ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

MASTER THESIS ECONOMETRICS & MANAGEMENT SCIENCE

TRACK ANALYTICS & OPERATIONS RESEARCH IN LOGISTICS

An ALNS Heuristic for the Dynamic Railway Crew Planning Problem

Author

R.W.J.M. KOOT [507379]

Supervisor NS

J. VAN 'T WOUT, MSC

Supervisor EUR

PROF. DR. D. HUISMAN

Co-reader

B.T.C. VAN ROSSUM, MSC

A PROJECT IN COLLABORATION WITH NETHERLANDS RAILWAYS

June 14, 2023

Abstract

Obtaining diverse and efficient crew rosters is a major challenge in the railway industry. Netherlands Railways (NS), the main Dutch railway operator, currently balances the amount of attractive and unattractive work over the crew bases by means of the Sharing-Sweet-and-Sour rules. A new optimization approach, the Dynamic Railway Crew Planning Problem with Fairness over Time, is proposed to apply these rules to the individual crew rosters in order to enhance the fairness among crew members. Due to its complexity, however, the problem is now solved separately for decompositions of the network. At the same time, the solutions are constructed very greedily. To cope with this, we introduce an Adaptive Large Neighborhood Search meta-heuristic that improves the fairness of the rosters by modifying the constructed solution. We apply the heuristic to three practical instances from NS, and show that the total amount of unattractive work can be substantially reduced. Moreover, the heuristic can be applied to larger instances than before, where we show that attractive and unattractive work can be exchanged throughout the network.

Keywords: Railway Crew Planning Problem, Fairness over Time, Adaptive Large Neighborhood Search, Meta-Heuristic

Acknowledgments

Finishing this thesis marks the end of my career as a student at Erasmus University Rotterdam. With lots of pleasure, I have pursued my Bachelor and Master degree in econometrics.

First of all, I would like to thank Joël and Dennis for supervising me on the thesis. Your thorough feedback and our regular discussions have contributed to the eventual result, and to me enjoying the process of writing it. I would also like to thank Bart for taking place in the thesis committee.

Furthermore, my thanks go to NS for providing me the opportunity to write the thesis, and to my colleagues at π in particular. I enjoyed our monthly game nights and re-introducing the lunch walks with our team. To everyone from team BOS, thanks for all little talks on related and unrelated topics. I especially would like to thank Marije. Starting at the same time and on the same project has been pleasant, as we commonly exchanged views on certain aspects.

Finally, I am grateful to my friends and family – my parents in particular – for their support. The years I fulfilled as a student have sometimes been tough as well, being required to study from home for a substantial period. The result of me reaching this point would not have been achievable without them.

Robin Koot
Waddinxveen, June 2023

Contents

Chapter 1

Introduction

The Netherlands has a dense railway network. Netherlands Railways (NS) is the main Dutch railway operator and provides transportation for over a million passengers every day. The operation of each of the trains is facilitated by one or more crew members. Thus, crew members constitute key resources for the railway operator. In order to employ them in the most efficient way, a substantial effort is put in generating their duties and rosters.

Hence, the construction of efficient yet diverse rosters is of great importance, and the corresponding planning problems are approached using advanced Operations Research techniques. However, due to the computational complexity of these problems, solutions obtained for practical instances are likely to be sub-optimal. Current research at NS proposes an alternative crew planning approach to improve the quality of the rosters from a fairness perspective. This thesis appends to this approach by presenting a heuristic framework that is applied to a set of initially generated rosters for distinct regions of the network. The heuristic aims to identify modifications to the joint solution that lead to overall improvements.

In the remainder of this chapter, we first give an overview of the various planning problems that are encountered at NS. Next, we describe the current crew planning approach, as well as the proposed alternative. Then, we summarize the contributions of this thesis. Lastly, the structure of the thesis is presented.

1.1 Planning Problems at Netherlands Railways

The planning process at NS comprises a number of interrelated planning problems (Huisman et al., 2005). These can be generalized into three categories: problems regarding the timetable, the rolling stock, and the crew.

Timetable-related planning problems include deciding on the length and frequency of the operated lines, as well as the exact departure and arrival times of the trains. After the timetabling decisions have been made, the *rolling stock* necessary for the train movements is planned. This also includes determining the compositions and corresponding (de)couplings. Finally, when having obtained the rolling stock circulation plan, the *crew* is scheduled accordingly.

All these planning problems undergo a number of subsequent steps, each adopting a different focus to the problem. That is, all problems are faced on a strategic, tactical, and operational level. Furthermore, beyond the regular planning, disruption management is applied to retain a feasible planning on the day of operation.

- **Strategic planning:** In this phase, long-term decisions are made regarding the total product that is offered to the passenger. Important aspects include the management of rolling stock, crew, and the network structure. Decisions made in the strategic planning phase usually span several years to decades.
- **Tactical planning:** In this phase, the decisions from the strategic phase are evolved at the network level. A tactical planning is made about once a year. For NS, this is strongly associated with the introduction of the new timetable in December.
- **Operational planning:** This phase is performed several times per year. About once every two months, an update to the tactical planning is made in which structural adjustments are implemented wherever required. Moreover, due to planned construction works and scheduled events, additional situation-specific modifications are carried out at the daily level.

This thesis further focuses on the crew planning problem in the operational phase. In particular, it complements existing research to a new crew planning approach at NS, aiming to improve the fairness of the current implementation. Before we dive into the details of this new planning approach, we first describe the current crew planning process in the next section.

1.2 Current Crew Planning Process

In the current crew planning process at NS, a work schedule is constructed for the guards and the drivers. For both groups this is done in a similar, yet separate way. The crew planning is conducted in two steps, a scheduling step followed by a rostering step.

First, in the scheduling step the total work to be performed is composed into *duties*. Each duty comprises the work for a single crew member for a single day. The scheduling typically takes place in the tactical planning phase. In particular, the so-called Sharing-Sweet-and-Sour (Dutch: *Lusten-en-Lasten-Delen*) rules are considered (Abbink et al., 2005). These rules were introduced by NS as a result of nationwide strikes in 2001 against a proposed new crew planning method that was feared to decrease the variation of the work. The rules intend to distribute the attractive (sweet) and unattractive (sour) work over the crew bases in a fair manner. For a detailed explanation of the implementation of the Sharing-Sweet-and-Sour rules in the current crew planning approach at NS, the reader is referred to Abbink et al. (2005) and Abbink et al. (2011).

Subsequently, the work is distributed over the roster groups at each of the 28 crew bases throughout the country. This step is called crew rostering, and is performed in the operational planning phase. Each roster group is assigned a set of duties, often covering the work for multiple weeks. The individuals in a roster group perform this work in a cyclic manner. The assignment of the duties to the crew members is performed by the roster groups themselves, and exact procedures differ among the crew bases. Frequently, however, this assignment is based on a seniority principle.

1.3 A New Approach to Crew Planning

From a fairness perspective, the current crew planning approach faces two main drawbacks: First, since the Sharing-Sweet-and-Sour rules are secured at the crew base level while the assignment of work to individuals takes place at a later time, it is not guaranteed that they also hold for each crew member individually. Second, due to the necessity of modifications to the planning in the operational phase, the content of the planned work often does not equal the work that is actually performed on the day of operation.

For this reason, NS recently started a research project called *Fundamenteel Spoor* to introduce a new way of generating the crew rosters. The proposed crew planning approach aims to overcome the currently experienced difficulties by taking into account the Sharing-Sweet-and-Sour rules at the individual level and based on the work that is actually performed. To that end, the order of generating and assigning the work is essentially reversed.

First, a capacity planning is made in the tactical planning phase. This planning assigns all crew members to *templates*. These templates indicate the days and time blocks on which the crew member will be working. The time blocks are typically slightly longer than the actual duty, as to allow for more flexibility during the planning process. The templates are assigned in the tactical planning phase, thereby enabling the possibility to give crew members an indication of their work schedule well in advance.

Next, in the operational planning phase, duties are generated and assigned to individual crew members according to their templates. This allows to control the assignment of attractive and unattractive work to each individual. Moreover, since the duties are constructed shorter before the day of operation, the most actual work can be included. This way, the discrepancy between planned work and performed work is reduced. The corresponding optimization problem is named the Dynamic Railway Crew Planning Problem with Fairness over Time, and aims to find for all crew members a planning that is balanced over a certain time period. The problem is modeled as a set covering problem with additional constraints and solved by means of a column generation heuristic. The fairness over time is assured by a feedback penalty mechanism.

1.4 Contributions

The computational complexity of the Dynamic Railway Crew Planning Problem with Fairness over Time, however, complicates solving practical instances exactly within a reasonable computation time. For this reason, the proposed solution approach is applied only to decompositions of the network. In each decomposition, duties are constructed in a very greedy manner, which leaves part of the work eventually unassigned. Therefore, in this thesis we propose an Adaptive Large Neighborhood Search meta-heuristic framework to improve the quality of the rosters obtained by the Dynamic Railway Crew Planning Problem with Fairness over Time. The heuristic modifies the incumbent solution by employing three destroy methods and one repair method.

The contributions of this thesis are threefold. First, we introduce the Adaptive Large Neighborhood Search heuristic which improves the fairness of the rosters in a dynamic fashion. The heuristic is applied directly to solutions obtained by the Dynamic Railway Crew Planning Problem with Fairness over Time and can be tuned towards the practical needs and desires of NS. Second, we compare the performance of the heuristic to the solution obtained by the new crew planning approach on two medium-sized instances from NS. We show that this solution can be improved by assigning a significant share of the work initially left unassigned. In addition, the amount of unattractive work can be reduced in a relatively short computation time by replanning a dedicated set of duties. Third, we apply the heuristic to a large instance, which previously could not be solved. We show that the initial solution can be substantially improved by exchanging attractive and unattractive work throughout the network. These improvements cannot be explored by the Dynamic Railway Crew Planning Problem with Fairness over Time alone.

1.5 Thesis Outline

The structure of this thesis is as follows. In Chapter 2, we give a detailed description of the Dynamic Railway Crew Planning Problem with Fairness over Time. In Chapter 3, the relevant literature is discussed. In Chapter 4, a mathematical formulation is given for the new crew planning approach. Subsequently, in Chapter 5, we present an Adaptive Large Neighborhood Search heuristic to enhance the quality of the crew rosters, which is applied to real-life instances from Netherlands Railways in Chapter 6. Finally, we conclude the thesis in Chapter 7.

Chapter 2

Problem Description

In this chapter, we provide a thorough description of the proposed new crew planning approach at NS. First, we introduce the relevant terminology to properly discuss the crew planning framework in Section 2.1. Then, the Dynamic Railway Crew Planning Problem with Fairness over Time is presented in Section 2.2.

2.1 Crew Planning Terminology

Crew planning essentially entails the process of generating a set of duties such that all tasks can be performed, and assigning these duties to the guards and the drivers. Here, a *task* is defined as a piece of work to be performed by a crew member. The task can be identified by the times and stations at which it starts and ends, respectively. A feasible sequence of tasks on a working day represents a duty.

A *duty* should satisfy a number of rules. First, all adjacent tasks within a duty must connect at the same station. Second, each duty must start and end at the same station, specifically at one of the 28 crew bases throughout the country. Furthermore, the duty must contain a meal break of at least half an hour at one of the designated relief locations in the network, roughly in the middle of the duty. The maximum length of a duty depends on its starting time, where duties starting during the day may be longer than duties performed overnight. Next, there must be a transfer time between two adjacent tasks conducted on different rolling stock units. Lastly, each duty must start and end with a predefined sign-on or sign-off time, in which the crew member can hand-over the train and read up on the actual particularities in the network. For drivers, an additional requirement is that their duty may only contain tasks on the routes and with the rolling stock types for which they are licensed.

The set of duties assigned to a single crew member constitutes his *roster*. The crew rosters in turn also have to comply with certain rules. First, the minimum rest time between two duties must be at least twelve hours. Second, additional constraints are imposed on the number of working days in a given period, according to the labor legislations and collective agreements. These also include agreements upon the number of days and weekends off. Third, it is desired that the crew rosters adhere to so-called rotation constraints. These imply that the starting times of successive early duties are non-decreasing, whereas the opposite holds for the end times of successive late duties.

At NS, an additional requirement is that the crew rosters pursue a similar degree of fairness. The requirements for this fairness are captured in the Sharing-Sweet-and-Sour (SS&S) rules (Abbink et al., 2005). The fairness of a roster is evaluated based on a number of *attributes*, associated with each SS&S rule. In addition, it is desired that the average duty length should be similar for every crew member. The following attributes are considered in the crew planning process at NS:

- **Type-A:** the fraction of 'pleasant' work in a duty. This consists, among others, of work on Intercity trains or on modern rolling stock units.
- **Aggression:** the fraction of work on trains with a lot of anticipated passenger aggression. The decision to regard a train as such is derived from historical observations.
- **Double Decker:** the fraction of work performed on double-decker trains. In these trains, guards need to climb a large number of stairs, which is considered unattractive.
- **Unique Kilometers:** the number of unique network kilometers within a certain time span. A roster is considered diverse if a crew member works on various parts of the network.
- **Repetition Within Duty (RWD):** similarly, within a single duty it is desired to work on different routes. The RWD attribute counts the number of unique segments within a duty.
- **Duty Length:** the difference between the end time and start time of a duty. Although strictly speaking this is not considered an SS&S rule, it is taken into account to balance the crew member's average duty length.

For each task, duty, and roster a score can be computed with respect to all attributes, representing its relative attractiveness. At the same time, for each of the attributes a threshold value on the scores is determined. These thresholds represent lower or upper bounds on the attractive or unattractive work, respectively. At NS, the SS&S rules for Aggression and Double Decker are considered for the guards only.

