operations Research, 79, 314-330.

[74] Favaretto, D., Moretti, E., Pellegrini, P. (2007). Ant colony system for a VRP with multiple time windows and multiple visits. Journal of Interdisciplinary Mathematics, 10(2), 263-284.

[75] Hoogeboom, M., Dullaert, W., Lai, D., Vigo, D. (2020). Efficient neighborhood evaluations for the vehicle routing problem with multiple time windows. Transportation Science, 54(2), 400-416.

[76] Roozbeh, I., Hearne, J. W., Pahlevani, D. (2020). A solution approach to the orienteering problem with time windows and synchronisation constraints. Heliyon, 6(6), e04202.

[77] Cordeau, J. F., Laporte, G., Pasin, F., Ropke, S. (2010). *Scheduling technicians and tasks in a telecommunications company*. Journal of Scheduling, 13(4), 393–409.

[78] Kovacs, A. A., Parragh, S. N., Doerner, K. F., Hartl, R. F. (2012). *Adaptive large neighborhood search for service technician routing and scheduling problems*. Journal of scheduling, 15(5), 579–600.

[79] Zamorano, E., Stolletz, R. (2017). *Branch-and-price approaches for the Multiperiod Technician Routing and Scheduling Problem*. European Journal of Operational Research, 257(1), 55–68.

[80] Mathlouthi, I., Gendreau, M., Potvin, J. Y. (2018). *Mixed integer linear programming for a multi-attribute technician routing and scheduling problem*. INFOR: Information Systems and Operational Research, 56(1), 33–49.

[81] Pillac V., Guéret C., Medaglia A.L. (2018). *A Fast Reoptimization Approach for the Dynamic Technician Routing and Scheduling Problem*. In: Amodeo L., Talbi EG., Yalaoui F. (eds) Recent Developments in Metaheuristics. Operations Research/Computer Science Interfaces Series, vol 62. Springer, Cham.

[82] López-Santana, E., Akhavan-Tabatabaei, R., Dieulle, L., Labadie, N., Medaglia, A. L. (2016). *On the combined maintenance and routing optimization problem*. Reliability Engineering and System Safety, 145, 199–214.

[83] Jbili, S., Chelbi, A., Radhoui, M., Kessentini, M. (2018). *Integrated strategy of Vehicle Routing and Maintenance*. Reliability Engineering and System Safety, 170, 202–214.

[84] Chen, Y., Polack, F., Cowling, P., Mourdjis, P., Remde,S. (2016). *Risk Driven Analysis of Maintenance for a Large-scale Drainage System*. In: Proceedings of 5th ICORES, pp. 296-303.

[85] Blakeley, F., Argüello, B., Cao, B., Hall, W., Knolmajer, J. (2003).*Optimizing periodic maintenance operations for Schindler Elevator Corporation*. Interfaces, 33(1), 67-79.

[86] Rashidnejad, M., Ebrahimnejad, S., Safari, J. (2018). *A bi-objective model of preventive maintenance planning in distributed systems considering vehicle routing problem*. Computers and Industrial Engineering, 120, 360-381.

[87] Mladenovic, N., Hansen, P.(1997). *Variable neighborhood search*. Computers and operations research, 24(11), 1097-1100.

[88] Hansen, P., Mladenovic, N. (2003). *A Tutorial on Variable Neighborhood Search*. Groupe d'études et de recherche en analyse des décisions. Les Cahiers du GERAD G–2003–46, HEC Montréal, Canada.

[89] Tsang, A. H. C. (1995). *Condition-based maintenance: tools and decision making*. Journal of Quality in Maintenance Engineering, 1(3), 3–17.

[90] Cota, L. P., Guimarães, F. G., Ribeiro, R. G., Meneghini, I. R., de Oliveira, F. B., Souza, M. J., Siarry, P. (2019). *An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem*. Swarm and Evolutionary Computation, 51,100601.

[91] Raknes, N., Ødeskaug, K., Stålhane, M., Hvattum, L. M. (2017). *Scheduling of Maintenance Tasks and Routing of a Joint Vessel Fleet for Multiple Offshore Wind Farms*. Journal of Marine Science and Engineering, 5(1), 11

[92] Lesain, D., Voudouris, C., Azarmi, N. (2000). *Dynamic Workforce Scheduling for British Telecommunications plc*. Interfaces, 30(1), 45-56.

[93] Duarte, A., Pantrigo, J. J., Pardo, E. G., Mladenovic, N. (2015). *Multi-objective variable neighborhood search: an application to combinatorial optimization problems*. Journal of Global Optimization 63(3), 515-536.

[94] Queiroz, T. A. D., Mundim, L. R. (2020). *Multiobjective pseudo-variable neighborhood descent for a bicriteria parallel machine scheduling problem with setup time*. International Transactions in Operational Research, 27(3), 1478-1500.

[95] Ke, L., Zhai, L. (2014, October). *A Multiobjective Large Neighborhood Search for a Vehicle Routing Problem*. In International Conference in Swarm Intelligence, (pp. 301-308) . Springer, Cham.

[96] Rifai, A. P., Nguyen, H. T., Dawal, S. Z. M. (2016). *Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling*. Applied Soft Computing 40, 42-57.

[97] Lust, T., Teghem, J. (2010). *Two-phase Pareto local search for the biobjective traveling salesman problem*. Journal of Heuristics 16(3), 475-510.

[98] Dahite, L., Kadrani, A., Benmansour, R., Guibadj, R.N., Fonlupt, C. (2020). *Optimization of Maintenance Planning and Routing Problems*. In: Benmansour R., Sifaleras A., Mladenović N. (eds) Variable Neighborhood Search. ICVNS 2019. Lecture Notes in Computer Science, vol 12010. Springer, Cham.

[99] Dahite L., Kadrani, A., Benmansour, R., Guibadj, R.N., Fonlupt, C. Multi-Objective Model and Variable Neighborhood Search Algorithms for the Joint Maintenance Scheduling and Workforce Routing Problem. Mathematics 2022, 10, 1807.

[100] Sindhya, K. An Introduction to Multiobjective Optimization, University of Jyvaskyla, Finland.

[101] Jaimes, A. L., Martınez, S. Z., Coello, C. A. C. (2009). An introduction to multiobjective optimization techniques. Optimization in Polymer Processing, 29-57.

[102] Dohi, T. , Kaio, N., Osaki, S. (2003). *Preventive Maintenance Models: Replacement, Repair, Ordering, and Inspection*.In : Hoang P. (eds) Handbook of Reliability Engineering, pp. 349-366. Springer, London.

[103] Dahite, L., Kadrani A., Benmansour R. (2018). Optimization models for train load and transport planning problems. MATEC Web of Conferences, vol 200, no 00010. EDP Sciences.

[104] Vidal, T., Crainic, T., G., Gendreau, M., Prins, C. (2013). *Heuristics for the multi-attribute vehicule routing problems: A survey and synthesis*. European Journal of Operational Research, 231(1), 1–21.

[105] Rezgui, D., Bouziri, H., Aggoune-Mtalaa, W., Siala, J.C. (2019). *An Evolutionary Variable Neighborhood Descent for Addressing an Electric VRP Variant*. In: Sifaleras A., Salhi S., Brimberg J. (eds) Variable Neighborhood Search. ICVNS 2018. Lecture Notes in Computer Science, vol 11328. Springer, Cham

[106] Chen, P., Huang, H. K., Dong, X. Y. (2010). *Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem*. Expert Systems with Applications, 37(2), 1620-1627.

