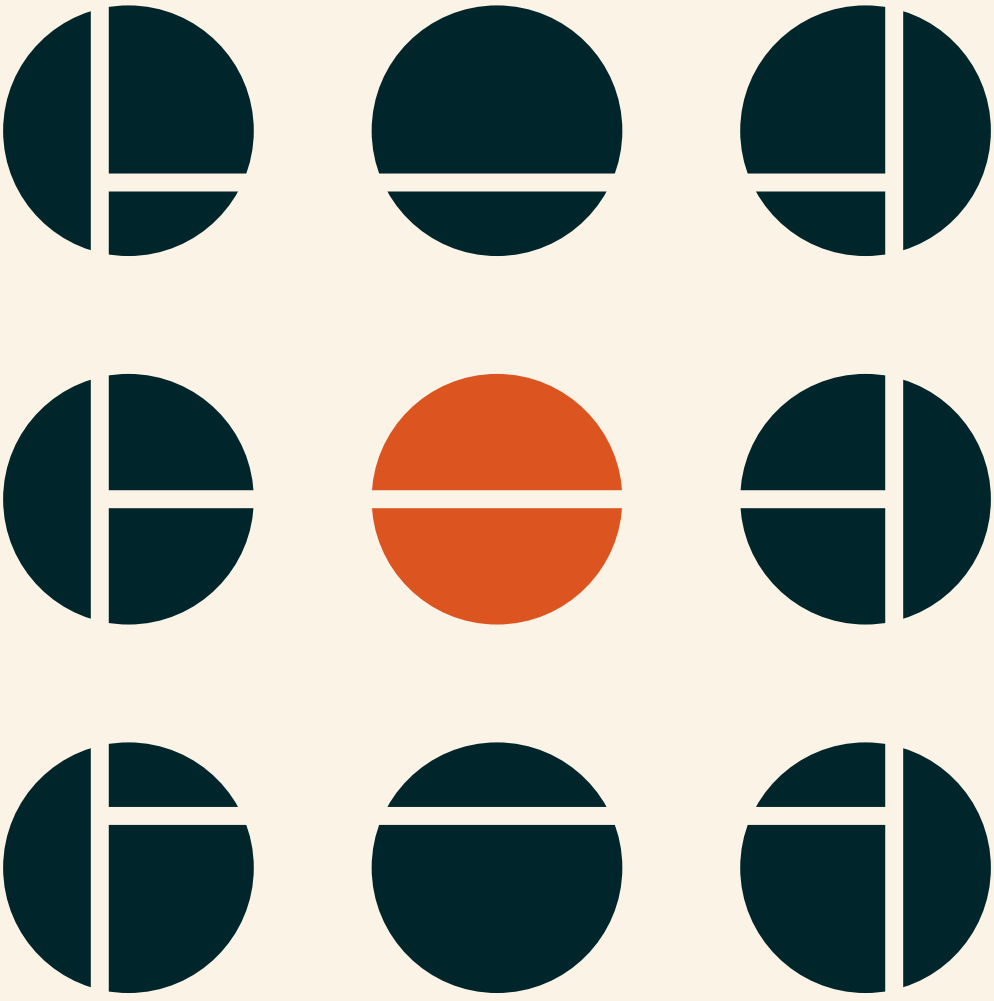
Bartholomeüs Theodorus Cornelis van Rossum

Optimising Fair Work Allocation

Applications in Railway Crew Planning



Optimising Fair Work Allocation

Applications in railway crew planning

Optimising Fair Work Allocation: Applications in railway crew planning

Optimalisatie van eerlijke werkverdeling:
Toepassingen in de planning van spoorpersoneel

Thesis

to obtain the degree of Doctor from the
Erasmus University Rotterdam
by command of the
Rector Magnificus

Prof.dr.ir. A.J. Schuit

and in accordance with the decision of the Doctorate Board.

The public defence shall be held on

Friday 16 May 2025 at 13:00 hours

by

Bartholomeüs Theodorus Cornelis van Rossum
born in Nijmegen, the Netherlands.

Erasmus University Rotterdam

Doctoral committee

Promotor:	Prof.dr. D. Huisman
Other members:	Prof.dr. E. Rönnberg
	Dr.ir. J.M. van den Akker
	Dr. R. Spliet
Co-promotor:	Dr. T.A.B. Dollevoet

Erasmus Research Institute of Management - ERIM
The joint research institute of the Rotterdam School of Management (RSM)
and the Erasmus School of Economics (ESE) at the Erasmus University Rotterdam
Internet: www.erim.eur.nl

ERIM PhD Series in Research in Management, 600
ERIM reference number: EPS-2025-600-LIS
ISBN 978-90-5892-732-3

Cover image: Kira van Landschoot, www.vankira.nl
Cover design: PanArt, www.panart.nl
Print: OBT bv, www.obt.eu

This publication (cover and interior) is printed on certified FSC® paper PlanoSuperior.

Acknowledgements

A PhD candidacy provides the luxurious freedom to work on a wide range of problems, study a variety of techniques, and explore the world. This inherent freedom also plays right into my less favourable character traits, which some would describe as laziness and sloppiness. It goes without saying that the past five years consisted of a sequence of highs and lows. One memorable low pertained to a rejected paper, accompanied by the following anonymous reviewer comment: "This work contains no significant contribution in terms of modelling or method". Figuring out the corresponding chapter is left as an exercise for the reader. As in life, the best thing you can do at such moments is to keep on going, relying on the steadily growing confidence that, indeed, each low will be followed by a high. Now, it feels appropriate to thank the people around me.

In the first place, I would like to thank my supervisors Twan and Dennis. Working with you was a great experience, and I could not have wished for a better supervisory team. You have always granted me a great sense of ownership, letting me explore sidetracks and trusting that I would eventually return to the main road. Dennis, you have consistently kept the bigger picture in mind and possess an unmatched skill in positioning academic research. Moreover, you exposed me to many interesting opportunities along the way. Twan, I hope you forgive me for knocking on your office door five times a day with a cup of coffee and the need for small talk. I admire your great sense of detail, and share your passion for sophisticated yet elegant methods.

Second, I thank Andrea and Rui for their generous hospitality in New York. It was a joy to collaborate with you, simultaneously exploring a new university, city, and field of research. Andrea, thank you for taking a leap of faith and accepting to host me, and for being a refreshingly honest and personal researcher. Rui, alternating

proof-writing on the whiteboard walls with rounds of ping-pong was tremendously fun. Thank you for making me feel home on Roosevelt Island, guiding me through the culinary scene of New York, and exposing me to a new level of mathematical rigour. I sincerely hope we continue to collaborate.

Third, I want to thank prof.dr. Elina Rönnberg, dr.ir. Marjan van den Akker, and dr. Remy Spliet for taking place in my inner doctoral committee. Similarly, my gratitude goes out to prof.dr. Ralf Borndörfer, dr. Thomas Breugem, and Wouter Koolmees for completing the opposition. The presence of so many experts at my defence is humbling.

This dissertation would not have been possible without NS, who partly funded my position and provided me with excellent support throughout the years. Wilmet, Joël, and Marije, it was extremely rewarding to work with you on crew planning for Fundamenteel Spoor. I am impressed by your rigour and dedication. Gábor, thank you for the stimulating discussions and healthy dose of cynicism. Finally, I thank all colleagues of Digitalization of Operations, formerly known as π, for their warm welcome.

Next, I thank all colleagues and PhDs of the Econometric Institute for their company throughout the past years. Remy and Wilco, for sharing non-academic interests in football and chess, respectively. My predecessors and successors at NS, Thomas, Rowan, Rolf, Pedro, Danny, and Khue. Thomas, it was a joy to collaborate on my first full paper. Rolf, I look forward to working with you in Eindhoven. A special thank you goes out to Mette and Rick, with whom I have had the unique pleasure of sharing this academic journey. Rick, celebrating our ROADEF award with cocktails and a view of the Manhattan skyline remains an unforgettable experience.

Finally, I thank those closest to my heart. My friends, for gently reminding me about the opportunity cost of doing a PhD. My brothers, for not taking me too seriously and keeping me down to earth. My dad, for showing me there is beauty in everything as long as you are willing to look for it, and for being the first to read my pre-prints, only to return with three (non-)related books from your personal library. My mom, for being my greatest fan. My extended family in Dordrecht, for embracing me with open arms, and for illustrating the practical consequences of unfairness when I am consistently handed the last slice of pie. Else-Marie, for letting me explore the world with open eyes, for pushing me to my limits, for calling out my occasional nonsense, and for making every day a little brighter.

Table of contents

Chapter 1

Introduction

...a landowner who went out early in the morning to hire workers for his vineyard. He agreed to pay them a denarius for the day and sent them into his vineyard. About nine in the morning he went out and saw others standing in the marketplace doing nothing. He told them, 'You also go and work in my vineyard, and I will pay you whatever is right.' So they went. He went out again about noon and about three in the afternoon and did the same thing. About five in the afternoon he went out and found still others standing around. He asked them, 'Why have you been standing here all day long doing nothing?' 'Because no one has hired us,' they answered. He said to them, 'You also go and work in my vineyard.' When evening came, the owner of the vineyard said to his foreman, 'Call the workers and pay them their wages, beginning with the last ones hired and going on to the first.' The workers who were hired about five in the afternoon came and each received a denarius. So when those came who were hired first, they expected to receive more. But each one of them also received a denarius. When they received it, they began to grumble against the landowner. 'These who were hired last worked only one hour,' they said, 'and you have made them equal to us who have borne the burden of the work and the heat of the day.' But he answered one of them, 'I am not being unfair to you, friend. Didn't you agree to work for a denarius?... - Matthew 20:1-14

Regardless of what the parable of the vineyard workers teaches us, many contemporary employers would receive strong complaints about such payment schemes. Generally, workers wish to be rewarded proportionally for their efforts, and consider it *fair*

if similar employees are treated in a similar manner. This not only applies to monetary payoffs, but extends to the workload and quality of work. Deviating from such conceptions of fairness can induce jealousy in the workplace, and lead to deteriorated working conditions for the worst-off.

In many real-life settings, the size and complexity of planning problems necessitate the use of operations research (OR) techniques for determining work assignments. As such, fairness considerations cannot be ignored by researchers and practitioners in the field of OR. In this thesis, we employ OR techniques to develop models and methods for fairly allocating work. In the first part of this thesis, we take an applied perspective and focus on a railway crew planning application. In the second part, we take a more theoretical point of view and examine more general fairness-oriented work assignment problems.

In Part I of this thesis, we study the feasibility of a novel crew planning process at Netherlands Railways (NS), the largest public transport operator in the Netherlands. Operating on the dense Dutch railway network, it serves over one million passenger trips on an average working day (Nederlandse Spoorwegen, 2023). NS employs approximately 3,200 drivers and 2,800 guards to serve all these trips (Nederlandse Spoorwegen, 2022). When assigning work to crew members, the operator must comply with complicated rules from collective labour agreements. The size and complexity of crew planning at NS necessitate the use of sophisticated decision support systems based on OR techniques.

To remain a competitive employer, it is crucial that the crew planning process results in attractive and fair working schedules for all employees. For example, railway guards in the Netherlands face over 1,000 serious aggression incidents per year (Nederlandse Spoorwegen, 2023), and no crew member should be disproportionately exposed to such harm. The current crew planning process does not adequately meet these needs. As such, NS is considering the implementation of a new planning process, with the aim of obtaining a more robust, fair, and flexible crew plan. In Part I of this thesis, we propose mathematical models and solution methods for the optimisation problems arising in this new setting.

In Part II of the thesis, we take a slightly broader perspective and propose methods and models for a general class of fairness-oriented assignment problems. We focus on problem settings where pieces of work are constructed and assigned to workers, as in, for example, crew scheduling and vehicle routing. While there is a growing

interest in considering equity in optimisation problems, many such problems remain very challenging from a computational point of view. Developing efficient algorithms for fairness-oriented optimisation is the main interest of Part II of this thesis.

The remainder of this chapter is structured as follows. In Sections 1.1 and 1.2, we provide introductions to railway crew planning and fair optimisation, respectively. We sketch the outline of the remainder of the thesis in Section 1.3, and summarise the main contributions in Section 1.4.

1.1 Railway Crew Planning

The planning process at a typical railway operator consists of multiple steps performed sequentially, as illustrated in Figure 1.1. Throughout the planning process, the railway operator aims to construct a cost-efficient planning that provides high-quality service to passengers, is robust to disruptions on the day of operations, and shapes good working conditions for crew members.

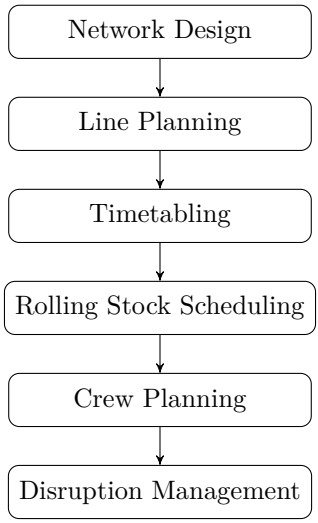


Figure 1.1: Overview of the railway planning process.

We now describe each of the planning steps in more detail. The goal of network design is to determine which parts of the railway track currently provide insufficient capacity and require infrastructural upgrades. In line planning, a set of lines and their frequencies are chosen. Each line reflects a direct train connection between a sequence of stations, and the goal is to satisfy passenger demand while minimising

travel time and the number of passenger transfers. Next, in timetabling the exact arrival and departure times of lines are determined, taking into account the limited capacity of stations and tracks. The timetable specifies a set of scheduled train trips, characterised by a start and end station and start and end time. In rolling stock scheduling, train units are assigned to trips such that all trips feature a sufficient number of passenger seats. The goal of crew planning is to assign a driver and one or more guards to each train trip, while respecting labour rules and minimising crew costs. Crew planning is typically decomposed into crew scheduling and crew rostering. Finally, a significant amount of disruption management takes place on the day of operations. Here, the timetable, rolling stock schedule, and crew plan are dynamically adjusted to account for unforeseen disturbances occurring in operation.

We refer the interested reader to Borndörfer et al. (2018) for a complete overview of all individual railway planning steps. In the following sections, we describe crew scheduling and crew rostering in more detail, and sketch the current and proposed crew planning processes at Netherlands Railways.

1.1.1 Crew Scheduling

The goal of crew scheduling is to assign all tasks, i.e., scheduled train trips, to crew members in the form of *duties*. A duty is a sequence of tasks that can be performed consecutively and that satisfies a set of labour rules specified in the collective labour agreements. First, it must start and end at the same crew base, allowing each crew member to return home after their shift. Second, each duty must contain a meal break at a station with a dedicated canteen. The meal break must be positioned approximately halfway the duty. Third, the duration of a duty may not exceed a maximum length, which depends on the starting time of the duty. For example, duties starting very early or very late on the day have a lower maximum length than duties starting around noon. Finally, when scheduling duties for drivers, we must ensure that the driver possesses sufficient rolling stock and route knowledge to be allowed to perform all tasks in the duty. Figure 1.2 shows an example of a duty starting in Rotterdam.

The main objectives in crew scheduling are as follows (see Heil et al. (2020) for a recent overview of crew scheduling literature). While the number of employed crew members largely determines the fixed crew costs, the efficiency of the crew schedule has an effect on variable crew costs. In addition, some robustness considerations are taken into account, for example, by minimising the number of rolling stock transitions

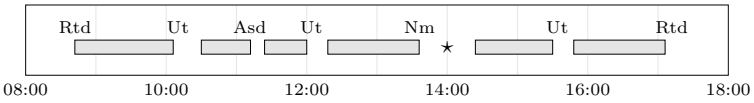


Figure 1.2: Example of a duty at crew base Rotterdam (Rtd). For each task the start station (top left) and/or end station (top right) are shown. The star indicates a meal break.

or imposing long minimum transition times between tasks. Most importantly, Netherlands Railways also pays strong attention to the attractiveness and fairness of the crew schedule. This is achieved by incorporating so-called *Sharing-Sweet-and-Sour* rules.

The Sharing-Sweet-and-Sour rules are a set of restrictions, agreed upon by the board and working council, which aim to ensure a fair and attractive work package for drivers and guards at NS. Initially introduced in 2001 to resolve large nation-wide strikes (Abbink et al., 2005), they specify that *sweet* (attractive) and *sour* (unattractive) work is balanced fairly over the various crew bases. Here, the quality of work is measured along several attributes, each of which is either sweet or sour. Some important examples include the duty length (sour), the fraction of work on type-A trains (sweet), and the fraction of work with high risk of passenger aggression (sour). The average scores of the duties assigned to each crew base should attain a certain minimum or maximum level and may not deviate too much across crew bases.

1.1.2 Crew Rostering

In crew rostering, all scheduled duties are assigned to crew members through *rosters*. A roster is a plan specifying the work to be performed by a crew member. For each day of the planning period, the roster contains either a rest day or a scheduled duty. Each roster must satisfy a complicated set of rostering rules, the most important of which we list here. First, the roster must contain a sufficiently long rest time between duties on consecutive days. In practice, this rest time should be at least 12 hours. Second, the roster should contain a minimum number of days off and rest periods. Notably, the roster should contain sufficiently many 'red weekends', in which the crew member has the entire weekend off. Third, the roster must satisfy several rules relating to the workload. There is a constraint on the maximum workload of a single week, as well as on the average weekly workload. Figure 1.3 shows a roster for a four-week planning period.

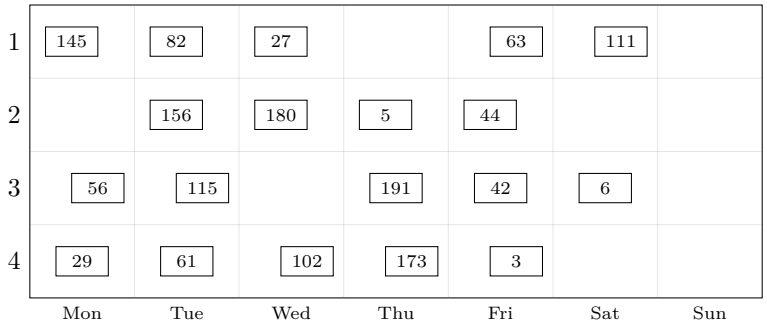


Figure 1.3: Example of a four-week roster. Numbered blocks indicate duties, the absence of blocks indicates a rest day.

We distinguish between *cyclic* and *acyclic* rosters. In a cyclic roster, the same duties are scheduled every week, performed by a different crew member in the roster group every time. Rotating through the rows (weeks) of the roster, the crew members in a roster group thus perform exactly the same work over the course of the planning period. In contrast, acyclic rosters can be used when the set of duties differs from week to week. Acyclic rosters are not assigned to a group, but specify the scheduled duties for an individual crew member over the course of the planning period. In Part I of this thesis, we study a crew planning process that can accommodate both cyclic and acyclic rosters.

Crew rostering features multiple goals. Under the assumption that sufficient crew members are available to cover all duties, the rosters have little impact on the crew costs. Instead, the main objective is to find a feasible crew plan. Attention is paid, however, to the attractiveness of a roster. This is measured in terms of, among other things, the frequency and length of rest periods, and the degree to which preferred roster patterns are adhered to. For example, it is preferred to schedule two early duties in a row, rather than an early duty after a night duty. Fairness also comes into play in the rostering phase. While fairness between crew bases is guaranteed by the Sharing-Sweet-and-Sour rules, and fairness within the roster group is ensured by the use of cyclic rosters, some care must be taken in ensuring that duties are fairly divided over the different roster groups within a crew base. We refer to Ernst et al. (2004) for a survey on crew rostering, and to Breugem (2020) for a recent dissertation on crew rostering at NS.

1.1.3 Crew Planning at Netherlands Railways

The motivation for Part I of this thesis is the design of a new crew planning process for Netherlands Railways. In this section, we first give a brief overview of the current planning process and outline some of its main drawbacks. Next, we sketch the proposed planning process and describe how it aims to eliminate the downsides of the current process.

Figure 1.4 illustrates the current planning process at NS. This consists of a tactical phase, several months before the day of operations, and an operational phase, spanning the several weeks before up to the day of operations. It starts in the tactical phase with the construction of a set of generic duties, which are assumed to repeat every week of the planning period. Sharing-Sweet-and-Sour rules are imposed on the generic duties, ensuring that workload is balanced over the various crew bases. Per crew base, the duties are assigned to roster groups and placed into cyclic rosters that are communicated to the crew. In the operational phase, timetable changes might cause the actual tasks to deviate from the generic tasks. To accommodate these changes, the duties are updated, and as a result, so are the rosters. We refer to Abbink et al. (2018) for a more detailed overview of crew planning at NS.

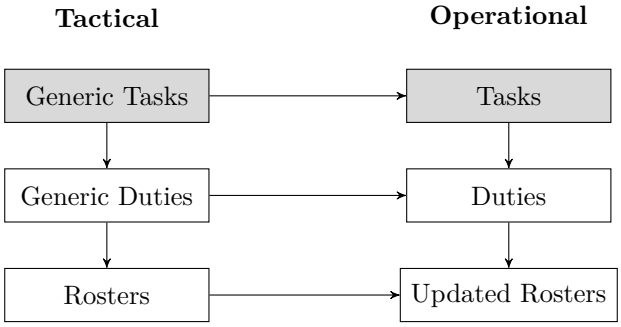


Figure 1.4: Current crew planning process. Crew planning decisions are indicated by white rectangles, inputs by shaded rectangles.

The process outlined above suffers from several drawbacks. First, many costly planning steps are repeated, since duties and rosters are generated in both the tactical and operational phase. Second, the crew members cannot reliably schedule their personal lives around the communicated rosters, as these are subject to frequent roster updates. Simultaneously, the railway operator has limited flexibility for updating the schedule in the operational phase, as this would result in even more roster up-

dates. Finally, the Sharing-Sweet-and-Sour rules do not always achieve their intended aims. As they are imposed on the crew base level, they do not provide attractiveness guarantees to individual crew members. Moreover, as they are imposed on the generic duties only, they are not necessarily satisfied by the duties performed in the operational phase.

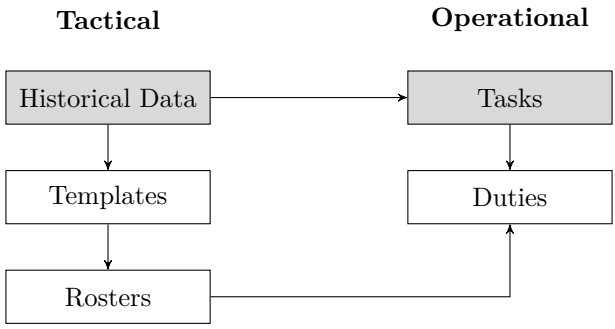


Figure 1.5: Proposed crew planning process. Crew planning decisions are indicated by white rectangles, inputs by shaded rectangles.

Netherlands Railways is considering to resolve these issues through a novel crew planning process, sketched in Figure 1.5. The process revolves around *templates*, duty placeholders that specify a location and time window during which a crew member can be called upon to work, but do not yet specify the exact work content. Templates typically span a time window that is slightly larger than the average duty length, and hence a single template can be used to perform one of many different duties. In the tactical phase of this process, NS selects a set of templates that provide sufficient capacity to cover all tasks in the operational phase. Since the exact tasks to perform are not yet known, this is done based on historical planning data. The templates are communnicated to crew members in the form of template-based rosters, which need not necessarily be cyclic anymore. In the operational phase, once the exact tasks to be performed are known, duties are constructed that cover these tasks and do not exceed the capacity provided by the templates. The Sharing-Sweet-and-Sour rules are incorporated in this phase too. Notably, they are applied at the individual level, and relate to the duties performed over the entire planning period.

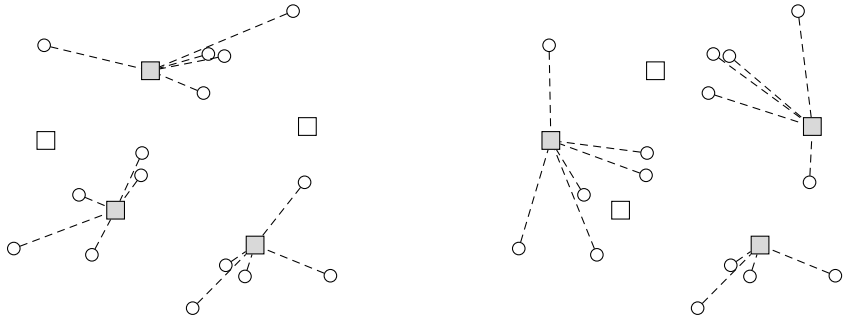
The proposed process does not suffer from the major issues encountered in the current planning process. First, fewer planning steps are required, since duties and rosters are only constructed once. Second, all roster updates are eliminated, providing the crew members with more certainty regarding their work schedules. Third, the time

windows specified by the templates provide the operator with additional flexibility in scheduling duties in the operational phase. Fourth, the newly enforced Sharing-Sweet-and-Sour rules offer stronger and more reliable guarantees to individual crew members. The new process gives rise to new crew planning problems, which we study in Part I of this thesis.

1.2 Fairness-Oriented Optimisation

Classical optimisation problems feature a central decision-maker who is tasked with selecting an 'optimal' solution from a set of feasible solutions. Optimality is typically measured in terms of efficiency, e.g., costs. In many cases, however, the problem involves a set of agents or stakeholders that derive some payoff or (dis)utility from the solution. In particular, each solution might benefit each agent differently, introducing a potential source of inequality.

In recent years, operations researchers have started to pay more attention to a fair division of the payoffs of solutions among the agents involved. This fairness is typically quantified through a *fairness function*, indicating the degree to which payoffs differ across the various stakeholders. A simple example includes the *range*, defined as the difference between the smallest and largest payoff. A growing body of literature studies how fairness objectives can be optimised and balanced against classical, efficiency objectives. This trade-off is commonly referred to as the *price of fairness* (Bertsimas et al., 2011).



(a) Average and maximum response time of 5.6 and 12.1 minutes, respectively.

(b) Average and maximum response time of 7.0 and 9.6 minutes, respectively.

Figure 1.6: Two solutions to the ambulance location problem.

As a motivating example, consider the ambulance allocation problem illustrated in Figure 1.6, loosely based on Lodi et al. (2024a). The goal of this problem is to open a set of ambulance depots from which potential patients in the region can be quickly serviced. The potential depots are indicated with squares, while the inhabitants that might request care are indicated with circles. A fixed budget allows us to open three out of five depots in total. The response time of a patient is proportional to the distance between the patient and the nearest depot. Our goal is to select a set of locations that minimise the average response time, while ensuring that each potential patient can be serviced within 15 minutes. Figures 1.6(a) and 1.6(b) present two possible solutions to this problem. The solution of Figure 1.6(a) minimises the average response time, but might be considered unfair as there are patients whose response time reaches 12.1 minutes. Figure 1.6(b) presents an alternative solution that is perhaps less efficient with respect to the mean response time, but could be deemed more fair as the maximum response time is reduced to only 9.6 minutes. This goes to show that decision-makers have a wide range of solutions to choose from and must carefully balance efficiency and fairness.

There are different ways of including fairness in optimisation models. First, the literature distinguishes between *group fairness* and *individual fairness*. In the first case, one imposes that different groups of stakeholders are treated similarly. The current Sharing-Sweet-and-Sour rules at NS form a nice example, as they prohibit large differences between various crew bases. In the second case, fairness is measured at the individual level. This is exactly the goal of the individual Sharing-Sweet-and-Sour rules in the proposed planning process that we study in Part I of the thesis. Second, one can consider fairness in single-period problems or in repeated problems. It is typically quite restrictive, or even impossible, to impose fair solutions in one-shot optimisation problems. This is where the concept of *fairness over time* comes into play, which relates to fair aggregate outcomes of multi-period optimisation problems (Lodi et al., 2024a). Part II of this thesis considers individual fairness in both single-period and online, multi-period problems. We refer to Karsu and Morton (2015) for a survey on fairness in OR models.