2.2 Dynamic Railway Crew Planning Problem

In order to assess the fairness of the rosters at the individual level, and based on the work that is actually performed, NS proposes a new approach to crew planning as part of the *Fundamenteel Spoor* research project. This approach has been introduced by Van Rossum et al. (2022) as the Dynamic Railway Crew Planning Problem (DRCPP) with Fairness over Time.

As input to the problem, we assume that a template-based capacity planning has been made in the tactical planning phase. In constructing the templates, the number of crew members available at each crew base has been taken into account in order to assure the later requirement that a duty must start and end at a specific crew base. Subsequently, each crew member has been assigned a number of templates with dedicated time blocks. We assume that the assignment of the templates to the crew members, as well as the start and end times of the template time blocks, cannot be changed. We also assume that this assignment complies with all relevant roster rules. That is, the rules on the number of days off, the minimum rest time between two shifts, and the rotation constraints on the start and end times are satisfied for all crew members.

The aim of the DRCPP, then, is to determine the exact content of the duties, and to assign these duties to the templates. The goal of this process is to assign the duties such that the SS&S rules are satisfied for all crew members individually. This assignment is performed in the operational planning phase, a number of weeks prior to the day of operation, such that additional tasks not known in the tactical phase can be incorporated in the planning. By assigning the duties to crew members, rather than crew bases, the compliance with the SS&S rules can be assessed at individual level.

Additionally, since the work is assigned later than before, the fairness of the work actually performed can be controlled over time. This is done using a rolling horizon approach, where it is imposed for a crew member that the SS&S rules should be satisfied over a certain (rolling) window. In other words, the characteristics of the recent work performed by a crew member to a certain extent determine the content of his next duties. Both the fact that the set of tasks to be performed is dynamically revealed, as well as that the history of the work is taken into account when constructing new duties, lead to a dynamic planning approach in which the fairness of the planing is considered over time.

The scope of the crew planning problem under consideration in this thesis is as follows. We consider the problem of constructing and assigning duties to crew members specifically in the operational planning phase. That is, we consider the template assignment that is performed in the tactical planning phase as input to our problem. Likewise, the modification of duties during disruption management is out-of-scope. Moreover, we solely consider the guards and drivers performing tasks on passenger trains on the Dutch national railway network (i.e. international trains and their personnel, as well as freight trains, are disregarded). Lastly, since the planning for shunting work is separated from the planning of daily operations, this is also not considered.

Chapter 3

Literature Review

In recent years, a renewed view within Operations Research has gained popularity. Rather than efficiency as sole objective, several studies have been initiated to capture the effect of fairness in resource allocation problems. Bertsimas et al. (2011) consider a framework for organ allocation that maximizes medical efficiency under certain fairness constraints. The required compromises are more extensively studied by several authors. Bertsimas et al. (2012) discuss the trade-off between fairness and efficiency, and propose various objectives to incorporate this balance in resource allocation problems. The corresponding losses are expressed by means of the *price of fairness* and *price of efficiency*, respectively. Similarly, Breugem et al. (2022a) analyze the trade-off between fairness and attractiveness in a fairness-oriented crew rostering problem for NS.

Along with this, Lodi et al. (2022) introduce the concept of *fairness over time* to achieve fairness in an ambulance allocation problem that is repeatedly solved. The authors present structural results on fair feasible allocation policies, which are identified by column generation and constraint programming techniques. Bampis et al. (2018) focus on both the quality and stability of the resource allocation over time. The authors consider multiple set-ups with different look-ahead information. Allocations that are feasible may vary over time, but can be identified in polynomial time. In a similar context, Van Rossum et al. (2022) solve an $\mathcal{NP}$-hard railway crew scheduling problem, where fairness over time is attained by means of a rolling horizon approach with feedback mechanism.

As the proposed crew planning framework at NS, the Dynamic Railway Crew Planning Problem with Fairness over Time introduced in Van Rossum et al. (2022) belongs to the general class of crew planning problems. The vast majority of the corresponding literature discusses applications in the railway, airline, and freight transportation industry. With regard to the railway industry, Abbink et al. (2018) provide an extensive introduction to crew planning. The crew planning problem is often decomposed into a scheduling and a rostering problem. Heil et al. (2020) survey the railway crew scheduling problem, while Van den Bergh et al. (2013) present an overview of different approaches to crew rostering problems.

In crew scheduling problems, the aim is to generate a set of feasible duties from a predefined set of tasks. Various modeling approaches exist, although most applications in the railway industry are formulated as a set partitioning problem or set covering problem (Heil et al., 2020). The latter brings the advantage that over-coverage is allowed, facilitating efficient deadheading opportunities. Although the crew scheduling problem itself is extensively studied, relatively little research is conducted that incorporates the fairness aspect.

The current crew scheduling approach in use at NS is based on the set covering formulation proposed by Abbink et al. (2011). A combination of Lagrangian heuristics, column generation, and fixing techniques is used to solve practical instances. The authors incorporate fairness at the group level by imposing additional constraints on the maximum amount of unattractive work for each crew base. On the other hand, Jütte et al. (2017) use soft constraints to penalize disproportional deviations in the share of unpopular duties between crew bases. The authors apply their method to instances for a large railway freight carrier, and show that a fair division of the work over the crew bases is obtained. Potthoff et al. (2010) show that similar column generation techniques can be applied to crew rescheduling executed in disruption management. However, due to the limited time at hand, the authors disregard fairness for evaluating the alternative schedules.

Crew rostering considers assigning the previously generated duties to individual crew members. Since the specific roster requirements and objectives strongly diverge across different industries, a wide variety of methods is applied (Van den Bergh et al., 2013). In contrast to crew scheduling, fairness is more prominently considered in crew rostering problems. A review of the available literature on personnel scheduling with fairness aspects is given by Wolbeck (2019).

Hartog et al. (2009) describe the current rostering approach at NS, where feasible cyclic rosters are constructed by means of a binary programming formulation. The fairness has already been incorporated in the crew schedules by the SS&S rules, and thus is propagated to the rosters. Other articles enhance roster fairness by incorporating personnel preferences. Er-Rbib et al. (2021) propose a mixed-integer linear program to assign duties to cyclic rosters for bus drivers. They use both hard and soft constraints to obtain balanced rosters and to incorporate group-based driver preferences, respectively. The authors propose two matheuristic approaches that effectively reduce computation times for large instances. Similarly, Maenhout and Vanhoucke (2010) incorporate crew preferences in an airline crew rostering problem that is heuristically solved. Fairness is attained by penalizing rosters that deviate from the average.

Although crew scheduling and crew rostering are often considered separately, studies that integrate the two problems do exist. Borndörfer et al. (2017) study the integrated duty scheduling and rostering problem, where they introduce *duty templates* to link duties and rosters. The problem is solved by Benders decomposition techniques. Also Van Rossum et al. (2022) integrate crew scheduling and rostering in a template-based approach, where duties are dynamically assigned using a column-generation based fixing heuristic with penalty mechanism.

Due to the computational complexity of $\mathcal{NP}$-hard optimization problems, a trade-off between solution time and solution quality is generally necessary for instances of practical size. To amend the incumbent solution, a wide variety of (meta-)heuristics is used. Shaw (1998) introduced Large Neighborhood Search (LNS), where the solution is repeatedly *destroyed* (relaxed) and *repaired* (re-optimized) by applying dedicated heuristics. Ropke and Pisinger (2006) extended this framework to an Adaptive Large Neighborhood Search (ALNS) by allowing the use of multiple destroy and repair methods.

Although (A)LNS heuristics used to be primarily used in vehicle routing problems (Pisinger & Ropke, 2007), over the years their applications have been extended towards other fields. Perumal et al. (2021) use an ALNS framework along with branch-and-price heuristics to improve the initial solution to the integrated electric vehicle and crew scheduling problem. Moreover, do Carmo Martins and Silva (2019) consider a crew scheduling problem of urban buses that most resembles our setting. The use of the removal and insertion heuristics is controlled by dynamically adjusted weights, representing the relative efficiency of each method. In their case, Adaptive Large Neighborhood Search is applied to a static problem and aims to enhance schedule efficiency. In our case, conversely, we consider a dynamic version of the problem which aims to improve fairness over time.

This thesis complements the existing literature as we design an ALNS heuristic framework to improve the solution to the Dynamic Railway Crew Planning Problem with Fairness over Time. For problems with these specific characteristics, to the best of our knowledge we are the first to apply such a framework. Specifically, in order to enhance the fairness of the resulting rosters, the heuristic directly exploits the dynamic nature of the underlying problem.

Chapter 4

Solving the Dynamic Railway Crew Planning Problem

In this chapter, we present a mathematical formulation and solution approach to solve the Dynamic Railway Crew Planning Problem with Fairness over Time. In order to provide a clear overview of the proposed crew planning approach at NS, we acknowledge the ideas in this chapter are summarized from Van Rossum et al. (2022).

We first present a mathematical formulation to the Dynamic Railway Crew Planning Problem in Section 4.1. Section 4.2 describes how the problem is solved using a column generation heuristic. Lastly, in Section 4.3, we discuss how fairness over time is incorporated in the problem.

4.1 Model Formulation

One of the main adjustments in the DRCPP compared to the current planning approach is that the schedules are constructed closer to the day of operation. This allows to incorporate the additional tasks that are revealed over time, thereby reducing the gap between planned work and executed work. In order to consider this dynamic aspect of the problem, we construct the crew planning based on a rolling planning horizon. We solve the DRCPP for each day in the planning period separately.

For each day t in the planning period a template-based capacity planning is available, indicating which crew members are scheduled to work on this day. We denote by R_t the set of these crew members. Similarly, the tasks to be performed on this day are given by K_t. The exact collection of tasks to be planned is dynamically revealed across the planning period, but is assumed to be completely known when the problem for the corresponding day is solved. Moreover, we aggregate all tasks an individual crew member can perform between two adjacent stations with a relief opportunity. At these locations, the crew member can hand-over his train to another crew member. By this, we can substantially reduce the size of the task set.

Given the sets of available crew members and tasks on day t, we denote for each crew member $r \in R_t$ the set of feasible duties by Δ_t^r. A duty is feasible for a specific crew member if the duty itself is feasible, and it adheres to the template time block assigned to the crew member. To each such duty $\delta \in \Delta_t^r$, a cost c_{rt}^δ is associated. This cost can be regarded as the penalty of assigning a duty with the respective content to this crew member. In addition, the parameter a_{tk}^δ indicates whether task $k \in K_t$ is contained in duty δ. Introducing the decision variable x_{rt}^δ, equaling 1 if duty δ is selected for crew member r on day t, we can formulate the DRCPP for day t by means of the following binary linear program:

$$\min \sum_{r \in R_t} \sum_{\delta \in \Delta_t^r} c_{rt}^\delta x_{rt}^\delta \tag{4.1}$$

$$\text{s.t.} \sum_{r \in R_t} \sum_{\delta \in \Delta_t^r} a_{tk}^\delta x_{rt}^\delta \geq 1 \qquad \forall k \in K_t \tag{4.2}$$

$$\sum_{\delta \in \Delta_t^r} x_{rt}^\delta = 1 \qquad \forall r \in R_t \tag{4.3}$$

$$x_{rt}^\delta \in \mathbb{B} \qquad \forall r \in R_t, \delta \in \Delta_t^r. \tag{4.4}$$

The objective (4.1) of the DRCPP is to minimize the costs associated with the selected duties. Constraints (4.2) ensure that each task is assigned to at least one duty. Allowing tasks to be covered by multiple duties may create efficient deadheading opportunities for a crew member. Constraints (4.3) ensure that each crew member assigned a template is assigned exactly one duty. Note that this may also be an 'empty' duty, for which the crew member is assumed to function as a reserve. Finally, constraints (4.4) denote the binary domain of the decision variables.

4.2 Solving the DRCPP by Column Generation

The presented mathematical formulation for the DRCPP selects a number of duties, covering the work for a day, against minimum cost. The number of feasible duties, and thus the potential number of decision variables, is exponential in the number of tasks. For either the guards or the drivers, practical instances for NS contain up to 18,000 tasks for over 1,300 duties (Abbink, 2014). It would require too much computational complexity to solve these problems exactly. For this reason, the DRCPP is solved by means of a column generation heuristic.

In particular, we solve the linear programming relaxation of the DRCPP by column generation. That is, we relax constraints (4.4) to $x^{\delta}_{rt} \geq 0$. The corresponding Restricted Master Problem (RMP) is initialized using slack variables for constraints (4.2) and (4.3). In each iteration, the RMP is solved to optimality and a series of pricing sub-problems is solved. The aim of these sub-problems is to find columns, representing feasible duties, having a negative reduced cost. Denoting by $\lambda_{tk} \in \mathbb{R}_{\geq 0}$ and $\pi_{rt} \in \mathbb{R}$ the dual variables corresponding to constraints (4.2) and (4.3) respectively, the reduced cost of a duty x^{δ}_{rt} becomes

$$c^{\delta}_{rt} - \sum_{k \in K_t} \lambda_{tk} a^{\delta}_{tk} - \pi_{rt}.$$

New negative reduced cost columns are obtained by solving a series of pricing sub-problems, one for each template. Each of the sub-problems can be modeled as a resource constrained shortest path problem on a directed and acyclic graph. The nodes on this graph correspond to all tasks that can be performed by the crew member assigned to this template, within the given template time block. The arcs indicate which task connections are feasible. The arc costs are given by the respective dual variables. To quickly find a path with negative reduced cost, we apply a heuristic labeling algorithm with completion bounds as in Breugem et al. (2022b). As a consequence, the RMP need not be solved to optimality when the column generation heuristic terminates.

Additionally, three acceleration techniques are implemented in order to speed up computation. First, the column generation terminates when the percentual decrease in objective value in two subsequent iterations is less than a predefined threshold ϵ. Second, we limit the columns added to the RMP in each of the pricing sub-problems to those generated that are sufficiently task-disjoint (Breugem et al., 2022b). Lastly, column management is applied to reduce the size of the RMP. Specifically, columns having a sufficiently large reduced cost for a predefined number of iterations are removed.

