SCHEDULING WITH TWO TYPES OF PENALTIES UNDER UNCERTAINTY AND ITS APPLICATION TO MAINTENANCE PLANNING

Hue Chi (Trudy) Lam

Supervisors: Hanyu Gu, Yakov Zinder

A thesis submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

School of Mathematical and Physical Sciences
Faculty of Science
University of Technology Sydney

November 2022

Certificate of Original Authorship

I, Hue Chi (Trudy) Lam declare that the thesis titled "*Scheduling with two types of penalties under uncertainty and its application to maintenance planning*", is submitted in fulfillment of the requirements for the award of Doctor of Philosophy, in the School of Mathematical and Physical Sciences/Faculty of Science at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Name: Hue Chi (Trudy) Lam

Signed: Production Note:
Signature removed prior to publication.

Date: 12 April 2022

Abstract

This thesis studies a number of combinatorial optimization problems with uncertainty, which have applications in the fields of maintenance planning. In particulars, the research in Chapter 3 was conducted as part of the project with a major maintenance center, and Chapter 5 considers a maintenance planning problem faced by a French electricity transmission system operator. The problem was subject of the recent ROADEF/EURO challenge 2020 competition.

This thesis is comprised of several chapters. Chapters 1 and 2 introduce the thesis topics and provide the literature survey, respectively. Chapter 3 focuses on developing maintenance plans for a fleet of passenger trains. The key feature of our model is that it considers the uncertain duration of maintenance, the center's engineering restrictions, and the rail operator's operational requirements. We propose a Genetic Algorithm, an Iterated Local Search, and a hybrid two-stage optimization procedure that combines Jensen's Inequality based relaxation with Iterated Local Search to solve the problem.

Chapter 4 focuses on improving the utilization of resources when scheduling jobs that share multiple resources. The motivation for this research comes from scheduling problems that arise in healthcare and rolling stock maintenance services. The problem is distinct from the problem in Chapter 3: the processing of each job requires several types of resources; and the penalty for capacity violation is calculated based on not only the capacity of resource but also an upper bound on the resource expansion. The solution approaches presented in Chapter 3 cannot be directly applied to this problem because their performances rely on having a good-quality initial solution produced by an exact method. We instead propose a Genetic Algorithm enhanced by local search and present a method for assessing solution quality based on Sample Average Approximation.

Chapter 5 is concerned with a large-scale planning problem arising in the maintenance of power distribution grid. Due to the extreme hazards involved when performing maintenance operations on the high-voltage lines, individual transmission lines have to be shut down for the duration of maintenance. The goal is to build intervention schedules that minimize the impact of planned outages on the reliability of power networks. A unique feature of this problem is that the duration, resource consumption, and risk value of an intervention depend on when it starts. Using the problem instances provided by the ROADEF competition, we demonstrated the effectiveness of the proposed Iterated Local Search with self-adaptive perturbation and obtained the 2nd prize.

Acknowledgements

I would like to start by expressing my sincere gratitude to Dr Hanyu Gu for giving me the opportunity to conduct the research under his supervision. This research would not have been possible without his guidance and advice, as well as the scholarly discussions that have sharpened my thinking and enhanced my knowledge in the area of Operations Research. I would like to thank Professor Yakov Zinder for his advice on my writings and his time and patience in answering my questions. Conversations with Professor Zinder have always encouraged me to challenge my ideas and claims and allowed me to think further than what meets the eyes.

My special thank to the Faculty of Science for providing the HDR fund, which supported me financially to attend two international conferences. The 9^{th} Manufacturing Modelling, Management and Control Conference (MIM), which was held in Berlin, Germany, in August 2019. The 6^{th} International Conference on Computer Science, Applied Mathematics, and Applications (ICCSAMA) was held in Hanoi, Vietnam, in December 2019. During my time in Hanoi, I have also had the chance to visit the Operations Research Laboratory (ORLab) at the University of Engineering and Technology, Vietnam National University. I would like to thank Dr Minh-Hoang Ha for giving me this opportunity and to the members of ORLab for taking the time to work with me and show me around Hanoi.

I gratefully acknowledge the Computational facilities provided by the UTS eResearch High Performance Computer Cluster.

I wish to thank my fellow doctoral students for their encouragement and support at various stages of my graduate study. The shared jokes and occasional drinks after work before lockdown have made this challenging but rewarding PhD journey less stressful and more enjoyable. I am indebted to my family, who has supported me through the most challenging

time, especially my parents in Vietnam, who constantly assured me that the PhD study is completely worth it despite not understanding what my research was. It is to my family that I dedicate this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is concerned with a number of combinatorial optimization problems with uncertainty, which have applications in the fields of industrial maintenance planning. The studied problems have some common inherent challenges: the presence of uncertainty and two types of penalties. Failure to consider the uncertainty may lead to inadequate decisions and adverse effects, including increased costs and disruptions. This thesis aims to address these challenges by proposing mathematical models and developing efficient solution approaches that improve upon existing methods.

The remainder of this chapter is organized as follows. The next section gives a short outline of the rolling stock high-level heavy maintenance planning problem and an overview of the research outcomes. In Section 1.2, we provide a synopsis of the problem of scheduling jobs sharing multiple resources under uncertainty, and draw on the many different applications of the problem. We then present, in Section 1.3, an overview of the grid operation-based outage maintenance planning problem, its unique features and difficulties as compared to previously tackled problems in the area. The last section provides a summary of the detailed contributions of this thesis.

1.1 Planning of Rolling Stock Maintenance

The first problem, discussed in Chapter 3, is an investigation into the optimization of a rolling stock heavy maintenance planning problem at a maintenance center. Classic rolling stock maintenance planning models and most models currently implemented in the industry assume the duration of maintenance is known constants. This usually creates an unrealistic plan that is prone to higher maintenance costs and a higher risk of not having enough trains in the fleet to support operations. The research in Chapter 3 addresses this gap in knowledge by proposing a new maintenance planning model that incorporates the uncertainty concerning the duration of maintenance into the model and studying how to generate maintenance plans efficiently and effectively.

Over the past 100 years, rail transport has played a vital role in the public transportation system of many cities. For the safety of passengers and reliable functioning of the railway system, the rolling stock (or trains), which is the most crucial component of such a system, must undergo regular maintenance at a specialized maintenance center. According to the engineering restrictions, the maintenance has a validity period after which another maintenance procedure must take place. A majority of maintenance has validity periods of 12 years, but a small few last 6 to 8 years.

In general, the maintenance process is comprised of functional inspections, components' change-out, cosmetic services, electrical and mechanical tests, trial and post-trial checks. Many large maintenance centers are equipped with an on-site facility for refitting and refurbishing components so that fresh ones are readily available for change-out. However, for older fleet types, the components must be fixed by external clients. Furthermore, some maintenance operations are only known after the trains have been disassembled. Late delivery of components and unplanned maintenance work are the main causes of uncertainty concerning the duration of maintenance.

Given that the maintenance operations are lengthy procedures, the maintenance center must

produce a maintenance plan (a contract) a year in advance, deciding when the maintenance of each train should commence. This is a long-term decision involved in the negotiation process between two separate organizations: the rolling stock operators and the maintenance center. On the one hand, the rolling stock operators wish the maintenance commencement dates to be close to the expiry dates. On the other, the maintenance center would like to complete the maintenance work on time to avoid contractual penalties while minimizing its cost due to subcontractors.

Chapter 3 presents a nonlinear integer programming formulation of the maintenance planning problem and several optimization procedures, based on metaheuristics, that produce maintenance plans less susceptible to the maintenance duration variability. The key feature of our model and methods is that it considers the uncertain duration of maintenance, the engineering restrictions of the center, and the operational requirements of the rail operator. At the time of implementation, it was reported that the proposed optimization procedures improved upon existing practice and reduced the plan development time from several days to a few hours.

The research in Chapter 3 was presented at the 9^{th} Manufacturing Modeling, Management and Control Conference (MIM 2019) and the 6^{th} International Conference on Computer Science, Applied Mathematics, and Applications (ICCSAMA 2019) in Germany and Vietnam, respectively. The research was included in the refereed conference proceedings as "H. Gu, M. Joyce, H.C. Lam, M. Woods, and Y. Zinder. A genetic algorithm for assigning train arrival dates at a maintenance center. In *IFAC-PapersOnLine*, 2019" [70] and "H. Gu and H.C. Lam. A genetic algorithm approach for scheduling trains maintenance under uncertainty. In *Advances in Intelligent Systems and Computing*, 2020" [71], respectively. The research was extended and published in the Journal of Industrial & Management Optimization [72].

1.2 Scheduling of Jobs Sharing multiple Resources

The second problem, discussed in Chapter 4, considers the scheduling of jobs where each job requires several types of resources. This problem shares a few common features with the maintenance planning problem discussed in the previous chapter: (1) there is uncertainty concerning the processing times (durations) of job (maintenance operation); and (2) it is possible to acquire extra units of resources at a cost. However, in contrast to the maintenance planning problem, this problem postulates that the penalty for requiring additional resources is a piece-wise linear penalty function, described in terms of the capacity and an upper bound on the resource expansion. Furthermore, during its processing, each job requires several types of resources. Due to these differences, direct application of existing solution approaches, presented in the previous chapter, cannot suffice.

The considered problem has an objective function that is comprised of two cost components, i.e. the scheduling cost concerning the time periods by which the jobs should be completed and the expansion of resource cost due to requiring additional resources beyond capacity. A feasible schedule is one where all jobs can start and finish within the length of the planning horizon. An optimal schedule is one that minimizes the sum of both scheduling and resource expansion costs.

The considered problem was motivated by the planning of rolling stock maintenance problem discussed in the previous chapter. In addition, the considered problem is also encountered in many business and service environments. For example, in surgery scheduling, a hospital must schedule a set of surgeries over a planning horizon (e.g. a week). Each surgery requires several significant hospital resources, such as operating rooms, surgeons, anesthesiologists, nurses, and equipment. Resources are of limited capacity, but additional quantity for some resource types can be obtained at a cost. For example, one can increase the available labor resource by allowing the existing personnel to work overtime. The duration of each surgery is uncertain. The goal is to generate a surgery schedule that maximizes the utilization of different types of resources and minimizes the operational costs. The tactical berth allocation

is another example of the considered problem. Upon arrival at the container terminal, the ship requires the allocation of a berthing position and different types of resources, such as quay cranes and port personnel. The number of containers that need to be handled during loading and unloading for each ship is uncertain, and therefore, the dwell time of the ship at the port is also uncertain. The port operator would like to satisfy the expected berthing time of ships given the berthing and quay spaces' limitations.

Chapter 4 presents a time-indexed zero-one linear programming formulation. This model uses binary variables that take the value 1 if the job starts in a given time period and 0 otherwise to handle scheduling decisions. This model allows us to use different scheduling costs without changing the problem structure. To solve the model, we develop a Genetic Algorithm enhanced by a local search procedure. The key differentiating feature of our solution method from existing approaches [92] is that we are able to exploit the problem structure and use an efficient method to compute the value of the objective function. The proposed method is fast enough that it can be included within the optimization procedure. We present computational experiments that confirm the effectiveness of our solution approach over the direct application of mathematical programming for small instances. For large instances of the considered problem, we assess the quality of the obtained solutions by means of Sample Average Approximation.

The research in Chapter 4 was submitted to the peer-reviewed journal under the title "A hybrid genetic algorithm for scheduling jobs sharing multiple resources under uncertainty".

1.3 Planning of Grid Operation-based Outage Maintenance

The third problem, discussed in Chapter 5, is a large-scale scheduling problem arising in the maintenance of the power distribution grid. Most of the current grid network maintenance scheduling models postulate that future grid operation is known with certainty when creating

the maintenance schedule. The research in Chapter 5 aims to address this gap in knowledge by proposing mathematical models and solution approaches to deal with the uncertainty in future grid operations.

Over the last 30 years, there has been tremendous growth in global energy consumption due to demographic explosion, strong economic growth, and technological progress. According to the Our World in Data [140], total worldwide electricity consumption has doubled from 11,957 terawatt-hours in 1990 to 25,849 terawatt-hours in 2020. To ensure the reliability of electricity supply for meeting the demand of all connected customers, the overhead power lines must be properly maintained. Due to the extreme hazards involved when working on high-voltage power lines, these lines need to be disconnected from the grid while performing maintenance operations. The disconnections of some lines (planned outages) put extra load on the remaining ones and cause the transmission system to be weakened. The impact of a planned outage on the system reliability can be quantified in terms of risk values. These risk values are input data to the optimization problem considered in Chapter 5.

We use the term *intervention* throughout Chapter 5 to refer to maintenance operation on transmission lines. Owing to the periodic nature of the scheduling, the transmission system operator is able to produce a maintenance schedule a year in advance, deciding when each intervention should commence while taking into consideration the constrained resources. Yet, the operator continue to face various major challenges. The top three challenges for the operator are (1) integration of renewable energies in the network, (2) changes in consumption patterns, and (3) aging infrastructure. As a result of these challenges, the planned outages generated based on the current practices might be infeasible when changes to the grid in the years ahead are uncertain.

Chapter 5 considers the intervention scheduling problem to minimize the average risk (which is expressed as a monetary cost) related to performing the interventions and the total cost for the deviation from the average risk. The problem is formulated as a mixed integer linear program, that uses binary variables to model scheduling decisions. The linearised model uses binary variables to model the quantile function. It also uses binary variables that take the

value 1 if the intervention starts in a given time period and 0 otherwise to handle scheduling decisions. Four solution approaches to the problem are presented. The first approach is a heuristic based on solving a mixed integer linear program, which is developed by approximating the quantile term in the objective function. The second and third approaches are based on the idea of *confidence approach* [95] - a method for solving stochastic optimization problem having quantile of the distribution of the random variables in the objective function. The fourth approach is an Iterated Local Search with self-adaptive perturbation, three local search optimization procedures, and a restart strategy.

The optimization problem, routinely faced by a major electricity transmission system operator, was proposed for the international competition ROADEF/EURO challenge 2020. The competition has attracted the participation of 140 academic researchers and non-academic experts, divided into 72 teams. The competition consists of four separate phases (i.e. Sprint (optional), Qualification, Semi-final, and Final) and lasted over a span of one year and three months from April 1st 2020 to June 29th 2021. The proposed Iterated Local Search with self-adaptive perturbation won the 2nd prize in the final phase of the competition. The research in Chapter 5 was subsequently submitted to the special issue by Journal of Heuristics under the title "Heuristics and meta-heuristic to solve the ROADEF/EURO challenge 2020 maintenance planning problem".

1.4 Contributions

Below is a summary of the contributions of the research presented in this thesis:

- For the rolling stock high-level heavy maintenance planning problem, discussed in Chapter 3:
  - a new nonlinear programming formulation of the considered planning problem, which takes into account the uncertain duration of maintenance and the unique operational constraints imposed by the industry;

- a polynomial-time algorithm for evaluating the objective function of the introduced nonlinear mathematical program for any feasible solution. The algorithm is used within the optimization procedure;
- a new mixed-integer linear programming relaxation that is based on Jensen's Inequality [27] and therefore provides a lower bound to the optimal value of the objective function for the nonlinear mathematical program and hence allows to assess the quality of approximate solutions;
- an Iterated Local Search (ILS), and a Genetic Algorithm (GA) metaheuristics that, in contrast to the previous publications, take into account the uncertain duration of maintenance;
- an amalgamation of Genetic Algorithm for global search and Sample Average Approximation (SAA) method for determining the arrival dates of train-sets when a sequence of train-sets is known;
- a hybrid two-stage optimization procedure that combines Jensen's Inequality based relaxation with either a local search subroutine or the presented ILS subroutine;
- by means of computational experiments on real-world data, it is shown that the hybrid two-stage optimization procedure performed the best, followed by Iterated Local Search metaheuristic, followed by the amalgamation of GA and SAA. The GA metaheuristic was the least successful among the four solution approaches.

- For the problem of scheduling jobs sharing multiple resources, discussed in Chapter 4:
  - a mixed-integer linear programming formulation of the problem, which takes into account the uncertain processing time of jobs;
  - a polynomial-time algorithm for evaluating the objective function of the considered problem for any feasible solution. The algorithm is used within the optimization procedure;
  - a lower bound on the optimal value of the considered problem via SAA and hence allows to assess the quality of approximate solutions;

  - an amalgamation of the Genetic Algorithm and a local search procedure that, in contrast to the previous publications, consider the uncertain processing times of jobs. We refer to this method as the hybrid Genetic Algorithm;
  - through computational experiments, it is shown that:
    * for small problems where only ten jobs have two processing times and the remaining ones have deterministic processing times, the proposed hybrid Genetic Algorithm outperforms the direct application of Integer Programming;
    * for large problems where every job has two processing times, the proposed hybrid Genetic Algorithm produces high-quality solutions in a practically acceptable time, with a confidence interval on the optimality gap of within 2% from the upper bound.

- For the grid operation-based outage maintenance planning problem, discussed in Chapter 5:
  - a new mixed-integer linear programming (MILP) formulation of the maintenance planning problem that uses binary variables to model the quantiles and scheduling decisions;
  - a new MILP that is based on approximating the quantile term in the objective function, and therefore provides approximate solutions to the considered problem;
  - four solution approaches are presented:
    * an approach based on solving the approximation problem by CPLEX and then applying rules to reduce the estimation error for the quantile term in the objective function. We refer to this method as IterUpdate heuristic;
    * two heuristic approaches which utilize the idea of confidence method [95] was proposed for the qualification phase of the ROADEF/EURO challenge 2020;
    * an Iterated Local Search metaheuristic with self-adaptive perturbation and a restart strategy, coupled with a Large Neighborhood Search, was proposed for the semi-final and final phases. The algorithm was implemented in Cython for speedup purposes;

- the solution approaches are compared by means of computational experimentation using benchmark instances provided by the competition;
- the proposed Iterated Local Search metaheuristic with self-adaptive perturbation and a restart strategy, coupled with a Large Neighborhood Search outperformed the solution approaches of 11/13 qualified teams in the final phase.

Chapter 2

Literature Review

For the purpose of giving a self-contained overview of our study, this chapter discusses the research that is most relevant to the considered optimization problems in this thesis. First, this chapter provides, in Section 2.1, a brief introduction of stochastic programming with a special focus on the stochastic programs with recourse and the solution approaches from the literature. The formulations and notations in Section 2.1 regarding stochastic programming have been taken from Birge's book [30] unless otherwise stated. In Section 2.2, an overview of Genetic Algorithm and Iterated Local Search is presented. The literature relevant to the research in Chapters 3, 4, and 5 is discussed in Sections 2.3, 2.4, and 2.5 respectively.

2.1 Stochastic Programming

Uncertainty is inherent in real-world problems and arises in different forms. For example, future demands for a product depend on future market conditions; processing times for job depend on the availability of materials and resources, which are random; and travel times for vehicle depends on unknown weather conditions, among others. Beginning with the seminal works of Beale [23] and Dantzig [44], there has been a growing interest of the scientific community in addressing optimization problems that include in their mathematical formulation

uncertain information. A variety of modeling approaches has been developed, ranging from recourse-based stochastic programming [30], robust stochastic programming [124], probabilistic (chance-constraint) programming [39, 40], fuzzy programming [191], and stochastic dynamic programming [25]. While differing in the details of their implementations, these modeling approaches provide a means for handling the uncertain parameters in optimization problems under uncertainty. A discussion of the advantages and shortcomings of each approach can be found in [147].

In stochastic programming, we assume that probability distributions governing the uncertain parameters are known or can be estimated and incorporate the probabilistic information into the models. Randomness usually enters the model in two ways: through the constraint set or the objective function. This thesis focuses on the latter case where stochasticity is present in the objective function. In the following sections, stochastic programs with recourse, including fixed recourse and simple recourse, are briefly described, and the relevant solution methods are discussed. A detailed explanation on this subject can be found in Birge's book [30], Kall and Wallace [90], Prékopa [137], Shapiro et al. [153], as well as the very informative Stochastic Programming Community Home Page [160].

2.1.1 Stochastic programs with recourse

Stochastic programs with recourse are a class of stochastic programs [44, 23] that hedge against the uncertain future by making decisions based on data available at the time the decisions are made. In these types of models, an initial decision is made before the realization of the uncertain data and recourse actions are needed upon disclosure of the uncertainty. Depending on the nature of the decision-making system, two categories of stochastic programs with recourse can be defined, namely, two-stage and multi-stage [30]. Two-stage stochastic programs with recourse make recourse decisions at only one time in the future, while multi-stage stochastic programs with recourse make recourse decisions at multiple future times. Below, we provide a review of two-stage stochastic linear program with recourse because

the models considered in this thesis are two-stage models. For more details on multi-stage models, we refer the reader to [29, 146, 148, 11] and a recent publication of [53].

A formal formulation of the two-stage linear stochastic programming problems is as follows:

$$\begin{aligned} \min\ & c^T x + \mathscr{Q}(x) \\ \text{subject to}\ & Ax = b, \\ & x \geq 0, \end{aligned} \tag{2.1}$$

where

$$\mathscr{Q}(x) = \mathbb{E}[\mathcal{Q}(x, \omega)] \tag{2.2}$$

and

$$\mathcal{Q}(x, \omega) = \min_y \{q(\omega)^T y(\omega) \mid W(\omega)y(\omega) = h(\omega) - T(\omega)x,\ y(\omega) \geq 0\} \tag{2.3}$$

where ω denote a scenario, x are the first-stage decision variables, $y(\omega)$ are the second-stage decision variables for each ω, and $\mathbb{E}$ is the expectation operator. The $q(\omega)$, $h(\omega)$ and $T(\omega)$ are parameters of the second-stage problem. In the stochastic programming community, the matrix $W(\omega)$ is known as the *recourse matrix* and $T(\omega)$ is known as the *technology matrix*. The objective function is comprised of two parts: the cost of the first-stage decision ($c^T x$) and the expected cost of the second-stage decision ($\mathbb{E}[\mathcal{Q}(x, \omega)]$).

The two-stage stochastic programs with recourse can be further distinguished by the feasible regions and objective values, and they could either be fixed recourse, complete recourse, or simple recourse [30]. Problem (2.1)-(2.3) is said to have fixed recourse if $W(\omega) = W$ for every ω. It is said to have complete recourse if the second-stage problems are always feasible for every solution x that satisfies $Ax = b$. The two-stage stochastic program with simple recourse is the one in which $\mathscr{Q}(x)$ simply involves calculating the linear penalties for the deviations from a target value [55]. As an example, the news vendor problem where a newsvendor must decide how many newspapers (x) to buy at a price c per paper from the publisher every morning, given that demand for the newspaper (ξ) is random. Let q denote the unit selling price and r denote the unit return price for any unsold newspaper. When the uncertain demand is revealed, the profit is merely equal to $q \min(\xi, x) + r \max(x - \xi, 0)$. Other examples

include knapsack problem under weights and capacity uncertainty [59], resource allocation problem with uncertain projects' size [96], employee scheduling with uncertain demand [131], and network design under traveling times uncertainty [101]. The problems considered in Chapters 3 and 4 can be viewed as a two-stage stochastic program with simple recourse.

Stochastic mixed integer program is a class of stochastic program in which the first-stage variables are binary, i.e. $x \in \{0, 1\}$. The stochastic mixed integer program combines the challenges from two types of programs: stochastic program and integer program [102]. The challenge of integer program is having to consider a huge number of discrete variables for problems of practical size. With stochastic program, the challenge comes from evaluating the expected cost of the second-stage decision $\mathscr{Q}(x)$. For stochastic mixed integer program with a continuous distribution, evaluating $\mathscr{Q}(x)$ for a given first-stage decision x may be impossible because it involves a multi-dimensional integral of a complicated function $\mathcal{Q}$, and therefore optimizing over x is even more challenging. For the case with a discrete distribution, evaluation still requires solving a large number of mixed integer programs.

2.1.2 Solution methods

There have been a growing number of studies concerning the two-stage stochastic program. Exact solution procedures such as the L-shaped method [172] and Lagrangian relaxation-based progressive hedging algorithm [142], showed success in solving problems with a finite number of scenarios. When the total number of scenarios is very large, exact solution procedures may become computationally demanding, and so approximation and bounding algorithms [28] and sampling-based algorithms [152] are preferred. The approximation and bounding algorithms calculate and improve the upper and lower bounds on the optimal objective function value as the algorithm progresses, while the sampling-based algorithms approximate (2.2) by using Monte Carlo simulation.

For the case where the first-stage decision variables are binary, one of the earliest proposed methods is the integer L-shaped method [102]. The integer L-shaped method uses a branch-

and-bound algorithm to search the space of the first-stage variables for the globally optimal solution, and the optimality cuts to approximate the second-stage value function $\mathscr{Q}(x)$. The integer L-shaped method has been successfully applied to solve many different stochastic scheduling problems from production scheduling [92], to project scheduling [35], and operating room scheduling [48, 49]. However, this method can be slow in practice due to requiring to solve the master problem which tends to be more difficult as more cuts are added. Strategies to enhance the performance of integer L-shaped method have been proposed in [5] and applied to solve the traveling salesman problem with drone [173]. Other techniques of using cutting planes to strengthen the Benders model have also been studied in the literature. For example, disjunctive programming techniques to convexify the second-stage problems [150, 151], Fenchel cuts to strengthen the subproblem relaxations [128], as well as Gomory cuts [62] and scaled cuts [171] to develop finitely convergent algorithm.

2.1.3 Sample Average Approximation

Sample Average Approximation (SAA) approach [96] belongs to the family of sampling-based algorithms, where the problem (2.1)-(2.3) is replaced with its sampling approximation. The history of SAA dates back to the 1990s when it was first known by the name "stochastic counterpart method" [144], and then "sample-path optimization method," [141, 136]. SAA is useful in practice, particularly when continuous distributions are present. Given $\omega_1, \omega_2, ..., \omega_N$ of N sample scenarios of the random vector, the Sample Average Approximation problem can be formulated as follows:

$$\begin{aligned} \min z_N &= c^T x + \frac{1}{N}\sum_{i=1}^{N} \mathcal{Q}(x, \omega_i) \\ \text{subject to } & Ax = b, \\ & x \geq 0, \\ & T(\omega_i)x + W(\omega_i)y(\omega_i) = h(\omega_i), \quad i = 1, ..., N \\ & y(\omega_i) \geq 0, \quad i = 1, ..., N \end{aligned} \tag{2.4}$$

Asymptotic results regarding the consistency and validity of SAA estimators have been extensively studied. For a recent survey see [153]. Approaches to solving (2.4) generally fall into three categories: (i) solving a single SAA problem with a fixed (possibly large) sample size [155, 156, 96]; (ii) solving a single SAA problem of a desired sample size N and then assessing the quality of the approximate solution [109]; and (iii) sequential sampling procedures [21, 22]. Many methods have been developed for assessing the quality of SAA solutions, for example, those that estimate optimality gaps [109, 19, 20, 18], test optimality conditions [78, 154], and check the stability of solutions [91].

2.2 Metaheuristic

A metaheuristic is a high-level problem-independent methodology that performs a search for solving a large class of difficult optimization problems. The term was introduced by Glover [65] and formed from the Greek word meta, which means *beyond*, and heuristic, which is related to the Greek word heuriskein meaning *to find*. In the first edition of the *Handbook of Metaheuristics* [66], metaheuristics are defined as "solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space".

Based on how solutions are manipulated, metaheuristics can be categorized into three groups [67]: (1) those that maintain solutions in a population and combine those solutions into new ones, known as population-based metaheuristics, (2) those that maintain and iteratively improve a single solution, known as local search metaheuristics, and (3) those that successively build solutions from their constituting elements, known as constructive metaheuristics. These classes of metaheuristics, while are similar in that they utilize *exploration* to move to the other regions of the search space and *exploitation* to find a local optimum, each class differs from one another in the ways in which the balance between *exploration* and *exploitation* are maintained during the search.

Although metaheuristics fail to guarantee the optimality of the obtained solutions, they are capable of finding sufficiently good solutions in an acceptable time for computationally hard problems. In addition, unlike exact optimization algorithms, metaheuristics do not require a mathematical formulation of the optimization problem. In the past two decades, metaheuristics have been successfully applied to many optimization problems with uncertainty ranging from production scheduling [159, 87] to project scheduling [13, 185] and vehicle routing [119, 129]. Two well-known metaheuristics, including Genetic Algorithm [80] (a population-based metaheuristic) and Iterated Local Search [107] (a local search metaheuristic) are briefly described in the following sections.

2.2.1 Genetic Algorithm

Genetic Algorithm (GA) metaheuristic is a population-based search methodology, inspired from the Darwinian theory of evolution. GA was first introduced by Holland [80] in the 1970s to simulate the biological evolution of adaptive natural systems. Later, the idea of using GA for solving optimization problems was introduced in De Jong [46]. A population of solutions (individuals) for a given problem evolves towards a high-quality solution through a series of recombination and mutation operators. The general structure of a GA includes evaluation, parent selection, recombination (crossover), mutation, and replacement. The pseudocode given in Algorithm 1 outlines the basic Genetic Algorithm.

Algorithm 1 Pseudocode of the Genetic Algorithm method

```
1: GenerateInitialPopulation(); // generate initial population of solutions
2: repeat
3:    Evaluate(); // calculate fitness of each solution in the population
4:    SelectParents(); // select solutions from the population to breed
5:    Crossover(); // apply crossover operator with a given probability
6:    Mutation(); // apply mutation operator with a given probability
7:    Replacement(); // generate new population of solutions
8: until TerminationCriterionSatisfied();
9: return Best generated solution;
```

Starting from the initial population of solutions, GA repeats those phases until a stopping

criterion has been reached. Each cycle is referred to as a generation. First, one of the most important features of a GA is the representation of a solution because suitable representations can enhance the efficiency of a search. Depending on the optimization problem and characteristic of a solution, different types of solution representation exist, for example, binary coding, permutation encoding, and real-value encoding. Second, the evaluation process is necessary to assign a fitness value to each individual, thereby ranking its performance. For deterministic problems, it is common to assign a fitness value which is computed exactly as, or directly from, the objective function. For problems with uncertainty, the fitness value is obtained through simulation. Third, parent individuals will be selected from the population, and they are recombined (via crossover operator) to produce offspring individuals, which are perturbed (via mutation operator) to ensure genetic diversity in the population. The two most commonly used selection methods are *Roulette-wheel* and *Tournament* operators [63]. Finally, the purpose of the replacement operator is to decide which individuals may perish and which ones will survive to the next generation.

As with most other metaheuristics, there are several variants of GA, one of which combines Genetic Algorithm with local search. This idea has appeared in the literature under different names: in Moscato [120], Moscato and Norman [122], Neri et al. [126], Cotta et al. [42], Moscato and Cotta [121], Neri and Cotta [125], it is called the memetic algorithm, whereas in Ulder et al. [167], Kolen and Pesch [97] it is called the genetic local search. The term "hybrid genetic algorithm" appears to have been used originally to refer to the combination of GA with other classes of metaheuristics [67]. But it is now widely used to imply the global-local complementary view of genetic hybrids by the research community, for example, El-Mihoub et al. [56], Gonçalves et al. [68], just to name a few. The local search procedure, sometimes called "education procedure" in GA, is often applied after the mutation phase on the new individuals in the population. It has been known that the performance of GA is better off in terms of both solution quality and efficiency when incorporating a local search method within its framework. These benefits result from the fact that local search can reduce the likelihood of premature convergence and the time needed to reach the global optimum [56]. Hybrid genetic algorithms have been successfully used in many different problems and

were reported to be more efficient than GAs in production scheduling [166, 174], vehicle routing [83], and engineering design problems [183, 184], among others.

2.2.2 Iterated Local Search

Iterated Local Search (ILS) metaheuristic, as its name suggests, is a method that embeds a local search heuristic within an iterative process building a sequence of solutions. This simple idea has a long history and appeared in the literature under different names: iterated descent [16, 17], large-step Markov chains [113], iterated Lin-Kernighan [88], and chained local optimization [112]. A single solution for a given optimization problem is perturbed and improved to obtain a high-quality solution at the end. Perturbation mechanism, local search techniques, and acceptance criterion are the main components of an ILS. The pseudocode given in Algorithm 2 outlines an Iterated Local Search algorithm.

Algorithm 2 Pseudocode of the Iterated Local Search method

1: Let S^* represent the best solution;
2: $S_0 \leftarrow$ `GenerateInitialSolution()`; // `generate initial solution`
3: $S^* \leftarrow$ `LocalSearch`(S_0); // `find local optimal solution`
4: **repeat**
5: $\quad S^{'} \leftarrow$ `Perturbation`$(S^*, history)$;
6: $\quad S^{*'} \leftarrow$ `LocalSearch`$(S^{'})$;
7: $\quad S^* \leftarrow$ `AcceptanceCriterion`$(S^*, S^{*'}, history)$;
8: **until** `TerminationCriterionSatisfied`();
9: **return** S^*;

Upon generating an initial solution, a local search is employed to obtain an improved solution. Then, ILS applies a perturbation to the current local minimum solution, leading to a new solution. This is followed by the local search, which is applied to the perturbed solution, leading to an improved solution. An acceptance criterion decides whether the newly found local minimum solution or the previous solution should be kept for continuing the process. The repetition of applications of perturbation and local search methods terminates when a stopping criterion is satisfied.

The phases are interrelated, requiring trade-offs between them in order to reach high performance with an ILS algorithm. To see that, note that an effective escape from local optima is dependent on the moves used in perturbation and the changes used in local search, which in turns depends on the problem and often on the specific problem instances, so how strong the perturbations should be is usually difficult to determine [107]. Setting the size of the perturbation to be aggressively large maximizes the exploration ability of the algorithm because it is unlikely for local search to undo the perturbations. However, the quality and the structure of the current candidate solution may be destroyed, potentially causing the algorithm to behave as a random restart algorithm, resulting in a much worse local optimum. To maximize exploitation, the size of the perturbation should be set conservatively small to enable repeated search in the current neighborhood of the search space. Adaptive perturbation modifies the size of the perturbation (i.e., perturbation strength) and adapts it during the search [107]. Many adaptive schemes have been suggested, for example, those that follow the ideas of reactive search [15, 14] and of variable neighborhood search [118, 76]. The adaptive perturbation strategy was used in various ILS algorithms across several problem domains, like production scheduling [52], workforce scheduling [179], and knapsack [9] problems.

2.3 Planning of Rolling Stock Maintenance

The body of literature on planning the rolling stock maintenance falls into two broad categories: (i) planning the rolling stock low-level maintenance subject to the trains timetable, and (ii) planning the rolling stock high-level maintenance for a long planning horizon. The first category considers only low-level maintenance, such as daily and monthly inspections, which is often combined with the decisions on the rolling stock utilization. Accordingly, the low-level maintenance is often considered as constraints in the planning process of rolling stock utilization (see for example, Zhong et al. [190], Lai et al. [100], Giacco et al. [64]).

The second category is concerned with the high-level maintenance which has a long cycle time. Among a few publications relevant to the problem considered in Chapter 3, Sriskandarajah

et al. [161] is concerned with maintenance scheduling at the Hong Kong Mass Transit Railway Corporation. The authors of [161] postulate that the duration of maintenance is given. This simplification allows them to approach the minimisation of the earliness and tardiness with respect to the completion of maintenance as a deterministic scheduling problem. The authors of [161] also assume that the given permissible number of trains which can dwell at the maintenance center simultaneously can not be violated, which may not be possible to ensure when the duration of maintenance is uncertain. The resultant deterministic scheduling problem is solved by a genetic algorithm. The authors reported that the proposed approach produced near optimal solutions for randomly generated instances with linear earliness and tardiness objective.

As in [161], Doganay and Bohlin [50] assumes that the duration of maintenance is known and that the limit on the number of trains that can dwell at the maintenance center simultaneously cannot be violated. In addition, Doganay and Bohlin [50] is concerned with planning under the condition that the maintenance operations must commence prior to their due dates. The planning problem was formulated as a mixed integer linear programming model and was solved using IBM ILOG CPLEX 11.2. The objective function is a weighted sum of the maintenance cost, the cost of shunting activities, the cost related to the spare parts, and the penalty for early maintenance. The authors reported that the inclusion of the cost related to the spare parts positively affected the quality of the plan.

The publication [105] is concerned with planning the heavy maintenance of the electric multiple units at the Shanghai Railway Bureau. As in [161] and [50], it is assumed that the duration of maintenance is known and that the limit on the number of trains that can dwell at the maintenance center simultaneously cannot be violated. In contrast to the above studies, Lin et al. [105] consider the case where each train has a time window when its maintenance should commence, and this time window cannot be violated. The problem is formulated as an integer linear program with the objective of minimizing the total penalty for early maintenance. It was reported that small instances can be solved exactly. For large instances, the authors propose a simulated annealing algorithm and report solving instances with up to 124

train-sets and a planning horizon of 607 days.

All three aforementioned papers model the rolling stock maintenance planning problem with deterministic approach, i.e. the input parameters to the optimization problems are known with certainty. By describing a real system as deterministic models, many characteristics of real-world problems are neglected, which can affect the quality of the maintenance plan. The study in chapter 3 employs a stochastic approach to incorporate the presence of uncertainty in the problem formulation. That is, we assume the time a train-set spends at the maintenance center is a random variable with known probability distribution. Applying stochastic optimization to the maintenance planning of rolling stock is fairly new, for example, [123, 116, 45, 187]. Mountakis [123] considers the train maintenance scheduling problem, in which the durations of maintenance activities are random. The authors present a heuristic procedure in which feasible schedules are obtained by means of solution enumerations in combination with the statistical static timing analysis (SSTA) technique for evaluating the objective function. Numerical results using real data from a maintenance facility in Netherlands show that the proposed heuristic is capable of solving large-scale problem instances with over 600 activities and with durations of maintenance activities exhibiting high variability. The authors also found out that the use of SSTA to evaluate stochastic-based objective function provide significant improvement in computation time as compared to the conventional Monte carlo simulation approach, and therefore increases the effectiveness of the solution enumeration scheme. The effect is more pronounced for larger variance.

More recently, Mira et al. [116] discuss the problem of scheduling rolling stock maintenance tasks for a given period, ranging from 1 day to 1 week, for a rail operating company in Portugal. The authors present a two-step decision support system in which the schedule of maintenance activities to train units is obtained in the first step. By ignoring uncertainty, the resulting scheduling problem, which takes into account several practical considerations such as the timetable of train services, can be formulated as an integer linear programming model and was solved using FICO® Xpress Solver. Then, in the second step, a reliability analysis is conducted to access the impact of the uncertain maintenance durations on the

feasibility of the obtained maintenance schedule.

2.4 Scheduling Jobs Sharing Multiple Resources

Scheduling problems where jobs require multiple types of resources exist in a large number of domains, e.g., operating room scheduling (see, e.g., Durán et al. [54]), workforce planning (see, e.g., Ho and Leung [79]), project scheduling (see, e.g., Valls et al. [170]), train fleet maintenance planning (see, e.g., Doganay and Bohlin [50]), outage maintenance planning in power system (see, e.g., Pandžić et al. [130]), to name but a few. For instance, in operating room scheduling, jobs correspond to different surgeries and resources correspond to doctors, nurses, and operating rooms. In workforce planning, jobs correspond to different services and resources correspond to workers with different skill sets. In project management, jobs correspond to different projects, and resources correspond to manpower, materials, and equipment. In maintenance management, jobs correspond to different maintenance operations and resources correspond to manpower, materials, and spare parts. In power system maintenance planning, jobs correspond to different maintenance jobs and resources correspond to maintenance workers and materials. More recently, one can easily find a large collection of studies that incorporate uncertainty into these scheduling problems. For example, operating room scheduling with uncertainty in duration of surgical procedures [48, 49, 8]; workforce planning with uncertainty in demand [77, 10]; project scheduling with uncertainty in activity durations [38, 35, 41]; rolling stock maintenance scheduling with uncertainty in maintenance durations [116]; power system maintenance planning with uncertainty coming from renewable energy [99, 84]. The problem addressed in Chapter 4 is inspired by the above-mentioned applications.

The considered stochastic optimization problem is closely related to the Stochastic Resource Constrained Project Scheduling Problem (SRCPSP) which can be described as follows. Given a set of activities $V = \{0, ..., n+1\}$, where each activity has to be processed in order to complete the project. Activities 0 and $n+1$ are two dummy activities representing the

project start and end, respectively. A set of renewable resources $K = \{1, ..., m\}$, each with finite capacity $R_k,\ k \in K$ and non-negative resource consumption $r_{jk},\ j \in J,\ k \in K$. A set of precedence constraints between different activities within the project E such that $(i, j) \in E$ means that activity j has to start after the completion of activity i. The goal is to determine a schedule for performing the activities, by minimizing the finish time of the last activity, while satisfying any precedence relationships among the activities and for resource availability [162]. The problem studied in Chapter 4 is distinct from SRCPSP in several ways. Our problem does not consider precedence constraints between different jobs since the jobs are assumed to be "projects" independent of each other except for the shared resources. Moreover, our problem allows resource capacity to be exceeded, and there is upper bound on the amount of temporary resource expansion. To deal with this expansion, our problem adds penalty for each extra unit of required resource beyond the capacity.

The deterministic RCPSP is known to be NP-hard and their solution approach is computationally expensive [6]. For this reason, most solution approaches are based on metaheuristics [47, 170, 85, 89]. In particulars, Valls et al. [170] propose a hybrid Genetic Algorithm, in which a local search is introduced to enhance the solution quality. In Chapter 4 we also consider a hybrid Genetic Algorithm which combines genetic algorithm and a local search procedure, but focus on problems with uncertain processing times. Extensions of the RCPSP, involving the minimization of the expected makespan of a project with stochastic activity durations, have been investigated within the stochastic project scheduling literature. Several models and solution approaches have been suggested for stochastic scheduling and typically fit into one of two categories: (1) reactive or (2) proactive scheduling. Reactive scheduling creates a policy to update a solution on an online basis, that determines which activities are to be started at a certain decision time t given prior knowledge up to $t-1$ (see, for example, Rostami et al. [143], Ballestín [12]). Our line of research falls into the second category of scheduling strategy - proactive scheduling. In this approach, some knowledge of the uncertain activities durations is incorporated in the decision-making stage, with the aim to generate a robust baseline for the project. This important research stream has received outstanding attention in the last years, and various approaches including exact (mathematical programming

based) [35, 38], heuristic [34, 7], and metaheuristics [185, 186] have been proposed.