Finally, multiple ways of quantifying the fairness of a solution exist, some of which are easier to interpret or model than others. Well-known examples include the range, Gini deviation, and max-min metric. The range, as discussed earlier, measures the difference between the smallest and largest payoff, whereas the Gini deviation is defined as the sum of absolute pairwise differences between all agents. Max-min

fairness is concerned with maximising the payoff of the agent that is 'worst-off'. This strongly relates to the concept of Rawlsian justice, stating that inequalities are permitted as long as they benefit the most disadvantaged (Rawls, 1991). Part II of the thesis focuses on the range and Gini deviation. We refer to Chen and Hooker (2023) for an introduction to formulating fairness objectives in optimisation models.

1.3 Thesis Overview

The thesis consists of two parts containing three and two chapters, respectively. Figure 1.7 provides a schematic overview of the chapters in this thesis, classifying the five chapters along four recurrent topics of this thesis. The three chapters of Part I all study railway crew planning, while Chapters 3 and 5 both consider problems featuring fairness over time. Column generation, a successful solution method for large-scale optimisation problems, is applied in all chapters, but only Chapters 4 and 6 present generic methodological contributions to column generation and branch and price. Finally, the two chapters of Part II consider range minimisation, either implicitly, as in Chapter 5, or explicitly, as in Chapter 6.

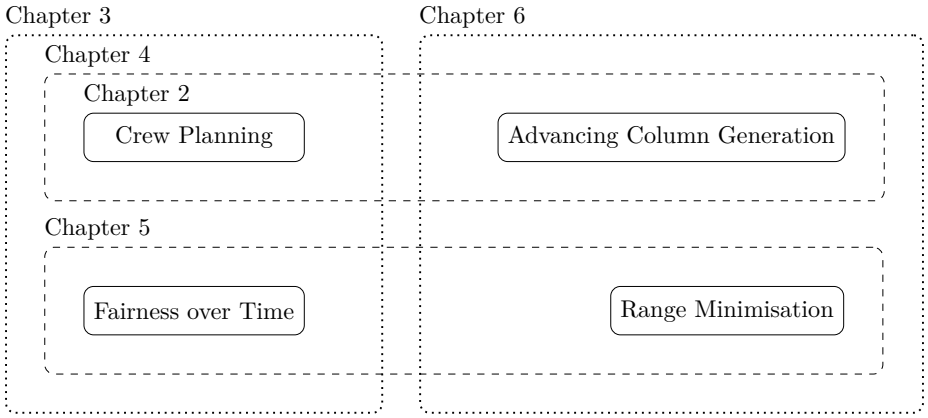


Figure 1.7: Schematic overview of the chapters in this thesis.

Part I focuses on models and methods for railway crew planning. Chapters 2 and 3 both study novel problems arising in the proposed crew planning process of Netherlands Railways. First, Chapter 2 considers a tactical crew scheduling problem, and presents a new Benders decomposition approach to determine a robust set of templates. Chapter 3 examines fair operational crew scheduling in the setting where template-based rosters are given. It proposes a dedicated algorithm to construct

individual crew schedules that are fair over time. Chapter 4 presents a new efficient exact pricing algorithm to be used in column generation algorithms for basic railway crew scheduling problems.

Part II studies generic methods for fairness-oriented optimisation. Chapter 5 centers on fairness over time in a setting where work must be assigned online to homogeneous workers. It proposes an intuitive assignment policy, and provides some theoretical justifications for using this particular policy. Chapter 6 considers the use of branch and price for minimising the range and other order-based objective functions. It proposes a generic branching rule that enables the use of classical, efficient branch-and-price methods for this type of problem.

The two parts of this thesis can be read separately from each other, and all chapters can be read in a stand-alone manner. We recommend to read Chapters 2 and 3 in their natural order, following the chronological order of the planning process. Chapter 4 can be read independently from the other two chapters of Part I. Similarly, we recommend to read Chapters 5 and 6 in the order of their presentation, since the computational results in Chapter 5 motivate the work in Chapter 6.

The remainder of this section provides a brief summary of the various chapters, each of which is a modification of an article that has been published in or submitted to academic journals or refereed conference proceedings. The thesis author is the main author of all five research chapters, responsible for, among other things, conceptualisation, algorithm design and implementation, experimental design and analysis, and drafting the articles. Chapters 2 and 3 are written under supervision of Twan Dollevoet and Dennis Huisman, who contributed to the conceptualisation and thoroughly revised multiple draft versions of each chapter. Chapter 4 is single-authored. Chapters 5 and 6 are joint work with Rui Chen and Andrea Lodi, largely conducted during a research visit to Cornell Tech in New York. Both co-authors greatly contributed to the conceptualisation and revision phase of the chapters. In addition, Rui Chen had a significant share in developing the mathematical framework and proofs that underlie Chapters 5 and 6. The PhD position of the thesis author has been partly funded by Netherlands Railways.

Chapter 2: “Benders Decomposition for Robust Tactical Railway Crew Scheduling”. Under review.

We consider robust tactical crew scheduling for a large passenger railway operator,

who aims to inform crew early on about their work schedules while also maintaining the ability to respond to changes in the daily timetables. To resolve this conflict, the operator considers a template-based planning process, templates being time windows during which duties can later be scheduled. The goal is to select a cost-efficient set of templates that is robust with respect to uncertainty in the work to be performed in the operational phase. A set of templates is deemed robust when few excess duties are required to cover all work in the operational planning phase. To enable the construction of efficient template-based rosters, we impose several template rostering constraints that proxy the actual rostering rules of later planning steps. We propose a two-phase accelerated Benders decomposition algorithm that can incorporate these restrictions. Computational experiments on real-life instances from Netherlands Railways, featuring up to 948 tasks per day, show that historical planning information can be used to obtain robust templates and that sparse solutions can be obtained at negligible extra costs. Compared to a literature benchmark, our Benders decomposition method solves three times as many instances without rostering constraints to optimality.

Chapter 3: "Railway Crew Planning with Fairness over Time". Published in *European Journal of Operational Research* as Van Rossum et al. (2024b). This paper received the first prize in the RAS Student Paper Award at INFORMS 2022.

Passenger railway operators typically employ large numbers of drivers and guards, and are interested in providing them with fair and attractive working conditions. At Netherlands Railways (NS), the largest passenger railway operator in The Netherlands, this challenge is addressed through the use of Sharing-Sweet-and-Sour rules, which specify a fair allocation of sweet (attractive) and sour (unattractive) work over the different crew bases. While these rules are currently implemented at the crew base level and in the tactical planning phase, NS considers formulating these rules at the individual level, in the operational planning phase, and with respect to a given planning period. This gives rise to a new problem, which we call the railway crew planning problem with fairness over time. We propose a rolling horizon approach with a penalty-based feedback mechanism and a column generation heuristic to solve this problem. On several real-life instances from NS, including up to 572 unique guards, this method is able to satisfy the individual rules for on average 95.2% of the employees.

Chapter 4: "A Fast Exact Pricing Algorithm for the Railway Crew Scheduling Problem". Published in *Operations Research Letters* as Van Rossum (2022a).

The railway crew scheduling problem consists of selecting a least cost set of duties that cover all tasks. Large-scale crew scheduling problems are typically solved with column generation, where in each iteration a pricing problem needs to be solved. When the duty constraints consist in a maximum duty length, a meal break constraint, and the requirement to start and end at the same crew depot, the fastest known exact pricing algorithm has $\mathcal{O}(|N|^3)$ complexity, $|N|$ being the number of tasks. In this chapter, we propose an $\mathcal{O}(|N|^2)$ exact pricing algorithm. We compare the two algorithms on randomly generated instances and show that a reduction in computation time of 95% is attained on instances of size $|N| = 1,250$.

Chapter 5: "Optimising Fairness over Time with Homogeneous Workers". Published in the *ATMOS* conference proceedings as Van Rossum et al. (2023).

There is growing interest in including fairness in optimization models. In particular, the concept of fairness over time, or, long-term fairness, is gaining attention. In this chapter, we focus on fairness over time in online optimization problems involving the assignment of work to multiple homogeneous workers. This encompasses many real-life problems, including variants of the vehicle routing problem and the crew scheduling problem. The online assignment problem with fairness over time is formally defined. We propose a simple and interpretable assignment policy with some desirable properties. In addition, we perform a case study on the capacitated vehicle routing problem. Empirically, we show that the most cost-efficient solution usually results in unfair assignments while much more fair solutions can be attained with minor efficiency loss using our policy.

Chapter 6: "Efficient Branching Rules for Optimising Range and Order-Based Objective Functions". Revision at *Mathematical Programming*. This chapter builds on a conference paper, published in the *IPCO* conference proceedings as Van Rossum et al. (2024a).

We consider range minimisation problems featuring exponentially many variables, as frequently arising in fairness-oriented or bi-objective optimization. While branch and price is successful at solving cost-oriented problems with many variables, the performance of classical branch-and-price algorithms for range minimisation is drastically impaired by weak linear programming relaxations. We propose range branching,

a generic branching rule that directly tackles this issue and can be used on top of problem-specific branching schemes. We show several desirable properties of range branching and show its effectiveness on a series of instances of the fair capacitated vehicle routing problem and fair generalised assignment problem. Range branching significantly improves multiple classical branching schemes in terms of computing time, optimality gap, and size of the branch-and-bound tree, allowing us to solve many more large instances than classical methods. Moreover, we show how range branching can be successfully generalised to order-based objective functions, such as the Gini deviation.

1.4 Contributions

The contributions of this thesis are fourfold. First, we introduce new problems in the field of railway crew planning and fairness-oriented optimisation. In Chapter 2, we extend existing robust tactical crew scheduling models with a form of rostering constraints. We are the first to consider multi-period individual fairness in operational crew scheduling, in Chapter 3. We formalise the online allocation problem with fairness over time in Chapter 5. Finally, in Chapter 6, we introduce a general class of range minimisation problems.

Second, we propose efficient heuristic and exact solution methods for the problems we consider, often employing advanced mathematical programming techniques. In Chapter 2, we introduce a two-stage accelerated Benders decomposition algorithm for robust tactical crew scheduling, and in Chapter 3, we develop a column generation heuristic with a penalty-based feedback mechanism for the crew planning problem with fairness over time. Both these methods are heuristic in nature and make use of several acceleration techniques, in order to handle the large size of the real-life instances from Netherlands Railways. In Chapter 4, we describe a pricing algorithm that can be used in column generation algorithms for basic railway crew scheduling problems. We introduce a novel, generic branching rule, tailored for range minimisation problems, in Chapter 6.

Third, we derive theoretical results on some of the problems and methods presented in this thesis. In Chapter 4, we provide a complexity analysis of the proposed pricing algorithm. We study the computational complexity of the online allocation problem and derive performance guarantees on the allocation policy in Chapter 5. In Chapter 6, we introduce a general class of formulations for range minimisation prob-

lems and show how, independent of the formulation at hand, our range branching rule drives up the dual bound.

Fourth, we evaluate the performance of our methods on real-life instances, randomly generated instances, and benchmark instances from the literature. In Chapters 2 and 3, we test our algorithms on historical crew planning data from Netherlands Railways and derive practical insights regarding the proposed planning process. We test the pricing algorithm on randomly generated crew scheduling instances in Chapter 4. In Chapter 5, we evaluate the online allocation policy on benchmark instances of the capacitated vehicle routing problem. The same instances, as well as benchmark instances of the generalised assignment problem, are used in Chapter 6.

Part I

Railway Crew Planning

Chapter 2

Benders Decomposition for Robust Tactical Railway Crew Scheduling

2.1 Introduction

Passenger railway operators engaging in crew planning must balance two conflicting goals. On the one hand, they wish to provide certainty to crew members about their working schedules early on. On the other hand, they like to maintain the flexibility to modify their planning in response to unforeseen changes. In particular, crew members wish to be informed about their rosters already in the tactical planning phase, months before the day of operations, in order to schedule their personal lives around their work. At the same time, the exact tasks, i.e., scheduled train trips, to be performed are not known until the operational phase, at most several weeks before the day of operations. While a yearly timetable and rolling stock schedule are generally available, they remain subject to changes resulting from, for example, scheduled maintenance works and events.

To handle these conflicting objectives, Netherlands Railways (NS), the largest passenger railway operator of the Netherlands, is considering a template-based planning process. Here, a template is a time window specifying the hours during which a crew member can be called upon to perform a duty, i.e., day of work. The proposed process starts with the construction of a template-based roster in the tactical planning phase. The template-based roster is communicated to crew several months before the day of operations and is considered fixed from that point onwards. The roster is detailed to a duty-based roster in the operational planning phase, once the timetable and rolling stock schedule for specific days are finalised. When the work to be performed is known, duties are generated that cover this work and can be performed by crew members within the time windows specified in their roster. Costly excess duties are available in case the template-based rosters fail to provide sufficient capacity.

The template-based planning process has several advantages. First, templates provide the operator a way of accounting for the uncertainty regarding the work to be performed. Since the length of template time windows typically exceeds the length of an average duty, a single template can accommodate one of multiple duties in different realisations of the daily timetable. Second, template-based rosters offer strong guarantees to crew members. This allows them to schedule their personal lives well in advance, knowing that they will not be called upon to work outside the hours specified in the templates. Finally, there is no need for many costly (re-)planning steps, since duties are generated only once, in the operational planning phase.

Motivated by NS, we study the viability of such a template-based process and consider

the tactical problem of choosing templates that are both cost-efficient and robust with regards to uncertainty in the work to be performed in the operational phase. To this end, we use the concept of recoverable robustness (Liebchen et al., 2009), and say that a set of templates is robust when few or no excess duties are required to cover all work in the operational planning phase. We assume that some historical planning information is available to inform the selection of robust solutions. To ensure that efficient template-based rosters can be constructed in subsequent planning steps, we also enforce several template rostering constraints that proxy the actual rostering rules. For example, we limit the number of unique template types, reserve templates, and early or late templates. While we focus on a large passenger railway operator, the setting where preliminary capacity plannings are constructed before detailed work assignments is encountered in many (crew) scheduling applications.

While there is a growing interest in robust railway planning, few authors study robust railway crew planning (Lusby et al., 2018). In addition, most authors focus on generating duties that are robust to daily disturbances, whereas our goal is to construct a capacity plan that is robust with respect to multiple timetable realisations. Our work is most closely related to that of Rählmann et al. (2021), who study robust tactical crew planning for a rail freight operator. Using a column generation heuristic, they choose robust templates based on a finite set of historical planning scenarios. We build on their work by imposing a wide range of template rostering constraints, necessitating a novel solution approach that is able to incorporate these restrictions. Moreover, our passenger railway application features a higher number of tasks per day and more tasks in a typical duty.

This chapter has four main contributions. First, we extend the problem of Rählmann et al. (2021) by imposing template rostering constraints. Second, to handle these restrictions, we propose an accelerated two-phase Benders decomposition algorithm. Third, we apply our method to real-life instances from NS featuring up to 948 tasks per day. We find that historical planning information can be used to obtain robust crew schedules and that sparse crew plans with a limited number of unique template types can be obtained at negligible extra costs. Fourth, we benchmark our approach against the column generation heuristic of Rählmann et al. (2021). The Benders decomposition algorithm solves three times as many large instances without rostering constraints to optimality, at the price of a two-fold increase in computing time.

The remainder of this chapter is structured as follows. We introduce the robust tactical crew scheduling problem in Section 2.2 and provide an overview of related

literature in Section 2.3. We motivate and present a mathematical formulation in Section 2.4, and propose an accelerated two-phase Benders decomposition algorithm in Section 2.5. In Section 2.6, we briefly present the benchmark algorithm of Rählmann et al. (2021). We conduct extensive computational experiments on instances from NS, discussing computational performance and practical insights in Sections 2.7 and 2.8, respectively. We conclude in Section 2.9.

2.2 Problem Description

The goal of crew planning is to assign all available work to crew members in an efficient manner while respecting all relevant labour rules. The work to be performed consists of tasks, i.e., scheduled train trips. In crew scheduling, tasks are aggregated to duties, i.e., feasible sequences of tasks constituting days of work. In crew rostering, duties are assigned to crew members in the form of rosters, detailing which duties are to be performed by which employee at each day of the planning period. Typically, crew scheduling and crew rostering are performed sequentially and in the tactical planning phase. Later, in the operational planning phase, duties and rosters are modified to account for changes in the timetable. The current crew planning process at NS follows this structure. The detailed planning steps in the tactical phase limit the flexibility of the operator in the operational phase, while the many roster updates provide little certainty to crew members. We refer to Abbink et al. (2018) for more information on the planning process at NS.

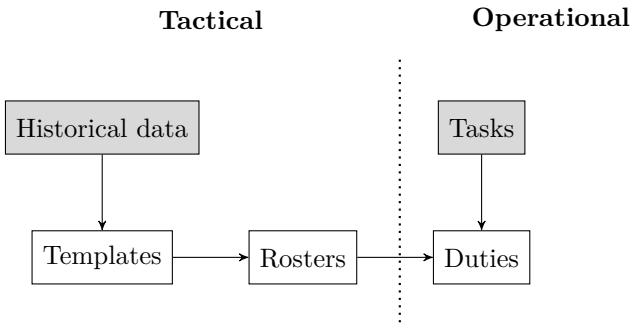


Figure 2.1: Proposed crew planning process. Crew planning decisions are indicated by white rectangles, inputs by shaded rectangles.

The proposed planning process, illustrated in Figure 2.1, effectively reverses the order of the two planning steps. A template-based crew rostering step takes place in

the tactical planning phase, and all crew scheduling is delegated to the operational planning phase. In particular, in the tactical planning phase we do not schedule duties but build a roster based on templates only. Here, a template is a combination of crew base and time window, representing a placeholder for a duty that is to be performed by a crew member of this base during the specified time window. In other words, a template specifies an interval during which a crew member will be called upon to work. The template-based roster is communicated to the crew before the start of the operational planning phase. The operator may not deviate from the working hours communicated here throughout the remainder of the planning period, enabling crew members to better schedule their personal lives. During the course of the operational phase, the daily timetables are finalised and the tasks to be performed on each day of the planning period are revealed. Duties must be generated that cover these tasks and can be performed with the capacity provided by the templates in the roster. If a duty cannot be performed by a crew member during their scheduled shift, as specified by the template, the duty becomes an excess duty that will be executed by an extra crew member at a high penalty cost.

We focus on the problem encountered in the tactical planning phase, in which several goals must be balanced. First, the railway operator desires an efficient crew plan with low labour costs, wishing to minimise the number of selected templates. Second, it must be possible to construct balanced rosters with the chosen templates, and hence some rostering compatibility constraints must be taken into account. Third, the templates must be chosen such that the number of excess duties required in the operational phase is minimal. In other words, the templates must be robust against many possible realisations of the tasks to be performed in the operational phase. We assume that historical planning information is available in the tactical planning phase, which can be leveraged to select robust templates. In short, the goal of the robust tactical crew scheduling problem is to select templates that satisfy roster compatibility constraints and minimise a weighted sum of template and excess duty costs.

We discuss the restrictions encountered in the tactical and operational planning phases in more detail in Sections 2.2.1 and 2.2.2, respectively.

2.2.1 Tactical Phase

In the tactical planning phase, we are provided with a pre-specified list of template types and must determine how often to use each template type. Each template acts

as a placeholder for duties, and is characterised by a crew base, day of the week, earliest start time, and latest end time. The list contains regular templates of a fixed length, as well as a reserve template spanning the entire day. Based on the chosen templates, each crew base will construct a preliminary roster that is communicated to crew members before the start of the operational phase. We focus on the problem of choosing templates, and consider the actual crew rostering step to be outside the scope of this chapter.

To ensure that efficient rosters can be constructed, the chosen templates at each crew base must satisfy several constraints. First, we impose some structure within the roster, increasing the opportunities for exchanging shifts, by limiting the number of unique template types that can be selected. Second, we limit the percentage of templates starting early (before 6:00h) or ending late (after 24:00h). The maximum percentage of early and late templates depends on the day of the week, as the percentage of night duties required is generally higher in the weekend. Third, the percentage of reserve templates may not exceed a certain upper bound. Finally, one could impose a maximum capacity constraint if the number of available crew members is known to be limited.

2.2.2 Operational Phase

Throughout the operational phase, the timetable and rolling stock schedule for each day of the planning period are modified to accommodate for, among other things, maintenance works and events. Once the schedules for a particular day are finalised, the set of tasks to be performed on that day is known. All tasks must be covered by duties, i.e., days of work. Each duty can either be assigned to a template or turned into an excess duty. When a duty is assigned to a template, it is performed by a crew member scheduled to work in a time window spanning the duty, and no additional cost is incurred. A duty that cannot be assigned to a template becomes an excess duty, to be performed by an excess crew member at a high penalty cost. The goal of the operational planning phase is to construct, for each day of the planning period, a set of duties that cover all tasks and minimise the sum of penalty costs and a variable cost depending on the total workload.

Each duty is a sequence of tasks that can be performed consecutively and must satisfy several duty rules. Two tasks can be performed consecutively when they end and start at the same station, respectively, and allow for sufficient transition time in case they take place on different rolling stock units. The duty rules are as follows. First,

each duty starts and ends at the same crew base. Second, the total length of a duty, defined as its ending minus starting time, may not exceed a maximum length that depends on the starting time of the duty. Third, each duty must include a meal break of at least 30 minutes at a station with a dedicated canteen. To position the meal break approximately halfway the duty, no duty may contain a period longer than 5.5 hours without a meal break.

2.3 Literature Review

This chapter belongs to a large body of literature on crew planning in public transport. Crew planning is generally decomposed into crew scheduling (or, crew pairing in airline applications) and crew rostering. We refer the interested reader to Heil et al. (2020) for a review on railway crew scheduling, and to Caprara et al. (1998) and Kohl and Karisch (2004) for introductions to railway and airline crew rostering, respectively. Abbink et al. (2018) provide an overview of crew planning at NS.

A considerable stream of research focuses on template-based scheduling approaches in tactical railway crew planning. Van den Hurk et al. (2024) propose optimisation models for template-based crew rostering at Netherlands Railways. Their methods can be used to construct rosters using the templates selected by our method. Borndörfer et al. (2017) integrate duty scheduling and rostering, coupling the two problems through duty templates. Similar to our work, Rählmann et al. (2021) consider the goal of selecting robust templates in tactical crew planning for a freight railway operator. Breugem et al. (2022a) study crew replanning in case of scheduled disruptions. The use of templates ensures that replanned rosters do not deviate much from the originally scheduled working hours. Finally, in Chapter 3 we consider the operational planning phase of the proposed planning process. In that phase, individual template-based rosters are fixed, and we aim to attain temporal individual fairness of the scheduled duties.

The large impact of disruptions on operational performance of railway systems has led to a large body of research on robustness in railway planning. We refer to Lusby et al. (2018) and Bešinović (2020) for recent reviews on robust planning. Within crew planning, robustness is less well-studied, with most authors focusing on constructing duties that are robust to daily disruptions. Veelenturf et al. (2016) consider real-time crew rescheduling in case of disruptions. They introduce the notion of 'quasi-robust' rescheduled duties: duties that are feasible in the most optimistic duration of the

disruption, but can be converted to feasibe duties in case the disruption lasts longer. Fuentes et al. (2019) argue that robustness can be incorporated by increasing the minimum transfer time between tasks, or by penalising deadheading movements and short transfers. Since we focus on the tactical planning phase where no duties are scheduled, robustness with regards to daily disruption is outside the scope of our work.

A larger stream of research considers robust airline crew pairing. Antunes et al. (2019) study airline crew pairing with the goal of generating pairings that are less likely to be disrupted by small delays. Muter et al. (2013) consider a setting where extra flights may be scheduled during operation, and aim to generate pairings that can accommodate extra flights with minimal disruptions. They propose a row-and-column-generation algorithm to solve the problem. Weide et al. (2010) provide an iterative solution approach for integrated airline crew pairing and aircraft routing, aiming to generate solutions that are robust against operational delays. Dunbar et al. (2014) further integrate re-timing decisions and aim to minimise the effect of delay propagations. Finally, Shebalov and Klabjan (2006) build robust crew pairings by maximising the number of so-called 'move-up crew', crew members whose tasks can be freely swapped in operation. They solve this problem using a combination of Lagrangian relaxation and column generation. Since we consider robustness against different timetable realisations, instead of robustness against operational delays, our work is most similar in spirit to that of Muter et al. (2013).

Our work is most closely related to that of Rählmann et al. (2021), who consider robust tactical crew planning for a railway freight operator. Similar to us, they aim to choose a set of templates that are able to fit efficient duties in multiple realised timetables. They propose a column generation approach with a two-layered pricing structure to solve the problem. Whereas we assume a pre-specified list of template types to choose from, they allow templates to start at arbitrary times of the day. In addition to their work, we impose template rostering constraints to incorporate some aspects of crew rostering, necessitating a novel solution approach.

2.4 Recoverable Robust Formulation

Two modelling choices underlie our mathematical formulation. First, we use the concept of recoverable robustness to model the interaction between the tactical phase and operational phase. Recoverable robustness, first introduced by Liebchen et al.

(2009), pertains to two-stage robust optimisation problems, where first-stage decisions might be rendered infeasible by a realisation of some uncertain data, but feasibility can be recovered at minor cost through second-stage decisions. This perfectly captures the relation between the tactical and operational phase: a set of templates is deemed recoverable robust when few excess duties are required to perform all tasks in the operational phase. The number of excess duties required in the second-stage can be determined by solving a (capacitated) crew scheduling problem. This approach suggests constructing a model that encapsulates both first-stage, tactical decisions, as well as second-stage, operational decisions.

Second, we capture the uncertainty regarding tasks to be performed in the operational phase using a finite scenario set. From one planning week to the next, tasks can be modified, added to, or removed from the schedule. Since detailed temporal and geographical information is needed to determine whether tasks can be placed in the same duty, such dynamics are hard to capture in a compact manner with traditional, polyhedral uncertainty sets. Instead, we use a scenario-based approach. The key assumption here is that a finite set of scenarios, each corresponding to a possible realisation of tasks in the operational phase, is representative of actual operations. In other words, we assume that if a tactical solution performs well in all the considered scenarios, it is likely to perform well in future operations. This approach is viable in practice, since railway operators typically have access to a large set of historical planning information that can be used to construct proper scenarios.

We finish with a brief discussion on how to construct scenario sets. While a large amount of historical planning information is typically available, attention must be paid to ensure that the chosen scenarios are both representative of future operations and contain sufficient variation to yield robust solutions. For example, historical data might be corrupted in case of replanning due to one-off events like scheduled maintenance works. In a scenario set, however, one rather aims to capture small-scale daily disturbances that are likely to be repeated, like additional shunting movements early in the morning or late in the evening. Finally, there is the question of how many distinct scenarios are required to obtain robust solutions. We will answer this question through extensive computational experiments in Section 2.8.