The solution obtained when the column generation heuristic terminates is fractional. We convert this solution to the DRCPP into a binary one by applying the following greedy fixing rule. All duties with a solution value higher than a predefined threshold $\tau > \frac{1}{2}$, if any, are fixed to 1. If no such duties exist, we only fix the duty with the highest value. We then solve the remaining problem again, until the solution is completely binary. We then proceed with solving the DRCPP for the next day in our planning horizon.

4.3 Incorporating Fairness over Time

On itself, the DRCPP thus constructs a feasible crew planning for the rolling planning horizon. That is, per day, a set of duties is determined covering the tasks and satisfying the templates for the corresponding crew members. The foremost intention of the new crew planning process at NS, however, is to obtain a crew planning that is more fair at the individual level. To this end, we now extend the DRCPP by introducing the concept of fairness over time.

At NS, the planning is considered to be fair when each individual crew member performs roughly the same amount of attractive and unattractive work over a certain time span. In order to attain this fairness over the rolling planning horizon, a feedback mechanism is included in the DRCPP to control the assignment of specific work to specific crew members. Essentially, each assignment is evaluated based on a number of fairness criteria. These criteria relate to upper and lower bounds on the attributes discussed in Section 2.1, encapsulating the SS&S rules. We denote by A the set of these attributes. In the subsequent reasoning, we assume all attributes to have an upper bound. The reverse then holds for lower-bounded attributes. Each attribute $a \in A$ can adopt a score in the range $[0, r_a]$. Additionally, a threshold value $b_a < r_a$ is specified, denoting the desired upper bound on the amount of unattractive work for this attribute.

We penalize the assignment of more unattractive work to crew members that have recently already performed a relatively high amount of such work. Therefore, we introduce a penalty function to measure the badness of assigning a piece of work with a certain level of unattractiveness to a crew member with a certain history. We compute this penalty for each of the attributes separately. The total penalty for the crew member then equals the sum of these penalties. For attribute a, the function $f_a(u, s)$ defines the penalty of assigning a piece of work with unattractiveness score u to a crew member with history score s. We compute the penalty function as the product of these two elements, $f_a(u, s) = g_a(u) \cdot h_a(s)$. In the following, we call $g_a(u)$ the *unattractiveness penalty*, whereas we refer to $h_a(s)$ as the *history penalty*.

To compute the unattractiveness penalty for a piece of work for attribute a, we divide the unattractiveness score by the corresponding upper bound, $g_a(u) = \frac{u}{r_a}$. In essence, this penalty is proportional to the amount of unattractive work being assigned to the crew member. For the history penalty $h_a(s)$ we differentiate between the extent to which a crew member has already performed such work, and penalize according to the piece-wise linear function

$$h_a(s) = \begin{cases} 0 & \text{if } 0 \leq s \leq (1-m)b_a, \\ \alpha(1-\gamma) \cdot \frac{s-(1-m)b_a}{r_a} & \text{if } (1-m)b_a < s \leq b_a, \\ h_a(b_a) + \alpha \cdot \frac{s-b_a}{r_a} & \text{if } b_a < s \leq r_a. \end{cases} \tag{4.5}$$

The penalty contribution of the history s of a crew member depends on the work that he has previously performed. By this, we regard all work performed by the crew member within the rolling planning horizon. If the crew member's history with respect to the corresponding attribute is well below the threshold value b_a, no penalty is accumulated. On the other side, a major penalty is incurred if the history score exceeds this threshold. In the intermediate case, a minor penalty is applied.

The parameter $\alpha > 0$ is a scaling parameter for the history penalty. Furthermore, $\gamma \in (0,1)$ determines the extent to which the minor penalty must be considered, whereas $m \in (0,1)$ controls the transition between the penalty cases. Higher values of α induce higher penalties, so do lower values for γ. A lower value of m results in the major penalty being sooner applied. Both the unattractiveness penalty $g_a(u)$ and the history penalty $h_a(s)$ are divided by the upper bound on the attribute score r_a, in order to normalize the contribution of each individual attribute.

Similar to the dual variables, the computed penalties can be decomposed on the arcs of the corresponding pricing sub-problems. This way, the crew member's history is taken into account to control the allocation of new work in the DRCPP, thereby steering towards an allocation in which all work is divided fairly over time. All penalties are normalized in order to ensure a fair comparison of all work. It should however be noted that due to the non-linearity in the SS&S attributes, the computed sum of penalties in a duty is at most an approximation to the true penalty.

Chapter 5

An ALNS Heuristic for the DRCPP

The Dynamic Railway Crew Planning Problem with Fairness over Time at NS is currently proposed to be solved by means of the solution approach presented in Chapter 4. This approach, however, relies on a variety of greedy assumptions and heuristic design choices. Consequently, the resulting solution suffers from a number of downsides. First, as a result of the heuristic fixing to guarantee an integer solution, multiple tasks may eventually not be assigned to any duty. Experimental tests indicate that per day about 1% of the tasks currently remain uncovered. Moreover, since the DRCPP comprises an enormous amount of constraints and variables, it is only solved effectively for sub-parts of the network. To that end the network is decomposed into several groups, each containing one or more crew bases that are geographically related. The individual rosters are then obtained by solving the DRCPP separately for each group. Nevertheless, this a priori decomposition reduces the solution space, which potentially yields an inferior solution.

Accordingly, taking into account the above drawbacks of the proposed solution approach, improvements to this approach may lead to more satisfactory results. For this reason, in this chapter we propose an Adaptive Large Neighborhood Search (ALNS) meta-heuristic to enhance the quality of the solution. The ALNS framework, initially introduced by Ropke and Pisinger (2006), aims to construct a solution of higher quality by applying multiple methods to *destroy* and subsequently *repair* the incumbent solution. In our case, we apply the ALNS heuristic for each day in the planning period to the duties obtained by solving the DRCPP. The ALNS heuristic targets to improve the fairness of the current rosters, considering the crew members' history. In addition, this way we are able to integrate the solutions obtained for distinct sub-parts of the network, and seek for improvements to the joint solution that could not be found by the column generation heuristic.

In this chapter, we describe our Adaptive Large Neighborhood Search framework. In Section 5.1, we first clarify its positioning in the overall solution process. Then, we discuss how the heuristic is applied to enhance the quality of the solution in Section 5.2. Subsequently, we discuss an iterative method that attempts to insert the initially uncovered tasks in Section 5.3. Finally, we introduce the respective destroy operators in Section 5.4, and the repair operators in Section 5.5.

5.1 Overview of the Solution Process

In the DRCPP, duties are constructed for each day of the planning period on a rolling basis. In order to cope with this dynamic structure, we apply our ALNS heuristic in a similar fashion. That is, having obtained a DRCPP solution for an arbitrary day, we instantiate the ALNS heuristic to improve this set of duties. In this respect, we regard the output of the column generation heuristic as input to our ALNS heuristic. This modular set-up allows to consider the design choices made in both components separately from each other. Besides, we note that a feasible solution to the problem is returned upon termination of either component.

Algorithm 1 provides an overview of the procedure to obtain a crew planning for the entire planning period. Given the constructed template assignment for all crew members, the tasks to be planned, as well as a predefined horizon, we can obtain a corresponding crew planning.

Algorithm 1 Obtain the crew planning for the planning period

```
    Input: Template assignment for crew members, tasks to be planned, a planning horizon
    Output: A roster $x^*$ for all crew members for the corresponding planning period
 1: Pre-processing: Set attribute score history for all crew members
 2: $x^* \leftarrow \emptyset$
 3: $t \leftarrow$ first day in planning horizon
 4: repeat
 5:     UpdateRollingHorizonHistory($t$)
 6:     $x_t \leftarrow$ SolveDRCPP($t$)
 7:     $x_t^* \leftarrow$ ApplyALNS($x_t$)
 8:     $x^* \leftarrow x^* \cup x_t^*$
 9:     $t \leftarrow t + 1$
10: until last day in planning horizon is planned
11: return $x^*$
```

We first perform a pre-processing step, in which we set the attribute scores for each crew member. These scores are used to steer the allocation of attractive and unattractive work in the initial days of the planning period. We derive the scores from the work that is assigned to a crew member in the period prior to the planning period. If such an assignment is not available, for instance because the crew member did not perform any duties in recent time, we assume that he perfectly satisfies all attributes.

Having obtained the attribute scores, we start generating the crew planning. For each day of the planning period, we perform the following procedure to generate the duties. We first update the rolling horizon window, and discard the history for all planned duties now out of the rolling window. Next, we solve the DRCPP by means of the column generation heuristic from Chapter 4. The resulting solution is subsequently modified by the ALNS heuristic, and finally the duties are fixed to the rosters of the corresponding crew members. We then proceed to the next day in the planning period, for which this process is repeated. As soon as all days in the planning horizon have been planned, the algorithm terminates and the total set of constructed rosters is returned.

5.2 Adaptive Large Neighborhood Search Framework

We now provide a thorough description of the ALNS heuristic itself. We first introduce the required notation, after which we discuss the algorithmic procedure. In this procedure, the incumbent solution is iteratively modified by applying a sequence of operations. Each sequence consists of a particular destroy operation, followed by a repair operation. We denote the set of destroy operators by $\mathcal{D}$, whereas the repair operators are given in $\mathcal{R}$.

To each individual operator belong three parameters. First, the *weight* of an operator keeps track of its relative efficiency. For destroy operator $d \in \mathcal{D}$ and repair operator $r \in \mathcal{R}$ the weights are given by $\rho_d \geq 0$ and $\rho_r \geq 0$, respectively. Second, each operator receives a *score*, indicating the extent to which it is helpful in the current phase of the algorithm. The non-negative scores are registered by ζ_d and ζ_r, respectively. Lastly, for each operator it is registered how many times it has been used. We refer to this as the number of *attempt*s, denoted by ν_d and ν_r for the respective operators. In the ALNS framework, we refer to the collection of each destroy parameter by the vectors $\boldsymbol{\rho}^{\text{destroy}}, \boldsymbol{\zeta}^{\text{destroy}}$ and $\boldsymbol{\nu}^{\text{destroy}}$. The equivalent holds for the repair operators.

The complete ALNS procedure for an arbitrary day t is summarized in Algorithm 2. As input, we consider the selection of duties x_t returned by the column generation heuristic, that is: $x_t := \left\{ \bigcup_{\substack{\delta \in \Delta_t^r, \\ r \in R_t}} \delta : x_{rt}^{\delta} = 1 \right\}$. This solution is subsequently modified by the ALNS heuristic, and we ultimately return the final set of duties x_t^*.

Algorithm 2 Adaptive Large Neighborhood Search for day t

```
    Input: A solution $x_t$ for day $t$ returned by the DRCPP
    Output: A solution $x_t^*$ for day $t$ modified by the ALNS heuristic
 1: $x \leftarrow$ InsertUncoveredTasks($x_t$)
 2: $x_t^* \leftarrow x$
 3: $\boldsymbol{\rho}^{\text{destroy}}, \boldsymbol{\rho}^{\text{repair}} \leftarrow$ InitializeWeights()
 4: $\boldsymbol{\zeta}^{\text{destroy}}, \boldsymbol{\zeta}^{\text{repair}} \leftarrow$ InitializeScores()
 5: $\boldsymbol{\nu}^{\text{destroy}}, \boldsymbol{\nu}^{\text{repair}} \leftarrow$ InitializeAttempts()
 6:
 7: while stopping criterion is not met do
 8:     $d \leftarrow$ SelectDestroyOperator($\boldsymbol{\rho}^{\text{destroy}}$)
 9:     $r \leftarrow$ SelectRepairOperator($\boldsymbol{\rho}^{\text{repair}}$)
10:     $x' \leftarrow$ Destroy($d, x$)
11:     $x'' \leftarrow$ Repair($r, x'$)
12:     if AcceptanceDecision($x, x''$) then
13:         $x \leftarrow x''$
14:     end if
15:     if $z(x) < z(x_t^*)$ then
16:         $x_t^* \leftarrow x$
17:     end if
18:     $\boldsymbol{\zeta}^{\text{destroy}} \leftarrow$ UpdateScores($d, \Psi$), $\boldsymbol{\zeta}^{\text{repair}} \leftarrow$ UpdateScores($r, \Psi$)
19:     $\boldsymbol{\nu}^{\text{destroy}} \leftarrow$ UpdateAttempts($d$), $\boldsymbol{\nu}^{\text{repair}} \leftarrow$ UpdateAttempts($r$)
20:
21:     if segment limit is reached then
22:         $\boldsymbol{\rho}^{\text{destroy}} \leftarrow$ UpdateWeights($\boldsymbol{\zeta}^{\text{destroy}}, \boldsymbol{\nu}^{\text{destroy}}$)
23:         $\boldsymbol{\rho}^{\text{repair}} \leftarrow$ UpdateWeights($\boldsymbol{\zeta}^{\text{repair}}, \boldsymbol{\nu}^{\text{repair}}$)
24:         $\boldsymbol{\zeta}^{\text{destroy}}, \boldsymbol{\zeta}^{\text{repair}} \leftarrow$ ResetScores()
25:         $\boldsymbol{\nu}^{\text{destroy}}, \boldsymbol{\nu}^{\text{repair}} \leftarrow$ ResetAttempts()
26:     end if
27: end while
28: return $x_t^*$
```

The ALNS algorithm is initialized as follows. First, we attempt to insert the tasks left uncovered in the DRCPP solution x_t. We do so using an iterative procedure, which is outlined in Section 5.3. The resulting set of duties is registered as the current solution x and as the current best solution x_t^*. Furthermore, the weights, scores, and attempts are initialized. The weights are set to one, whereas the other parameters are assigned the initial value zero. The heuristic then performs a number of iterations, until a predefined stopping criterion is met.