[107] Thomas, E., Levrat, E., Iung, B. (2008). Overview on opportunistic maintenance. IFAC Proceedings Volumes, 41(3), 245-250.

[108] Thomas, E., Levrat, E., Iung, B., Monnin, M. (2006, August). 'Odds algorithm'-based opportunity-triggered preventive maintenance with production policy. In 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Safeprocess' 06 (pp. 835-840). Elsevier.

[109] Budai-Balke, G., Dekker, R., Nicolai, R. P. (2006). A review of planning models for maintenance and production. Report/Econometric Institute, Erasmus University Rotterdam, (EI 2006-44).

[110] Cheung, K. Y., Hui, C. W., Sakamoto, H., Hirata, K., O Young, L. (2004). Short-term site-wide maintenance scheduling. Computers and chemical engineering, 28(1-2), 91-102.

[111] Tan, J. S., Kramer, M. A. (1997). A general framework for preventive maintenance optimization in chemical process operations. Computers and Chemical Engineering, 21(12), 1451-1469.

[112] Li, Y., Chu, F., Chu, C., Zhu, Z. (2019). An efficient three-level heuristic for the large-scaled multi-product production routing problem with outsourcing. European Journal of Operational Research, 272(3), 914-927.

[113] Ropke, S., Pisinger, D. (2006). *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows*. Transportation science, 40(4), 455-472.

[114] Cota, L. P., Guimarães, F. G., de Oliveira, F. B., Souza, M. J. F. (2017, June). *An adaptive large neighborhood search with learning automata for the unrelated parallel machine scheduling problem*. In 2017 IEEE Congress on Evolutionary Computation (CEC) (pp. 185-192). IEEE.

[115] Pisinger, D., Ropke, S. (2007). *A general heuristic for vehicle routing problems*. Computers and operations research, 34(8), 2403-2435.

[116] Lusby, R. M., Schwierz, M., Range, T. M., Larsen, J. (2016). *An adaptive large neighborhood search procedure applied to the dynamic patient admission scheduling problem*. Artificial intelligence in medicine, 74, 21-31.

[117] Chen Y., Cowling P., Remde S. (2014) *Dynamic Period Routing for a Complex Real-World System: A Case Study in Storm Drain Maintenance*. In: Blum C., Ochoa G. (eds) Evolutionary Computation in Combinatorial Optimisation. EvoCOP 2014. Lecture Notes in Computer Science, vol 8600. Springer, Berlin, Heidelberg.

[118] Laborie, P., Godard, D. (2007). *Self-adapting large neighborhood search: Application to single-mode scheduling problems*. Proceedings MISTA-07, Paris, 8.

[119] Fonseca, C. M., Knowles, J. D., Thiele, L., Zitzler, E. (2005, July). A tutorial on the performance assessment of stochastic multiobjective optimizers. In Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005) (Vol. 216, p. 240).

[120] Pisinger, D., Røpke, S. (2010). *Large Neighborhood Search*. In M. Gendreau (Ed.), Handbook of Metaheuristics (2 ed., pp. 399-420). Springer.

[121] Amine, K. (2019). Multiobjective simulated annealing: Principles and algorithm variants. Advances in Operations Research, 2019.

[122] Alipour, M. M. (2012). *A learning automata based algorithm for solving capacitated vehicle routing problem*. International Journal of Computer Science Issues (IJCSI), 9(2), 138.

[123] Khalfay, A., Crispin, A., Crockett, K. (2017, September). *A review of technician and task scheduling problems, datasets and solution approaches*. In 2017 Intelligent Systems Conference (IntelliSys) (pp. 288-296). IEEE.

[124] Yu, Z., Zhang, P., Yu, Y., Sun, W., Huang, M. (2020). *An Adaptive Large Neighborhood Search for the Larger-Scale Instances of Green Vehicle Routing Problem with Time Windows*. Complexity.

[125] Hemmelmayr, V. C., Cordeau, J. F., Crainic, T. G. (2012). *An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics*. Computers and operations research, 39(12), 3215-3228.

[126] Sze, J. F., Salhi, S., Wassan, N. (2016). A hybridisation of adaptive variable neighbourhood search and large neighbourhood search: Application to the vehicle routing problem. Expert Systems with Applications, 65, 383-397.

[127] Shaw P. (1998) Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: Maher M., Puget JF. (eds) Principles and Practice of Constraint Programming — CP98. CP 1998. Lecture Notes in Computer Science, vol 1520. Springer, Berlin, Heidelberg.

[128] Fontecha, J. E., Akhavan-Tabatabaei, R., Duque, D., Medaglia, A. L., Torres, M. N., Rodríguez, J. P. (2016). *On the preventive management of sediment-related sewer blockages: a combined maintenance and routing optimization approach*. Water Science and Technology, 74(2), 302-308.

[129] Çakırgil, S.; Yücel, E.; Kuyzu, G. An integrated solution approach for multi-objective, multi-skill workforce scheduling and routing problems. *Comput. Oper. Res.* **2020**, *118*, 104908.

Appendix A

Résumé étendu

Outline of the current chapter

A.1 Contexte et motivation

Les systèmes sont nécessaires au fonctionnement de notre société. Ils sont présents dans tous les secteurs et aspects de notre vie : systèmes de transport, systèmes de communication, services publics, usines de fabrication, usines de transformation, hôpitaux et banques [3].

Les systèmes se dégradent avec l'âge et l'usage. Une défaillance de certains systèmes complexes critiques peut avoir des conséquences dramatiques sur la sûreté et la sécurité. Le crash d'un avion ou l'effondrement d'un pont en sont des exemples illustratifs [3]. En outre, des pertes économiques et des dommages environnementaux peuvent être encourus.

En raison de la nature du comportement de vieillissement des systèmes, une bonne stratégie de maintenance doit être conçue et appliquée pour réduire le risque de défaillance. La maintenance vise à maintenir un équipement en fonctionnement ou à le ramener à l'état antérieur où il remplit sa fonction requise. La gestion de la maintenance de systèmes complexes est souvent compliquée et nécessite une stratégie adaptée. La stratégie de maintenance doit décrire les opérations de maintenance appropriées ainsi que la manière et le moment de les exécuter. Elles comprennent la maintenance préventive et corrective (réparation ou remplacement), l'inspection et la surveillance.

La maintenance a évolué au fil des années, passant d'une préoccupation technique à une une question de gestion stratégique. L'évolution des technologies a également ajouté des dimensions scientifiques et technologiques à la maintenance [3]. En effet, l'externalisation de la maintenance, le suivi en temps réel à l'aide de capteurs et l'analyse des données de maintenance sont souvent rencontrés dans ce domaine d'étude [3].

Les entreprises sont davantage conscientes de l'importance de la gestion de la maintenance des actifs. La plupart externalisent cette fonction à des sociétés spé-

cialisées, car elles comprennent la complexité de sa gestion et de son exécution. C'est aussi l'occasion pour elles de se concentrer sur leur activité principale. Le fournisseur de services de maintenance a pour objectif de fournir des approches efficaces de la gestion du service de maintenance par une planification continue afin d'obtenir les meilleurs résultats au moindre coût.