2.4.1 Mathematical Formulation

We now show how to formalise the modelling choices and provide a mathematical formulation of the robust tactical crew scheduling problem. We introduce the following

notation. Let P denote the set of templates and c_p the cost of template p. Denote the maximum number of unique template types by Γ, and the set of template rostering constraints by M. All constraints we consider, such as the maximum percentage of early templates, can be modelled in a linear fashion. For each constraint $m \in M$, E_m is a right-hand side constant, and coefficient e_p^m denotes the left-hand side contribution of template p. Let S denote the set of scenarios, and K_s the set of tasks to be covered in scenario s. Let Δ_p^s denote the set of duties that can be assigned to template p in scenario s, and let binary parameter $a_{pk}^{\delta s}$ indicate whether task $k \in K_s$ is covered by duty $\delta \in \Delta_p^s$. Finally, let c_E denote the excess duty cost.

Introduce binary and integer decision variables z_p and y_p to denote if and how often template p is used, respectively. Moreover, let continuous variable v_p^s denote the number of excess duties of template p used in scenario s, and let η denote the maximum recovery cost across all scenarios. Finally, let binary decision variable $x_p^{\delta s}$ denote if duty δ is assigned to template p in scenario s. The recoverable-robust formulation of the tactical crew scheduling problem then reads:

$$\min \quad \sum_{p \in P} c_p y_p + \eta \tag{2.1a}$$

$$\text{s.t.} \quad y_p \leq C z_p \qquad \forall p \in P \tag{2.1b}$$

$$\sum_{p \in P} z_p \leq \Gamma \tag{2.1c}$$

$$\sum_{p \in P} e_p^m y_p \leq E_m \qquad \forall m \in M \tag{2.1d}$$

$$\sum_{p \in P} c_E v_p^s \leq \eta \qquad \forall s \in S \tag{2.1e}$$

$$\sum_{\delta \in \Delta_p} x_p^{\delta s} \leq y_p + v_p^s \qquad \forall s \in S, p \in P \tag{2.1f}$$

$$\sum_{p \in P} \sum_{\delta \in \Delta_p^s} a_{pk}^{\delta s} x_p^{\delta s} \geq 1 \qquad \forall s \in S, k \in K_s \tag{2.1g}$$

$$\eta \geq 0 \tag{2.1h}$$

$$v_p^s \geq 0 \qquad \forall s \in S, p \in P \tag{2.1i}$$

$$x_p^{\delta s} \in \mathbb{B} \qquad \forall s \in S, p \in P, \delta \in \Delta_p \tag{2.1j}$$

$$y_p \in \mathbb{N} \qquad \forall p \in P \tag{2.1k}$$

$$z_p \in \mathbb{B} \qquad \forall p \in P. \tag{2.1l}$$

The objective (2.1a) is to minimise the sum of template costs and the maximum recovery cost. Constraints (2.1b)-(2.1c), where parameter C is an upper bound on the number of required templates, ensure that no more than Γ different template types are used. Constraints (2.1d) enforce the template rostering restrictions, and constraints (2.1e) correctly set the maximum recovery cost. Constraints (2.1f) ensure that duties of a template are covered by either available templates or excess duties, while constraints (2.1g) ensure that each task is covered by at least one duty. Constraints (2.1h)-(2.1l) model the variable domains.

Model (2.1) is challenging to solve for multiple reasons. First, it contains many integer and binary template variables, coupled to each other through weak big-M type constraints (2.1b). Second, the number of possible duty variables is already large for a single scenario, and this model contains multiple scenarios. Since our attempts at column generation-based heuristics have proven unsuccessful, we propose a Benders decomposition algorithm in Section 2.5 instead. Column generation is still used to solve all Benders subproblems.

2.4.2 Evaluation in Operational Phase

The solution quality of templates selected in the tactical planning phase is evaluated in the operational planning phase. On each day of the planning period, we aim to select duties that minimise the sum of excess duty penalties and variable workload costs. Let K be the set of tasks to be covered on a particular day and $\bar{y}_p$ the number of templates of type p selected in the tactical phase. The operational crew scheduling problem can then be formulated as follows:

$$\min \quad \sum_{p \in P} \sum_{\delta \in \Delta_p} l_p^\delta x_p^\delta + \sum_{p \in P} c_E v_p \tag{2.2a}$$

$$\text{s.t.} \quad \sum_{\delta \in \Delta_p} x_p^\delta \leq \bar{y}_p + v_p \qquad \forall p \in P \tag{2.2b}$$

$$\sum_{p \in P} \sum_{\delta \in \Delta_p} a_{pk}^{\delta s} x_p^\delta \geq 1 \qquad \forall k \in K \tag{2.2c}$$

$$v_p \geq 0 \qquad \forall p \in P \tag{2.2d}$$

$$x_p^\delta \in \mathbb{B} \qquad \forall p \in P, \delta \in \Delta_p. \tag{2.2e}$$

In essence, model (2.2) is identical to model (2.1) with a single scenario, containing the actual tasks to be performed, and fixed templates. We solve this using a

restricted master heuristic. First, the LP-relaxation of (2.2) is solved to optimality using the column generation algorithm presented in Section 2.5.3. The incumbent columns are then converted back to binary variables and the model is solved using the commercial mixed-integer programming solver CPLEX with a time limit of 15 minutes. Preliminary computational experiments show that this approach attains low optimality gaps.

2.5 Benders Decomposition

Benders decomposition is a successful variable partitioning approach for problems featuring *complicating variables*: variables whose fixing results in a problem that is significantly easier to solve. In short, the Benders algorithm decomposes the problem into a Benders master problem (BMP) and one or more Benders subproblems (BSPs), the BMP containing all complicating variables and the BSPs containing all remaining variables. The BMP is solved and the values of the complicating, first-stage variables are passed to the subproblems. If a subproblem is infeasible given these first-stage decisions, a feasibility cut is added to the BMP. If the estimated objective value of the BMP underestimates the true objective value, an optimality cut is added to the BMP. The process repeats until no more cuts are generated and the original problem is solved to optimality. We refer to Rahmaniani et al. (2017) for a recent overview on Benders decomposition.

Model (2.1) naturally lends itself to Benders decomposition, with template variables $\mathbf{y}$ and $\mathbf{z}$ acting as complicating variables: once template capacities have been decided, the problem decomposes into a series of independent crew scheduling problems, one per scenario. We keep the complicating variables in the BMP and construct a single BSP per scenario, containing only duty variables. In the BMP, we choose template capacities subject to all template restrictions, providing a lower bound (LB) on the optimal objective value. The BMP solution is passed to the BSP of each scenario, where it is determined how many excess duties are required given the template capacity chosen in the BMP. Since excess duties are available, this problem is always feasible and no feasibility cuts are generated. Moreover, the sum of BMP costs and excess duty costs provides an upper bound (UB) on the optimal objective value. If the number of excess duties exceeds the estimated recovery cost of the BMP, an optimality cut is added to the BMP. This process repeats until no more cuts are identified or the gap between LB and UB is sufficiently small.

As it is intractable to solve all subproblems to optimality with integral duty variables, we use a heuristic two-phase approach where we initially relax duty integrality in the first phase and heuristically repair integrality in the second phase. Effectively, we apply Benders decomposition on a partial continuous relaxation of model (2.1). As such, the Benders LB obtained in the first phase is still a valid lower bound on the optimal objective value of (2.1). The second phase returns a solution that is feasible in (2.1), providing a valid UB. Despite its heuristic nature, the two-phase solution approach thus allows us to compute valid optimality gaps.

The remainder of this section is structured as follows. We present the Benders master problem and Benders subproblems in Sections 2.5.1 and 2.5.2, respectively. In Section 2.5.3, we present the column generation algorithm used to solve all subproblems, and in Section 2.5.4, we present two acceleration techniques. In Section 2.5.5, we describe the two-phase approach in more detail, and we conclude with a complete overview of the Benders algorithm in Section 2.5.6.

2.5.1 Benders Master Problem

The Benders master problem is a subset of model (2.1) and only involves decisions and constraints regarding templates. In particular, the goal is to select templates that satisfy all template restrictions and minimise a weighted sum of template costs and estimated recovery costs. An estimate of the recovery cost is provided by optimality cuts returned by the subproblems, which couple the chosen templates to the estimated recovery cost. Denote by O_s the set of optimality cuts separated in scenario s. This set is empty in the first iteration of the algorithm. For each cut $o \in O_s$, the coefficient of template p is given by θ_p^{os}, and the constant term induced by task $k \in K_s$ is given by λ_k^{os}. The BMP can now be formulated as the following mixed-integer linear program:

$$\min \quad \sum_{p \in P} c_p y_p + \eta \tag{2.3a}$$

$$\text{s.t.} \quad y_p \leq C z_p \qquad \forall p \in P \tag{2.3b}$$

$$\sum_{p \in P} z_p \leq \Gamma \tag{2.3c}$$

$$\sum_{p \in P} e_p^m y_p \leq E_m \qquad \forall m \in M \tag{2.3d}$$

$$\sum_{p \in P} \theta_p^{os} y_p + \sum_{k \in K_s} \lambda_k^{os} \leq \eta \qquad \forall s \in S, o \in O_s \tag{2.3e}$$

$$\eta \geq 0 \tag{2.3f}$$

$$y_p \in \mathbb{N} \qquad \forall p \in P \tag{2.3g}$$

$$z_p \in \mathbb{B} \qquad \forall p \in P. \tag{2.3h}$$

The BMP is solved with a standard mixed-integer programming solver, after which the selected templates $\bar{\mathbf{y}}$ and recovery cost $\bar{\eta}$ are passed on to the BSPs.

2.5.2 Benders Subproblem

There is a single Benders subproblem per scenario, in which the templates $\bar{\mathbf{y}}$ chosen in the BMP are considered fixed. The goal is to determine the minimum number of excess duties needed, in addition to the capacity provided by the templates, to cover all tasks in the scenario. Following our two-phase approach, we relax the integrality of all duty variables in the BSP. Omitting scenario indices for convenience, the BSP in primal form reads:

$$\min \quad \sum_{p \in P} c_E v_p \tag{2.4a}$$

$$\text{s.t.} \quad \sum_{\delta \in \Delta_p} x_p^\delta \leq \bar{y}_p + v_p \qquad \forall p \in P \tag{2.4b}$$

$$\sum_{p \in P} \sum_{\delta \in \Delta_p} a_{pk}^\delta x_p^\delta \geq 1 \qquad \forall k \in K \tag{2.4c}$$

$$v_p \geq 0 \qquad \forall p \in P \tag{2.4d}$$

$$x_p^\delta \geq 0 \qquad \forall p \in P, \delta \in \Delta_p. \tag{2.4e}$$

The BSP can be readily solved using column generation (see Section 2.5.3). Due to the excess duties, the problem is always feasible and no feasibility cuts are generated. If the optimal objective value of (2.4) exceeds the estimated recovery cost $\bar{\eta}$, we separate an optimality cut. To determine the expression of these optimality cuts, we first write the BSP in dual form. Introducing dual vectors θ and λ for constraints (2.4b) and (2.4c), respectively, the dual BSP reads:

$$\max \quad \sum_{p \in P} \bar{y}_p \theta^p + \sum_{k \in K} \lambda_k \tag{2.5a}$$

$$\text{s.t.} \quad \theta_p + \sum_{k \in K} a_{pk}^\delta \lambda_k \leq 0 \qquad \forall p \in P, \delta \in \Delta_p \tag{2.5b}$$

$$-c_E \leq \theta_p \leq 0 \qquad \forall p \in P \tag{2.5c}$$

$$\lambda_k \geq 0 \qquad \forall k \in K. \tag{2.5d}$$

The Benders optimality cuts can thus be written as:

$$\sum_{p \in P} \theta_p y_p + \sum_{k \in K} \lambda_k \leq \eta. \tag{2.6}$$

Intuitively, these cuts indicate at which times of the day more template capacity is required to reduce the recovery cost. The cuts are only valid if the BSP is solved to optimality.

2.5.3 Column Generation

We solve all subproblems using column generation. Here, the subproblems are initialised with only a small subset of all possible duties, yielding a restricted master problem (RMP). After solving the RMP, a pricing problem is solved to identify columns with negative reduced cost that potentially improve the objective value. The algorithm repeats until no more columns with negative reduced cost exist and the subproblem is solved to optimality. The reduced cost of duty x_p^δ is given by

$$-\theta_p - \sum_{k \in K} \lambda_k a_{pk}^\delta. \tag{2.7}$$

The pricing problem of identifying a duty with negative reduced cost can be decomposed over all templates. The pricing problem for a single template can be modelled as a resource-constrained shortest path problem on a suitably-defined directed acyclic graph. This graph includes all tasks that can potentially be performed within the template, as well as all arcs modelling feasible connections between consecutive tasks. A source and sink node model the departure from and arrival at the crew base, respectively. The reduced cost can be easily decomposed on the arcs in this network. Resources model the maximum duty length and maximum time without meal break.

We start by solving all pricing problems with the heuristic labelling algorithm of Breugem et al. (2022a), augmented with a completion bound on the duty length and time without break. Once the heuristic pricing fails to return columns with negative reduced cost, we switch to the exact pricing algorithm of Huisman (2007) with $\mathcal{O}(|K|^3)$ time complexity. The subproblems must be solved to optimality in order for the Benders cuts to be valid.

We apply several acceleration strategies to speed up the column generation algorithm. First, we limit the number of columns returned by any pricing problem. Second, we select negative reduced cost columns to add to the RMP that are sufficently task-disjoint from previously selected columns, until no more such columns exist (Breugem et al., 2022a). Third, we apply column management and remove columns whose reduced cost exceeds a given threshold for a certain number of iterations. Fourth, we solve all pricing problems in parallel.

2.5.4 Acceleration Techniques

In this section, we propose two acceleration techniques to speed up convergence of the Benders decomposition algorithm: valid inequalities and the separation of Pareto-optimal cuts.

Valid Inequalities

It is well-known that including a relaxation of the subproblem in the master problem can speed up convergence of the Benders algorithm (Rahmaniani et al., 2017). To this end, we relax the template capacity constraints and several complicating duty rules (maximum length, meal break requirement, identical start and end depot requirement) in the crew scheduling subproblem. What remains is a min-cost flow problem, in which all tasks must be covered by flows starting from and ending at a depot. This problem can be solved in polynomial time, allowing us to efficiently compute strong lower bounds on the template capacity required at various times of the day.

Omitting scenario indices for simplicity, recall that K is the set of tasks to cover. As in the pricing problem (see Section 2.5.3), we construct a directed acyclic graph containing the tasks in K as well as an artificial source and sink node representing the depot. An arc in this graph models a feasible connection between two tasks. Let A be the set of arcs, and denote by $A^+(k)$ and $A^-(k)$ the set of outgoing and ingoing arcs of task $k \in K$, respectively. Moreover, let $A(t)$ denote the set of arcs whose traversal requires a crew member to work in the interval $[t, t+1]$. We introduce the continuous decision variable f_a to represent the flow over arc $a \in A$. A lower bound on the number of required duties in the interval $[t, t+1]$ can be obtained by solving

the following minimum cost network flow problem:

$$\min \quad \sum_{a \in A(t)} f_a \tag{2.8a}$$

$$\text{s.t.} \quad \sum_{a \in A^+(k)} f_a = \sum_{a \in A^-(k)} f_a \qquad \forall k \in K \tag{2.8b}$$

$$\sum_{a \in A^+(k)} f_a \geq 1 \qquad \forall k \in K \tag{2.8c}$$

$$f_a \geq 0 \qquad \forall a \in A. \tag{2.8d}$$

The objective (2.8a) is to minimise the number of arcs simultaneously used at time t. Constraints (2.8b) ensure flow balance, while constraints (2.8c) ensure that each task is covered at least once. The domain of flow variables is given by (2.8d).

Let V_t denote the optimal objective value of (2.8), and let $P(t)$ be the set of templates active during the interval $[t, t+1]$. We add to the BMP the valid inequality

$$\sum_{p \in P(t)} y_p + \frac{\eta}{c_E} \geq V_t. \tag{2.9}$$

We compute an inequality for each scenario and for values of t five minutes apart, balancing computing time and the goal of appropriately covering all times of day. Since the optimal objective value of model (2.8) is integral, it is not necessary to replace V_t by $\lceil V_t \rceil$.

Pareto-Optimal Cut Selection

The strength of the optimality cuts has a great impact on the convergence rate of the Benders decomposition algorithm (Rahmaniani et al., 2017). Since the crew scheduling problem is highly degenerate, however, multiple optimal dual solutions can exist which might not all be equally effective at approximating the true recovery cost function. Magnanti and Wong (1981) propose a method of choosing among optimal dual solutions by introducing the concept of Pareto-efficient cuts. In particular, they say an optimality cut is Pareto-efficient when, among all the solutions that are optimal at the incumbent BMP solution, it is also maximal with respect to some fixed 'core point' solution to the BMP. Such a cut can be obtained by solving an auxiliary subproblem. This approach has been successfully applied to crew planning by, among others, Mercier et al. (2005).

Let $\bar{\gamma}$ denote the objective of the regular BSP, and let $\mathbf{w}$ denote the core point vector. The auxiliary subproblem in dual form then reads:

$$\max \quad \sum_{p \in P} w_p \theta_p + \sum_{k \in K} \lambda_k \tag{2.10a}$$

$$\text{s.t.} \quad \theta_p + \sum_{k \in K} a_{pk}^{\delta s} \lambda_k \leq 0 \qquad \forall p \in P, \delta \in \Delta_p \tag{2.10b}$$

$$\sum_{p \in P} \bar{y}_p \theta_p + \sum_{k \in K} \lambda_k = \bar{\gamma} \tag{2.10c}$$

$$-c_E \leq \theta_p \leq 0 \qquad \forall p \in P \tag{2.10d}$$

$$\lambda_k \geq 0 \qquad \forall k \in K. \tag{2.10e}$$

The modified objective (2.10a) is to maximise the value of the cut at the core point vector. Constraint (2.10c) ensures that the cut remains optimal at the incumbent BMP solution. Together, this ensures Pareto-efficiency of the resulting cut.

In primal form, the changes to the dual problem result in a modification of the right-hand side of each template capacity constraint and an additional unrestricted variable u:

$$\min \quad \sum_{p \in P} c_E v_p + \bar{\gamma} u \tag{2.11a}$$

$$\text{s.t.} \quad \sum_{\delta \in \Delta_p} x_p^{\delta} + \bar{y}^p u \leq w_p + v_p \qquad \forall p \in P \tag{2.11b}$$

$$\sum_{p \in P} \sum_{\delta \in \Delta_p} a_{pk}^{\delta} x_p^{\delta} + u \geq 1 \qquad \forall k \in K \tag{2.11c}$$

$$v_p \geq 0 \qquad \forall p \in P \tag{2.11d}$$

$$x_p^{\delta} \geq 0 \qquad \forall p \in P, \delta \in \Delta_p \tag{2.11e}$$

$$u \in \mathbb{R}. \tag{2.11f}$$

We solve the primal auxiliary subproblem (2.11) with the same column generation algorithm as the regular BSP. As before, the optimal dual variables of this problem are used to construct optimality cuts. Following Mercier et al. (2005), we initialise the core point as a constant, positive vector. In each iteration, we update the core point as λ times the current BMP solution plus $(1 - \lambda)$ times the old core point. In line with Papadakos (2008), we set $\lambda = 1/2$.

2.5.5 Two-Phase Approach

It is challenging to solve all subproblems to optimality when the duty variables are required to be integral: this would require one to incorporate the Benders decomposition algorithm into a branch-and-bound framework. To avoid this issue, we adopt a heuristic two-phase Benders decomposition approach, similar to the works of Cordeau et al. (2001) and Mercier et al. (2005). In the first phase, we relax the integrality of all duty variables in the BSPs. The resulting subproblems can be readily solved to optimality using column generation. We apply Benders decomposition until some stopping criterion is met, obtaining a solution with potentially fractional duty variables in the subproblems. This solution is not necessarily feasible in the original problem but provides a lower bound on the optimal objective value. To repair integrality, we proceed to the second phase. Here, we iteratively fix duty variables to one in the BSPs and re-apply Benders decomposition to the resulting, restricted problem. We repeat this until all subproblem solutions are fully integral, thereby obtaining a feasible solution to the original problem and an upper bound on the optimal objective value. Mercier et al. (2005) precede the first step by an additional step where integrality of the complicating variables in the BMP is relaxed. Since solving our BMP is not very time-consuming, we choose not to do this here. While the two-phase approach is heuristic in nature, the bounds obtained in the first and second phase allow us to compute optimality gaps. As we will see, these gaps are typically rather small. We now describe each phase in more detail.

In the first phase we relax integrality of the duty variables in the Benders subproblems, allowing us to solve the BSPs to optimality using column generation. We apply regular Benders decomposition until no more optimality cuts are generated or a time limit of one hour is reached. The resulting first-phase solution is not necessarily a feasible solution to (2.1), since (i) it might contain fractional duty variables and (ii) some violated optimality cuts might not have been separated yet. Instead, the objective value of this solution is a lower bound on the true optimal objective value. This lower bound is generally tight, since the linear programming relaxation of the crew scheduling problem typically provides a good approximation of optimal integer solution values.

Once the first phase terminates, we enter the second phase in which integrality of the duty variables is repaired. The second phase alternates between fixing iterations and resolving iterations. In fixing iterations, we permanently fix the fractional duty variable with highest value in every Benders subproblem to one. After fixing, we add

a valid inequality to the BMP stating that the selected templates provide capacity for at least all fixed duties. In resolving iterations, several iterations of Benders decomposition are applied to allow the BMP solution to change in response to the fixing: it might happen that additional templates must be selected in response to duty variables being rounded to one. We repeat the sequence of fixing and resolving iterations until all duty variables are integral and the solution is feasible in model (2.1). The objective value of this solution provides an upper bound on the optimal objective value. The Benders lower bound obtained in the second phase is no longer valid, as fixing decisions are not necessarily optimal.

Algorithm 2.1 Two-phase accelerated Benders decomposition

```
Phase ← I
Add valid inequalities to BMP                                  ▷ Section 2.5.4
Initialise core point vector w
while Phase = I or solution not integral do
    Solve the BMP to obtain (ȳ, η̄)                             ▷ Section 2.5.1
    for Each scenario do
        Solve a BSP given ȳ                                    ▷ Sections 2.5.2-2.5.3
        if Objective exceeds η̄ then
            Solve auxiliary BSP given w                        ▷ Section 2.5.4
            Add optimality cut to BMP
        end if
    end for
    if Phase = I, and no cuts or time limit reached then
        Phase ← II
    end if
    if Phase = II, and no cuts or iteration limit reached then
        for Each scenario do
            Fix one or more duties in BSP                      ▷ Section 2.5.5
        end for
    end if
    w ← λȳ + (1 − λ)w
end while
```

2.5.6 Overview

Algorithm 2.1 provides a brief overview of the complete two-phase accelerated Benders decomposition algorithm. We decide to solve the BMP heuristically by imposing a time limit of ten seconds to the mixed-integer programming solver. Whenever no more cuts are generated and optimality is required, we iteratively increase the BMP time limit to obtain provably optimal solutions.

2.6 Benchmark

We benchmark our Benders decomposition algorithm against the column generation heuristic of Rählmann et al. (2021), who studied a highly similar robust tactical crew scheduling problem. Their problem setting differs from ours in two important ways. First, they do not assume a pre-specified list of template types, but allow templates to start at all possible times of the day. Second, they do not impose rostering restrictions on the set of chosen templates, i.e., they do not impose constraints (2.1b)-(2.1d).

We provide a brief overview of the benchmark algorithm in Section 2.6.1, and refer to Rählmann et al. (2021) for a detailed description. We describe several changes in implementation in Section 2.6.2, and compare the performance of the two algorithms on instances with pre-specified templates and no template restrictions in Section 2.7.5.

2.6.1 Overview

Whereas we assume that templates must be chosen from a pre-specified list, Rählmann et al. (2021) consider a slightly more flexible setting where templates are allowed to start at arbitrary times of the day. In particular, a template is defined as a compatible sequence of duties across multiple scenarios, duties being compatible when their start and end times comply with the maximum template length restriction. In other words, each template specifies a duty to be performed in each scenario. Since this leads to a huge number of possible templates, a column generation algorithm is proposed to solve the problem.

This algorithm alternates between two pricing problems. The first pricing problem consists of aggregating duties to feasible templates with negative reduced cost. To this end, a tailored pricing network is proposed. It contains a partition with a layer for each scenario, a node for each duty in a scenario, and an artificial source and sink node. Duties in different layers are connected when they can be performed in a single template. Each feasible template corresponds to a source-sink path in this network, and a labelling algorithm is used to identify negative reduced cost templates. Since the number of possible duties is large, the network is initialised with a small subset of duties only. In particular, an ordinary crew scheduling problem is solved for each scenario, and the solution to this problem is used as initial node set.

When the first pricing problem fails to return new templates, a second pricing prob-

lem is invoked to identify promising duties whose addition to the network might allow for the generation of new negative reduced cost templates. This pricing problem decomposes per scenario, and is highly similar in structure to the pricing problem described in Section 2.5.3. However, the reduced cost criterion accounts for the fact that duties must be aggregated to templates later on. Finally, a fixing rule is used to obtain integer solutions.

2.6.2 Adaptations in Implementation

We make several modelling choices to ensure that the solution space of the benchmark algorithm is the same as that of our Benders decomposition algorithm. In contrast with Rählmann et al. (2021), we assume a single crew base and single day of operations, and omit capacity constraints from consideration (though these can easily be incorporated in our approach). Since our problem setting features a prespecified list of allowed template starting times, we artificially modify the source arcs in the tailored network to align with these starting times. We use the same parameter settings as in the benchmark paper, and use the same pricing algorithms as for the Benders decomposition algorithm.

The number of possible duties in our passenger railway application is considerably larger than in freight railway, leading to a rapid growth of the tailored network throughout the course of the column generation algorithm. In response, we have made several minor changes in implementation to limit the size of the network. First, we only connect the source node to duties in the first scenario. Without loss of optimality, this reduces the number of source arcs and imposes that all templates use exactly one duty in each scenario. Second, we apply a task-disjointness filter (see Section 2.5.3) before adding duties from the second pricing problem to the network. Third, we remove duties based on their reduced cost as soon as the network contains more than a specified number of nodes. In our experiments, we use a maximum size of 5,000.

2.7 Computational Performance

In this section, we evaluate the computational performance of the proposed Benders algorithm by conducting extensive experiments on real-life instances from NS. We describe the instances and parameter settings in Sections 2.7.1 and 2.7.2, respectively. In Section 2.7.3, we analyse the effect of the various acceleration techniques

of Section 2.5.4 on the first phase performance, and in Section 2.7.4, we consider the overall performance of the two-phase algorithm. We compare our algorithm with that of Rählmann et al. (2021) in Section 2.7.5.

2.7.1 Instances

We use seven weeks of historical planning data of NS, spanning the period of October 4 to November 21 of 2021. For each day, the detailed data contains the set of tasks to be performed and the scheduled duties for all train guards in the Netherlands. The first three weeks are used to construct scenario sets, i.e., as input in the tactical planning phase, while the last four weeks are used for evaluation of the operational planning phase.

Table 2.1: Instance characteristics. For each crew base and day of week, the number of tasks and similarity ratio are given. Results are averaged over all weeks.