The problem under study in chapter 4 is closely related to [92], in which the problem is formulated as a two-stage stochastic model with complete recourse. The first-stage decision is to determine the starting times for jobs based on knowledge about the distribution of the uncertain processing times. The second-stage recourse cost describes a penalty for the amount in which the resource capacity is exceeded. This formulation allows for the application of L-shaped method. To solve large instances, the authors of [92] develop a sequential sampling procedure. In contrast to [92], our study presents an alternative approach based on Genetic Algorithm metaheuristic. Furthermore, we exploit the problem structure and uses an efficient method to compute the value of the objective function. The proposed method is fast enough that it can be included within the optimization procedure.

One of the key issues when applying Genetic Algorithm to stochastic optimization problems is the computation of the objective function, which involves expected values. The majority of work in the literature use simulation methods to approximate the objective function. One strategy is to evaluate all solutions in all generations of GA with a small number of scenarios (see, for example, Li and Demeulemeester [104], Gu et al. [75], and Wang et al. [175]). However, it is possible that the final solution returned by GA as the best solution is of low-quality with respect to the true objective value [178]. This issue is addressed in [181] and [82], where a set of good solutions found by GA is re-evaluated with a sample of 10^5 scenarios to accurately select the best solution. We also propose a solution approach based on Genetic Algorithm, but in contrast to the above-mentioned studies, we can compute the objective function. So, at each generation, our GA can precisely determine whether a solution is better than another one.

The proposed approach is different from the Genetic Algorithm in our previous studies (see, Gu et al. [70], Gu and Lam [71]) in several ways. First, Gu et al. [70] and Gu and Lam [71] use only the standard Genetic Algorithm. As the application of local search methods can improve solutions and speed up the convergence process [168], the proposed solution method incorporates local search within the GA procedure. Second, the proposed approach employs

SAA for assessing the quality of GA solutions. Third, Gu et al. [70] and Gu and Lam [71] use the random key encoding scheme, which requires a procedure to transform the chromosome into a solution. We instead use a solution-based representation, wherein a solution is directly represented by a chromosome.

The idea to incorporate local search method within a Genetic Algorithm can be traced back at least to Reeves [138] and has been successfully applied to the permutation flowshop scheduling problem [166], parallel machines scheduling problem [159], and job shop scheduling problem [174], among others. Most studies found in the literature design and use the hybrid Genetic Algorithm for deterministic problems. Under the assumption that parameters are known constants, it is easy to incorporate the local search method within the genetic algorithm framework as the evaluation of neighbor solutions does not require much time. However, for optimization problems where random parameters are present in the objective function, it is expensive to compute the objective function value for a solution. For this reason, studies that use genetic algorithm enhanced by local search for solving stochastic optimization problems is limited. The local search used in this study is designed with the aim that the neighbor solutions can be evaluated quickly, making it efficient enough to be used in the GA framework.

2.5 Planning of Grid Operation-based Outage Maintenance

Maintenance management is an important function as energy industry organizations are making an effort to ensure the reliability of the electric power system for meeting demand, and is therefore a focus of a large number of scheduling studies [60]. Most of these studies deal with maintenance scheduling of generation units (see, for example [115, 149, 139]), while some consider finding an optimal outage schedules for both the generation units and the transmission lines (see, for example [176, 1, 61]). For recent literature review on generation units maintenance scheduling, see [93]. A thorough review of maintenance scheduling in the

electricity industry is provided in [60]. In this section, we focus on articles that tackle only the transmission-line maintenance scheduling problem.

In [130], the authors present a bi-level outage scheduling model, where the upper-level objective is to maximize the unused transmission capacity while the lower-level objective is to minimize the impact on the functioning of the electricity market. Using equilibrium constraints, the problem is recast as a mixed integer-linear program and solved with CPLEX. This approach schedules considered maintenance during time periods in which its effect on transmission system adequacy is the least. [114] develops a short-term transmission maintenance scheduling model that minimizes the transmission line maintenance cost while satisfying hourly line maintenance constraints, line reservations and system reliability. Using Benders decomposition, the authors decompose the transmission maintenance problem into a maintenance master problem and two types of sub-problems which include transmission sub-problems and voltage sub-problems. [108] formulates the short-term transmission maintenance scheduling problem as a mixed-integer nonlinear program that aims to minimize the maintenance cost and the expected cost of lost load. The model considers failures of transmission components and system reliability as constraints. More recently, [3] proposes a maintenance scheduling optimization model that minimizes the equipment unavailability and avoids simultaneous disconnection of equipment that generates insecure conditions for the operation of the network. The uncertainty in both demand and wind-power generation for transmission line maintenance in long-term horizon is considered in [99]. They develop a two-stage stochastic program to minimize the total expected maintenance cost and cost of lost load of an outage schedule for the transmission lines, under different demand and wind scenarios. All five above-mentioned papers address the transmission-line maintenance scheduling problem with the objective of minimizing maintenance costs subject to achieving a certain required level of reliability, while our study aims at achieving the best level of reliability subject to constraints on resources.

Models for scheduling planned outages for transmission lines that consider the impact of a given outage schedule on maintenance costs and system reliability are presented in [43]

and [165]. In particulars, the authors of [43] and [165] develop a machine learning tool for predicting power system operating conditions during the maintenance of grid components. The supervised learning model helps to identify the time periods during which a maintenance outage can be safely accommodated. The approach of [43] and [165] differs from our approach in that they use machine learning proxy for contingency analyses, while we focus on identifying the best outage schedule, considering contingency analyses as a preliminary stage.

In [86], the authors present a maintenance selection and scheduling approach that considers the long-term risk caused by equipment failure and outage consequence in term of overload and voltage security. Similarly, [2] describes a mixed-integer linear formulation for the long-term maintenance scheduling of distribution overhead lines based on the risk of equipment failure and its consequences on network reliability. More recently, [111] proposes an outage scheduling model over mid-term horizon that minimizes the overall risk of carrying out the maintenance over the set of scenarios of future operating conditions. As in our study, manpower constraints are hard constraints, which must not be violated for an outage schedule to be considered feasible. To solve the problem, the authors design a greedy algorithm that schedules outages one by one, starting from the one with the maximal (negative) impact on system operation. The impact of an outage on system operation is evaluated by Monte Carlo simulations. The proposed approach is able to optimally solve a case study with five interventions and a scheduling horizon of 182 days. In contrast to [111], our study focus on problems with up to 528 interventions and a planning horizon of 300 days.

Chapter 3

Planning of Rolling Stock Maintenance

A railway network is an indispensable part of the public transportation system in many major cities around the world. In order to provide a safe and reliable service, a fleet of passenger trains must undergo regular maintenance. These maintenance operations are lengthy procedures, which are planned for one year or a longer period. The planning specifies the dates of trains' arrival at the maintenance center and should take into account the uncertain duration of maintenance operations, the periods of validity of the previous maintenance, the desired number of trains in service, and the capacity of the maintenance center. This chapter presents a nonlinear programming formulation of the considered problem and several optimization procedures which were compared by computational experiments using real world data. The results of these experiments indicate that the presented approach is capable to be used in real world planning process.

3.1 Introduction

Passenger trains provide one of the major means of public transport in many cities around the world. For example, the suburban passenger rail network in Sydney, Australia connects central Sydney with northern, southern, western, and eastern suburbs with 174 stations.

The network spans over 813 kilometers of track and delivered about 359.2 million passenger journeys in 2017/18 [163].

There is a risk of sudden breakdowns or even a derailment if the trains (rolling stock) do not undergo maintenance regularly. There are several mandatory levels of maintenance that differ from each other by their scope and periodicity. Readers are referred to Lai et al. [100] for a detailed description of the various maintenance levels. This chapter focuses on the high-level heavy maintenance which is the most involved and time consuming procedure and is performed at a specialized maintenance center. The rolling stock arrives at the maintenance center in groups. Each group is comprised of several cars coupled together and is referred to as a set or a train-set [100]. All cars in a train-set undergo maintenance in the maintenance center simultaneously.

The reliable functioning of a passenger transportation system is not possible without a plan, specifying the availability of the rolling stock. The dates when the train-sets should be withdrawn from service and sent to the maintenance center are crucial for the rolling stock operator as well as for the planning at the maintenance center. Given that the heavy maintenance of a train-set is a long process, both, rolling stock operator and maintenance center, need to specify at least for a year when the heavy maintenance of each train-set should commence [105, 161]. Furthermore, often the transportation of passengers and the rolling stock heavy maintenance are carried out by two separate organizations which operate on the basis of a long term contract that specifies for each train-set the precise date of the commencement of its maintenance.

This chapter is concerned with the development of the plan that specifies the dates when the train-sets should arrive at the maintenance center. The presented optimization procedures take into account the specifics of the heavy maintenance procedure, including its uncertain duration and the restriction on the period between consecutive arrivals of the train-sets at the maintenance center, as well as the information provided by the rolling stock operator, including the engineering restrictions on the permissible period between heavy maintenance procedures and the demand for transportation.

The heavy maintenance has a validity period after which another heavy maintenance procedure must take place according to the engineering restrictions. Consequently, the rolling stock operator determines for every train-set a time window within which the maintenance of this train-set should commence (see, for example, Lin et al. [105]). The length of such time window depends on the practice of the rolling stock operator and in some cases can be zero (see, for example, Sriskandarajah et al. [161]). Although it is equally possible to start maintenance at any point in time within the corresponding time window, for the purpose of the optimization procedures below, the middle of a time window will be considered as a preferred date and the time window will be viewed as a specification of the permissible deviation from this preferred date.

The dates when the train-sets should arrive at the maintenance center, specified by the mentioned above plan, may violate the time windows of the train-sets which are dictated only by the practice of the rolling stock operator and do not take into account the capacity of the maintenance center, the uncertain duration of maintenance, and the demand for transportation. Such violation is highly undesirable which often is modeled by introducing a penalty for the violation of the time windows.

A train-set that arrives at the maintenance center before its time window is referred to as "early", while a train-set that arrives after its time window is referred to as "tardy". If despite the penalties for the violation of the time windows the plan still contains some early and tardy train-sets, this situation is resolved by the consultations between the rolling stock operator and the maintenance center. Similar to the majority of publications on this topic, the consultation phase is beyond the scope of this study.

Upon arriving at the maintenance center, a train-set is shunted to the first operation line, where thorough inspections and replacement of some components and parts such as air-conditioning units are performed. A train-set can arrive only if the first operation line is not occupied by the previous train-set. Normally, there are several types of trains and each type may require different time at the first operation line.

After the first operation line, the train-set undergoes various maintenance operations such as bogies replacement, system testing, refurbishment, etc. In order to perform these operations, the train-set has to be shunted between several lines. The duration of each operation depends on the condition of the train-set; the availability and composition of the workforce; the availability of spare parts; and many other factors. Consequently, the dwelling time of a train-set at the maintenance center (also referred to as a cycle time) is uncertain at the time of planning.

On the arrival at the maintenance center, a train-set is completely withdrawn from service and must stay at the maintenance center for at least one month. This long cycle time directly impacts the number of train-sets available in active service. Therefore, as part of the input data for the heavy maintenance planning, a permissible number of train-sets that can be out of service simultaneously are specified for each type of train-sets. Furthermore, the capacity of the maintenance center also imposes the restriction on the number of train-sets which can undergo maintenance simultaneously. Any violation of all these restrictions causes serious problems and therefore must be taken into consideration at the planning stage.

The goal of the planning process is to determine an arrival plan, specifying the arrival dates for all train-sets. The discussion above suggests that it is reasonable to have the objective function as a weighted sum of two components: the total penalty for the deviation (earliness and tardiness) from the arrival time windows, and the expected total penalty for violating the center capacity as well as for violating the permissible number of out-of-service train-sets of all types.

The remainder of this chapter is organized as follows. In Section 3.2, we present a nonlinear mathematical programming formulation of the considered problem, an efficient algorithm for evaluation of the objective function, and a Jensen's Inequality based mixed integer linear programming relaxation. Section 3.3 presents the first solution approach, which is based on Genetic Algorithm. Section 3.4 presents the second approach, which is an amalgamation of Genetic Algorithm and Integer Programming. In Section 3.5, we present an alternative mixed integer linear program, local search and iterated local search subroutines, and a fast method

for the neighborhood evaluation. In Section 3.6, we discuss the results of computational experiments that use real-world data provided by a major maintenance center in Australia. The conclusion can be found in Section 3.7.

3.2 Mathematical Programming Formulation

This section presents a Nonlinear Integer Programming Model (NIPM) which is a formulation of the original problem and a Mixed Integer Programming Model (MIPM) that provides a lower bound for NIPM. The advantage of the lower bound relaxed integer linear programming model MIPM is that feasible solutions are significantly easier to find. Therefore, MIPM is used to provide a high-quality starting solution for the proposed hybrid two-stage optimization procedure.

The considered planning problem can be stated as follows. A set $N = \{1, \cdots, n\}$ of train-sets is to undergo maintenance at a maintenance center. The planning period is T days which are numbered from 0 to $T-1$. An arrival plan specifies for each train-set $j \in N$ the day s_j of its arrival at the maintenance center. No two train-sets can arrive on the same day. Furthermore, the train-sets are of m different types and, for each type k, there exists a restriction when the next train-set can arrive after the arrival of a train-set of type k. This restriction is given by the number of days τ_k. If a train-set j of type k arrives on day s_j, then, regardless of the type of the next train-set, it can arrive only on the day $s_j + \tau_k$ or later.

It is convenient to consider the partition F^1, ..., F^m of N, where each F^k is comprised of all train-sets of the same type k. Observe that arrival days that satisfy the restriction on the time between two consecutive arrivals exist if and only if

$$\sum_{1 \le k \le m} |F^k| \tau_k - \max_{1 \le k \le m} \tau_k \le T - 1.$$

In what follows, it is assumed that this inequality holds.

For each $j \in N$, the time that a train-set j spends at the maintenance center (its cycle time) is a discrete random variable D_j which assumes integer values. All these random variables are independently distributed and, for each $1 \leq k \leq m$, the cycle times of all train-sets in F^k are identically distributed between a_k - the minimal possible duration of a cycle for a train-set of type k, and b_k - the maximal possible duration of a cycle for a train-set of type k. For each $1 \leq k \leq m$, $\tau_k < a_k$.

Each train-set j has the associated time window $[\theta_j - \Delta, \theta_j + \Delta]$, where θ_j is the preferred day of the commencement of maintenance, i.e. the preferred day of the arrival at the maintenance center, and Δ is the permissible deviation from this preferred day of arrival. The penalty for the violation of time window is defined by the function:

$$g_j(s_j) = \begin{cases} \lambda_1(\theta_j - s_j)^2 & \text{if } s_j < \theta_j - \Delta \\ \lambda_2(\theta_j - s_j)^2 & \text{if } s_j > \theta_j + \Delta \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$. For any arrival plan $\sigma = (s_1, \cdots, s_n)$, the total penalty for the violation of time windows will be denoted by $G_1(\sigma) = G_1(s_1, \cdots, s_n)$:

$$G_1(\sigma) = G_1(s_1, \cdots, s_n) = \sum_{j \in N} g_j(s_j). \tag{3.2}$$

For any arrival plan σ, any $1 \leq k \leq m$, and any day t, denote by $W_t^k(\sigma)$ the number of train-sets in F^k that are at the maintenance center on day t. Since all cycle times are random variables, $W_t^k(\sigma)$ is a random variable. If $W_t^k(\sigma)$ exceeds the given limit C_{kt}, this attracts the penalty $\delta_{kt}(W_t^k(\sigma) - C_{kt})$, where $\delta_{kt} > 0$. The total number of train-sets at the maintenance center on day t is

$$W_t(\sigma) = \sum_{k=1}^{m} W_t^k(\sigma).$$

If this number exceeds the given limit C_t, this attracts the penalty $\delta_t(W_t(\sigma) - C_t)$, where $\delta_t > 0$. Let σ be an arbitrary arrival plan and $G_2(\sigma)$ be the mathematical expectation of the

total penalty for the violation of the limits on the number of train-sets that can dwell at the maintenance center simultaneously. Then

$$G_2(\sigma) = \sum_{t=0}^{T-1} \Big(\mathbb{E}\Big[\max\Big\{0, \delta_t\Big(W_t(\sigma) - C_t\Big)\Big\}\Big] + \sum_{k=1}^{m} \mathbb{E}\Big[\max\Big\{0, \delta_{kt}\Big(W_t^k(\sigma) - C_{kt}\Big)\Big\}\Big]\Big), \quad (3.3)$$

where $\mathbb{E}[\cdot]$ is the mathematical expectation. The goal is to minimize

$$G(\sigma) = \alpha G_1(\sigma) + \beta G_2(\sigma) \quad (3.4)$$

where α and β are two positive weights.

We summarise the notation for this chapter as follows: **Sets:**

N: set of train-sets, indexed by j;

F^k: set of train-sets of type k, indexed by j;

E: set of arcs representing the precedence relations between any two train-sets;

Parameters:

T: number of days;

m: number of train-sets' types;

Γ: number of scenarios;

τ_k: minimum days requirement when the next train-set can arrive after the arrival of a train-set of type k;

D_j: cycle time of train-set $j \in N$;

D_j^γ: realization of cycle time of train-set $j \in N$ in scenario γ;

a_k: minimal possible duration of a cycle for a train-set of type k;

b_k: maximal possible duration of a cycle for a train-set of type k;

θ_j: preferred day of the commencement of maintenance of train-set $j \in N$;

Δ: permissible deviation from the preferred day of the commencement of maintenance;

λ_1: earliness cost factor;

λ_2: tardiness cost factor;

C_t: limit on the number of train-sets that can dwell at the maintenance center on day t;

C_{kt}: limit on the number of train-sets of type k that can dwell at the maintenance center on day t;

δ_t: daily penalty per train-set for violating C_t;

δ_{kt}: daily penalty per train-set of type k for violating C_{kt};

α: weight reflecting the relative importance of the first component of the objective function;

β: weight reflecting the relative importance of the second component of the objective function;

Decision Variables:

s_j: arrival day of train-set $j \in N$;

$W_t(\sigma)$: total number of train-sets at the maintenance center on day t with respect to arrival plan σ;

$W_t^k(\sigma)$: total number of train-sets of type k at the maintenance center on day t with respect to arrival plan σ;

w_t: additional train-sets beyond that limit C_t on day t;

w_t^k: additional train-sets of type k beyond the limit C_{kt} on day t;

w_t^γ: additional train-sets beyond that limit C_t on day t under scenario γ;

$w_t^{k,\gamma}$: additional train-sets of type k beyond the limit C_{kt} on day t under scenario γ;

$x_{jt} \in \{0, 1\}$: 1 if train-set $j \in N$ arrives on day t, or 0 otherwise.

3.2.1 Nonlinear Integer Programming Formulation

The heavy maintenance planning problem can be formulated as follows.

$$\text{(NIPM)} \quad Z_{NIPM} = \min \alpha G_1(s_1, \cdots, s_n) + \beta G_2(s_1, \cdots, s_n) \tag{3.5}$$

subject to

$$\sum_{t=0}^{T-1} x_{jt} = 1, \quad \forall j \in N \tag{3.6}$$

$$\sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=\max(0,t-\tau_k+1)}^{t} x_{js} \leq 1, \quad \forall t \in \{0, ..., T-1\} \tag{3.7}$$

$$s_j = \sum_{t=0}^{T-1} t x_{jt}, \quad \forall j \in N \tag{3.8}$$

(3.1), (3.2), (3.3)

$$x_{jt} \in \{0, 1\}, \quad \forall j \in N, \quad \forall t \in \{0, ..., T-1\} \tag{3.9}$$

In the above formulation, the objective function (3.5) is the weighted sum of two components: the total penalty for the violation of time windows; and the expected penalties for the violation of the limits C_t and C_{kt}. Constraint set (3.6) ensures that each train-set must arrive for maintenance within the planning horizon. Constraint set (3.7) enforces that at most one train-set can occupy the first operation line on any given day. The arrival day of each train-set can be calculated as (3.8). Constraint set (3.9) states that the decision variables are binary.

3.2.2 Evaluation of the Objective Function

In this section, we discuss an analytical approach to evaluate the objective function (3.4). While it is possible to approximate the objective function by using Monte Carlo sampling [81], this method is not more efficient than the proposed analytical method because it is still computationally expensive to perform a large number of Monte Carlo evaluations (in return

for a small error). We must note that in the proposed method, no property particular to the distributions of cycle times is used and therefore it can be generalized for any distribution.

Given an arrival plan $\sigma = (s_1, \ldots, s_n)$, let

$$Y_j(s_j, t) = \begin{cases} 1 & \text{if } s_j \leq t \text{ and } s_j + D_j \geq t + 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

which is a Bernoulli random variable that takes value 1 if the train-set j is at the maintenance center on day t and 0 otherwise. Then, the probability of $Y_j(s_j, t) = 1$ can be computed as

$$\begin{cases} \text{Prob}(Y_j(s_j,t) = 1) = \text{Prob}(D_j \geq t - s_j + 1) \\ \qquad\qquad\qquad\qquad = \displaystyle\sum_{i=t-s_j+1}^{b_k} \text{Prob}(D_j = i), \\ \qquad\qquad\qquad\qquad 1 \leq k \leq m,\ j \in F^k,\ t \in \{s_j, ..., T-1\} \\ \text{Prob}(Y_j(s_j,t) = 1) = 0, \quad j \in N,\ t \in \{0, ..., s_j - 1\} \end{cases} \tag{3.11}$$

For any arrival plan σ, and any day t, $W_t(\sigma) = \sum_{j \in N} Y_j(s_j, t)$ is a sum of Bernoulli random variables with success probabilities according to (3.11). Therefore, $W_t(\sigma)$ is a random variable that follows Poisson Binomial distribution [31]. For any arrival plan $\sigma = (s_1, \cdots, s_n)$, any $1 \leq i \leq n$, and any day t, denote $p_i = \text{Prob}(Y_i(s_i, t) = 1)$. Let $\text{Prob}(W_t(\sigma^l) = i)$ be the probability that i train-sets from the partial arrival plan $\sigma^l = (s_1, \ldots, s_l)$ dwell at the maintenance center on day t. Then,

$$\text{Prob}(W_t(\sigma^1) = 1) = p_1 \quad \text{and} \quad \text{Prob}(W_t(\sigma^1) = 0) = 1 - p_1$$

and for all $1 \leq l < n$

$$\begin{cases} \text{Prob}(W_t(\sigma^{l+1}) = 0) = (1 - p_{l+1})\text{Prob}(W_t(\sigma^l) = 0) \\ \text{Prob}(W_t(\sigma^{l+1}) = i) = (1 - p_{l+1})\text{Prob}(W_t(\sigma^l) = i) \\ \qquad\qquad + p_{l+1}\text{Prob}(W_t(\sigma^l) = i - 1), \quad 1 \leq i \leq l \\ \text{Prob}(W_t(\sigma^{l+1}) = l + 1) = p_{l+1}\text{Prob}(W_t(\sigma^l) = l) \end{cases}. \tag{3.12}$$

Let PMF be the probability mass function of the Poisson binomial distributed variable. The entire procedure is outlined in Algorithm 3 [31].

Algorithm 3 Direct Convolution (DC)

Input: total number of Bernoulli random variables n; success probability p_i of the i-th random variable
Output: PMF
procedure DC($p_1, \cdots, p_n$)
    $PMF(0) = 1 - p_1$, $PMF(1) = p_1$
    for i from 2 to n **do**
        $j = 1$
        $new\_PMF(0) = (1 - p_i)\ PMF(0)$
        while $j < i$ **do**
            $new\_PMF(j) = p_i\ PMF(j - 1) + (1 - p_i)\ PMF(j)$
        end while
        $new\_PMF(i) = p_i\ PMF(i - 1)$
        $PMF = new\_PMF$
    end for
    return PMF
end procedure

Similarly, for any arrival plan σ, any $1 \leq k \leq m$, and any day t, $W_t^k(\sigma) = \sum_{j \in F^k} Y_j(s_j, t)$ is a sum of Bernoulli random variables with success probabilities according to (3.11). Then, all probabilities $\text{Prob}(W_t^k(\sigma))$ can be obtained using Algorithm 3.

As a result, the objective function (3.4) can be computed as

$$\begin{aligned}\alpha G_1(\sigma) + \beta G_2(\sigma) = \alpha G_1(\sigma) + \\ \beta \sum_{t=0}^{T-1} \Bigg[\sum_{w=C_t+1}^{|N|} \delta_t (w - C_t) \text{Prob}(W_t(\sigma) = w) + \\ \sum_{k=1}^{m} \sum_{w=C_{kt}+1}^{|F^k|} \delta_{kt} (w - C_{kt}) \text{Prob}(W_t^k(\sigma) = w) \Bigg].\end{aligned} \tag{3.13}$$

3.2.3 Integer Linear Programming Relaxation based on Jensen's Inequality

For a two-stage stochastic program, under certain convexity properties, lower bounds on the optimal objective value of a stochastic program are typically based on Jensen's inequality (see e.g., [90, 135]). The Jensen's inequality lower bound is obtained by replacing all of the random variables with their expected values. An important advantage of this approach is that it requires only one function evaluation for each element of the partition. While other methods to compute lower bounds have been studied, i.e. Monte Carlo lower bound [109], improved Jensen's lower bound [164] and second-order lower bound [51], Jensen's lower bound typically requires less effort to compute.

As in (3.3), denoting the mathematical expectation by $\mathbb{E}[\cdot]$, the right-hand side in (3.13) can be written as

$$\alpha G_1(\sigma) + \beta \sum_{t=0}^{T-1} \Big(\delta_t \mathbb{E}[\max\{0, W_t(\sigma) - C_t\}] + \sum_{k=1}^{m} \delta_{kt} \mathbb{E}[\max\{0, W_t^k(\sigma) - C_{kt}\}] \Big)$$

and, taking into account that $\max\{0, \cdot\}$ is convex, by Jensen's Inequality [27],

$$\begin{aligned}
&\geq \alpha G_1(\sigma) + \beta \sum_{t=0}^{T-1} \Big(\delta_t \max\{0, \mathbb{E}[W_t(\sigma) - C_t]\} + \sum_{k=1}^{m} \delta_{kt} \max\{0, \mathbb{E}[W_t^k(\sigma) - C_{kt}]\} \Big) \\
&= \alpha G_1(\sigma) + \beta \sum_{t=0}^{T-1} \Big(\delta_t \max\{0, \mathbb{E}[W_t(\sigma)] - C_t\} + \sum_{k=1}^{m} \delta_{kt} \max\{0, \mathbb{E}[W_t^k(\sigma)] - C_{kt}\} \Big)
\end{aligned}$$

Denote

$$G_2'(\sigma) = \sum_{t=0}^{T-1} \Big(\delta_t \max\{0, \mathbb{E}[W_t(\sigma)] - C_t\} + \sum_{k=1}^{m} \delta_{kt} \max\{0, \mathbb{E}[W_t^k(\sigma)] - C_{kt}\} \Big)$$

Then,

$$\alpha G_1(\sigma) + \beta G_2(\sigma) \geq \alpha G_1(\sigma) + \beta G_2'(\sigma). \tag{3.14}$$

Using the variables x_{jt}, for any $t \in \{0, ..., T-1\}$,

$$\begin{aligned}
\mathbb{E}[W_t(\sigma)] &= \mathbb{E}\Big[\sum_{k=1}^{m} \sum_{j \in F^k} Y_j(s_j, t) \Big] = \sum_{k=1}^{m} \sum_{j \in F^k} \text{Prob}\Big(Y_j(s_j, t) = 1 \Big) \\
&= \sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js},
\end{aligned} \tag{3.15}$$

and, for any $t \in \{0, ..., T-1\}$ and any $1 \leq k \leq m$,

$$\begin{aligned}
\mathbb{E}[W_t^k(\sigma)] &= \mathbb{E}\Big[\sum_{j \in F^k} Y_j(s_j, t) \Big] = \sum_{j \in F^k} \text{Prob}(Y_j(s_j, t) = 1) \\
&= \sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js}.
\end{aligned} \tag{3.16}$$

This leads to the following Mixed Integer Programming Model (MIPM):

$$\text{(MIPM)} \quad Z_{MIPM} = \min \alpha \sum_{j \in N} \Big(\lambda_1 \sum_{t=0}^{\theta_j - \Delta - 1} (\theta_j - t)^2 x_{jt} + \lambda_2 \sum_{t=\theta_j+\Delta+1}^{T-1} (t - \theta_j)^2 x_{jt} \Big)$$
$$+ \beta \sum_{t=0}^{T-1} \Big(\delta_t \; w_t + \sum_{k=1}^{m} \delta_{kt} w_t^k \Big) \tag{3.17}$$

subject to (3.6), (3.7), (3.9)

$$\sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js} \leq C_t + w_t,$$
$$\forall t \in \{0, ..., T-1\} \tag{3.18}$$

$$\sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js} \leq C_{kt} + w_t^k,$$
$$1 \leq k \leq m, \; \forall t \in \{0, ..., T-1\} \tag{3.19}$$

$$w_t \geq 0, \; w_t^k \geq 0, \quad 1 \leq k \leq m, \; \forall t \in \{0, ..., T-1\}. \tag{3.20}$$

The nonlinear programming problem NIPM and the mixed integer linear programming problem MIPM use the same variables x_{jt}, that satisfy the same set of constraints (3.6), (3.7), (3.9). Therefore, for any arrival plan $\sigma = (s_1, ..., s_n)$ that is feasible for NIPM, there exists a feasible solution for MIPM with variables x_{jt}, w_t and w_t^k satisfying, for all $j \in N$, $t \in \{0, ..., T-1\}$ and $1 \leq k \leq m$,

$$s_j = \sum_{t=0}^{T-1} t x_{j,t}$$

$$w_t = \max \left\{ 0, \; \sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js} - C_t \right\}$$

$$w_t^k = \max \left\{ 0, \; \sum_{j \in F^k} \sum_{s=0}^{t} \sum_{i=t-s+1}^{b_k} \text{Prob}(D_j = i) x_{js} - C_{kt} \right\}.$$

or, by taking into account (3.15) and (3.16),

$$w_t = \max\{0,\ \mathbb{E}[W_t(\sigma)] - C_t\}$$
$$w_t^k = \max\{0,\ \mathbb{E}[W_t^k(\sigma)] - C_{kt}\}.$$

In other words, for any arrival plan $\sigma = (s_1, ..., s_n)$ that is feasible for NIPM, there exists a solution for the mixed integer linear programming problem MIPM which variables x_{jt}, w_t and w_t^k satisfy

$$\alpha G_1(\sigma) + \beta G_2'(\sigma) = \alpha \sum_{j \in N} \Big(\lambda_1 \sum_{t=0}^{\theta_j - \Delta - 1} (\theta_j - t)^2 x_{jt} + \lambda_2 \sum_{t=\theta_j + \Delta + 1}^{T-1} (t - \theta_j)^2 x_{jt} \Big) + \beta \sum_{t=0}^{T-1} \Big(\delta_t\ w_t + \sum_{k=1}^{m} \delta_{kt} w_t^k \Big).$$

This, by virtue of (3.14), implies that the optimal value Z_{MIPM} of the objective function for MIPM is a lower bound on the optimal value Z_{NIPM} of the objective function for NIPM.

3.3 Genetic Algorithm Approach

Since the considered problem involves the uncertainty of cycle times, we may expect that it can be very difficult to obtain exact solutions to problem instances beyond a certain size in a reasonable time. Correspondingly, we decided that approximation solutions of good quality should suffice and propose a Genetic Algorithm (GA) based metaheuristic. GA is a population-based search methodology combining principles of natural evolution and genetics for problem solving [46]. The underlying foundation of GA is related to the Darwinian theory of natural selection that fittest individuals have higher chances of surviving and breeding and that the fitness of an individual depends on the characteristics of its genes.

Li and Demeulemeester [104] proposed a genetic algorithm for the resource leveling problem in which the activity durations are random. The results presented in [104] indicate that GA is an effective metaheuristic for the stochastic optimisation problems. Here, we build upon the GA of [104] in two ways. First and foremost, at each generation, our GA can precisely determine whether a solution is better than another one by computing the objective function, instead of evaluating the solutions with a small number of scenarios. Second, our GA incorporates a migration phase that randomly generates some chromosomes before the offspring selection phase. The purpose of the migration phase is to prevent premature convergence.

In the considered problem, a solution is an arrival plan $\sigma = (s_1, \cdots, s_n)$ that specifies the arrival days for all train-sets. In our implementation of GA, starting with an initial population of POP different chromosomes, a decoding procedure transforms each chromosome into a feasible arrival plan (see Section 3.3.1). Next, the fitness value of each arrival plan of the initial population is computed by evaluating the objective function as described in Section 3.2.2. The plan with the smallest value on the objective function is the current best plan σ^*, and its corresponding objective value is the current best objective G^*. In each generation of the GA, the chromosomes in the current population are evolved to new chromosomes that enter the new population. The evolutionary process is comprised of four main components: parents selection, crossover, mutation, and offspring selection (see Section 3.3.2). The GA procedure stops when the total number of generations equal to $\text{GEN}_{\max}$. Algorithm 4 outlines our complete GA implementation.

In our implementation of GA, each arrival plan is encoded as a chromosome whose size is equal to the number of train-sets. Each gene j in the chromosome is a random number generated according to the uniform distribution $U(0, 1)$ which determines the priority of a train-set $j \in N$ (see Figure 3.1 for an example). The benefit of using the random key to encode the solution is that we do not directly deal with the train-set indexes. Hence, all offspring chromosomes generated by crossover and mutation are guaranteed to be decoded into feasible arrival plans.

Algorithm 4 Genetic Algorithm

```
Initialise population by generating POP different chromosomes randomly.
Set the current best objective G*, and the current best plan σ*
Set i = 1
while i ≤ GEN_max do
    Select a pool of parent chromosomes
    Generate POP/2 offspring chromosomes (resource-based crossover and two-point crossover)
    Diversify offspring chromosomes (mutation)
    Obtain a new population of POP chromosomes from the union set of parent chromosomes, offspring chromosomes, and some randomly generated chromosomes.
    Decode chromosomes into feasible arrival plans and evaluate their fitness.
    if (new best plan is found) then
        Update G*, σ*
    end if
    Set i = i + 1
end while
return σ*
```

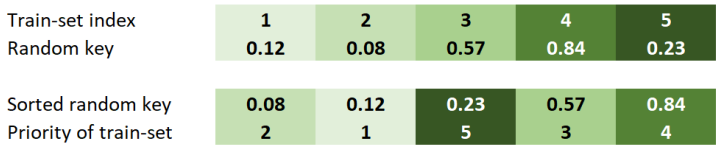


Figure 3.1: Random key encoding example. Each train-set is assigned a random number generated according to the uniform distribution $U(0, 1)$, which determine its priority in the decoding procedure. Here, train-set 2 has the highest priority, followed by train-set 1, 5, 3, and train-set 4 will be the last.

3.3.1 The decoding procedure

The decoding procedure transforms a chromosome into a feasible arrival plan. The decoding procedure is inspired by Li and Demeulemeester [104] and is detailed in Algorithm 5. In this pseudocode, N' is the set of planned train-sets, $W_t(N')$ is the total number of train-sets, residing in the maintenance center on the day t, and $W_t^k(N')$ denote the total number of train-sets of family F^k, residing in the maintenance center on the day t. The mean cycle time for the train-sets in F^k (denoted by $\bar{q}_k$) is used in the decoding procedure because of

the following reasons. First, the decoding procedure must be computationally efficient since it is frequently invoked in the GA. Second, by using the mean cycle time of the train-sets, some levels of uncertainty have been taken into consideration.

The decoding procedure starts with an empty set of planned train-sets (i.e. $N' = \emptyset$). In each iteration of the decoding procedure, a train-set $j \in N$, whose gene value is the smallest, is selected. From all feasible days within the planning horizon, the arrival day of train-set j is chosen such that it gives the smallest value on the performance measure. If a train-set j of type k arrives on the day s_j, the performance measure can be computed as

$$PM = \sum_{t=s_j}^{s_j+\bar{q}_k-1} \left(\delta_t \Big(W_t(N') + 1 \Big) + \delta_{kt} \left(W_t^k(N') + 1 \right) \right) \tag{3.21}$$

We do not use the objective function $G = \alpha G_1 + \beta G_2$ as the performance measure for deciding the best arrival day for a train-set because violation of the limits is always equal to zero at the early stages of the decoding procedure. The procedure is repeated until the arrival days of all the train-sets have been determined.

3.3.2 Evolutionary process

The evolutionary process evolves the current generation towards a better successive generation. It includes parent selection, resource-based crossover, two-point crossover, mutation, and offspring selection. The evolutionary process, inspired by Li and Demeulemeester [104], is described in the subsequent paragraphs.

The parent selection phase selects the top $POP/2$ best chromosomes in the current population as the father chromosomes. The remaining chromosomes form a pool of candidates from which two chromosomes are randomly chosen each time. The one with a smaller fitness value is nominated as the mother chromosome. The process is repeated until $POP/2$ mother chromosomes are obtained. Given the list of father chromosomes and the list of mother chromosomes, to form a pair of parent chromosomes, we pick one chromosome from the

Algorithm 5 Decoding Procedure

Input: A chromosome
Output: An arrival plan $\sigma = (s_1, \cdots, s_n)$
procedure
    Set $N' = \emptyset$, $W_t(N') = 0$, $W_t^k(N') = 0$, for $t = 0, ..., T-1$
    while $N \setminus N^{'} \neq \emptyset$ **do**
        $CB = \infty$
        Select a train-set j from $N \setminus N^{'}$ with the smallest gene value
        for ψ from 0 to $T-1$ **do**
            if ψ satisfies the restriction on the time between two consecutive arrivals **then**
                $PM = 0$
                for t from ψ to $\psi + \bar{q}_k - 1$ **do**
                    $PM = PM + \delta\Big(W_t(N^{'}) + 1\Big) + \delta_{kt}\Big(W_t^k(N^{'}) + 1\Big)$
                end for
                $PM = PM^2 + g_j(s_j)$
                if $PM < CB$ **then**
                    $CB = PM$, $s_j = \psi$
                end if
            else continue
            end if
        end for
        $N^{'} \leftarrow N^{'} \cup j$
    end while
    return $\sigma = (s_1, \cdots, s_n)$
end procedure

father list and one chromosome from the mother list. As a result, we have $POP/2$ pairs of parent chromosomes in total.

The crossover phase operates on the $POP/2$ pairs of parent chromosomes to produce $POP/2$ offspring chromosomes. Following the idea of Li and Demeulemeester [104], the crossover phase includes a resource-based crossover operator, which is applied to the parent chromosomes with the top $POP/4$ best father chromosomes, and a two-point crossover operator, which is applied to the remaining parent chromosomes. The details of these two crossover operators are given below. In the resource-based crossover operator, for a father chromosome and its corresponding arrival plan σ, a partial plan of length ε is randomly chosen in $[0.25T, 0.75T]$, a crossover point t is selected such that $G_2(\sigma)$ is minimized for the interval $[t, t+\varepsilon]$. Then, for the train-sets whose arrival days fall within the interval $[t, t+\varepsilon]$, the value of the corresponding gene is added by a big enough number (i.e., 5000) and given to the child chromosome. Here, we can choose any arbitrary number to be added to the gene value as long as it is much larger than 1. This is to ensure that the corresponding train-set will have the least priority in the decoding procedure at the next generation. The number 5000 is chosen in this study as recommended in [104]. The values of other genes are obtained from the mother chromosome (see Figure 3.2 for an example).

Father	0.98	0.41	0.05	0.94	0.81
Mother	0.69	0.01	0.16	0.37	0.45
Assume train-sets 2 and 3 has arrival days between t and t + ε					
Child	0.69	5000.41	5000.05	0.37	0.45

Figure 3.2: Resource-based crossover example. The crossover operator takes the gene values 0.41 and 0.05 from the father chromosome and places them in the same position in the child chromosome. The crossover operator fills in the missing gene values in the remaining places in the order they are defined in the mother chromosome. A big enough number (i.e., 5000) is then added to the genes indexed 2 and 3 to give the values 5000.41 and 5000.05.

In the two-point crossover operator, two crossover points t_1 and t_2 are randomly selected. The genes between t_1 and t_2 of the child chromosome are set equal to the corresponding genes in the father chromosome while the values of other genes are obtained from the mother

chromosome (see Figure 3.3 for an example).

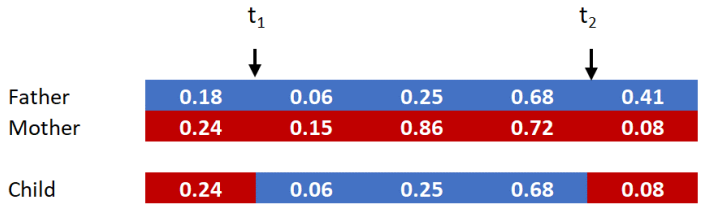


Figure 3.3: Two-point crossover example. Two crossover points $t_1 = 1$ and $t_2 = 4$ are selected. The values of genes 2, 3, and 4 are obtained from the father chromosomes, whereas the values of genes 1 and 5 are obtained from the mother chromosomes.