In each iteration of the heuristic, we first select a destroy operator $d \in \mathcal{D}$ and a repair operator $r \in \mathcal{R}$ according to the roulette wheel principle. Here, the probability that a certain operator is selected is proportional to its weight, representing its relative efficiency. For an arbitrary destroy operator $d \in \mathcal{D}$, the selection probability π_d is given by

$$\pi_d = \frac{\rho_d}{\sum\limits_{i \in \mathcal{D}} \rho_i}, \tag{5.1}$$

whereas the equivalent holds for the repair operators.

Having selected the operators, we first apply the destroy operator to the current solution. Next, the forthcoming solution is modified by the repair operator. The quality of the new solution x'' is assessed by means of its objective value $z(x'')$. The goal of the ALNS heuristic is to assign duties to crew members such that the total work in all rosters over the rolling window is fairly distributed. As in the DRCPP, the fairness of a roster is evaluated by the extent to which the SS&S rules are satisfied. Put differently, in order to maximize the roster fairness we aim to minimize allocations that exceed the attribute threshold values.

For an arbitrary day t in the planning horizon, the objective is to minimize the sum of penalties assigned to the crew members scheduled to work on that day. For this, we denote by $h^r_a(s)$ the penalty assigned to crew member $r \in R_t$ for attribute $a \in A$ based on the score s of the work assigned to him inside the rolling horizon window. This penalty is computed similar to the history penalty (4.5). In the ALNS heuristic, we minimize the sum of those penalties, $\sum\limits_{r \in R_t} \sum\limits_{a \in A} h^r_a(s)$. The rolling horizon is updated as we proceed through the planning period, ensuring that we steer towards a balanced roster for each crew member at each point in the planning period.

The modified solution is either accepted and registered as the new current solution, or rejected and disregarded further. This decision is made using a Simulated Annealing acceptance criterion. We denote by x the solution before both operations were applied, and by x'' the solution after modification. If the new solution improves the current objective, i.e. $z(x'') < z(x)$, we accept the new solution. Otherwise, the acceptance probability of the new solution is given by $\exp\left\{\Big(z(x) - z(x'')\Big) / \, T\right\}$. The temperature parameter T is initialized by an initial temperature $T_0 > 0$ and decreased by a factor $\theta \in (0, 1)$ at the end of each iteration. This way, modified solutions with a deteriorating objective value have a higher acceptance probability in earlier stages of the algorithm, facilitating diversification in the initial iterations.

If the new solution is the best solution currently found, we register it as such. In addition, we update the scores of the operators. Depending on the quality of the new solution, as well as on the decision on accepting it or not, the respective scores are accumulated by a value Ψ given by

$$\Psi = \begin{cases} \psi_1 & \text{if the new solution is the new overall best,} \\ \psi_2 & \text{if the new solution improves the current solution,} \\ \psi_3 & \text{if the new solution is accepted,} \\ \psi_4 & \text{if the new solution is rejected.} \end{cases} \tag{5.2}$$

The scores are ordered in strictly decreasing order, i.e. $\psi_1 > \psi_2 > \psi_3 > \psi_4 \geq 0$. Since the exact contribution of either operator to the new solution is unknown, we update the scores of both the destroy and repair operator employed by the same amount. Furthermore, we update the attempt vectors to keep track of the number of times the distinct operators have been used.

This procedure is repeated in each iteration of the ALNS heuristic. In addition, each time the algorithm completes a predefined number of iterations (called *segments*), we perform an update to the weights of the operators that have been used. This way, the algorithm adapts to the instance at hand by favoring operators that turned out to be effective to recent solutions. For an arbitrary destroy operator $d \in \mathcal{D}$, the weights are updated by

$$\rho_d = (1 - \lambda) \cdot \rho_d + \lambda \cdot \frac{\zeta_d}{\nu_d}, \tag{5.3}$$

where $\lambda \in [0, 1]$ is a decay parameter controlling the extent to which new information is included in the weights. Again, the equivalent holds for the repair operators. Along with updating the weights, we reset the score and attempt vectors to their initial values, after which we start the next segment.

This design of the ALNS heuristic yields two direct advantages. First, a feasible solution is returned in each iteration. This is a particularly nice feature for the railway operator, since multiple solutions of a similar quality can be compared. Second, the framework comes with a variety of parameters that can be tuned. Based on expert knowledge or preference, the heuristic can be tailored entirely towards the railway operator's desires. Again, these parameters can be chosen independent of the parameters used in the DRCPP algorithm.

5.3 Inserting Uncovered Tasks

One of the major drawbacks of the current solution approach outlined in Chapter 4 is the fact that eventually not all tasks may be assigned to a duty. The greedy fixing rule used to convert the fractional column generation solutions into an integer solution is able to obtain a reasonable set of duties quickly, but in doing so it may leave some tasks uncovered unnecessarily. Obviously, a practical set of duties covers all tasks, or at least as many as possible. Therefore, we now describe an iterative insertion algorithm that attempts to cover the initially unassigned tasks before commencing the ALNS heuristic. The following procedure relies on the ideas proposed by Potthoff et al. (2010) for crew rescheduling.

Let $\hat{K}_t \subseteq K_t$ be the current set of uncovered tasks for an arbitrary day t. In order to insert these tasks, we restrict our attention to only that part of the solution bearing a high probability of offering insertion opportunities. This typically does not comprise the complete instance, but rather reduces to a set of crew members performing duties that in some measure are related to the uncovered tasks. We refer to this reduced problem as the *insertion core problem.*

Given the initial set of uncovered tasks and the duties generated, we construct the insertion core problem as follows. We first identify the covered tasks that are chronologically and geographically 'close' to the uncovered tasks. We regard a covered task to be close if it has the same starting and ending station as any of the uncovered tasks. Additionally, the starting time of this covered task must be within the interval $[t_0, t_1 + 60 \text{ minutes}]$. Here, t_0 represents the earliest starting time of an uncovered task with the same starting and ending station, whereas t_1 denotes the latest ending time of an uncovered task with equal station pair. The insertion core problem then consists of all crew members performing any of these identified tasks, as well as all crew members currently assigned a reserve duty.

Denoting the set of crew members obtained in the insertion core problem by $\hat{R}_t \subseteq R_t$, we subsequently attempt to insert the uncovered tasks by re-planning the duties of these crew members. Similar to the approach described in Section 4.2, we use column generation to find a suitable set of candidate duties $\hat{\Delta}_t^r \subseteq \Delta_t^r$ for the corresponding crew members. Since the size of the insertion core problem is considerably smaller than that of the crew planning problem for the complete instance, we are able to find candidate duties using an exact rather than a heuristic pricing algorithm. Contrary to the initial planning approach, the goal of this procedure is to cover as many tasks as possible. Therefore, we refrain from putting the fairness penalties on the arcs of the pricing sub-problems as in Section 4.3. Having generated the candidate duties, we find a new duty assignment by solving the following insertion model:

$$\min \sum_{r \in \hat{R}_t} \sum_{\delta \in \hat{\Delta}_t^r} c_{rt}^{\delta} x_{rt}^{\delta} + \sum_{k \in \hat{K}_t} f_k y_k \tag{5.4}$$

$$\text{s.t.} \sum_{r \in \hat{R}_t} \sum_{\delta \in \hat{\Delta}_t^r} a_{tk}^{\delta} x_{rt}^{\delta} + y_k \geq 1 \qquad \forall k \in \hat{K}_t \tag{5.5}$$

$$\sum_{\delta \in \hat{\Delta}_t^r} x_{rt}^{\delta} = 1 \qquad \forall r \in \hat{R}_t \tag{5.6}$$

$$x_{rt}^{\delta} \in \mathbb{B} \qquad \forall r \in \hat{R}_t, \delta \in \hat{\Delta}_t^r \tag{5.7}$$

$$y_k \in \mathbb{B} \qquad \forall k \in \hat{K}_t. \tag{5.8}$$

Similar to the DRCPP model (4.1)-(4.4), the objective (5.4) of the insertion model is to minimize the cost of the selected duties. Additionally, we penalize the case whenever an uncovered task

$k \in \hat{K}_t$ is left out of the final solution. The decision to cover a task is established by constraints (5.5). Constraints (5.6) and (5.7) are equivalent to constraints (4.3) and (4.4), respectively. Finally, constraints (5.8) denote the binary domain of the cancellation decision variables.

By setting the task cancellation parameter value f_k sufficiently large, the insertion model (5.4)-(5.8) constructs the duty assignment that covers the highest number of distinct tasks. In case this assignment covers all tasks, we terminate the insertion procedure and return the corresponding set of duties. Otherwise, instead of focusing on all tasks currently uncovered, in the subsequent we iteratively attempt to insert one specific task. For this, we pick at random one of the uncovered tasks and explore an *insertion sub-problem* for this task.

The insertion sub-problem for an uncovered task $\hat{k} \in \hat{K}_t$ is constructed as follows. First, we identify from the current set of duties a number of candidates that may be capable of covering $\hat{k}$. To this end, we filter all duties containing a task k starting at the same station as $\hat{k}$. From these duties, we then select the `insertionNumCandidates` duties for which the departure time of task k is closest before the departure time of $\hat{k}$. Similarly, we select the same number of duties for which the departure time of task k is closest after that of $\hat{k}$.

Merely including the selected candidate duties, however, would probably result in a solution that leaves other tasks uncovered. For this reason, we find for each candidate the `insertionNumSimilar` most similar duties. The candidate and similar duties together offer more opportunities to insert the uncovered task. The similarity between two duties δ^1 and δ^2 is measured by the similarity score

$$S(\delta^1, \delta^2) := \beta\, \mathbb{I}\Big[\delta^1_b = \delta^2_b\Big] + \sum_{k^1 \in \delta^1} \sum_{k^2 \in \delta^2} \mathbb{I}\Big[k^1_{ds} = k^2_{ds}\Big]\, \mathbb{I}\Big[|k^1_{dt} - k^2_{dt}| \leq \omega\Big], \tag{5.9}$$

where $\mathbb{I}[\cdot]$ is the indicator function and δ_b denotes the crew base of a duty δ. For each task k, the starting station is denoted by k_{ds} and the starting time by k_{dt}. An upper bound on the absolute difference in starting time between two tasks is given by ω. The higher the similarity score, the more similar are two duties. We adopt the parameter values for the crew base bonus coefficient $\beta = 0.6$ and the time difference threshold $\omega = 60$ minutes from Potthoff et al. (2010). Nevertheless, we note that for the bonus coefficient any value in the interval $(0, 1)$ can be chosen, as it intends to favor duties from the same crew base in case multiple duties happen to be equally similar. Similar to the insertion core problem, the insertion sub-problem comprises all crew members performing the selected candidate and corresponding similar duties.

The insertion sub-problem is solved similar to the insertion core problem, after which we update the current set of duties and the list of uncovered tasks. We repeat this procedure until either all tasks have been covered or no new uncovered task can be explored. From all sets of duties constructed in the insertion sub-problems, the one leaving the least number of tasks uncovered is returned to the ALNS framework.

5.4 Destroying the Solution

In this section, we present the destroy operators used in the ALNS heuristic outlined in Section 5.2. In each iteration of the heuristic, a part of the solution is destroyed and subsequently repaired in order to obtain a solution of better quality. The destroy operations in our heuristic are targeted to remove a subset of the duties of the incumbent solution. For this, we aim to select duties with a high probability of improving the solution. In our ALNS framework we include three destroy operators, which we discuss below.

5.4.1 Remove Worst and Similar Crew Members

Since the primary goal of the ALNS heuristic is to attain a more fair allocation of the work among the crew members, it seems intuitive to consider more closely the crew members who currently violate one or more SS&S rules. These crew members currently perform relatively much unattractive work, hence it is likely that a better solution can be obtained by re-planning these crew members.

This destroy operator (abbreviated RemoveWorst) identifies for a given SS&S attribute the crew members being assigned most unattractive work in the current rolling window. The extent to which a crew member has been assigned (un)attractive work can be deduced from his attribute history score s, where a low (in case of a lower-bounded attribute) or high (for upper-bounded attributes) score indicates that much unattractive work of the corresponding attribute has been assigned. In the following, we refer to these crew members as crew members having a *bad* score. The equivalent holds for crew members assigned relatively much attractive work.

In order to achieve a more balanced allocation of the attractive and unattractive work over the crew members, it is unlikely that re-planning the duties for crew members having bad scores alone leads to a fairness improvement. Therefore, we additionally identify the crew members with good scores who are assigned a duty similar to the duties of the bad scoring crew members. Since crew members with good scores have recently performed relatively much attractive work, profitable exchange opportunities may arise that result in a more balanced schedule for all crew members.

Given an attribute and a corresponding crew member having a bad score, we identify the desired crew members with good scores and performing similar duties using the same approach as in the insertion sub-problems in Section 5.3. For the duty assigned to the bad scoring crew member, we first compute its similarity to all other duties in the solution using the similarity score (5.9). Remember that this score considers the crew base as well as the departure time and station of all tasks performed during the duty. We subsequently order the similarity scores in decreasing order, and select the duties for which the corresponding crew member currently does not violate the attribute threshold. In total, we identify the duties corresponding to the `numWorstDuties` crew members with the worst scores. For each such duty, we identify a number of `numSimilarDuties` similar duties assigned to the corresponding crew members with good scores.

In each iteration of the ALNS heuristic this operator is applied for one of the attributes. The attribute decision is made by applying the roulette wheel principle, similar as in Equation (5.1). The weights are represented by the sum of attribute penalties incurred by the selected crew members with the worst scores. Once the attribute is selected, all desired similar duties are identified and we complete the destroy operation by removing the selected duties from the solution. In order to prevent cycling in subsequent iterations, we keep track of the crew members identified as having the worst scores. These crew members cannot be identified as such again in the next ϕ iterations.