Base	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Ah	507	0.69	500	0.64	471	0.83	576	0.73	552	0.72	460	0.51	349	0.34
Amf	499	0.61	499	0.59	465	0.69	509	0.66	496	0.51	366	0.13	345	0.16
Asd	859	0.71	839	0.69	893	0.79	940	0.74	910	0.62	669	0.21	560	0.17
Ehv	593	0.60	612	0.58	576	0.68	641	0.63	587	0.55	502	0.36	418	0.31
Gvc	898	0.79	917	0.80	909	0.86	948	0.77	914	0.71	762	0.42	631	0.33
Nm	430	0.68	440	0.70	429	0.84	466	0.76	496	0.74	382	0.37	307	0.37
Rtd	615	0.74	598	0.79	552	0.81	655	0.81	633	0.70	502	0.41	386	0.25
Ut	843	0.73	814	0.73	801	0.77	838	0.76	891	0.65	744	0.35	581	0.25
Avg.	**656**	**0.69**	**652**	**0.69**	**637**	**0.79**	**697**	**0.73**	**685**	**0.65**	**549**	**0.35**	**447**	**0.27**

Using the historic assignment of tasks to crew bases through duties, we construct 7×8 instances, one for each combination of weekday and one of eight crew bases. Table 2.1 lists, for each instance, the average number of tasks and the similarity ratio. The similarity ratio, defined as the average fraction of tasks that appears in two consecutive planning weeks, is a rough proxy of the variability in task sets. The instances range in size: Crew bases Amersfoort (Amf) and Nijmegen (Nm) contain between 300 and 500 tasks per day, while crew bases The Hague (Gvc) and Amsterdam (Asd) can contain well over 900 tasks per day. Generally, more tasks are performed throughout the week than in the weekend. The task sets appear to be more stable throughout the week than in the weekend: the average similarity ratio on Wednesday equals 0.79, but drops to 0.27 on Sunday. This can be explained by the fact that most maintenance works and events are scheduled in the weekend, necessitating many changes to the timetable.

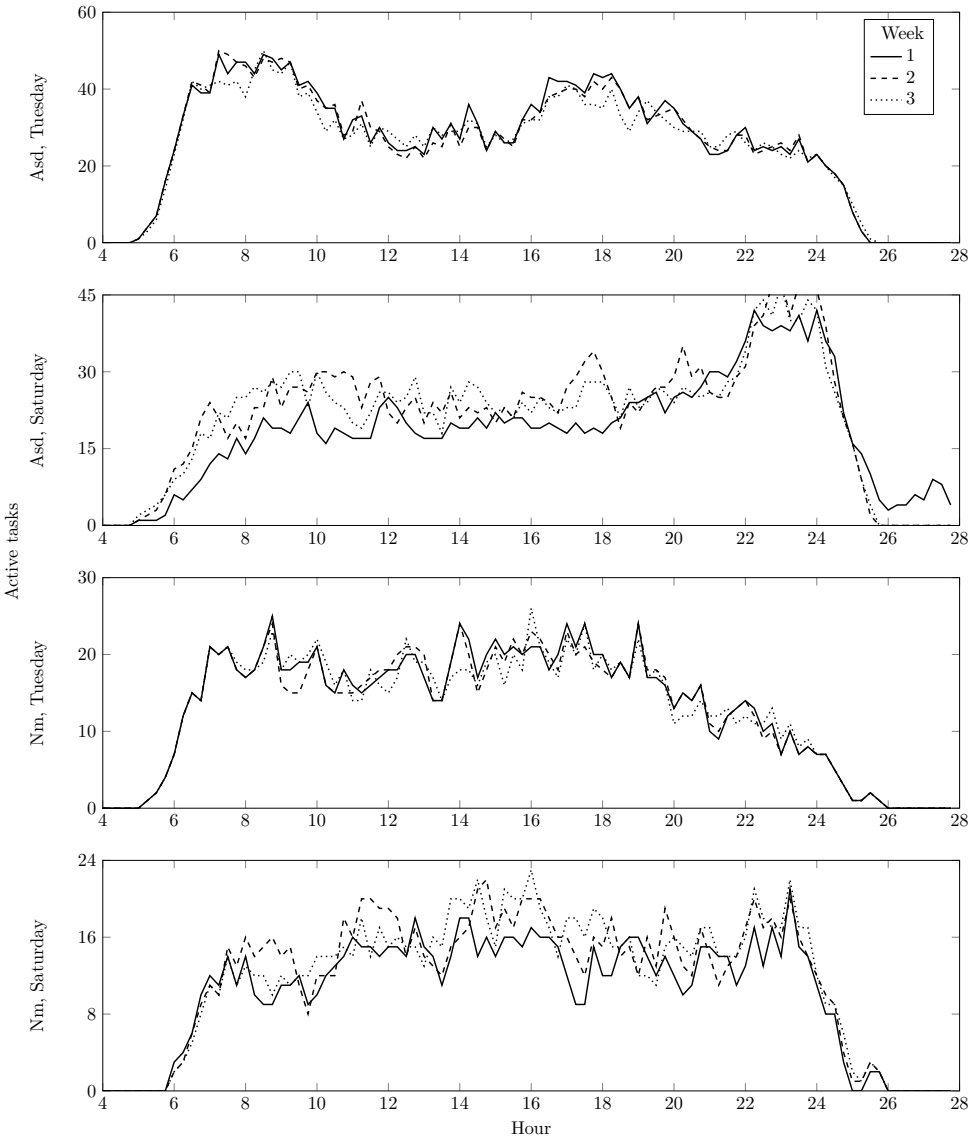


Figure 2.2: Temporal distribution of tasks for crew bases Amsterdam (Asd) and Nijmegen (Nm) on Tuesdays and Saturdays in the first three weeks of planning data.

To gain more insight into the scenario sets, Figure 2.2 presents the number of active tasks throughout the day for three scenarios of crew bases Amsterdam (Asd) and Nijmegen (Nm) on Tuesday and Sunday. The number of active tasks is computed as the number of tasks whose start and end time are before and after a given time, respectively. We observe a strong heterogeneity in capacity demand patterns across crew bases and days of the week. Nonetheless, the pattern of capacity demand is generally quite similar across weeks. This suggests that robust template capacities might exist, i.e., templates that are able to accurately provide capacity in all weeks of the planning period. The relatively low similarity ratios on Saturday (see Table 2.1) are reflected in Figure 2.2, showing more variety on the Saturdays than on the Tuesdays. As a result, providing robust schedules for the weekend is expected to be more challenging.

2.7.2 Parameter Settings

Unless indicated otherwise, we use the following parameter settings. All regular templates span 9.5 hours and are allowed to start every 30 minutes. There is a single reserve template that spans the whole day. The maximum duty length equals nine hours. We consider problems with three scenarios, in which at most 15 unique template types and 10% reserve templates may be selected. To reflect the different demand patterns, the maximum percentage of early and late templates depends on the day of the week: from Monday to Wednesday we allow at most 30% early and 30% late templates, on Thursday and Friday 30% and 45%, and on Saturday and Sunday at most 20% and 55%.

A single template costs 10,000, whereas an excess duty costs 40,000. In the evaluation phase, we use a variable cost of one unit per second, i.e., each worked hour contributes 3,600 towards the objective.

We use the following column generation parameters. At most 50 columns are returned per pricing problem, and only columns that are at least 50% task-disjoint from previously selected columns are added to the RMP. We remove columns whose reduced cost was positive in the last 15 iterations, and solve all pricing problems in parallel on eight threads. All experiments are conducted on a computing server with 16GB RAM and an AMD Rome 7H12 processor. CPLEX 22.1.0 is used to solve all (mixed-integer) linear programs.

2.7.3 Effect of Acceleration Techniques

To analyse the effect of acceleration techniques on the performance of the Benders decomposition algorithm, we conduct experiments on the Monday instances where we only consider the first phase of the algorithm and impose a time limit of three hours. We compare three versions of the Benders algorithm: a default version, a version with Pareto-efficient cuts, and a version with Pareto-efficient cuts and valid inequalities. Table 2.2 reports the results of these experiments. For each parameter setting and version of the algorithm, it reports the average number of iterations, computing time in seconds, split up over time spent on the BMP and time spent on the BSPs, the optimality gap in the final iteration, and the number of instances solved to optimality. Since we consider only the first phase of the algorithm, where duty variables are allowed to be fractional, optimality in this experiment does not necessarily imply global optimality in the original problem.

Table 2.2 shows that separating Pareto-efficient cuts has a drastic positive impact on the computational performance. Although the time per iteration goes up slightly, the algorithm requires only one fourth of the number of iterations, less than half of the computing time, and attains almost a percentage point reduction in final optimality gap. Adding valid inequalities leads to a reduction in average computing time of about seven percent. This effect is mostly driven by a sharp reduction in the number of iterations, as the time spent on the BMP per iteration expectedly goes up due to the extra constraints. The valid inequalities mainly lead to reduced computing times on instances that were already solved to optimality, as the final optimality gap and number of instances solved to optimality does not increase compared to the version with only Pareto-efficient cuts. We observe that non-optimality is often caused by a small optimality gap in the BMP that could not be closed by the solver within the time limit. Nonetheless, we observe consistent speed-ups across all instances.

To better understand the impact of the acceleration techniques, Figure 2.3 displays the development of the first phase Benders upper and lower bounds of the three versions of the algorithm. It shows the LB and UB as fraction of the best known LB, averaged over all instances and truncated to the first hour to highlight differences between the various approaches. It is clear that the performance improvement is purely driven by better UB behaviour. While the lower bound is often already at its optimal value after a single iteration in all approaches, the acceleration techniques lead to much faster convergence of the upper bound. The Pareto-efficient cuts have a particularly large impact, as they ensure that the recovery costs are better estimated

through the optimality cuts. The valid inequalities lead to a small further improvement, by guaranteeing that solutions are close to those that require few or no excess duties. The lower bound of the version with valid inequalities slightly lags behind the other two versions, as computing the valid inequalities takes a small amount of time.

Table 2.2: Effect of acceleration techniques on computational performance of first phase Benders decomposition for Monday instances.

				Time (s)				
Method	Types	Scenarios	Iterations	BMP	BSP	Total	Gap (%)	Optimal
Default	10	1	458.5	4,758.5	1,531.6	6,290.1	0.15	7/8
		2	351.9	3,388.4	1,862.4	5,250.8	1.34	6/8
		3	529.4	3,904.3	4,443.4	8,437.6	1.27	4/8
	15	1	596.0	2,787.8	2,292.6	5,080.4	0.15	7/8
		2	584.0	1,267.4	4,294.9	5,562.3	1.14	6/8
		3	593.3	1,373.5	5,282.1	6,655.6	1.07	5/8
	20	1	621.4	2,234.4	2,140.3	4,374.6	0.50	7/8
		2	703.0	1,306.5	4,774.0	6,080.5	0.22	7/8
		3	574.3	1,335.1	5,460.6	6,795.8	2.33	4/8
	Avg.		**556.8**	**2,484.0**	**3,564.7**	**6,048.6**	**0.91**	**53/72**
Pareto	10	1	192.8	2,851.9	1,309.9	4,161.8	0.15	7/8
		2	186.0	2,860.3	2,384.1	5,244.4	0.15	7/8
		3	130.6	1,885.0	1,761.1	3,646.1	0.08	7/8
	15	1	160.4	1,513.8	983.4	2,497.1	0	8/8
		2	126.4	1,012.9	1,555.0	2,567.9	0.02	7/8
		3	85.8	507.5	1,313.5	1,821.0	0	8/8
	20	1	188.3	1,290.5	969.6	2,260.1	0	8/8
		2	93,1	869.0	951.9	1,820.9	0	8/8
		3	88.5	619.4	1,472.4	2,091.8	0	8/8
	Avg.		**139.1**	**1,490.0**	**1,411.2**	**2,901.2**	**0.04**	**68/72**
Pareto + VI	10	1	183.1	3,424.3	1,090.3	4,514.5	0.20	6/8
		2	147.8	3,109.4	1,736.9	4,846.3	0.08	7/8
		3	115.5	2,296.4	1,811.4	4,107.8	0.08	7/8
	15	1	105.8	712.5	777.8	1,490.3	0	8/8
		2	105.0	963.8	1,379.5	2,343.3	0	8/8
		3	73.8	523.8	1.052.8	1,576.5	0	8/8
	20	1	135.9	1,268.6	840.6	2,109.3	0	8/8
		2	102.0	704.9	1,219.1	1,924.0	0	8/8
		3	70.6	427.9	986.1	1,414.0	0	8/8
	Avg.		**115.5**	**1,492.4**	**1,210.5**	**2,702.9**	**0.04**	**68/72**

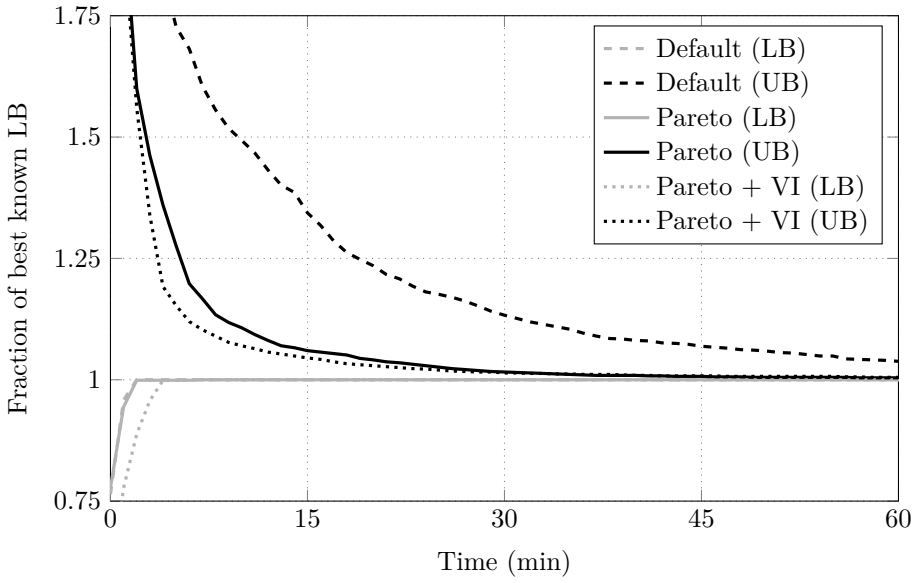


Figure 2.3: Effect of Pareto-optimal cuts and valid inequalities on progression of first phase lower and upper bounds.

2.7.4 Performance Two-Phase Approach

We now analyse the performance of the complete two-phase algorithm with both acceleration techniques. We impose a time limit of one hour for the first phase, and consider all instances in our sample. Table 2.3 reports the results. For all instances and each of the two phases, it reports the average number of iterations, computing time, and optimality gap. The first-phase gap is computed using the first-phase lower and upper bound, whereas the second-phase gap is computed using the first-phase lower bound and second-phase upper bound. We refer to Section 2.5.5 for more detail. In addition, we report the maximum second-phase optimality gap over all instances and number of instances solved to optimality.

We find that most computing time is spent on the first phase. The number of iterations and time per iteration is higher there, indicating that the variable fixing in the second phase restricts the solution space and reduces computing time. A large majority of instances is solved to optimality in the first phase, where a very low average optimality gap of 0.24% is attained. The optimality gap increases to an average 0.81% due to the heuristic fixing decisions in the second phase. Still, over 65% of instances is solved to global optimality, highlighting the success of our two-phase

Table 2.3: Computational performance of two-phase accelerated Benders decomposition.

		First phase			Second phase				
Types	Scenarios	Iterations	Time (s)	Gap (%)	Iterations	Time (s)	Gap (%)	Max gap (%)	Optimal
	1	143.5	1,966.4	0.28	19.3	165.5	0.82	8.33	38/56
10	2	113.5	1,933.9	0.43	34.2	233.5	1.78	43.75	31/56
	3	109.0	2,170.1	0.65	34.6	166.4	1.21	6.82	29/56
	1	131.5	1,283.9	0.14	23.7	139.5	0.36	3.24	43/56
15	2	104.1	1,357.6	0.08	35.3	151.8	0.37	4.00	43/56
	3	88.5	1,476.1	0.22	44.8	259.9	0.81	8.26	34/56
	1	131.5	1,325.6	0.06	22.3	97.7	0.39	3.08	43/56
20	2	95.2	1,290.5	0.14	34.3	123.1	0.58	5.26	40/56
	3	78.4	1,262.6	0.13	43.1	168.9	0.96	3.92	29/56
Avg.		**110.6**	**1,563.0**	**0.24**	**32.4**	**167.3**	**0.81**	**11.99**	**330/504**

approach. This is further confirmed by the fact that all second-phase optimality gaps are well below ten percent, except for a single outlier at 43.75% due to a sequence of ill-performing fixing decisions. Interestingly, the number of iterations decreases as the number of scenarios grows. This might arise from the fact that more cuts are separated per iteration, speeding up convergence of the Benders algorithm. At the same time, the computing time per iteration and optimality gap slightly increase as the number of scenarios grows. Nonetheless, average computing times below half an hour are more than reasonable for a tactical planning problem.

2.7.5 Comparison with Benchmark

We benchmark our Benders decomposition algorithm against the benchmark algorithm of Rählmann et al. (2021). To this extent, we alter our instances and remove all template rostering constraints. In particular, we impose no restriction on the maximum number of template types or maximum number of early and late templates. We forbid the use of reserve templates, and enforce that all templates start at the half hour in both methods. In case the benchmark algorithm outputs templates with length below 9.5 hours, we articially expand these before entering the operational planning phase.

Table 2.4 reports the computational performance of both methods. For each number of scenarios and method, it shows the average computing time, optimality gap, and number of instances solved to optimality. While the benchmark is on average significantly faster than our method, the Benders decomposition algorithm performs best in terms of quality for all sizes of the scenario set. The difference is especially large

Table 2.4: Results Benders decomposition and benchmark algorithm for the tactical problem.

	Benders			Benchmark		
Scenarios	Time (s)	Gap (%)	Optimal	Time (s)	Gap (%)	Optimal
1	1,252.7	0.39	49/56	48.9	0.72	30/56
2	1,282.2	0.55	42/56	252.7	1.04	25/56
3	1,185.4	0.62	34/56	582.8	1.62	11/56

for the case of interest with three scenarios, where our method attains an optimality gap that is one full percentage point lower and solves 23 more instances to optimality. The two algorithms display different scaling behaviour, both in terms of time and quality. While the Benders algoritm shows stable performance with respect to the number of scenarios, the running times and optimality gaps of the column generation algorithm grow rapidly as the scenario set grows.

Table 2.5: Results Benders decomposition and benchmark algorithm for the operational problem.

	Benders				Benchmark			
Scenarios	Templates	Duties	Excess	Avg. workload (h)	Templates	Duties	Excess	Avg. workload (h)
1	60.07	62.47	3.03	7.68	60.27	61.81	2.25	7.76
2	61.63	62.40	1.51	7.70	61.84	61.97	1.08	7.73
3	62.09	62.48	1.25	7.68	62.86	62.41	0.80	7.67

Table 2.5 shows the solution quality of both approaches as evaluated in the operational planning phase. For each combination of solution approach and number of scenarios, it lists the number of selected templates, number of duties and excess duties in the operational phase, and average workload in hours. All results are averaged over all instances and weeks of the evaluation period. We observe that the improved solution quality in the tactical planning phase is somewhat penalised in the operational planning phase. Since the Benders algorithm selects fewer templates in the first stage, it requires on average more duties and excess duties in the second stage. For three scenarios, however, the total number of crew members required, obtained by summing template capacity and excess duties, is approximately equal for both methods. As we will also see in Section 2.8, the results in Table 2.5 mainly show that a finite scenario set will never yield solutions that are fully robust, i.e., require no excess duties.

2.8 Practical Insights

In this section, we provide several practical insights by analysing the effect of various restrictions on solution quality. Throughout the following, we will evaluate the quality of chosen templates by analysing the number of templates chosen in the tactical phase, the number of regular and excess duties required in the operational phase, and the average workload of the operational duties. Unless indicated otherwise, all results are averaged over the eight crew bases, seven days of the week, and four weeks used for evaluation purposes. We use the same instances and parameters as in Section 2.7, and choose templates using the Benders decomposition algorithm with both acceleration techniques.

We analyse the effect of the number of scenarios and template types in Section 2.8.1, and present results on the number of reserves in Section 2.8.2. In Section 2.8.3, we present the effect of the template length on solution quality, and we split up results by day of the week in Section 2.8.4.

2.8.1 Number of Scenarios and Types

Table 2.6 displays the effect of the maximum number of template types and the number of scenarios on the solution quality. Increasing the number of scenarios generally leads to more robust solutions. With a larger scenario set, more templates are chosen to cover the tasks in all scenarios, leading to a reduction in the number of excess duties required in the operational phase. For example, for at most 15 unique template types, moving from one to three scenarios means that, on average, 1.51 additional templates are selected and 1.60 fewer excess duties are required. This is beneficial, since scheduling crew already in the tactical phase is preferred over calling upon additional crew last-minute in the operational phase. Even with three scenarios, however, a significant number of excess duties is still required. In other words, it is hard to be prepared for demand peaks in the operational phase that never occur in historical data. Moreover, fixing the capacity plan in the tactical phase comes at a sizeable efficiency loss. In the fully flexible case where all duties can be freely scheduled in the operational phase, only 60.44 duties are required on average. The efficiency loss results in (i) the need for excess duties and (ii) templates not assigned a duty but left empty. The number of templates left empty can be computed based on Table 2.6 by adding the number of excess duties to and subtracting the number of total duties from the number of templates. This value is quite stable, ranging around 0.5 for all settings.

Table 2.6: Effect of number of scenarios and maximum number of types.

Types	Scenarios	Templates	Duties	Excess	Avg. workload (h)
	1	60.45	62.32	2.24	7.71
10	2	61.82	62.40	1.13	7.70
	3	62.14	62.57	1.09	7.67
	1	60.18	62.14	2.50	7.72
15	2	61.29	61.96	1.15	7.74
	3	61.69	62.06	0.90	7.73
	1	60.16	62.07	2.39	7.72
20	2	61.41	62.08	1.17	7.73
	3	61.91	62.09	0.88	7.72

Table 2.6 also shows the cost of obtaining sparse solutions. Enforcing a maximum of ten unique template types, as compared to 15, appears to be somewhat restrictive, yielding solutions featuring more chosen templates and duties. Allowing for more than 15 template types does not appear to have a significant benefit. Solutions with up to 20 unique templates are similar in terms of duties and average workload, and actually require slightly more capacity. This results from the slightly higher optimality gap on these relatively unrestricted instances (see Section 2.7). All in all, a setting with three scenarios and up to 15 template types seems to be preferred, validating this as our choice of default setting in the remainder of the experiments.

To gain more insight into the effect of the scenario set on solution quality, Figure 2.4 shows the temporal distribution of capacity on the same four instances as Figure 2.2 for different numbers of scenarios. For each instance, number of scenarios, and time of the day, it shows the number of active scheduled template (positive part of y-axis) and the number of active excess duties (negative part of y-axis). The latter number is computed by aggregating over all four weeks of the evaluation period. Enlarging the scenario set leads to the scheduling of additional templates at very specific moments of the day, typically early in the morning or late in the evening. It also happens that redundant capacity is shifted away from other times of the day, as illustrated on Tuesday morning in Amsterdam or Tuesday evening in Nijmegen. All in all, adding scenarios leads to a strong reduction in the need for excess duties. Peaks in the number of required excess duties are strongly reduced or even eliminated completely.

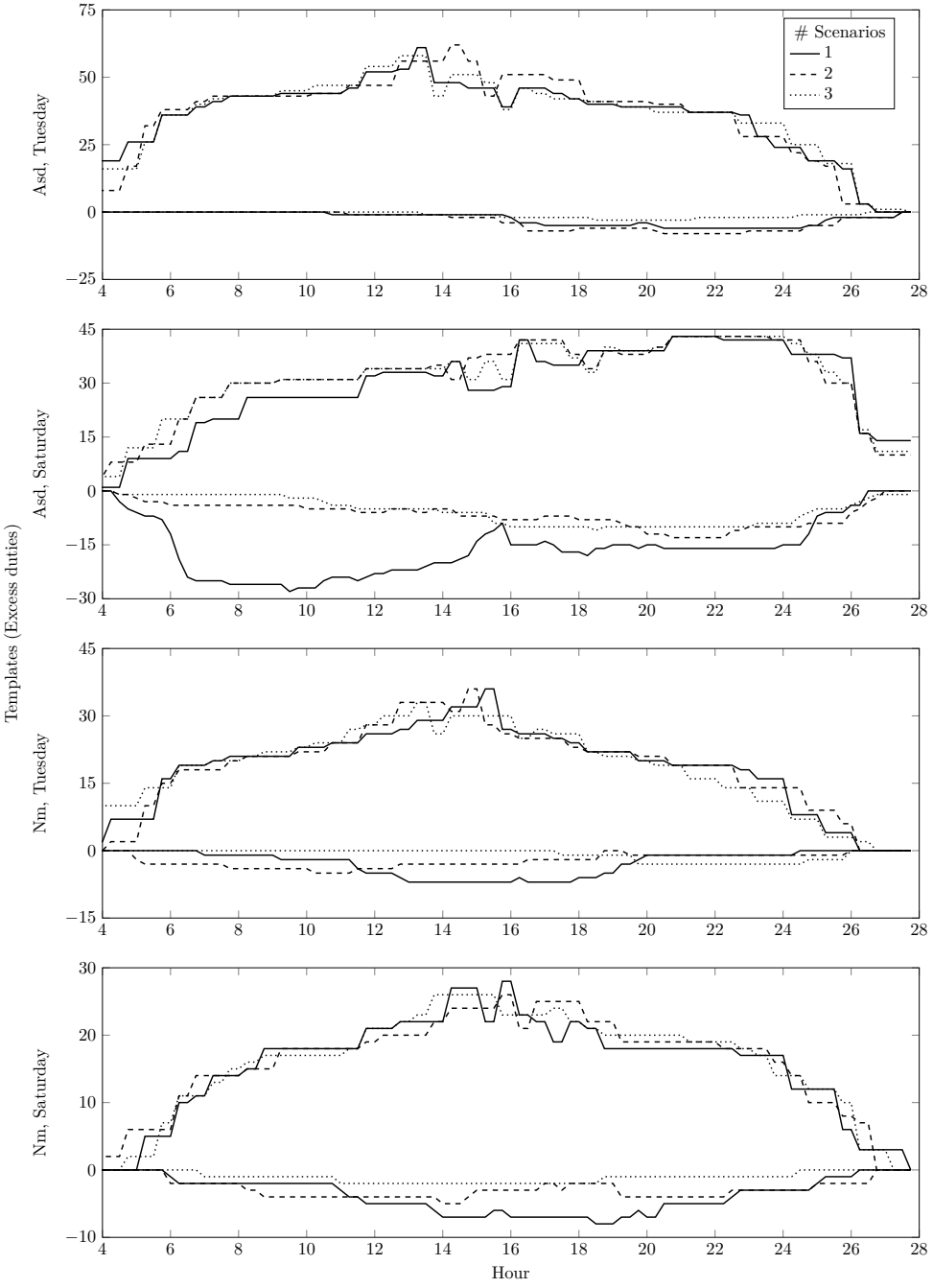


Figure 2.4: Temporal distribution of templates (positive part of y-axis) and excess duties (negative part of y-axis) for crew bases Amsterdam (Asd) and Nijmegen (Nm) on Tuesdays and Saturdays in the last four weeks of the planning period. At most 15 unique templates are allowed.

2.8.2 Reserves

Table 2.7 presents the effect of varying the percentage of reserve templates. Recall that reserve templates span the whole day, ideally suited for accommodating large demand shocks in the set of tasks to be performed. This is reflected in the results in Table 2.7, where we find that allowing for some reserves is crucial in attaining efficient schedules. Increasing the percentage of reserves from zero to ten percent leads to a major reduction in the number of required templates, regular duties, and excess duties. The average workload also increases, indicating that reserve templates are assigned efficient duties with a high workload. The marginal gain of adding reserves drops sharply, however, and there appears to be no sizeable benefit in employing more than ten percent reserves. A large fraction of reserve templates is also undesirable from a practical point of view, as it requires crew members to be on stand-by for the whole day. Finally, the results also allow us to estimate the effect of imposing template restrictions. By comparing the first row of Table 2.7 with the last row of Table 2.5, we find that on average more than one template fewer and fewer excess duties are required when omitting all template restrictions.