The mutation phase attempts to replace the values of some genes in the child chromosome. For each gene in the child chromosome, a random number is generated according to the uniform distribution $U(0, 1)$. Suppose this random number is less than *mutation_prob*. In that case, the corresponding gene value is replaced by a new random key generated according to the uniform distribution $U(0, 1)$ (see Figure 3.4 for an example).

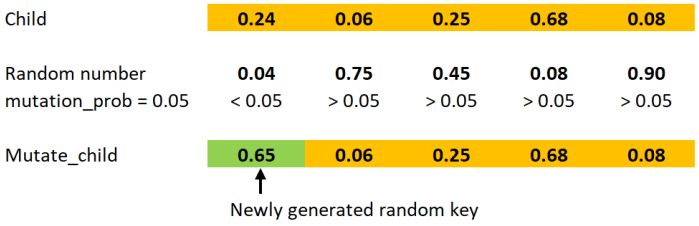


Figure 3.4: Mutation example. For each gene, one random number is generated from $U(0, 1)$. Since the random number of gene 1 is smaller than *mutation_prob* (i.e., $0.04 < 0.05$), the corresponding gene value is replaced by a new random key, i.e. 0.65.

Before the process of offspring selection, a migration phase is performed in which new chromosomes are randomly generated. The number of newly generated chromosomes is set equal to a proportion *mig_prob* of the population size. This step is added to the GA to prevent premature convergence [169]. Finally, the top *POP* best chromosomes from the union set of parent chromosomes, offspring chromosomes, and the newly generated chromosomes are selected as the candidates of the new population.

3.4 Genetic Algorithm based Matheuristic

The Genetic Algorithm presented in Section 3.3 transforms a sequence into a feasible arrival plan by means of a simple greedy heuristic. In doing so, the GA may suffer from poor performance as the idle time inserted between train-sets is not optimal. This observation leads to the development of an algorithm based on the idea of the amalgamation of GA and Integer Programming. The algorithm consists of a genetic algorithm for global search and an exact method to determine the arrival dates of train-sets when a sequence of train-sets is known.

Although both the Genetic Algorithm based Matheuristic and the GA presented in Section 3.3 are based on GA, the implementation of GA in these two algorithms is quite different:

1. The decoding procedure in Section 3.3 has been redesigned
   - by implementing a new method of choosing the priority of train-sets, i.e. the sequence of train-sets;
   - by changing the procedure that transforms a sequence into a feasible arrival plan.
2. The evolution model in Section 3.3 is a steady-state GA, where new chromosomes are added to, and unfit chromosomes are removed from the population. On the other hand, the GA in this section applies the generational with elitism strategy, where the most fitting individuals from the current population are artificially inserted in the new population.

In the developed version of GA, the arrival plans in the initial population (of size POP) are generated using a four-step procedure. First, we initialize the chromosomes using the random key encoding as described in Section 3.3, i.e., each gene in the chromosome corresponds to a train-set, and the gene value is a random number generated from the uniform distribution $U(0, 1)$. Second, we decide the priority of train types by sorting the list of train types by

Train-set Index	1	2	3	4	5
Train type	V	K	T	T	T
Due window	[0, 28]	[19, 47]	[30, 58]	[34, 62]	[41, 69]
Chromosome	0.57	0.08	0.84	0.12	0.23
Sorted gene values	0.08	0.12	0.23	0.57	0.84
Priority of train types	K	T	T	V	T
Decoded sequence	2	3	4	1	5

Figure 3.5: Example of the sequence decoding by train types.

their gene values in non-decreasing order. Third, we decide the sequence of train-sets by sorting the train-sets in the same train family based on their preferred arrival time windows. This process of decoding a chromosome into a sequence of train-sets is illustrated in Figure 3.5. Once a sequence is known, we determine the dates when the train-sets should arrive at the maintenance center by solving a mixed-integer linear program. This is achieved by using the Sample Average Approximation (SAA) method - an approach for solving stochastic optimization problems by using Monte Carlo simulation [155, 96, 109]. The SAA, as its name suggests, is an approach of replacing the original problem with its sampling approximation. The implementation of SAA is as follows. Let E be a set of arcs representing the precedence relations obtained from the sequence. Let D_j^γ denote the realization of cycle time of train-set $j \in N$ in scenario γ. Given a sample $\gamma^1, \gamma^2, ..., \gamma^\Gamma$ of Γ scenarios, we can then estimate the expected total penalty for violating the limits C_t and C_{kt} in (3.3) by the average total penalty over all scenarios. The resulting SAA problem is a large mixed integer program. An optimal solution to the SAA problem gives an optimal arrival plan with respect to the given sequence.

$$\text{(SAA)} \quad Z_{SAA} = \min\ \alpha \sum_{j \in N} \Big(\lambda_1 \sum_{t=0}^{\theta_j - \Delta - 1} (\theta_j - t)^2 x_{jt} + \lambda_2 \sum_{t=\theta_j + \Delta + 1}^{T-1} (t - \theta_j)^2 x_{jt} \Big)$$

$$+ \beta \frac{1}{\Gamma} \sum_{i=1}^{\Gamma} \sum_{t=0}^{T-1} \Big(\delta_t\ w_t^{\gamma^i} + \sum_{k=1}^{m} \delta_{kt} w_t^{k,\gamma^i} \Big) \tag{3.22}$$

subject to (3.6), (3.7), (3.9)

$$\sum_{t=0}^{T-1} t\ x_{jt} \leq \sum_{t=0}^{T-1} t\ x_{j't}, \quad \forall (j, j') \in E \tag{3.23}$$

$$\sum_{j \in N} \sum_{s=\max(t - D_j^{\gamma^i} + 1, 0)}^{t} x_{js} \leq C_t + w_t^{\gamma^i},\ 1 \leq i \leq \Gamma,\ t \in \{0, ..., T-1\} \tag{3.24}$$

$$\sum_{j \in F^k} \sum_{s=\max(t - D_j^{\gamma^i} + 1, 0)}^{t} x_{js} \leq C_{kt} + w_t^{k,\gamma^i}, \tag{3.25}$$

$$1 \leq i \leq \Gamma,\ 1 \leq k \leq m,\ t \in \{0, ..., T-1\} \tag{3.26}$$

$$w_t^{\gamma^i} \geq 0, \quad 1 \leq i \leq \Gamma,\ t \in \{0, ..., T-1\} \tag{3.27}$$

$$w_t^{k,\gamma^i} \geq 0, \quad 1 \leq i \leq \Gamma,\ 1 \leq k \leq m,\ t \in \{0, ..., T-1\} \tag{3.28}$$

The constraints (3.23) enforce the precedence relations among train-sets according to the given sequence. Constraints (3.24) calculate the additional train-sets ($w_t^{\gamma^i}$) beyond the limit C_t on day t under scenario γ^i, and likewise Constraints (3.26) calculate the additional train-sets of a particular type k (w_t^{k,γ^i}) beyond the limit C_{kt} on day t under scenario γ^i. Constraints (3.27) and (3.28) describe the domains for $w_t^{\gamma^i}$ and w_t^{k,γ^i}, respectively.

For $\Gamma = 5$, the instances of (SAA) can be solved quickly. In particulars, CPLEX solved (SAA) model in less than 10 seconds, roughly 22 times shorter than that required to solve (SAA) model without Constraint (3.23).

The next population of POP chromosomes is produced through elite selection, crossover,

and mutation. In the elite selection phase, the best fitting individuals of current population will continue to exist in the new population. The number of surviving individuals is equal to POP_{elite}. In the crossover phase, we use the two-point crossover, as discussed in Section 3.3.2. Two parent chromosomes are picked at random from a pool of chromosomes. This pool consists of the POP_{elite} best fitting individuals of current population and $POP - POP_{elite}$ individuals randomly selected from Roulette Wheel. In the mutation phase, some gene values in the offspring chromosomes will be replaced according to the mutation operator discussed in Section 3.3.2.

The Algorithm 6 below outlines the GA-based matheuristic.

Algorithm 6 GA-based Matheuristic

Initialise population by generating POP different chromosomes randomly.
Set the current best objective G^*, and the current best plan σ^*
Set $i = 1$
while $i \leq \text{GEN}_{\max}$ **do**
    Select POP_{elite} best fitting individuals from current population
    for $j = 1$ to $POP - POP_{elite}$ **do**
        Pick 2 parent chromosomes
        Generate 1 offspring chromosome (two-point crossover)
        Diversify offspring chromosome (mutation)
    end for
    Decode chromosomes into sequences of train-sets, obtain arrival plans by solving (SAA) model, and evaluate their fitness
    if (new best plan is found) **then**
        Update G^* and σ^*
    end if
    Set $i = i + 1$
end while
return σ^*

3.5 Hybrid Two-stage optimization Procedure

As has been discussed in Section 3.1, the existing literature often ignores the stochastic nature of cycle times and focuses on the optimization procedures that construct arrival plans assuming that the duration of maintenance is known. This also reflects the practice often

encountered in industry. Under the assumption of deterministic cycle times, the arrival plan is constructed either by solving a mathematical programming problem [105, 50] or by some metaheuristic [105, 161]. The Mixed Integer Linear Programming Problem (MILP) below also assumes that the cycle times are known constants, but in contrast to the previous publications the resulting arrival plan is evaluated as described in Section 3.2.2.

As far as metaheuristics are concerned, this study presents an Iterated Local Search (ILS) algorithm which is embedded into the multi-start framework. In contrast to the previous publications, the presented ILS does not assume that the cycle times are deterministic and evaluates each element in a neighborhood taking into account the stochastic nature of cycle times.

Furthermore, this study presents a hybrid optimization procedure which generates a starting solution by solving either the mixed integer linear programming problem MILP or the mixed integer linear programming problem MIPM and then enhances the obtained solution using the mentioned above ILS algorithm. As has been discussed above, when the hybrid optimization procedure generates the starting solution using the mixed integer linear programming problem MIPM, the optimal value of the objective function is a lower bound on the optimal value of the original nonlinear programming problem.

Section 3.6 reports the results of computational experiments which were aimed at the comparison of the quality of the solutions produced by the presented optimization procedures and the times needed to generate these solutions. Since the considered optimization problem is a component of the complex planning process which normally involves a lot of negotiations between the rolling stock operator and the maintenance center, both characteristics, quality and time, are important.

3.5.1 Mixed Integer Linear Program MILP

It is reasonable to model random cycle times using a beta-PERT distribution [110], which is commonly used in project management [133]. In this case, the constant duration is chosen as the most likely value of the cycle time according to the beta-PERT distribution. This approach is adopted in the computational experiments below. Let q_k be the most likely value of the cycle time for the train-sets in F^k. Then, the arrival day for each train-set j can be determined by

$$s_j = \sum_{t=0}^{T-1} t x_{jt}$$

where variables x_{jt} are obtained by solving the following mixed integer linear program

$$\text{(MILP)} \quad Z_{MILP} = \min \ \alpha \sum_{j \in N} \Big(\lambda_1 \sum_{t=0}^{\theta_j - \Delta - 1} (\theta_j - t)^2 x_{jt} + \lambda_2 \sum_{t=\theta_j+\Delta+1}^{T-1} (t - \theta_j)^2 x_{jt} \Big)$$
$$\beta \sum_{t=0}^{T-1} \Big(\delta_t \ w_t + \sum_{k=1}^{m} \delta_{kt} w_t^k \Big) \tag{3.29}$$

subject to

$$\text{(3.6), (3.7), (3.9)}$$
$$\sum_{k=1}^{m} \sum_{j \in F^k} \sum_{s=\max(0, t-q_k+1)}^{t} x_{js} \le C_t + w_t, \ t \in \{0, ..., T-1\} \tag{3.30}$$
$$\sum_{j \in F^k} \sum_{s=\max(0, t-q_k+1)}^{t} x_{js} \le C_{kt} + w_t^k,$$
$$1 \le k \le m, \ \forall t \in \{0, ..., T-1\} \tag{3.31}$$
$$w_t \ge 0, \ w_t^k \ge 0, \quad 1 \le k \le m, \ \forall t \in \{0, ..., T-1\}. \tag{3.32}$$

3.5.2 Local Search Subroutines

As has been discussed above, the considered planning problem can be solved either by some local search based metaheuristic or by a hybrid algorithm that produces a starting solution by solving one of the two mixed integer linear programs, MIPM or MILP, and then enhances this solution by some local search based optimization procedure. Four neighborhood operators, $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_1'$, and $\mathcal{N}_2'$, as described below, will be used for this purpose.

Neighborhood operators

Consider an arbitrary arrival plan $\sigma = [s_1, ..., s_n]$ and a sequence $t_1, ..., t_n$ of arrival days which are listed in a nondecreasing order and which are obtained by changing a single arrival day in σ, say by changing the arrival day s_g of some train-set g. For each $1 \leq j \leq n$, let $n(j)$ be the train-set that arrives at t_j, and let $k(j)$ be the type of $n(j)$. Let t_i be the new arrival day assigned to the train-set g. The replacement of s_g by t_i results in a new feasible arrival plan if

(p1) $i = 1$ and $t_1 + \tau_{k(1)} \leq t_2$;

(p2) $1 < i < n$, $t_{i-1} + \tau_{k(i-1)} \leq t_i$ and $t_i + \tau_{k(i)} \leq t_{i+1}$;

(p3) $i = n$ and $t_{n-1} + \tau_{k(n-1)} \leq t_n$.

The neighborhood explored by the operator $\mathcal{N}_1$ is comprised of all feasible arrival plans (solutions) that can be obtained from the input arrival plan σ by assigning a different arrival day to a single train-set. In other words, the neighborhood explored by the operator $\mathcal{N}_1$ is comprised of all arrival plans that are obtained when the change of a single arrival day in σ results either in (p1), or (p2), or (p3). The benefit of using $\mathcal{N}_1$ is in the fast evaluation of each solution in the neighborhood because each solution is obtained by changing only a single arrival day and this change does not affect any other arrival days. The operator $\mathcal{N}_2$

explores all solutions that can be obtained by changing any two arrival days of the input arrival plan. Similar to $\mathcal{N}_1$, these two changes must not affect any other arrival days in the input arrival plan.

Other two operators $\mathcal{N}_1'$ and $\mathcal{N}_2'$ are similar to $\mathcal{N}_1$ and $\mathcal{N}_2$ but their neighborhoods are constructed without the restriction that the change of an arrival day in σ does not affect any unchanged days in σ. More specifically, the operator $\mathcal{N}_1'$ explores the neighborhood comprised of the solutions that result from a change of a single arrival day in the input arrival plan, but in contrast to $\mathcal{N}_1$, this change is allowed to violate all three feasibility conditions (p1), (p2), and (p3). In such case, the set of arrival days is transformed into a feasible arrival plan by the algorithm TRANSFORMATION and its two subroutines LEFT and RIGHT. Similarly, the operator $\mathcal{N}_2'$ explores all solutions that can be obtained by changing any two arrival days in the input arrival plan, but in contrast to $\mathcal{N}_2$, these two changes may affect some other arrival days in the input solution. In such case, analogously to $\mathcal{N}_1'$, the resultant set of arrival days is transformed into a feasible arrival plan using the same algorithm TRANSFORMATION.

Algorithm 7 TRANSFORMATION

```
1: if t_i < s_g then
2:     LEFT(i, t_1, ..., t_n)                    ▷ Algorithms 8
3: else
4:     RIGHT(i, t_1, ..., t_n)                   ▷ Algorithm 9
5: end if
6: for j = 1; j ≤ n; j++ do
7:     s_{n(j)} = t_j
8: end for
9: return [s_1, ..., s_n]
```

Algorithm 8 LEFT

$t_i^{'} = t_i$
if $i > 1$ **then**
    for $j = i - 1; j > 0; j$- **do**
        $t_j^{'} = \min\{t_j, t_{j+1}^{'} - \tau_{k(j)}\}$
    end for
    if $t_1^{'} < 0$ **then**
        $t_1 = 0$
        for $j = 2; j \leq i; j$++ **do**
            $t_j = t_{j-1} + \tau_{k(j-1)}$
        end for
    else
        for $j = 1; j \leq i; j$++ **do**
            $t_j = t_j^{'}$
        end for
    end if
end if
if $i < n$ **then**
    for $j = i; j < n; j$++ **do**
        $t_{j+1} = \max\{t_j + \tau_{k(j)}, t_{j+1}\}$
    end for
end if
return $[t_1, ..., t_n]$

Algorithm 9 RIGHT

$t_i^{'} = t_i$
if $i < n$ **then**
    for $j = i; j < n; j$++ **do**
        $t_{j+1}^{'} = \max\{t_{j+1}, t_j^{'} + \tau_{k(j)}\}$
    end for
    if $t_n^{'} > T - 1$ **then**
        $t_n = T - 1$
        for $j = n - 1; j \geq i; j$- **do**
            $t_j = t_{j+1} - \tau_{k(j)}$
        end for
    else
        for $j = i + 1; j \leq n; j$++ **do**
            $t_j = t_j^{'}$
        end for
    end if
end if
if $i > 1$ **then**
    for $j = i; j > 1; j$- **do**
        $t_{j-1} = \min\{t_j - \tau_{k(j-1)}, t_{j-1}\}$
    end for
end if
return $[t_1, ..., t_n]$

Let $\mathfrak{N}$ be one of the four operators, $\mathcal{N}_1$, or $\mathcal{N}_1'$, or $\mathcal{N}_2$, or $\mathcal{N}_2'$. For each solution $\tilde{\sigma} = [\tilde{s}_1, ..., \tilde{s}_n]$ in the neighborhood of an input arrival plan σ, the operator $\mathfrak{N}$ computes the value of the objective function

$$\begin{aligned} \alpha G_1(\tilde{\sigma}) + \beta G_2(\tilde{\sigma}) = \alpha \sum_{1 \le j \le n} g_j(\tilde{s}_j) + \beta \sum_{t=0}^{T-1} \Big(\mathbb{E}\left[\max\{0, \delta_t(W_t(\tilde{\sigma}) - C_t)\}\right] \\ + \sum_{k=1}^{m} \mathbb{E}\left[\max\{0, \delta_{kt}(W_t^k(\tilde{\sigma}) - C_{kt})\}\right] \Big). \end{aligned} \tag{3.33}$$

Let $\hat{\sigma}$ be a solution in the neighborhood of σ with the smallest value of the objective function. Denote by $\mathfrak{N}(\sigma)$ the output solution produced by the operator $\mathfrak{N}$. Then

$$\mathfrak{N}(\sigma) = \begin{cases} \sigma & \text{if } \alpha G_1(\sigma) + \beta G_2(\sigma) \le \alpha G_1(\hat{\sigma}) + \beta G_2(\hat{\sigma}) \\ \hat{\sigma} & \text{otherwise} \end{cases} \tag{3.34}$$

The Algorithm 10 below outlines the local search procedure for an input arrival plan σ. If $\mathfrak{N}$ is $\mathcal{N}_i$, where $i \in \{1, 2\}$, then this procedure will be referred to as LSi. If $\mathfrak{N}$ is $\mathcal{N}_i'$, where $i \in \{1, 2\}$, this procedure will be referred to as LSi$'$.

Algorithm 10 Local Search

```
repeat
    σ̄ = σ
    σ = 𝔑(σ)
until αG1(σ) + βG2(σ) == αG1(σ̄) + βG2(σ̄)
return σ̄
```

The four search operators, mentioned above, can be combined in different ways. In this study, four options are considered: (1) LS1 is a local search that uses only the operator $\mathcal{N}_1$; (2) LS1$'$ is a local search that uses only the operator $\mathcal{N}_1'$; (3) Sequential Local Search (SLS) applies LS1 to the input arrival plan and when LS1 terminates applies LS2 to the output of LS1; and (4) SLS$'$ applies LS1$'$ to the input arrival plan and when LS1$'$ terminates applies LS2$'$ to the output of LS1$'$. Each option has its own merit as demonstrated in the computational study.

Evaluation of solutions in a neighborhood

The main computational burden in many local search algorithms is the evaluation of the elements constituting a neighborhood. The local search procedures LS1, LS2, LS1$'$ and LS2$'$ are no exception. Therefore, an algorithm for computing (3.33) for each element of the neighborhood of an input solution is the key factor that determines the efficiency of these optimization procedures.

As (3.13) indicates, in order to recompute (3.33) efficiently, one requires a fast algorithm for recomputing the probability mass functions of the random variables W_t, W_t^k for all $t \in \{0, ..., T-1\}$ and $1 \leq k \leq m$. All these random variables have the same nature - they are sums of Bernoulli random variables. Given the structure of neighborhoods explored by the operators $\mathcal{N}_1$ and $\mathcal{N}_2$, each element in the neighborhood of the input arrival plan can be obtained by changing the arrival day of one or two trains-sets in this input solution. For each such train-set, the change in the arrival day results in the change of one of the Bernoulli random variables in the sums defining the random variables W_t, W_t^k for all $t \in \{0, ..., T-1\}$ and $1 \leq k \leq m$. This replacement can be viewed as the elimination of one of the Bernoulli random variables from the sum followed by the addition of another Bernoulli random variable to the result of the elimination. The Algorithm 11 and Algorithm 12 below do this efficiently for any probability mass function of a random variable that is a sum of Bernoulli random variables, and therefore, are used for recomputing probability mass functions of the random variables W_t, W_t^k for all $t \in \{0, ..., T-1\}$ and $1 \leq k \leq m$.

Let p be the success probability of the Bernoulli random variable that is to be eliminated, PMF be the original probability mass function, and new_PMF be the probability mass function resulted from the elimination of this Bernoulli random variable. The probability mass function new_PMF is defined for all integers $0 \leq i \leq n-1$, whereas the probability mass function PMF is defined for all integers $0 \leq i \leq n$. For each i from the domain of new_PMF, if $p = 0$, then $new\_PMF(i) = PMF(i)$, whereas if $p = 1$, then

$new\_PMF(i) = PMF(i+1)$. If $0 < p < 1$, then

$$PMF(0) = (1-p)\, new\_PMF(0)$$

and, for $1 \leq i \leq n-1$,

$$PMF(i) = p\, new\_PMF(i-1) + (1-p)\, new\_PMF(i)$$

This observations lead to the Algorithm 11 below.

Algorithm 11 Single Train-set Removal (STR)

```
1: if 0 < p < 1 then
2:     new_PMF(0) = PMF(0) / (1 − p)
3:     for i from 1 to n - 1 do
4:         new_PMF(i) = [PMF(i) − new_PMF(i − 1) ∗ p] / (1 − p)
5:     end for
6: else
7:     for i from 0 to n - 1 do
8:         if p = 1 then
9:             new_PMF(i) = PMF(i + 1)
10:        else
11:            new_PMF(i) = PMF(i)
12:        end if
13:    end for
14: end if
15: return new_PMF
```

Let p be the success probability of the Bernoulli random variable that is to be added to a sum of Bernoulli random variables which has the probability mass function PMF. Let new_PMF be the probability mass function of the sum after this addition. The probability mass function new_PMF is defined for all integers $0 \leq i \leq n$, whereas the probability mass function PMF is defined for all integers $0 \leq i \leq n-1$. The reasoning similar to the above lead to the Algorithm 12.

Algorithm 12 Single Train-set Addition (STA)

if $0 < p < 1$ **then**
  $new\_PMF(0) = PMF(0) * (1 - p)$
  for i from 1 to $n - 1$ **do**
    $new\_PMF(i) = PMF(i) \ * \ (1 - p) \ + PMF(i - 1) \ * \ p$
  end for
  $new\_PMF(n) = PMF(n - 1) * p$
end if
if $p = 0$ **then**
  $new\_PMF(n) = 0$
  for i from 0 to $n - 1$ **do**
    $new\_PMF(i) = PMF(i)$
  end for
end if
if $p = 1$ **then**
  $new\_PMF(0) = 0$
  for i from 1 to n **do**
    $new\_PMF(i) = PMF(i - 1)$
  end for
end if
return new_PMF

3.5.3 Iterated Local Search

Iterated local search (ILS) is one of the commonly used metaheuristics which was successful in solving a wide range of optimization problems [107]. The Algorithm 13 below outlines this metaheuristic as it was implemented and used in the computational experiments, the results of which are reported in Section 3.6. In this pseudocode, G is the objective function (3.4) and the parameter U specifies the maximum permissible number of consecutive unsuccessful attempts to improve the current best known arrival plan σ^*.

The Algorithm 13 interchangeably invokes two subroutines, SEARCH and PERTURB. In some computational experiments, reported in Section 3.6, the subroutine SEARCH is a local search procedure LS1 or LS1$'$ (see Algorithm 10), whereas in the others, the subroutine SEARCH is the sequential local search SLS or SLS$'$. The subroutine PERTURB randomly chooses three train-sets and one by one assigns to them new arrival days without violating

the feasibility (that is, without violating the restrictions on the duration of time intervals between any two consecutive arrivals of train-sets). In the process of assigning the arrival days to the three selected train-sets, the subroutine PERTURB does not take into account the new value of the objective function G. In what follows, for any arrival plan $\tilde{\sigma}$, SEARCH($\tilde{\sigma}$) and PERTURB($\tilde{\sigma}$) denote the output of SEARCH and PERTURB, respectively, resulted from their application to $\tilde{\sigma}$. The arrival plan σ in SEARCH(σ) in line 1 is a feasible (in terms of the time between the consecutive arrivals) input solution.

Algorithm 13 Iterated Local Search

```
1: σ* = SEARCH(σ)
2: u = 0
3: while u ≤ U do
4:     σ = PERTURB (σ*)
5:     σ = SEARCH (σ)
6:     if G(σ*) > G(σ) then
7:         σ* = σ
8:         u = 0
9:     else
10:        u = u + 1
11:    end if
12: end while
13: return σ*
```

An input arrival plan for the ILS is generated either by solving one of the two mixed integer linear programs, MIPM or MILP, or by the heuristic INITIAL described below. The heuristic INITIAL chooses randomly n different arrival days $t_1 < ... < t_n$ and associates randomly with each t_i one of m types of train-sets in such a manner that each type $1 \leq k \leq m$ receives $|F^k|$ arrival days. Denote by $k(t_i)$ the type associated with t_i. If, for each $1 \leq i < n$,

$$t_i + \tau_{k(t_i)} \leq t_{i+1},$$

then the arrival days are feasible. If they are not feasible, then the heuristic INITIAL transforms the generated arrival days into feasible arrival days, using the Algorithm 14 below. After obtaining feasible arrival days, the heuristic INITIAL generates an arrival plan by

assigning the days associated with each type of train-sets to the train-sets of this type in the increasing order of their preferred days θ_j.

Algorithm 14 Arrivals Adjustment

for i from $n-1$ to 1 **do**
    $t_i = \min[t_i, t_{i+1} - \tau_{k(t_i)}]$
end for
if $t_1 < 0$ **then**
    $t_1 = 0$
    for i from 2 to n **do**
        $t_i = \max[t_i, t_{i-1} + \tau_{k(t_{i-1})}]$
    end for
end if
return $t_1, ..., t_n$

3.6 Computational Results

The proposed solution approaches are tested and evaluated on data provided by one of the leading maintenance centers in Australia. The planning horizon is one year. There are 35 train-sets of 3 different types. The parameters of train-sets in F^k, $1 \leq k \leq 3$ are given in Table 3.1. The column '$|F^k|$' reports the total number of train-sets of the same type k; the column 'τ_k' gives the number of days a train-set of the corresponding type spends on the first operation line. Since an increase in transport demand is often expected during public holidays, the number of train-sets of type k which can dwell at the maintenance center simultaneously during these days is normally less than that on any other days of the year. The out-of-service limit 'C_{kt}' for type k on day t is reported in column 'non-PH' if day t is not a public holiday, and in column 'PH' if day t is a public holiday, where PH is an abbreviation of public holiday.

Table 3.1: Parameters for the train types.

Train type k	$\|F^k\|$	τ_k	C_{kt}	
			non-PH	PH
1	25	4	3	1
2	5	5	2	1
3	5	5	1	1

For any day t, the capacity of the maintenance center imposes a limit $C_t = 5$, and the daily penalty factor δ_t is fixed at 1. For any $1 \leq k \leq 3$, and any day t, the daily penalty factor δ_{kt} $= 1$ if t is not a public holiday, and $\delta_{kt} = 10$ if t is a public holiday. A larger penalty is applied for public holidays because it is more undesirable to violate the given limit C_{kt} during these periods. With a larger penalty factor δ_{kt}, the violation of the permissible number of train-sets of type k that can dwell at the maintenance center simultaneously on public holidays is still possible in the optimal solution.

The permissible deviation from the preferred day of the commencement of maintenance is 14 days, i.e. $\Delta = 14$. The penalty factors for the violation of time windows are chosen as $\lambda_1 = \lambda_2 = 1$.

As has been discussed in Section 3.5.1, the random cycle times are modelled using beta-PERT distribution with the minimum, most likely, and maximum values as given in Table 3.2 for each train type k. Note that the beta-PERT distribution is a continuous probability distribution. So for the computational experiments, beta-PERT distribution is discretized into days.

Table 3.2: Parameters of probability distribution for cycle time by train types.

Train type	Minimum	Most likely	Maximum	Distribution
1	20	25	40	beta-PERT
2	27	30	46	beta-PERT
3	29	30	52	beta-PERT

Extensive experiments were performed on numerous settings for the weights α and β. The choice of values presented in Table 3.3 corresponds to the appropriate weight coefficients

that can generate solutions representing typical scenarios. For case 1, a large relative weight is assigned to the second component of the objective function $G_2(\sigma)$, and the optimization procedures aim at minimizing the expected penalties for the violation of the limits C_t and C_{kt}. The importance of the objective $G_1(\sigma)$ increases when proceeding from case 1 to case 9.

Table 3.3: Assignment of α and β for all the cases.

Case	α	β	Case	α	β
1	1	1000	6	1	100
2	1	300	7	1	50
3	1	200	8	1	10
4	1	180	9	1	1
5	1	150			

All algorithms are implemented in Python 2.7. The mixed integer linear programs are solved with IBM ILOG CPLEX 12.7 via the mathematical programming modeling language PuLP [117]. All tests are run on a computer with Intel i5-6300U 2.4GHz processor and 8GB of RAM.

3.6.1 Comparison of GA and GA-based matheuristic

Table 3.4 shows the performance of Genetic Algorithm (GA) and GA-based matheuristic (GA-M) with $\Gamma = 10$ in terms of solution quality and time. For each case, i.e., for each choice of the parameters α and β, we ran the algorithms 10 times. Because both algorithms start with a randomly generated initial population, we report the average value of the objective function of the best solution found in initial population (In. Obj.) and the average value of the objective function of the best solution found when the algorithm terminates (Obj.). The column 'Time' displays the average computation time in seconds. For all cases, the objective values of the solutions are computed as described in Section 3.2.2. We have fixed the parameter values as follows: $POP = 20$, $POP_{elite} = 2$, $\mathrm{GEN}_{\max} = 40$, $mutation\_prob = 0.05$, and $mig\_prob = 0.05$.

The performance of GA-M ($\Gamma = 10$) is consistently better than GA irrespective of the assign-

Table 3.4: Comparison of the performance of GA and GA-based matheuristic

Case	GA			GA-M ($\Gamma = 10$)		
	In. Obj.	Obj.	Time	In. Obj.	Obj.	Time
1	417,581	341,962	960	332,519	233,634	11878
2	150,158	132,345	600	123,468	87,171	10937
3	111,955	99,165	600	95,445	68,758	8300
4	104,314	92,363	1200	78,083	64,514	11355
5	92,853	72,170	2460	71,392	58,163	11041
6	73,752	59,084	2400	63,792	41,351	10565
7	52,509	35,868	1380	33,754	29,228	10370
8	34,363	27,959	600	24,696	11,979	10103
9	30,280	22,235	900	28,127	7,772	10074
Average	118,641	98,128	1233	94,586	66,952	10514

ment of α and β. Using GA-M ($\Gamma = 10$) results in final solutions with objective values that are, on average, 35% better than GA. The percentage relative improvement in the objective function value over the best solution of the initial population is 33% for GA-M ($\Gamma = 10$) and 19% for GA. It can be observed that the proposed GA-based matheuristic produced superior solutions but at the cost of longer computation time. The GA-M ($\Gamma = 10$) took between 138 and 198 minutes, with an average of 8 times longer than the time required by GA.

From Table 3.4, for all 9 cases, it is observed that the best solutions found by GA are always worse than the best solution found in the initial population of GA-M. Hence, even if the same amount of time is given to both GA and GA-M, the latter will continue to be better. GA-M is better than GA because of the inclusion of CPLEX that is able to determine the optimal idle time to be inserted between train-sets when a sequence of arrival is given.

Table 3.5 presents the analysis on the performance of the GA-based matheuristic with $\Gamma \in \{1, 5, 10\}$. We find in our preliminary experiments that, the time required to solve the problem (3.22) - (3.28) grows rapidly as Γ increases. Therefore, as a good trade-off between solution quality and computation time, we set $\Gamma = 1, 5, 10$. In all cases, the algorithm was permitted to run for 1800 seconds. Using $\Gamma = 10$ results in initial solutions with objective values that are, on average, 4% better than $\Gamma = 5$ and 9% better than $\Gamma = 1$. This result is not surprising since the solution quality of the approximation of the objective function (3.22) by its sample

Table 3.5: Analysis of the performance of GA-based matheuristic with $\Gamma \in \{1, 5, 10\}$, when the algorithm is run in 1800 seconds.

Case	GA-M ($\Gamma = 1$)		GA-M ($\Gamma = 5$)		GA-M ($\Gamma = 10$)	
	In. Obj.	Obj.	In. Obj.	Obj.	In. Obj.	Obj.
1	355,145	240,559	345,911	278,635	301,782	255,545
2	131,396	101,181	128,202	98,163	122,936	90,590
3	88,170	59,281	78,966	68,635	95,427	82,762
4	97,717	64,695	85,242	77,847	90,411	75,436
5	92,109	62,948	81,867	70,249	69,464	69,464
6	56,825	44,869	70,249	48,164	56,888	50,456
7	45,398	26,176	42,503	28,164	45,152	31,035
8	26,238	15,480	24,466	13,836	32,462	19,477
9	27,809	8,081	19,992	9,411	25,265	11,727
Average	102,312	69,252	97,489	77,012	93,310	76,277

average increases with increasing number of scenarios. The average relative improvement in the objective function value over the best solution of the initial population is approximately 37%, 27%, 23%, respectively, for 1, 5, and 10 scenarios. The small relative improvement of GA-M ($\Gamma = 10$) is due to the amount of time required for solving the (SAA) model. As a result, only a few generations are explored (see Table 3.6) and the search capability of Genetic Algorithm is not fully employed. The comparison of the different GA-M variants for Case 1 considering number of generations, i.e., 40 generations, is reported in Table 3.7.

Table 3.6: The number of generations performed to produce the solutions in Table 3.5.

Case	GA-M ($\Gamma = 1$)	GA-M ($\Gamma = 5$)	GA-M ($\Gamma = 10$)
1	35	12	6
2	33	12	5
3	28	13	6
4	27	11	5
5	30	8	6
6	28	10	6
7	26	13	6
8	25	13	7
9	25	13	7
Average	29	12	6

Table 3.7: Performance of GA-based matheuristic with $\Gamma \in \{1, 5, 10\}$ for Case 1, when the algorithm is run for 40 generations.

Case	GA-M ($\Gamma = 1$)			GA-M ($\Gamma = 5$)			GA-M ($\Gamma = 10$)		
	In. Obj.	Obj.	Time (s)	In. Obj.	Obj.	Time (s)	In. Obj.	Obj.	Time (s)
1	345,745	270,301	1973	337,906	240,495	5845	332,519	233,634	10514

3.6.2 Comparison of the Performance of MIPM and MILP

Table 3.8 compares the results of the two mixed integer linear programs. As has been discussed in Section 3.2.3, solving the MIPM produces a lower bound which is reported under the column titled 'LB'. The column 'Time' gives the running time (in seconds) by CPLEX to obtain an optimal solution. For all cases, the objective values of the solutions are computed as described in Section 3.2.2 and reported under the column titled 'Obj.'. The relative gap is calculated as $\%\text{Rel} = (\text{Obj.} - \text{LB})/\text{LB} \times 100$.

The results in Table 3.8 shows that the MILP can be solved in short computation time, yet the MIPM has the advantage of providing a lower bound. It is observed that the MIPM gives better solutions in 6 out of 9 cases, and on average 0.54% better than the MILP. For all cases, CPLEX obtains an optimal solution to MILP in less than 11 seconds, whereas more time is needed to solve the MIPM to optimality. By investigating the output of CPLEX in case 1 which takes the longest time, it was found that the optimal solution in fact was obtained in less than 2 minutes. The remaining time was taken by CPLEX for proving optimality, which is a common behavior of this software. As the relative weight of β decreases, the computation time of MIPM reduces substantially. This observation suggests that the second component of the objective function is harder to optimize for the MIPM.

Table 3.8: Comparison of the performance of MIPM and MILP

Case	LB	MIPM			MILP		
		Time	Obj.	%Rel	Time	Obj.	%Rel
1	156,744	2,078	206,060	31.46	7	**201,203**	**28.36**
2	62,612	235	**77,764**	**24.20**	7	80,353	28.33
3	48,629	68	59,178	21.69	8	**59,103**	**21.54**
4	45,768	100	**55,368**	**20.98**	8	56,322	23.06
5	40,190	54	**48,447**	**20.54**	11	49,369	22.84
6	30,789	46	**36,245**	**17.72**	8	36,572	18.78
7	20,940	41	23,624	12.82	8	**23,337**	**11.45**
8	10,594	39	**10,753**	**1.50**	8	10,818	2.11
9	6,706	38	**6,725**	**0.28**	8	6,748	0.63
Average	46,997	300	58,240	16.80	8	58,203	17.46

Table 3.9: Improvements in solution quality of MIPM and MILP by the local search LS1.

Case	MIPM-LS1			MILP-LS1		
	LS Time	Obj.	%Rel	LS Time	Obj.	%Rel
1	46.32	**189,551**	**20.93**	19.59	192,637	22.90
2	44.29	**72,584**	**15.93**	32.72	75,368	20.37
3	36.11	56,483	16.15	46.72	**55,102**	**13.31**
4	28.95	53,125	16.07	60.71	**51,999**	**13.61**
5	67.76	**42,884**	**6.70**	46.84	44,547	10.84
6	22.06	**32,971**	**7.09**	13.01	35,356	14.83
7	7.44	22,948	9.59	26.11	**22,905**	**9.38**
8	10.56	**10,752**	**1.49**	15.11	10,793	1.88
9	6.89	**6,724**	**0.27**	6.85	6,732	0.39
Average	30.04	54,225	10.47	29.74	55,049	11.95

Table 3.10: Improvements in solution quality of MIPM and MILP by the sequential local search SLS.

Case	MIPM-SLS			MILP-SLS		
	LS Time	Obj.	%Rel	LS Time	Obj.	%Rel
1	879	**189,551**	**20.93**	2,366	190,584	21.59
2	871	**72,584**	**15.93**	3,593	72,868	16.38
3	884	56,483	16.15	931	**54,766**	**12.62**
4	802	53,125	16.07	3,245	**51,208**	**11.89**
5	862	**42,884**	**6.70**	3,589	44,173	9.91
6	838	**32,971**	**7.09**	3,592	34,911	13.39
7	3,196	**22,631**	**8.08**	899	22,905	9.38
8	699	**10,752**	**1.49**	687	10,793	1.88
9	708	**6,724**	**0.27**	749	6,732	0.39
Average	1,082	54,189	10.30	2,183	54,327	10.82

The improvements in solution quality by the hybrid two-stage optimization procedure that combines the mixed integer linear program with the local search LS1 and the Sequential Local Search (SLS) are reported in Tables 3.9 and 3.10, respectively. In both tables, the column 'LS Time' gives the computation time (in seconds) of the local search procedure. Results in Table 3.9 show that the local search LS1 is effective in improving the solutions of both models for all cases. On average, the relative gap is reduced by 30.56% for MIPM and 29.71% for MILP. The behavior of the local search is consistent for both models: the best performance is achieved in case 5 with a change in the objective value of 11.48% for MIPM, and 11.76% for MILP; whereas the worst performance is observed in case 8 at 0.004% and 0.23%, respectively. The MIPM with LS1 performs better than the MILP with LS1. The

former produces solutions that are, on average, 1.48% better than the latter, with the same average time of 30 seconds. Moreover, the local search applied to a better initial solution does not necessarily produce a better local optimum, as demonstrated by the result of case 1.

If the SLS is used to enhance the starting solutions of MIPM and MILP, results in Table 3.10 show that the MIPM with SLS yields better solutions in shorter running times. For MIPM, the search in the neighborhood explored by the operator $\mathcal{N}_2$ finds better solution in only one of the nine cases, i.e. case 7. For MILP, the SLS is seen to be especially useful on cases with large relative weight β. Since the MIPM produces better results over the MILP for most cases, MIPM is used in the remainder of the computational experiments.

3.6.3 Comparison of Hybrid ILS and Multi-start ILS

In this section, eight algorithms, namely hybrid ILS with LS1, hybrid ILS with LS1$'$, hybrid ILS with SLS, hybrid ILS with SLS$'$, multi-start ILS with LS1, multi-start ILS with LS1$'$, multi-start ILS with SLS, and multi-start ILS with SLS$'$, are compared by means of computational experiments. A summary of the algorithms can be found in Table 3.11.

For all eight algorithms, a maximum permissible number of iterations without improvement $U = 20$ is used. The computational results are reported in Tables 3.12 - 3.15. In the preliminary testing, performing an exhaustive search on the neighborhood by $\mathcal{N}_2$ and $\mathcal{N}_2'$ is too time consuming, which significantly reduces the number of perturbation in the ILS procedure due to the time limit of 1 hour. As a result, the ability of ILS to escape the local optimum is severely impaired. Therefore instead of allowing the arrival days of two train-sets to be changed to any feasible days from $\{0, ..., T-1\}$, the newly assigned arrival day t of each such train-set j is selected from the range $s_j - 36 \leq t \leq s_j + 36$, where s_j is obtained from the input arrival plan σ. Such restriction substantially reduces the size of the search space so that the neighborhood can be explored in a reasonable time. The reduced structure of neighborhood explored by $\mathcal{N}_2$ and $\mathcal{N}_2'$ is employed in the computational experiments of

Table 3.11: Summary of the eight algorithms used in Section 3.6.3.