Cette recherche étudie la planification des opérations de maintenance pour un fournisseur de services de maintenance. Ce type d'entreprise est une entreprise de services. Une partie essentielle de la qualité du service est liée à la prestation de ce service. Par conséquent, cette entreprise de services doit assurer une maintenance de bonne qualité à ses clients géographiquement répartis. Cette qualité de maintenance est fortement liée au temps optimal de réalisation de ce service.

Des modèles appropriés doivent être construits pour déterminer le niveau optimal de maintenance préventive (PM) et le moment optimal pour l'exécuter afin de réduire les défaillances et les coûts globaux de maintenance. L'ingénierie de la fiabilité est prédominante dans cette phase. Elle contrôle le fonctionnement des systèmes.

Les politiques de maintenance mathématiques sont utilisées pour développer un plan de maintenance préventive efficace. L'objectif de ces politiques est de concevoir un programme de maintenance comprenant à la fois le remplacement préventif et le remplacement correctif. Les modèles de maintenance qui traitent de ces politiques sont basés sur la théorie des probabilités. La raison principale est l'incertitude du mécanisme à l'origine de la défaillance [102].

Le service de maintenance est également étroitement lié à la gestion des transports pour un prestataire de services. Les techniciens doivent arriver au bon moment pour les opérations de maintenance des différents clients. Il est conditionnée par le respect des délais et l'organisation efficace du routage des techniciens de maintenance. L'arrivée des techniciens doit coïncider avec le moment optimal des opérations de maintenance tout en satisfaisant les contraintes de prestation de service. La gestion du transport est fortement impliquée dans cette phase. Il est nécessaire de séquencer les visites et de déterminer la meilleure politique de routage pour les machines géographiquement distribuées. Les problèmes de tournées de véhicules (VRP) sont des outils essentiels pour modéliser le transport au sein d'un réseau. Le VRP de base utilise quelques informations: les emplacements des clients et les temps ou distances de déplacement.

Des contraintes plus sophistiquées ont été ajoutées par la suite pour faire face à la complexité des exigences du monde réel telles que les fenêtres de temps, les temps de trajet stochastiques, la planification en temps réel, etc. Le problème de tournées des véhicules avec fenêtres de temps (VRPTW) est la variante la plus utilisée en raison de ses applications théoriques et pratiques. Le VRPTW a été

utilisé avec succès dans plusieurs applications, notamment la planification des opérations de maintenance.

A.2 Énoncé du problème, objectifs et cibles de la recherche

La planification de la main-d'œuvre consiste à construire des horaires de travail pour le personnel afin de permettre à l'organisation de répondre à la demande de services. Il vise à programmer l'affectation de la main-d'œuvre pour effectuer des tâches. Il s'agit d'un problème d'ordonnancement rencontré par de nombreuses organisations où la main-d'œuvre, les techniciens, est la principale ressource.

Les fournisseurs de services de maintenance font partie de ces organisations. Ils doivent planifier les opérations de maintenance sur des machines géographiquement dispersées de manière efficace tout en satisfaisant les contraintes opérationnelles. Le problème est complexe, surtout si l'on tient compte de d'autres exigences qui découlent de situations réelles, comme l'incertitude.

Dans ce qui suit, les machines sont sujettes à des défaillances aléatoires qui peuvent par conséquent causer des pertes économiques substantielles et des dommages environnementaux. Le fabricant de l'équipement définit parfois les temps de réalisation de la maintenance. Cependant, les entreprises clientes confient généralement à leurs prestataires de services de maintenance la détermination de la planification de la maintenance qui assure un service de haute qualité au coût global le plus bas. En conséquence, le prestataire de services essaie de réduire au maximum le nombre d'opérations de maintenance, ce qui peut être réalisé en maximisant les périodes entre deux visites successives. En même temps, il cherche à éviter le coût énorme résultant de l'exécution d'opérations de maintenance corrective lorsque la machine tombe soudainement en panne. Il s'agit donc de trouver le meilleur compromis du nombre de visites à chaque machine pour effectuer les opérations de maintenance. Les techniciens sont ensuite acheminés vers les opérations de maintenance pour arriver à temps.

La figure A.1 décrit le problème. Plusieurs machines sont situées sur des sites clients dispersés. Elles sont sujettes à des pannes aléatoires dues à la défaillance d'un composant critique. Pour chaque machine et à intervalles de temps réguliers, des interventions de maintenance préventive (PM) sont programmées. Les techniciens doivent effectuer les opérations PM au meilleur moment possible afin de réduire les coûts de maintenance et de transport. En cas de pannes soudaines, les techniciens effectuent des opérations de maintenance corrective (CM).

Le décideur dans ce type d'entreprise doit concevoir une stratégie de main-

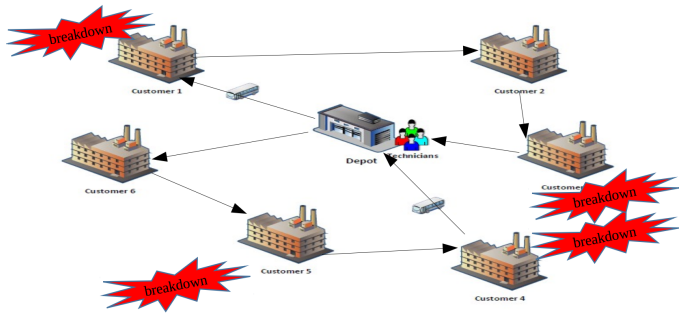

Figure A.1: Le schéma du problème

tenance et de transport des techniciens avec une vision globale du réseau de la chaîne logistique en intégrant efficacement les questions techniques et organisationnelles. Le coût, la disponibilité, la fiabilité et la maintenabilité sont les critères d'évaluation des décisions de maintenance, tout comme les objectifs de coût, de qualité et de temps en logistique [62]. Dans la gestion de la chaîne logistique, il est essentiel de trouver un équilibre entre le coût et la valeur. La valeur est mesurée par la qualité d'un produit ou d'un service et le délai de livraison. La maintenabilité est un concept critique utilisé lorsque les considérations de maintenance sont prises en compte dans les étapes de conception des systèmes [64]. L'ingénierie de la fiabilité contrôle le fonctionnement des systèmes. La fiabilité est la probabilité qu'un système fonctionne correctement et fournisse les résultats souhaités. La disponibilité est la probabilité qu'un système soit opérationnel et disponible pour être utilisé. Tous ces concepts sont interconnectés. Un système bien conçu est plus fiable. Un système fiable est moins susceptible de tomber en panne. Un système moins susceptible de tomber en panne a plus de temps pour fonctionner et est disponible. Une disponibilité, une fiabilité et une maintenabilité élevées réduisent considérablement les coûts de maintenance.

La société de maintenance souhaite trouver le meilleur compromis entre les coûts d'entretien et de maintenance d'une part et les coûts d'exploitation et de transport d'autre part. Cependant, de nos jours, les fabricants s'orientent vers l'intégration d'objectifs supplémentaires afin de construire des systèmes intégrés basés sur la vision globale de la chaîne logistique. Cette thèse vise à considérer conjointement les aspects techniques de la maintenance et organisationnels de la gestion des opérations. La thèse vise à concevoir des stratégies efficaces d'opérations de maintenance des actifs et à assurer leur exécution de la meilleure façon possible.