Table 2.7: Effect of reserve templates.

Reserves (%)	Templates	Duties	Excess	Avg. workload (h)
0	63.39	63.66	1.36	7.58
5	61.86	62.30	1.07	7.70
10	61.79	62.06	0.90	7.73
15	61.79	62.15	0.89	7.71
20	61.71	62.05	0.86	7.72

2.8.3 Template Length

Table 2.8 shows the effect of varying the template length, displaying results for template lengths ranging from eight hours and 45 minutes to ten hours in steps of 15 minutes. When the template length is reduced to nine hours or less, the number of templates and duties grows rapidly. The average workload drops sharply, as the short templates do not allow for efficient, long duties. A reversed, but weaker effect is observed when increasing the template length beyond nine and a half hours. The additional flexibility of longer templates allows for more efficient crew scheduling solutions. Less capacity is required in the tactical phase, and fewer duties are scheduled in the operational phase. Notably, the number of excess duties is not strongly affected by the template length, except in the case of a ten-hour template length.

The average workload strongly increases when the template length is increased from nine hours by half an hour or more, indicating that more efficient duties are scheduled and less total workload is required to cover all tasks. From the crew's point of view, however, shorter template lengths are strongly preferred, as they make it easier to schedule their personal lives around the reserved working hours. All in all, a template length of nine and a half hours seems to adequately balance solution quality and crew preferences.

Table 2.8: Effect of template length.

Length (h)	Templates	Duties	Excess	Avg. workload (h)
8:45	67.02	67.34	0.96	7.25
9:00	64.11	64.49	0.89	7.49
9:15	62.07	62.44	0.98	7.69
9:30	61.79	62.06	0.90	7.73
9:45	61.75	62.03	0.85	7.72
10:00	61.80	61.97	0.76	7.73

2.8.4 Day of Week

Table 2.9 illustrates the strong heterogeneity in solution quality across days of the week, reflecting the variety observed earlier in Table 2.1. While the lowest number of templates is scheduled in the weekend, the number of excess duties is relatively highest on Saturday and Sunday. This reflects the fact that maintenance works are typically scheduled over the weekend, as observed in the low similarity ratio in Table 2.1. In contrast, few excess duties are required from Tuesday to Thursday, days with a high similarity ratio where demand appears to be more stable and scenarios are more representative of future operations.

Table 2.9: Effect of day of week.

Day	Templates	Duties	Excess	Avg. workload (h)
Mon	63.13	63.66	0.84	7.74
Tue	63.50	63.88	0.47	7.66
Wed	62.50	62.84	0.47	7.82
Thu	70.13	70.34	0.38	7.73
Fri	70.13	70.69	1.28	7.68
Sat	57.88	57.84	1.84	7.70
Sun	45.25	45.19	1.03	7.79

2.9 Conclusion

Motivated by Netherlands Railways, we consider robust tactical railway crew scheduling for a large passenger railway operator. The goal is to select a cost-efficient set of templates in the tactical planning phase, that is robust with respect to uncertainty about the work to be performed in the operational planning phase. Building on the work of Rählmann et al. (2021), we impose several template rostering constraints, and propose a two-phase accelerated Benders decomposition algorithm that is able to incorporate these restrictions. Experiments on real-life instances of Netherlands Railways show the efficacy of our method and provide practical insight in how to choose robust templates. Historical planning information can be effectively leveraged to obtain robust templates, sparse solutions can be obtained at low extra costs, and a high number of excess duties can be avoided by using templates of reasonable length and a minimum number of reserve templates. Moreover, our method compares favourably with a literature benchmark on instances without rostering constraints, solving three times as many large instances to optimality. This comes at the cost of a doubling in average computing time, but computing time is typically not a bottleneck in the tactical planning phase.

Some care must be taken when extrapolating our results to the full network of Netherlands Railways. First, our experiments show that a small number of excess duties is still required on most instances. This necessitates the need for stand-by crew, on top of the crew members already assigned a reserve template. Second, we focus on medium to large crew bases of the Netherlands. Constructing robust rosters at small stations can be more challenging, as minor timetable changes can have a relatively bigger impact. Third, the template rostering constraints do not guarantee that efficient template-based rosters can indeed be constructed using the chosen templates. A simulation of the complete crew planning process is required to determine whether this is the case.

This chapter also provides interesting directions for future research. While we have constructed scenarios from historical planning data in a straightforward manner, one can imagine that more sophisticated, data-driven approaches offer more representative scenario sets. Alternatively, one could consider imposing some smoothness conditions on the temporal distribution of template capacity to avoid overfitting on historical data. Moreover, integrating the template selection step with crew rostering, in the line of Borndörfer et al. (2017), would be a challenging but promising exercition. Although the method proposed in this chapter is not yet computation-

ally tractable for the full Dutch railway network, it might lend itself to geographic decomposition strategies or meta-heuristic approaches. Finally, we believe that our method can be applied to other crew planning problems where the construction of a preliminary capacity plan precedes a more detailed work assignment.

Chapter 3

Railway Crew Planning with Fairness over Time

This chapter is based on B.T.C. van Rossum, T. Dollevoet and D. Huisman (2024b). 'Railway crew planning with fairness over time'. European Journal of Operational Research *318.1, pp. 55–70.*

3.1 Introduction

Passenger railway operators typically employ large numbers of drivers and guards, who ensure that daily operations run smoothly and passengers experience high service levels. Apart from the general goal of improving employee well-being, there are several reasons why these operators are concerned with the attractiveness of the working conditions of their employees. First, it is important to remain an attractive employer and be able to attract new employees in today's highly competitive labour market. Second, better working conditions can reduce the amount of sick leave and increase overall performance. Third, a favourable working environment reduces the risk of conflicts and strikes, a threat that is particularly imminent in the railway sector where many workers are represented by strong labour unions. Recently, massive strikes in the United Kingdom and the Netherlands demonstrated the power of the unions (Albers and Nandramn, 2022; Topham, 2022).

At Netherlands Railways (NS), the largest passenger railway operator of the Netherlands, the challenge of shaping attractive working conditions is addressed through the use of Sharing-Sweet-and-Sour rules (Abbink et al., 2005). These rules, initially developed to resolve large nationwide strikes in 2001, specify that sweet (attractive) and sour (unattractive) work is allocated fairly over the 28 different crew bases of NS. Here, a *crew base* corresponds to a large station with a fixed group of crew members who are allowed to operate train trips throughout the country but must start and end their work at this station. In the case of guards, sweet work might include trips on modern rolling stocks units, while sour work could consist in working on trains with high risk of passenger aggression. The current Sharing-Sweet-and-Sour rule corresponding to the latter attribute states that the average percentage of aggression work must not exceed 50% per crew base.

In the current crew planning process at NS, the Sharing-Sweet-and-Sour rules are incorporated in the tactical planning phase. Here, a generic crew schedule for the annual plan is constructed based on the yearly timetable and rolling stock schedule. This schedule contains duties, i.e., days of work, that are assumed to be performed every week of the year. The Sharing-Sweet-and-Sour rules require that the sweet and sour work in the generic schedule is distributed fairly over the different crew bases. Next, within each crew base duties are assigned to various roster groups. Each roster group consists of a set of crew members that rotate through a cyclic roster. Under the assumption of generic duties that are repeated every week, members of the same roster group thus perform the same work. Finally, in the operational phase the daily

timetable and rolling stock schedule may deviate from the yearly ones to account for, e.g., planned maintenance works and events. As a result, some of the generic duties might be modified as well.

Unfortunately, the current implementation of Sharing-Sweet-and-Sour rules at NS suffers from two main drawbacks. First of all, these rules impose restrictions on the attractiveness of work at the crew base level but do not offer any guarantees on the attractiveness of any particular roster group. As such, large differences in the quality of rosters of individuals at the same crew base can persist. Second, the rule set is imposed on the yearly schedule in the tactical planning phase, whereas the daily operated schedules might significantly deviate from this yearly schedule and change from week to week. As a result, it is not guaranteed that the realised crew schedules actually satisfy the Sharing-Sweet-and-Sour rules.

To tackle these two issues, NS is currently designing a new crew planning process. Here, in the tactical phase a capacity plan is constructed, i.e., a general roster specifying the days and time windows at which each crew member is expected to work. In the operational planning phase this capacity plan is detailed to a crew schedule by assigning duties to specific crew members. The inputs, i.e., the tasks to be performed on each day, are dynamically revealed, while planned duties must be communicated to crew members well before the day of operations. This imperfect information gives rise to an online optimisation problem (Audibert et al., 2014; Grötschel et al., 2001). In the new setting, Sharing-Sweet-and-Sour rules are enforced at the individual level. This is now possible since duties are assigned directly to individuals. In addition, they would be enforced over a planning period of several months, as it is too restricting to enforce them on each single day or week. As such, the new rules ensure *fairness over time* between the various crew members, a concept introduced by Lodi et al. (2024a). Moreover, it is less likely that operated duties deviate much from the planned ones, as the duties are constructed closer to the day of operations. All in all, the Sharing-Sweet-and-Sour rules in the new process would yield stronger and more reliable guarantees to individual crew members. An auxiliary benefit of this approach is the additional flexibility for the railway operator, as the exact work content can be determined closer to the day of operations.

The resulting problem, which we call the railway crew planning problem with fairness over time, features a challenging combination of characteristics that have rarely been studied in combination. In particular, it is a dynamic resource allocation problem, in which fairness over time with respect to multiple, conflicting attributes is to be

attained across individuals, and in which the set of feasible allocations varies over time and between individuals. Static allocation problems with fairness considerations have received considerable attention in the literature, with successful applications to crew rostering (Breugem et al., 2022b) and vehicle routing (Matl et al., 2018; Matl et al., 2019). Lodi et al. (2024a) and Lodi et al. (2024b) study fairness over time in offline dynamic resource allocation. Bampis et al. (2018) consider online allocation and are therefore more closely related to our work. In their work, however, feasible allocations can be identified in polynomial time, whereas in our case it requires solving a $\mathcal{NP}$-hard crew scheduling problem. Existing literature thus does not provide any practical methods towards solving a large-scale practical problem with this challenging combination of characteristics. It is our goal to address this gap in the literature. In particular, we aim to provide an effective solution approach for attaining fairness over time in a large-scale passenger railway application. Our methods are generalisable to other settings where work must be fairly allocated to individuals in a dynamic fashion.

The contributions of this chapter are threefold. First, we introduce the railway crew planning problem with fairness over time. In other words, we formulate the problem that arises in the operational planning phase of the proposed crew planning process of NS and which features several challenging characteristics. Second, we propose a rolling horizon approach with a penalty-based feedback mechanism and a column generation heuristic to solve this problem. The feedback mechanism steers the work allocation towards one that respects the Sharing-Sweet-and-Sour rules. We propose an integrated method, which simultaneously determines the work content and assigns it to employees, as well as a sequential approach that tackles these two problems sequentially and forms a benchmark for our integrated approach. Third, we apply both approaches to large practical instances from NS with up to 572 guards. The integrated approach is able to satisy the individual rules for on average 95.2% of the crew members, as compared to 86.7% in the sequential approach. Strikingly, this increase in satisfaction levels comes at negligible additional computational cost.

The remainder of this chapter is structured as follows. We formally introduce the railway crew planning problem with fairness over time in Section 3.2. We provide an overview of relevant literature in Section 3.3, and in Section 3.4 we introduce our rolling horizon approach and feedback mechanism. We compare the integrated and sequential approaches on several real-life instances from Netherlands Railways in Section 3.5, and we conclude in Section 3.6.

3.2 Problem Description

We present an overview of the railway crew planning problem with fairness over time, as well as some practical considerations, in Section 3.2.1. We discuss the relevant duty and rostering rules in detail in Sections 3.2.2 and 3.2.3, respectively. Finally, in Section 3.2.4 we present the individual Sharing-Sweet-and-Sour rules.

3.2.1 Overview

Before introducing the problem, we provide a brief overview of railway crew planning terminology. The goal of railway *crew planning* is to assign work to crew members while respecting several labour rules. The work to be performed consists of a set of tasks that result from the timetable and rolling stock schedule. A *task* corresponds to a scheduled train trip characterised by a start and end station, and start and end time. Tasks are assigned to crew members in the form of duties. A *duty* is a sequence of tasks, constituting a day of work, that satisfies all duty rules. For example, each duty must respect a maximum duty length. Constructing duties from tasks is called *crew scheduling*. In *crew rostering* duties are assigned to crew members in the form of a roster. A *roster* specifies, for each crew member and day, whether he or she works on that particular day, and if so, which duty is to be performed. Each roster must satisfy multiple rostering rules, e.g., it must guarantee a minimum number of days off. Figure 3.1 shows an example of a four-week roster for a crew member based in Rotterdam, detailing the duties to be performed throughout this period. In addition to the common terminology above, we will make use of the concept of a template. A *template* is characterised by a day and time window in which an employee can be scheduled to work, and acts as a placeholder for a duty in a roster.

We are now ready to introduce the railway crew planning problem with fairness over time. We assume a planning period of given length, and consider crew planning in the operational planning phase. In this phase, all strategic and tactical crew planning decisions have already been taken. In particular, we assume that a capacity plan is given in the form of a template-based roster for each crew member. To allow some flexibility, the average template spans 9 hours while the length of a typical duty is 8 hours. Moreover, we assume that a set of Sharing-Sweet-and-Sour rules is given, specifying bounds on the (un)attractiveness of work along several attributes. We require that, for each crew member, the total work performed in the planning period satisfies all Sharing-Sweet-and-Sour rules, resulting in fairness over time across crew members.

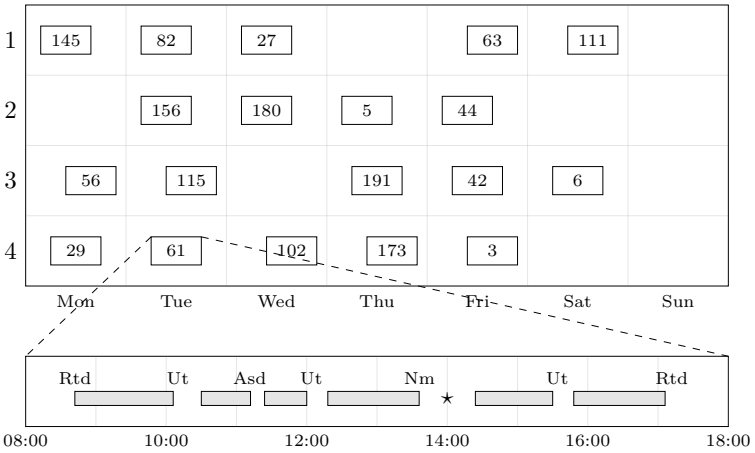


Figure 3.1: Example of a four-week roster of a crew member based in Rotterdam (Rtd). Numbered blocks indicate duties assigned to this crew member, the absence of blocks indicates a rest day. The duty on Tuesday in week four is highlighted, where each block represents a task. For each task the start station (top left) and/or end station (top right) are shown. The star indicates a meal break.

The work to be performed consists of a set of tasks that result from the timetable and rolling stock schedule. It is not known in advance which tasks are to be covered on any particular day of the planning period. Instead, the timetable and rolling stock schedule for specific days are constructed on a rolling horizon basis. This allows the railway operator some flexibility in accounting for, e.g., events and maintenance works. As a result, the tasks to be covered are revealed in a dynamic fashion. The goal of the railway crew planning problem with fairness over time is to detail the template-based rosters to actual rosters by assigning to each template a matching duty, such that all tasks in the planning period are covered, all rostering rules are satisfied, and the work of each crew member satisfies all Sharing-Sweet-and-Sour rules. This is essentially a feasibility problem, since all major crew cost components are already determined by the capacity plan.

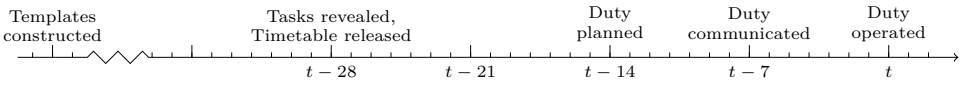


Figure 3.2: Timeline of a typical duty in the proposed crew planning process.

We conclude this section with some remarks on the practical setting of the problem. First, the timetable and rolling stock schedule for a specific day are usually construc-

ted four to six weeks in advance. The updated timetables are released to planners and passengers approximately 28 days in advance. The crew planning problem for this day must then be solved at least two weeks before the day of operations, as to leave enough time for planners to perform some post-processing and to communicate the final schedule to crew members. In practice one could solve a single crew planning problem every day, and communicate the planned duties in batches of, e.g., one week. Since duties are planned much closer to the day of operations than in the current planning process, they are less likely to undergo major modifications in the period between planning and day of operations. Figure 3.2 illustrates the timeline of a typical duty in the proposed crew planning process. Second, while we assume that a fixed template-based roster is given for each crew member, in practice changes to the capacity plan may occur even in the operational planning phase. For example, crew members may request extra days off during the planning period. We do not consider such modifications in this chapter, but the first remark shows that such changes can be easily accomodated in our setting as long as they are known before the crew schedule is constructed. Thirdly, disruptions on the day of operations may cause the performed schedule to deviate from the planned schedule, potentially compromising our ability to satisfy Sharing-Sweet-and-Sour rules. The effect of such disruptions on the ability to satisfy individual Sharing-Sweet-and-Sour rules is outside the scope of this chapter.

3.2.2 Duty Rules

In the course of the planning period we detail the capacity plan to a complete roster by assigning a duty to each template. Each duty must satisfy the following duty rules. First, two tasks can only be performed consecutively when the first task ends at the station where the second task starts, and there is sufficient transition time between the two tasks. The minimum transition time depends on whether two tasks are on the same rolling stock unit or not. Second, each duty should start and end at the same crew base. Third, duties should respect a maximum duty length. The maximum duty length depends on the start time of the duty, e.g., duties starting at the middle of the day have a longer maximum length than duties starting late at night. Finally, each duty should contain a meal break of at least half an hour at a station with a dedicated canteen. The time before and after the meal break should be at most 5.5 hours, ensuring that the meal break is positioned approximately halfway a duty.

3.2.3 Rostering Rules

Not all rostering rules can be incorporated in the construction of a template-based roster since the length of a template is generally significantly longer than that of a duty. For example, incorporating a minimum rest time constraint based on templates would yield overly restricted rosters. As such, we need to respect several roster rules when assigning duties to crew members and our problem becomes a planning problem rather than merely a scheduling problem. First, when assigning a duty to a crew member, this duty should match the template of the corresponding day in the roster of this crew member. In particular, the duty should be contained in the time window specified by the template. Second, there should be a minimum rest time of twelve hours between consecutive duties. Finally, we impose rotation constraints on duty patterns to align them with the employees' circadian rhythms and thereby increase the attractiveness of rosters (see Appendix 3.A for a formal definition). In practice, labour legislations and collective agreements specify additional rules that crew rosters should satisfy. In this chapter, we assume that these need not be modelled explicitly in the crew planning problem as they can be (i) incorporated in the construction of the template-based roster (e.g., minimum number of days off) or (ii) are best incorporated implicitly (e.g., at most 12 very long duties per year).

3.2.4 Sharing-Sweet-and-Sour Rules

We consider Sharing-Sweet-and-Sour (SS&S) rules at the individual level, as compared to the current rules on crew base level. This implies that the fractions of sweet and sour work of each crew member are lower- and upper-bounded, respectively, thus guaranteeing a minimum level of attractiveness for each individual employee. The individual rules do not explicitly preclude some variation between crew members. However, choosing appropriate bounds will automatically reduce such variation, and we do not consider variation problematic when the overall guarantee is tight.

Each individual SS&S rule concerns a specific *attribute*, i.e., a characteristic of work defining its (un)attractiveness. We assume that each piece of work can be given a *score* with respect to each attribute. For our purpose it is important to distinguish between duty-averaged and time-averaged attributes. In case of a *duty-averaged* attribute, the score is computed based on a complete duty and captures possible interactions between tasks in the duty. The score of a roster is computed by averaging the scores of all duties in the roster. In case of a *time-averaged* attribute, each task contributes an attribute duration: a value between zero and the task duration. The

score of a duty is then computed by dividing the sum, over all tasks in the duty, of attribute durations by the sum of task durations, yielding a fraction between zero and one. Similarly, the score of a roster is computed by dividing the sum, over all duties and tasks therein, of attribute durations by the sum of task durations.

An individual SS&S rule consists of a combination of an attribute and a lower or upper bound on the score of the roster performed in the planning period with respect to this attribute. In practice, one might also consider a rolling window set-up where the score is computed over the work performed in the last months.

In this chapter we consider SS&S rules, which apply to the guards at Netherlands Railways, with regards to the following four attributes:

- *Duty length*: The duty length is defined as the difference in start and end times of a duty. Duties of shorter length are preferred.
- *Fraction of Type-A work*: Type-A work consists of desirable work on, for example, Intercity trains, which only stop at large stations, or modern rolling stock units.
- *Fraction of aggression work*: Work with a high risk of passenger aggression is undesirable. It occurs mostly in the metropolitan areas of the Netherlands.
- *Fraction of work on double-decker trains*: Work on double-decker trains is considered undesirable, as it requires guards to climb a large number of stairs.

Duty length is the only duty-averaged attribute, and its score is computed by averaging the lengths over duties in a roster. All other attributes are time-averaged attributes, and their scores are obtained by dividing the contribution of tasks by the total duration of all tasks in a roster. The fraction of Type-A work is lower-bounded by its corresponding SS&S rule, whereas all other attributes are upper-bounded.

3.3 Literature Review

Recently, several authors have started to study fairness over time in dynamic resource allocation problems. Lodi et al. (2024a) consider the problem of achieving fair allocations over time, deriving some structural results regarding fairness-enhancing allocation policies. They apply their framework to the ambulance allocation problem. Lodi et al. (2024b) propose a probabilistic reformulation to remove symmetry from the formulation of Lodi et al. (2024a) in case utilities are time-invariant. In

Chapter 5, we study the performance of an online allocation policy with an application to capacitated vehicle routing. In that chapter, we consider a fixed pool of fully homogeneous workers, whereas in this chapter, the crew members work at heterogeneous locations and times. Si Salem et al. (2022) study the case where utility functions, governing the utility derived from a given allocation, vary over time. Bampis et al. (2018) consider feasible allocations that are not known in advance and might vary between individuals and time periods. They propose several offline and online approximation algorithms. While their setting most resembles ours, their methods do not translate to our case, where determining a feasible allocation requires solving a crew scheduling problem. To address this gap in the literature, we propose a rolling horizon approach with feedback mechanism that is able to attain fairness over time in a large-scale crew planning application.

Our work belongs to a large body of literature on crew planning in public transport and airline operations. Caprara et al. (1999) and Abbink et al. (2018) offer excellent introductions to the field of railway crew planning. The crew planning problem is commonly decomposed into crew scheduling and crew rostering. We refer to Heil et al. (2020) for a recent survey on railway crew scheduling and to Kohl and Karisch (2004) for an introduction to airline crew rostering.

The crew planning problem we consider is situated after the tactical planning phase and before the day of operations. Several authors have studied a similar setting, mainly with the goal of minimising disturbances due to devations from the tactical planning. Stojković et al. (1998) consider the problem of generating modified airline crew pairings that minimise disturbances from a planned schedule. Huisman (2007) considers rescheduling railway crew due to maintenance works and solves this problem using a column generation heuristic. Breugem et al. (2022a) extend his work by integrating the rescheduling for multiple days simultaneously and providing a formulation for the integrated crew replanning problem. They solve the problem using valid inequalities and a column generation heuristic. Saddoune et al. (2009) solve an operational airline crew pairing problem using a rolling horizon approach and column generation, showing that schedules obtained in early planning phases have little added value when the flight schedule is not regular. Rählmann et al. (2021) perform robust tactical planning by choosing *duty frames* that are able to cover duties in different demand scenarios, i.e., that lead to robust solutions in the operational planning phase. In our work we consider crew scheduling in the operational phase given only a template-based capacity plan.

Next to all relevant duty rules, in this chapter we consider all rostering rules that cannot be included in the construction of the template-based roster directly. We thus partially integrate the crew scheduling and crew rostering problems, an integration that has received relatively little attention in the literature. Ernst et al. (2001) propose an integrated model to compute cyclic and acyclic rosters. Their solution approach relies on a complete enumeration of all possible duties, and hence does not scale well beyond the sparse Australian railway network under consideration. Mesquita et al. (2013) develop a heuristic Benders decomposition method to solve the integrated problem of vehicle scheduling, crew scheduling, and crew rostering. Borndörfer et al. (2017) consider the integrated duty scheduling and rostering problem. Here, duties and rosters are linked through *duty templates*, coarse duty representations that are similar to our templates, and the problem is solved using Benders decomposition. Saddoune et al. (2012) solve the integrated crew pairing and crew assignment problem using column generation and dynamic constraint aggregation, while Zeighami et al. (2020) consider the same integration in a personalised context, and tackle the problem using a combination of Lagrangian decomposition, column generation, and dynamic constraint aggregation. Finally, Breugem et al. (2022a) propose a mathematical formulation of the integrated crew replanning problem and solve this using a column generation-based fixing heuristic.

The goal of the SS&S rules at NS is to obtain fair and attractive crew schedules, a topic that is gaining popularity in recent years (see Wolbeck (2019) for a review of personnel scheduling with fairness aspects). Most research in this area focuses on including fairness and attractiveness in the crew rostering problem. Hartog et al. (2009) describe the role of SS&S rules in the crew rostering process of NS. Borndörfer et al. (2015) and Maenhout and Vanhoucke (2010) propose heuristic solution methods for the crew rostering problem with personal preferences and fairness considerations, whereas Er-Rbib et al. (2021) tackle the problem of constructing cyclic bus driver rosters with group-based preferences through matheuristic approaches. Zeighami and Soumis (2019) use Benders decomposition and column generation to integrate the airline crew pairing and crew rostering problems, and use their approach to maximise the number of granted personal vacation requests by (co)pilots. Quesnel et al. (2020) consider airline crew planning too, and include crew preferences for complex features of crew pairings in the crew pairing problem with the goal to improve the quality of the resulting crew rosters. Nishi et al. (2014) suggest a two-level decomposition for crew rostering, attaining fairness by minimising the maximum workload per roster group. Several authors particularly consider workload balancing in the

crew scheduling problem. Wang et al. (2017) focus on workload balancing for depot shunting drivers, while Amaya and Uribe (2018) consider workload balancing in a mine railway setting. Balakrishnan et al. (2022) propose to use minimum duty length times in order to avoid workload imbalances in crew assignment problems.

Similar to our work, Breugem et al. (2022b) consider Sharing-Sweet-and-Sour rules at NS. In particular, they analyse the trade-off between fairness and attractiveness in crew rostering, where a given set of duties is repeated every week and must be allocated to different cyclic roster groups. Fairness refers to the imbalance in attribute scores across roster groups, while attractiveness refers to the extent to which sweet and sour work are balanced within the weeks of a roster group. They propose a branch-price-and-cut method, and an efficient three-phase heuristic for the same problem is introduced in Breugem et al. (2023). Our work differs from theirs in three important ways. First, our concept of fairness strongly differs from the one used in Breugem et al. (2022b) as we consider SS&S at the individual level, and deem a schedule fair when all crew members satisfy the same set of rules. We do not explicitly consider attractiveness in our work, since we strongly prioritise the individual SS&S rules. However, our penalty-based feedback mechanism is likely to automatically correct for crew schedules featuring strong week-to-week variations in attribute scores. Second, constructing duties that cover a given set of tasks is part of our problem, whereas duties covering all tasks are provided as input in the crew rostering application. Third, we face an online optimisation problem, as the time-varying task sets are dynamically revealed throughout the planning period. This prevents the use of a single large offline optimisation model.