Algorithm	Description of the algorithm
Hybrid ILS with LS1	hybrid algorithm that produces a starting solution by solving MIPM and then enhances this solution by a local search that uses only the operator $\mathcal{N}_1$.
Hybrid ILS with LS1$'$	hybrid algorithm that produces a starting solution by solving MIPM and then enhances this solution by a local search that uses only the operator $\mathcal{N}_1'$.
Hybrid ILS with SLS	hybrid algorithm that produces a starting solution by solving MIPM and then enhances this solution by a sequential local search utilising LS1 and LS2.
Hybrid ILS with SLS$'$	hybrid algorithm that produces a starting solution by solving MIPM and then enhances this solution by a sequential local search utilising LS1$'$ and LS2$'$.
Multi-start ILS with LS1	application of the Algorithm 18 to five different initial solutions generated by the heuristic INITIAL and then enhances these solutions by a local search that uses only the operator $\mathcal{N}_1$.
Multi-start ILS with LS1$'$	application of the Algorithm 18 to five different initial solutions generated by the heuristic INITIAL and then enhances these solutions by a local search that uses only the operator $\mathcal{N}_1'$.
Multi-start ILS with SLS	application of the Algorithm 18 to five different initial solutions generated by the heuristic INITIAL and then enhances these solutions by a sequential local search utilising LS1 and LS2.
Multi-start ILS with SLS$'$	application of the Algorithm 18 to five different initial solutions generated by the heuristic INITIAL and then enhances these solutions by a sequential local search utilising LS1$'$ and LS2$'$.

hybrid ILS with SLS, hybrid ILS with SLS$'$, multi-start ILS with SLS, and multi-start ILS with SLS$'$. We will discuss the results of hybrid ILS first.

The hybrid ILS is implemented by solving the MIPM, which provides an input arrival plan to the iterated local search in Algorithm 13. Because of the faster evaluation of solutions in a neighborhood and the smaller neighborhood size, the versions of the hybrid iterated local search with the operators $\mathcal{N}_1$ and $\mathcal{N}_2$ are faster than their counterparts with the operators $\mathcal{N}_1'$ and $\mathcal{N}_2'$ often at a cost of inferior solution quality. The computational experiments took this into account and ran versions with the operators $\mathcal{N}_1$ and $\mathcal{N}_2$ with one hour limit on the permissible computation time, recorded for each such optimization procedure the average of the actual computation times for ten runs, and then set this recorded average time as the limit on the computation time for the version with the corresponding operators $\mathcal{N}_1'$ and/or $\mathcal{N}_2'$.

For each case, i.e. for each choice of the parameters α and β, Tables 3.12 and 3.13 present the average computation time in seconds (Time), the average value of the objective function

Table 3.12: Performance of hybrid ILS with LS1 and LS1′ .

Case	In. Obj.	Time	Hybrid ILS with LS1		Hybrid ILS with LS1′	
			Obj.	%Rel	Obj.	%Rel
1	206,060	2,685	189,487	20.89	**180,530**	**15.17**
2	77,764	864	71,715	14.54	**67,092**	**7.16**
3	59,178	581	55,624	14.38	**51,344**	**5.58**
4	55,368	932	50,780	10.95	**47,803**	**4.45**
5	48,447	805	42,416	5.54	**41,666**	**3.67**
6	36,245	472	32,828	6.62	**32,371**	**5.14**
7	23,624	817	**22,494**	**7.42**	22,948	9.59
8	10,753	483	10,752	1.49	10,752	1.49
9	6,725	355	6,724	0.27	6,724	0.27
Average	58,240	888	53,647	9.12	51,248	5.84

Table 3.13: Performance of hybrid ILS with SLS and SLS′.

Case	In. Obj.	Time	hybrid ILS with SLS		hybrid ILS with SLS′	
			Obj.	%Rel	Obj.	%Rel
1	206,060	3,600	187,992	19.94	**179,847**	**14.74**
2	77,764	2,482	70,845	13.15	**66,967**	**6.96**
3	59,178	1,640	55,100	13.31	**51,324**	**5.54**
4	55,368	2,281	51,459	12.43	**47,785**	**4.41**
5	48,447	2,496	42,193	4.98	**41,432**	**3.09**
6	36,245	1,634	32,968	7.08	**32,371**	**5.14**
7	23,624	1,413	**22,613**	**7.99**	22,948	9.59
8	10,753	1,191	10,752	1.49	10,752	1.49
9	6,725	1,078	6,724	0.27	6,724	0.27
Average	58,240	1,979	53,405	8.96	51,128	5.69

(Obj.), and the average relative gap (%Rel) obtained for ten runs of the hybrid ILS. The column In. Obj. displays the value of the objective function obtained by solving the MIPM. Table 3.12 indicates that the version with $\mathcal{N}_1'$ obtains better quality solutions in six of the nine cases, with the average relative gap improving from 9.12% to 5.84%. Table 3.13 also indicates the superior performance of the version with $\mathcal{N}_1'$ and $\mathcal{N}_2'$ in comparison with the version with $\mathcal{N}_1$ and $\mathcal{N}_2$.

The output of the multi-start ILS is the best output obtained by the application of the Algorithm 13 to five different input arrival plans generated by the heuristic INITIAL. These five applications of the Algorithm 13 constitute one run of the multi-start ILS. In the course of the computational experiments, the duration of each run of the multi-start ILS was limited by one hour. For each case, i.e. for each choice of the parameters α and β, Tables 3.14 and 3.15 present the average required time (Time), average value of the objective function (Obj.),

Table 3.14: Performance of multi-start ILS with LS1 and LS1′.

Case	multi-start ILS with LS1				multi-start ILS with LS1′			
	Time	In. Obj.	Obj.	%Rel	Time	In. Obj.	Obj.	%Rel
1	2,904	731,900	**200,646**	**28.01**	3,600	701,192	221,792	41.50
2	3,206	246,939	**76,713**	**22.52**	3,600	228,459	82,564	31.87
3	3,525	205,225	**56,717**	**16.63**	3,600	161,630	62,662	28.86
4	3,101	157,225	**52,039**	**13.70**	3,600	141,340	57,548	25.74
5	2,946	155,164	**46,623**	**16.01**	3,600	115,812	48,889	21.65
6	2,996	110,872	**35,309**	**14.68**	3,600	97,424	39,585	28.57
7	2,736	80,850	**23,722**	**13.29**	3,600	60,620	29,355	40.19
8	2,548	50,200	**11,109**	**4.86**	3,600	29,562	14,635	38.15
9	2,386	40,485	**6,917**	**3.14**	3,600	30,494	11,311	68.67
Average	2,928	197,651	56,644	14.76	3,600	174,059	63,149	36.13

Table 3.15: Performance of multi-start ILS with SLS and SLS′.

Case	multi-start ILS with SLS				multi-start ILS with SLS′			
	Time	In. Obj.	Obj.	%Rel	Time	In. Obj.	Obj.	%Rel
1	3,600	704,669	**207,738**	**32.53**	3,600	792,732	229,339	46.31
2	3,600	234,564	**76,228**	**21.75**	3,600	242,8870	80,410	28.43
3	3,600	176,729	**55,596**	**14.33**	3,600	149,560	61,766	27.01
4	3,600	165,457	**52,054**	**13.73**	3,600	139,882	56,795	24.09
5	3,600	141,360	**45,025**	**12.03**	3,600	112,386	50,796	26.39
6	3,600	112,397	**35,946**	**16.75**	3,600	109,447	43,065	39.87
7	3,600	78,051	**23,870**	**13.99**	3,600	59,009	29,110	39.02
8	3,600	57,798	**11,192**	**5.64**	3,600	30,803	15,081	42.35
9	3,600	35,437	**6,903**	**2.93**	3,600	28,582	9,512	41.85
Average	3,600	189,607	57,173	14.85	3,600	185,030	63,986	35.04

and average relative gap (%Rel) obtained for ten runs of the multi-start ILS. The column In. Obj. contains the average value of the objective function for the input arrival plans that resulted in the output of the multi-start ILS.

In Table 3.14, the multi-start ILS with LS1 is superior to the multi-start ILS with LS1′ in both time and solution quality. The same observation can be seen in Table 3.15, in which the multi-start ILS with SLS showing better performance over the version with SLS′. It is worth noting that although the multi-start ILS algorithms begin with poor initial solutions, significantly better results are achieved after the local search based enhancement procedures, with the best reported average improvement of 72% observed in the multi-start ILS with LS1.

In summary, the comparison of the eight optimization procedures indicates that the best

Table 3.16: Summary of the effects of the different neighborhoods.

Tables	$\mathcal{N}$	$\mathcal{N}'$	EQUAL	Winner
3.7	1	6	2	MIPM + ILS + $\mathcal{N}_1^{'}$
3.8	1	6	2	MIPM + ILS + $\mathcal{N}_1^{'}$+ $\mathcal{N}_2^{'}$
3.9	9	0	0	multi-start ILS + $\mathcal{N}_1$
3.10	9	0	0	multi-start ILS + $\mathcal{N}_1$ + $\mathcal{N}_2$

solution quality is obtained by the combination of the mixed integer linear program MIPM and the iterated local search with the operators $\mathcal{N}_1'$ and $\mathcal{N}_2'$. The comparison of the different neighborhoods is summarized in Table 3.16. The column Tables contains references to the tables that present the results of computational experiments. The column $\mathcal{N}$ contain the number of cases for which the neighborhood structure that does not require the subroutine TRANSFORMATION yields a better solution quality. The column $\mathcal{N}'$ contain the number of cases for which the neighborhood structure obtained, using the subroutine TRANSFORMATION, yields a better solution quality. The column EQUAL contains the number of cases when both neighborhood types produced the same solution quality.

Table 3.17 provides a summary of the results for all solution approaches. If all algorithms have the same time, the conclusion that hybrid ILS is best is still correct because of the following reasons: (1) the starting solution of hybrid ILS is already better than the best solution found by the other algorithms, and (2) the time required to find the starting solution is less than the time used by the other algorithms.

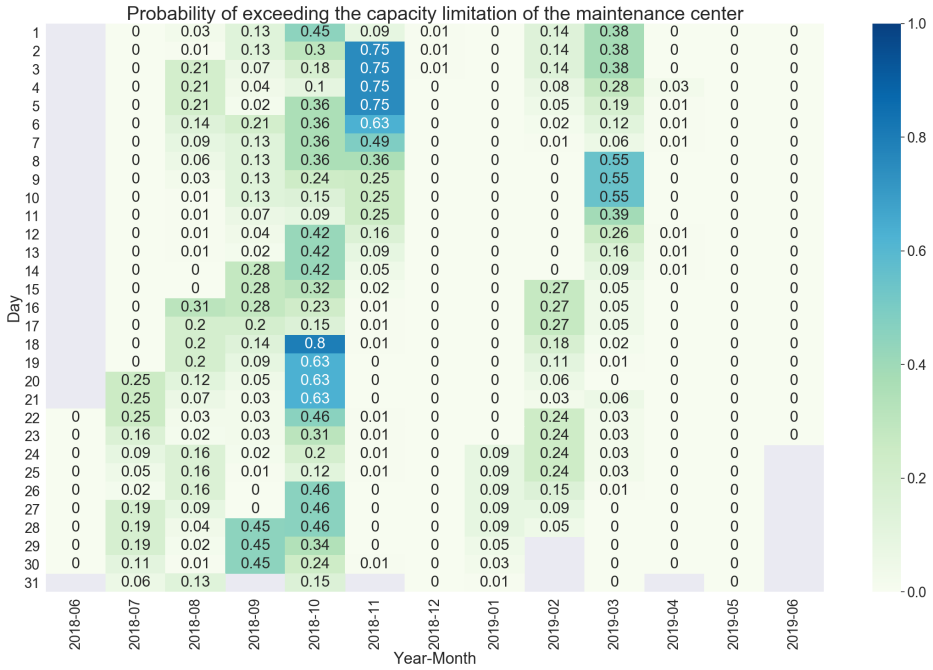
Table 3.17: Comparison between the performance of all solution approaches.

Case	GA	GA-M	Hybrid ILS				Multi-start ILS			
			LS1	LS1$^{'}$	SLS	SLS$^{'}$	LS1	LS1$^{'}$	SLS	SLS$^{'}$
1	341,962	233,634	189,487	180,530	187,992	**179,847**	200,646	221,792	207,738	229,339
2	132,345	87,171	71,715	67,092	70,845	**66,967**	76,713	82,564	76,228	80,410
3	99,165	68,758	55,624	51,344	55,100	**51,324**	56,717	62,662	55,596	61,766
4	92,363	64,514	50,780	47,803	51,459	**47,785**	52,039	57,548	52,054	56,795
5	72,170	58,163	42,416	41,666	42,193	**41,432**	46,623	48,889	45,025	50,796
6	59,084	41,351	32,828	**32,371**	32,968	**32,371**	35,309	39,585	35,946	43,065
7	35,868	29,228	**22,494**	22,948	22,613	22,948	23,722	29,355	23,870	29,110
8	27,959	11,979	**10,752**	**10,752**	**10,752**	**10,752**	11,109	14,635	11,192	15,081
9	22,235	7,772	**6,724**	**6,724**	**6,724**	**6,724**	6,917	11,311	6,903	9,512
Average	98,128	66,952	53,647	51,248	53,405	**51,128**	56,644	63,149	57,173	63,986

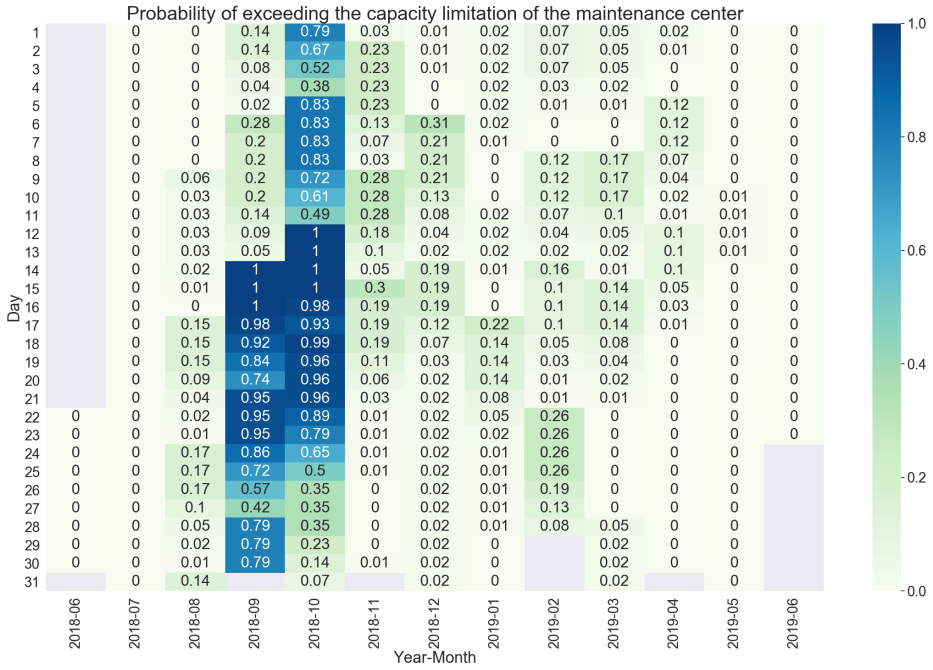
3.6.4 Visualization of Quality of Arrival Plan

During the negotiations between the rolling stock operator and the maintenance center, it is useful to have information about the risk of violating the center capacity which may occur as a result of the uncertain duration of maintenance. For this reason, a powerful visualization tool, based on the idea of heat map [177], is developed to provide insights into the risk over the planning horizon. Figures 3.6(a) and 3.6(b) show examples of the risk heat map associated with the arrival plans of cases 1 and 9, respectively. The horizontal axis indicates the month and year (for example, 2018-06 stands for June 2018), while the vertical axis indicates the day of the month. Each cell of the heat map corresponds to a particular day in the planning horizon, and the probability of violating the limit is clearly stated in each cell. The color intensity reflects the level of risk whereby the darker the color, the higher the risk.

The resulting heat map in Figures 3.6(a) and 3.6(b) show a trade-off example in which the constructed arrival plan must prioritize either the technological restrictions of the maintenance center or the arrival time windows. Figure 3.6(a) considers the perspective of the maintenance center who is more concerned about keeping the number of train-sets below the capacity of the maintenance center. As a result, the total penalty for the violation of the capacity limitation is insignificant and it can be seen on Figure 3.6(a) that there are few days which have high probability of exceeding the center capacity. However, the total penalty for the violation of time windows, $G_1(\sigma)$, is 23,487. On the other hand, the heat map in Figure 3.6(b) is associated with an arrival plan $\sigma^{'}$ that is constructed considering the perspective of the rolling stock operator, the main concern of whom is to satisfy the arrival time windows. In this case, the total penalty $G_1(\sigma^{'})$ is only 6,218 but the maintenance center has a high risk of violating the capacity, i.e. it is harder to have an efficient operational plan.



(a)



(b)

Figure 3.6: Heat maps displaying the probability of having more than 5 train-sets residing in the maintenance center for each day across the planning horizon of one year for cases (a) $\alpha = 1$, $\beta = 1000$; and (b) $\alpha = 1$, $\beta = 1$.

3.7 Summary

This chapter contributes to the existing body of literature on train maintenance by introducing a nonlinear programming problem that determines the arrival days of train-sets to a maintenance center, taking into account the uncertain duration of maintenance and the requirements specified by the rolling stock operator as well as the technological restrictions of the maintenance center.

A fast method of evaluation of the objective function for any feasible solution of the nonlinear program is presented together with a mixed integer programming relaxation based on Jensen's inequality. This relaxation provides a lower bound on the optimal value of the objective function of the nonlinear program and generates approximate solutions. Four different solution approaches were developed. The first is based on Genetic Algorithm. The second is an amalgamation of Genetic Algorithm for global search and Integer Programming for determining the optimal arrival dates when a sequence of arrival is fixed. The third is an Iterated Local Search algorithm. The fourth is a hybrid two-stage optimization procedure that combines Jensen's inequality based relaxation with either a local search subroutine or the presented ILS subroutine.

By means of computational experiments on real-world data, it is shown that the hybrid two-stage optimization procedure performed the best, followed by Iterated Local Search metaheuristic, followed by the amalgamation of GA and SAA. The GA metaheuristic was the least successful among the four solution approaches.

Further research should be focused on the operational level of the maintenance planning for a shorter planning horizon and more detailed information about maintenance procedures.

Chapter 4

Scheduling of Jobs Sharing multiple Resources

In this chapter, we study the scheduling problem where every job requires several types of resources. At every point in time, the capacity of resources is limited. When necessary, the capacity can be increased at a cost. Each job has a due date and the processing times of jobs are random variables with a known probability distribution. The considered problem is to determine a schedule that minimizes the total cost, which consists of the cost incurred due to the violation of resource limits and the total tardiness of jobs. A genetic algorithm enhanced by local search is proposed. The sample average approximation method is used to construct a confidence interval for the optimality gap of the obtained solutions. Computational study on the application of the sample average approximation method and genetic algorithm are presented. It is revealed that the proposed method is capable of providing high quality solutions to large instances in reasonable time.

4.1 Introduction

This study is concerned with the problem of scheduling the set of jobs $J = \{1, ..., \mathcal{J}\}$ where each job requires several types of resources. The planning horizon is discretised into a number of time periods of equal length indexed $1, ..., H$ and the set of all time periods is denoted by $T = \{1, ..., H\}$. The processing time of each job j is a discrete random variable p_j which assumes integer values and $0 \leq p_j^{\min} \leq p_j \leq p_j^{\max} \leq H$, for all $j \in J$, where $p_j^{\min}$ and $p_j^{\max}$ are the minimal and maximal possible processing times of job j, respectively. All random variables p_j are independently distributed.

All jobs are available for processing from period $t = 1$ and have to be executed without preemption, i.e. once the processing of a job starts, no interruption is allowed until its completion. A schedule $\mathbf{s}$ specifies for each job j the period s_j when its processing starts. Each job j is given a period d_j and it is desired to complete this job at period d_j or earlier. The tardiness of any job j is $\max\{s_j + p'_j - d_j - 1, 0\}$, where p'_j is a realization of p_j, i.e. an element of the set $\{p_j^{\min}, ..., p_j^{\max}\}$. A realization of the job's processing times is referred to as a scenario. For any schedule $\mathbf{s} = [s_1, ..., s_{\mathcal{J}}]$, the expected total tardiness is

$$G_1(\mathbf{s}) = G_1(s_1, ..., s_{\mathcal{J}}) = \sum_{j \in J} \sum_{p'_j \in \{p_j^{\min}, ..., p_j^{\max}\}} \Pr(p_j = p'_j) \max\{s_j + p'_j - d_j - 1, 0\} \qquad (4.1)$$

where $\Pr(p_j = p'_j)$ is the probability that the processing time of job j is p'_j.

The processing of each job requires $\mathcal{K}$ types of renewable resources. The set of resources is denoted by $K = \{1, ..., \mathcal{K}\}$. In each period t, it is desired that the total consumption of resource k should not exceed a certain non-negative integer R_{tk}, which will be referred to as the capacity of resource k in period t. If the capacity R_{tk} is exceeded, this attracts a certain penalty. During its processing, at each period, a job $j \in J$ consumes r_{jk} units of resource k, where r_{jk} is a non-negative integer. Given a schedule $\mathbf{s}$, for any $k \in K$ and any period t, denote by $C_{tk}(\mathbf{s})$ the total amount of resource k consumed in period t. Since all processing times are random variables, $C_{tk}(\mathbf{s})$ is a random variable. The penalty for the violation of the

limit, imposed by the capacity R_{tk}, is calculated using the three given parameters U_{tk}, α_{tk} and β_{tk}, where $\beta_{tk} > \alpha_{tk} > 0$ and U_{tk} is a positive integer. Each extra unit of resource k in the range $[R_{tk}, R_{tk} + U_{tk}]$ increases the penalty by α_{tk}, whereas each extra unit of resource k in addition to $R_{tk}+U_{tk}$ increases the penalty by β_{tk}. In other words, the penalty is calculated as follows:

$$f_{tk}(C_{tk}(\mathbf{s})) = \begin{cases} \alpha_{tk}(C_{tk}(\mathbf{s}) - R_{tk}) & \text{if } R_{tk} < C_{tk}(\mathbf{s}) \leq R_{tk} + U_{tk} \\ (\alpha_{tk} - \beta_{tk})U_{tk} + \beta_{tk}(C_{tk}(\mathbf{s}) - R_{tk}) & \text{if } C_{tk}(\mathbf{s}) > R_{tk} + U_{tk} \\ 0 & \text{otherwise} \end{cases} . \quad (4.2)$$

For any schedule $\mathbf{s}$, denote by $G_2(\mathbf{s})$ the expected total penalty for violating the resource capacity in $\mathbf{s}$, i.e.

$$G_2(\mathbf{s}) = G_2(s_1, ..., s_{\mathcal{J}}) = \sum_{t \in T} \sum_{k \in K} \mathbb{E}\big[f_{tk}(C_{tk}(\mathbf{s}))\big] \quad (4.3)$$

where $\mathbb{E}$ is the expectation operator. The goal is to minimize

$$G(\mathbf{s}) = G_1(\mathbf{s}) + G_2(\mathbf{s}). \quad (4.4)$$

The problem stated above can also be viewed as a two-stage stochastic program with simple recourse [30], where the first stage requires to determine the starting time of each job (variables s_j), whereas the second stage is concerned with the expansion of the capacity of each resource.

The considered scheduling problem has been motivated by a project with a rolling stock maintenance center. This maintenance center is responsible for the heavy maintenance of passenger trains. Each of these trains has a desired time window within which the maintenance of this train should commence. This time window is determined by the rolling stock operator, and is based on the validity period of the heavy maintenance that was previously performed. Due to the limited capacity at the maintenance center, it may not be possible for all the trains to arrive at the center within the desired time windows. If the capacity is exceeded, this attracts a certain penalty. The dwell time of the trains at the maintenance

center is uncertain at the time a decision must be made. This is because the number of maintenance activities that need to be performed on a train vary depending on the actual condition when it arrives at the center. Research on the planning of rolling stock maintenance, undertaken by the authors of this study, has been published in [71], [70] and [72], where Genetic Algorithm was used in [71] and [70].

The similarity between the considered problem with the maintenance planning problem presented in Chapter 3 is that temporary resource capacity expansion for a penalty is allowed and the processing time (duration) of jobs (train-sets) is uncertain. However, the considered problem has two distinguished features: (1) the processing of each job requires several types of resources; (2) the penalty for capacity violation is calculated based on not only the capacity of resource but also an upper bound on the resource expansion.

The remainder of this chapter is organized as follows. Section 4.2 presents a mixed integer linear programming formulation of the considered problem, and an efficient algorithm for computing the value of the objective function. The sample average approximation approach for assessing solution quality is described in Section 4.3. In section 4.4, a genetic algorithm enhanced by local search is proposed for solving the stochastic programming problem. This is followed in Section 4.5 by the results of computational experiments. Conclusions and directions for further research are given in Section 4.6.

4.2 Mixed integer linear programming formulation

4.2.1 Mixed integer linear program

In order to rewrite the objective function in a more convenient form, let Ω denote the set of all scenarios, π_ω denote the probability of a scenario $\omega \in \Omega$, and $(p_1^\omega, ..., p_\mathcal{J}^\omega)$ denote the realization of processing times in scenario ω. Also, let $y_{tk\omega}$ and $o_{tk\omega}$ be the variables corresponding to the expansion of resource k in period t in the range $[R_{tk}, R_{tk} + U_{tk}]$ and in

addition to $R_{tk} + U_{tk}$, respectively, given the realization of processing times under a scenario $\omega \in \Omega$. The objective function can be written as

$$G_1(\mathbf{s}) + G_2(\mathbf{s}) = \sum_{\omega \in \Omega} \pi_\omega \left\{ \sum_{j \in J} \max\{s_j + p_j^\omega - d_j - 1, 0\} + \sum_{t \in T} \sum_{k \in K} (\alpha_{tk} y_{tk\omega} + \beta_{tk} o_{tk\omega}) \right\}, \quad (4.5)$$

To guarantee job j can complete within the planning horizon under all scenarios, the latest time to start j is restricted to $t_j^{\max} = H - p_j^{\max} + 1$. Therefore, job j can start in any periods in $T_j = \{1, ..., t_j^{\max}\}$. For any job j, any period t, and any scenario ω, let

$$S_{jt}^\omega = \{\tau \mid \tau + p_j^\omega - 1 \geq t, \tau \leq t\} \cap \ T_j \quad (4.6)$$

denote the set of starting times which makes job j to be processed during time period t. Then, the starting time for each job j can be determined by

$$s_j = \sum_{t \in T_j} t x_{jt}$$

where variables $x_{jt} \in \{0, 1\}$ are obtained by solving the problem (4.7) - (4.12). We summarise the notation for this chapter as follows:

Sets:

$J = \{1, ..., \mathcal{J}\}$: set of jobs, indexed by j;

$T = \{1, ..., H\}$: set of time periods, indexed by t;

$K = \{1, ..., \mathcal{K}\}$: set of renewable resources, indexed by k;

Ω: set of scenarios;

S_{jt}^ω: set of starting times which makes job $j \in J$ to be processed during time period $t \in T$ under scenario $\omega \in \Omega$;

$T_j = \{1, ..., t_j^{\max}\}$: set of allowed starting times of job $j \in J$;

Parameters:

$t_j^{\max}$: latest time to start job $j \in J$;

p_j: processing time of job $j \in J$;

p_j^ω: realization of processing time of job $j \in J$ in scenario $\omega \in \Omega$;

π_ω: probability of scenario $\omega \in \Omega$;

$p_j^{\min}$: minimal possible processing times of job $j \in J$;

$p_j^{\max}$: maximal possible processing times of job $j \in J$;

d_j: due date of job $j \in J$;

R_{tk}: capacity of resource $k \in K$ in period $t \in T$;

U_{tk}: upper limit on temporary expansion of resource $k \in K$ in period $t \in T$;

r_{jk}: amount of resource $k \in K$ consumed by job $j \in J$ at each time period;

α_{tk}: penalty per extra unit of resource $k \in K$ in the range $[R_{tk}, R_{tk} + U_{tk}]$;

β_{tk}: penalty per unit of resource $k \in K$ exceeding $R_{tk} + U_{tk}$;

Decision Variables:

s_j: starting time for job $j \in J$;

$y_{tk\omega}$: amount of expansion of resource $k \in K$ in period $t \in T$ in the range $[R_{tk}, R_{tk} + U_{tk}]$;

$o_{tk\omega}$: amount of expansion of resource $k \in K$ in period $t \in T$ beyond $R_{tk} + U_{tk}$;

$x_{jt} \in \{0, 1\}$: 1 if job $j \in J$ starts in period $t \in T_j$, or 0 otherwise.

The considered scheduling problem can be formulated as follows.

$$\text{(MILP)}\ z^* = \min \sum_{\omega \in \Omega} \pi_\omega \left\{ \sum_{j \in J} \sum_{t \in T_j} \max\{t + p_j^\omega - d_j - 1, 0\}\ x_{jt} + \sum_{t \in T} \sum_{k \in K} (\alpha_{tk} y_{tk\omega} + \beta_{tk} o_{tk\omega}) \right\} \tag{4.7}$$

$$\text{s.t.} \sum_{t \in T_j} x_{jt} = 1, \qquad j \in J \tag{4.8}$$

$$\sum_{j \in J} \sum_{s \in S_{jt}^\omega} r_{jk} x_{js} - y_{tk\omega} - o_{tk\omega} \leq R_{tk},\ \omega \in \Omega,\ t \in T,\ k \in K \tag{4.9}$$

$$0 \leq y_{tk\omega} \leq U_{tk}, \qquad \omega \in \Omega,\ t \in T,\ k \in K \tag{4.10}$$

$$o_{tk\omega} \geq 0, \qquad \omega \in \Omega,\ t \in T,\ k \in K \tag{4.11}$$

$$x_{jt} \in \{0, 1\}, \qquad j \in J,\ t \in T_j \tag{4.12}$$

The objective function (4.7) is a weighted sum of two components: the expected total tardiness and the expected total penalty for violating the capacity, which we wish to minimize. Constraint (4.8) ensures that all jobs are completed by the end of planning horizon. Constraint (4.9) calculates the additional units of resource k required beyond the capacity R_{tk} in period t under scenario ω. Constraints (4.10) - (4.11) describes the domain for $y_{tk\omega}$ and $o_{tk\omega}$, respectively. Constraint (4.12) states the integrality restriction on the decision variable x_{jt}.

4.2.2 Evaluation of the objective function

In this section, we discuss the evaluation of the objective function (4.4). We first present a method to find, for a given schedule, the probability distribution of $C_{tk}(\mathbf{s})$ in (4.2), $k \in K,\ t \in T$. Then, we show how the objective function can be computed.

Let $y_j(u)$ denote the probability that the processing time of job j is u. For any time period t and any resource k, job j requires r_{jk} if it is being processed in period t. Given a schedule $\mathbf{s} = [s_1, ..., s_{\mathcal{J}}]$, the probability that job j is still being processed in period t, denoted by $e_{jt}(\mathbf{s})$, can be computed as follow.

$$e_{jt}(\mathbf{s}) = \begin{cases} \sum_{u=t-s_j+1}^{p_j^{\max}} y_j(u), & t \in \{s_j, ..., H\} \\ 0, & t \in \{1, ..., s_j - 1\} \end{cases}. \tag{4.13}$$

For any job $j \in J$, any period $t \in T$, and any resource $k \in K$, let

$$\mathscr{E}_{jtk}(\mathbf{s}) = \begin{cases} r_{jk} & \text{with probability } e_{jt}(\mathbf{s}) \\ 0 & \text{with probability } 1 - e_{jt}(\mathbf{s}) \end{cases}. \tag{4.14}$$

The total resource consumption of resource k in period t, resulting from the schedule $\mathbf{s}$, is the sum $C_{tk}(\mathbf{s}) = \sum_{j \in J} \mathscr{E}_{jtk}(\mathbf{s})$. For a specific period t, the random variables $\mathscr{E}_{jtk}(\mathbf{s})$ are independently distributed. Therefore, $C_{tk}(\mathbf{s})$ is a random variable that follows the Generalized Poisson-Binomial (GPB) distribution [188].

In the following, we propose to compute $C_{tk}(\mathbf{s}), k \in K,\ t \in T$ by means of convolutions. Such a method was presented in [72] to calculate the exact distribution of the number of trains residing at a maintenance center on a particularly day, which is a random variable that follows Poisson Binomial distribution. We extend this method to the case of Generalized Poisson Binomial distributed random variables. The method is outlined in Algorithm 15. To simplify notation, we suppress dependence on the given schedule $\mathbf{s}$ in Algorithm 15, and simply use e_{jt} instead of $e_{jt}(\mathbf{s})$, $\mathscr{E}_{jtk}$ instead of $E_{jtk}(\mathbf{s})$ and C_{tk} instead of $C_{tk}(\mathbf{s})$. Furthermore, we apply Algorithm 15 for each period t and each resource k. If it is clear which period and resource are considered, the subscript t and k can be dropped and the notation e_j, $\mathscr{E}_j$, and C can be used instead of e_{jt}, $\mathscr{E}_{jtk}$, and C_{tk}, respectively.

Algorithm 15 Direct convolution

Input: The set of two-point random variable $\mathscr{E}_j$, which takes the value 0 with probability $1 - e_j$, and r_j with probability e_j, $j \in \{1, ..., \mathcal{J}\}$
Output: The probability mass function of the sum $C = \sum_{j=1}^{\mathcal{J}} \mathscr{E}_j$
procedure
    $\Pr(C^1 = 0) = 1 - e_1$, $\Pr(C^1 = r_1) = e_1$
    $\Pr(C^1 = c) = 0$, for $c = 1, ..., r_1 - 1$
    Set $\ell = 1$
    for j from 2 to $\mathcal{J}$ **do**
        $\Pr(C^{\ell+1} = 0) = (1 - e_j) \cdot \Pr(C^\ell = 0)$
        $R = \sum_{i=1}^{j} r_i$
        for i from 1 to $R - 1$ **do**
            $\Pr(C^{\ell+1} = i) = e_j \cdot \Pr(C^\ell = i - r_j) + (1 - e_j) \cdot \Pr(C^\ell = i)$
        end for
        $\Pr(C^{\ell+1} = R) = e_j \cdot \Pr(C^\ell = R - r_j)$
        Set $\ell = \ell + 1$
    end for
    $\Pr(C = c) = \Pr(C^\ell = c)$, for $c = 0, ..., \sum_{j=1}^{N} r_j$
    return the probability mass function of C
end procedure

As a result, let $C_k^{\max} = \sum_{j \in J} r_{jk}$, the objective function (4.4) can be computed as

$$\begin{aligned} G_1(\mathbf{s}) + G_2(\mathbf{s}) = G_1(\mathbf{s}) & \\ + \sum_{t \in T} \sum_{k \in K} \sum_{c=R_{tk}+1}^{R_{tk}+U_{tk}} & \alpha_{tk}(c - R_{tk}) \Pr(C_{tk}(\mathbf{s}) = c) \\ + \sum_{t \in T} \sum_{k \in K} \sum_{c=R_{tk}+U_{tk}+1}^{C_k^{\max}} & \beta_{tk}(c - R_{kt} - U_{tk}) \Pr(C_{tk}(\mathbf{s}) = c) \end{aligned} \tag{4.15}$$

4.3 Sample Average Approximation

The Sample Average Approximation (SAA), as its name suggests, is an approach of replacing the original problem with its sampling approximation. In this section, we use the SAA approach to obtain a statistical estimate for a lower bound on the optimal value z^* of the objective function for the considered stochastic optimization problem (also called the true problem). Then, the optimality gap and statistical confidence intervals on the quality of the approximate solutions are constructed.

The implementation of SAA is as follows. Given a sample $\omega^1, \omega^2, ..., \omega^N$ of N scenarios, we can estimate the expected total tardiness in (4.1) and the expected total penalty for capacity violation in (4.3) by the average total tardiness and average total penalty over all scenarios, respectively. The resulting SAA problem is a large mixed integer program. The SAA problem is given below:

$$\begin{aligned} z_N^* = \min \frac{1}{N} \sum_{i=1}^{N} \Bigg\{ & \sum_{j \in J} \sum_{t \in T_j} \max\{t + p_j^{\omega^i} - d_j - 1, 0\}\ x_{jt} \\ & + \sum_{t \in T} \sum_{k \in K} (\alpha_{tk} y_{tk\omega^i} + \beta_{tk} o_{tk\omega^i}) \Bigg\} \end{aligned} \tag{4.16}$$

subject to (4.8), (4.12)

$$\sum_{j\in J}\sum_{s\in S_{jt}^{\omega^i}} r_{jk}x_{js} - y_{tk\omega^i} - o_{tk\omega^i} \leq R_{tk},\ 1 \leq i \leq N,\ t \in T,\ k \in K \tag{4.17}$$

$$0 \leq y_{tk\omega^i} \leq U_{tk},\ 1 \leq i \leq N,\ t \in T,\ k \in K \tag{4.18}$$

$$o_{tk\omega^i} \geq 0, \qquad 1 \leq i \leq N,\ t \in T,\ k \in K \tag{4.19}$$

Let $\mathbf{x}$ be a vector of decision variables $x_{jt},\ \forall j \in J,\ \forall t \in T_j$. The SAA method provides a mean to obtain estimate of lower bound and the optimality gap. It consists of generating M independent random samples, each of size N, and solving the resulting SAA problems. The optimal objective value is denoted by $z_N^m,\ m = 1, ..., M$, and the optimal solution is denoted by $\mathbf{x}_N^m$. The average of the optimal objective values of the M SAA problems

$$\bar{z}_N = \frac{1}{M}\sum_{m=1}^{M} z_N^m \tag{4.20}$$

is a statistical estimate for a lower bound on z^* ([109], [127]). The sample variance can be estimated by

$$\sigma^2_{z_N^*} = \frac{1}{(M-1)}\sum_{m=1}^{M}(z_N^m - \bar{z}_N)^2 \tag{4.21}$$

Given a solution $\hat{\mathbf{x}}$, it is clear that the objective value $G(\hat{\mathbf{x}})$ is an upper bound for z^*. This upper bound can be computed as described in Section 4.2.2. The quality of the solution $\hat{\mathbf{x}}$ can be determined by computing the optimality gap estimate

$$G(\hat{\mathbf{x}}) - \bar{z}_N, \tag{4.22}$$

The solution $\hat{\mathbf{x}}$ can be obtained as $\hat{\mathbf{x}} \in \arg\min\{G(\mathbf{x}_N^m) : m = 1, ..., M\}$ or from the method proposed in Section 4.4. Given a small non-negative number α, let $t_{M,\alpha}$ denote the (1-α) quantile of the Student's t-distribution [103, 98] with M degrees of freedom. we use

$$P\left(G(\hat{\mathbf{x}}) - z^* \leq G(\hat{\mathbf{x}}) - \bar{z}_N + \frac{t_{M-1,\alpha}\ \sigma_{z_N^*}}{\sqrt{M}}\right) \approx 1 - \alpha, \tag{4.23}$$

to construct the one-sided confidence interval of the level $(1-\alpha)$ for the optimality gap at $\hat{\mathbf{x}}$. That is,

$$\left[\ 0, G(\hat{\mathbf{x}}) - \bar{z}_N + \frac{t_{M-1,\alpha}\ \sigma_{z_N^*}}{\sqrt{M}}\ \right] \tag{4.24}$$

Instead of developing a confidence interval for the optimality gap by computing $G(\hat{\mathbf{x}})$, we may develop a confidence interval for the optimality gap by the upper-bound estimator. An estimate of the upper bound for z^* can be obtained by evaluating the solution $\hat{\mathbf{x}}$ using a sample $\omega^1, \omega^2, ..., \omega^{N'}$ of N' scenarios, where $N' > N$ [96]. Let $\hat{z}_{N'}(\hat{\mathbf{x}})$ denote the standard sample mean estimator of $G(\hat{\mathbf{x}})$ and $\sigma^2_{\hat{z}_{N'}(\hat{\mathbf{x}})}$ denote the standard sample variance estimator. An approximate $(1-2\alpha)$-level confidence interval for the optimality gap at $\hat{\mathbf{x}}$ is

$$\left[\ 0, \hat{z}_{N'}(\hat{\mathbf{x}}) - \bar{z}_N + \frac{t_{M-1,\alpha}\ \sigma_{z_N^*}}{\sqrt{M}} + \frac{t_{N'-1,\alpha}\ \sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}}{\sqrt{N'}}\ \right] \tag{4.25}$$

The above procedure for determining solution quality in stochastic programs was suggested in [127] and developed in [109].

4.4 Hybrid Genetic Algorithm

The mixed integer linear program, presented in Section 4.2, can only be solved within reasonable computation time for small instances. For large instances of the considered problem, a Genetic Algorithm enhanced by local search is developed. We refer to the proposed method as the Hybrid Genetic Algorithm (HGA). The motivation for using HGA comes from Chapter 3, whereby it was shown that the iterated local search is sensitive to the initial solution. On the other hand, GA's parallel search mechanism has the ability to maintain the diversity of population, making it less sensitive to the initial population.