Déterminer la stratégie optimale de maintenance d'actifs géographiquement

distribués nécessite de construire des modèles appropriés. La gestion du transport des techniciens est nécessaire pour assurer le respect de cette stratégie dans le temps. Toutes les parties prenantes doivent être prises en compte dans la conception du problème global. La modélisation doit intégrer les particularités de ce dernier. L'ingénierie de la fiabilité et la gestion du transport sont des enjeux importants dans la conception et la gestion d'un tel réseau logistique.

L'utilisation de techniques d'optimisation est alors essentielle pour résoudre le problème. Des algorithmes performants ont été développés dans la littérature pour plusieurs problèmes combinatoires et les VRPs. Cependant, il reste encore plusieurs défis à relever pour traiter ces problèmes. En effet, ils proviennent de scénarios du monde réel en raison du nombre de parties impliquées et de leur interaction complexe, tels que les objectifs multiples, la grande échelle, l'incertitude, le temps et l'effort de calcul limités, etc. Par conséquent, la thèse se concentrera également sur la conception d'algorithmes efficaces à objectif unique et multi-objectifs capables de résoudre le problème identifié dans un délai raisonnable.

A.3 Approches empiriques et théoriques

L'approche empirique utilisée jusqu'à présent est une approche mixte séquentielle exploratoire (quantitative puis qualitative) avec une dominance de l'approche quantitative. En effet, les hypothèses sont générées qualitativement sur la base de la réalité du terrain et de l'examen critique de la littérature antérieure. Des cas individuels sont également étudiés. Ces hypothèses sont ensuite testées quantitativement par la simulation d'un jeu de données généré sur les modèles et algorithmes construits. La position épistémologique aménagée est adoptée pour la conception de notre recherche, combinant une approche de recherche inductive (constructiviste et interprétative) et une approche hypothético-déductive (positiviste). La première étape est utilisée lors de la construction du système d'aide à la décision. Nous considérons la nature complexe des situations de gestion et les dimensions humaines dans la construction du système d'aide à la décision destiné aux gestionnaires. La visée projective, et pas seulement interprétative, des connaissances produites contribue à cette approche des sciences de l'ingénieur et de gestion ainsi que dans notre cas. L'application de nos modèles sera également généralisée à toutes les industries similaires, ce qui soutient cette approche puisque nous partons d'un point particulier vers un cas général. La deuxième approche est utilisée lors de la construction des méthodes et modèles mathématiques. Nous commençons par des modèles mathématiques génériques de maintenance et de routage pour construire un modèle particulier inexistant. La même approche est adoptée dans la conception de nos algorithmes de résolu-

tion. La combinaison de modèles mathématiques inhérente à notre conception nécessite cette seconde approche généralement adoptée pour les sciences naturelles. Des détails sur la conception d'une méthodologie de recherche sont disponibles dans [4].

A.4 État de l'art et revue critique de la littérature

Plusieurs sujets pertinents de la littérature sont passés en revue. Le problème est analysé à la fois du point de vue de l'ingénierie technique et de la gestion dans le chapitre 1. La première partie du chapitre présente les méthodes de maintenance depuis leur création jusqu'à aujourd'hui. L'accent est mis sur la distinction entre les différents concepts et leur interaction. Les orientations de maintenance sont présentées afin de comprendre l'essence des notions impliquées dans la définition des concepts de maintenance dans la littérature. Elles ont conduit dans la littérature à la définition de concepts essentiels de la maintenance, et certains de ces concepts ont même évolué pour devenir des stratégies de maintenance. Ces dernières sont passées en revue en mettant l'accent sur la Maintenance Centrée sur la Fiabilité utilisée dans cette thèse. Les modèles de base formalisés pour l'optimisation des intervalles de maintenance utilisés dans l'ingénierie de la fiabilité sont ensuite discutés. Ces modèles de maintenance peuvent être basés sur le temps (Time Based Maintenance TBM) [62] [102] ou sur la condition des équipements (Condition Based Maintenance CBM) [62] [102]. Les modèles de maintenance basés sur le temps incluent le modèle de remplacement d'équipements en fonction de l'âge (Age Replacement Policy ARP) [62] [102], le modèle de remplacement périodique [82] [62] [102], le modèle de remplacement périodique avec réparation minimale [62], et le modèle avec commandes de pièces de rechange [102]. Enfin, la deuxième partie du chapitre présente les principales variantes du problème de tournées des véhicules (VRP). Le problème original du voyageur de commerce (TSP) a été étendu à plusieurs variantes du VRP pour répondre à un large éventail d'applications industrielles. Cependant, une analyse plus poussée a conduit l'étude à se concentrer sur le problème de tournées des véhicules avec fenêtres de temps (VRPTW). Le VRPTW impose les visites dans des fenêtres de temps donnée. Avec la croissance des plateformes de calcul et des problèmes rencontrés dans l'industrie, des techniques de résolution rapides et robustes doivent être développées. Le chapitre 2 est consacré à la littérature relative aux méthodes de résolution. L'accent est mis sur les approches approximatives. En effet, les approches exactes prennent un temps très long pour produire une solution, dépendant de la taille du problème à résoudre. Le chapitre 2 commence par un aperçu des différentes classes de problèmes d'optimisation pour identifier les problèmes étudiés dans cette thèse. Ensuite, les

heuristiques et métaheuristiques existantes et leurs catégories sont présentées et discutées. Enfin, le chapitre passe en revue les classes et les techniques d'optimisation multi-objectifs.

A.4.1 État de l'art et revue critique de la littérature sur le problème conjoint de la planification de la maintenance et du routage des techniciens

La majorité des études existantes abordant les problèmes de routage et de maintenance séparément. Deux principaux courants de recherche peuvent être distingués dans la littérature :

> Premier courant: Le routage est utilisé pour planifier les opérations de maintenance.
>
> Second courant: La maintenance et le routage sont considérés comme un problème intégré en tenant compte de leurs spécificités respectives.

Le premier courant de recherche comprend les problèmes de planification des techniciens qui sont particulièrement utiles dans la réalité et qui concernent de nombreuses organisations. En effet, le coût de la main-d'œuvre constitue l'un des coûts les plus élevés de toute organisation. Les principaux problèmes abordés dans le premier courant sont : Technician Routing and Scheduling Problem (TRSP) [81], Technician and Task Scheduling Problem (TTSP) [77], Service Technician Routing and Scheduling Problem (STRSP) [78], Workforce Scheduling, et Geographically Distributed asset Maintenance Problems (GDMP) [84]. Le deuxième courant de recherche s'intéresse à la combinaison des caractéristiques de la maintenance et du routage. La planification de la main-d'œuvre n'est pas le seul problème traité dans ce cas puisque d'autres problèmes de maintenance sont pris en compte. L'analyse de la fiabilité peut être incluse pour concevoir des solutions qui affectent les techniciens au bon moment pour effectuer les opérations de maintenance. Les coûts de main-d'œuvre et les coûts de maintenance sont tous deux traités dans ce cas. Les recherches combinant les stratégies de maintenance préventive et corrective sont relativement peu nombreuses, car la plupart des articles examinés ne traitaient que de la planification de la maintenance préventive, à l'exception de [82,84]. De plus, ils ne considéraient pas l'incertitude des pannes et ne traitaient pas le problème d'une perspective multi-objectifs. Les articles qui ont pris en compte les pannes aléatoires parmi ceux examinés sont [82, 83]. La formulation multi-objectifs est considérée dans [86, 129]. L'analyse des problèmes précédents met en évidence la nécessité de proposer des solutions au problème réel qui comprend les caractéristiques réalistes suivantes : grande

échelle, incertitude, combinaison de la maintenance corrective et préventive, et enfin, routage des techniciens dans une perspective multi-objectifs. À notre connaissance, aucune étude antérieure n'a étudié ces caractéristiques ensemble. Ces aspects rendent le problème d'optimisation NP-Difficile encore plus difficile. Par conséquent, les approches heuristiques sont nécessaires pour les grandes instances car les méthodes exactes sont pratiquement limitées.