5.4.2 Remove Random Crew Members

The previous operator targets at intensification of the current solution: By considering all attributes, history scores, and other crew members performing similar duties, a set of duties with high probability of improving the solution is identified. Despite obtaining the potential best candidates for destruction, considering more criteria induces the risk of the solution to get stuck in local optima. To be able to escape from these local optima, we introduce an operator targeted to diversify the solution. Proceeding with a diverse and potentially worse solution may eventually enable the algorithm to reach solutions in better regions of the solution space.

This operator (brief: REMOVERANDOM) identifies at random a number of crew members, and removes the duties assigned to them on the corresponding day. Since we do not consider any additional criterion, the operator is entirely targeted to solution diversification. The extent to which the current solution is destroyed is regulated by the parameter `numRandomDuties`.

5.4.3 Swap Duty End for Two Duties

A third approach that may yield promising destroy opportunities focuses on the allocation of work among crew members within the same crew base. In the current way of planning, some crew members at a crew base are assigned relatively attractive work, whereas others perform relatively unattractive work. In order to obtain a more balanced division of the work, it would make sense to assign the former group of crew members some of the unattractive work initially assigned to the latter group, and vice versa.

This operator, SwapEnds in short, identifies for each pair of crew members of the same crew base the instants at which they would 'meet' during their duty. The idea is that, if two crew members are at the same location (roughly) at the same time, they can swap the remainder of their duties. If one crew member has been assigned relatively unattractive work and can perform a more attractive duty by swapping with a colleague, the allocation of work overall will be more balanced. We note that in reality such an agreement between crew members cannot be arranged. However, since the rosters are not communicated to the crew members at this stage, we can exploit these opportunities to improve the solution.

We call an instant at which two crew members meet an *exchange opportunity*. A feasible exchange opportunity is characterized as follows. First, the two crew members must be from the same crew base. Given the restriction that a duty must start and end at the crew base and we only swap their duty ends, it follows that the crew members must be of the same crew base. Second, the new duties must be compatible with the ending time of the template assigned to both crew members. If crew members A and B swap their duty ends, the end time of the duty initially assigned to crew member A must be within the template time block for crew member B. The equivalent holds of course for the duty initially assigned to crew member B. Since we only swap the duty ends, the duty start times remain untouched. Lastly, the early and late rotation constraints should remain satisfied for both crew members, whenever applicable, and both duties should eventually include a proper meal break.

Swapping the ends of a crew member's duty at a feasible exchange opportunity is executed by assigning the remainder of the tasks to be performed to the other crew member, as depicted in Figure 5.1. Here, the exchange opportunity outlined in red occurs upon completion of tasks $A3$ and $B2$, respectively. Since both crew members are at the same location (roughly) at the same time, crew member A can be assigned the remainder of duty B starting with task $B3$. Similarly, crew member B proceeds by performing task $A4$ initially assigned to crew member A. Before accepting the swap, it should be checked whether both crew members are allowed to connect from their original duty to the other duty at this exchange opportunity. This essentially ensures a sufficiently long transfer time and forces swaps to take place only at stations with a relief opportunity.

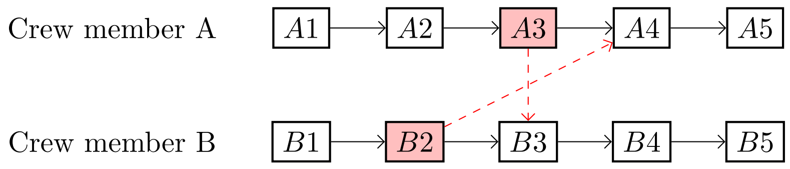

Figure 5.1: Swapping the ends of two duties at a feasible exchange opportunity.

Having identified all feasible exchange opportunities for all pairs of duties, we compute for each individual swap the estimated change in objective value for the ALNS heuristic. We subsequently perform all mutually and collectively disjoint swaps that result in an improving objective value. We note that the total number of exchange opportunities is of quadratic order, but by fixing the first duty and identifying all swaps in parallel we can reduce the computation time substantially. In addition, since this operator identifies both the opportunity of swapping and proposes the new content of the duties, it effectively encompasses both a destroy and a repair operation. For this reason, no additional repair operation is performed in the heuristic after selecting this destroy operator.

5.5 Repairing the Solution

In this section, we present an approach to repair the destroyed solution in the ALNS heuristic. A feasible solution is reconstructed by re-planning duties for the crew members removed in the previous destroy operation. Since constructing a feasible duty given a crew member's roster entails a large number of restrictions, finding suitable alternative solutions is very challenging. For this reason, the single repair operator employed in the ALNS heuristic constructs a new set of duties using an exact approach. New candidate duties are first obtained by column generation, after which a new duty allocation is achieved by solving a binary linear program similar to the insertion model (5.4)-(5.8). In the remainder of this section, we elaborate on the use of this repair operator in our ALNS heuristic.

The repair operator SolveExact aims to find a suitable replacement duty for all crew members that need to be re-planned. This group again consists of the crew members removed in the destroy phase, as well as the crew members currently on reserve. In order to generate duties with appropriate content, we consider only a specific subset of the tasks rather than the complete task set. This subset comprises three groups of tasks. First, all tasks that were already uncovered prior to the last destroy operation. Second, the tasks that became uncovered after applying the destroy operation. Third, the destroyed tasks that remain covered by another, non-destroyed duty. Taking into account the last group enables to reconstruct the original solution, if this turns out to be the best alternative. We disregard all still covered tasks unrelated to the last destroy operation, which typically involves the majority of the task set. Focusing on a promising subset of the tasks thus significantly reduces the size of the repair operation. We note that this approach may overlook potentially profitable deadheading opportunities that utilize a covered task out of scope. However, we deem that the reduction in problem size outweighs the impact of these events on the resulting solution.

Having identified the tasks to consider, we apply column generation to find a set of candidate duties for all crew members. We initialize the Restricted Master Problem with slack variables for the templates and the tasks in the second group, since these tasks are set uncovered by the last destroy operation and need to be re-planned. No slack variables are added for the first and third group of tasks. The tasks in the first group may not be compatible for the crew members to be re-planned, given the start and end times of their templates or the location of the tasks in the second group. On the other hand, the tasks in the third group are still covered and thus need not be covered again. They did however provide a useful deadheading opportunity in the destroyed duty, and for this reason they are still considered. In addition, we provide the column generation algorithm with the destroyed duties as initial columns in order to further reduce the computation time. The size of the repair problem again permits the use of an exact pricing algorithm to find candidate duties.

Having obtained the candidate duties, we obtain a new duty assignment for the crew members to be re-planned by solving the insertion model (5.4)-(5.8). We register the new duties and update the list of uncovered tasks. The duties are subsequently added to the rosters of the corresponding crew members, and their history scores are updated. The new solution is then returned to the ALNS framework.

Chapter 6

Computational Experiments

In this chapter, we analyze the performance of our ALNS heuristic in improving the quality of the current rosters. For this, we apply the heuristic to two medium-sized and one large instance provided by Netherlands Railways. We first describe the characteristics of the instances in Section 6.1. In Section 6.2, we provide the parameter values used throughout the experiments. Next, in Section 6.3, we discuss the impact of applying our ALNS heuristic to the rosters of the medium-sized instances. In Section 6.4, we show that an even more substantial fairness improvement can be achieved by applying the heuristic to a larger instance. Finally, we justify the parameters settings used by performing a sensitivity analysis in Section 6.5.

6.1 Problem Instances

We apply our ALNS heuristic to three instances of the Dutch national railway network. These comprise two medium-sized instances earlier used in Van Rossum et al. (2022), and one instance of a larger size. The medium-sized instances comprise an instance in the eastern part of the Netherlands and an instance in the middle of the country, respectively. We refer to these instances as `East` and `Center`, respectively. Instance `East` consists of the crew bases Arnhem (Ah), Enschede (Es), Hengelo (Hgl), Nijmegen (Nm), and Zwolle (Zl), whereas instance `Center` comprises the crew bases of Amersfoort (Amf), Amsterdam (Asd), and Utrecht (Ut).

The computational complexity of the DRCPP necessitates the current decomposition of the country in multiple regions. A dedicated feature of the ALNS heuristic, however, is its capability of combining several initial solutions and identifying improvements to the joint solution. For this reason, we also study a large instance `Joint` equivalent to the union of instances `East` and `Center`. Due to its size, this instance cannot be solved by Van Rossum et al. (2022), however by exploiting the initial solutions of both medium-sized instances we can handle it using our ALNS heuristic. A graphical interpretation of all instances and their respective crew bases is given in Figure 6.1.

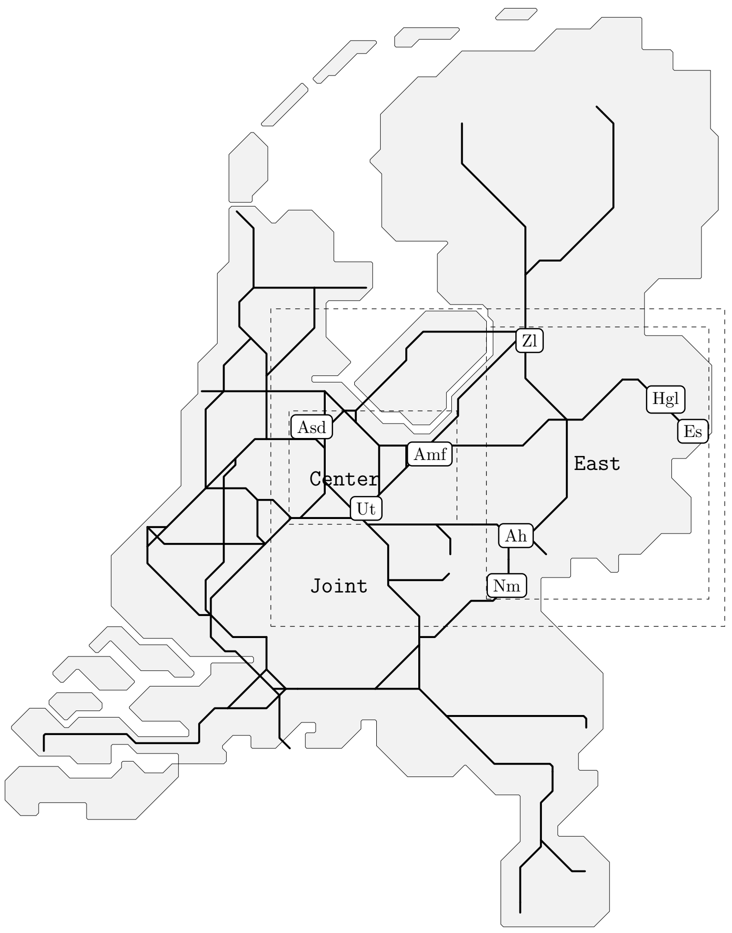

Figure 6.1: Geographical representation of the instances under consideration.

We focus on the crew planning problem for the guards. This implies that we disregard any constraints regarding route and rolling stock knowledge, which are applicable for the drivers. With respect to the attributes discussed in Section 2.1, we include the attributes for Type-A work, Aggression, Double Decker, and Duty Length. We do not consider the Unique Kilometers and RWD attributes, since their exact implementation within the *Fundamenteel Spoor* project has not yet been decided.

We make use of original data derived from the realized crew plan of Netherlands Railways. These data consist, apart from the planned duties, of information on the characteristics of individual tasks and restrictions on what tasks can be scheduled adjacently. Seven weeks of original data are available, covering the period from October 4 to November 21 of 2021. Similar to Van Rossum et al. (2022), we artificially expand the planning period by generating data for another seven weeks. We adopt the exact assignment of crew members to templates and tasks to crew bases used in their paper. In our experiments, we use the first period to generate the attribute history scores in pre-processing. The ALNS heuristic is subsequently tested on the second period, spanning from November 22 of 2021 until January 9 of 2022.

Table 6.1 shows per crew base the number of crew members, templates, and tasks in the realized crew plan. Despite containing fewer crew bases, instance `Center` is the largest among the two medium-sized instances, consisting of 11,074 templates. This is a result of the instance being situated in a more metropolitan area of the country, where line frequencies are higher compared to the eastern part of the country. This leads to more tasks to be executed and more crew members needed to cover them. With respect to instance `East`, it is apparent that the crew bases located near the ends of the network (Es, Hgl, Nm) are substantially smaller than crew bases on more connected lines (Ah, Zl). In total, our `Joint` instance consists of 925 crew members, for whom slightly more than 20,000 duties must be scheduled.

Table 6.1: Characteristics of the instances.

Instance	Crew base	Crew	Templates	Tasks
`East`	Ah	131	2,415	23,907
	Es	53	1,260	15,626
	Hgl	23	469	4,085
	Nm	80	1,974	20,658
	Zl	118	2,884	27,535
	Total	405	9,002	91,811
`Center`	Amf	96	2,282	22,260
	Asd	216	4,235	39,696
	Ut	208	4,557	38,595
	Total	520	11,074	100,551
`Joint`	**Total**	925	20,076	192,362

The current and threshold values for each of the SS&S attributes considered are displayed in Table 6.2. The values for `East`, `Center`, and `Joint` are the average values for each instance based on the realized crew plan. From these statistics, a substantial difference can be noted between the instances especially in the Type-A and Aggression attribute: In the metropolitan part of the country covered by instance `Center`, the share of regional trains and aggression trains is higher compared to instance `East`. Additionally, we observe that complying with the average duty length requirement is challenging in all instances. Finally, the threshold values are determined such that exchange potential for attractive and unattractive work between the instances is facilitated for the ALNS heuristic, while at the same time resembling the realistic thresholds in use in current operation.

Table 6.2: Current average and threshold values for all attributes.