Genetic Algorithm (GA) is a population-based search algorithm that can generates high-quality solution for many mathematical optimisation problems. GA has been successfully used in many different applications of stochastic optimization as reported in the literature. For example, [37] propose a two-stage GA for solving a stochastic parallel machine scheduling problem; [26] propose a biased random-key GA for the two-stage capacitated facility location problem; [189] develop a GA for solving large-scale instances of the two-stage stochastic programming for single yard crane scheduling problem. In a recent review paper, [69] report that out of 100 papers about flow-shop scheduling problems under uncertainty published from 2001 to 2016, Genetic Algorithm was the most used metaheuristics method and constituted the largest proportion (42%) of the total study papers (53 papers).

Our hybrid GA utilises genetic algorithm for global search and an efficient local search method for intensification purpose. Algorithm 16 presents the pseudocode for the hybrid GA. The input is comprised of the population size (P), crossover probability (λ_c), mutation probability (λ_m), and maximum number of generations ($\text{GEN}_{\max}$). First, the initial population, consisting of P chromosomes, is created randomly (line 3). Then, the main loop (lines 4 to 11) performs the half-uniform crossover, uniform mutation, and local search on the current population until the maximum number of generations ($\text{GEN}_{\max}$) is reached. The inner loop (lines 5 to 9) uses the binary tournament operator [134] to pick two parents for reproduction. A new offspring is created by applying the half-uniform crossover operator on the two parents with probability λ_c, and applying the uniform mutation operator according to the probability λ_m. The population of the next generation is formed by applying the local search procedure on the children chromosomes (line 10), where it takes a chromosome μ as an input and returns a chromosome in the neighborhood of μ with the smallest value of the objective function.

Algorithm 16 Hybrid genetic algorithm

1: **Input**: population size (P), crossover probability (λ_c), mutation probability (λ_m), termination condition ($\text{GEN}_{\max}$)
2: **procedure**
3: Initialize population consisting of P randomly generated chromosomes
4: **while** $\text{GEN}_{\max}$ is not satisfied **do**
5: **for** i from 1 to P **do**
6: Use the binary tournament to select two parents from the population
7: Create a new offspring by applying the half-uniform crossover with probability λ_c on the selected parents
8: With probability λ_m, apply the uniform mutation operator.
9: **end for**
10: Use local search to educate each chromosome in the population
11: **end while**
12: **return** the best solution found
13: **end procedure**

4.4.1 Representation of chromosome and definition of fitness function

In GA, the chromosome representation of a solution is important so that it is not only susceptible of the required genetic operators but also fully characterize the solution. In our implementation of GA, we have chosen the solution-based representation, in which a schedule **s** of jobs is directly represented by a chromosome. The rationale behind this chromosome representation is that no further decoding procedure is needed since each gene of a chromosome corresponds to the starting period of the job. Solution-based representation is widely used in the permutation flow shop scheduling problem [166], and the job-shop problem [180] where the chromosome denotes the order in which jobs are processed. In the remainder of this study, chromosome, solution, and schedule have the same meaning. The three terms are used interchangeably. The fitness of a chromosome is calculated as the inverse of the objective function value, which is computed as described in Section 4.2.2. The chromosome with smaller value of the objective function will have higher fitness.

4.4.2 Parent selection and crossover

The purpose of the selection operator is to decide which solutions will be selected from the population to generate offspring. In our implementation of GA, we have chosen the binary tournament operator [134]. The binary tournament operator picks two randomly selected individuals from the population and the one with a better fitness is selected for reproduction. Two rounds of tournament are executed to get two parent chromosomes from which an offspring will be generated.

The purpose of the crossover operator is to take pair of chromosomes and combine them to produce an offspring. The selected parent chromosomes will undergo crossover according to the crossover probability λ_c. In our implementation of GA, we have chosen to use the half-uniform crossover operator [58, 63, 132]. The half-uniform crossover operator compares the genes from the two parent chromosomes, copies the matching genes, and places them in the same position in the offspring partial solution. Then, the Hamming distance, i.e. the number of non-matching genes, is calculated. The offspring inherits exactly half of the non-matching genes (at random) from the first parent, and the remaining from the second parent. Figure 4.1 shows an example of the half-uniform crossover operator. The half-uniform crossover is suitable for the solution based representation because it is able to promote the level of diversification that our HGA needs. The half-uniform crossover is much less likely than the traditional one- or two-point crossover to produce the same offspring twice from the same parents [58]. The half-uniform crossover with solution based representation had been used in [182] to solve the job scheduling problem in grid computing.

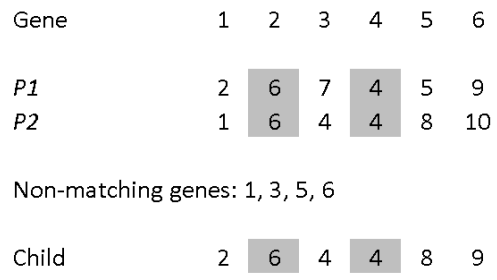


Figure 4.1: Example of the half-uniform crossover.

4.4.3 Mutation

The purpose of the mutation operator is to maintain genetic diversity in the population. In our implementation of GA, uniform mutation is applied at gene level to retain population diversity. Each gene in the child chromosome is assigned a random number sampled from the uniform distribution $U(0,1)$. If this random number is not greater than λ_m, the corresponding gene value is replaced by an integer randomly drawn from 1 to $H - p_j^{\max} + 1$, where j corresponds to the index of the chosen gene in the chromosome.

4.4.4 Local search method

The purpose of the local search is to find a better solution within the neighbourhood of a given solution. The idea to incorporate local search method within a Genetic Algorithm can be traced back at least to [138], and has been successfully applied to the permutation flowshop scheduling problem [166], parallel machines scheduling problem [159], and job shop scheduling problem [174], among others. The above mentioned studies, as well as most studies found in the literature, design and use the hybrid GA for deterministic problems. Under the assumption that parameters are known constants, it is easy to incorporate the local search method within the genetic algorithm framework as the evaluation of neighbour solutions does not require much time. However, for optimization problems where random parameters are present in the objective function, it is expensive to compute the objective function value for a solution. For this reason, studies that use genetic algorithm enhanced by local search for solving stochastic optimization problems is limited. The local search used in this study is designed with the aim that the neighbour solutions can be evaluated quickly, making it efficient enough to be used in the GA framework.

For a given schedule $\mathbf{s}$, the local search procedure defines a neighbourhood constituting a set of all solutions that can be reached by applying some search operator to $\mathbf{s}$. This study proposes the operator *shift*(τ, τ') which shifts the starting period from τ to τ' for a given

job. The procedure **Shift Search** is described in the following. In this procedure, a solution $\mathbf{s} = [s_1, ...s_{\mathcal{J}}]$ represents a chromosome, and $G(\mathbf{s})$ represents the objective function value of $\mathbf{s}$ (inverse of $G(\mathbf{s})$ gives the fitness of $\mathbf{s}$).

Procedure Shift Search (s, $G(\mathbf{s})$)

Step 1. Job list $JL \leftarrow \{1, 2, ..., , \mathcal{J}\}$

Step 2. If JL is not empty, choose a job j in JL, remove j from JL, and go to step 3. Otherwise, stop and return $\mathbf{s}$ and $G(\mathbf{s})$.

Step 3. Execute operator *shift*(τ, τ') on $\mathbf{s}$ for $\tau = s_j$ and $\tau' = \{1, ..., H - p_j^{\max} + 1\} \setminus \tau$. Compute the objective function value after each shift operator had been applied and choose the best one. If the objective function value of the best solution is better than $G(\mathbf{s})$, let $\mathbf{s}$ be the best solution and $G(\mathbf{s})$ be the objective function value of the best solution, then go to Step 2.

In step 3 of the above procedure, when a shift is performed, the chosen job j is first removed from the current solution $\mathbf{s}$, and the distribution of resource consumption is updated for all periods that are affected by this removal, i.e. $\forall t \in \{\tau, ..., \tau + p_j^{\max} - 1\}$. Let PMF be the resultant probability mass function. After assigning a new starting period τ' to job j, the distribution is again updated by convolving PMF with the distribution of job j for all the impacted periods, i.e. $\forall t \in \{\tau', ..., \tau' + p_j^{\max} - 1\}$. The fitness of a solution resulting from a shift can be computed quickly since the distribution of resource consumption is updated by considering one job at a time, leaving all other jobs and all unaffected periods unchanged.

In the following, we discuss the relationship between the features of the problem instances and the performance of the proposed local search method. For a solution $\mathbf{s}$ consisting of $\mathcal{J}$ jobs, as many as $\mathcal{J} \times (H - 1)$ shift operators can be applied to this solution, where H is the planning horizon. For each operator, the total number of operations to compute the value of the objective function is $\mathcal{K} \times H$, where $\mathcal{K}$ is the number of resources. Thus, the complexity of the procedure **Shift Search** is $\mathcal{O}(\mathcal{J} \times \mathcal{K} \times H^2)$. The total run time grows linearly with increases in the number of jobs and resources, but quadratically with increase in planning horizon.

However, it is noted that not all $\mathcal{J} \times (H-1)$ shift operators can improve the solution $\mathbf{s}$. This observation leads to the development of a dominance rule which reduces the size of the neighbourhood. The dominance rule is described as follow. Consider a solution $\mathbf{s}$ whose value of the objective function is $G(\mathbf{s})$. Let $\mathbf{s}'$ be a solution obtained by applying a specific operator *shift*(τ, τ') to a given job j. If the sum of $G_1(\mathbf{s}')$ and $G_2(\mathbf{s} \setminus j)$ is greater than or equal to $G(\mathbf{s})$, then the solution $\mathbf{s}'$ is dominated and the operation *shift*(τ, τ') can be ignored, since it is better to start job j on period τ than τ'. Applying the dominance rule described above to the procedure **Shift Search** can significantly reduce the run time of HGA because only the dominant solutions are considered during the local search process, i.e. the size of the search space is reduced. In our implementation of the hybrid GA, the dominance rule is used in the local search procedure.

4.5 Computational Results

In this section, we report the computational results of the proposed hybrid GA on 24 randomly generated problem instances. We first validate the performance of the hybrid GA against the exact solutions on the set of small problems. Then, we run the hybrid GA on the set of large problems, report the computational results, and present the 95% confidence interval for the optimality gap at the best solution returned by HGA. Finally, sensitivity analysis on the performance of the proposed hybrid GA and SAA methods is studied.

The proposed hybrid genetic algorithm was implemented in cython [24]. The testing system was a cluster with Intel Xeon Gold 6150 2.7GHz 8 cores CPU with 180GB RAM, running Red Hat Enterprise Linux. The computational facilities were provided by the UTS eResearch High Performance Computer Cluster.

4.5.1 Generation of test instances

The length of the planning horizon is 50 time periods. In this study, we consider 4 classes of instances: 20 jobs & 5 resources, 40 jobs & 5 resources, 60 jobs & 5 resources, and 80 jobs & 5 resources. For the random processing time of jobs (i.e. p_j), they are assumed to have two possible values with equal probability. The first value is generated from a uniform distribution on the integers $1, .., 50$. The second value is set to the first value plus ψ if the resulting sum is not larger than H. Otherwise the second value is set to the first value minus ψ. The value of ψ is chosen to be $\psi = 5$.

For job j, the parameter d_j is generated from a uniform distribution on the integers $1, ..., 10$, i.e. $d_j \sim U(1, 10)$. The amount of resource k consumed by job j (i.e. r_{jk}) is generated from a uniform distribution on the integers $1, ..., 5$, i.e. $r_{jk} \sim U(1, 5)$. It is assumed that the amount of resource available does not change significantly over the planning horizon, thus we set $R_{tk} = R_k, \forall t \in T$. For the same reason, we set $U_{tk} = U_k, \forall t \in T$. For resource k, the parameter R_k is generated from a uniform distribution on the integers between $\bar{p}\bar{r}_k\mathcal{J}/H$ and $\bar{p}\bar{r}_k\mathcal{J}/(0.6H)$, where $\bar{p} = 1/\mathcal{J}\sum_{j=1}^{\mathcal{J}} p_j$ and $\bar{r}_k = 1/\mathcal{J}\sum_{j=1}^{\mathcal{J}} r_{jk}$. For resource k, the parameter U_k is set to the greatest integer less than or equal to 10% of R_k, i.e. $U_k = \lceil 0.1R_k \rceil$.

For the penalty rates (i.e. α_{tk} and β_{tk}), it is assumed that they do not vary drastically over the planning horizon, thus we set $\alpha_{tk} = \alpha_k,\ \beta_{tk} = \beta_k, \forall t \in T$. The parameter α_k is generated from a uniform distribution on the integers $1, ..., 10$, i.e. $\alpha_k \sim U(1, 10)$, whereas β_k is calculated as $\max\{2\alpha_k, 10\}$. The setting on the above parameters follows the setting in the study by [92].

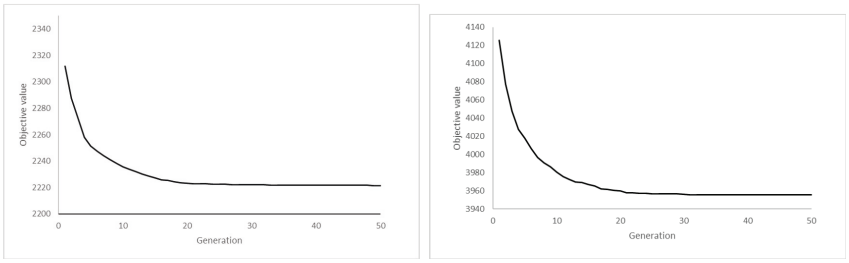
To test the performance of the proposed solution approaches on problem instances of varying number of scenarios, we consider two cases:

- Small problem instance where only ten jobs have two processing times, the remaining jobs have deterministic processing times. The total number of scenario is $2^{10} = 1024$ scenarios.

- Large problem instance where every job has two processing times resulting in a total of $2^{\mathcal{J}}$ scenarios, where $\mathcal{J}$ is number of jobs.

4.5.2 HGA parameter setting

For the hybrid GA, we set the values $P = 20$, $\lambda_c = 0.9$, $\lambda_m = 0.01$ for all the instances. The stopping criterion of the proposed hybrid GA were determined using a small pilot set of six problem instances included in the test instances described above. For each instance, 10 runs of the algorithm were performed. The results of the experiments is displayed in Figure 4.2. The horizontal axis indicates the generation, while the vertical axis indicates the value of the objective function corresponding to the best solution found. From Figure 4.2, the algorithm converges rapidly in less than 50 generations. Therefore, $\text{GEN}_{\max} = 50$ is used.



(a) (b)

Figure 4.2: Change in the objective function value with number of generations of the hybrid GA on the pilot set of (a) small problems and (b) large problems.

4.5.3 Comparisons between the proposed HGA and CPLEX on small problem instances

As has been discussed in Section 4.2.1, the stochastic programming problem can be formulated as the model MILP. Solving the model MILP by CPLEX gives an exact solution to the stochastic programming problem. Table 4.1 presents results of the proposed hybrid GA and the CPLEX solver for the small problem instances. For all the instances, CPLEX stopped af-

Table 4.1: Comparison of the exact method and the proposed hybrid GA on small instances.

Instance	CPLEX		Hybrid GA					
	Time	Obj	AvgTime	MinObj	AvgObj	MaxObj	Std	Gap(%)
20-5-1024-S1	912	1156	4	1156	1156	1156	0.00	0.00%
20-5-1024-S2	2264	1885	5	1885	1885	1885	0.00	0.00%
20-5-1024-S3	6363	585	8	585	588	591	1.85	0.51%
40-5-1024-S1	3244	1093	33	1093	1096	1105	4.69	0.27%
40-5-1024-S2	2333	1412	28	1412	1413	1417	1.80	0.07%
40-5-1024-S3	3386	2056	25	2056	2061	2068	4.40	0.24%
60-5-1024-S1	13455	1553	129	1557	1563	1574	6.35	0.64%
60-5-1024-S2	2496	1388	112	1391	1394	1397	1.66	0.43%
60-5-1024-S3	4298	4014	70	4014	4024	4043	9.01	0.25%
80-5-1024-S1	5449	1821	197	1822	1823	1825	0.92	0.11%
80-5-1024-S2	16106	1996	234	1999	2001	2005	2.12	0.25%
80-5-1024-S3	4921	2818	208	2819	2840	2878	18.01	0.78%

Notes: Instance '20-5-1024-S1' means 20 jobs, 5 resources, and 1024 scenarios;
'S1' means first instance for the jobs-resources-scenarios combination.

ter reaching a 5-hour limit or when an optimal solution was found. The column "Time" gives the solve time (in seconds) used by CPLEX to obtain the optimal solution (column "Obj"). For each instance, Table 4.1 also shows the average running time in seconds (AvgTime), the minimum, average, and maximum values of the objective function (MinObj, AvgObj, MaxObj), and the standard deviation (Std) obtained for ten runs of the proposed hybrid GA. Solution quality of the hybrid GA is measured by the percentage relative difference $\text{Gap}(\%) = (\text{AvgObj} - \text{Obj})/\text{Obj} \times 100$, and is reported under the column titled "Gap(%)".

The results in Table 4.1 show that the hybrid GA can find high quality solutions with low running time in comparison with the optimal solutions by CPLEX. When $\mathcal{J} \leq 40$, the algorithm obtains optimal solutions in all 6 instances (column "MinObj"). When $\mathcal{J} \geq 60$, the gaps between HGA and CPLEX increase but are not significant. The average percentage relative difference are about 0.44% and 0.38% for the instances with 60 jobs and 80 jobs, respectively. In terms of the running times of the methods, the proposed hybrid GA can solve all the instances with $\mathcal{J} \leq 40$ in less than 40 seconds. For the instances with 80 jobs, HGA

takes less than 4 minutes, whereas CPLEX requires as much as 1 hour to solve the problems to within 1% optimality, and significantly more time to prove optimality. This shows the limitation of CPLEX and the strength of the proposed hybrid GA for the considered problem. Furthermore, the running times of the proposed HGA do not vary between instances of the same jobs-resources-scenarios combination, and only increase with problem scale, whereas the solve time by CPLEX vary significantly between instances of the same jobs-resources-scenarios combination.

4.5.4 Performance evaluation of the proposed HGA on large problem instances

We further analyze the performance of the hybrid GA by running the algorithm on 12 large problem instances where every job has two processing times. Table 4.2 provides a summary of results. As mentioned in Section 4.3, solving the SAA problems (by CPLEX) repeatedly produces a statistical estimate for a lower bound on the optimal value of the objective function for the true problem. The parameters for solving the SAA problems are: $M = 30$ and $N = 50$. We report the statistical lower bound under the column titled '$\bar{z}_N$'. For all the instances, CPLEX stopped after reaching a 1-hour time limit or when an optimal solution was found. For the method SAA, we report the average solve time by CPLEX (in seconds) over 30 SAA problems under the column titled 'AvgTime'; and the standard deviation under the column titled 'Std'. For each SAA problem, the solution obtained from SAA is evaluated as described in Section 4.2.2. The average objective value of the solutions, over the 30 SAA problems, is reported under the column titled 'AvgObj'. As such, $\bar{z}_N$ is different from AvgObj. For the method HGA, we report the average running time in seconds (AvgTime), the average value of the objective function (AvgObj), and the standard deviation (Std) over all ten runs. For both methods, the deviation of the average objective value from $\bar{z}_N$ is reported under the columns titled "Gap(%)", and is calculated by $\text{Gap}(\%) = (\text{AvgObj} - \bar{z}_N)/\bar{z}_N \times 100$.

The results in Table 4.2 show that the proposed HGA outperform the SAA method. Over all

Table 4.2: Summary of results for hybrid GA on large instances.

Instance	$\bar{z}_N$	SAA				Hybrid GA			
		AvgTime	AvgObj	Std	Gap(%)	AvgTime	AvgObj	Std	Gap(%)
20-5-2^{20}-S1	854	18	869	4.44	1.88%	6	865	1.64	1.29%
20-5-2^{20}-S2	1322	17	1356	7.60	2.88%	6	1348	0.87	1.97%
20-5-2^{20}-S3	472	23	474	1.43	0.42%	9	473	0.95	0.21%
40-5-2^{40}-S1	4453	69	4593	33.66	3.14%	32	4541	11.18	1.98%
40-5-2^{40}-S2	7117	39	7294	25.96	2.52%	24	7244	14.55	1.78%
40-5-2^{40}-S3	2460	46	2520	18.24	2.52%	28	2492	2.73	1.30%
60-5-2^{60}-S1	4626	108	4832	25.62	4.45%	83	4770	15.40	3.11%
60-5-2^{60}-S2	2051	88	2060	10.87	0.83%	78	2051	1.23	0.00%
60-5-2^{60}-S3	3216	1440	3320	19.67	3.23%	95	3286	5.76	2.18%
80-5-2^{80}-S1	4982	105	5126	18.71	2.89%	174	5077	7.62	1.91%
80-5-2^{80}-S2	3266	3186	3359	16.57	2.85%	228	3323	6.44	1.75%
80-5-2^{80}-S3	3387	546	3507	17.01	3.54%	185	3470	9.94	2.45%

Notes: Instance '20-5-2^{20}-S1' means 20 jobs, 5 resources, and 2^{20} scenarios;
'S1' means first instance for the jobs-resources-scenarios combination.

the 12 problem instances, the hybrid GA not only obtains better solutions in all of them but also achieved more stable results across the different runs of the algorithm. The average gap to $\bar{z}_N$ is about 1.74% for the hybrid GA, whereas it is about 2.60% for the SAA method. The results suggest that the proposed hybrid GA is capable of obtaining high quality solutions for large instances of the considered problem. From the experiment, we observe that SAA with $N = 100$ produces superior solution quality than SAA with $N = 50$ at a cost of significantly longer running time. For the running time of the methods, the average time required by HGA was 79 seconds, and the increase in running time between the test cases with (20 jobs, 2^{20} scenarios) and with (80 jobs, 2^{80} scenarios) is moderate. On the other hand, although all the SAA problems can be solved to optimality, the average solve time of CPLEX was 474 seconds, which is about 6 times more than HGA.

Tables 4.3 and 4.4 present the 95% confidence interval (CI) for the optimality gap at $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the best solution found after ten runs of the hybrid GA for each instance. We report the test results for developing the CI based on $G(\hat{\mathbf{x}})$ according to (4.24) and based on the upper-bound estimator $\hat{z}_{N'}(\hat{\mathbf{x}})$ according to (4.25) (see Section 4.3). The upper-bound

Table 4.3: The 95% confidence interval (CI) for the optimality gap at $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the best solution obtained from the hybrid GA.

Instance	20-5-2^{20}-S1	20-5-2^{20}-S2	20-5-2^{20}-S3	40-5-2^{40}-S1	40-5-2^{40}-S2	40-5-2^{40}-S3
$\bar{z}_N$	852.83	1317.85	472.06	4453.38	7115.35	2457.71
$\sigma_{z_N^*}$	12.83	29.73	1.63	61.97	93.28	36.44
	CI construction using (4.24)					
$G(\hat{\mathbf{x}})$	863.75	1347.88	473.00	4527.02	7214.22	2490.35
$\frac{t_{M-1,0.05}\ \sigma_{z_N^*}}{\sqrt{M}}$	3.98	9.22	0.50	19.22	28.94	11.30
95% CI	[0, 14.90]	[0, 39.25]	[0, 1.45]	[0, 92.87]	[0, 127.81]	[0, 43.94]
	CI construction using (4.25)					
$\hat{z}_{N'}(\hat{\mathbf{x}})$	864.43	1357.54	481.00	4521.60	7210.75	2490.75
$\sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}$	128.12	235.70	3.91	481.66	602.93	253.79
$\frac{t_{M-1,0.025}\ \sigma_{z_N^*}}{\sqrt{M}}$	4.79	11.10	0.61	23.14	34.83	13.61
$\frac{t_{N'-1,0.025}\ \sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}}{\sqrt{N'}}$	2.51	4.62	0.08	9.45	11.83	4.98
95% CI	[0, 18.90]	[0, 55.42]	[0, 9.63]	[0, 100.81]	[0, 142.06]	[0, 51.63]

estimator $\hat{z}_{N'}(\hat{\mathbf{x}})$ is obtained using $N' = 10^4$. From Table 4.3, it can be observed that tighter confidence interval on the optimality gap can be obtained using the CI construction based on $G(\hat{\mathbf{x}})$ rather than $\hat{z}_{N'}(\hat{\mathbf{x}})$. Indeed, the latter yields confidence interval widths that are within 2.78% and 2.09% from the upper bound $G(\hat{x})$, roughly 1.68 and 1.12 times larger than that obtained from the former, for the instances with 20 and 40 jobs, respectively. This is expected since the random CI width in (4.25) consists of not only the sampling error from estimating the lower bound but also that from the upper-bound estimator. The same observation can be seen in Table 4.4, in which constructing the CI using (4.24) results in sufficiently tight confidence intervals.

4.5.5 Sensitivity analysis

The results in Table 4.2 indicate that instance 80-5-2^{80}-S2 is harder to solve than the other instances. The reason could be that there is a well-balanced share between the expected total tardiness of jobs and expected total penalty for violating the resource capacity, which results in both SAA and HGA spending substantial amount of time in an attempt to find

Table 4.4: The 95% confidence interval (CI) for the optimality gap at $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the best solution obtained from the hybrid GA (continue).

Instance	60-5-2^{20}-S1	60-5-2^{20}-S2	60-5-2^{20}-S3	80-5-2^{40}-S1	80-5-2^{40}-S2	80-5-2^{40}-S3
$\bar{z}_N$	4625.95	2042.61	3215.88	4982.28	3266.44	3386.94
$\sigma_{z_N^*}$	57.47	15.18	46.31	44.95	47.61	50.98
	CI construction using (4.24)					
$G(\hat{\mathbf{x}})$	4749.98	2050.58	3280.43	5068.31	3314.92	3460.76
$\frac{t_{M-1,0.05}\ \sigma_{z_N^*}}{\sqrt{M}}$	17.83	4.71	14.37	13.94	14.77	15.81
95% CI	[0, 141.86]	[0, 12.68]	[0, 78.91]	[0, 99.97]	[0, 63.25]	[0, 89.63]
	CI construction using (4.25)					
$\hat{z}_{N'}(\hat{\mathbf{x}})$	4753.09	2057.37	3277.23	5079.33	3334.72	3475.48
$\sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}$	505.62	122.74	292.91	389.73	284.11	342.30
$\frac{t_{M-1,0.025}\ \sigma_{z_N^*}}{\sqrt{M}}$	21.46	5.67	17.29	16.78	17.77	19.03
$\frac{t_{N'-1,0.025}\ \sigma_{\hat{z}_{N'}(\hat{\mathbf{x}})}}{\sqrt{N'}}$	9.92	2.41	5.75	7.65	5.57	6.72
95% CI	[0, 158.52]	[0, 22.84]	[0, 84.39]	[0, 121.48]	[0, 91.62]	[0, 114.29]

the solution that minimizes the total cost. In this section, using the large instance 80-5-2^{80}-S2, we first analyze the performance of SAA and HGA with the variation of two problem parameters: H and ψ. Because the time required by the proposed HGA to solve the instance is at most 30 minutes, we impose a 30-minute time limit for CPLEX. This ensures a fair comparison. Next, using the large instances, the performance of HGA is analyzed with the variation of two GA parameters: λ_m and λ_c.

Table 4.5 presents the analysis on the performance of SAA and HGA with $H \in \{50, 60, 70, 80, 90, 100\}$ when the remaining problem parameters stay unchanged. In this table, the average optimality gap reported by CPLEX across 10 runs is reported under the column titled "AvgMIP-gap(%)". It is observed that the required computation time by HGA increases when H increases. This is expected since increasing H will inevitably increase the number of moves in the local search procedure which is the most time consuming component in the proposed HGA. Indeed, the HGA took, on average, 6.67 times longer to solve instance 80-5-2^{80}-S2 with $H = 100$ than with $H = 50$. When looking at the average objective value when H is increased from 50 to 60, we note that the AvgObj is improved since the penalty incurred for exceeding the capacity of resources is reduced. As H continues to be increased from 70 to

Table 4.5: Sensitivity analysis on the performance of SAA and HGA with H for instance 80-5-2^{80}-S2.

Instance	H	SAA				Hybrid GA		
		AvgTime	AvgObj	Std	AvgMIP-gap(%)	AvgTime	AvgObj	Std
80-5-2^{80}-S2	50	1800	3360.70	19.17	0.60%	228	3323.29	6.44
	60	1800	2115.50	3.32	0.11%	430	2109.48	0.59
	70	1800	2108.32	2.73	0.19%	655	2104.69	0.57
	80	1800	2109.44	2.41	0.21%	906	2104.90	0.54
	90	1800	2107.40	2.05	0.18%	1244	2104.69	0.43
	100	1819	2108.45	2.20	0.24%	1521	2104.31	0.21

100, there is little difference in the AvgObj because resource capacity is always sufficient, and the cost incurred due to the violation of resource limits are dominated by the total tardiness of jobs. For SAA method, it is observed that CPLEX fails to solve the SAA problems to optimality before the 30-minute time limit is reached. The lowest average optimality gap reported by CPLEX was 0.11% when $H = 60$.

Table 4.6 presents the analysis on the performance of SAA and HGA with $\psi \in \{5, 10, 15, 20\}$ when the remaining problem parameters stay unchanged. It is observed that an increase in ψ leads to an increase in the objective value due to violation of the resource limits, but has little impact on the solution time of HGA. Also, increasing ψ makes the SAA problems become substantially easier to solve. This is due to the fact that the total tardiness of jobs is negligible when ψ is large. Indeed, the ratios of the expected total penalty for violating the resource capacity to the total tardiness of jobs were 0.48, 2.13, 3.47, and 4.21, respectively, for the four settings of ψ.

Table 4.7 presents the analysis on the performance of HGA using a combination of $\lambda_m \in \{0.01, 0.05\}$ and $\lambda_c \in \{0.85, 0.9, 0.95\}$ when $P = 20$ and $\text{GEN}_{\max} = 50$. In this table, the instances are grouped according to the number of jobs. The first four columns are as follows: the instance group (Group), the average objective value (AvgObj), the standard deviation (Std), the average time taken by HGA to terminate (AvgTime), across the 10 runs of the HGA with $\lambda_m = 0.01$ and $\lambda_c = 0.85$. In this table, all the percentage differences are relative

Table 4.6: Sensitivity analysis on the performance of SAA and HGA with ψ for instance 80-5-2^{80}-S2.

Instance	ψ	SAA				Hybrid GA		
		AvgTime	AvgObj	Std	AvgMIP-gap(%)	AvgTime	AvgObj	Std
80-5-2^{80}-S2	5	1800	3360.70	19.17	0.600%	228	3323	6.44
	10	1420	8002.33	24.79	0.170%	207	7884	14.34
	15	460	11211.83	88.86	0.008%	216	10990	14.79
	20	269	12384.08	108.30	0.009%	213	12198	0.00

Table 4.7: Sensitivity analysis on the performance of HGA with λ_m and λ_c for large instances, in terms of solution quality and time.

Group	$\lambda_m = 0.01$									$\lambda_m = 0.05$								
	$\lambda_c = 0.85$			$\lambda_c = 0.9$			$\lambda_c = 0.95$			$\lambda_c = 0.85$			$\lambda_c = 0.9$			$\lambda_c = 0.95$		
	AvgObj	Std	AvgTime	$\%_O$	$\%_S$	$\%_T$	$\%_O$	$\%_S$	$\%_T$	$\%_O$	$\%_S$	$\%_T$	$\%_O$	$\%_S$	$\%_T$	$\%_O$	$\%_S$	$\%_T$
20 jobs	896	1.29	7	-0.03%	-22%	1%	-0.01%	-12%	2%	0.01%	-11%	44%	-0.06%	-45%	46%	-0.05%	-36%	46%
40 jobs	4763	10.05	28	-0.12%	-12%	1%	-0.10%	-2%	2%	-0.10%	-18%	56%	-0.13%	-30%	61%	-0.12%	-32%	62%
60 jobs	3372	8.29	84	-0.11%	-21%	2%	-0.06%	-5%	2%	-0.11%	-25%	80%	-0.20%	-27%	81%	-0.17%	-16%	80%
80 jobs	3959	6.61	186	-0.11%	-17%	4%	-0.07%	-1%	5%	-0.12%	-16%	86%	-0.17%	-17%	89%	-0.14%	-17%	93%
Average	3247	6.56	76	-0.09%	-18%	2%	-0.06%	-5%	3%	-0.08%	-17%	66%	-0.14%	-30%	69%	-0.12%	-25%	70%

to the corresponding results obtained by the HGA with $\lambda_m = 0.01$ and $\lambda_c = 0.85$. It can be clearly seen that the solution quality improves at the cost of longer computation time when λ_m increases. This is essentially because more genes in the chromosomes would likely be perturbed when λ_m is large. As a result of the differences between the original chromosomes and the mutated chromosomes, more iterations of local search would likely be required to explore the search space. Indeed, the HGA with $\lambda_m = 0.05$ consistently obtains a better solution than the HGA with $\lambda_m = 0.01$. However, the former takes on average 1.7 times longer than the latter. The parameter λ_c controls how diversified the chromosomes in a population are. Based on the results in Table 4.7, increasing or decreasing λ_c relative to $\lambda_c = 0.9$ can slightly reduce the solution quality. Since the half-uniform crossover operator gives us a highly disruptive crossover [63], using a large crossover probability ($\lambda_c = 0.95$) may decrease the likelihood that better solutions will be kept in the population.

4.6 Summary

In this chapter, we examine the problem of scheduling jobs where each job requires several types of resources with uncertain job' processing times. A method for calculating the exact distributions of resource consumption resulting from a given schedule was presented. Knowing the distribution enables us to compute the value of the expected cost incurred from exceeding the resource capacity. A genetic algorithm enhanced by local search was then proposed to find a schedule that minimizes costs. The hybrid GA incorporates the "standard" implementation of the GA as the global search scheme and a simple but effective shift search procedure as the local search scheme. The computational results on small problem instances show that the proposed hybrid GA yields high quality solutions with low computation time. Although it is possible to solve these small test problems to optimality by using commercial MIP solvers, the running times grows rapidly as the number of scenarios increases. For large applications, computational results show that the hybrid GA outperforms the SAA strategy and that changes in ψ have negligible effect on the computation time with this method. However, the influence of planning horizon on the performance of hybrid GA is, due to the embedded local search procedure, more prominent.

Further studies should consider other modeling extensions. For example, one could include a metric to deal with severe uncertainty of job processing times. In particular, the model can be adjusted to determine the required capacity of the resource types to ensure that capacity will not be exceeded with a certain probability. Other metaheuristics can be developed or the hybrid GA proposed in this study can be adapted to solve this new problem.

Chapter 5

Planning of Grid Operation-based Outage Maintenance

This chapter considers the planning problem arising in the maintenance of a power distribution grid. Maintenance works require the corresponding parts of the grid to be shut down for the entire duration of maintenance which could range from one day to several weeks. The planning specifies the starting times of the required outages for maintenance and should take into account the constrained resources as well as the uncertainty involved in the maintenance works which is characterized by the risk values provided by the grid operator. The problem was presented by the French company Réseau de Transport d'Électricité for the 2020 ROADEF/EURO challenge. Several approaches were developed during the competition and all approaches are reported in this chapter. We evaluate our approaches on the benchmark instances proposed for the competition. It is reported that the iterated local search metaheuristic with self-adaptive perturbation and restart strategy, coupled with a Large Neighborhood Search performed the best and has won the 2nd place in the competition.

5.1 Introduction

This chapter is concerned with the grid operation-based maintenance planning problem introduced during the 2020 ROADEF/EURO challenge by the French company Réseau de Transport d'Électricité (also known as RTE). The challenge consists of four separate phases, i.e. sprint, qualification, semi-final, and final. The first set of instances (set A) was released at the beginning of the challenge (April 1, 2020) for the sprint and qualification phases. The second set of instances (set B) was published on January 15, 2021 for the semi-final phase. For the ranking of the qualified teams in the final phase, two sets of instances were used, namely sets C (published) and X (hidden). Set C was made available to the participants on April 6, 2021 while the hidden instances (set X) were published after the challenge ended.

RTE, Europe's largest electricity transmission system operator, is responsible for operating, maintaining and developing the electricity transmission system that spans over 105,000 kilometers of lines. The voltages on RTE's electricity network range from 63kV to 400kV [158]. Due to the extreme hazards involved when performing maintenance operations on the high-voltages lines, individual transmission lines have to be shut down for the duration of maintenance. Given that RTE has to handle hundreds of maintenance operations a year and that the maintenance of a transmission line is a long process, it is among the operator's highest priorities to carefully schedule the required outages due to maintenance. In the context of this chapter, maintenance work and intervention have the same meaning. The two terms are used interchangeably.

Interventions are carried out by some workforce which is split into teams (or resources), each of which has different sizes and skill sets. The skilled workers are not available during weekends and public holidays. For this reason, resources are not available all the time. Consequently, the duration of an intervention is variable and depends on the time when it starts. Furthermore, resources, e.g. equipment and materials, must be brought to the maintenance site when the intervention commences and removed when it finishes. For this reason, it is expected that the amount of resources required at the beginning and the end

of the intervention are higher. Therefore, the resource workload of an intervention is also time-dependent. For every time period, the total consumption of a resource is bounded from below and above.

Certain transmission lines are too close to each other and the corresponding interventions should not take place at the same time. This is because the system is unable to handle the electricity demand if an unexpected outage occurs on another close line during the interventions.

Because the transmission lines must remain switched off for the entire duration of maintenance which could range from one day to several weeks, this causes the electricity system to be weakened and implies a certain risk for RTE. For each intervention and each scenario of grid operation, RTE can compute the risk. According to the energy usage in France, the risk values are dependent on time, because it is less risky to perform interventions in summer (when the demand for electricity is low) than in winter.

The goal of the planning process is to determine a schedule that specifies the starting times of all interventions. The presented optimization procedures take into account the characteristics of the considered problem, including the limitation of resources, and the parameters provided by RTE which reflect the risk associated with maintenance works. RTE conducted studies and decided that the objective function is a weighted sum of two components: the average risk (which is expressed as a monetary cost) related to performing the interventions and the total cost for the deviation from the average risk.

The considered problem is similar to the problem presented in Chapter 3 in that they both have applications in the fields of maintenance planning. On the other hand, the similarity between the considered problem with the scheduling problem presented in Chapter 4 is that the processing of each job requires several types of resources. However, the considered problem is distinguished from them in a number of important ways. First, the resource capacity constraint must not be violated for a solution to be considered feasible. Second, the presence of exclusion constraints that enforces pairs of maintenance tasks cannot be concurrently

performed.

The contributions of this research can be summarized as follows: (1) a new mixed integer linear programming (MILP) formulation of the grid maintenance planning problem, that takes into consideration the risk associated with maintenance works; (2) a new MILP that is based on approximating the quantile term in the objective function; and (3) several solution approaches are developed and compared by means of computational experimentation using instances provided by the competition.

The remainder of this chapter is organized as follows. Section 5.2 presents a mixed integer linear programming formulation of the considered problem. In Section 5.3, we discuss an approximation of the quantile term in the objective function and derive a mixed integer linear programming formulation. In Sections 5.4 and 5.5, several heuristic and metaheuristic algorithms are developed to solve the considered problem. Section 5.6 presents computational comparisons for the various proposed algorithms using benchmark instances provided by the ROADEF/EURO challenge 2020. Finally, our conclusions are given in Section 5.7.

5.2 Mathematical programming formulation

In this section, we first introduce the notations and describe the objective function. Next, we present a mixed integer linear programming model to find a schedule of interventions, subject to available resources, non-overlapping restrictions between some pairs of intervention, and risk associated with the maintenance works.

5.2.1 Notations

Planning horizon: The schedule has to be established over a one-year period. The planning horizon is partitioned into intervals of equal length indexed $1, ..., H$ and the set of all time periods is denoted by $T = \{1, ..., H\}$. Depending on the required precision of the schedule,

the time step of a schedule can be either a day or a week.

Interventions: Consider a set of N interventions: $I = \{1, ..., N\}$, that have to be planned in the coming year. Each intervention i has a duration ($\Delta_{i,d}$) which assumes integer values and depends on the period d at which it starts. During its processing, at each period t, an intervention $i \in I$ consumes $r_{i,d}^{k,t}$ units of resource k if it starts in period d, where $r_{i,d}^{k,t}$ is a non-negative integer and will be referred to as the resource workload. For each intervention i, the earliest starting period is 1 and the latest is $t_i^{\max}$. Denote by $T_i = \{1, ..., t_i^{\max}\}$ the list of allowed starting periods of intervention i. For any intervention i, any period t, let

$$D_{i,t} = \{d : d + \Delta_{i,d} \geq t + 1, d \leq t\} \cap \ T_i$$

denote the set of starting periods which makes intervention i to be processed during period t. The restriction on which interventions can be carried out simultaneously is given by the set of exclusions, denoted by E. It is a set of triplets (i, j, t) designate that i and j cannot be concurrently performed in period t, where $i, j \in I$ and $t \in T$.

Resources: The processing of each intervention requires M types of resources with different sizes. The set of resources is denoted by $K = \{1, ..., M\}$. In each period t, the total consumption of resource $k \in K$ should be at least l_t^k and should not exceed u_t^k, where both l_t^k and u_t^k are non-negative integers.

5.2.2 Objective function

According to RTE, the objective function is a weighted sum of two components: the average risk (which is expressed as a monetary cost) related to performing the interventions, and the total cost for the deviation from the average risk. Both criteria are quantified in Euros.