A smaller stream of literature considers fairness and attractiveness explicitly in crew scheduling. Abbink et al. (2005) describe how the SS&S rules at NS are incorporated in the crew scheduling problem. Jütte et al. (2017) consider crew scheduling for a railway freight carrier where unpopular duties must be distributed fairly over different crew bases. Similar to the SS&S rules, constraints limit the unpopularity and unfairness per crew base. Gattermann-Itschert et al. (2023) use a machine learning model to predict railway planner's preferences for nonmonetary duty characteristics and maximise the likelihood that a solution is accepted by planners. A related stream of literature considers the problem of balancing workload across routes in vehicle routing problems, see Matl et al. (2018) for an overview thereof. Matl et al. (2019) distinguish between variable-sum and constant-sum resources, i.e., resources whose total sum does or does not depend on the chosen resource allocation. We

contribute to the applied fairness literature as we consider a dynamic setting, and we are one of the few to consider fairness at the level of the individual crew member. Moreover, we consider both variable-sum (duty length) and constant-sum (all other attributes) resources.

3.4 Solution Approach

We propose a rolling horizon approach to cope with the multi-period nature of the problem. Similar approaches have been successfully applied to crew planning problems (see, e.g., Saddoune et al. (2009) and Quesnel et al. (2020) for applications to airline crew pairing). For each day of the planning period, we solve a crew planning problem that covers this day and potentially the next day. We fix the duties in the solution for the first day of the problem in our crew plan and proceed to the next day, until a crew plan for the entire planning period has been constructed.

Within each planning problem we place penalties on the assignment of work to individuals to steer towards satisfaction of individual SS&S rules at the end of the planning period. At each day we compute, for each crew member and each SS&S attribute, the score of the work assigned to this crew member until this day. When this score (nearly) violates the threshold, we place penalties on the assignment of more sour work, with regards to this attribute, to this particular crew member. These penalties are set to be (i) increasing in the amount of sour work to be assigned and (ii) increasing in the amount of sour work already performed, i.e., the score of previously assigned work. By minimising the sum of penalties in each planning problem, we aim to establish a feedback mechanism that increases the quality of work of this individual and steers the SS&S score towards the right side of the threshold.

In each planning problem we need to (i) generate duties covering all tasks and (ii) assign these duties to crew members as to minimise penalties. In our proposed *integrated* approach we integrate these two steps: we generate duties that minimise penalties and simultaneously assign them to individual crew members. Since no obvious benchmark method for our problem exists, as NS does not yet use a planning process with individual SS&S rules, we also propose the *sequential* approach, in which we do not integrate the two steps. Instead, we first generate duties that satisfy crew base level SS&S rules but do not take into account individual work histories, thereby mimicking the current crew planning process of NS. Within each crew base, we then assign these duties to crew members in a way that minimises the sum of penalties.

Comparing the two approaches allows us to analyse to what extent it is beneficial to tailor duties to the work history of individual crew members.

The remainder of this section is structured as follows. We provide a mathematical formulation of the crew planning problem in Section 3.4.1 and describe the column generation heuristic, as well as several acceleration strategies, in Section 3.4.2. We define the penalty function used in the feedback mechanism in Section 3.4.3. Finally, we present the integrated and sequential approaches in Sections 3.4.4 and 3.4.5, respectively.

3.4.1 Mathematical Formulation of the Crew Planning Problem

We now provide a mathematical formulation of the crew planning problem that is repeatedly solved in the rolling horizon approach. Recall that we consider problems spanning either one or two days. Here, we limit ourselves to the formulation of the single-day problems, as this is the set-up used in the majority of our experiments. We show the extension to multi-day problems by introducing coupling roster constraints in Appendix 3.A.

Let t be the next planning day encountered in the rolling horizon approach, and denote by K_t and R_t the set of tasks to be covered and the set of crew members working on this day, respectively. For each crew member $r \in R_t$, let Δ_t^r be the set of duties that can be performed by this crew member on this day. This set is implicitly defined by the duty rules, the template in the roster, and the work of the previous day (through the rest time and rotation constraints). The objective coefficient of duty $\delta \in \Delta_t^r$ is given by c_{rt}^δ. In the integrated approach, this coefficient is computed as the sum, over all tasks in the duty, of the penalty of assigning a particular task to crew member r. In the sequential approach, this coefficient equals the penalty of assigning the complete duty to individual r. We describe the computation of the objective coefficient in more detail in Sections 3.4.4 and 3.4.5. Finally, the binary parameter a_{tk}^δ indicates whether task $k \in K_t$ is covered by duty δ.

For each $r \in R_t$ the binary decision variable x_{rt}^δ indicates whether duty δ is assigned to crew member r on day t. The crew planning problem for day t can then be formulated as the following binary linear program:

$$\min \quad \sum_{r \in R_t} \sum_{\delta \in \Delta_t^r} c_{rt}^\delta x_{rt}^\delta \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{r \in R_t} \sum_{\delta \in \Delta_t^r} a_{tk}^{\delta} x_{rt}^{\delta} \geq 1 \qquad \forall k \in K_t \tag{3.1b}$$

$$\sum_{\delta \in \Delta_t^r} x_{rt}^{\delta} = 1 \qquad \forall r \in R_t \tag{3.1c}$$

$$x_{rt}^{\delta} \in \mathbb{B} \qquad \forall r \in R_t, \delta \in \Delta_t^r. \tag{3.1d}$$

The objective (3.1a) is to minimise the penalties of selected duties, i.e., to avoid duties with high penalty scores and select low-penalty duties whenever possible. In this way we aim to obtain a duty assignment where sour work is shifted away from crew members currently not satisfying the rules, and allocated to those that are currently well below the threshold values. Constraints (3.1b) ensure that each task is covered at least once, allowing for potential overcoverage through deadheading, while constraints (3.1c) guarantee that each active crew member is assigned exactly one duty. The domain of the decision variables is defined by constraints (3.1d).

3.4.2 Column Generation Heuristic

We solve each crew planning problem with a column generation heuristic. We refer to Desaulniers et al. (2005) and Lübbecke (2010) for general introductions to column generation, and Huisman (2007), Potthoff et al. (2010), and Breugem et al. (2022a) for successful applications to railway crew planning problems. In particular, we relax constraints (3.1d) to $x_{rt}^{\delta} \geq 0$ and initialise model (3.1) with slack variables for constraints (3.1b) and (3.1c). We refer to the resulting model as the restricted master problem (RMP). In each iteration of the algorithm, we solve the RMP to optimality and solve a series of pricing problems to identify columns, i.e., duty variables, with negative reduced cost that could potentially improve the objective value. These columns are added to the RMP and the algorithm repeats until no such columns are found.

Let $\lambda_{tk} \in \mathbb{R}_{\geq 0}$ and $\pi_{rt} \in \mathbb{R}$ be the dual variables corresponding to constraints (3.1b) and (3.1c), respectively. The reduced cost of duty x_{rt}^{δ} is then given by

$$c_{rt}^{\delta} - \sum_{k \in K_t} \lambda_{tk} a_{tk}^{\delta} - \pi_{rt}. \tag{3.2}$$

The pricing problem now corresponds to finding the duty, i.e., sequence of tasks to be performed consecutively, with minimum reduced cost. Recall that each duty must start and end at the same depot, respect the maximum duty length, allow for a meal

break approximately halfway the duty, and fall within the time window specified by the template of a crew member.

While polynomial-time exact algorithms exist for this problem (see, e.g., Huisman (2007)), their running time becomes prohibitively large on the instances we consider. As such, we model the pricing problem as a series of resource constrained shortest path problems (RCSPPs), one for each template, and apply a heuristic labelling algorithm with completion bounds (Breugem et al., 2022a). For each template a directed acyclic graph is constructed containing all tasks that lie within the time window specified by the template. The reduced cost, including the penalties used in the integrated approach, can be distributed on the arcs in this graph. The pricing problem for this template can then be modelled as a RCSPP with resources for the presence of a meal break, the time without meal break, and the current duty length. We refer to Appendix 3.B for more details on the pricing problems. Since we use a heuristic pricing algorithm, the master problem is not necessarily solved to optimality when the column generation algorithm terminates.

We apply several acceleration techniques to speed up the algorithm (see Desaulniers et al. (2002) for an overview of commonly used techniques). First, to avoid the notorious tailing-off effect, we terminate the algorithm when the relative decrease in objective value in two consecutive iterations is below a threshold parameter ϵ. Second, to limit the number of columns entering the RMP, we do not add all columns returned from the pricing problems, but only those that are sufficiently task-disjoint (Breugem et al., 2022a). Finally, we perform column management to limit the number of columns in the RMP. More specifically, in each iteration we remove all columns whose reduced cost exceeded a certain threshold for more than a given number of iterations.

The algorithm outlined above yields a fractional solution to the RMP, whereas in our rolling horizon approach we aim to fix duties for the current day in our crew plan. As such, we need to convert the fractional solution to a binary one. Necessitated by the large scale of our problem, we use the following diving heuristic (see Joncour et al. (2010) for an overview of column generation based primal heuristics). Once the column generation algorithm terminates, all duties whose solution value is above a threshold parameter τ are fixed. The threshold value must be strictly larger than $\frac{1}{2}$ to avoid double assignments to the same crew member. If no such duties exist, we fix the duty with the highest solution value. The remaining problem, which is now strictly smaller in size, is re-solved using column generation and the fixing rule is applied again until the solution is completely binary. We remark that the strong

heuristic nature of this approach may cause some tasks to remain uncovered. In our experiments we find that less than 1% of the tasks remains uncovered.

3.4.3 Feedback Mechanism

Our feedback mechanism relies on a penalty function to penalise the allocation of more sour work to crew members whose work history does not satisfy the individual SS&S rules. Minimisation of penalties in the objective function of the model outlined in Section 3.4.1 should then lead to a more favourable work assignment for said crew members, thereby bringing their SS&S scores (closer) to the right side of the threshold. We compute penalties for all SS&S attributes separately. The penalty for a piece of work is defined as the sum of penalties over all attributes.

The penalty for a given combination of work, attribute, and crew member depends on two factors: (i) the score of the work with respect to the attribute, and (ii) the score of the work history of the crew member with respect to the attribute. The penalty is increasing in the score of the work, as a higher score corresponds to less favourable work and is therefore more likely to drive the crew member away from satisfying the individual SS&S rules. The penalty is also increasing in the score of the work history, since we prioritise the allocation of attractive work to employees who have already performed relatively high amounts of unfavourable work.

We now formally define the penalty function. Without loss of generality, assume that the score of each attribute $a \in A$ takes values in the range $[0, r_a]$, and that the SS&S rule specifies that the score should be at most b_a, where $b_a < r_a$. The penalty $f_a(u, s)$ of assigning a piece of work with attribute score u to a crew member with work history score s is then defined as the product $f_a(u, s) = g_a(u) \cdot h_a(s)$. Here, the function g captures the score of a single piece of work, while h reflects the score of the work history of the crew member. In our work we use $g_a(u) = \frac{u}{r_a}$. For $h_a(s)$ we use the piece-wise linear function illustrated in Figure 3.3. This piece-wise structure prioritises the assignment of sweet work to crew members whose current score is above the threshold, and simultaneously attempts to avoid the assignment of sour work to crew members that currently satisfy the SS&S rule yet risk crossing the threshold.

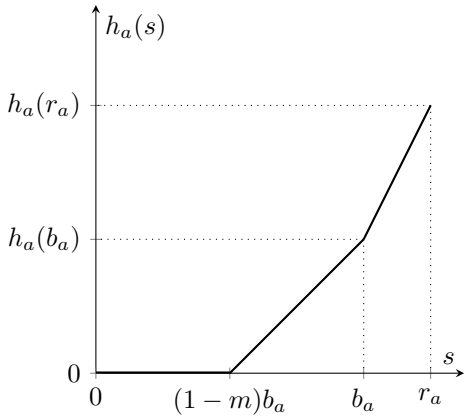


Figure 3.3: Example of penalty function.

The function h_a is mathematically defined as follows:

$$h_a(s) = \begin{cases} 0 & \text{if } 0 \leq s \leq (1-m)b_a, \\ \alpha(1-\gamma) \cdot \frac{s-(1-m)b_a}{r_a} & \text{if } (1-m)b_a < s \leq b_a, \\ h_a(b_a) + \alpha \cdot \frac{s-b_a}{r_a} & \text{if } b_a < s \leq r_a. \end{cases} \tag{3.3}$$

Here, $\alpha > 0$ is a scaling parameter, $m \in (0,1)$ is a parameter indicating how close the history score must be to the threshold for the penalty function to be activated, and $\gamma \in (0,1)$ regulates the scaling factor between penalties for crew members that currently satisfy the rule versus those that do not. The function evaluates to zero when $s \leq (1-m)b_a$, increases linearly in s when $s > (1-m)b_a$ and $s \leq b_a$, and increases linearly at a faster rate when $s > b_a$. Both g_a and h_a are normalised by dividing through r_a, to ensure that different attributes contribute equally to the objective function. While computational experiments showed that this function performs well, our approach allows for any choice of functional form for f_a.

3.4.4 Integrated Approach

In the integrated approach we integrate the generation of duties and the minimisation of penalties in each planning problem. For each crew member $r \in R$, day t, and potential duty δ, the objective coefficient c^{δ}_{rt} is defined as the sum, over all tasks in the duty and all attributes, of the penalty score of assigning a particular task to crew member r. This penalty is computed using the penalty function described

in Section 3.4.3. Each penalty is weighted by the duration of the task divided by the average length of a duty, to avoid small tasks contributing disproportionally to the objective function. Note that we do not divide by the actual duty length, as its exact value is yet to be determined within the pricing problems. The advantage of this approach is that the penalties can be easily decomposed on the arcs of the pricing graph without increasing the complexity of the pricing problems. A downside is that in general, due to the nonlinearities in the SS&S attributes, the penalty of a duty does not equal the sum of penalties of its tasks. As such, the penalties used in the integrated approach constitute an approximation of the true penalties. The assignment of duties to crew members in the solution to the crew planning problem is used to build the crew plan in the rolling horizon approach.

Table 3.1: Example duty containing four tasks. Durations are denoted in minutes.

Task			Aggression work		Double-decker work	
Start	End	Duration	Duration	Fraction	Duration	Fraction
Rtd	Ut	37	0	0.00	37	1.00
Ut	Nm	54	0	0.00	54	1.00
Nm	Asd	82	23	0.28	82	1.00
Asd	Rtd	75	18	0.24	0	0.00
Total		248	41	0.17	173	0.70

We now illustrate the use of penalties in the integrated approach by means of a small example with two SS&S attributes, a single crew member, and a single duty. Assume that the only SS&S attributes are the fractions of aggression (`agg`) and double-decker (`dd`) work with thresholds $b_{\texttt{agg}} = 0.30$ and $b_{\texttt{dd}} = 0.50$, respectively. Furthermore, assume that the current scores of our crew member on these attributes are $s_{\texttt{agg}} = 0.29$ and $s_{\texttt{dd}} = 0.55$, respectively. In other words, the level of aggression work performed so far is slightly below the threshold, whereas the fraction of double-decker work is currently above the maximum allowed value. Finally, the duty we consider assigning to this crew member is given in Table 3.1: a round trip from Rotterdam (Rtd) to Utrecht (Ut), Nijmegen (Nm), Amsterdam (Asd), and back to Rotterdam.

We compute the penalties that the integrated approach will place on the assignment of this duty to our crew member using the following parameters for the penalty function: $\alpha = 1{,}000$, $m = 0.1$, and $\gamma = 0.5$. Recall that the penalty of a duty is the sum, over all attributes, of the penalties of all tasks. The penalty of a single task can

be obtained using the penalty function $f = g \cdot h$ as described in Section 3.4.3, and is then weighted by the relative duration of the task. To weigh the relative contribution of each task, we assume an average duty length of four hours. For the aggression attribute, the function h evaluates to

$$h_{\texttt{agg}}(0.29) = 1{,}000 \cdot (1 - 0.5) \cdot \frac{0.29 - (1 - 0.1) \cdot 0.30}{1} = 10.$$

For each single task, the component g equals the fraction of aggression work as listed in Table 3.1. Summing over all tasks in the duty, we find that the total penalty for aggression equals

$$10 \cdot \left(\frac{37}{240} \cdot 0.00 + \frac{54}{240} \cdot 0.00 + \frac{82}{240} \cdot 0.28 + \frac{75}{240} \cdot 0.24 \right) = 1.71.$$

Since the current score on double-decker work exceeds the threshold, the value $h_{\texttt{dd}}(0.55)$ is computed as follows:

$$h_{\texttt{dd}}(0.55) = 1{,}000 \cdot (1 - 0.5) \cdot \frac{0.50 - (1 - 0.1) \cdot 0.50}{1} + 1{,}000 \cdot \frac{0.55 - 0.50}{1} = 75.$$

The total penalty for double-decker work then equals

$$75 \cdot \left(\frac{37}{240} \cdot 1.00 + \frac{54}{240} \cdot 1.00 + \frac{82}{240} \cdot 1.00 + \frac{75}{240} \cdot 0.00 \right) = 54.06.$$

The total penalty of this duty in the integrated method thus evaluates to 55.77. The relatively high contribution of double-decker work is caused by the high fraction of double-decker work in this duty in combination with the current score of the crew member exceeding the threshold.

3.4.5 Sequential Approach

In this section we propose the sequential method, which we use to evaluate the relative quality of the integrated method. The sequential method consists of two steps. First, duties are generated by solving model (3.1) without penalties in the objective function, i.e., with all c^{δ}_{rt} equal to zero. Instead, we include SS&S rules at the crew base level to increase the likelihood that these duties allow for satisfaction of SS&S rules at the individual level. All duty variables in the model remain indexed by crew member, to ensure that they match the individual template-based rosters. In the second step, the first stage solution is improved by optimally (re-)allocating

the chosen duties to specific crew members. Similar to the integrated method, we place a penalty on the allocation of duties to crew members and minimise the sum of penalties. A practical benefit of this approach is that it would not require any modifications to the way in which NS currently generates duties.

We now describe the sequential method in more detail by introducing the SS&S rules at the crew base level. Let B be the set of crew bases, and R_t^b be the set of crew members working from crew base $b \in B$ at day t. Moreover, let s_a^δ be the score of duty δ with respect to attribute a. Finally, let t_a^δ equal one in case a is the duty length attribute, and the total duration of tasks in δ otherwise. The SS&S rules then read

$$\frac{\sum_{r \in R_t^b} \sum_{\delta \in \Delta_t^r} s_a^\delta x_{rt}^\delta}{\sum_{r \in R_t^b} \sum_{\delta \in \Delta_t^r} t_a^\delta x_{rt}^\delta} \leq (1-\eta) b_a \qquad \forall a \in A, b \in B. \tag{3.4}$$

The parameter $\eta \in (0,1)$ is used to slightly decrease the value of the SS&S threshold, since it is nearly impossible to satisfy SS&S rules at the individual level with duties that only just satisfy the rules at the crew base level. Computational experiments showed that $\eta = 0.025$, i.e., a tightening of the threshold by 2.5%, worked well.

We linearise constraints (3.4) as follows. First consider the case where attribute a is the duty-averaged duty-length attribute. Let ℓ^δ be the length of duty δ. The linearised constraints read

$$\sum_{r \in R_t^b} \sum_{\delta \in \Delta_t^r} (\ell^\delta - (1-\eta) b_a) x_{rt}^\delta \leq p_{ab} \qquad \forall b \in B. \tag{3.5}$$

Here, $p_{ab} \geq 0$ is a slack variable which enters the objective function as a penalty to ensure that a feasible solution always exists. Now consider the case where attribute a is a time-averaged attribute, i.e., the Type-A, aggression, or double-decker attribute. In this case we need to formulate the SS&S constraint at the task level. Let u_a^k be the total time task k contributes to attribute a, e.g., the total time of Type-A work, and w^k be the total work time in task k. The resulting SS&S constraints read

$$\sum_{r \in R_t^b} \sum_{\delta \in \Delta_t^r} x_{rt}^\delta \sum_{k \in K_t} a_{tk}^\delta (u_a^k - (1-\eta) b_a w^k) \leq p_{ab} \qquad \forall b \in B. \tag{3.6}$$

Both types of constraints are easily amendable to the pricing problems: the duals of (3.5) can be decomposed over source and sink arcs, whereas the duals of (3.6) can be

placed on the incoming arcs of each task.

To summarise, the first step of the sequential approach now consists of solving program (3.1), (3.5), and (3.6) for all time-averaged attributes using the column generation heuristic of Section 3.4.2. In preliminary experiments we observe that this implementation of the sequential approach performs particularly poorly with respect to the duty length attribute. It proved to be beneficial to reduce the maximum allowed duty length by half an hour to at most 9 hours. From now on, we will use this reduction in maximum duty length in all experiments involving the sequential approach. A similar modification did not improve the performance of the integrated approach.

Once a solution is obtained, we proceed to the second step. For each selected duty variable we iterate over all crew members, and add a duplicate of this variable whenever a crew member is allowed to perform this duty. A crew member is allowed to perform the duty when it starts at his or her crew base, and when it is compatible with the template and work history of this crew member. It might happen that a duty can only be assigned to its original, first-stage crew member. Finally, we place a penalty c^{δ}_{rt} on each duty variable, measuring how desirable it is to assign the duty to a particular crew member. In contrast to the integrated approach, these penalties are not approximations but equal to the true penalties of the duties. We then solve this model to optimality with the requirement that all duty variables on the first day of the planning problem are binary, and fix the duties of the first day in our crew plan.

The first step of the sequential approach mimicks the current crew planning process at NS, where SS&S rules are included at the crew base level. The second step of the sequential approach constitutes a form of post-optimisation: we exchange duties between crew members to minimise the sum of penalties, steering the work allocation towards satisfaction of individual SS&S rules. We have also experimented with applying this post-optimisation on top of the integrated method, but this did not yield any significant gains.

We now return to the example from Section 3.4.4 to see what penalty would be used in the sequential method. The penalty is now computed over the score of the full duty, and not over the single tasks. The values $h_{\texttt{agg}}(0.29) = 10$ and $h_{\texttt{dd}}(0.55) = 75$ depend only on the work history and are therefore identical to those in the integrated approach. The attribute scores of the duty equal $g_{\texttt{agg}}(0.17) = 0.17$ and $g_{\texttt{dd}}(0.70) =$

0.70, respectively. The penalty of the duty therefore equals

$$10 \cdot 0.17 + 75 \cdot 0.70 = 54.20.$$

Note that this penalty is slightly below the one computed in the integrated approach.

3.5 Computational Experiments

In this section we show the effectiveness of the integrated approach by evaluating it on three real-life instances from Netherlands Railways. We describe the instances and the parameter settings in Sections 3.5.1 and 3.5.2, respectively. In Section 3.5.3, we show that, in contrast to the sequential method, the integrated method is able to achieve high satisfaction levels of individual SS&S rules. We further illustrate the strength of the integrated approach by analysing the results of one instance in detail in Section 3.5.4, and show that extending the optimisation horizon from one to two days has no apparent benefits in Section 3.5.5. In Section 3.5.6, we evaluate computation times, showing that the integrated approach is competitive with the sequential approach and fast enough for use in an operational setting. Finally, we justify our parameter settings with a sensitivity analysis in Section 3.5.7.

3.5.1 Instances from Netherlands Railways

In practice, the crew scheduling problem encountered by Netherlands Railways features the majority of the Dutch railway network and contains over 10,000 tasks per day. Solving a single daily problem with our proposed method takes prohibitively long, and simulating a planning period of multiple weeks would be a time-consuming exercise. Instead, we artificially derive three smaller instances based on real-life data of NS, which we call `Center`, `South-West`, and `East`. Each instance contains several crew bases that are geographically close to each other, as illustrated in Figure 3.4. Here, the tracks operated by Netherlands Railways are indicated by black lines, and the crew bases included in the instances are illustrated by white labels. Instance `Center` contains the crew bases Amersfoort (Amf), Amsterdam (Asd), and Utrecht (Ut). Instance `South-West` contains the crew bases Dordrecht (Ddr), The Hague (Gvc), Roosendaal (Rsd), Rotterdam (Rtd), and Vlissingen (Vs). Finally, instance `East` contains the crew bases Arnhem (Ah), Enschede (Es), Hengelo (Hgl), Nijmegen (Nm), and Zwolle (Zl).

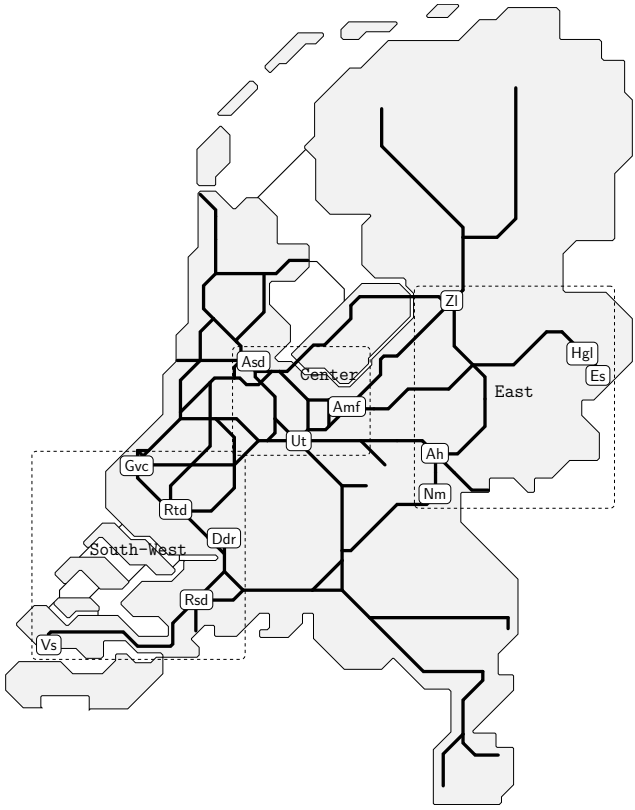


Figure 3.4: Location of the three instances on the Dutch railway network.

Our planning period spans the seven weeks from October 4 to November 21 of 2021. This period is representative of regular operations as the timetable was largely unaffected by COVID-19 measures. Since NS currently works with cyclic duty-based rosters, whereas we assume that template-based individual rosters are given, we artifically construct the necessary inputs to our problem. We do so by converting the cyclic rosters to template-based individual rosters. The tasks to be performed on each day of the planning period are obtained from the realised crew plan of NS. For each instance and day, we group all duties that were performed by guards of the crew bases of this instance on this day. The set of tasks to be performed equals the set of tasks spanned by these duties. Note that the assignment of tasks to crew bases is already partly fixed by the way in which our instances are constructed, and that potentially better solutions could be obtained by considering more crew bases simultaneously.

Table 3.2 displays some characteristic of the three instances. For each crew base the number of guards, number of templates in the rosters, and number of tasks to be performed in the 49-day planning period are shown. Moreover, for each crew base the average score of these tasks on SS&S attributes Type-A work, aggression work, and work on double-decker trains are given. These scores are given in grey as they are based on the realised crew plan, and are therefore not strictly spoken input to our problem. However, these scores do reflect the strong geographical variation in the distribution of sweet and sour work. Finally, totals and averages per instance are given in bold.