Attribute	`East`	`Center`	`Joint`	Threshold
Type-A	0.54	0.38	0.45	0.35
Aggression	0.12	0.23	0.18	0.30
Double Decker	0.28	0.23	0.25	0.45
Duty Length (h)	7:57	8:01	8:00	8:00

6.2 Parameter Settings

In this section, we describe the parameter values used throughout the experiments. We need to set the parameters for two distinct frameworks, the DRCPP algorithm and our ALNS heuristic. Since we essentially consider the output of the former component as input to our algorithm, we use similar settings as in Van Rossum et al. (2022). With respect to the column generation parameters, we terminate the column generation when the percentual decrease in objective value is below $\epsilon = 0.01\%$ and we set the fixing value τ to 0.6. Moreover, in each heuristic pricing sub-problem we keep track of 50 labels and only register duties that are at least 50% disjoint from each other. We update the column pool after each iteration and remove columns whose reduced cost was at least 250 in the last 5 iterations. Lastly, the history penalty function is initialized with identical parameter values, $(\alpha; \gamma; m) = (1,000; 0.5; 0.1)$.

In the ALNS heuristic, we use a rolling horizon window where for each crew member the latest 45 duties are registered. We need to set the parameters for three categories: parameters corresponding to the general ALNS framework, parameters used for inserting the uncovered tasks, and operator-specific parameters. We perform a detailed sensitivity analysis to determine a sensible composition for the last group of parameters in Section 6.5.

The general ALNS framework encompasses a number of design parameters. First, we terminate the heuristic when a time limit of 15 minutes has been passed, or sooner when the number of subsequent iterations without improvement exceeds 20. Although in practice the time available for computation is in the order of magnitude of several hours (only one crew plan should be constructed each day, i.e.), these stricter limits turn out to provide already a solid insight in the solution direction. Furthermore, each segment consists of 15 iterations. We define the initial temperature T_0 specific to each instance by setting it as 10% of the objective value of the initial solution for the corresponding day. The temperature decrease factor θ is set to 0.95. In addition, we use score values of $\psi_1 = 10, \psi_2 = 5, \psi_3 = 3$, and $\psi_4 = 1$, respectively, and set the decay parameter to $\lambda = 0.2$. In the second category of parameters, we set the value for both `insertionNumCandidates` and `insertionNumSimilar` in the insertion sub-problems to 3.

Finally, we set the following parameter values for the distinct operators. In RemoveWorst, the values for `numWorstDuties` and `numSimilarDuties` are set to 3 and 10, respectively, and the cycling parameter ϕ is set to 10. To allow for a fair comparison of the performance between the RemoveWorst and RemoveRandom operators, we set the value of `numRandomDuties` such that the same number of duties is removed from the solution. That is, the number of duties to be removed by RemoveRandom is given by `numRandomDuties` := `numWorstDuties` + `numWorstDuties` · `numSimilarDuties`.

6.3 ALNS Performance on Medium-Sized Instances

In this section, we discuss the results obtained by applying our ALNS heuristic to the rosters of the medium-sized instances `East` and `Center`. In particular, we address the performance of the heuristic with respect to inserting the initially uncovered tasks, since this is regarded the main drawback of the current solution approach. Furthermore, we analyze the ability of the heuristic in improving the fairness among the crew rosters. Finally, we evaluate the technical performance of the ALNS framework as a whole, and of the employed operators. All subsequent experiments are performed on a Windows desktop with Intel Xeon 3.60GHz processor, four cores and 16 GB RAM. The mathematical programming models are solved using the commercial solver CPLEX version 20.1.0.

Since the current solution approach is not able to assign all tasks to a duty, we initiate our ALNS heuristic with an iterative procedure attempting to reinsert these tasks. Table 6.3 shows for both instances the total and median number of uncovered tasks, as well as the minimum and maximum values encountered per day. The entries correspond to the number of uncovered tasks before starting the insertion step (i.e. inherited from the DRCPP), after solving the insertion core problem, and after solving the insertion sub-problems, respectively.

Table 6.3: Evolution of the number of uncovered tasks when applying the insertion procedure.

Instance	Method	Total	Median	Min	Max
`East`	DRCPP	494	9	2	36
	Core problem	467	8	1	36
	Sub-problems	454	8	0	34
`Center`	DRCPP	289	5	1	30
	Core problem	214	3	0	25
	Sub-problems	197	2	0	22

Over all, we are able to insert a substantial amount of the uncovered tasks in both instances. The reduction of the number of uncovered tasks by 32% in instance `Center` is stronger compared to instance `East`. From the initial solutions, we observe that instance `Center` inherits substantially more empty duties, hence providing more potential to insert the initially uncovered tasks. For both instances, the majority of the reduction is achieved by solving the insertion core problem. The complete insertion procedure yields a solution with fewer uncovered tasks for 25 (instance `East`) and 37 (`Center`) of the 49 days in our planning horizon, respectively. We note however that the initial number of uncovered tasks per day varies greatly across the planning period, where for some days the DRCPP solution already has nearly all tasks assigned. Still, we are not able to re-insert all uncovered tasks. This is predominantly caused by the heuristic character of the insertion procedure, as well as the a priori assignment of the templates to the crew members. The establishment of the template time blocks for certain crew members prevents some tasks to be inserted.

We now analyze the power of the ALNS heuristic in increasing the fairness among the rosters. For this, we discuss both the total amount of unattractive work assigned over the planning horizon, as well as the ability to shift unattractive work to crew members with a better work history. We first show that our ALNS heuristic reduces the total amount of penalties allocated, and so equivalently the total amount of unattractive work assigned to the crew members. Table 6.4 displays the objective value both before and after applying the heuristic, as well as the individual contribution of each attribute.

Table 6.4: Objective contribution per attribute.

Instance	Method	Objective	Type-A	Aggression	Double Decker	Duty Length
`East`	DRCPP	478,563.05	654.83	0.00	47,419.68	430,488.54
	ALNS	468,042.24	74.33	0.00	39,735.62	428,232.29
	Delta	-10,520.81	-580.50	0.00	-7,684.06	-2,256.25
`Center`	DRCPP	581,226.07	86,926.92	69,276.76	359.28	424,663.11
	ALNS	536,238.21	56,885.59	54,805.03	21.78	424,525.81
	Delta	-44,987.86	-30,041.33	-14,471.73	-337.50	-137.30

Applying the ALNS heuristic results in a decrease in objective value of about 2% (instance `East`) and 8% (`Center`). Similar to inserting the uncovered tasks, we observe that the reduction in the latter instance is higher due to its larger initial freedom. Remember from Table 6.2 that satisfying the average duty length requirement has been challenging for both instances, leading to a high initial penalty contribution. Our heuristic achieves a penalty decrease for all days in the planning horizon, and across all attributes. For instance `East`, however, we note that initially there was already so little aggression work that no crew member has been assigned a penalty for this. For instance `Center`, we see that the largest reductions are achieved in the

Type-A and Aggression attributes. Again, this is in line with our previous observation that the metropolitan part of the country contains relatively much of this work. The two instances together show that our heuristic is able to reduce the penalties on every attribute.

We observe for both instances the objective value to rise after a successful insertion procedure. For both instances this increase is less than 0.2%, and is incurred since the procedure prioritizes inserting the uncovered tasks above enhancing the fairness of the rosters. The ALNS heuristic subsequently improves the solution after inserting the uncovered tasks, eventually returning a set of duties that contains fewer unattractive work than the initial solution.

The decrease in objective value shows that our ALNS heuristic reduces the total amount of unattractive work in the rosters. Nevertheless, it does not allow us to make statements on the distribution of this work among the rosters. To that end, we now consider the extent to which our heuristic is able to shift away unattractive work from the crew members assigned too much of such work. Recall that we use a history penalty function (4.5) that induces a major penalty in case a crew member is assigned too much unattractive work and therefore violates the attribute threshold value. At the same time, a minor penalty is applied for crew members nearly violating this threshold. Table 6.5 shows for each attribute the number of crew members in each category of the penalty function. It is indicated whether a crew member violates (V) or nearly violates (NV) the SS&S threshold, or currently adheres (A) to this value.

Table 6.5: Number of crew members per adherence category per attribute.

		Type-A			Aggression			Double Decker			Duty Length		
Instance	Method	V	NV	A	V	NV	A	V	NV	A	V	NV	A
`East`	DRCPP	3	12	8,987	0	0	9,002	56	2,846	6,100	3,134	5,868	0
	ALNS	0	6	8,996	0	0	9,002	20	2,832	6,150	3,017	5,985	0
`Center`	DRCPP	41	6,593	4,440	55	3,713	7,306	1	26	11,047	432	10,612	30
	ALNS	13	6,355	4,706	41	3,551	7,482	0	7	11,067	531	10,502	41

In line with the objective decrease, we observe a reduction in the number of crew members (nearly) violating the thresholds for the three SS&S attributes. For the Duty Length attribute, relatively many crew members from instance `East` violate the threshold. Nevertheless, the corresponding attribute penalty is comparable with that of the `Center` instance, indicating that most crew members violating the threshold in instance `East` do so only marginally. For instance `Center`, we observe an increase in the number of crew members violating the Duty Length attribute. This increase is due to the nature of our heuristic, which intends to minimize the total amount of penalty allocated to all crew members. Since we consider all attributes to be equally-weighted, extending a duty's length can be profitable. More attractive (in terms of the other attributes) work can be performed in a longer duty, which then results in the penalty for these attributes to be lower. Due to the choice of our penalty function, minimizing the total amount of unattractive work is not equivalent to maximizing the number of crew members adhering to the attribute thresholds. For this reason, the number of crew members (nearly) violating the Duty Length attribute can still increase despite the corresponding penalty decreases.

In general, the best solution returned by the ALNS heuristic need not necessarily be the solution with the highest number of adhering crew members. The penalty function aims to push away crew members from the imposed thresholds by assigning a penalty also when they nearly violate this threshold. This may lead to the case where two nearly violating crew members together induce a higher penalty than one crew member marginally violating the threshold. Despite this subtle difference between the two situations, we stress that by reducing the total amount of penalty across all attributes, our heuristic still attains allocations with more crew members satisfying the thresholds in most cases. Nonetheless, recall that each iteration in our ALNS heuristic returns a feasible solution, which can be revised at a later moment.

Decreasing the total amount of unattractive work assigned and inserting the initially uncovered tasks, however, does not come completely priceless. Foremost indicators for this are the number of empty duties and the number of deadheading tasks in the solution. As a result of the DRCPP solution being constructed very greedily, we observe some of the initial solutions to contain both duties without content, while at the same time some tasks are still left uncovered. Table 6.6 shows the number of empty duties and deadheading tasks in the instances both before and after applying the ALNS heuristic.

Table 6.6: Evolution of the number of empty duties and deadheading tasks.

Instance	Method	Empty	Deadheading
`East`	DRCPP	39	3,309
	ALNS	0	2,877
`Center`	DRCPP	260	1,474
	ALNS	11	2,016

Again, the DRCPP solution for instance `Center` contains substantially more empty duties compared to instance `East`. We observe that our heuristic is capable of filling nearly all of these empty duties, mostly by assigning attractive work to the corresponding crew members. Assigning attractive work improves the crew member's attribute scores, eventually resulting in a decreasing objective contribution. Filling the empty duties however leads to an increase of the average duty length, and thereby more crew members violating this attribute for the `Center` instance. At the same time, filling duties with attractive work is at the cost of increasing the number of deadheading tasks. Since this is currently not penalized explicitly in our heuristic, assigning attractive deadheading tasks is an effective way to reduce the amount of unattractive work assigned to a crew member. We note however that in practice an excessive amount of deadheading tasks is not desired from an efficiency point of view. In case of disruptions, a crew planning with excessive deadheading is very fragile. In addition, this planning incurs additional unnecessary costs for the railway operator.

On the other hand, for instance `East` the initial number of empty duties is more moderate. This enables us to fill these duties more efficiently, resulting in a reduction of the number of deadheading tasks of about 13%. Opposite to instance `Center`, this simultaneously leads to a decrease in the number of crew members violating the average duty length requirement.

Finally, we consider the performance of our ALNS framework on itself. We consecutively discuss the computation time used for the different components, and the operators used in the heuristic. Table 6.7 displays the average computation time for all relevant components of the solution process.

Table 6.7: Average computation time in seconds for each solution component.

Instance	DRCPP	Core problem	Sub-problems	ALNS
`East`	155	35	83	821
`Center`	407	16	43	905

On average, it takes only a couple of minutes to obtain an initial solution by solving the DRCPP algorithm. For the larger and more complex instance `Center`, this indeed takes longer than for instance `East`. The computation time displayed for the insertion core and sub-problems entail both the time used for column generation and for solving the insertion model. The time needed to solve all insertion sub-problems on the same day is aggregated. As we expect, the time spent in the total insertion procedure is higher for instance `East`, since it comprises substantially more initially uncovered tasks compared to instance `Center`. We find however that the insertion core problem, despite contributing to the majority of the success of the insertion procedure, requires fewer computation time than the insertion sub-problems. In other words, on average relatively much computation time is spent to a component yielding relatively little improvement.

We observe that the average time spent in the ALNS heuristic is lower for instance `East`. For this instance, the heuristic terminated prematurely more often since no improving solution had been found in the predefined number of successive iterations (20 times against 7 for instance `Center`). Again, we indicate that the `Center` instance provides more freedom for making objective improvements. For all other days, although still finding better solutions, the heuristic is terminated upon hitting the time limit.

Most of the computation time spent in the ALNS heuristic is used in the SwapEnds and SolveExact operators, as these aim to reconstruct a new solution. The other (destroy) operators are performed in negligible time. Table 6.8 shows for each destroy operator the average time (in seconds) required to restore a new solution. Moreover, it displays the average number of times each operator is selected, the average number of times an improvement is realized, and the average improvement itself.

Table 6.8: Average performance of the destroy operators used in the ALNS heuristic.