A.4.2 État de l'art et revue critique de la littérature sur le problème conjoint de la planification de la maintenance opportuniste et du routage des techniciens

Nous avons effectué une étude de la littérature pour proposer une extension nouvelle pour le problème de thèse. Nous avons pensé à la production comme composante intéressante pouvant s'ajouter à notre modèle. Nous avons réalisé une étude bibliographique approfondie qui nous a permis de détecter que les aspects liés à la production doivent être considérés pendant le processus d'optimisation de la maintenance afin de réduire les coûts globaux. En effet, la planification des opérations de maintenance dans des intervalles d'arrêt de production permet d'obtenir un gain considérable lorsque c'est possible. Les problèmes de planification conjointe de la maintenance et de la production représentent un large champ de recherche dans la littérature avec différents courants existants. Nous les avons analysés pour prouver l'originalité du problème triple proposé combinant maintenance, production et routage.

Deux principaux courants de recherche peuvent être distingués dans la littérature des problèmes intégrés avec le VRP et qui sont nécessaire à l'analyse de notre problème:

> Premier courant: Le problème de planification des opérations de maintenance et de routage des techniciens. Il s'agit de l'intégration du problème de tournées de véhicules avec la maintenance.
>
> Second courant: L'intégration du problème de tournées de véhicules avec la production.

L'intégration du problème de maintenance et de production est un large champ de recherche dans la littérature où les courants suivant peuvent être distingués [109]: la planification de la maintenance tout en tenant compte de la production, la planification de la production en tenant compte de la maintenance, et enfin, la planification de la maintenance comme s'il s'agissait d'un processus de production. Le premier courant est composé des sous-courants [109]: le calcul du coût des temps d'arrêt, la maintenance opportuniste et la planification

de la maintenance en phase de la production. La maintenance opportuniste (OM) est définie [107] comme le regroupement d'opérations de maintenance, l'association d'une opération de maintenance préventive à des opérations de maintenance corrective pour bénéficier de la proximité entre les composants et enfin la réalisation d'une opération de maintenance préventive lors d'une opportunité.

A notre connaissance, aucun travail antérieur dans la littérature ne considère la planification du routage et de la maintenance avec des contraintes de production.

A.4.3 État de l'art et revue critique des méthodes d'optimisation multi-objectifs

Les problèmes multi-objectifs sont généralement classés en fonction du rôle joué par le décideur (DM) dans le processus de décision. Les quatre classes comprennent les méthodes sans préférence, a priori, a posteriori et interactives. Dans les méthodes sans préférence, il n'y a pas de décideur, le problème multi-objectifs (MOP) est résolu et une solution optimale unique de Pareto est retournée. Il s'agit de la solution la plus proche du point idéal en termes de distance euclidienne. Dans les méthodes a priori, un DM exprime sa préférence avant l'optimisation. Dans la plupart des méthodes à priori, les objectifs sont agrégés en un seul objectif. La méthode a priori la plus populaire est appelée l'approche de la somme pondérée. Les autres méthodes incluses dans cette catégorie sont l'ordre lexicographique et le goal programming. Dans ces deux classes, le MOP est transformé en un problème mono-objectif. La troisième classe de méthodes MO est la méthode a posteriori. Dans cette classe, l'objectif est de trouver toutes les solutions optimales de Pareto. Le décideur en choisit une parmi elles dans la deuxième étape. Des exemples de méthodes de cette catégorie sont la méthode de la contrainte epsilon, l'approche de la somme pondérée avec variation des poids, les métaheuristiques de recherche locale et les algorithmes multi-objectifs évolutionnaires (EMO). Ils se répartissent en trois groupes principaux : les algorithmes basés sur l'agrégation, la dominance et les indicateurs de performance [100]. Les méthodes interactives et progressives sont similaires dans la résolution et la formulation aux méthodes a posteriori. Dans les deux catégories, la formulation multi-objectifs du problème est maintenue. Cependant, dans les méthodes interactives, le DM guide le processus d'optimisation en donnant sa préférence pendant l'optimisation. Les méthodes de référence en optimisation multi-objectifs sont des méthodes a posteriori basées sur la population et utilisant le concept de dominance de Pareto. Cependant, les méthodes basées sur la population montrent une convergence lente lorsqu'elles sont appliquées à

un VRP à objectif unique [104]. Les métaheuristiques basées sur la trajectoire sont très efficaces pour le problème du VRP à objectif unique mais sont moins utilisées dans le contexte multi-objectif car elles utilisent traditionnellement une solution unique. À notre connaissance, il n'existe pas de méthode multi-objectif de référence basée sur la recherche locale, à l'exception de la recherche locale de Pareto (PLS) [69]. Il est nécessaire de proposer des métaheuristiques multi-objectifs basées sur la trajectoire, efficaces pour ce problème et pour le VRP en général. Ces méthodes doivent être adaptées pour assurer la diversification suffisante habituellement obtenue en utilisant une population de solutions et des opérateurs génétiques.

A.5 Contributions

Cette thèse traite du problème de la planification de la maintenance et du routage de la main-d'œuvre. Le problème se pose pour les fournisseurs de services de maintenance. Cette société de service doit assurer une bonne qualité de service au coût global le plus bas à ses clients géographiquement distribués. La qualité de ce service est liée au bon moment pour exécuter les opérations de maintenance. Ce bon moment est d'abord lié aux modèles de fiabilité qui servent à le déterminer. Ensuite, les techniciens doivent arriver au bon moment en suivant un plan de routage conçu à cet effet. Dans cette thèse, nous avons examiné de plus près comment les problèmes de tournées des véhicules et l'ingénierie de la fiabilité peuvent contribuer à la planification de la maintenance des fournisseurs de services. Nous étudions le problème d'un point de vue technique et de gestion afin de comprendre comment intégrer les aspects techniques de la fiabilité aux aspects organisationnels de la gestion du transport. Une analyse plus poussée amène l'étude à se concentrer sur les problèmes de tournées de véhicules, en particulier le problème de tournées de véhicules avec fenêtres de temps (VRPTW). Cette variante est utilisée pour planifier le transport au sein d'un réseau. Des contraintes plus sophistiquées ont été ajoutées pour faire face à la complexité des exigences du monde réel spécifiques au problème étudié. Les considérations techniques de maintenance sont intégrées comme des contraintes et des objectifs supplémentaires qui doivent être pris en compte pour résoudre le problème. Nous proposons de nouveaux modèles basés sur les paradigmes de modélisation de la fiabilité et du VRP. Dans la deuxième partie de la thèse, la maintenance opportuniste est considérée avec la maintenance préventive et corrective. Le problème doit satisfaire à la fois ses contraintes opérationnelles et ses contraintes de production. La maintenance opportuniste est exécutée pendant l'arrêt planifié des systèmes. Les problèmes définis sont trop difficiles pour les solveurs commerciaux contemporains pour des instances de taille in-