Table 3.2: Characteristics of the three instances.

Instance	Crew base	Crew	Templates	Tasks	Type-A	Aggression	Double-decker
`Center`	Amf	104	2,282	22,260	0.51	0.20	0.15
	Asd	230	4,235	39,696	0.34	0.31	0.30
	Ut	227	4,557	38,595	0.32	0.17	0.20
	Total	**561**	**11,074**	**100,551**	**0.37**	**0.23**	**0.23**
`South-West`	Ddr	61	1,267	15,522	0.37	0.17	0.41
	Gvc	203	3,605	41,861	0.41	0.22	0.16
	Rsd	80	1,778	19,547	0.48	0.11	0.47
	Rtd	186	2,639	27,584	0.38	0.22	0.34
	Vs	42	721	8,215	0.40	0.11	0.80
	Total	**572**	**10,010**	**112,729**	**0.41**	**0.19**	**0.34**
`East`	Ah	146	2,415	23,907	0.51	0.14	0.38
	Es	60	1,260	15,626	0.74	0.08	0.25
	Hgl	28	469	4,085	0.53	0.04	0.16
	Nm	90	1,974	20,658	0.48	0.13	0.45
	Zl	124	2,788	27,535	0.47	0.15	0.10
	Total	**448**	**8,906**	**91,811**	**0.52**	**0.13**	**0.27**

Instance `South-West` is the largest, with a total of 572 crew members and 112,729 tasks to be performed over the seven-week planning period. In other words, the average daily crew scheduling problem contains 2,301 tasks, yielding a computationally challenging instance. The ratios of templates to crew members and of tasks to templates are approximately constant over the different crew bases: the average crew member works 19 days during the planning period and the average duty contains about 10 tasks. Moreover, sweet and sour work is not distributed evenly over the country. In particular, crew bases located in `East`, an area that is not very densely populated, perform relatively more Type-A work. Work with high passenger aggression, on the other hand, is performed more frequently by crew members working in `Center` or `South-West` as their crew bases are located in the more metropolitan areas

of the Netherlands. A strong discrepancy occurs even within instances, however: for example, the fraction of aggression work in Amsterdam (0.31) is almost twice that of Utrecht (0.17).

Finally, Table 3.3 shows the threshold values of the individual SS&S rules. For each instance the maximum average duty length, minimum fraction of Type-A work, maximum fraction of aggression work, and maximum fraction of work on double-decker trains are listed. We have chosen to let these values differ across instances, since Table 3.2 revealed sharp differences in the allocation of sweet and sour work over the different regions of the Netherlands. In particular, we use a relatively high threshold value of 0.30 for aggression in `Center`, whereas the bound for double-decker work is highest at 0.50 in `South-West`. We use a higher average duty length in `South-West` and `East`, since here the ratio of tasks to templates is slightly higher than in `Center`. Generally, we have chosen bounds close to the average attribute scores of the work of each instance, but not too close as to become unattainable. We apply the same bounds to all crew members of an instance. In practice, a railway operator might prefer the use of one set of bounds that applies to all employees. Moreover, in a real-life setting, bounds would have to be based on historical data.

Table 3.3: Sharing-Sweet-and-Sour rules of the three instances.

Instance	Duty length (h, $\leq$)	Type-A ($\geq$)	Aggression ($\leq$)	Double-decker ($\leq$)
`Center`	8:00	0.35	0.30	0.30
`South-West`	8:15	0.35	0.25	0.50
`East`	8:15	0.45	0.20	0.45

3.5.2 Settings

In order to apply the integrated and sequential methods described in Section 3.4, we need to choose values for three classes of parameters: (i) parameters that govern the column generation acceleration techniques, (ii) parameters that regulate the aggressiveness of the column generation algorithm and fixing heuristic, and (iii) parameters that define the penalty function. Note that, at least in theory, parameters in the first category affect the computation time but not the solution quality, whereas parameters in the second and third categories can affect both. In the first category, we use the following values. We return at most 50 columns per pricing problem, greedily select columns based on reduced cost that are at least 50% pair-wise task-disjoint from all previously selected columns, and in each iteration we remove columns from

the RMP whose reduced cost has been at least 250 in the last 5 iterations. In the second category, we choose to terminate the column generation process when the relative decrease in objective value is below $\epsilon = 0.01\%$, and we fix all duties whose solution value is at least $\tau = 0.55$. We justify the choice for these parameters with a sensitivity analysis in Section 3.5.7. Finally, for the penalty function we use a scaling constant $\alpha = 1{,}000$, a margin threshold $m = 0.1$, and a margin scaling factor $\gamma = 0.5$, as this combination of parameters appeared to give stable performance in preliminary experiments. All experiments were conducted using CPLEX 20.1.0 and four cores of an Intel Xeon Gold 6130 processor.

3.5.3 Satisfaction of Individual Sharing-Sweet-and-Sour Rules

We now analyse the practical performance of the integrated approach, i.e., to what extent it is able to construct rosters that satisfy the individual SS&S rules. Table 3.4 shows, for each combination of instance and method, the percentage of crew members whose total work history, as evaluated on the last day of the planning period, satisfies each of the four SS&S rules. We will refer to this value as the satisfaction level. The results for the sequential method without reduction in maximum duty length are presented in Table C1 of Appendix 3.C.

Table 3.4: Satisfaction levels (%) of the sequential (S) and integrated (I) approaches using a one-day optimisation horizon.

Instance	Method	Duty length	Type-A	Aggression	Double-decker
`Center`	S	69.9	86.6	89.1	89.3
	I	99.1	98.0	97.6	96.9
`South-West`	S	87.9	87.7	93.3	87.5
	I	83.2	91.8	95.8	88.1
`East`	S	92.9	93.1	99.1	92.0
	I	96.9	97.8	98.9	98.4

We find that the integrated approach yields high satisfaction levels on nearly all instances and attributes, attaining scores close to 100% on `Center` and `East`. The scores on instance `South-West` are slightly lower, with for example a satisfaction level of 88.1% on the double-decker attribute. This can be attributed to the fact that crew base Vlissingen (Vs) is reachable through double-decker trains only (see Table 3.2). Moreover, we observe that the satisfaction levels of the sequential method are well below those of the integrated approach. Due to the heterogeneity in the

individual template-based rosters, satisyfing SS&S rules at the crew base level is not a sufficient condition for ensuring satisfaction at the individual level. Instead, it is beneficial to tailor the construction of duties to the needs of individual crew members. The differences are particularly large for attributes at instances featuring crew bases whose realised work packages (as shown in Table 3.2) have attribute scores close to the instances thresholds. Examples include the Type-A and aggression attribute in `Center` or the double-decker attribute in `East`. For these instances it is relatively hard to obtain high satisfaction levels at all crew bases, and the benefits of the tailored integrated approach become more pronounced. Finally, we note that, on all instances and for both approaches, our heuristic leaves less than 1% of the tasks uncovered. Additional experiments indicated that this is largely caused by a mismatch between the templates derived from the cyclical rosters and the true need for capacity during the planning period. In practice, such uncovered tasks can be assigned to reserve crew members that are stand-by on the day of operations.

Additional computations show that violations of SS&S rules in the integrated approach are not clustered among crew members, i.e., there is no large group of crew members violating multiple rules. We find that, averaged over all three instances, 82.8% of the crew members satisfies each of the four SS&S rules. For 14.3% and 2.7% of the crew members we observe one or two rules being violated, respectively. It is for only 0.2% of the crew that three rules are simultaneously violated, and for all these crew members it holds that they performed less than ten duties over the planning period. In other words, there is little room for the feedback mechanism to provide them with a fair allocation of work. It never occurs that a crew member violates all four rules.

3.5.4 Detailed Results of `Center` Instance

We further illustrate the power of the integrated approach, as compared to the sequential one, by analysing in detail the characteristics of the solutions obtained on the `Center` instance. We choose this instance since (i) it contains the two largest crew bases in the Netherlands, Amsterdam and Utrecht, and (ii) the results in Table 3.4 showed a strong performance gap between the sequential and integrated approaches on this instance. In the following, we first show how the integrated approach is able to shift the distribution of the attribute scores of crew members towards the desired side of the threshold. We then show that the integrated approach converges to high satisfaction levels already halfway in the planning period. Third, we explore the

effect of the penalty mechanism on the work of individual crew members in detail and show that it is able to rapidly correct violations of SS&S rules. Finally, we study the impact of a non-empty work history on the temporal evolution of the satisfaction levels.

Figure 3.5 shows how the SS&S scores of the rosters of the `Center` instance, obtained using either the sequential or the integrated approach, are distributed over the crew members. Here, the attribute score and the number of crew members are displayed on the horizontal and vertical axes, respectively. The values of the three different crew bases are plotted in different colours. The threshold of each rule is indicated by a dashed line. We find that the distributions obtained using the integrated approach are located on the desired side of the boundaries and display a sharp peak right next to those boundaries. This is a direct result of the piece-wise structure of our penalty function. In contrast, the rosters obtained using the sequential approach are distributed on both sides of the threshold, illustrating the lower satisfaction levels of this method. Moreover, the distributions of the integrated approach display a lower degree of inequality between crew bases. This is nicely illustrated by the Type-A scores: while Amersfoort enjoys high Type-A scores compared to Amsterdam in the sequential approach (recall Table 3.2), the distribution of their scores are highly comparable in the integrated approach. Finally, as can be observed from the narrower range of the distributions, the inequality between individuals of the same crew base also tends to decrease when using the integrated approach.

Second, we study how fast the satisfaction levels of the sequential and integrated approaches converge to acceptable levels. Figure 3.6 shows the temporal evolution of satisfaction levels of the rosters obtained using both methods. The days in the planning period are listed on the horizontal axis, whereas the satisfaction levels of each of the four SS&S rules are shown on the vertical axis. To better highlight the temporal evolution, these satisfaction levels are computed based on the scores of the last 15 duties performed by each crew member. The scores are based on fewer duties in case a crew member has not yet performed 15 days of work. In both approaches, the satisfaction levels start off low at the beginning of the planning period. Since we do not initialise the rosters with any work history, a single sour duty can cause a roster to violate rules for multiple days. This effect is more pronounced for the type-A attribute than the double-decker or aggression attribute, as the proportion of type-A work in the `Center` instance is much closer to its threshold value (see Tables 3.2-3.3). Halfway through the planning period, however, all satisfaction levels of the integrated

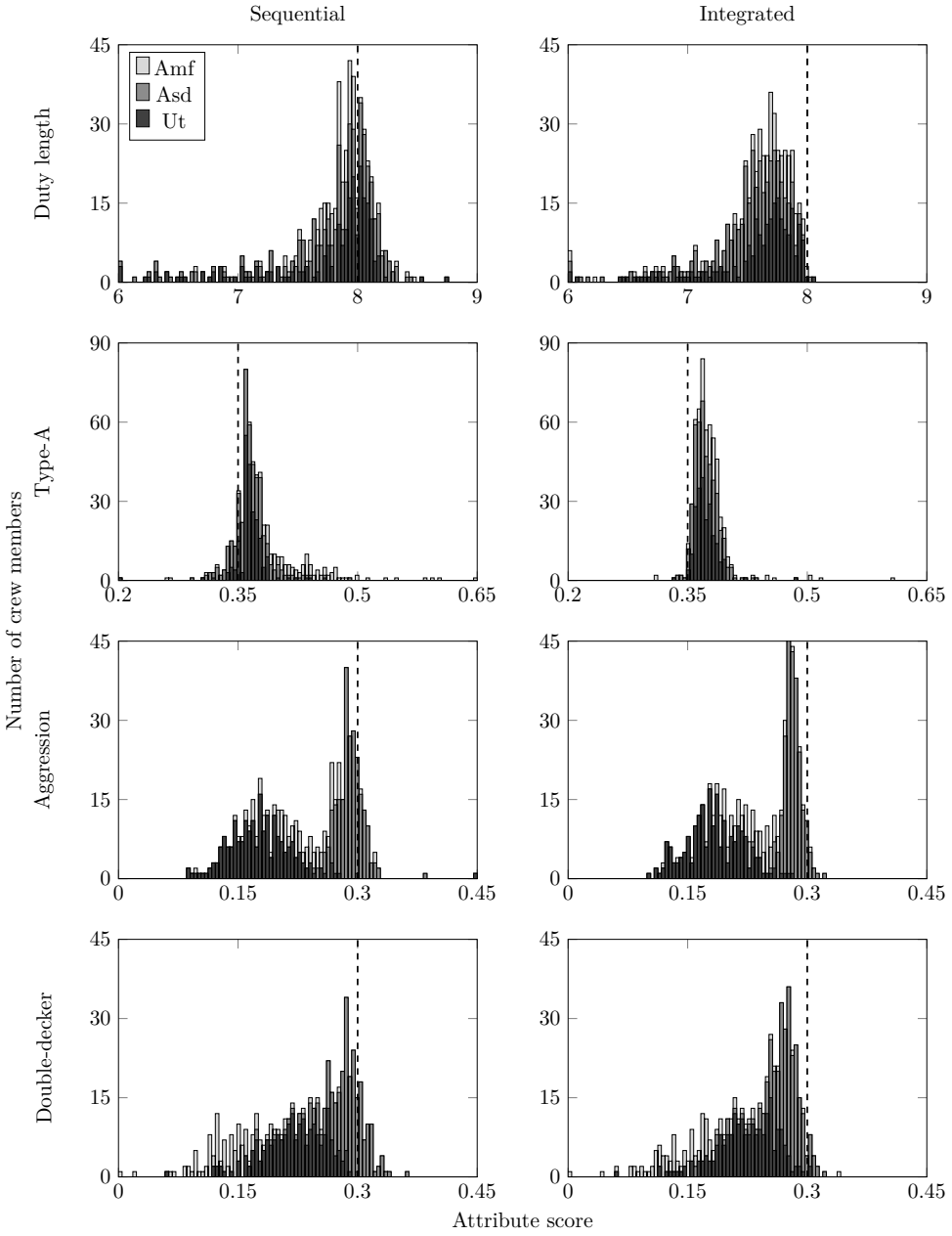


Figure 3.5: Distribution of Sharing-Sweet-and-Sour attribute scores over crew members of crew bases Amersfoort (Amf), Amsterdam (Asd), and Utrecht (Ut) of the sequential and integrated approaches on instance `Center`.

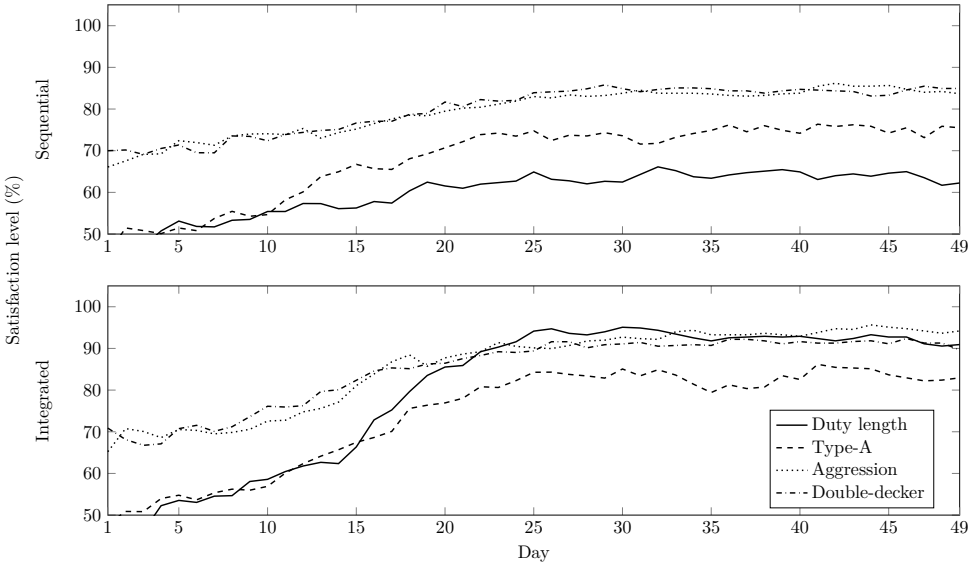


Figure 3.6: Temporal evolution of satisfaction levels of the sequential and integrated approaches on instance `Center`. Satisfaction levels are based on scores computed as 15-duty moving averages.

approach have reached values above 80%, after which they slowly rise to near 90% and flatten out near the end of the planning period. This last property is desirable in practice as it shows that stable satisfaction levels can be attained. The satisfaction level of Type-A work slightly lags behind the other attributes. This is most likely caused by the fact that the large crew bases Amsterdam and Utrecht naturally do not perform much Type-A work (see Table 3.2), and more time is needed to correct for this initial imbalance. The sequential approach fails to deliver compared to the integrated approach, since its satisfaction levels increase at a slower rate and plateau at a lower level.

Third, we analyse the attribute scores of several individual rosters to gain a better understanding of the effect of penalties in the integrated approach. Figure 3.7 shows, for the SS&S attributes Type-A and aggression, the evolution of the scores of the rosters of crew members #148 and #455 who work at crew bases Amersfoort and Amsterdam, respectively. The day in the planning period is listed on the horizontal axis and the attribute score on the vertical axis. Once again, these scores are computed as moving averages of the last 15 duties. The solid line indicates the threshold of the respective attribute. Finally, we highlighted periods of interest, to be discussed

in the next paragraph, in grey.

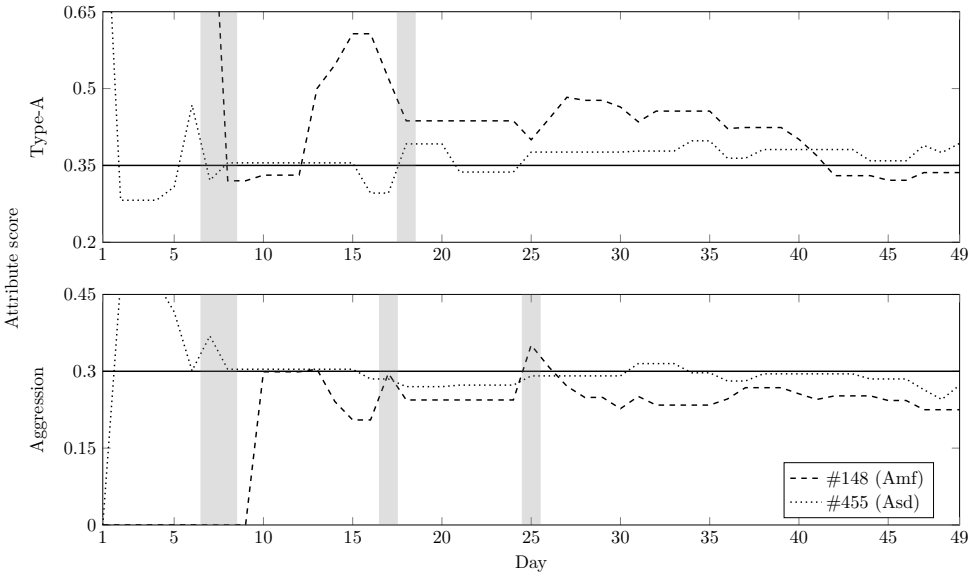


Figure 3.7: Temporal evolution of individual Sharing-Sweet-and-Sour scores of the integrated approach on instance `Center`. Scores are computed as 15-duty moving averages.

We find that the scores heavily fluctuate at the start of the planning period and slowly dampen towards the end of the planning period, a logical result of the fact that attribute scores are averaged over a larger number of duties. When the attribute score of a crew member exceeds the threshold, we find that, usually already in the next duty, this score is driven towards the desired side of the boundary. Examples include the Type-A score of #455 on days 7 and 18, and the aggression score of #148 on day 25, respectively. This shows that the penalties used in the integrated approach are an effective way of steering towards satisfaction of SS&S rules. We also find that our method is able to steer on multiple attributes simultaneously, see for example the development in Type-A and aggression scores of crew member #455 from day 7 to day 8. In addition, the margin in our penalty function successfully prevents attribute scores from crossing the thresholds, as illustrated by the aggression score of guard #148 on day 17. Finally, we note that differences between individuals do persist throughout the planning period, reflecting the initial imbalance across crew bases. Given our choice of penalty function, there is no mechanism in our algorithm that actively corrects such differences.

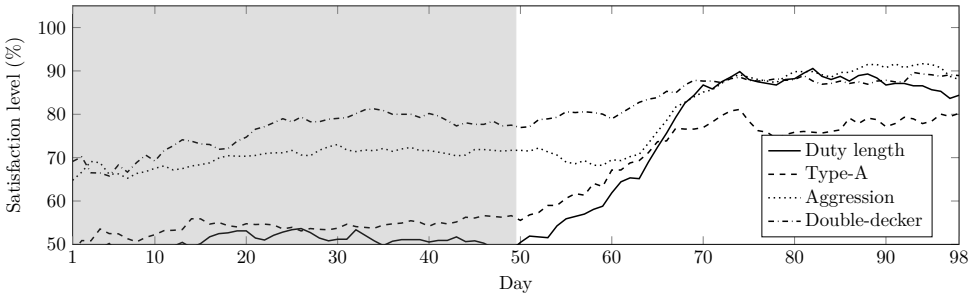


Figure 3.8: Temporal evolution of satisfaction levels of the integrated approach on instance `Center` with a 49-day work history initialisation period. Satisfaction levels are based on scores computed as 15-duty moving averages.

Finally, we study the effect of the initial work history, which up to now has been assumed to be empty, on the temporal performance of the integrated approach. We initialise the work history of each guard by performing a full run over our planning horizon without performing any steering towards SS&S rules, i.e., without the use of the penalty-based feedback mechanism. Starting with this history, we then apply the integrated approach to conduct another run over the planning horizon. Figure 3.8 shows the temporal evolution of the resulting satisfaction levels throughout this 98-day planning period, where the interpretation of the values and axes is identical to that of Figure 3.6. The 49-day initialisation period, in which no penalties are used, is highlighted in grey. The results indicate that the integrated approach is insensitive to the initial work history. The satisfaction levels display only a slight upward trend in the initialisation period, and rapidly increase and converge to high levels when the penalty-based feedback mechanism is activated. Interestingly, the satisfaction levels on day 49, the last day of the initialisation period, are highly similar to the levels on day 1 in Figure 3.6. This could be attributed to the fact that, in both cases, the work histories are obtained through what is essentially a random assignment over crew members.

3.5.5 Effect of Two-Day Optimisation Horizon

We now analyse the potential benefits of increasing the optimisation horizon, i.e., the number of days included in each crew planning problem, from one to two days. After solving the two-day problem, the first-day duties in the solution are fixed in the crew plan and we proceed to the next day. We only consider the integrated approach, as it has shown to outperform the sequential approach in all dimensions.

Table 3.5 shows, for each combination of instance and horizon, the percentage of crew members satisfying each of the four SS&S rules. The results show that there is no apparent benefit in enlarging the optimisation horizon from one to two days, as the satisfaction levels are not significantly affected. This might indicate that, due to the scale of the instances, there is sufficient capacity and flexibility in the rosters, and explicitly incorporating look-ahead information about roster rules therefore does not yield any improvements. Alternatively, the performance of the diving heuristic might deteriorate when applied to two-day instances. Given that computation times are significantly higher when using a two-day horizon, as we will show in Section 3.5.6, we conclude that it is preferable to use a one-day optimisation horizon.

Table 3.5: Satisfaction levels (%) of the integrated approach using an optimisation horizon of one versus two days.

Instance	Horizon	Duty length	Type-A	Aggression	Double-decker
`Center`	1	99.1	98.0	97.6	96.9
	2	98.1	96.7	98.5	95.5
`South-West`	1	83.2	91.8	95.8	88.1
	2	84.4	92.6	94.2	89.6
`East`	1	96.9	97.8	98.9	98.4
	2	97.3	97.5	98.0	98.2

3.5.6 Computational Performance

In this section we analyse the computational performance of our methods to determine whether they are suitable for use in an operational setting. Table 3.6 shows the computational performance of the sequential approach with a one-day optimisation horizon, and the integrated approach with a one- or two-day horizon, on each of the three instances. For each combination of instance, method, and horizon, the average computation time per day of the planning period is shown. This is decomposed into time spent solving pricing problems, solving the RMP, adding and removing columns to and from the RMP, and exchanging duties between crew members (i.e., the second step of the sequential method). In addition, the average number of column generation iterations and number of iterations in which columns are fixed are shown.

The computation times of the integrated approach with a one-day horizon, which proved to be the preferable configuration, are more than reasonable for use in a daily, operational setting. Even on the largest instance `South-West`, the average

Table 3.6: Computational performance of the sequential (S) and integrated (I) approaches. The values are averaged over all days of the planning period.

			Time (s)					Iterations	
Instance	Method	Horizon	Total	Pricing	RMP	Columns	Exchange	Total	Fixing
	S	1	534.4	46.8	477.2	10.4	0.1	106.8	13.4
`Center`	I	1	228.0	39.2	182.8	6.0	-	111.3	12.0
	I	2	1,333.6	177.8	1,135.0	20.8	-	109.8	11.5
	S	1	1,680.7	138.0	1,515.8	26.9	0.1	173.2	12.9
`South-West`	I	1	452.2	94.6	347.7	9.9	-	165.7	13.8
	I	2	1,714.8	282.8	1,404.0	28.0	-	169.2	12.5
	S	1	492.2	36.7	446.8	8.7	0.1	91.3	12.0
`East`	I	1	145.5	28.7	112.6	4.2	-	94.5	11.1
	I	2	569.7	75.7	483.0	11.0	-	85.4	9.8

total computation time per day does not exceed eight minutes. By far the largest share of computation time is required to solve the RMP, followed at a distance by the time needed to solve pricing problems. Integrating the duty generation and assignment does not come at a computational cost, since the integrated method is significantly faster than the sequential approach. This is fully driven by the additional reduction in maximum duty length, however: the running times of the sequential approach without this additional reduction are highly similar to those of the integrated approach (see Table C2 of Appendix 3.C). As expected we find that performing the post-optimisation step in which duties are exchanged requires little computation time. Finally, we observe that computation times of the integrated approach increase drastically when the optimisation horizon is enlarged. This effect is mainly driven by an increase in the RMP time per iteration, as the total number of iterations stays approximately constant.

Our computational experiments suggest that the integrated and sequential approaches are comparable in terms of scalability when using the same maximum duty length. We now conduct some additional experiments to confirm whether this is indeed the case. In particular, we apply both approaches to the set of instances obtained by taking all possible subsets of crew bases of the `South-West` instance. Here, we have treated crew bases Rsd and Vs as one, since individually these would be too small to yield meaningful instances. This yields a total of seven instances, the original `South-West` instance being the largest, and hence 343 different daily planning problems.