For each intervention, the grid operator characterized the risk related to performing the intervention by some risk values. These values are positive real numbers and are given as input data. For each period t, we are given a set Ω_t of grid operation scenarios. Let $risk_{i,d}^{\omega,t}$

denote the risk value for period t, scenario ω, and intervention i when it starts at d. Also, let $x_{i,d} \in \{0,1\}$ to be 1 if $i \in I$ starts at $d \in T_i$, and 0 otherwise. Then, the first component of the objective function, denoted by Z_1, can be expressed as follows:

$$risk^{\omega,t} = \sum_{i \in I} \sum_{d \in D_{i,t}} risk_{i,d}^{\omega,t}\ x_{i,d}, \quad \omega \in \Omega_t,\ t \in T \tag{5.1}$$

$$\overline{risk^t} = \frac{1}{|\Omega_t|} \sum_{\omega \in \Omega_t} risk^{\omega,t}, \quad t \in T \tag{5.2}$$

$$Z_1 = \frac{1}{H} \sum_{t=1}^{H} \overline{risk^t} \tag{5.3}$$

Alternatively, one could choose to express Z_1 based on $T_i,\ i \in I$:

$$\overline{risk}_{i,d} = \sum_{t=d}^{d+\Delta_{i,d}-1} \frac{1}{|\Omega_t|} \sum_{\omega \in \Omega_t} risk_{i,d}^{\omega,t}, \quad i \in I,\ d \in T_i \tag{5.4}$$

$$Z_1 = \frac{1}{H} \sum_{i \in I} \sum_{d \in T_i} \overline{risk}_{i,d}\ x_{i,d} \tag{5.5}$$

For each period t, let $\mathfrak{R}^t = \{risk^{\omega,t}, \omega \in \Omega_t\}$, where $risk^{\omega,t}$ is a sum of risk values in scenario ω of the interventions that are in process in period t, and can be obtained according to (5.1). Then, the τ-quantile is given by

$$Q_\tau^t = \min\{q \in \mathbb{R} : \exists \mathcal{R} \subseteq \mathfrak{R}^t, |\mathcal{R}| \geq \tau * |\mathfrak{R}^t| \text{ and for } r \in \mathcal{R}, r \leq q\},\ t \in T \tag{5.6}$$

The second component of the objective function, denoted by Z_2, can be expressed as follows:

$$Z_2 = \frac{1}{H} \sum_{t=1}^{H} \max\{0, Q_\tau^t - \overline{risk^t}\} \tag{5.7}$$

Let $\alpha \in [0,1]$ denote the weight. The goal is to minimize

$$Z = \alpha Z_1 + (1-\alpha) Z_2 \tag{5.8}$$

We summarise the notation for this chapter as follows:

Sets:

$I = \{1, ..., N\}$: set of interventions, indexed by i;

$T = \{1, ..., H\}$: set of time periods, indexed by t and d;

$K = \{1, ..., M\}$: set of resources, indexed by k;

Ω_t: set of grid operation scenarios;

E: set of exclusions, where $(i, j, t) \in E$ for $i, j \in I$, $i \neq j$, $t \in T$;

$D_{i,t}$: set of starting periods which makes intervention $i \in I$ to be processed during period $t \in T$;

$T_i = \{1, ..., t_i^{\max}\}$: set of allowed starting periods of intervention $i \in I$;

$\mathcal{U}_t$: set of all maximal cliques of $G_t = (I, E)$ in period $t \in T$;

Parameters:

$t_i^{\max}$: starting time to start intervention $i \in I$;

$\Delta_{i,d}$: duration of intervention $i \in I$ if it starts in period $d \in T_i$;

l_t^k: minimum workload of resource $k \in K$ in period $t \in T$;

u_t^k: capacity of resource $kinK$ in period $t \in T$;

$r_{i,d}^{k,t}$: amount of resource $k \in K$ consumed by intervention $i \in I$ in period $t \in T$ if i starts in period $d \in T_i$;

$risk_{i,d}^{\omega,t}$: risk value for period $t \in T$, scenario $\omega \in \Omega_t$, and intervention $i \in I$ when it starts at $d \in T_i$;

M_t: a sufficiently large constant (big-M);

Decision Variables:

$q^{\omega,t} \in \{0, 1\}$: 1 if $risk^{\omega,t}$ is larger than Q_τ^t, or 0 otherwise;

$x_{i,d} \in \{0, 1\}$: 1 if intervention $i \in I$ starts at $d \in T_i$, or 0 otherwise;

Auxiliary Variables:

$\overline{risk}_{i,d}$: average risk value for intervention $i \in I$ when it starts at $d \in T_i$;

$risk^{\omega,t}$: sum of risk values in scenario ω of the interventions that are in process in period t;

$\overline{risk^t}$: average risk value in period $t \in T$;

Q_τ^t: the τ-quantile;

y_t: the positive difference between Q_τ^t and $\overline{risk^t}$.

5.2.3 Mixed integer linear programming formulation

We can eliminate (5.6) by introducing binary variables $q^{\omega,t}$, auxiliary variables y_t, together with Constraints (5.13) - (5.16) to give the mixed integer linear program MILP. The resulting formulation is given as below.

$$\text{(MILP)}\min : \alpha \times \frac{1}{H}\sum_{i\in I}\sum_{d\in T_i}\overline{risk}_{i,d}\ x_{i,d} + (1-\alpha)\times\frac{1}{H}\sum_{t=1}^{H} y_t \tag{5.9}$$

s.t. (5.1), (5.2), (5.4)

$$\sum_{d\in T_i} x_{i,d} = 1, \quad i \in I \tag{5.10}$$

$$l_t^k \le \sum_{i\in I}\sum_{d\in D_{i,t}} r_{i,d}^{k,t}\ x_{i,d} \le u_t^k, \quad k\in K,\ t\in T \tag{5.11}$$

$$\sum_{d\in D_{i,t}} x_{i,d} + \sum_{d\in D_{j,t}} x_{j,d} \le 1, \quad (i,j,t)\in E,\ D_{i,t}\neq\emptyset,\ D_{j,t}\neq\emptyset \tag{5.12}$$

$$risk^{\omega,t} - Q_\tau^t \le M_t q^{\omega,t}, \quad \omega\in\Omega_t,\ t\in T \tag{5.13}$$

$$\sum_{\omega\in\Omega_t}(1-q^{\omega,t}) \ge \lceil \tau\times|\Omega_t|\rceil, \quad t\in T \tag{5.14}$$

$$Q_\tau^t - \overline{risk^t} \le y_t, \quad t\in T \tag{5.15}$$

$$y_t \ge 0, \quad t\in T \tag{5.16}$$

$$q^{\omega,t}\in\{0,1\}, \quad \omega\in\Omega_t,\ t\in T \tag{5.17}$$

$$x_{i,d}\in\{0,1\}, \quad i\in I,\ d\in T_i \tag{5.18}$$

The objective function (5.9) is the weighted sum of two components: the average cost for performing the interventions and the total cost for the difference between the τ-quantile and the average risk. Constraint (5.10) ensures that each intervention must start within the planning horizon. Constraint (5.11) expresses the requirement that the total resource consumption must be between the limits l_t^k and u_t^k. Constraint (5.12) enforces that the

pairwise exclusive interventions cannot be concurrently performed. Constraints (5.13) - (5.16) define the τ-quantile, where M_t is a large number and $\lceil a \rceil$ denotes the smallest integer greater than or equal to a. In particulars, for any period t and scenario ω, if $risk^{\omega,t}$ is larger than Q^t_τ, Constraint (5.13) ensures that $q^{\omega,t}$ is 1, but arbitrary otherwise. By Constraint (5.14), the total number of scenarios that have risks below Q^t_τ must be at least $\lceil \tau \times |\Omega_t| \rceil$. The term $\max\{0, Q^t_\tau - \overline{risk^t}\}$ in (5.7) is substituted by a new decision variable (y_t) together with Constraints (5.15) and (5.16). Constraints (5.17) and (5.18) state the integrality restriction on the variables $q^{\omega,t}$ and $x_{i,d}$, respectively.

The exclusion representation in the form (5.12) is widely used in the scheduling domain [57]. However, as the planning horizon and number of interventions grow very large, the existence of a large number of exclusion constraints may render the solution to the MILP model inefficient. By observing the triplets in E, it is not uncommon to have several interventions that are pair-wise exclusive. The exclusion constraints (5.12) can be accordingly modified to exclude these exclusions together, which reduces the number of exclusion constraints significantly. Formally, the set of pairwise exclusive interventions can be represented as a maximal clique on a undirected *conflict graph* $G_t = (I, E)$, where each vertex corresponds to one intervention in I and each edge between vertices i and j, i.e. $(i, j, t) \in E$, corresponds to the pairwise conflict relationship between interventions i and j in period t. We denote the set of all maximal cliques of G_t by $\mathcal{U}_t = \{U : U \text{ is a maximal clique of } G_t\}$. With the cliques-based constraint, an alternative is to replace Constraint (5.12) with (5.19) below.

$$\sum_{i \in U} \sum_{d \in D_{i,t}} x_{i,d} \leq 1, \quad U \in \mathcal{U}_t, t \in T \tag{5.19}$$

Our experience with the instances proposed by the competition indicates that the resulting problem with cliques-based constraint (5.19) has fewer constraints.

Solving MILP is not an easy task due to the combinatorial structure of mixed integer programs. However, tightening the big-M constants for each constraint can make the formulation substantially stronger than if big-Ms are assigned arbitrarily large values. We propose

the below method to determine tightened but sufficiently large big-Ms, that is the risk and resources information.

Proposition 1. *Let* $risk_{i,d}^{t^*} = \max_{\omega \in \Omega_t} risk_{i,d}^{\omega,t},\ i \in I, t \in T, d \in D_{i,t}$ *is the largest value in the risk profile at time* t *for intervention* i *starting at* d*. Let* $risk_i^{t^*} = \max_{d \in D_{i,t}} risk_{i,d}^{t^*},\ i \in I, t \in T$ *is the largest risk value of intervention* i *at time* t *among all possible starting times. Let* N_t *be the maximum number of interventions that can be carried out simultaneously at* t *without violating the resource capacity. Consider any time* t*, if* $B_1, B_2, ..., B_N$ *is the sequence of the values of* $risk_i^{t^*}$ *which are listed in a non-increasing order, then* $M_t = \sum_{i=1}^{N_t} B_i$ *is a valid value of big-*M*s for MILP.*

5.3 Approximation of quantile term in objective function and iterative updating algorithm

Although the MILP formulation is compact, solving it presents a formidable computational challenge. During the development of solution methods for the qualification phase, we conducted experiments on the MILP formulation and observed that even for a small test instance with 12 scenarios (179 interventions, 9 resources, and 90 time periods), it cannot be solved to optimality by CPLEX in a 2-hour time limit.

Since the team rankings were determined based on the solution values obtained within fifteen minutes execution (with weight 0.8) and one hour and a half execution (with weight 0.2) per instance on the organizers' computer, it is critical that we develop an approximation of the the grid operation-based outage maintenance planning problem that enables it to be solved in a relatively short period of time (e.g. fifteen minutes). With this in mind, we propose a new mixed-integer linear programming relaxation, denoted as A-MILP. The key idea is to approximate the τ-quantile Q_τ^t by the sum of τ-quantile of the individual interventions' risk. The A-MILP can be solved to provide an initial feasible solution for the heuristic and metaheuristic approaches in this chapter.

For each intervention i that starts in d, and for any period t, denote by $Q^t_{i,d,\tau}$ the τ-quantile of the risk values in period t when i starts in d. The parameter $Q^t_{i,d,\tau}$ can be precalculated from the data. Then, for each period t, the approximation of the τ-quantile Q^t_τ is given by

$$\widehat{Q}^t_\tau = \sum_{i \in I} \sum_{d \in D_{i,t}} Q^t_{i,d,\tau}\, x_{i,d}, \qquad t \in T \tag{5.20}$$

A plot of the risk value versus scenario can help to visualize the relationship between Q^t_τ and $\widehat{Q}^t_\tau$. Consider the simple case with two interventions whose risk values are a linear function of scenarios (see Figure 5.1). The horizontal axis represents scenarios, and the vertical axis represents the risk values. The total risk values (green line) is the sum of the individual risk values for the ten scenarios. When both interventions have positive (negative) slopes, the τ-quantile of the total risk and the sum of the τ-quantile of the individual risk are the same, i.e. Q^t_τ and $\widehat{Q}^t_\tau (= Q^t_{1,\tau} + Q^t_{2,\tau})$ are the same. However, for the combination of interventions with both positive and negative slopes (see Figure 5.2), the value of $\widehat{Q}^t_\tau$ will likely be an overestimation or underestimation of Q^t_τ.

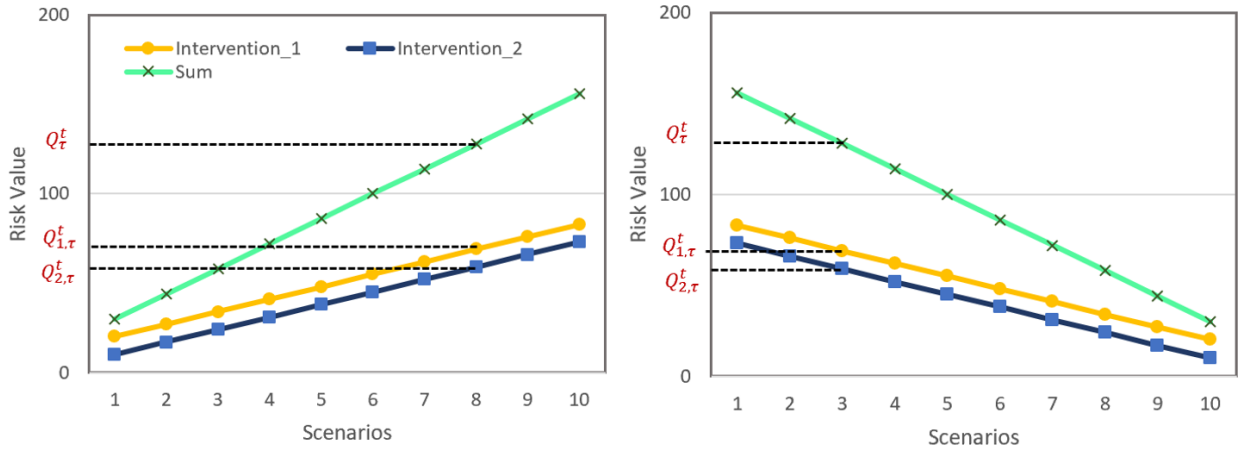


Figure 5.1: An example of the interventions having (left) positive and (right) negative slopes. Given a time t and $\tau = 0.8$, in the left figure, we have $Q^t_\tau = 128$ and $\widehat{Q}^t_\tau = Q^t_{1,\tau} + Q^t_{2,\tau} = 69 + 59 = 128$. In the right figure, we have $Q^t_\tau = 128$ and $\widehat{Q}^t_\tau = Q^t_{1,\tau} + Q^t_{2,\tau} = 69 + 59 = 128$.

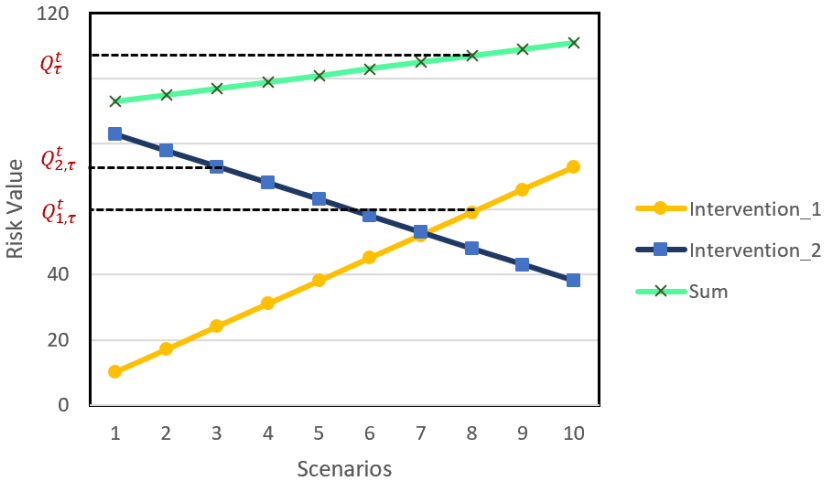


Figure 5.2: An example of the interventions having both positive and negative slopes. Given a time t and $\tau = 0.8$, we have $Q^t_\tau = 107$ but $\widehat{Q}^t_\tau = Q^t_{1,\tau} + Q^t_{2,\tau} = 59 + 73 = 132$.

With the above illustrations and discussions, consider a positive scaling factor $\beta_t, t \in T$, as the parameter compensating for the difference between Q^t_τ and $\widehat{Q}^t_\tau$. This leads to the following approximation of Z_2 in (5.8)

$$\frac{1}{H}\sum_{t=1}^{H} \max\{0, \beta_t \widehat{Q}^t_\tau - \overline{risk^t}\} \tag{5.21}$$

With this approximation and a linearisation of (5.21) in the same way as with (5.7), we introduce formulation A-MILP as follows:

$$\text{(A-MILP)}\ \min : \alpha \times \frac{1}{H}\sum_{i\in I}\sum_{d\in T_i} \overline{risk}_{i,d}\ x_{i,d} + (1-\alpha) \times \frac{1}{H}\sum_{t=1}^{H} y_t \tag{5.22}$$

$$\text{subject to } (5.1),\ (5.2),\ (5.4),\ (5.10)-(5.12),\ (5.20)$$

$$\beta_t \widehat{Q}^t_\tau - \overline{risk^t} \le y_t, \quad t \in T \tag{5.23}$$

$$y_t \ge 0, \quad t \in T \tag{5.24}$$

$$x_{i,d} \in \{0,1\}, \quad i \in I,\ d \in T_i \tag{5.25}$$

For problem with only one scenario, model A-MILP with $\beta_t = 1, t \in T$ is equivalent to the original model MILP. In other words, solving A-MILP to optimality leads to the optimal solution to the considered problem. For problem with more than one scenario, an optimal

solution to A-MILP given some vector $\beta = (\beta_1, ...\beta_H)$ is a feasible solution to the original model MILP. The advantage of having A-MILP is that optimal solution is significantly easier to find as the omission of binary variables $q^{\omega,t}$ and constraints (5.13) and (5.14) leads to a model containing fewer variables and constraints.

A question of interest is, which values of $\beta_t, t \in T$ in (5.23) should we use to have a good approximation of the τ-quantile. To answer this question, an iterative procedure is proposed to update β iteratively. Assume $\beta = \beta^\eta$ at the η iteration and σ is the schedule obtained by solving A-MILP with β^η. Given σ, for each period t, we can calculate Q_τ^t and $\hat{Q}_\tau^t$. Based on the difference between $\beta_t^\eta \hat{Q}_\tau^t$ and $\overline{risk^t}$, and the difference between Q_τ^t and $\overline{risk^t}$, β is updated according to (5.26)

$$\beta_t^{\eta+1} = \begin{cases} \beta_t^\eta & \text{if } \beta_t^\eta \hat{Q}_\tau^t \leq \overline{risk^t}, \text{ and } Q_\tau^t \leq \overline{risk^t} \\ \overline{risk^t}/\hat{Q}_\tau^t & \text{if } \beta_t^\eta \hat{Q}_\tau^t > \overline{risk^t}, \text{ and } Q_\tau^t \leq \overline{risk^t} \\ (1-\gamma)\beta_t^\eta + \gamma Q_\tau^t/\hat{Q}_\tau^t & \text{if } \beta_t^\eta \hat{Q}_\tau^t \leq \overline{risk^t}, \text{ and } Q_\tau^t > \overline{risk^t} \\ Q_\tau^t/\hat{Q}_\tau^t & \text{if } \beta_t^\eta \hat{Q}_\tau^t \geq Q_\tau^t > \overline{risk^t} \\ (1-\gamma)\beta_t^\eta + \gamma Q_\tau^t/\hat{Q}_\tau^t & \text{if } Q_\tau^t > \beta_t^\eta \hat{Q}_\tau^t > \overline{risk^t}, \end{cases} \tag{5.26}$$

For any iteration η and any period t, when both Q_τ^t and $\beta_t^\eta \hat{Q}_\tau^t$ are less than $\overline{risk^t}$ (line 1 in (5.26)), both $\max\{0, Q_\tau^t - \overline{risk^t}\}$ and $\max\{0, \beta_t \widehat{Q}_\tau^t - \overline{risk^t}\}$ are equal to 0, and it is therefore not necessary to adjust the value of β_t^η. When $\beta_t^\eta \hat{Q}_\tau^t$ is an overestimation of Q_τ^t (lines 2 and 4 in (5.26)), the value of β_t^η must be reduced. When $\beta_t^\eta \hat{Q}_\tau^t$ is an underestimate of Q_τ^t (lines 3 and 5 in (5.26)), β_t^η must be increased. A new value at the $(\eta+1)$th iteration can be obtained by $\beta_t^{\eta+1} = (1-\gamma)\beta_t^\eta + \gamma Q_\tau^t/\hat{Q}_\tau^t$, where $\gamma \in (0, 1]$. Given the values of $\beta_1^{\eta+1}, ..., \beta_H^{\eta+1}$, the resulting A-MILP model is solved to produce a feasible solution. We terminate the updating procedure when the estimation error (Δ^η) drops below a threshold $\epsilon \geq 0$. That is,

$$\Delta^\eta = \max_{t \in T}\{|\beta_t^{\eta+1} - \beta_t^\eta|\} \leq \epsilon \tag{5.27}$$

The proposed iterative updating approach, which will be referred to as IterUpdate, does not guarantee to yield an optimal solution to the considered problem since the updating rule for $\beta_t, t \in T$ in (5.26) only considers the estimation errors for the quantile term in the objective function. In other words, the solution with the smallest estimation error does not necessarily be the solution with the smallest value of the objective function. However, the IterUpdate algorithm (see Algorithm 17) will always yield a feasible solution to the considered problem.

Algorithm 17 Iterative Updating Algorithm (IterUpdate)

Input: A problem instance
Output: A feasible schedule σ
Step 1: Select the initial values for $\beta_t^0, \forall t \in T$
Step 2: Solve A-MILP using $\beta_t^0, \forall t \in T$, resulting in a solution σ.
Set $\eta = 1$
while stopping criterion (5.27) is not satisfied and $\eta <$ *maximum number of iterations* **do**
    Update $\beta_t^\eta, t \in T$ according to (5.26)
    Solve A-MILP using $\beta_t^\eta, t \in T$
    Set $\eta = \eta + 1$
end while
return *the best feasible schedule* σ

5.4 Confidence method approaches

The guaranteeing (confidence) approach [95] is a method for solving stochastic optimization problem in which the quantile of the distribution of an objective function is the criterion to be optimized. The problem is known by the name “Quantile Optimization Problem" in the literature. In this section, we develop two heuristic approaches which utilize the idea of confidence approach. The heuristics will be referred to as confidence-method heuristic and critical-scenario confidence-method heuristic, respectively.

In the confidence-method heuristic, we determine which values will be assigned to the binary variables $q^{\omega,t},\ \omega \in \Omega_t,\ t \in T$ in the original model MILP, using information from an initial

solution, which can be obtained using the IterUpdate algorithm described in Section 5.3. For any solution σ, denote by $\mathscr{C}_{t,\tau}(\sigma) = \{\omega : risk^{\omega,t} \leq Q^t_\tau,\ \omega \in \Omega_t\}$ the confidence set in period t. Then, the corresponding $q^{\omega,t}$ variable in (5.13) is set to 0 if the scenario belongs to $\mathscr{C}_{t,\tau}(\sigma)$, and to 1 otherwise. The process of finding the confidence sets is applied iteratively. When the confidence-method heuristic reaches the time limit or when it cannot find an improved solution after a maximal permissible number of consecutive iterations, the procedure is terminated and the solution from the last iteration is returned. To simplify notation, we suppress dependence on the given solution σ and simply use $\mathscr{C}_{t,\tau}$ instead of $\mathscr{C}_{t,\tau}(\sigma)$. The MILP model, subject to the imposed confidence sets, is as follows:

$$\text{(C-MILP)}\ \min : \alpha \times \frac{1}{H} \sum_{i \in I} \sum_{d \in T_i} \overline{risk}_{i,d}\ x_{i,d} + (1-\alpha) \times \frac{1}{H} \sum_{t=1}^{H}\ y_t \tag{5.28}$$

subject to (5.1), (5.2), (5.4), (5.10) − (5.12),

$$risk^{\omega,t} \leq Q^t_\tau, \quad \omega \in \mathscr{C}_{t,\tau},\ t \in T \tag{5.29}$$

$$Q^t_\tau - \overline{risk^t} \leq y_t, \quad t \in T \tag{5.30}$$

$$y_t \geq 0, \quad t \in T \tag{5.31}$$

$$x_{i,d} \in \{0,1\}, \quad i \in I,\ d \in T_i \tag{5.32}$$

where (5.29) imposes the requirements that the risk values corresponding to the scenarios in $\mathscr{C}_{t,\tau}$ must be less than or equal to the τ-quantile. For each period t, the set Ω_t is split into two sets $\mathscr{C}_{t,\tau}$ and $\Omega_t \setminus \mathscr{C}_{t,\tau}$ and the Constraint (5.13) is replaced by (5.29). At each iteration of the confidence-method heuristic, we solve the subproblem C-MILP. Since the solution at iteration η is feasible at iteration $\eta+1$, it can be provided to the IP solver as a "warm start". Naturally, each iteration therefore results in a solution no worse than the previous. The confidence-method heuristic is summarized below.

Confidence-method heuristic

Step 0. (Initialization) Generate an initial solution σ by the IterUpdate algorithm described in Section 5.3.

Step 1. Set $\eta = 1$, construct the confidence sets $\mathscr{C}^{\eta}_{t,\tau}(\sigma),\ t \in T$ for the initial solution σ.

Step 2. Using $\mathscr{C}_{t,\tau} = \mathscr{C}^{\eta}_{t,\tau}(\sigma),\ t \in T$, solve the C-MILP to obtain a new solution σ'.

Step 3. If stopping criterion is satisfied, then go to Step **5**.

Step 4. Set $\eta = \eta + 1$ and $\sigma = \sigma'$, construct the confidence sets $\mathscr{C}^{\eta}_{t,\tau}(\sigma),\ t \in T$ and return Step **2**.

Step 5. Output solution σ'.

Another heuristic based on the confidence approach is the critical-scenario confidence-method heuristic. In the critical-scenario confidence-method heuristic, inequality in the form of (5.29) is only added to the problem when it is necessary. The reason is because having too many variables fixed as with the confidence-method heuristic, the opportunity for finding an improved solution can be low. Moreover, the resulting C-MILP model is still too large to be solved to optimality in a reasonable time given the large instances.

Given an initial solution σ, one can find the confidence sets $\mathscr{C}_{t,\tau}(\sigma),\ t \in T$. The scenario in the confidence set with largest $risk^{\omega,t}$ value will be used to generate a constraint (5.29) that is added to the reduced problem. We refer to this scenario as critical scenario. The process of finding the critical scenarios is applied iteratively. The constraints in (5.29) are incrementally added based on critical scenario at each iteration in an attempt to improve the lower approximation to the τ-quantile Q^t_τ. Let $risk^{\omega^\eta_t,t}$ denote the risk value for period t and critical scenario ω^η_t at iteration η, the reduced problem can be written as

(reduced-MILP)

$$\min : \alpha \times \frac{1}{H} \sum_{i \in I} \sum_{d \in T_i} \overline{risk}_{i,d}\ x_{i,d} + (1-\alpha) \times \frac{1}{H} \sum_{t=1}^{H}\ y_t \tag{5.33}$$

$$\text{subject to } (5.1),\ (5.2),\ (5.4),\ (5.10)-(5.12),\ (5.30),\ (5.31),\ (5.32)$$

$$risk^{\omega^i_t,t} \leq Q^t_\tau, \quad i = 1, ...\eta, t \in T \tag{5.34}$$

The critical-scenario confidence-method heuristic is summarized below.

Critical-scenario confidence-method heuristic

Step 0. (Initialization) Generate an initial solution σ by the IterUpdate algorithm described in Section 5.3.

Step 1. Set $\eta = 1$, construct the confidence sets $\mathscr{C}^{\eta}_{t,\tau}(\sigma),\ t \in T$ for the initial solution σ, and find the critical scenarios $\omega^{\eta}_t,\ t \in T$.

Step 2. Solve the reduced-MILP to obtain a new solution σ'.

Step 3. If stopping criterion is satisfied, then go to Step **5**.

Step 4. Set $\eta = \eta + 1$ and $\sigma = \sigma'$, construct the confidence sets $\mathscr{C}^{\eta}_{t,\tau}(\sigma),\ t \in T$, find the critical scenarios $\omega^{\eta}_t,\ t \in T$ and return to Step **2**.

Step 5. Output solution σ'.

Both the CM and cs-CM algorithms are vulnerable to becoming stuck in a local optimum. It is possible to overcome such an issue by considering perturbations. For example, one can perturb the best currently found solution by randomly choosing some interventions and one by one assigning to them new starting times. It is not strictly necessary to take into account the resource and exclusion constraints during perturbation since the purpose of perturbation is to yield a different confidence set. Moreover, allowing infeasible solutions may lead to the confidence sets that render a desired solution for the original problem.

5.5 Iterated local search

Iterated Local Search (ILS) [107] has been widely applied to solve a variety of combinatorial optimization problems and has delivered high-quality solutions (see, for example, [33, 4, 72, 94]). In this section, we propose an ILS algorithm for the grid operation-based outage maintenance planning problem. The key idea of the proposed ILS is the self adaptive perturbation strategy, which dynamically modifies the perturbation strength based on the

evaluation of the neighborhoods around the local optimum. Failure to improve the local optimum after a certain number of iterations is an indication that the perturbation strength should be amplified. Additionally, a restart strategy is incorporated into the ILS framework, which permits the algorithm to restart the search from the current best solution. This restart strategy can prevent the algorithm from spending too much time in an unpromising region of the search space and help find better solutions for some hard instances. In what follows, whenever this restart strategy is invoked, a new “path" is created.

The motivation for using iterated local search is as follows. The problem in this chapter was put forward for the ROADEF/EURO challenge 2020 and the iterated local search (ILS) method was developed during our participation in the competition. The teams were evaluated by executing (only once) the submitted computer programs on the organizer's computer with a time limit of 15 minutes and another one of 1.5 hours. Therefore, in order to achieve high score in the competition, our solution method must produce high-quality solutions in a short time. For this reason, we decided to use ILS because it is more time-efficient than GA.

The general framework of the proposed ILS is outlined in Algorithm 18. In this pseudocode, Z is the objective function (5.8), and λ and λ_{big} are the notations for perturbation strength. Additionally, the parameter W is the time limit imposed on the algorithm, Y specifies the maximum permissible number of consecutive unsuccessful attempts to improve the current best solution $\hat{\sigma}^*$, and Λ is the upper bound for the perturbation strength.

The proposed ILS starts with the subroutine INITIAL (line 1) which generates a high-quality feasible solution to the original problem. This initial solution is considered as the current best solution. In the pseudocode below, the current best solution of a particular path and over all paths is denoted by $\hat{\sigma}^*$ and σ^*, respectively.

The parameter W (line 8) specifies the time limit for the ILS procedure to find a better solution with respect to the value of the objective function (5.9) (WHILE loop lines 8 - 33). Each iteration of the WHILE loop starts with a solution σ obtained by applying a perturbation on either $\hat{\sigma}^*$ or σ^*. The perturbed solutions are always produced by a sequential

Algorithm 18 Iterated local search

$\sigma \leftarrow$ INITIAL()
$\sigma^* \leftarrow \sigma$ ▷ σ^* is the global optimum
$\hat{\sigma}^* \leftarrow \sigma$ ▷ $\hat{\sigma}^*$ is the local optimum
$\lambda,\ \lambda_{\text{big}} \leftarrow \lambda^0$
$i \leftarrow 0$
$\sigma \leftarrow$ PERTURB_SHIFT(σ^*, $\frac{1}{3}\lambda$)
$\sigma \leftarrow$ PERTURB_SWAP(σ, $\frac{1}{3}\lambda$)
while $time < W$ **do**
    $\sigma \leftarrow$ SEARCH(σ)
    if $Z(\sigma) < Z(\hat{\sigma}^*)$ **then**
        $\hat{\sigma}^* \leftarrow \sigma$
    end if
    if $Z(\sigma) < Z(\sigma^*)$ **then**
        $\sigma^*, \hat{\sigma}^* \leftarrow \sigma$
        $i \leftarrow 0$
        $\lambda,\ \lambda_{\text{big}} \leftarrow \lambda^0$
        $\sigma \leftarrow$ PERTURB_SHIFT(σ^*, $\frac{1}{3}\lambda$)
        $\sigma \leftarrow$ PERTURB_SWAP(σ, $\frac{1}{3}\lambda$)
    else
        if $i > Y$ **then**
            $\lambda \leftarrow \gamma\lambda$ ▷ increase perturbation strength
        end if
        if $\lambda > \Lambda$ **then**
            $\sigma \leftarrow$ PERTURB_SWAP(σ^*, λ_{big}) ▷ start a new “path"
            $\lambda \leftarrow \lambda^0$
            $\lambda_{\text{big}} \leftarrow \lambda_{\text{big}} + 1$
        else
            $\sigma \leftarrow$ PERTURB_SHIFT($\hat{\sigma}^*$, $\frac{1}{3}\lambda$)
            $\sigma \leftarrow$ PERTURB_SWAP(σ, $\frac{1}{3}\lambda$)
        end if
    end if
    $i \leftarrow i + 1$
end while
return σ^*

application of two types of perturbation moves (a call of the subroutines PERTURB_SHIFT and PERTURB_SWAP). If, however, a new “path" is invoked as the perturbation strength exceeds the given permissible number Λ, then the perturbed solution is produced by the subroutine PERTURB_SWAP (line 24) only.

Given the perturbed solution, the ILS algorithm attempts to find a better solution using the subroutine SEARCH, which is a sequence of local search procedures. Three neighborhood operators will be used for this purpose: *one-shift*, *two-swap*, and *clique based large neighborhood search (C-LNS)*. Each iteration of the selected operator performs an exhaustive search in the corresponding neighborhood, and selects the solution with the smallest value of the objective function (5.8).

5.5.1 Subroutine INITIAL

The subroutine INITIAL for constructing an initial solution required in step 1 of Algorithms 18 includes the following steps. First, a feasible solution is obtained by solving the MILP model or the A-MILP model. The former is always used, except if the problem has a maximum number of scenarios ($\max_{t\in T} \Omega_t$) more than six. If this happens, the feasible solution is obtained using the A-MILP model. Next, the quality of the solution should be improved whenever possible before entering the ILS procedure. For this reason, the confidence method described in Section 5.4 is used, followed by a further improvement from applying the local search with *one-shift* and *two-swap*.

In our implementation of the subroutine INITIAL, we solve the A-MILP model in two stages. The first stage aims to find a feasible solution with high probability in a short time. This can be achieved by solving the model without an objective function. In the second stage, the populate function of CPLEX is used to generate a pool of solutions. Each solution in the pool is examined and the one with the smallest value of the objective function (5.8) will be selected.

5.5.2 Subroutine SEARCH

The big challenge for applying local search idea to the considered problem is the extensive computational effort for assessing the quality of candidate solutions with a large number of scenarios. This is due to the quantile term in the objective function. We tested the sample average approximation method commonly used in the literature, but found that it is not competitive in terms of solution quality. We believe it is important for the local search operators that we can efficiently calculate for a given solution the exact objective function value. Therefore, we propose three simple local search operators: (i) *clique based large neighborhood search (C-LNS)*, (ii) *one-shift*, and (iii) *two-swap*.

Clique based large neighborhood search (C-LNS) - The basic idea of large neighborhood search is to explore a complex neighborhood, aiming to travel across promising search path and find better solutions at each iteration [157]. In the case of *C-LNS*, a move consists of deleting some non-overlapping interventions from the current solution $\tilde{\sigma}$ and then finding the new starting times for these interventions by solving an integer program where the starting times of the remaining interventions are fixed. CPLEX might be able to find better solution than $\tilde{\sigma}$, and if this happens, the move is immediately executed and the search goes on. We discuss how to formulate the integer programming model below.

Let $\tilde{\sigma}$ be the current solution. Denote by U the list of selected non-overlapping interventions. The starting time of the remaining interventions will be fixed, i.e. $x_{i,t} = \tilde{x}_{i,t}, \forall i \in I \setminus U$. The resources consumed by this fixed partial solution is

$$\tilde{r}_t^k = \sum_{i \in I \setminus U} \sum_{d \in D_{i,t}} r_{i,d}^{k,t} \times \tilde{x}_{i,d}, \quad k \in K,\ t \in T \tag{5.35}$$

Given that the interventions in U are non-overlapping, i.e. cannot be processed concurrently, one can determine the set of allowed starting times $\tilde{T}_i,\ i \in U$ such that the resource constraints and disjunctive constraints are satisfied with respect to the fixed partial solution.

That is,

$$\begin{aligned}\tilde{T}_i =&\{d \in T_i \mid \ t \in [d, d+\Delta_{i,d}-1],\ k \in K,\ \tilde{r}_t^k + r_{i,d}^{k,t} \leq u_t^k,\\ &(i,j,t) \in E, j \notin U,\ t \in [d, d+\Delta_{i,d}-1],\\ &j \text{ does not overlap with } [d, d+\Delta_{i,d}-1]\}\end{aligned} \tag{5.36}$$

Let $\tilde{D}_{i,t} = \{d \in \tilde{T}_i : d+\Delta_{i,d}-1 \geq t, d \leq t\}$, for $i \in U$. The cumulative planning risk $risk_{fixed}^{\omega,t}$ of the fixed partial solution can be computed as:

$$risk_{fixed}^{\omega,t} = \sum_{i \in I \setminus U} \sum_{d \in \tilde{D}_{i,t}} risk_{i,d}^{s,t} \times \tilde{x}_{i,d} \tag{5.37}$$

The mean cumulative planning risk at t of the fixed partial solution will be:

$$\overline{risk_{fixed}^{t}} = \frac{1}{|\Omega_t|} \sum_{\omega \in \Omega_t} risk_{fixed}^{\omega,t} \tag{5.38}$$

The τ quantile of the risk profile of the fixed partial solution at t, denoted by $Q_{\tau,fixed}^t$, is computed according to (5.6). The objective value at t resulting from the fixed partial solution is:

$$f_{fixed}^t = \alpha\ \overline{risk_{fixed}^t} + (1-\alpha) \max\{0, Q_{\tau,fixed}^t - \overline{risk_{fixed}^t}\} \tag{5.39}$$

Assigning intervention $i \in U$ to a starting time $d \in \tilde{T}_i$ leads to changes in the values of $risk_{fixed}^t$, $Q_{\tau,fixed}^t$, and therefore f_{fixed}^t. Equation (5.40) calculates the change in the objective value, denoted by $\gamma_{i,d}$, for starting intervention i at time $d \in \tilde{T}_i$.

$$\gamma_{i,d} = \sum_{t=d}^{d+\Delta_{i,d}-1} (f_{fixed+i}^t - f_{fixed}^t), \tag{5.40}$$

where $f_{fixed+i}^t$ denote the objective value at t as a result of including i to the fixed partial solution. The discussion above leads to the following integer programming model:

Model (LNS-IP) :

$$\min \sum_{i \in U} \sum_{d \in \tilde{T}_i} \gamma_{i,d}\ x_{i,d} \tag{5.41}$$

subject to (5.35), (5.36)

$$\sum_{t \in \tilde{T}_i} x_{i,t} = 1, \quad i \in U \tag{5.42}$$

$$l_t^k \leq \sum_{i \in U} \sum_{d \in \tilde{D}_{i,t}} r_{i,d}^{k,t}\ x_{i,d} + \tilde{r}_t^k, \quad k \in K,\ t \in T \tag{5.43}$$

$$\sum_{i \in U} \sum_{d \in \tilde{D}_{i,t}} x_{i,d} \leq 1, \quad t \in T \tag{5.44}$$

$$x_{i,t} \in \{0,1\}, \quad i \in U,\ t \in \tilde{T}_i \tag{5.45}$$

At each iteration of *(C-LNS)*, our procedure for selecting the interventions to free includes the following steps. First, the current list (I) of interventions contains all interventions and the list (U) of selected non-overlapping interventions is empty. One intervention from I will be chosen at random and inserted to U. Then, the procedure scans the list I and attempts to remove the interventions which overlap with the intervention just added to U. This procedure terminates when list I becomes empty, i.e. each intervention is either added to U or removed from I (because it overlaps with some interventions in U), or when the list U reaches the required size.

As might be expected, the more interventions whose starting times are fixed, the faster the (LNS-IP) model can be solved, but it is more likely that no improvement could be made in term of the objective function value. On the other hand, the fewer interventions whose starting times are fixed, there is greater opportunity for finding a solution with an improved value of the objective function, but significantly more computational effort might be required for solving the (LNS-IP) model. We propose not to impose a restriction on the size of U so that we can free as many non-overlapping interventions as possible. At the beginning of the

first iteration of *(C-LNS)*, the input solution can be provided to the IP solver as a "warm start". In the subsequent iterations, the current best solution can be provided to the IP solver as a "warm start".

One-shift - The neighborhood explored by the operator *one-shift* is comprised of all feasible solutions that can be obtained from a solution σ by assigning a different starting time to a single intervention. A total of $n(H-1)$ possible solutions can be produced. The benefit of using *one-shift* is in the fast evaluation of each solution in the neighborhood. For example, we change the starting time of intervention 3 from time 6 to 1 in Figure 5.3. The objective value of the time periods marked in the boxes are the same, so we just need to consider the changed objective value for the affected periods, i.e. 1, 2, 3, 6, 7, and 8.