Instance	Iterations	Operator	Selected	Improvements	Avg. improvement	Time[1]
`East`	223.5	RemoveWorst	69.5	21.3	4.8	4.79
		RemoveRandom	92.1	34.3	2.9	3.90
		SwapEnds	61.9	20.3	6.7	1.05
`Center`	148.0	RemoveWorst	55.5	36.8	14.4	7.60
		RemoveRandom	61.3	44.4	8.9	4.51
		SwapEnds	31.2	9.9	37.6	1.12

[1]Average computation time for destroy and repair operation.

In both instances we observe that, among the three destroy operators, RemoveRandom is used most often, followed by RemoveWorst. The success rates of these operators are close to each other, however the average improvement made by RemoveWorst is substantially higher. Both operators are succeeded by the SolveExact repair operator, which constructs the best feasible solution. We acknowledge that using this operator after an arbitrary destroy operation gives rise to an objective improvement in many cases. This improvement, however, turns out to be stronger when the set of duties to be destroyed is consciously selected. We therefore consider the RemoveWorst operator as the most important in our heuristic. The number of duties to be destroyed and repaired is identical for both destroy operators, leading to similar average computation times. In both instances, slightly more computation time is consumed by the RemoveWorst operator, returning solutions in which the average improvement is higher.

The SwapEnds operator, conversely, is used relatively less often and less often an improvement is registered. This is illustrated especially in the `Center` instance. Since we perform all swaps that yield an objective improvement, its average improvement outperforms that of the other destroy operators. Additionally, since we efficiently identify all swaps in parallel instead of retrieving the new solution exactly, the average computation time consumed by this operator is lower compared to the other operators.

The performance of the SolveExact repair operator can be concluded from the destroy operator performance. Since the repair operator is instantiated if and only if RemoveWorst or

RemoveRandom is performed, the (average) number of times it is selected and the number of improvements equals the sum of both destroy operators. Similarly, the average improvement constitutes the weighted average of these operators. Given that the preceding destroy operators require negligible destroy time, we conclude that most of the computation time of the ALNS heuristic is spent in the SolveExact operator. In particular, this results in fewer iterations performed on average in the more challenging `Center` instance.

6.4 Improving Roster Fairness in a Large Instance

Applying our ALNS heuristic to the medium-sized `East` and `Center` instances shows that the total amount of penalty corresponding to crew members performing unattractive work can be reduced within relatively short computation time. In addition, a substantial share of the tasks uncovered in the DRCPP solution can be inserted by the insertion procedure.

In this section, we focus on another drawback of the current solution approach. Since the DRCPP is currently only solved for sub-parts of the network, profitable assignments of attractive and unattractive work over multiple regions are not considered. For this reason, we apply our ALNS heuristic to the `Joint` instance containing the crew members of both the `East` and `Center` instance. For each day in the planning horizon, we instantiate the heuristic with the same DRCPP solutions retrieved in the two medium-sized instances. That is, we do not solve the DRCPP again. Although this problem can be solved efficiently for both instances in isolation, considering identical input solutions allows us to fairly compare the performance of our heuristic when applied to the large instance.

For each day in the planning horizon, we first apply the insertion procedure to the union of the initial DRCPP solutions in order to insert the tasks still uncovered. Table 6.9 shows how the number of uncovered tasks in the `Joint` instance decreases during the insertion procedure. Given the fact that we consider the DRCPP solutions from the `East` and `Center` instance together, the initial number of uncovered tasks indeed equals the sum of tasks initially left uncovered in these instances.

Table 6.9: Evolution of the number of uncovered tasks when applying the insertion procedure to the `Joint` instance.

Instance	Method	Total	Median	Min	Max
	DRCPP	783	14	4	66
`Joint`	Core problem	605	9	1	61
	Sub-problems	572	9	1	56

When applying the insertion procedure to the large instance, we observe a similar tendency as in the medium-sized instances. The number of uncovered tasks is reduced by about 27%, most of which is again achieved by solving the insertion core problems. We are able to insert 211 of the uncovered tasks, which is substantially more than in the instances `East` and `Center` (132 tasks). Finally, it turns out that inserting uncovered tasks in the `Joint` instance is also possible more often: A successful insertion procedure is performed on 43 days of our planning horizon.

Inserting more tasks is possible since we now consider the duties of both medium-sized instances combined, thereby enlarging the solution space of the insertion core problem. On the other hand, this creates a computationally more complex problem to solve. For this reason, we relax some of the settings for solving the insertion core problem in the `Joint` instance. In particular, we use a heuristic pricing algorithm (similar to the one we use in the DRCPP) to find candidate duties. Moreover, solving the insertion model is terminated after reaching a predefined time limit equal to $\frac{1}{3}$ of the time limit imposed for the ALNS heuristic. For the insertion sub-problems, we use identical settings as in the medium-sized instances.

The impact of inserting the uncovered tasks on the number of empty duties and deadheading tasks is also similar as encountered in the medium-sized instances. Remember that, although having very different characteristics, almost all empty duties were exploited in both instances. Moreover, for instance `East` we were able to reduce the number of deadheading tasks, whereas for the larger instance `Center` containing more empty duties, this number increased. In the `Joint` instance, again nearly all empty duties are filled, leaving only 12 remaining after applying the heuristic. Since the majority of the (empty) duties originates from the `Center` instance, we still observe an increase in the number of deadheading tasks, although less severe compared to considering instance `Center` alone. Thanks to the dampening influence of instance `East`, the 211 uncovered tasks are inserted only against 108 additional deadheading tasks (4,783 to 4,891). In this respect, considering the larger instance is beneficial both in terms of inserting the uncovered tasks as in efficiently constructing new duties.

We now study the effect of combining instance `East` and instance `Center` on reducing the total amount of unattractive work, and on shifting the unattractive work between the crew members. Table 6.10 first shows the objective contribution for the different attributes when applying the ALNS heuristic to the `Joint` instance. In italics, we specify the share of penalty originating from crew members corresponding to either the `East` or `Center` instance. Despite using the same initial solutions as in the medium-sized instances, remember that the objective value of the heuristic corresponds to the sum of history penalties assigned to all crew members, based on the duties inside their rolling horizon window. The penalties are essentially registered when the duties are added to the rosters of the crew members. Since these duties might have been modified by the ALNS heuristic, the resulting objective values might also differ.

Table 6.10: Objective contribution per attribute in the `Joint` instance.

Method	Instance	Objective	Type-A	Aggression	Double Decker	Duty Length
DRCPP	`Joint`	1,088,518.73	94,923.96	96,471.31	42,614.34	854,509.12
	`East`	*471,323.57*	*767.47*	*0.00*	*42,128.97*	*428,427.13*
	`Center`	*617,195.16*	*94,156.49*	*96,471.31*	*485.37*	*426,081.99*
ALNS	`Joint`	1,029,316.48	59,665.85	83,228.08	33,864.44	852,558.11
	`East`	*460,197.06*	*95.97*	*0.00*	*33,815.64*	*426,285.45*
	`Center`	*569,119.42*	*59,569.88*	*83,228.08*	*48.80*	*426,272.66*
Delta	`Joint`	-59,202.25	-35,258.11	-13,243.23	-8,749.90	-1,951.01
	`East`	*-11,126.51*	*-671.50*	*0.00*	*-8,313.33*	*-2,141.68*
	`Center`	*-48,075.74*	*-34,586.61*	*-13,243.23*	*-436.57*	*190.67*

The total amount of penalties assigned decreases by about 5%. Again, a penalty reduction is achieved across all attributes and in all days of our planning horizon. Although the exact objective values are hard to compare to the medium-sized instances, in absolute terms the decrease achieved per attribute is very similar to the reduction reported in the previous section. Most importantly, we observe that applying the ALNS heuristic to the `Joint` instance leads to a slight increase in the duty length penalty for crew members from the `Center` crew bases. Remember that relatively many crew members from instance `East` violate this attribute. Combining the two medium-sized instances demonstrates that part of this unattractive work is shifted towards the `Center` crew members, thereby obtaining a more balanced division of the work. Because of the considerable impact of the Duty Length attribute on the overall objective, similar shifts for other attributes are unfortunately not clearly visible from the obtained results.

According to the attribute objective contributions, applying the ALNS heuristic to the `Joint` instance thus leads to a more fair distribution of the duty lengths. Likewise, the number of crew members violating the Duty Length attribute, as displayed in Table 6.11, decreases as well. In other words, the heuristic successfully shifts part of the unattractively long duties between the different regions. In this case, the excessively long duties initially assigned to the `East` crew members are to some extent transferred to the `Center` crew members. The number of crew members violating the Duty Length attribute for the former group reduces, against an increase in the latter group. This eventually results in the overall number of violators to be lower.

Table 6.11: Number of crew members per adherence category per attribute in the `Joint` instance.

		Type-A			Aggression			Double Decker			Duty Length		
Instance	Method	V	NV	A	V	NV	A	V	NV	A	V	NV	A
`Joint`	DRCPP	66	5,764	14,246	198	3,569	16,309	82	2,376	17,618	3,451	16,610	15
	ALNS	20	5,528	14,528	120	3,681	16,275	41	2,282	17,753	3,425	16,631	20

The computation time used to solve the `Joint` instance is displayed in Table 6.12. The DRCPP is not solved again, although the problem can be solved for multiple instances at the same time in parallel. The average computation time for this component then corresponds to the average of the maximum time needed to solve all instances. In our case, the average computation time required for the DRCPP would likely approach the time spent solving the most complex medium-sized instance `Center`.

Table 6.12: Average computation time in seconds for each solution component in the `Joint` instance.

Instance	DRCPP	Core problem	Sub-problems	ALNS
`Joint`	NA	103	123	911

Contrary to instances `East` and `Center`, a more substantial share of the insertion procedure is now used solving the insertion core problems. Indeed, combining the two instances results in a more complex insertion core problem. For most of the days, however, the realized computation time remains quite moderate thanks to the relaxations applied. We reach the imposed time limit for solving the insertion core problem on 11 days of the planning horizon. On average, it still takes a little longer to solve all insertion sub-problems in the `Joint` instance. Remember that these problems are solved sequentially. Combining the medium-sized instances leads to more tasks to be explored, thereby increasing the computation time required. The ALNS heuristic itself is terminated prematurely on 5 days, implying that still about once every ten days the heuristic is unable to improve the crew rosters over the predefined maximum number of subsequent iterations.

In order to put the prematurely termination of the heuristic into context, we analyze the performance of the destroy operators in Table 6.13. First, we observe that the average number of iterations performed by the heuristic is comparable to the number of iterations performed in the medium-sized instances. Likewise, the average improvement and computation time for the operators are in line with that of instance `East` and `Center`. Remember that we perform all experiments with identical operator parameter values. In particular, this implies that the RemoveWorst and RemoveRandom operators consistently remove the same number of duties from the incumbent solution. This number is kept constant to prevent the heuristic from consuming a disproportional amount of time repairing the solution. With respect to the SwapEnds operator, note that we again only restrict ourselves to swapping crew members from the same crew base, resulting in the average improvement being in line with the results for the medium-sized instances.

Similar to the medium-sized instances, the RemoveRandom operator is selected most often. However, in the `Joint` instance the success rate of this operator appears to be substantially higher compared to the other operators. We find that in many cases the improvement made in RemoveRandom is marginal, with the percentual improvement only in the second or third decimal. Nevertheless, since the resulting solution is still registered as the new best, the operator score (and indirectly the weight) is updated favorably. Albeit reflecting a similar effect as in the medium-sized instances, we observe that the `Joint` instance simply brings more opportunities to attain these (marginal) improvements. This causes the success rate to be higher. On the other hand, using this greedy operator more often also enhances the probability of finding subsequent non-improving solutions. This could result in the heuristic terminating prematurely frequently. In conclusion, we acknowledge that thorough additional parameter tuning is required in order to obtain more satisfactory operator performance.

Table 6.13: Average performance of the destroy operators in the `Joint` instance.

Instance	Iterations	Operator	Selected	Improvements	Avg. improvement	Time[1]
		RemoveWorst	81.9	42.0	12.3	6.25
`Joint`	226.3	RemoveRandom	98.3	62.8	7.3	3.35
		SwapEnds	46.1	14.8	38.1	1.22

[1]Average computation time for destroy and repair operation.

6.5 Sensitivity Analysis

In this section we justify our choices regarding the operator-specific parameters of our ALNS heuristic, as outlined in Section 6.2. In particular, these choices relate to the parameters of the destroy operators RemoveWorst and RemoveRandom. As previously stressed, we regard the RemoveWorst operator as our most important operator, especially since the average improvement when consciously selecting the duties to remove exceeds that of a random selection.

Remember that the RemoveWorst operator consists of two parameters that need to be set. First, the number of duties to be removed corresponding to the crew members with the worst history scores, `numWorstDuties`. Second, for each of these duties we remove the `numSimilarDuties` duties corresponding to crew members performing a similar duty. We define a *configuration* as a composition (`numWorstDuties`, `numsimilarDuties`) of these two parameters. In our experiments, we have used the configuration $(3, 10)$. In this section, we show for varying configurations the trade-off between the average improvement and the time required to repair the corresponding destroyed solution. This repair operation, SOLVEEXACT, involves generating candidate duties by column generation and solving an $\mathcal{NP}$-hard binary linear program. Therefore, it is of considerable importance to consciously establish the configuration.

Since our aim is to compare the performance of various parameter configurations within the ALNS framework, we perform all experiments by means of the following set-up. We solely focus on the optimization problem for instance `East`. This instance initially contains relatively little empty duties, allowing the ALNS heuristic to be applied most efficiently. We first generate DRCPP solutions both for the warm-up period and our 49-day planning horizon. We do so without applying our heuristic. Then, the ALNS heuristic is performed for each day of the planning horizon, where we allow only the REMOVEWORST operator to be used. Moreover, we refrain from applying the insertion procedure, and devote to each day the full computation time of 15 minutes. This way, all parameter experiments are performed on the exact same set of duties, and we can make valid statements on the performance of the respective configuration.