téressante. De plus, trouver les solutions optimales pour ces problèmes est crucial et pourtant difficile. Par conséquent, nous proposons des heuristiques innovantes et des algorithmes mono-objectif et multi-objectifs pour résoudre les problèmes et gérer les contraintes opérationnelles. Chaque équipement se dégrade avec le temps. Les décisions de maintenance déterminent le moment où les opérations de maintenance doivent être exécutées. Les décisions de routage doivent garantir que les services de maintenance sont fournis à temps. Nous avons développé des modèles et des algorithmes pour le problème étudié et son extension en nous basant sur l'étude bibliographique susmentionnée. En effet, ce manuscrit regroupe nos investigations sur le problème étudié et sur plusieurs méthodes approchées.

Les contributions de cette thèse sont résumés ci-dessous.

A.5.1 Le problème de la planification de la maintenance et du routage des techniciens: Modèle et approches de résolution basées sur la recherche à voisinage variable

Dans le chapitre 3, nous abordons un problème conjoint de planification de la maintenance et de routage de la main d'œuvre. L'ensemble des machines est supposé être géographiquement distribué et sujet à des pannes aléatoires. Nous cherchons à déterminer les moments optimaux pour effectuer les opérations de maintenance préventive sur chaque machine et à trouver la séquence optimale des opérations de maintenance à exécuter qui minimise chaque coût considéré. Nous avons proposé un nouveau modèle mathématique qui vise à minimiser les coûts de défaillance, de maintenance et de transport dans le cas d'une maintenance préventive basée sur le temps. Nous avons proposé un coût de défaillance stochastique non linéaire qui utilise des informations provenant de la dégradation de l'équipement. Il comprend des coûts directs et des coûts indirects. Cet objectif est à considérer pour les industries à haut risque où la sécurité est une préoccupation primordiale. Nous étudions également le coût de maintenance proposé précédemment dans la littérature. Il équilibre les coûts préventives et correctives liées aux opérations de maintenance. Le temps de la dernière restauration est inclus dans les deux coûts pour vérifier l'hypothèse de la théorie du renouvellement. Ces objectifs sont étudiés en même temps que le coût de routage. Enfin, le modèle prend en compte plusieurs contraintes opérationnelles telles que les fenêtres de temps des opérations de maintenance, les pénalités liées aux non-respect des fenêtres de temps de maintenance, et l'incertitude des pannes pour s'adapter aux problèmes industriels réels. De nouvelles heuristiques constructives basées sur le comportement des défaillances et de la maintenance sont proposées pour générer des solutions initiales, suivies

d'une métaheuristique de recherche générale à voisinage variable, General Variable Neighborhood Search (GVNS), pour résoudre le problème. La performance de l'algorithme GVNS est examinée sur des instances générées. Les résultats de calcul indiquent que les solutions initiales dédiées améliorent la qualité des solutions finales et réduisent considérablement le temps d'exécution de la métaheuristique GVNS. Les résultats expérimentaux révèlent également l'efficacité de l'algorithme GVNS. En effet, les résultats de l'algorithme ont largement dépassé ceux du solveur commercial.

La figure A.2 montre les deux cycles de maintenance, un cycle préventif suivi d'un cycle correctif.

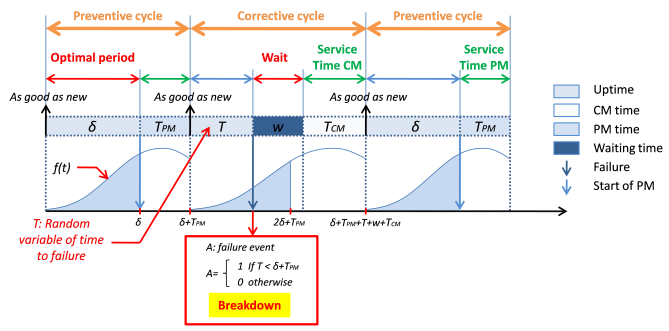

Figure A.2: Exemple de cycles de maintenance avec des opérations de maintenance préventive et corrective [82]

La représentation de la solution utilisée est illustrée dans la figure A.3. La solution est représentée par K tours où chaque tour est une permutation de tâches de maintenance. Dans l'exemple de la figure, la solution qui minimise le coût est composée de deux tours (deux véhicules sont requis) et chaque tour comprend trois opérations de maintenance dans l'ordre indiqué. Nous disposons de 5 machines. Une des machines requiert les deux opérations 3 et 4.

Routing cost	Failure cost	Maintenance cost	Penalty
110.88	935.206	43.663	0

Route 1	Task	1	5	6
	Start time	36.2588	41.7832	45.5967
	Renewal time	0	0	0

Route 2	Task	3	2	4
	Start time	29.4947	44.109	62.657
	Renewal time	0	0	29.4947

Figure A.3: Représentation de la solution du résultat minimisant le coût de routage pour l'instance Re/6/2/80/80/0.07/2 (temps en heures)

A.5.2 Le problème conjoint de la planification de la maintenance et du routage des techniciens : Algorithmes multi-objectifs de recherche et de descente à voisinage variable

Le chapitre 4 étudie la variante bi-objectif du problème précédent. Le modèle bi-objectif traite les mêmes contraintes. En outre, nous associons le coût de défaillance ou le coût de maintenance au coût de routage au niveau des fonctions objectifs. Cette approche vise à déterminer les moments optimaux pour effectuer les opérations de maintenance préventive sur chaque machine et la séquence optimale de ces opérations qui minimise simultanément les coûts de routage et de maintenance ou de défaillance. Des adaptations de la descente à voisinage variable, Variable Neighborhood Descent, et de la recherche générale à voisinage variable, General Variable Neighborhood Search, qui sont des métaheuristiques mono-objectif, sont proposées pour contenir des scénarios à objectifs multiples. Elles sont appelées MOVND/P, MOVND/PI et MOGVNS/P. La conception de chaque composant est détaillée, y compris la méthode d'amélioration, la recherche locale multi-objectifs, le critère d'acceptation, le critère d'arrêt et la procédure de changement de voisinage. Ces algorithmes sont basés sur le concept de la dominance de Pareto. La recherche locale multi-objectifs incorporée proposée, Multi-Objective Best Improvement Strategy based on Pareto dominance (MOBI/P), s'est avérée être une nouvelle stratégie multi-objectifs de recherche du meilleur améliorant. Tout en explorant le voisinage, la méthode MOBI/P permet une convergence rapide si la nouvelle solution domine la solution courante et assure une diversification dans le cas de solutions incomparables. Les méthodes sont simples, avec peu de paramètres à régler. Cette caractéristique a permis l'intégration de l'algorithme de recherche locale MOVND/P plus

tard dans la thèse dans des cadres multi-phases avancés. Plusieurs comparaisons avec les métaheuristiques multi-objectifs VND et GVNS existantes [93] sont effectuées. Les résultats expérimentaux révèlent l'efficacité des algorithmes proposés sur tous les indicateurs de qualité et de temps mesurés. Plusieurs variantes basés sur GVNS sont proposés. Leurs performances sont évaluées pour mesurer l'influence des techniques de littérature telles que l'approche de décomposition ou l'utilisation d'une population initiale de solutions.