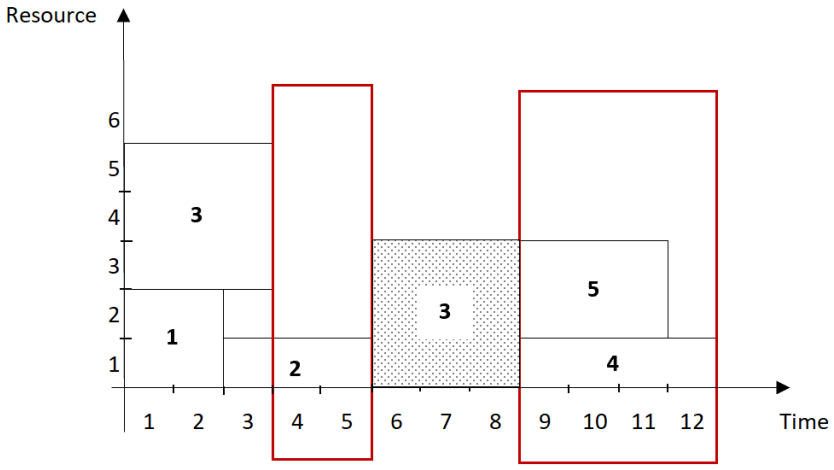


Figure 5.3: *one-shift* evaluation.

Two-swap - The *two-swap* is motivated by the classic 2-Opt approach known from the traveling salesman literature [106]. It consists of exchanging the starting times of two interventions i and j, which generates a total of $n(n-1)/2$ possible solutions, where n is the number of interventions. As in *one-shift*, each neighbor solution obtained by *two-swap* can be evaluated efficiently. For example, we swap the starting times of interventions 1 and 5 in Figure 5.4. The objective value of the time periods marked in the boxes are the same, so we just need to consider the changed objective value for the affected periods, i.e. 1, 2, 3, 9, 10,

and 11.

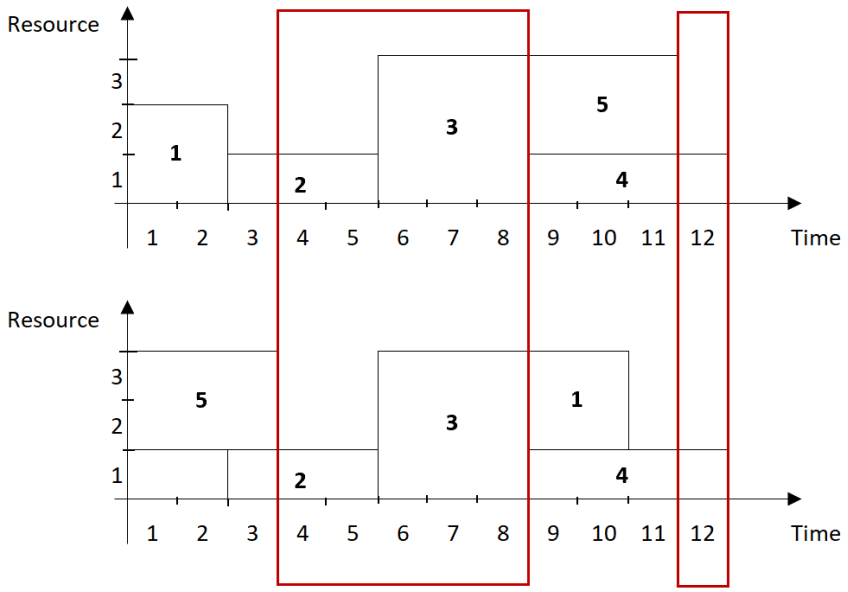


Figure 5.4: *two-swap* evaluation.

Let N_0, N_1, N_2 denote the three operators *C-LNS*, *one-shift*, *two-swap*, respectively. For a current solution σ, let $N_i(\sigma)$ denote the output solution produced by the operator N_i, $i = 0, 1, 2$. Let $\hat{\sigma}$ be a solution in the neighborhood of σ with the smallest value of the objective function, then $\hat{\sigma} = N_i(\sigma)$. If such a solution does not exist in the current neighborhood, then $\sigma = N_i(\sigma)$. The algorithm 19 below outlines the subroutine SEARCH for an input solution σ.

The subroutine SEARCH starts with an iterative local search optimization procedure with operator N_0 (REPEAT loop lines 1 - 4). This loop repeats until the permissible number of iterations is reached. The local optimum found by the local search with operator N_0 is used as an input to the composite local search with operators N_1 and N_2 (REPEAT loop lines 5 - 13). When the local search with the operator N_1 (REPEAT loop lines 8 - 11) finds a local minimum, this local minimum is used as an input to the local search with operator N_2. The subroutine SEARCH terminates if the composite local search algorithm is unable to improve the solution.

Algorithm 19 SEARCH(σ)

repeat
  $\bar{\sigma} = \sigma$
  $\sigma = N_0(\sigma)$
until termination condition met
repeat
  $\bar{\sigma} = \sigma$
  for i from 1 to 2 **do**
    repeat
      $\hat{\sigma} = \sigma$
      $\sigma = N_i(\sigma)$
    until $Z(\sigma) = Z(\hat{\sigma})$
  end for
until $Z(\sigma) = Z(\bar{\sigma})$
return $\bar{\sigma}$

5.5.3 Subroutines PERTURB_SHIFT and PERTURB_SWAP

The perturbation mechanism is responsible for providing a new starting solution for the next iteration of the subroutine SEARCH. It is a crucial component of the ILS procedure as it controls the diversification aspect of ILS. If the amount of perturbation is too large, the algorithm may behave as a random restart method, resulting in much worse local optimal solutions. Conversely, if the perturbations are too small, it may prevent the algorithm to escape from the local optimum. For this reason, the characteristics of the perturbation operator and perturbation strength (i.e. the number of components that are modified in a solution) must be carefully controlled in order to avoid over-disturbance and/or under-disturbance. In this work, we propose an adaptive perturbation mechanism that changes the perturbation strength as the algorithm progresses, and two types of perturbation operators.

The adaptive perturbation mechanism dynamically tunes the perturbation strength using the information about the quality of the neighbor solutions. In the case that the local optimum sits among many good possible solutions and the subroutine SEARCH can progressively find solution with an improved value of the objective function, then the perturbation strength λ remains unchanged to allow for more detailed exploration of the current neighborhood of the

search space. In the case that the subroutine SEARCH fails to improve the local optimum after a certain number of consecutive iterations (specified by the parameter Y in line 20 of Algorithm 18), then the perturbation strength λ is increased to $\gamma\lambda$, where $\gamma \geq 1$. This increment will enable new areas of the search space to be explored. When the value of λ reaches the upper bound Λ, further searches becomes unnecessary, so the restart strategy is invoked to replace the current best solution σ^* by a new "path". This "path" concept can enlarge the search space and help find better solution for some hard instances.

The subroutine PERTURB_SHIFT uses the *one-shift* operator, which randomly selects an intervention i, determines all feasible starting times for this intervention in the existing solution, and randomly chooses from this list a new starting time to assign to i. If there are no feasible starting times available, then no change will be made to intervention i. The subroutine PERTURB_SHIFT is terminated when the total number of operations equals to $\frac{1}{3}\lambda$, where λ is the perturbation strength.

The subroutine PERTURB_SWAP uses the *two-swap* operator, which randomly selects a pair of interventions, and examines whether the two may overlap with one another. If they are non-overlapping and swapping the starting times of these two interventions results in a feasible solution, then the swap is applied. The number of swaps depends on the perturbation strength. When the total number of swaps equals to $\frac{1}{3}\lambda$, the subroutine PERTURB_SWAP is terminated.

In the proposed ILS, whenever a new path is created (line 24 in Algorithm 18), the perturbed solution is produced by the subroutine PERTURB_SWAP. Otherwise, the perturbation mechanism consists of the subroutine PERTURB_SHIFT, followed by subroutine PERTURB_SWAP. The perturbation with *one-shift* operator is not as strong as the perturbation with *two-swap* operator. It is evident that a combination of these two types of perturbations introduces increasingly larger degrees of diversification to the search space.

5.6 Computational results

In this section, we apply the developed heuristic and metaheuristic algorithms to the four sets of instances used during the ROADEF/EURO challenge 2020. Each set includes 15 test instances. These instances can be downloaded from the Github repository of the competition `https://github.com/rte-france/challenge-roadef-2020/`. The Roadef 2020 also provided a solution checker - developed in Python3 - that allows the participants to check whether or not a solution is feasible. More about the format of the instances and solution checker can be found on [145].

For each instance, we ran each algorithm one time. The code is implemented in cython [24] with C++ STL. IBM ILOG CPLEX 12.10.0 was used to solve the mathematical programming models. The testing system was a cluster with Intel Xeon E-2288G 3.7GHz 8 cores CPU with 64GB RAM, running Red Hat Enterprise Linux.

Table 5.1 summarizes the main characteristics of the 60 instances. In Table 5.1, N is the number of interventions, M is the number of resources, H is the length of planning horizon, avg. Ω is the average number of scenarios, E is the number of exclusions, avg. Δ is the average duration of interventions in all allowed starting times, τ is the quantile level for the planning risk, and α is the weight of the first objective (Z_1). avg. Ω is calculated as $1/H \sum_{t=1}^{H} |\Omega_t|$, while avg. Δ is calculated as $1/N \sum_{i=1}^{N} (1/t_i^{\max} \sum_{t=1}^{t_i^{\max}} \Delta_{i,t})$.

For the parameter $M_t, t \in T$ in (5.13), a choice of very large M can lead to slow progress in solving the MILP due to weak relaxation. On the other hand, if M is too small, a valid choice of $risk^{\omega,t}$ may violate the constraint even when $q^{\omega,t} = 1$. With this in mind, in our implementation, we use one M parameter for each time t and compute the values using the data for risk and resources.

Table 5.1: Instances characteristics in ROADEF/EURO challenge 2020.

Instance	N	M	H	avg. Ω	E	avg. Δ	τ	α
Sprint and Qualification phases								
A_01	181	9	90	1	81	5.5	0.95	0.5
A_02	89	9	90	120	32	4.6	0.95	0.5
A_03	91	10	90	1	12	4.6	0.95	0.5
A_04	706	9	365	1	1377	8.6	0.95	0.5
A_05	180	9	182	120	87	4.9	0.95	0.5
A_06	180	10	182	1	87	4.9	0.95	0.5
A_07	36	9	17	6	3	1.4	0.50	0.5
A_08	18	9	17	646	4	1.2	0.95	0.5
A_09	18	10	17	6	0	1.8	0.50	0.5
A_10	108	9	53	6	40	1.8	0.50	0.5
A_11	54	9	53	640	4	1.2	0.95	0.5
A_12	54	10	53	6	0	1.1	0.50	0.5
A_13	179	9	90	12	136	5.2	0.50	0.5
A_14	108	10	53	160	22	1.7	0.95	0.5
A_15	108	10	53	320	22	1.7	0.95	0.5
Semi-final phase								
B_01	100	9	53	191	26	10.6	0.90	0.5
B_02	100	9	53	191	19	11.7	0.90	0.5
B_03	706	9	53	63	1192	11.0	0.90	0.5
B_04	706	9	53	63	1192	11.0	0.90	0.5
B_05	706	9	53	63	1377	1.7	0.90	0.5
B_06	100	9	53	255	19	11.7	0.90	0.5
B_07	250	9	53	191	186	10.8	0.80	0.5
B_08	119	9	42	254	37	8.8	0.95	0.5
B_09	120	9	42	127	44	7.5	0.95	0.5
B_10	398	9	25	192	344	5.1	0.80	0.5
B_11	100	9	53	191	34	11.1	0.90	0.5
B_12	495	9	102	63	570	24.8	0.95	0.5
B_13	99	9	102	159	4	24.6	0.90	0.5
B_14	297	9	191	95	207	47.1	0.80	0.5
B_15	495	9	250	63	665	52.2	0.80	0.5
Final phases								
C_01	120	9	53	191	54	11.1	0.95	0.5
C_02	120	9	53	191	43	11.4	0.80	0.5
C_03	706	9	53	63	1223	10.9	0.85	0.5
C_04	706	9	53	63	1194	11.0	0.90	0.5
C_05	706	9	53	63	1377	1.7	0.95	0.5
C_06	280	9	53	191	183	11.1	0.80	0.5
C_07	120	9	42	126	38	7.7	0.95	0.5
C_08	426	9	25	192	340	5.0	0.80	0.5
C_09	110	9	53	191	38	11.7	0.90	0.5
C_10	522	9	102	63	705	24.9	0.95	0.5
C_11	89	9	102	191	35	27.0	0.90	0.5
C_12	298	9	191	95	195	47.0	0.80	0.5
C_13	505	9	230	63	53	58.9	0.95	0.5
C_14	465	9	220	95	620	54.5	0.85	0.5
C_15	528	9	300	51	624	74.7	0.95	0.5
X_01	120	9	53	191	48	10.9	0.80	0.5
X_02	706	9	53	63	1234	11.0	0.85	0.5
X_03	280	9	53	191	162	10.7	0.80	0.5
X_04	426	9	25	188	490	5.1	0.80	0.5
X_05	467	9	220	95	604	55.3	0.85	0.5
X_06	528	9	300	50	703	77.7	0.95	0.5
X_07	209	9	300	63	80	74.9	0.90	0.5
X_08	209	9	300	63	57	75.0	0.90	0.5
X_09	548	9	30	156	820	6.5	0.80	0.5
X_10	460	9	35	159	527	7.3	0.95	0.5
X_11	521	9	131	63	725	32.4	0.95	0.5
X_12	522	9	131	63	723	33.1	0.95	0.5
X_13	336	9	212	95	248	54.7	0.90	0.5
X_14	613	9	180	63	951	47.3	0.95	0.5
X_15	613	9	180	63	917	45.8	0.95	0.5

Table 5.2: Comparison of models MILP and A-MILP on dataset A instances.

Instance	MILP				A-MILP		
	Time (s)	UB	LB	Gap (%)	Time (s)	Z	Gap (%)
A_01	2	**1767.81561**	1767.81561	0	2	1767.81561	0
A_02	7200	4673.95911	2112.63630	54.80	4	4736.84755	124
A_03	1	**848.17861**	848.17861	0	0.4	848.17861	0
A_04	74	**2085.87605**	2085.87605	0	68	2085.87605	0
A_05	7200	638.44659	594.46740	6.89	4	645.42793	8.57
A_06	4	**590.62359**	590.62359	0	3	590.62359	0
A_07	0.3	**2272.78227**	2272.78227	0	0.5	2280.60656	0.34
A_08	7200	744.70441	692.65000	6.99	0.2	749.95088	8.27
A_09	0.2	**1507.28478**	1507.28478	0	0.1	1525.64978	1.21
A_10	4	**2994.84873**	2994.84873	0	0.3	3016.16708	0.71
A_11	7200	495.86537	461.75280	6.88	2	504.78716	9.31
A_12	2	**789.63492**	789.63492	0	0.7	789.78896	0.02
A_13	7200	1998.90557	1998.84030	0.003	12	2004.39403	0.28
A_14	7200	2275.50198	2085.27950	8.35	3	2295.81830	10.10
A_15	7200	2290.71433	2072.28270	9.53	4	2302.47084	11.11

5.6.1 Model MILP vs. Model A-MILP

We first evaluate the proposed models MILP and A-MILP, by solving the instances in dataset A by CPLEX. Table 5.2 summarizes the outcomes. The columns 'Time (s)', 'UB', 'LB', and 'Gap (%)' report the running time (in seconds), upper bounds, lower bounds, and gaps obtained by CPLEX, respectively. For model A-MILP, parameter $\beta_t,\ \forall t \in T$ was fixed at 1; the objective values of the solutions are computed according to (5.8) and reported under the column titled 'Z'; the last column 'Gap (%)' reports the deviation of the objective value from LB and is calculated as Gap (%) $= (Z - \text{LB})/\text{LB} * 100$. For both models, CPLEX stopped after reaching a 2 hours limit or when an optimal solution was found.

The results in Table 5.2 show that CPLEX obtained optimal solutions to MILP when avg. $\Omega \leq 6$. The smallest instance A_09 is solved to optimality in less than 1 second while the largest instance A_04 requires approximately 74 seconds. As the number of scenarios in an instance increases, the required computational effort for a solution of MILP grows rapidly. For the instances where CPLEX could not produce an optimal solution in 2 hours, the average gap was 13.35%. A_02 has the largest gap of 54.8% among the instances in dataset A. Further investigation into the characteristics of A_02 suggests that the difficulty of this instance is due to its high risk values. When A-MILP is used, CPLEX can solve all

instances to optimality in less than 100 seconds. The times taken to solve A-MILP is on average 71% less than MILP, but solution quality is on average 0.59% worse than MILP.

5.6.2 Evaluation of the IterUpdate algorithm

We conduct an experiment to assess the effect of the initial values $\beta_t^0, \forall t \in T$ (in line 3 of Algorithm 17) on solution quality of the IterUpdate algorithm. We initialized the parameter $\beta_t^0, \forall t \in T$ with three different settings, i.e. 1, 0.01, and $U[0,1]$. For all instances, the parameter γ in (5.26) was set to 0.6. Termination criteria is a 900 second time limit (excluding the time to read the data file) or the smallest estimation error of less than 1×10^{-5} or the number of iterations without improvement of no more than 20. For each iteration within the heuristic, CPLEX stopped after reaching a 500 seconds limit or when the problem is solved to within 1%-optimality.

We use two criteria of "Number of best obtained solutions (N_{best})" and "Average relative percentage time ($ARPT$)" to compare the results obtained for different settings of β_t^0. The metric $ARPT$ is obtained as follows: first, we compute the average computation time, denoted by ACT, for all three settings on the same instance; then, for each instance, we find the relative percentage computation time of a setting, denoted by RPT, using the formula $RPT = (Time(s) - ACT)/ACT \times 100$; and finally, $ARPT$ can be obtained by averaging the RPT of all instances in one dataset. It is possible that CPLEX cannot find a solution in 500 seconds for some large instances (e.g. dataset X). If this happens, the instance is excluded from the comparison.

A summary of the parameters tuning for the IterUpdate algorithm can be found in Table 5.3. The highlighted numbers denote the outperforming values. For dataset A, the random initialization obtains better solutions than the other two settings at a cost of longer computing time. For dataset B, both settings of 1 and 0.01 achieve the same number of best obtained solution of 7 but the computing time of the former is significantly shorter. $\beta_t^0 = 1$ is the superior setting for instances of dataset C, while $\beta_t^0 = 0.01$ is more suitable for dataset X

Table 5.3: Summary of $\beta_t^0, t \in T$ for the IterUpdate algorithm on four datasets.

Metric	Dataset	$\beta^0 = 1$	$\beta^0 = 0.01$	$\beta^0 =$ random
N_{best}	A	4	8	**10**
	B	**7**	**7**	3
	C	**8**	4	4
	X	1	**9**	4
	Sum	20	**30**	19
ARPT	A	**-17.45**	6.77	10.68
	B	**-64.00**	76.02	-12.02
	C	**-51.28**	39.44	11.84
	X	**-45.05**	26.45	18.60
	Average	**-44.44**	37.17	7.27

instances. Overall, $\beta_t^0 = 0.01$ is the best setting in term of N_{best}, and $\beta_t^0 = 1$ is the second best. However, the latter takes nearly a hundredfold less in computing time. To conclude, in terms of solution quality and time, $\beta_t^0 = 1$ is the preferred setting for the IterUpdate algorithm. Therefore, the IterUpdate algorithm with $\beta_t^0 = 1, \forall t \in T$ is used to generate initial solutions in the remainder of this chapter.

We conclude this section with Figure 5.5 illustrating the convergence of the approximate Z_2 to the true Z_2 for instances A_02 and A_11.

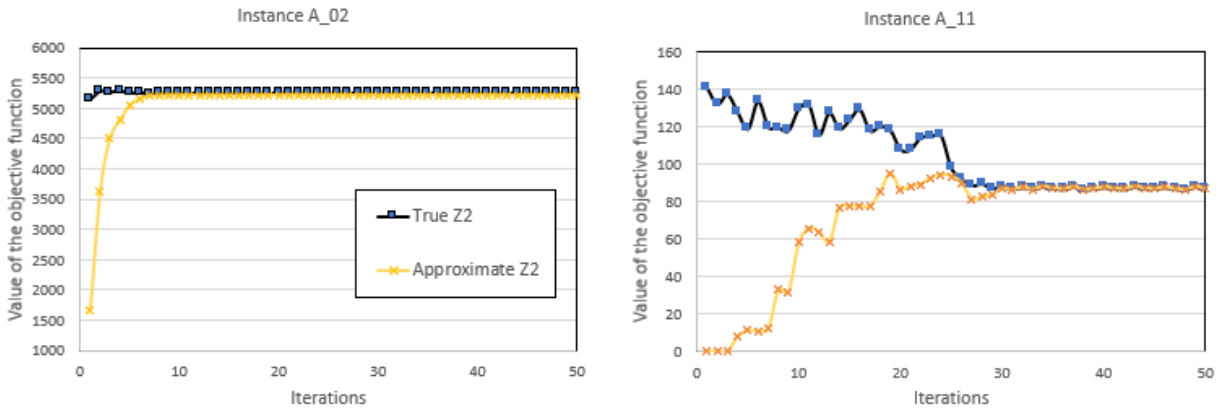


Figure 5.5: Convergence of the approximate Z_2 to the true Z_2.

5.6.3 CM-heuristic vs. cs-CM heuristic

In the following, we compare the performance of CM-heuristic and cs-CM heuristic on the four datasets, the results of which are provided in Tables 5.4 - 5.7. We report the change in objective value which is given by subtracting the objective value of the final solution from the objective value of the initial solution; the total time (in seconds) which includes time for running the methods but does not include time for generating the initial solution; the average time spent by CPLEX (in seconds); the average MIP relative gap by CPLEX (%); the average number of constraints of the MIP models over all iterations; and total number of iterations the procedure took to terminate.

For the sake of brevity, these tables do not present results of the instances, where both CM and cs-CM fail to improve the initial solutions. The reason is that both CM and cs-CM heuristics obtain a starting solution by the IterUpdate algorithm, hence both methods will output the same result if both fail to improve the starting solution. Therefore, omitting these results will not affect the conclusion. To ensure fair comparisons, we initialize both methods with the same initial solution, i.e. we use the solution obtained by CPLEX for the IterUpdate algorithm with $\beta_t^0 = 1$. The CM-heuristic terminates when the solution has converged, whereas the cs-CM heuristic is to stop after 10 iterations. For both approaches, CPLEX is used to solve the resulting mixed integer linear programming models and a time limit of 100 seconds is imposed on CPLEX. For the first iteration of both methods, we use the initial solution to warm start CPLEX. For iteration $\eta > 1$, we use both the current best solution and the solution from iteration $\eta - 1$ to warm start CPLEX.

Results in Table 5.4 indicate that the CM-heuristic works well on small instances, quickly finding a better solution and exiting in a few iterations. This is because CPLEX can easily solve these small problems to optimality in 100 seconds. With cs-CM, at the early iterations of this method, the solutions are often of poor quality because the estimation of the quantile term in the objective function is not accurate. As more constraints are added to the model in consecutive iterations, the discrepancy between the actual objective value of model objective

Table 5.4: Results obtained by CM-heuristic and cs-CM heuristics for dataset A.

Instance	Method	Change in objective value	Total time	Average solution time	Average gap	Average constraints	Total Iterations
A_02	CM	57.45	408	100	0.073%	13468	4
	cs-CM	58.94	320	32	0.001%	2964	10
A_05	CM	6.89	230	100	0.775%	28980	2
	cs-CM	3.75	842	84	0.334%	7560	10
A_07	CM	5.43	0.5	0.2	0.000%	348	2
	cs-CM	5.43	0.3	0.0	0.000%	254	10
A_08	CM	0.14	2.9	0.7	0.000%	11192	2
	cs-CM	0.00	1.5	0.1	0.000%	250	10
A_09	CM	18.02	0.1	0.0	0.000%	318	2
	cs-CM	18.02	0.2	0.0	0.000%	223	10
A_10	CM	16.83	1.2	0.3	0.001%	1685	3
	cs-CM	16.15	1.8	0.1	0.001%	1394	10
A_11	CM	1.52	146	34	0.001%	34628	4
	cs-CM	0.00	6.2	1	0.001%	818	10
A_12	CM	0.34	0.5	0.1	0.000%	991	2
	cs-CM	0.00	1.2	0.1	0.000%	707	10
A_13	CM	1.76	47	15	0.001%	7198	3
	cs-CM	2.04	68	7	0.001%	6162	10
A_14	CM	29.47	403	100	0.230%	9672	4
	cs-CM	1.65	367	36	0.001%	1344	10
A_15	CM	19.99	508	100	0.350%	18139	5
	cs-CM	0.00	434	43	0.001%	1340	10

value reduces, and therefore, CPLEX can start to improve the initial solution.

Results in Table 5.5 - 5.7 indicate that, for the large instances in dataset B, C, and X, cs-CM outperform CM-heuristic in terms of solution quality. For example, for X_07, cs-CM yields an improvement of about 0.05% as compared to the initial solution, whereas CM-heuristic is not able to give any improvement on this instance. Solution time can grow with the number of iterations, and so cs-CM, which could stop only after a predetermined number of iterations, i.e. 10 iterations, can require more computational effort. However, note that, for example, for X_07, the number of iterations of cs-CM is 10 times that of CM, yet the solution time only grew 4 times. This is because solving 10 smaller problems (cs-CM with 11002 constraints on average) can be more efficient than solving one large problem (CM with 29633 constraints).

Table 5.5: Results obtained by CM-heuristic and cs-CM heuristics for dataset B.

Instance	Method	Change in objective value	Total time	Average solution time	Average gap	Average constraints	Total Iterations
B_02	CM	0.86	214	101	0.430%	11186	2
	cs-CM	0.00	908	90	1.370%	1256	10
B_03	CM	9.67	324	100	0.060%	22799	3
	cs-CM	15.81	1022	100	0.053%	19592	10
B_04	CM	19.92	324	100	0.008%	22799	3
	cs-CM	16.91	923	91	0.005%	19589	10
B_05	CM	4.27	412	100	0.103%	25542	4
	cs-CM	3.29	923	91	0.055%	22351	10
B_09	CM	1.34	41	18	0.000%	6442	2
	cs-CM	1.33	28	3	0.001%	1096	10
B_12	CM	0.00	157	101	100%	23257	1
	cs-CM	18.79	962	92	0.009%	17001	10

Table 5.6: Results obtained by CM-heuristic and cs-CM heuristics for dataset C.

Instance	Method	Change in objective value	Total time	Average solution time	Average gap	Average constraints	Total Iterations
C_03	CM	17.77	416	100	0.070%	22735	4
	cs-CM	18.06	1011	100	0.040%	19533	10
C_04	CM	27.75	715	100	0.013%	23299	7
	cs-CM	22.06	914	90	0.007%	10107	10
C_05	CM	4.55	807	100	0.110%	25542	8
	cs-CM	1.22	1004	100	0.137%	22320	10
C_10	CM	0.00	145	100	100%	27071	1
	cs-CM	25.74	964	94	0.005%	20817	10
C_11	CM	0.00	86	63	0.001%	22015	1
	cs-CM	0.44	26	2	0.001%	2730	10

Table 5.7: Results obtained by CM-heuristic and cs-CM heuristics for dataset X.

Instance	Method	Change in objective value	Total time	Average solution time	Average gap	Average constraints	Total Iterations
X_02	CM	6.48	217	100	0.088%	23715	2
	cs-CM	0.00	1012	100	0.113%	20472	10
X_07	CM	0.00	268	124	100%	29633	1
	cs-CM	7.04	1051	99	0.032%	11002	10
X_11	CM	0.00	165	101	100%	34672	1
	cs-CM	20.51	1029	99	0.030%	26643	10

5.6.4 Comparison of the ILS with benchmark results

We test the proposed ILS algorithm and compare its performance with the best known results from other participants. As in the competition, we run the proposed ILS with a time limit of 15 minutes and another one of one hour and a half. Tables 5.8 - 5.11 report the results on the four datasets, respectively. The highlighted numbers denote the outperforming or the same values. In these tables, 'Best' is the best known results from other participants; 'ILS' is the

results from our proposed ILS algorithm; and '%Diff' is the percentage difference calculated as %Diff = (ILS − Best)/Best × 100.

For the instances with avg. $\Omega \leq 6$, initial solution to the iterated local search in Algorithm 18 is obtained by solving the (MILP). For all other instances, the IterUpdate algorithm with $\beta_t^0 = 1$ is used to provide the initial solution. A time limit of 500 seconds is given to the IterUpdate algorithm. At each iteration of the local search LS(*C-LNS*), the set of non-overlapping tasks U is formed in the following way: a task u is chosen at random from the set I; The remaining tasks are removed from I if they overlap with u; the procedure continues until set I is empty. At each iteration, the model (LNS-IP) is solved by CPLEX with a time limit of 100 seconds.

The parameters of ILS are set as follows. The value of W is calculated using the formula $W = \mathcal{T} - \mathcal{T}_R - \mathcal{T}_I$, where $\mathcal{T}$ is either 15 minutes or 1.5 hours, $\mathcal{T}_R$ is the amount of time it takes to read the instance, and $\mathcal{T}_I$ is the time used for obtaining the initial solution. The initial value of perturbation strength (λ) is 3. We set $\Lambda = 12$, $\gamma = 1.2$, and $Y = 10 + N/250$, where N is the number of interventions. Note that Y will be rounded to the nearest integer if the division $N/250$ gives a non-integer value.

Table 5.8: Performance of ILS on dataset A instances.

Instance	15 minutes			1.5 hours		
	Best	ILS	%Diff	Best	ILS	%Diff
A_01	1767.81561	**1767.81561**	0.00%	1767.81560	**1767.81561**	0.00%
A_02	4671.37661	**4671.37661**	0.00%	4671.37660	4671.37661	0.00%
A_03	848.17861	**848.17861**	0.00%	848.17861	**848.17861**	0.00%
A_04	2085.87605	**2085.87605**	0.00%	2085.87605	**2085.87605**	0.00%
A_05	635.22178	635.59898	0.06%	635.22178	635.298076	0.01%
A_06	590.62359	**590.62359**	0.00%	590.62359	**590.62359**	0.00%
A_07	2272.78227	**2272.78227**	0.00%	2272.78227	**2272.78227**	0.00%
A_08	744.29323	**744.29323**	0.00%	744.29323	**744.29323**	0.00%
A_09	1507.28478	**1507.28478**	0.00%	1507.28478	**1507.28478**	0.00%
A_10	2994.84873	**2994.84873**	0.00%	2994.84873	**2994.84873**	0.00%
A_11	495.25577	495.32171	0.01%	495.25577	**495.25577**	0.00%
A_12	789.63492	**789.63492**	0.00%	789.63492	**789.63492**	0.00%
A_13	1998.66216	1999.62679	0.05%	1998.66216	1998.79003	0.01%
A_14	2264.12432	**2264.12432**	0.00%	2264.12432	**2264.12432**	0.00%
A_15	2268.56915	**2268.56915**	0.00%	2268.56915	2269.54047	0.04%

Table 5.9: Performance of ILS on dataset B instances.

Instance	15 minutes			1.5 hours		
	Best	ILS	%Diff	Best	ILS	%Diff
B_01	3986.20283	**3986.20283**	0.00%	3986.20283	**3986.20283**	0.00%
B_02	4302.77452	**4300.05660**	-0.06%	4301.65660	**4299.00566**	-0.06%
B_03	35279.53018	35284.81226	0.01%	35277.22830	35281.73773	0.01%
B_04	34827.86981	34828.87547	0.00%	34826.94622	34828.84056	0.01%
B_05	2397.09905	**2396.70660**	-0.02%	2397.10094	2397.18301	0.00%
B_06	4287.89434	**4283.36320**	-0.11%	4284.67169	4284.73867	0.00%
B_07	7564.03490	**7555.35566**	-0.11%	7555.95000	**7551.01792**	-0.07%
B_08	7435.71904	**7435.71904**	0.00%	7435.71904	**7435.71904**	0.00%
B_09	7491.75357	7493.61666	0.02%	7491.75357	7496.87857	0.07%
B_10	10637.62000	**10602.87600**	-0.33%	10633.01600	**10611.42199**	-0.20%
B_11	3626.27169	3629.57735	0.09%	3626.03490	3623.07452	-0.08%
B_12	37602.96666	**37600.49166**	-0.01%	37601.38382	37601.55637	0.00%
B_13	5024.49264	5024.96813	0.01%	5024.49264	**5024.49264**	0.00%
B_14	11905.10994	11915.00471	0.08%	11901.76858	11908.87356	0.06%
B_15	22566.00340	**22564.34420**	-0.01%	22563.53880	22563.64279	0.00%

Table 5.10: Performance of ILS on dataset C instances.

Instance	15 minutes			1.5 hours		
	Best	ILS	%Diff	Best	ILS	%Diff
C_01	8515.90377	8517.58301	0.02%	8515.90377	**8515.90377**	0.00%
C_02	3541.65377	3548.55471	0.19%	3539.80377	3541.53301	0.05%
C_03	33511.70000	33517.10094	0.02%	33512.25660	33519.07261	0.02%
C_04	37585.73113	37589.06981	0.01%	37586.30849	37588.03867	0.00%
C_05	3166.88962	3167.43773	0.02%	3166.18207	3167.00000	0.03%
C_06	8396.00094	8418.44245	0.27%	8394.48301	8411.25943	0.20%
C_07	6083.27023	6083.60000	0.01%	6083.04404	6083.60000	0.01%
C_08	11162.83600	11193.16198	0.27%	11155.64000	11191.93800	0.33%
C_09	5586.97924	5598.36037	0.20%	5585.65188	5595.71037	0.18%
C_10	43342.48872	43343.28235	0.00%	43341.83676	43344.15490	0.01%
C_11	5749.95735	**5749.95735**	0.00%	5749.95735	**5749.95735**	0.00%
C_12	12721.13324	12735.89214	0.12%	12718.79057	12730.73507	0.09%
C_13	42487.99282	42489.57130	0.00%	42484.56065	42485.73760	0.00%
C_14	26467.22113	26467.73954	0.00%	26457.11454	26479.44295	0.08%
C_15	39758.02750	39759.60233	0.00%	39757.54750	39759.35333	0.00%

Table 5.11: Performance of ILS on dataset X instances.

Instance	15 minutes			1.5 hours		
	Best	ILS	%Diff	Best	ILS	%Diff
X_01	4014.37075	4020.20377	0.15%	4011.37641	4018.60849	0.18%
X_02	32231.43867	32237.97075	0.02%	32228.63679	32238.56698	0.03%
X_03	8104.53773	8126.47075	0.27%	8102.58962	8118.05283	0.19%
X_04	11315.94600	11354.26599	0.34%	11303.40000	11335.16399	0.28%
X_05	22858.11477	22872.21181	0.06%	22837.42068	22857.14295	0.09%
X_06	47032.95633	47035.15216	0.00%	47032.16366	47033.91750	0.00%
X_07	13221.61783	13222.48349	0.00%	13221.35849	13222.34350	0.00%
X_08	13717.37033	13726.39583	0.07%	13707.28500	13724.84233	0.13%
X_09	20195.40500	20195.99833	0.00%	20180.45000	20196.45833	0.08%
X_10	17289.31571	17302.40000	0.08%	17267.81857	17269.86714	0.01%
X_11	39121.52404	39121.53206	0.00%	39115.26526	39119.86755	0.01%
X_12	47502.80992	47582.45305	0.17%	47441.36908	47462.10419	0.04%
X_13	15784.25141	15794.64339	0.07%	15784.16933	15788.57924	0.03%
X_14	79417.02750	**79416.08194**	0.00%	79416.86527	**79414.84444**	0.00%
X_15	45491.80749	45493.04388	0.00%	45422.28999	45426.09194	0.00%

Tables 5.8 - 5.9 indicate that the proposed ILS performs very well on dataset A and B which contain mostly small and medium instances. The algorithm obtains better solution than the best known result for some instances in dataset B. Comparing the results of 15 minutes and 1.5 hours, it is noted that the improvement in solution quality is not significant.

Tables 5.10 - 5.11 indicate that the proposed ILS is less effective on the large instances of dataset C and X. This is not surprising since increasing the size of the instances will increase the computational burden of model A-MILP. Hence, achieving an initial solution takes too much time because we rely on CPLEX. Another possible reason is that the subroutine SEARCH in the ILS, which involves exhaustive search on the neighborhood by *one-shift* and *two-swap*, become time consuming for large instances. This significantly reduces the number of perturbation in the ILS procedure due to the imposed time limits. For a few instances, e.g. C_14 and X_02, running the algorithm for 1.5 hours leads to a worse solution than the result obtained in 15 minutes. The same observation was reported by Roadef Challenge in the semi-final results for instance B05 (see `https://www.roadef.org/challenge/2020/en/semifinalfresult.php`). One possible reason is that for the 1.5 hours, given a better starting solution, the perturbation mechanism was unable to escape from the local optimum.

5.7 Summary

In this chapter, we studied the grid operation-based outage maintenance planning problem which was proposed for the ROADEF/EURO challenge 2020. The problem possesses several unique features and is fundamentally different from the typical stochastic programming problems due to the quantile criterion in the objective function. Several MILP models were derived and three heuristic algorithms were developed for the considered problem.

The first MILP formulation uses binary variables as indicators to model quantile. The advantage of this model is that we can obtain exact solutions to problem instances with up to 6 scenarios in less than 100 seconds. However, the MILP becomes intractable as the number of scenarios grow very large, such as the test cases considered in this study. This is because the size of the first model is dependent on the total number of scenarios as well as the planning horizon. The second MILP formulation uses general variables to model quantile, owing to the assumption that the quantile of the sum of distributions is proportional by a fixed parameter to the sum of quantile of the individual distribution. Solving this model provides a good feasible solution which is important for the proposed heuristics and metaheuristic.

We apply the confidence (guaranteeing) approach for solving stochastic program with quantile objective function and discrete distribution of the random parameters. The confidence approach leads to two solution methods: (i) confidence-method heuristic (CM) which uses the confidence sets to measure quantile; and (ii) critical-scenario confidence-method heuristic (cs-CM) which uses the critical scenarios of the confidence sets. In our experiments, CM works well on dataset A which contains mostly small-to-medium instances but is less effective as the problem size grows much larger. On the other hand, cs-CM produces poor-quality solution in the first few iterations due to the large differences between the approximate objective function and the true objective function. As a result, cs-CM requires more iterations to improve the initial solution but each iteration needs less time than that of CM.

An iterated local search algorithm was developed for solving the maintenance planning prob-

lem. The performance of ILS on the four given datasets was compared with the best known results from the other participants. Based on the computational results, we observed that our ILS performed very well on the instances of datasets A, giving an average relative percentage difference of about 0.004% for both 15 minutes and 1.5 hours tests. The effectiveness of our ILS was more profound for instances in dataset B where we obtained better solution for 50% of instances. As the problem size increases, the growth in computation time for executing the local search procedures is significant. This, in turn, substantially reduces the number of perturbation in the ILS procedure due to the imposed time limit. As a consequence, ILS could not escape the local optimum and find high-quality solutions to the large instances in datasets C and X.

Chapter 6

Conclusion

In this thesis, we develop and solve models for several difficult real-world optimization problems, which have applications in the fields of maintenance planning. The input data of these problems can be described with a probability distribution, and the objective function is comprised of two types of penalties. For each of the problems discussed, contributions to the existing body of knowledge are summarized, and some suggestions for future research are offered.

6.1 Summary of Work

This section provides a summary of the problems discussed in Chapters 3 - 5, outlines the proposed solution approaches, and recaps the major findings.

1. In Chapter 3, we propose a nonlinear integer programming formulation and a mixed-integer linear programming relaxation based on Jensen's inequality for the high-level heavy maintenance planning problem arising in practice at an Australian rolling stock maintenance center. The formulation considers the uncertain duration of maintenance, the permissible number of out-of-service train-sets specified by the rolling stock oper-

ator, and the capacity of the maintenance center. We provide an efficient method to evaluate the objective function for any feasible solutions and incorporate this method into the optimization procedures.

Four solution approaches were presented. The first approach was a Genetic Algorithm, which includes a greedy heuristic that decodes a chromosome into a feasible solution by selecting the arrival dates according to their contributions to the number of train-sets in the maintenance center. The second approach combines the Genetic Algorithm with Sample Average Approximation method, whereby a sequence of train-sets is formed by sorting the train-sets in the same train family by their desired arrival time windows. The sequence enforces the precedence relations among train-sets to significantly reduce the solution space, enabling the problem to be solved by an exact method. The third approach incorporates the Iterated Local Search algorithm within a multi-start framework. The fourth approach was a two-stage heuristic approach that constructs an initial solution by solving the mixed-integer programming model and then attempts to improve the solution by means of local search procedure or Iterated Local Search metaheuristic.

Using real-world data provided by a maintenance center, it is shown that the two-stage heuristic approach with Iterated Local Search performed the best, followed by the Sample Average Approximation-based Genetic Algorithm approach and the multi-start Iterated Local Search. The Genetic Algorithm was the least successful among the four proposed approaches. We have demonstrated the effectiveness and efficiency of the presented approaches. Notably, at the time of implementation at the maintenance center, the decision support tool was reported to improve the maintenance planning significantly. The plan development time was reduced from several days to a few minutes.

We attribute the success of our two-stage heuristic approach to the characteristics of the method used in each stage: (1) the Jensen's Inequality based mixed-integer linear programming relaxation provides a high-quality initial solution; and (2) the application of local search with different neighborhood structures is indeed crucial for

improving the quality of candidate solutions, thus providing a reason for why the Iterated Local Search-based approach is more successful than the Genetic Algorithm-based approaches.

2. In Chapter 4, we study a scheduling problem where every job requires several types of resources and the processing time of each job follows a discrete distribution. It is assumed that job processing times are independent across jobs and that processing time realizations should be integers. The study of this problem was motivated by the fact that the processing of a job is rarely based on a specific resource but rather a composite of different types of resources (e.g., manpower, equipment and material), each with differing individual capacity.