Table 6.14 shows for various configurations of the REMOVEWORST operator the average number of iterations performed per day, the average number of improvements, and the average objective improvement. Furthermore, the average time required to perform the corresponding repair operation is reported. In all configurations, we have `numWorstDuties` $\in \{1, 3, 5\}$ and `numSimilarDuties` $\in \{1, 5, 10, 20\}$.

Table 6.14: Results for different configurations for the REMOVEWORST operator. The repair time is reported in seconds.

Configuration	Iterations	Improvements	Avg. improvement	Time
$(1, 1)$	559.2	1.0	3.9	0.9
$(1, 5)$	590.3	3.8	7.1	0.8
$(1, 10)$	557.9	9.7	5.8	1.0
$(1, 20)$	305.8	20.4	6.2	2.4
$(3, 1)$	538.8	6.4	2.6	1.1
$(3, 5)$	460.3	27.0	4.8	1.5
$(3, 10)$	178.8	35.1	5.2	4.5
$(3, 20)$	23.1	15.1	13.8	39.4
$(5, 1)$	497.2	16.6	2.3	1.5
$(5, 5)$	240.3	39.9	4.3	3.2
$(5, 10)$	42.8	23.0	8.9	20.8
$(5, 20)$	6.0	5.5	42.9	166.8

As we expect, increasing the size of the configuration progressively increases the required repair time. Remember that the impact of `numWorstDuties` is biggest, since the total number of duties involved corresponds to `numWorstDuties` + `numWorstDuties` · `numSimilarDuties`. Considering more duties also results in more improvement, as it leads to more freedom when repairing the solution. Furthermore, we observe that the success rate increases the more bad duties we consider, all else equal. Likewise, increasing the number of similar duties gives rise to an increase in the average improvement: Apart from the worst duties, the repair operator also exploits favorable exchanges between the other duties. Including more duties that are similar to each other, then, is likely to provide more of these opportunities.

In Figure 6.2, we visualize the trade-off between repair time and the objective improvement. In order to fairly compare all configurations, we compute the *normalized objective improvement* by multiplying the success rate of each configuration by its average improvement. This way, configurations having a high average improvement and low success rate (and vice versa) can be compared to each other. The normalized objective improvement can alternatively be interpreted as the average improvement achieved per iteration, rather than per improvement. The points in this logarithmic plot resemble a linear trend, illustrating the exponential increase in repair time when the number of duties in the configuration increases.

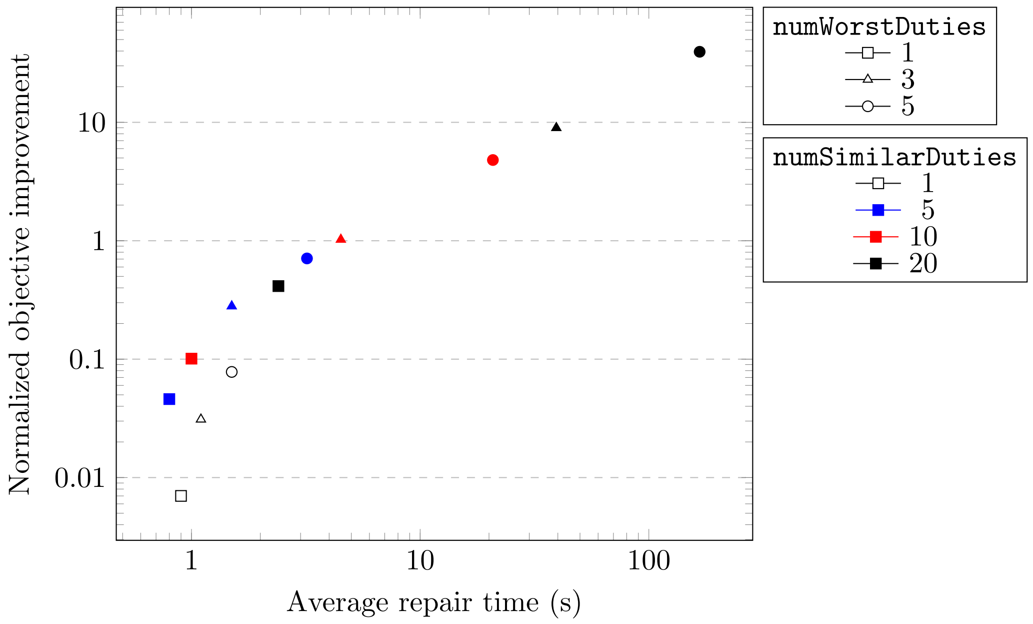

Figure 6.2: Effect of varying configurations for the RemoveWorst operator on repair time and the normalized objective improvement.

Based on the results of our experiments, we conclude that four of the configurations are worth considering: $(3, 10), (3, 20), (5, 5)$, and $(5, 10)$. We disregard the configuration $(5, 20)$ due to its excessive repair time. All remaining configurations are excluded because of their normalized objective improvement, primarily caused by their inferior success rates. The configurations still under consideration can be classified into two groups. On the one hand, the configurations

$(3, 10)$ and $(5, 5)$ both repair the solution relatively quickly, but have modest success rates. On the other hand, the success rate is higher for configurations $(3, 20)$ and $(5, 10)$. This does however come at the cost of increased repair time. In terms of normalized objective improvement, the configurations in either group are close to each other. However, the former group requires substantially less computation time compared to the latter, for which reason it better scales with the SwapEnds operator in our heuristic, which also proposes a repaired solution. For this reason, we prefer these configurations over those in need of longer computation time.

Among the configurations examined, we conclude that the configuration $(3, 10)$ best balances the probability of finding an improving solution, the improvement of the solution itself, and the repair time required. In order to facilitate an equal comparison with the RemoveRandom destroy operator, the value for `numRandomDuties` is set accordingly to 33.

Chapter 7

Conclusions

In this thesis, we study the crew planning problem at Netherlands Railways (NS). This goal of this problem is to construct rosters in which the amount of attractive and unattractive work as defined by the Sharing-Sweet-and-Sour rules is balanced among the crew members. The ongoing *Fundamenteel Spoor* research project proposes to solve the crew planning problem for individual crew members and in the operational planning phase, rather than per crew base and in the tactical planning phase. This led to the introduction of the Dynamic Railway Crew Planning Problem (DRCPP) with Fairness over Time by Van Rossum et al. (2022), which strives to cover all tasks and attain balanced crew rosters that satisfy the Sharing-Sweet-and-Sour rules as much as possible.

To further enhance the DRCPP solution, we develop an Adaptive Large Neighborhood Search meta-heuristic containing three destroy operators and one repair operator. We evaluate the performance of this heuristic over a 49-day planning period with rolling history window on two medium-sized instances and one large instance of the Dutch railway network. We find that our heuristic successfully decreases the amount of unattractive work assigned in all instances. Additionally, a substantial amount of the tasks that were initially left unassigned is inserted. Moreover, our heuristic is able to deal with larger instances that previously could not be solved. Here, attractive and unattractive work is shifted throughout the network, thereby enhancing the fairness of the overall set of rosters.

The modularity of the heuristic demonstrates its applicability to a variety of instances and with a variety of parameter configurations. In particular, the framework can be tuned entirely towards the practical needs and desires of the railway operator. Nonetheless, the current performance of the heuristic seems to be sensitive to the characteristics of the instance to which it is applied. Overall, the heuristic tends to reduce the amount of unattractive work in the solution by exploiting the duties left empty in the DRCPP solution. When the number of empty duties is initially high, we eventually find the number of deadheading tasks to rise when applying the heuristic. Additionally, more crew members violate the average duty length requirement. At the same time, our heuristic does effectively enhance the DRCPP solutions in case the number of empty duties is more moderate. In addition, considering larger instances enables attractive and unattractive work to be exchanged throughout the network. This indicates that our method still provides high potential for NS.

This thesis touches upon a few aspects that deserve future attention. First of all, in order to inspect its sensitivity to the initial number of empty duties, we need the heuristic to limit exploiting these using excessive deadheading tasks. Remedies for this can be achieved both in a primal (via the objective function) and dual (in the construction of the paths) way. Furthermore, additional tuning of the parameters in the heuristic is required. In particular, this holds for the parameters corresponding to the overall framework. Determining a more sensible composition for this set is necessary to reduce the probability of the heuristic terminating prematurely, and to more deliberately update the operator scores. Moreover, operator-specific parameters may be additionally tuned in order to attain superior success rates.

Apart from considering adjustments to the heuristic itself, we also see valuable directions in studying its behavior when using different input. First, we have applied our method only to the crew planning problem for the guards. Considering the drivers, alternatively, involves a number of additional features, such as incorporating the route and rolling stock knowledge. Including these is likely to limit the opportunities to exchange work with other crew members. It is therefore interesting to analyze to what extent incorporating the drivers affects the total improvement potential of our heuristic. Similarly, we wonder how the heuristic behaves in even bigger instances. The powerful crew planning solver currently in use at NS (Abbink et al., 2011) is expected to solve the DRCPP for the entire country. It is interesting to see to what extent additional improvement can be achieved by applying the heuristic as a local re-optimization method. The fact that our heuristic improves the solution in a short amount of time is promising in terms of its applicability in the planning process. Finally, we have regarded the assignment of the templates as input. Optimizing this assignment before applying the heuristic, however, might give rise to the heuristic being able to improve the eventual rosters even more.

Bibliography

Abbink, E. (2014). *Crew Management in Passenger Rail Transport* (Doctoral dissertation). Erasmus Research Institute of Management (ERIM).

Abbink, E., Albino, L., Dollevoet, T., Huisman, D., Roussado, J., & Saldanha, R. L. (2011). Solving Large Scale Crew Scheduling Problems in Practice. *Public Transport*, *3*(2), 149–164.

Abbink, E., Fischetti, M., Kroon, L., Timmer, G., & Vromans, M. (2005). Reinventing Crew Scheduling at Netherlands Railways. *Interfaces*, *35*(5), 393–401.

Abbink, E., Huisman, D., & Kroon, L. (2018). Railway Crew Management. *Handbook of optimization in the railway industry* (pp. 243–264). Springer.

Bampis, E., Escoffier, B., & Mladenovic, S. (2018). Fair resource allocation over time. *AAMAS 2018-17th International Conference on Autonomous Agents and MultiAgent Systems*, 766–773.

Bertsimas, D., Farias, V. F., & Trichakis, N. (2011). Fairness, Efficiency and Flexibility in Organ Allocation for Kidney Transplantation. *Operations Research*, *61*(1), 73–87.

Bertsimas, D., Farias, V. F., & Trichakis, N. (2012). On the Efficiency-Fairness Trade-off. *Management Science*, *58*(12), 2234–2250.

Borndörfer, R., Schulz, C., Seidl, S., & Weider, S. (2017). Integration of duty scheduling and rostering to increase driver satisfaction. *Public Transport*, *9*(1), 177–191.

Breugem, T., Dollevoet, T., & Huisman, D. (2022a). Is Equality Always Desirable? Analyzing the Trade-Off between Fairness and Attractiveness in Crew Rostering. *Management Science*, *68*(4), 2619–2641.

Breugem, T., Van Rossum, B. T. C., Dollevoet, T., & Huisman, D. (2022b). A Column Generation Approach for the Integrated Crew Re-Planning Problem. *Omega*, *107*, 102555.

do Carmo Martins, L., & Silva, G. P. (2019). An Adaptive Large Neighborhood Search Heuristic to Solve the Crew Scheduling Problem. *Smart and digital cities: From computational intelligence to applied social sciences* (pp. 45–64). Springer.

Er-Rbib, S., Desaulniers, G., Elhallaoui, I., & Munroe, P. (2021). Preference-based and cyclic bus driver rostering problem with fixed days off. *Public Transport*, *13*(2), 251–286.

Hartog, A., Huisman, D., Abbink, E., & Kroon, L. (2009). Decision support for crew rostering at ns. *Public Transport*, *1*(2), 121–133.

Heil, J., Hoffmann, K., & Buscher, U. (2020). Railway crew scheduling: Models, methods and applications. *European Journal of Operational Research*, *283*(2), 405–425.

Huisman, D., Kroon, L., Lentink, R., & Vromans, M. (2005). Operations Research in passenger railway transportation. *Statistica Neerlandica*, *59*(4), 467–497.

Jütte, S., Müller, D., & Thonemann, U. W. (2017). Optimizing railway crew schedules with fairness preferences. *Journal of Scheduling*, *20*(1), 43–55.

Lodi, A., Olivier, P., Pesant, G., & Sankaranarayanan, S. (2022). Fairness over time in dynamic resource allocation with an application in healthcare [Advance online publication]. *Mathematical Programming*.

Maenhout, B., & Vanhoucke, M. (2010). A hybrid scatter search heuristic for personalized crew rostering in the airline industry. *European Journal of Operational Research*, *206*(1), 155–167.

Perumal, S. S., Dollevoet, T., Huisman, D., Lusby, R. M., Larsen, J., & Riis, M. (2021). Solution approaches for integrated vehicle and crew scheduling with electric buses. *Computers & Operations Research*, *132*, 105268.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, *34*(8), 2403–2435.

Potthoff, D., Huisman, D., & Desaulniers, G. (2010). Column Generation with Dynamic Duty Selection for Railway Crew Rescheduling. *Transportation Science*, *44*(4), 493–505.

Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, *40*(4), 455–472.

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. *Principles and Practice of Constraint Programming-CP98*, 417–431.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research*, *226*(3), 367–385.

Van Rossum, B. T. C., Dollevoet, T., & Huisman, D. (2022). *Dynamic Railway Crew Planning with Fairness over Time* [Econometric Institute Report Series, EI 2022-10].

Wolbeck, L. A. (2019). *Fairness Aspects in Personnel Scheduling* (Discussion Paper). http://hdl.handle.net/10419/210988