A.5.3 Le problème conjoint de la planification de la maintenance opportuniste et du routage des techniciens : Modèle et une approche de résolution basée sur la recherche adaptative à voisinage variable

Le chapitre 5 étend le problème précédent pour traiter de la Maintenance Opportuniste (OM). La politique de maintenance opportuniste est un nouveau concept fortement lié au système de maintenance optimale. Elle consiste à planifier la maintenance au bon moment sur les équipements tout en pénalisant le moins possible le processus de production. La détermination de ce bon moment est difficile car elle inclut non seulement des considérations techniques sur la durée de vie de la machine mais aussi des considérations de gestion sur le routage des techniciens. De plus, les arrêts de production programmés doivent être favorisés pour réduire l'impact négatif sur la production. À cet égard, le problème est caractérisé par trois aspects interdépendants : le routage, la maintenance et la production qui influencent la décision finale. À notre connaissance, cette thèse présente la première étude qui inclut la formalisation du concept de maintenance opportuniste dans un contexte de planification des tournées de la main-d'œuvre. Nous avons conçu un nouveau modèle qui est conforme à la formalisation précédente. Des contraintes et des coûts objectifs supplémentaires sont considérés. Les pertes de production dépendent de l'utilisation des intervalles d'arrêts de production pour les opérations de maintenance. Une nouvelle heuristique, CSTMPSTW, est proposée pour déterminer le meilleur moment de démarrage dans les intervalles de maintenance et pour sélectionner un arrêt de production. Elle est intégrée à une recherche semi-adaptative à voisinage large, Semi Adaptive Large Neighborhood Search (SALNS), pour résoudre le problème. SALNS est une adaptation de la métaheuristique ALNS qui intègre un nouvel opérateur de destruction basé sur le risque de défaillance. Il détruit les opérations de maintenance relatives aux équipements ayant une probabilité potentiellement élevée de défaillance. Elle intègre également un mécanisme semi-adaptatif permettant d'améliorer significativement les performances de l'approche de résolution. Ce dernier vise à trouver un équilibre entre la procé-

dure de diversification et l'apprentissage à partir des informations passées lors de la sélection. Les résultats de calcul démontrent la faisabilité du modèle. En effet, il permet de satisfaire un nombre important de contraintes supplémentaires. Les expériences computationnelles confirment l'efficacité de l'approche de résolution adoptée pour le problème par rapport au solveur commercial. SALNS a également été testé sur le problème conjoint de la planification de la maintenance et du routage des techniciens du Chapitre 3 et a fourni de bons résultats.

A.5.4 Le problème conjoint de la planification de la maintenance opportuniste et du routage des techniciens : Multi-objectifs modèle et algorithmes de résolution basées sur et la recherche locale de Pareto et la recherche adaptative à voisinage variable

L'objectif du chapitre 6 est de traiter une variante bi-objectif du problème défini précédemment. Plusieurs caractéristiques du problème sont identifiées, notamment la taille du problème et les niveaux de contraintes. En plus du problème précédent, les techniciens commencent les opérations de maintenance dès leur arrivée en respectant les intervalles de maintenance. Nous avons formulé un nouveau modèle bi-objectif qui intègre l'hypothèse supplémentaire. Le premier objectif minimise le coût de routage et les éventuelles pertes de production si les intervalles d'arrêt de production disponibles ne peuvent pas être utilisés partiellement ou entièrement pour les opérations de maintenance. Le second objectif minimise le coût total de maintenance avec le coût de pénalité du non-respect des intervalles de maintenance. Les contraintes précédentes sont considérées, ainsi que des contraintes supplémentaires. Dans la deuxième partie de ce chapitre, des extensions des algorithmes PLS et ALNS sont proposées pour traiter les problèmes d'optimisation combinatoire multi-objectifs en général et ce problème en particulier. Tout d'abord, nous avons proposé PLS-MOVND, qui peut améliorer considérablement les performances de PLS en intégrant des structures de voisinage multiples et plusieurs caractéristiques de conception. Ensuite, nous proposons une recherche multi-objectifs adaptative à voisinage large, appelée MOSALNS/P. L'algorithme MOSALNS/P capture de manière appropriée les nouveaux composants de SALNS et l'étend pour traiter l'optimisation multi-objectifs. De plus, il incorpore MOVND/P comme procédure de recherche locale pour aider la méthode à converger vers le front de Pareto complet. Enfin, les algorithmes proposés, MOSALNS/P, PLS-MOVND, MOVND/PI, et l'ALNS multi-objectif de la littérature [90], ont été comparés. Les résultats expérimentaux révèlent l'efficacité des méthodes dans tous les indicateurs de temps et de qualité

mesurés. En effet, toutes les méthodes proposées surpassent considérablement la méthode multi-objectif ALNS de la littérature [90]. De plus, les deux méthodes, PLS-MOVND et MOVND/PI, avec des performances presque similaires, ont surpassé MOSALNS/P. Les performances exceptionnelles de PLS-MOVND et MOVND/PI et leur conception encouragent leur application à de nouveaux problèmes complexes. Ils sont également conçus de manière efficace et sans paramètres à régler, ce qui permet de les intégrer facilement dans d'autres cadres multi-phases.

A.6 Recherche future

Cette recherche fournit des modèles et des algorithmes efficaces qui peuvent être intégrés à un système d'aide à la décision pour traiter les problèmes de gestion tactique d'un fournisseur de services de maintenance. Les questions et les pistes de recherche ouvertes sont, en fait, nombreuses. Les recherches futures dans ce domaine peuvent aborder le même problème sous d'autres angles ou étendre les contributions pour répondre davantage aux exigences du monde industriel. La première piste de recherche future concerne l'exploration de l'intégration de différentes stratégies de maintenance avec le problème de routage pour réduire les coûts de maintenance et améliorer la qualité des services de maintenance. Dans les chapitres 5 et 6, nous avons considéré la maintenance opportuniste et défini un nouveau problème. Étant donné que le courant de recherche concernant la maintenance opportuniste et le problème décrit n'en sont qu'à leurs débuts et qu'ils sont proposés pour la première fois, il existe encore une vaste marge d'amélioration. D'autres éléments de production pourraient être intégrés dans la conception afin d'améliorer l'efficacité du modèle. Un exemple de ces nouveaux éléments est un planificateur de production qui aide les décideurs dans la phase de planification à construire des plans de travail efficaces. Nous avons considéré, dans ce travail, les arrêts de production pour effectuer des opérations de maintenance comme des intervalles d'opportunité. Nous aurions plutôt pu inclure le plan de production comme une contrainte que nous devons respecter. De plus, la production peut être planifiée simultanément avec la maintenance et le routage pour concevoir un modèle complet à trois décisions.