   We propose a mixed-integer linear programming formulation to minimize the expected total tardiness and the expected total penalty for violating the capacity. Notably, the formulation is flexible, and therefore it allows for modeling different scheduling costs. We provide an efficient method to evaluate the objective function for any feasible solutions and incorporate this method into the optimization procedures. We propose a Genetic Algorithm enhanced by a local search. Unlike the sampling-based approaches in the existing literature [92], this solution method is easy to implement and far less problem-specific. The reasons for choosing GA over ILS for the considered problem are as follows: (1) based on the results of Chapter 3, it was shown that the iterated local search is sensitive to the initial solution. On the other hand, GA's parallel search mechanism has the ability to maintain the diversity of population, making it less sensitive to the initial population; (2) GA is commonly used in the literature to solve optimisation problems under uncertainty; and (3) GA is suitable for global search and thus with the help of a simple local search method, its performance can be significantly enhanced.

   Through computational experiments with a number of randomly generated test instances, we demonstrate the effectiveness and efficiency of our method. We show that by incorporating local search into a Genetic Algorithm framework, our method outperforms a direct application of Integer Programming in terms of solution time. The superior performance of our approach is also evident on large instances with up to

10^{24} scenarios where the Genetic Algorithm enhanced by local search outperforms the Sample Average Approximation method and yields confidence interval widths that are on average within 2% from the upper bound.

3. Finally, in Chapter 5, we study the grid operation-based outage maintenance planning problem, which was put forward by RTE - a major electricity transmission system operator in France - for the ROADEF/EURO challenge 2020. The problem is characterized by the time-varying durations, time-varying resource availability, potential vulnerability of the power grid due to the maintenance works, and uncertainty in future grid operations. The goal is to find a schedule of interventions (i.e. required outages due to maintenance works) to minimize the total mean cumulative risk of carrying out the maintenance and the total deviation of quantiles of the risk distributions from the mean values.

   We provide a method to approximate the quantile in the objective function and then reformulate the problem as a mixed-integer linear program. This approximation model is the core of our optimization procedures as it is the only means of obtaining initial solutions. Four solution approaches were presented. The first approach was an iterative updating heuristic that iteratively improves the estimation error of the approximated τ-quantile values. The second and third approaches, which were used in the qualification phase, are based on the *confidence method* [95]. The fourth approach, which was used in the semi-final and final phases of the competition, incorporates three local search optimization procedures, a self-adaptive perturbation, and a restart strategy in the Iterated Local Search algorithm. The reason for choosing ILS over GA for the considered problem is that it can produce high-quality solutions in 15 minutes, which helps us to achieve high score in the Roadef competition. Although the hybridization of GA with local search performs well for the problem in Chapter 4, the big challenge for applying the hybrid GA to the problem in Chapter 5 is the extensive computational effort for assessing the quality of solutions in each neighborhood.

   Using problem instances provided by the competition, we demonstrate that our proposed Iterated Local Search method outperforms the solution approaches of 11/13

qualified teams in the final phase and obtains the 2nd prize in the competition.

The success of the proposed method stems from the following key attributes: (1) the self-adaptive perturbation mechanism provides a method for adjusting the perturbation strength so as to balance between exploration and exploitation as the algorithm progresses; (2) our implementation of the local search procedures includes techniques for speeding up the check of disjunctive and resource constraints, as well as the move evaluations; and (3) the code was implemented in Cython [24] which results in a 100 times speedup over the pure Python code, thus enabling a good performance on the large instances in dataset C and anonymous dataset X.

Based on the findings in this thesis, given an optimization problem under uncertainty, we would recommend using the hybrid genetic algorithm which employs genetic algorithm to do the global search and some local search methods to do the local search. In the case that a good solution needs to be sought in a short time (e.g. 15 minutes), an iterated local search with self-adaptive perturbation mechanism should be considered.

The overall scientific discoveries from solving these three problems that could be generalized to solving other combinatorial problems are as follows. First, the exact evaluation of the objective function instead of Markov Chain Monte Carlo simulation could be generalized to more complex resource sharing problems, e.g., resource-constrained project scheduling problem. Second, the integration of genetic algorithm (GA) with local search significantly enhances the performance of traditional GA. Third, an amalgamation of deterministic MIP and adaptive local search can produce good solutions to optimization problem with uncertainty.

6.2 Future Work

In this section, we reflect on the limitations of this research, draw on the contributions to offer a number of recommendations for future research for each of the problems discussed in this thesis.

1. In Chapter 3, we addressed the rolling stock heavy maintenance planning problem through strategic planning, which aims at developing a plan of train-sets' arrivals for a year. Given that heavy maintenance is a lengthy procedure that takes more than a month, it is common to consider a year-length scale. For example, in 2017, the China Railway Society awarded a Science and Technology Prize for a case study involving 124 trains and the planning horizon of 607 days [105]. In the context of strategic planning, the maintenance center is considered as a single unit. This was done not only to increase the tractability of the considered problem but also to simplify the modeling process. A direct extension of this work could be to address the operational level of scheduling the maintenance which will consider the uncertain duration of maintenance operations and the technological specifics of the maintenance of trains such as the order of maintenance operations, space limitations and the necessity of shunting. As a starting point, the existing nonlinear integer programming model could introduce an additional subscript l to the x variables, corresponding to the starting time of a particular train-set j on operation line l. This change will lead to an increase in the number of variables and constraints, for which we anticipate additional computational challenges. As a result, the solution approaches presented in Chapter 3 may not be as attractive. For this reason, further research should investigate more efficient modeling approaches and alternative solution methods so as to achieve greater tractability as well as performance guarantees.

   In alignment with the current practice at the maintenance center, the mathematical models presented in Chapter 3 were developed based on fixed TAKT time (i.e. maintenance duration). In this context, the labor hours demand is evenly distributed across the duration to give the daily demand. The drawback of this approach is that labor resource is not used efficiently. A potential future research direction is to investigate the case with varying TAKT time, whereby the more resources we have, the less time is taken to complete a maintenance task. For example, currently, D_j is the cycle time of a particular train-set j. If instead we replace D_j by D'_j and Y_j, where D'_j is the normal cycle time of train-set j and Y_j is the units of resources allocated to train-set

j. Then, the actual cycle time of train-set j is given by the formula $D'_j - Y_j$. In the resulting problem, both the days of arrival at the maintenance center and the cycle times are decision variables, and the objective function should include an additional component representing the compression cost. We believe that the resource-dependent TAKT time approach, with the ability to affect the duration of maintenance by choice of the allocated resources, represents a promising direction for improving the efficiency of resource utilization in the maintenance center.

2. In Chapter 4, we solved the considered problem by means of a hybrid optimization procedure that combines Genetic Algorithm with a local search procedure. It may be interesting to investigate exact mathematical programming methods for the problem considered. For example, Keller and Bayraksan [92] present an L-shaped algorithm, which decomposes the considered problem into one master problem and $|\Omega|$ subproblems. At iteration i of the L-shaped algorithm, the master problem is solved for the first-stage decision variables $\mathbf{x}$, corresponding to the jobs' starting times. Then, the optimal dual solution from the subproblems is used to form a cut that is added to the master problem. One possible direction for extending the work of Keller and Bayraksan [92] is to enable both the objective function and subgradients to be evaluated exactly. Given that the objective function in our problem is a convex function but is not differentiable at $\mathbf{x}$, so there exist more than one subgradients at $\mathbf{x}$. Finding the set of subgradients of the objective function will be very challenging; however, one could employ the sum, and integral rule [32] for constructing subgradients of convex functions.

   The computational experiments showed that some good solutions could be obtained by solving with CPLEX the Sample Average Approximation problems despite the small sample size (i.e. $N = 50$). Indeed, the reported solutions were within an estimated 1.7%, 2.7%, 2.8%, and 3.1% from the lower bounds for the 20-5-2^{20}, the 40-5-2^{40}, the 60-5-2^{60}, and the 80-5-2^{80}, respectively. It is unclear whether this is the case in general or simply a result of the properties of the test instances. Future research should study the theoretical analysis for tighter sample-size bounds for SAA for the problem considered in Chapter 4. Such sample-size bounds, which are logarithmic in the problem dimension,

was proposed in [36] for the capacity- or budget-constrained optimization problems.

3. Finally, in Chapter 5, the proposed solution approaches depend on an initial feasible solution which is generated by solving with CPLEX the approximate MILP model. The computational experiments showed that this model became intractable as the problem size grew very large. Investigating more efficient methods for finding the initial solutions to the considered problem can be a direct follow-up of our work. One possible research direction is the application of Lagrangian Relaxation, whereby we can formulate the Lagrangian Relaxation model by moving either the resource or the disjunctive constraints to the objective function. However, the solutions obtained from solving the Lagrangian Relaxation model may be infeasible with respect to the original problem since some of the constraints are relaxed. Further research should investigate how to transform the obtained infeasible initial solution into a feasible one.

   Another future research opportunity is to permit infeasible solutions for the Iterated Local Search metaheuristic, particularly in the subroutines SEARCH and PERTURBATION. The success of using infeasible solutions within the ILS procedure has been demonstrated in a number of studies by our research team for the Workforce Scheduling and Routing Problem [73, 74].

Bibliography

[1] Abirami, M., Ganesan, S., Subramanian, S., and Anandhakumar, R. (2014). Source and transmission line maintenance outage scheduling in a power system using teaching learning based optimization algorithm. *Applied Soft Computing*, 21:72–83.

[2] Abiri-Jahromi, A., Fotuhi-Firuzabad, M., and Abbasi, E. (2009). An efficient mixed integer linear formulation for long-term overhead lines maintenance scheduling in power distribution systems. *IEEE Transactions on Power Delivery*, 24(4):2043–2053.

[3] Alayo, H. and Paucar, E. (2018). A milp model for maintenance scheduling in transmission systems and an example application to the peruvian system. *IEEE Latin America Transactions*, 16(4):1099–1104.

[4] Almakhlafi, A. and Knowles, J. (2015). Iterated local search for the generator maintenance scheduling problem. In *Proceedings of the 7th Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2015)*, pages 708–742, Prague, Czech Republic.

[5] Angulo, G., Ahmed, S., and Dey, S. S. (2016). Improving the integer l-shaped method. *Journal on Computing*, 28(11):483–499.

[6] Arkhipov, D., Battaïa, O., and Lazarev, A. (2019). An efficient pseudo-polynomial algorithm for finding a lower bound on the makespan for the resource constrained project scheduling problem. *European Journal of Operational Research*, 275(1):35–44.

[7] Artigues, C., Leus, R., and Nobibon, F. T. (2013). Robust optimization for resource-

constrained project scheduling with uncertain activity durations. *Flexible Services and Manufacturing Journal*, 25:175–205.

[8] Atighehchian, A., Sepehri, M. M., and Shadpour, P. (2020). A two-step stochastic approach for operating rooms scheduling in multi-resource environment. *Annals of Operations Research*, 292:191–214.

[9] Avci, M. and Topaloglu, S. (2017). A multi-start iterated local search algorithm for the generalized quadratic multiple knapsack problem. *Computers & Operations Research*, 83:54–65.

[10] Avramidis, A. N., Chan, W., Gendreau, M., L'Ecuyer, P., and Pisacane, O. (2010). Optimizing daily agent scheduling in a multiskill call center. *European Journal of Operational Research*, 200(3):822–832.

[11] Bakker, H., Dunke, F., and Nickel, S. (2020). A structuring review on multi-stage optimization under uncertainty: Aligning concepts from theory and practice. *Omega*, 96:102080.

[12] Ballestín, F. (2007). When it is worthwhile to work with the stochastic rcpsp? *Journal of Scheduling*, 10:153–166.

[13] Ballestín, F. and Leus, R. (2009). Resource-constrained project scheduling for timely project completion with stochastic activity durations. *Production and Operations Management*, 18(4):459–474.

[14] Battiti, R., Brunato, M., and Mascia, F. (2008). *Reactive Search and Intelligent Optimization*, volume 45. Springer, New York.

[15] Battiti, R. and Protasi, M. (1997). Reactive search, a history-based heuristic for max-sat. *ACM Journal of Experimental Algorithmics*, 2:2.

[16] Baum, E. B. (1986a). Iterated descent: a better algorithm for local search in combinatorial optimization problems. Technical report, California Institute of Technology, Pasadena, CA.

[17] Baum, E. B. (1986b). Towards practical "neural" computation for combinatorial optimization problems. In Denker, J., editor, *Neural Networks for Computing*, pages 53–64. AIP Conference Proceedings.

[18] Bayraksan, G. (2018). An improved averaged two-replication procedure with latin hypercube sampling. *Operations Research Letters*, 46(2):173–178.

[19] Bayraksan, G. and Morton, D. P. (2006). Assessing solution quality in stochastic programs. *Mathematical Programming*, 108(2-3):495–514.

[20] Bayraksan, G. and Morton, D. P. (2009). Assessing solution quality in stochastic programs via sampling. *INFORMS Tutorials in Operations Research*, pages 102–122.

[21] Bayraksan, G. and Morton, D. P. (2011). A sequential sampling procedure for stochastic programming. *Operations Research*, 59(4):898–913.

[22] Bayraksan, G. and Pierre-Louis, P. (2012). Fixed-width sequential stopping rules for a class of stochastic programs. *SIAM Journal on Optimization*, 22(4):1518–1548.

[23] Beale, E. M. L. (1955). On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society*, 17(2):173–184.

[24] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., and Smith, K. (2011). Cython: the best of both worlds. *Computing in Science & Engineering*, 13(2):31–39.

[25] Bertsekas, D. P. (2017). *Dynamic Programming and Optimal Control*. Athena Scientific. 4^{th} edition.

[26] Biajioli, F. L., Chaves, A. A., and Lorena, L. A. N. (2019). A biased random-key genetic algorithm for the two-stage capacitated facility location problem. *Expert Systems with Applications*, 115:418–426.

[27] Billingsley, P. (1995). *Probability and Measure*, volume 3^{rd} edition. John Wiley & Sons, New York.

[28] Birge, J. and Wets, R. (1989). Sublinear upper bounds for stochastic programs with recourse. *Mathematical Programming*, 43:131–149.

[29] Birge, J. R. (1985). Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs. *https://doi.org/10.1287/opre.33.5.989*, 33(5):989–1007.

[30] Birge, J. R. and Louveaux, F. (1997). *Introduction to Stochastic Programming*. Springer, New York. 2^{nd} edition.

[31] Biscarri, W., Zhao, S. D., and Brunner, R. J. (2018). A simple and fast method for computing the poisson binomial distribution function. *Computational Statistics & Data Analysis*, 122:92–100.

[32] Boyd, S., Duchi, J., and Vandenberghe, L. (2018). *Subgradients*. `https://stanford.edu/class/ee364b/lectures/subgradients_notes.pdf` [Accessed: 11 November 2019].

[33] Brandão, J. (2020). A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem. *European Journal of Operational Research*, 284(2):559–571.

[34] Bruni, M. E., Beraldi, P., Guerriero, F., and Pinto, E. (2011). A heuristic approach for resource constrained project scheduling with uncertain activity durations. *Computers & Operations Research*, 38(9):1305–1318.

[35] Bruni, M. E., Pugliese, L. D. P., Beraldi, P., and Guerriero, F. (2018). A two-stage stochastic programming model for the resource constrained project scheduling problem under uncertainty. *In proceedings of the 7th International Conference on Operations Research and Enterprise System (ICORES 2018)*.

[36] Bugg, C. and Aswani, A. (2021). Logarithmic sample bounds for sample average approximation with capacity- or budget-constraints. *Operations Research Letters*, 49(2):231–238.

[37] Cao, Z., Lin, C., Zhou, M., Zhou, C., and Sedraoui, K. (2022). Two-Stage Genetic Algorithm for Scheduling Stochastic Unrelated Parallel Machines in a Just-in-Time Manufacturing Context. *in IEEE Transactions on Automation Science and Engineering*.

[38] Chakrabortty, R. K., Sarker, R. A., and Essam, D. L. (2017). Resource constrained project scheduling with uncertain activity durations. *Computers & Industrial Engineering*, 112:537–550.

[39] Charnes, A. and Cooper, W. (1959). Chance-constrained programming. *Management Science*, 6(1):73–79.

[40] Charnes, A. and Cooper, W. (1963). Deterministic equivalents for optimizing and satisficing under chance constraints. *Operations Research*, 11(1):18–39.

[41] Chen, Z., Demeulemeester, E., Bai, S., and Guo, Y. (2018). Efficient priority rules for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, 270(3):957–967.

[42] Cotta, C., Mathieson, L., and Moscato, P. (2018). Memetic algorithms. *Handbook of Heuristics*, 1-2:607–638.

[43] Dalal, G., Gilboa, E., Mannor, S., and Wehenkel, L. (2019). Chance-constrained outage scheduling using a machine learning proxy. *IEEE Transactions on Power Systems*, 34(4):2528–2540.

[44] Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science*, 1(3-4):197–206.

[45] D'Ariano, A., Meng, L., Centulio, G., and Corman, F. (2019). Integrated stochastic optimization approaches for tactical scheduling of trains and railway infrastructure maintenance. *Computers & Industrial Engineering*, 127:1315–1335.

[46] De Jong, K. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan.

[47] Debels, D. and Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research*, 55(3):457–469.

[48] Denton, B., Viapiano, J., and Vogl, A. (2007). Optimisation of surgery sequencing and scheduling decisions under uncertainty. *Health Care Management Science*, 10:13–24.

[49] Denton, B. T., Miller, A. J., Balasubramanian, H. J., and Huschka, T. R. (2010). Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research*, 58(4):802–816.

[50] Doganay, K. and Bohlin, M. (2010). Maintenance plan optimization for a train fleet. *Computers in Railways XII*, 114:349–358.

[51] Dokov, S. P. and Morton, D. P. (2005). Second-order lower bounds on the expectation of a convex function. *Mathematics of Operations Research*, 30(3):662–677.

[52] Dong, X., Nowak, M., Chen, P., and Lin, Y. (2015). Self-adaptive perturbation and multi-neighborhood search for iterated local search on the permutation flow shop problem. *Computers & Industrial Engineering*, 87:176–185.

[53] Dowson, O., Morton, D. P., and Downward, A. (2022). Bi-objective multistage stochastic linear programming. *Mathematical Programming 2022*, pages 1–27.

[54] Durán, G., Rey, P. A., and Wolff, P. (2017). Solving the operating room scheduling problem with prioritized lists of patients. *Annals of Operations Research*, 258(2):395–414.

[55] Dye, S. (2009). Simple recourse problem. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*. Springer, Boston.

[56] El-Mihoub, T. A., Hopgood, A. a., Nolle, L., and Battersby, A. (2006). Hybrid genetic algorithms: A review. *Engineering Letters*, 13(2).

[57] Erschler, J. (1976). *Analysis under constraints and decision support for certain scheduling problems*. PhD thesis, University of Toulouse.

[58] Eshelman, L. J. and Schaffer, J. D. (1991). Preventing premature convergence in genetic algorithm by preventing incest. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 115–121.

[59] Fortz, B., Labbé, M., Louveaux, F., and Poss, M. (2013). Stochastic binary problems with simple penalties for capacity constraints violations. *Mathematical Programming*, 138(1-2):199–221.

[60] Froger, A., Gendreau, M., Mendoza, J., Pinson, E., and Rousseau, L. (2016). Maintenance scheduling in the electricity industry: A literature review. *European Journal of Operational Research*, 251:695–706.

[61] Fu, Y., Shahidehpour, M., and Li, Z. (2007). Security-constrained optimal coordination of generation and transmission maintenance outage scheduling. *IEEE Transactions on Power Systems*, 22(3):1302–1313.

[62] Gade, D., Küçükyavuz, S., and Sen, S. (2014). Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, 144(1-2):39–64.

[63] García-Martínez, C., Rodriguez, F. J., and Lozano, M. (2018). Genetic algorithms. In Martí, R., Pardalos, P. M., and Resende, M. G. C., editors, *Handbook of Heuristics*, pages 431–464. Springer, Cham.

[64] Giacco, G. L., Carillo, D., D'Ariano, A., Pacciarelli, D., and Marín, A. G. (2014). Short-term rail rolling stock rostering and maintenance scheduling. *Transportation Research Procedia*, 3:651–659.

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

[66] Glover, F. and Kochenberger, G. A. (2003). *Handbook of Metaheuristics*, volume 1^{st} edition. Springer, US.

[67] Glover, F. and Sörensen, K. (2015). Metaheuristics. *Scholarpedia*, 10(4):6532.

[68] Gonçalves, J. F., Mendes, J. J. M., and Resende, M. G. C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1):77–95.

[69] Gonzalez-Neira, E., Montoya-Torres, J. R., and Barrera, D. (2017). Flow-shop scheduling problem under uncertainties: review and trends. *International Journal of Industrial Engineering Computations*, 8:399–426.

[70] Gu, H., Joyce, M., Lam, H. C., Woods, M., and Zinder, Y. (2019a). A genetic algorithm for assigning train arrival dates at a maintenance centre. Paper presented at the 9th IFAC Conference on Manufacturing Modelling, Management and Control, Berlin School of Economics and Law, 28-30 August.

[71] Gu, H. and Lam, H. C. (2019). A genetic algorithm approach for scheduling trains maintenance under uncertainty. In Le Thi, H. A., Pham Dinh, T., and Nguyen, N., editors, *Advanced Computational Methods for Knowledge Engineering. ICCSAMA 2019. Advances in Intelligent Systems and Computing*, volume 1121, pages 106–118. Springer, Cham.

[72] Gu, H., Lam, H. C., and Zinder, Y. (2022). Planning rolling stock maintenance: optimisation of train arrival dates at a maintenance centre. *Journal of Industrial & Management Optimisation*, 18(2):747–772.

[73] Gu, H., Zhang, A., and Zinder, Y. (2021). An efficient optimisation procedure for the workforce scheduling and routing problem: Lagrangian relaxation and iterated local search. *Computers & Operations Research*, Preprint.

[74] Gu, H., Zhang, Y., and Zinder, Y. (2019b). Lagrangian relaxation in iterated local search for the workforce scheduling and routing problem. In Kotsireas, I., Pardalos, P., Parsopoulos, K. E., Souravlias, D., and Tsokas, A., editors, *Analysis of Experimental Algorithms. SEA 2019. International Symposium on Experimental Algorithms*, pages 527–540. Springer, Cham.

[75] Gu, J., Gu, M., Cao, C., and Gu, X. (2010). A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. *Computers & Operations Research*, 37(5):927–937.

[76] Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

[77] Harper, P. R., Powell, N. H., and Williams, J. E. (2010). Modelling the size and skill-mix of hospital nursing teams. *Journal of the Operational Research Society*, 61(5):768–779.

[78] Higle, J. L. and Sen, S. (1996). Duality and statistical tests of optimality for two stage stochastic programs. *Mathematical Programming*, 75(2):257–275.

[79] Ho, S. C. and Leung, J. M. Y. (2010). Solving a manpower scheduling problem for airline catering using metaheuristics. *European Journal of Operational Research*, 202(3):903–921.

[80] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Michigan.

[81] Homem-de Mell, T. and Bayraksan, G. (2014). Monte carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science*, 19(1):56–85.

[82] Horng, S. C., Lin, S. S., and Yang, F. Y. (2012). Evolutionary algorithm for stochastic job shop scheduling with random processing time. *Expert Systems with Applications*, 39(3):3603–3610.

[83] Jeon, G., Leep, H. R., and Shim, J. Y. (2007). A vehicle routing problem solved by using a hybrid genetic algorithm. *Computers & Industrial Engineering*, 53(4):680–692.

[84] Ji, G., Wu, W., and Zhang, B. (2016). Robust generation maintenance scheduling considering wind power and forced outages. *IET Renewable Power Generation*, 10(5):634–641.

[85] Jia, Q. and Seo, Y. (2013). An improved particle swarm optimization for the resource-constrained project scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 67(9-12):2627–2638.

[86] Jiang, Y., Zhang, Z., Mccalley, J., Van Voorhis, T., and Ames, I. A. (2006). Risk-based maintenance optimisation for transmission equipment. *IEEE Transactions on Power Systems*, 21.

[87] Jing, X., Pan, Q., and Gao, L. (2021). Local search-based metaheuristics for the robust distributed permutation flowshop problem. *Applied Soft Computing*, 105:107247.

[88] Johnson, D. S. (1990). Local optimization and the travelling salesman problem. In *In proceedings of the 17th Colloquium on Automata, Languages, and Programming. Lecture Notes in Computer Science*, volume 443, pages 446–461. Springer, Heidelberg.

[89] Kadri, R. L. and Boctor, F. F. (2018). An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *European Journal of Operational Research*, 265(2):454–462.

[90] Kall, P. and Wallace, S. W. (1994). *Stochastic programming*. Wiley, New York.

[91] Kaut, M. and Wallace, S. W. (2007). Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271.

[92] Keller, B. and Bayraksan, G. (2009). Scheduling jobs sharing multiples resources under uncertainty: A stochastic programming approach. *IIE Transactions*, 42(1):16–30.

[93] Khalid, A. and Ioannis, K. (2012). A survey of generator maintenance scheduling techniques. *Global Journal of Researches in Engineering*, 12(1).

[94] Khatami, M., Salehipour, A., and Hwang, F. J. (2019). Makespan minimization for the m-machine ordered flow shop scheduling problem. *Computers & Operations Research*, 111:400–414.

[95] Kibzun, A. I. and Kurbakovskiy, V. Y. (1991). Guaranteeing approach to solving quantile optimisation problems. *Annals of Operations Research*, 30(1-4):81–94.

[96] Kleywegt, A. J., Shapiro, A., and Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.

[97] Kolen, A. and Pesch, E. (1994). Genetic local search in combinatorial optimization. *Discrete Applied Mathematics*, 48(3):273–284.

[98] Krishnamoorthy, K. (2006). *Handbook of Statistical Distributions with Applications*. Chapman & Hall/CRC.

[99] Kulkarni, R., Khuntia, S. R., Joseph, A., Rueda, J. L., and Palensky, P. (2018). Economic outage scheduling of transmission line for long-term horizon under demand and wind scenarios. In *Proceedings of the 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe*.

[100] Lai, Y. C., Fang, D. C., and Huang, K. L. (2015). Optimizing rolling stock assignment and maintenance plan for passenger railway operations. *Computers & Industrial Engineering*, 85:284–295.

[101] Lanza, G., Crainic, T. G., Rei, W., and Ricciardi, N. (2021). Scheduled service network design with quality targets and stochastic travel times. *European Journal of Operations Research*, 288(1):30–46.

[102] Laporte, G. and Louveaux, F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142.

[103] Law, A. M. and Kelton, W. D. (2000). *Simulation Modeling and Analysis*. McGraw-Hill, New York, 5 edition.

[104] Li, H. and Demeulemeester, E. (2016). A genetic algorithm for the robust resource leveling problem. *Journal of Scheduling*, 19(1):43–60.

[105] Lin, B., Wu, J., Lin, R., Wang, J., Wang, H., and Zhang, X. (2019). Optimization of high-level preventive maintenance scheduling for high-speed trains. *Reliability Engineering & System Safety*, 183:261–275.

[106] Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, 21(2):498–516.

[107] Lourenço, H. R., Martin, O. C., and Stützle, T. (2010). Iterated local search: framework and applications. In Gendreau, M. and Potvin, J., editors, *Handbook of Metaheuristics*, volume 2^{nd} edition, pages 363–397. Springer, US.

[108] Lv, C., Wang, J., You, S., and Zhang, Z. (2014). Short-term transmission maintenance scheduling based on the benders decomposition. *International Transactions on Electrical Energy Systems*, 25(4):697–712.

[109] Mak, W., Morton, D. P., and Wood, R. K. (1999). Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1-2):47–56.

[110] Malcolm, D. G., Rooseboom, J. H., Clark, C. E., and Fazar, W. (1959). Application of a technique for research and development program evaluation. *Operations Research*, 7(5):646–669.

[111] Marin, M., Karangelos, E., and Wehenkel, L. (2017). A computational model of mid-term outage scheduling for long-term system studies. In *Proceedings of the 2017 IEEE Manchester PowerTech*, pages 1–7, Manchester, UK.

[112] Martin, O. and Otto, S. W. (1996). Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75.

[113] Martin, O., Otto, S. W., and Felten, E. W. (1991). Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5(3):299–326.

[114] Marwali, M. and Shahidehpour, S. (2000). Short-term transmission line maintenance scheduling in a deregulated system. *IEEE Transactions on Power Systems*, 15(3):1117–1124.

[115] Mazidi, P., Tohidi, Y., Ramos, A., and Sanz-Bobi, M. (2018). Profit-maximization generation maintenance scheduling through bi-level programming. *European Journal of Operational Research*, 264(3):1045–1057.

[116] Mira, L., Andrade, A. R., and Gomes, M. C. (2020). Maintenance scheduling within rolling stock planning in railway operations under uncertain maintenance durations. *Journal of Rail Transport Planning & Management*, 14:100177.

[117] Mitchell, S., O'Sullivan, M., and Dunning, I. (2011). Pulp : A linear programming toolkit for python.

[118] Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

[119] Moghaddam, B. F., Ruiz, R., and Sadjadi, S. J. (2012). Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm. *Computers & Industrial Engineering*, 62(1):306–317.

[120] Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical report, California Institute of Technology.

[121] Moscato, P. and Cotta, C. (2019). An accelerated introduction to memetic algorithms. *International Series in Operations Research and Management Science*, 272:275–309.

[122] Moscato, P. and Norman, M. G. (1992). A "memetic" approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. *In proceedings of the International Conference on Parallel Computing and Transputer Applications*, pages 177–186.

[123] Mountakis, K. S. (2013). *Stochastic Scheduling of Train Maintenance Projects*. PhD thesis, Delft University of Technology.

[124] Nemirovski, A., Ben-Tal, A., and El Ghaoui, L. (2009). *Robust Optimization*. Princeton University Press, Princeton.

[125] Neri, F. and Cotta, C. (2012). Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14.

[126] Neri, F., Cotta, C., and Moscato, P. (2012). Handbook of Memetic Algorithms. *Studies in Computational Intelligence*.

[127] Norkin, V. I., Pflug, G. C., and Ruszczynski, A. (1998). A branch and bound method for stochastic global optimisation. *Mathematical Programming*, 83:425–450.

[128] Ntaimo, L. (2013). Fenchel decomposition for stochastic mixed-integer programming. *Journal of Global Optimization*, 55(1):141–163.

[129] Özarik, S. S., Veelenturf, L. P., Woensel, T. V., and Laporte, G. (2021). Optimizing e-commerce last-mile vehicle routing and scheduling under uncertain customer presence. *Transportation Research Part E: Logistics and Transportation Review*, 148:102263.

[130] Pandžić, H., Conejo, A., Kuzle, I., and Caro, E. (2012). Yearly maintenance scheduling of transmission lines within a market environment. *IEEE Transactions on Power Systems*, 27(1):407–415.

[131] Parisio, A. and Neil Jones, C. (2015). A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand. *Omega*, 53:97–103.

[132] Peña, D., Tchernykh, A., Dorronsoro, B., and Ruiz, P. (2022). A novel multi-objective optimization approach to guarantee quality of service and energy efficiency in a heterogeneous bus fleet system. *Engineering Optimization*.

[133] Pérez, J. G., Martín, M., García, C., and Granero, M. (2016). Project management under uncertainty beyond beta: the generalized bicubic distribution. *Operations Research Perspectives*, 3:67–76.

[134] Pezzella, F., Morganti, G., and Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, 35(10):3202–3212.

[135] Pierre-Louis, P., Bayraksan, G., and Morton, D. P. (2011). A combined deterministic and sampling-based sequential bounding method for stochastic programming. In Jain, S.,

Creasey, R. R., Himmelspach, J., White, K. P., and Fu, M., editors, *Proceedings of the 2011 Winter Simulation Conference*, pages 4167–4178.

[136] Plambeck, E. L., Fu, B. R., Robinson, S. M., and Suri, R. (1996). Sample-path optimization of convex stochastic performance functions. *Mathematical Programming: Series A and B*, 75(2):137–176.

[137] Prékopa, A. (1995). *Stochastic programming*. Kluwer Academic Publishers, Dordrecht.

[138] Reeves, C. R. (1994). Genetic algorithms and neighbourhood search. In Fogarty, T. C., editor, *Evolutionary Computing*, volume 865, pages 115–130, Berlin, Heidelberg. Springer.

[139] Reihani, E., Sarikhani, A., and Davodi, M. (2012). Reliability based generator maintenance scheduling using hybrid evolutionary approach. *International Journal of Electrical Power & Energy Systems*, 42(1):434–439.

[140] Ritchie, H. and Roser, M. (2021). France: Energy Country Profile. `https://ourworldindata.org/energy/country/france` [Accessed: 03 October 2021].

[141] Robinson, S. M. (1996). Analysis of sample-path optimization. *Mathematics of Operations Research*, 21(3):513–528.

[142] Rockafellar, R. and Wets, R. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16:119–147.

[143] Rostami, S., Creemers, S., and Leus, R. (2018). New strategies for stochastic resource-constrained project scheduling. *Journal of Scheduling*, 21(3):349–365.

[144] Rubinstein, R. Y. and Shapiro, A. (1993). *Sensitivity Analysis and Stochastic Optimization by the Score Function Method*, volume 1^{st} edition. John Wiley & Sons, Chichester, England.

[145] Ruiz, M. (2020). Github Repository for ROADEF/EURO 2020 Challenge. `https://github.com/rte-france/challenge-roadef-2020/` [Accessed: 01 September 2020].

[146] Ruszczyński, A. (1999). Some advances in decomposition methodsfor stochastic linear programming. *Annals of Operations Research 1999 85:0*, 85(0):153–172.

[147] Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers and Chemical Engineering*, 28(6-7):971–983.

[148] Sauma, E. (2013). A Survey and Comparison of Optimization Methods for Solving Multi-Stage Stochastic Programs with Recourse. *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/joris.2013040102*, 4(2):22–35.

[149] Schlünz, E. and van Vuuren, J. (2013). An investigation into the effectiveness of simulated annealing as a solution approach for the generator maintenance scheduling problem. *International Journal of Electrical Power & Energy Systems*, 53:166–174.

[150] Sen, S. and Hugle, J. L. (2005). The c3 theorem and a d2 algorithm for large scale stochastic mixed integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20.

[151] Sen, S. and Sherali, H. D. (2006). Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223.

[152] Shapiro, A. (2003). Monte carlo sampling methods. *Handbooks in Operations Research and Management Science*, 10:353–425.

[153] Shapiro, A., Dentcheva, D., and Ruszczýnski, A. (2009). *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial Mathematics, Philadelphia.

[154] Shapiro, A. and Homem-de Mello, T. (1998). A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81:301–325.

[155] Shapiro, A. and Homem-De-Mello, T. (2000). On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs. *SIAM Journal of Optimization*, 11(1):70–86.

[156] Shapiro, A., Homem-de Mello, T., and Kim, J. (2002). Conditioning of convex piecewise linear stochastic programs. *Mathematical Programming*, 94:1–19.

[157] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. and Puget, J. F., editors, *Principles and Practice of Constraint Programming. CP 1998. International Conference on Principles and Practice of Constraint Programming*, volume 1520, pages 417–431. Springer, Berlin.

[158] Skytte, K. and Ropenus, S. (2005). Regulatory Review and International Comparison of EU-15 Member States. Technical report.

[159] Soares, L. C. R. and Carvalho, M. A. M. (2020). Biased random-key genetic algorithm for scheduling identical parallel machines with tooling constraints. *European Journal of Operational Research*, 285(3):955–964.

[160] SPCHP (2021). Stochastic Programming Society. `https://www.stoprog.org/` [Accessed: 06 September 2021].

[161] Sriskandarajah, C., Jardine, A. K. S., and Chan, C. K. (1998). Maintenance scheduling of rolling stock using a genetic algorithm. *Journal of the Operational Research Society*, 49(11):1130–1145.

[162] Stork, F. (2001). *Stochastic resource-constrained project scheduling*. PhD thesis, Technische Universität Berlin.

[163] Sydney Trains (2018). *Sydney Trains Annual Report 2017-18*. `https://www.transport.nsw.gov.au/news-and-events/reports-and-publications/sydney-trains-annual-reports` [Accessed: 15 August 2019].

[164] Topaloglu, H. (2009). A tighter variant of jensen's lower bound for stochastic programs and separable approximations to recourse functions. *European Journal of Operational Research*, 199(2):315–322.

[165] Toubeau, J., Pardoen, L., Hubert, L., Marenne, N., Sprooten, J., De Grève, Z., and Vallée, F. (2022). Machine learning-assisted outage planning for maintenance activities in power systems with renewables. *Energy*, 238:121993.

[166] Tseng, L. Y. and Lin, Y. T. (2009). A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 198(1):84–92.

[167] Ulder, N. L. J., Aarts, E. H. L., Bandelt, H. J., van Laarhoven, P. J. M., and Pesch, E. (1991). Genetic local search algorithms for the traveling salesman problem. In Schwefel, H. P. and Männer, R., editors, *Parallel Problem Solving from Nature. PPSN 1990. Lecture Notes in Computer Science*, volume 496. Springer, Berlin.

[168] Ünal, H. T. and Başçiftçi, F. (2020). Using evolutionary algorithms for the scheduling of aircrew on airborne early warning and control system. *Defence Science Journal*, 70(3):240–248.

[169] Valente, J. M. S., Moreira, M. R. A., Singh, A., and Alves, R. (2011). Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs. *The International Journal of Advanced Manufacturing Technology*, 54(1-4):251–265.

[170] Valls, V., Ballestín, F., and Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 185(2):495–508.

[171] van der Laan, N. and Romeijnders, W. (2020). A converging benders' decomposition algorithm for two-stage mixed-integer recourse models. `http://www.optimization-online.org/DB_HTML/2020/12/8165.html` [Accessed: 10 September 2020].

[172] Van Slyke, R. M. and Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17(4):638–663.

[173] Vásquez, S. A., Angulo, G., and Klapp, M. A. (2021). An exact solution method for the tsp with drone based on decomposition. *Computers & Operations Research*, 127:105127.

[174] Vela, C. R., Varela, R., and Gonzalez, M. A. (2010). Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. *Journal of Heuristics*, 16(2):139–165.

[175] Wang, K. J., Wang, S. M., and Chen, J. C. (2008). A resource portfolio planning model using sampling-based stochastic programming and genetic algorithm. *European Journal of Operational research*, 184(1):327–340.

[176] Wang, Y., Zhong, H., Xia, Q., Kirschen, D. S., and Kang, C. (2016). An approach for integrated generation and transmission maintenance scheduling considering n-1 contingencies. *IEEE Transactions on Power Systems*, 31(3):2225–2233.

[177] Waskom, M. (2017). *Seaborn: v0.8.1.* `https://seaborn.pydata.org` [Accessed: 15 August 2019].

[178] Watson, J. P., Rana, S., Whitley, L. D., and Howe, A. E. (1999). The impact of approximate evaluation on the performance of search algorithms for warehouse scheduling. *Journal of Scheduling*, 2(2):79–98.

[179] Xie, F., Potts, C. N., and Bektaş, T. (2017). Iterated local search for workforce scheduling and routing problems. *Journal of Heuristics*, 23:471–500.

[180] Yamada, T. and Nakano, R. (1992). A genetic algorithm applicable to large-scale job-shop problems. Paper presented at the Parallel Problem Solving from Nature 2, Belgium, 28-30 September.

[181] Yoshitomi, Y. and Yamaguchi, R. (2003). A genetic algorithm and the monte carlo method for stochastic job-shop scheduling. *International Transactions in Operational Research*, 10(6):577–596.

[182] Younis, M. T. and Yang, S. (2018). Hybrid meta-heuristic algorithms for independent job scheduling in grid computing. *Applied Soft Computing*, 72:498–517.

[183] Yun, Y., Gen, M., and Seo, S. (2003). Various hybrid methods based on genetic algorithm with fuzzy logic controller. *Journal of Intelligent Manufacturing*, 14:401–419.

[184] Yun, Y. and Moon, C. (2003). Comparison of adaptive genetic algorithms for engineering optimization problems. *International Journal of Industrial Engineering: Applications and Practice*, 10:584–590.

[185] Zaman, F., Elsayed, S., Sarker, R., Essam, C., and Coello Coello, C. A. (2021). An evolutionary approach for resource constrained project scheduling with uncertain changes. *Computers & Operations Research*, 125:105104.

[186] Zaman, M. F., Elsayed, S., Ray, T., and Sarker, R. (2018). Scenario-based solution approach for uncertain resource constrained scheduling problems. *In proceedings of the 2018 IEEE Congress on Evolutionary Computation*, pages –.

[187] Zhang, C. and Li, Y. F. (2021). Imperfect Maintenance Optimization of Multi-State Rolling Stocks Based on Deep Reinforcement Learning. *Proceedings - 2021 3rd International Conference on System Reliability and Safety Engineering, SRSE 2021*, pages 265–269.

[188] Zhang, M., Hong, Y., and Balakrishnan, N. (2018). An algorithm for computing the distribution function of the generalized poisson binomial distribution. *Journal of Statistical Computation and Simulation*, 88(8):1515–1527.

[189] Zheng, F., Man, X., Chu, F., Liu, M., and Chu, C. (2019). A two-stage stochastic programming for single yard crane scheduling with uncertain release times of retrieval tasks. *International Journal of Production Research*, 57(13):4132–4147.

[190] Zhong, Q., Lusby, R. M., Larsen, J., Zhang, Y., and Peng, Q. (2019). Rolling stock scheduling with maintenance requirements at the chinese high-speed railway. *Transportation Research Part B: Methodological*, 126:24–44.

[191] Zimmermann, H. J. (1991). *Fuzzy set theory and its application.* Kluwer Academic Publishers, Boston. 2^{nd} edition.