La maintenance basée sur la condition pourrait également constituer une alternative à la maintenance basée sur le temps, car elle sera représentative des scénarios intéressants du monde réel. La maintenance conditionnelle ou prédictive utilise des données pour fournir des modèles de décision basés sur les performances passées. La maintenance peut être retardée jusqu'à ce que certains seuils de paramètres mesurés soient atteints. Les données exploitées

peuvent être liées à l'historique des pannes de maintenance, à la demande de production ou aux plans de travail antérieurs. Une grande quantité de données peut également être enregistrée grâce au suivi des véhicules. Ces données offrent des indications essentielles sur des informations complémentaires telles que le temps de trajet, le temps de service, les perturbations, les retards, etc.

L'étude de la version dynamique des problèmes étudiés présente également un intérêt primordial. Le planificateur peut être conçu de manière dynamique pour interagir en temps réel avec les techniciens en partageant des informations connexes et en réorganisant les plans alternatifs en fonction de la nouvelle situation. En outre, l'incorporation d'informations incertaines (temps de trajet, temps de service, demande, etc.) peut améliorer le réalisme du modèle.

Il est nécessaire de tirer des conclusions managériales sur l'applicabilité des approches proposées à d'autres problèmes. Parmi les orientations futures dans le domaine général, citons l'extension des modèles aux nouveaux systèmes qui apparaissent actuellement dans la planification de la main-d'œuvre pour les prestataires de services. Parmi les exemples, citons les problèmes de soins de santé à domicile, la planification des prestations des infirmières, l'établissement des listes d'infirmières, la planification des équipages et une liste générale et exhaustive des problèmes de transport dans une chaîne logistique. Le développement de ces modèles doit être naturellement adapté aux problèmes afin de s'adapter à la pratique et d'inclure leurs différents scénarios.

D'autre part, cette thèse propose plusieurs algorithmes multi-objectifs basés sur la recherche locale. Ils ont été conçus pour assurer une diversité suffisante, généralement obtenue avec une population de solutions et des opérateurs génétiques. Puisque la plupart des travaux de la littérature se concentrent sur les méthodes basées sur la population, des recherches supplémentaires sont nécessaires pour explorer d'autres algorithmes basés sur la recherche locale et leurs interactions.

Bien que la présente étude démontre l'efficacité des méthodes proposées par rapport à plusieurs méthodes de recherche locale multi-objectifs de la littérature, des études supplémentaires sont nécessaires pour les améliorer en ce qui concerne certains mécanismes. La procédure MOBI/P peut offrir plus de contrôle sur la procédure de diversification, limitant le temps passé dans cette dernière et accélérant la recherche. L'algorithme MOVND/PI peut inclure une procédure de diversification plus sophistiquée pour surpasser MOGVNS/P en termes de qualité de solution. Puisque l'hybridation de la recherche locale et des méthodes basées sur la population a donné des résultats remarquables pour le VRP dans la littérature; il serait prometteur d'hybrider les techniques proposées avec les méthodes basées sur la population pour résoudre ce problème. Une direction de recherche potentiellement intéressante est l'intégration des

algorithmes multi-objectifs proposés, en particulier MOVND/P, MOVND/PI, et PLS-MOVND, dans des cadres plus sophistiqués avec les métaheuristiques multi-objectifs de la littérature. En effet, les méthodes basées sur la population peuvent être couplées avec MOVND/P pour améliorer leurs performances. La recherche peut également être guidée vers des régions plus prometteuses de l'espace de recherche en incluant des mécanismes d'apprentissage dans la procédure de recherche locale. Les techniques d'apprentissage automatique peuvent être utilisées pour piloter les métaheuristiques. Nous pensons sincèrement que guider la recherche en apprenant des performances passées accélérera cette dernière. Cette procédure permettra d'éviter les déplacements inutiles et de réduire considérablement le temps de calcul, et offrira donc la possibilité de résoudre des instances plus importantes. Les techniques d'apprentissage supervisé telles que les réseaux de neurones ou les Support Vector Machines peuvent être entraînées avec des programmes optimaux. Nous pensons que l'apprentissage par renforcement est plus adapté à ces méthodes car il dépend d'un agent interagissant avec l'environnement aléatoire pour apprendre de ses expériences passées en utilisant un système de récompense et de pénalité.

Une partie essentielle de la thèse a été consacrée à la proposition d'heuristiques dédiées pour faire face aux particularités des problèmes définis. Ces heuristiques peuvent être améliorées et couplées pour obtenir de meilleures performances. L'heuristique constructive gloutonne de maintenance, GCHM, doit être combinée avec une version simplifiée de l'heuristique CSTMPSTW qui détermine uniquement les heures optimales de début des opérations de maintenance afin de fournir une solution initiale efficace. Cette solution peut surpasser la solution initiale fournie par la célèbre heuristique de meilleure insertion en termes de temps de calcul et sera particulièrement puissante pour les instances de grande taille. Les heuristiques GCHM et GCHF intégrées à CSTMPSTW peuvent être utilisées par la suite comme nouveaux opérateurs d'insertion pour SALNS et MOSALNS/P.

A.7 Conclusion

Cette thèse s'est concentrée sur le développement de modèles et d'algorithmes pour les problèmes de planification de la maintenance et de routage de la main-d'œuvre. Le modèle proposé a intégré plusieurs caractéristiques du monde réel telles que l'incertitude et les considérations de fiabilité. En outre, le modèle a été étendu pour tenir compte des contraintes opérationnelles critiques liées à la production. Une telle approche vise à intégrer tous les éléments clés qui influencent le plus la structure des coûts. Le point de vue systémique a été le principal moteur du développement des modèles.

La thèse a également été consacrée à la conception d'heuristiques et de métaheuristiques dédiées qui s'adaptent aux tailles des systèmes réels. Leurs objectifs étaient de trouver des solutions de haute qualité dans des temps d'exécution courts tout en considérant plusieurs aspects qui découlent de situations réelles. L'accent a été mis sur la qualité des solutions et le temps passé à les obtenir.

Les approches proposées dans cette thèse s'appliquent à de grands réseaux interconnectés de la vie réelle. Les fournisseurs de services peuvent offrir des services de qualité tout en réduisant leurs coûts de fonctionnement. De plus, les méthodes de résolution proposées tout au long de ce travail peuvent être appliquées à n'importe quel problème d'optimisation combinatoire mono-objectif ou multi-objectifs avec seulement une adaptation appropriée des opérateurs.

Des directions futures ont été proposées dans ce chapitre. Elles peuvent se concentrer sur l'exploration de d'autres aspects auxquels les fournisseurs de services sont confrontés dans la pratique. Davantage de composants et de contraintes opérationnelles ayant un impact direct sur le coût total peuvent être inclus pour permettre le bon fonctionnement du système. D'autres stratégies de maintenance efficaces pourraient être envisagées. L'une des directions de recherche consiste à tirer parti des données en temps réel pour ajuster les plans en conséquence.

En outre, les approches de solution proposées sont actuellement dans une phase précoce, et il y a une grande marge pour l'optimisation, notamment en les intégrant à des algorithmes multi-phases plus sophistiqués. Les métaheuristiques proposées peuvent également être pilotées par des techniques d'apprentissage automatique pour améliorer leurs performances.