Figure 3.9 displays for each method and each daily planning problem, the number

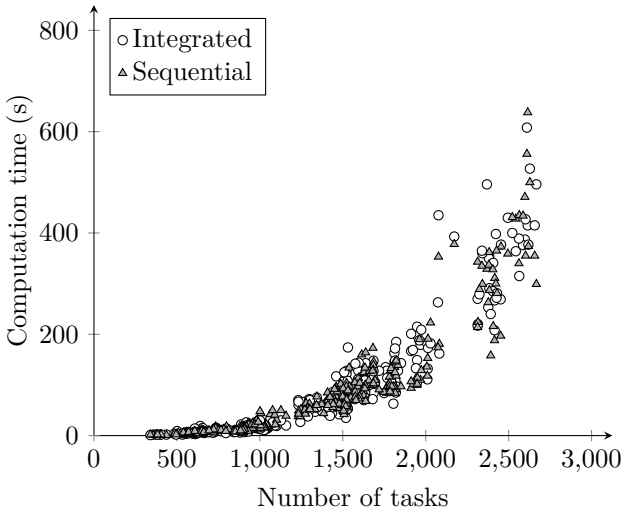


Figure 3.9: Effect of number of tasks per day on computation times of the integrated and sequential approach without reduction in maximum duty length.

of tasks in the instance and the computation time (in seconds) required to solve it. We clearly observe the exponential growth in computation time typically encountered in crew planning problems. Moreover, we find that the integrated and sequential method display highly similar scaling behaviour. Since a variant of the crew scheduling algorithm used as part of the sequential method is currently in use at Netherlands Railways, one can reasonably expect the integrated approach to be a practically viable option there as well. In addition, the `South-West` instance is already large compared to what is typically studied in the crew scheduling literature (Heil et al., 2020).

3.5.7 Effect of Column Generation Parameters

In order to cope with the magnitude of our crew planning problems, we have used a rather aggressive configuration of our column generation heuristic. In particular, we used a fixing threshold of $\tau = 0.55$, and have terminated column generation iterations when the improvement in objective value between consecutive iterations was below $\epsilon = 0.01\%$. In this section we experiment with varying these parameters to analyse the trade-off between computation time and solution quality. Figure 3.10 shows, for various combinations of τ and ϵ, the average satisfaction levels and average total computation time per day of the planning period when applying the integrated approach

with a one-day optimisation horizon on the `Center` instance. Here, parameters τ and ϵ refer to the fixing threshold and threshold on improvement in objective value, respectively.

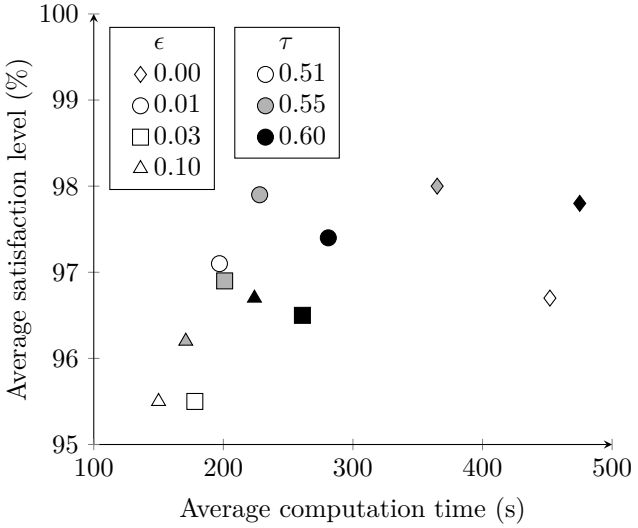


Figure 3.10: Effect of column generation parameter on satisfaction levels and computation times of the integrated approach on instance `Center`. Satisfaction levels are averaged over all four attributes, and computation times are averaged over all days of the planning period.

Prematurely terminating the column generation iterations, i.e., using any $\epsilon > 0$, drastically reduces the computation times of the algorithm. In particular, computation times are more than halved when switching from $\epsilon = 0.00$ to $\epsilon = 0.10$. While increasing ϵ from 0.00 to 0.01 does not seem to have a strong effect on the satisfaction levels, we find that further increasing ϵ does reduce the solution quality. The effect of the fixing threshold τ appears to be less pronounced, and while a higher threshold generally increases the computation time, it does not necessarily lead to a higher solution quality. To conclude, our current parameter setting $(\tau, \epsilon) = (0.55, 0.01)$ is on the Pareto front and adequately balances solution quality and computation time.

3.6 Conclusion

In this chapter, we consider fairness over time in a large-scale public transport application. In particular, we introduce the railway crew planning problem with fairness over time. Here, the goal is to assign duties to crew members over the course of

a planning period, such that all tasks are covered and the total work of each crew member satisfies all Sharing-Sweet-and-Sour rules. We propose sequential and integrated rolling horizon approaches to solve the problem, and evaluate both methods on three large real-life instances from NS, the largest passenger railway operator of the Netherlands, spanning a seven-week planning period. The integrated approach is able to achieve high satisfaction levels and significantly outperforms the sequential approach at negligible additional computational cost. The integrated approach is able to accurately and rapidly steer towards a fair allocation of attractive and unattractive work, and converges to high satisfaction levels already halfway in the planning period.

Our approach has high potential for passenger railway operators, as it shows that tight guarantees on the attractiveness of individual work schedules can be made. The set-up we consider can be modified in several interesting directions. First, our approach allows for differential preferences across crew bases or even crew members. Our method can be used to evaluate the impact of using one set of nationwide bounds, instead of the regional bounds, which reflect the geographical distribution of sweet and sour tasks, used in this chapter. In addition, one could consider a setting where employees agree to relax Sharing-Sweet-and-Sour rules in return for a monetary compensation. Second, one could evaluate the Sharing-Sweet-and-Sour rules on a rolling window basis instead of over a fixed planning period. Given the stable temporal performance of the integrated method, this should be attainable with our approach. Third, while this chapter focuses on guards, there exist Sharing-Sweet-and-Sour attributes, such as the number of unique traversed kilometers, that are of particular interest to drivers. NS is currently conducting experiments in this direction, showing that our method can be succesfully applied here with some minor modifications. Finally, as our approach is situated in the operational planning phase, it allows for dynamic classifications of attractive and unattractive work.

The work in this chapter raises some relevant practical questions. While we assume that individual template-based rosters are given, it is interesting to investigate how one can construct rosters that provide capacity at the right time and location without access to detailed information about the content of work to be performed. Chapter 2 shines some light on this question. Moreover, we have not yet explored the efficiency gains arising from the fact that the operator can determine the exact work content closer to the day of operations. Next, while in this chapter our only goal is to ensure that all individuals satisfy a certain rule set, it is interesting to what extent a penalty-

based approach is able to reduce the remaining variation between crew members. Finally, it is important to analyse the effect of disruptions, i.e., discrepancies between the planned schedule and the operated schedule, on the ability to satisfy Sharing-Sweet-and-Sour rules. We expect that our penalty mechanism largely mitigates this effect.

While our work shows the high potential of the penalty mechanism, more sophisticated solution methods are required to apply this approach to larger instances with more crew bases, e.g., on the full network of Netherlands Railways. One promising option is to use a modified version of the powerful Lagrangian-based solver currently used by NS (Abbink et al., 2011). Alternatively, one could consider geographical decomposition techniques (Jütte and Thonemann, 2012).

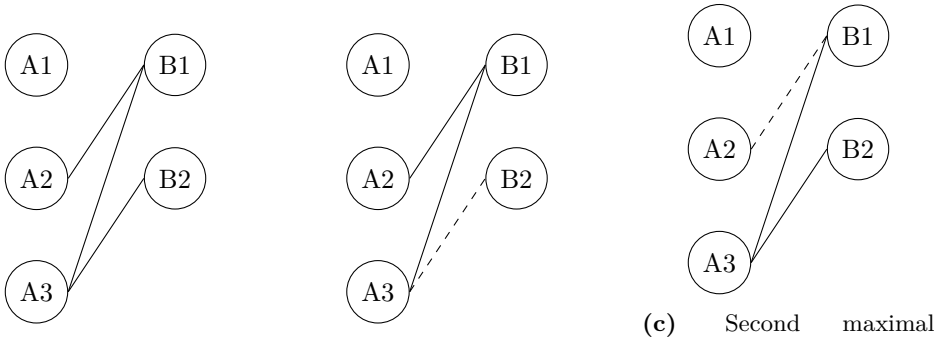
Appendix

3.A Coupling Constraints for the Multi-Day Crew Planning Problem

We start by introducing the roster rotation constraints. When the start times of two consecutive early templates are non-decreasing, we require that the start times of the duties assigned to these templates are non-decreasing too. Here, early templates are defined as those starting before 10:00. We say that a duty pattern satisfying this rule is *forward rotating*. Similarly, we require that the end times of duties in late templates are non-increasing when the end times of these templates are non-increasing. Here, late templates are defined as those ending after 20:00, and a duty pattern satisfying this constraint is called *backward rotating*. The two rotation constraints ensure that the rosters align with the employees' circadian rhythms.

We now show how model (3.1) can be extended to multiple days. In such a setting the roster rules, i.e., the minimum rest time and rotation constraints, act as coupling constraints between consecutive days. Following Breugem et al. (2022a) we model these rules using clique inequalities. Such inequalities are obtaining by enumerating maximal bicliques in the bipartite violation graph connecting duties whose joint selection would yield a constraint violation. Here, a maximal biclique is defined as a clique in a bipartite graph that is not fully contained in any other clique. Let P be the set of pairs of consecutive days in the crew planning problem, and denote by

R_p the set of crew members working on both days of pair $p \in P$. Finally, denote by Q_p^r the set of clique constraints linking days in p for crew member r, and let binary parameter $o_{rpt}^{\delta q}$ indicate whether duty $\delta \in \Delta_t^r$, for $t \in p$, appears in constraint $q \in Q_p^r$.



(a) Violation graph. **(b)** First maximal biclique. **(c)** Second maximal biclique.

Figure A1: Modelling the forward rotation constraint using maximal bicliques.

Figure A1 presents an example of how maximal bicliques are obtained from the violation graph corresponding to the forward rotation constraints. We assume that the duty of a given crew member starts with either task A1, A2, or A3 on the first day, and with task B1 or B2 on the consecutive day. Moreover, we assume that all tasks start before 10:00. The bipartite violation graph corresponding to the forward rotation constraint of this crew member, as shown in Figure A1(a), connects start tasks of the first day with all earlier start tasks of the second day. This graph contains two maximal bicliques, shown in Figures A1(b) and A1(c), respectively. Each such biclique yields a clique inequality stating that at most one duty with a start task in the clique is assigned to this crew member.

Let T be set of days in the multi-day problem. The formulation of the multi-day crew planning problem now reads:

$$\min \quad \sum_{t \in T} \sum_{r \in R_t} \sum_{\delta \in \Delta_t^r} c_{rt}^{\delta} x_{rt}^{\delta} \tag{3.7a}$$

$$\text{s.t.} \quad \sum_{r \in R_t} \sum_{\delta \in \Delta_t^r} a_{tk}^{\delta} x_{rt}^{\delta} \geq 1 \qquad \forall t \in T, k \in K_t \tag{3.7b}$$

$$\sum_{\delta \in \Delta_t^r} x_{rt}^{\delta} = 1 \qquad \forall t \in T, r \in R_t \tag{3.7c}$$

$$\sum_{t \in p} \sum_{\delta \in \Delta_t^r} o_{rpt}^{\delta q} x_{rt}^{\delta} \leq 1 \qquad \forall p \in P, r \in R_p, q \in Q_p^r \tag{3.7d}$$

$$x_{rt}^{\delta} \in \mathbb{B} \qquad \forall t \in T, r \in R_t, \delta \in \Delta_t^r. \tag{3.7e}$$

Here, the clique inequalities are given by constraints (3.7d). These constraints are easily amendable to the pricing problem, as their duals can be placed on source or sink arcs of tasks in the bicliques. In the multi-day problem we use the same diving rule as before, i.e., we fix duties on the first day of the planning problem until the first-day solution is completely binary.

3.B Pricing Problem

We solve a single pricing problem for each template. This problem can be modelled as a RCSPP on a suitably defined pricing graph. This graph contains a source and sink node, representing departure from and arrival at the depot, respectively. Moreover, it contains a node for each task, as well as connections between each pair of nodes that can be performed consecutively. The graph is trimmed to include only those tasks and connections that fall within the time window specified by the template. Figure B1 shows an example of such a pricing graph.

The reduced cost of a duty can be decomposed over the arcs in this graph. The dual variable corresponding to assignment constraint (3.1c) can be placed on all arcs leaving the source, while the dual variables corresponding to constraints (3.1b) can be placed on the incoming arcs of the respective task nodes. Sections 3.4.4 and 3.4.5 describe how the penalty coefficients and duals of Sharing-Sweet-and-Sour constraints are integrated in the pricing, respectively.

The pricing problem is now modelled as a RCSPP on the directed acyclic pricing graph, with resources for the presence of a meal break, the time without meal break, and the current duty duration. The first resource must be at least one at the sink node, the second resource may never exceed 5.5 hours at any partial path, and the third resource may not exceed the maximum duty length. As in Breugem et al. (2022a), we solve this problem using a heuristic labelling algorithm. In particular, we use a heuristic dominance rule based on a completion bound on reduced cost. In addition, we store at most 100 labels simultaneously, and process labels in increasing order of completion bound.

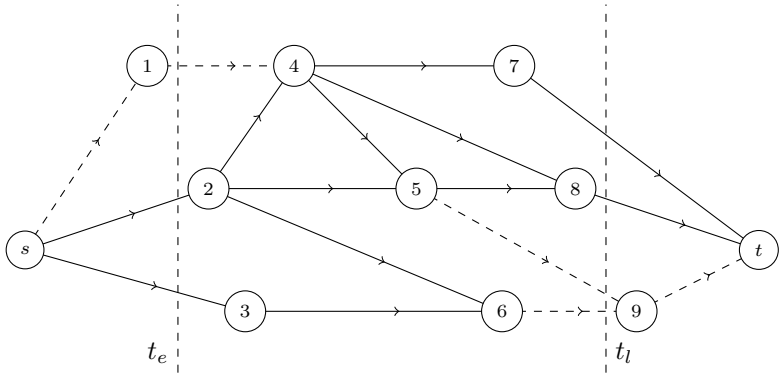


Figure B1: Pricing graph corresponding to a template with earliest start time t_e and latest end time t_l. Each node corresponds to a task and each arc represents a feasible connection between two tasks. In this graph, a total of nine tasks are shown. The arcs to and from tasks 1 are 9 are dashed, representing the fact that these tasks fall outside the time window specified by the template.

3.C Results Sequential Method Without Reduction in Maximum Duty Length

Table C1: Satisfaction levels (%) of the sequential approach without reduction in maximum duty length using a one-day optimisation horizon.

Instance	Duty length	Type-A	Aggression	Double-decker
`Center`	64.6	85.1	86.6	88.0
`South-West`	74.6	90.0	91.9	88.2
`East`	87.7	92.8	97.3	93.5

Table C2: Computational performance of the sequential approach without reduction in maximum duty length using a one-day optimisation horizon. The values are averaged over all days of the planning period.

	Time (s)					Iterations	
Instance	Total	Pricing	RMP	Columns	Exchange	Total	Fixing
`Center`	217.2	68.1	143.6	5.3	0.2	103.7	14.1
`South-West`	447.7	153.0	284.3	10.3	0.1	178.6	15.0
`East`	123.0	33.7	85.6	3.6	0.1	93.6	13.5

Chapter 4

A Fast Exact Pricing Algorithm for the Railway Crew Scheduling Problem

This chapter is based on B.T.C. van Rossum (2022a). 'A fast exact pricing algorithm for the railway crew scheduling problem'. Operations Research Letters *50.6, pp. 707–711.*

4.1 Introduction

The goal of the railway crew scheduling problem is to select a least cost set of duties, i.e., days of work, such that all tasks in the timetable are covered. Large-scale crew scheduling problems, as encountered by railway operators, are typically solved with column generation approaches (Heil et al., 2020). Here, the problem is initialised with a restricted set of duties, and in each iteration a pricing problem is solved to identify duties that can potentially improve the solution, if any. The pricing problem can be modelled as a shortest path problem with resource constraints when the duty constraints consist of the following requirements: a maximum duty length, a meal break constraint, and the requirement to start and end at the same crew depot.

For an instance with $|N|$ tasks, the fastest known exact algorithm has time complexity $\mathcal{O}(|N|^3)$ (Huisman, 2007). In this chapter, we propose an $\mathcal{O}(|N|^2)$ algorithm for the pricing problem. We attain this complexity by using a linearly sized time-space network, exploiting the topological ordering of the resulting network, and making efficient use of array data structures. We perform computational experiments on randomly generated instances, and show that our proposed algorithm is 95% faster than the benchmark on instances of size $|N| = 1,250$.

The remainder of this chapter is structured as follows. In Section 4.2, we formulate the crew scheduling problem and show a typical column generation approach to this problem. We show how to construct a linearly sized time-space network in Section 4.3. In Section 4.4, we describe the proposed $\mathcal{O}(|N|^2)$ algorithm, and in Section 4.5, we discuss the computational experiments. We conclude in Section 4.6.

4.2 Problem Description

We consider the following basic railway crew scheduling problem. A set of tasks $i \in N$ is given, constituting the total work that needs to be performed. This work can consist in either driving a train or assisting passengers, depending on whether we are scheduling drivers or guards. Each task i is characterised by start and end stations s_s^i and s_e^i and start and end times t_s^i and t_e^i. For example, driving the train that departs from Rotterdam at 11:11 and arrives in Amsterdam at 11:52 could constitute a task. Each crew depot $d \in D$ has an unlimited number of crew members that can perform tasks. We assume that each depot is also a start or end station. Each task should be performed by at least one crew member. Tasks are assigned

to crew members in the form of duties. A *duty* is a day of work constituted by a sequence of tasks that can be performed consecutively. Task $j \in N$ can be performed after $i \in N$ when $s_e^i = s_s^j$ and $t_e^i < t_s^j$, i.e., when j starts at the end station of i and j starts strictly later than i ends.

Each duty must also satisfy several labour constraints. First, each duty must start and end at the same depot. Second, the duty length, defined as the difference in end time of the last task and start time of the first task, must not exceed the maximum duty length W. Third, each duty must contain exactly one meal break. A meal break can be scheduled between consecutive *pre-break task* j and *post-break task* k when $t_e^j + B < t_s^k$, where B is the minimum break duration. The duty length without break, i.e., the duration from start task to pre-break task and from post-break task to end task, must not exceed the maximum duration without break L. In practice, a typical duty contains about ten tasks and lasts about eight hours.

The cost of a duty is the sum of a fixed cost c_F and a variable cost equal to c_V times the duty length. The goal of the crew scheduling problem (CSP) is to find a minimum cost set of duties such that each task is covered by at least one duty.

The CSP is commonly formulated as a set covering problem. Let Δ be the set of feasible duties, and let c_δ equal the cost of duty $\delta \in \Delta$. For each task $i \in N$, the binary parameter $a_{i\delta}$ indicates whether task i is covered by duty δ. We introduce the binary variable x_δ, which equals one when duty δ is selected. The CSP can then be formulated as follows:

$$
\begin{aligned}
(\text{CSP}) \quad \min \quad & \sum_{\delta \in \Delta} c_\delta x_\delta && && (4.1a)\\
\text{s.t.} \quad & \sum_{\delta \in \Delta} a_{i\delta} x_\delta \geq 1 && \forall i \in N && (4.1b)\\
& x_\delta \in \{0,1\} && \forall \delta \in \Delta. && (4.1c)
\end{aligned}
$$

Since the set of feasible duties can be huge, the above formulation contains too many variables to be useful in practice. Instead, researchers and practitioners typically solve the LP-relaxation of (4.1) using a column generation algorithm. Here, the CSP is initialised with a proper subset $\Delta' \subset \Delta$ of the duties, and each binary variable is relaxed to a non-negative one. We refer to the resulting model, not containing the full set of feasible duties, as the restricted master problem (RMP). In each iteration of the algorithm, the RMP is solved to optimality. Then, a pricing problem (PP) is

solved using a pricing algorithm, as will be explained later, to determine whether any column, i.e., duty, with negative reduced cost exists. If so, one or multiple negative reduced cost columns are added to the RMP, and the algorithm continues. If not, the current solution to the RMP is also an optimal solution to the LP-relaxation of (4.1), and the algorithm terminates. Column generation can be embedded in a branch-and-bound framework to obtain integer solutions, in which case we refer to the resulting algorithm as branch and price (Barnhart et al., 1998).

The reduced cost of a column can be computed as follows. Let λ denote the dual vector of constraints (4.1b). The reduced cost of x_δ is then given by

$$\mathrm{RC}(x_\delta) = c_\delta - \sum_{i \in N} a_{i\delta} \lambda_i. \tag{4.2}$$

The PP consists of finding the duty with minimum reduced cost, and can be formulated as a shortest path problem with resource constraints (SPPRC) on a suitably defined graph (see Section 4.3). Note that any feasible duty corresponds to a path of tasks, from start task to end task, that are performed consecutively, such that the duty length, meal break, and depot constraints are satisfied. These constraints can be modelled by including resources. The reduced cost can be decomposed on the arcs by placing c_F on all source arcs, $-\lambda_i$ on the source and task arcs corresponding to task i, and c_V times the duration of an arc on each arc in the network. The problem of finding the duty with minimum reduced cost is then equivalent to that of finding the shortest s, t-path satisyfing the resource constraints.

4.3 Time-Space Network

We have seen that the pricing problem can be formulated as a SPPRC on a suitably chosen network. In Section 4.3.1, we show how a time-space network can be constructed whose size, measured by the number of nodes and arcs, is linear in the number of tasks and which still allows all feasible duties to be represented as paths in the network. In Section 4.3.2, we discuss how we can solve shortest path problems on such a network in linear time, and how several duty constraints can be easily incorporated.

4.3.1 Network Construction

We construct a directed time-space network $G = (V, A)$, in which each node corresponds to a combination of time and station. In particular, we add a node for each task $i \in N$, representing the end time and end station of this task. Moreover, we add a source node s and sink node t, representing the departure from and arrival at the depot, respectively. The cardinality of the node set equals $|V| = |N| + 2 = \mathcal{O}(|N|)$. We assume that the node list V is topologically sorted by increasing end time, with the source and sink node naturally being the first and last in this ordering, respectively.

The time-space network contains four different types of arcs: source arcs, sink arcs, waiting arcs, and task arcs. A source arc connects the source s with a task i, and we add such an arc to each task that starts at one of the depots. We refer to such tasks as *start tasks*. Similarly, a sink arc connects a task i with the sink t, and we add such an arc from each task that ends at one of the depots. We refer to such tasks as *end tasks*. Wait arcs represent idle time at a station, and connect task nodes at the same end station with each other. We add a single waiting arc from each task i to the first next task j, as seen in the topological ordering, that ends at the same end station. Note that such a task j need not exist for each i, e.g. when there are no tasks with later end times than i. Finally, task arcs represent a task being performed. For each task $i \in N$, we construct an incoming task arc from the latest task j, as seen in the topological ordering, for which it holds that j and i can be performed consecutively. Once again, no such task j may exist for a given i, e.g. when there are no tasks with earlier start times than i. Since we construct at most four arcs for each task, the cardinality of the arc set is $|A| = \mathcal{O}(|N|)$.

The total size of the network becomes $|V| + |A| = \mathcal{O}(|N|)$. It is clear that each feasible duty can be represented by a s, t-path in the network. Figure 4.1 shows an example of a time-space network based on six tasks. The path $(s \to 1 \to 3 \to 4 \to t)$ could constitute a feasible duty, containing a waiting arc between tasks 1 and 3 and thus contributing to the cover constraints (4.1b) of tasks 1 and 4 only.

4.3.2 The Single-Source Shortest Paths Problem

Any time-space network constructed using the procedure of Section 4.3.1 is a directed acyclic graph (DAG). It is well-known that on any DAG $G = (V, A)$ we can solve the single-source shortest path problem (SSSPP) in $\mathcal{O}(|V| + |A|)$ time using dynamic

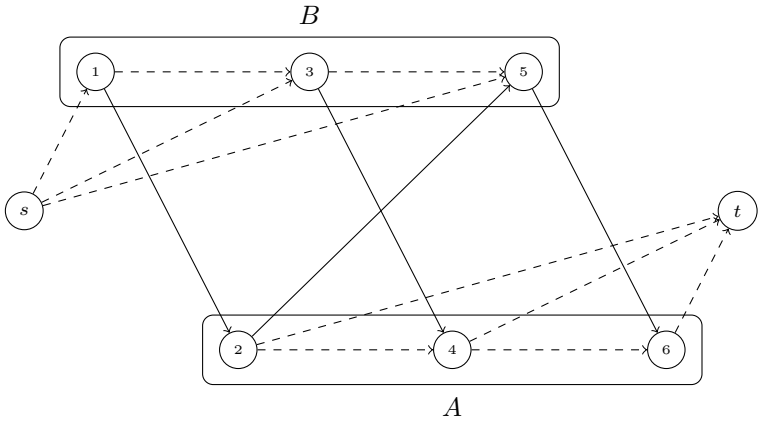


Figure 4.1: Example of time-space network based on tasks $N = \{1, \ldots, 6\}$ and depot $D = \{A\}$. All odd tasks start at A and go to B, while all even tasks start at B and go to A. Source, sink, and waiting arcs are represented by dashed lines, while task arcs are represented by solid lines. Tasks 1 and 3 do not have an incoming task arc, since no other task finishes sufficiently early to be performed before these tasks.

programming (see, e.g., Cormen et al. (1990)). This dynamic program processes all nodes of the network in topological order, starting at a specified origin node. Note that this origin node need not be the source s of the network. At each node $v \in V$, all its out-arcs are processed, and distance labels, tracking the distance of the shortest path to the origin node, of out-neighbours of v are updated. Since the time-space network is linearly sized in $|N|$, the running time of the dynamic program becomes $\mathcal{O}(|N|)$.

The structure of the dynamic program allows us to easily incorporate some duty constraints. Assume that our origin node corresponds to start task $i \in N$. The maximum duty length constraint, for example, can then be incorporated by skipping all arcs to nodes whose end time is beyond $t_s^i + W$. Similarly, we can incorporate the depot constraint by skipping all sink arcs from tasks that do not end at depot s_s^i. Note that we directly obtain an $\mathcal{O}(|N|^2)$ algorithm for the pricing problem where the meal break constraint is relaxed, by solving a SSSPP, incorporating duty length and depot constraints, from each start task.

4.4 An $\mathcal{O}(|N|^2)$ Pricing Algorithm

The feasible duty with minimum reduced cost can be fully characteristed by start, pre-break, post-break, and end tasks, which are subsequently connected by the shortest path from start to pre-break task, the waiting arc from pre-break to post-break task, and the shortest path from post-break to end task. We can directly obtain an $\mathcal{O}(|N|^4)$ pricing algorithm as follows. First, we solve a SSSPP from each node in the network. Next, we loop over all possible combinations of start, pre-break, post-break, and end tasks, checking for each combination whether a feasible duty with negative reduced cost can be constructed. Observe that, given a start and post-break task, we can independently search for the optimal pre-break and end task. This allows us to separate the loops over pre-break and end tasks, yielding a pricing algorithm with $\mathcal{O}(|N|^3)$ time complexity. This is the fastest exact pricing algorithm in literature, and is described in, e.g., Huisman (2007). Note that this algorithm generates at most $\mathcal{O}(|N|^2)$ negative reduced cost columns in each iteration.

We are able to drive down the complexity to $\mathcal{O}(|N|^2)$ using two insights. First, in a time-space network, it suffices to consider only one pre-break task for each post-break task. Hence, we only need to search across combinations of start, post-break, and end tasks. Second, by construction, the topological ordering in the time-space network also provides a non-decreasing ordering of the end times of tasks. This implies that, for each start task, the maximum duty length constraint provides a maximum allowed value for the index of the end task that can be used to complete a duty starting at this task. Once more exploiting the topological ordering of the network, we show that it is possible to efficiently compute the distances from any post-break task to an end task with at most a given index and ending at a specified depot. Since we store these values in an array for each post-break task, for a given combination of start and post-break task we can find the distance to the optimal end task in constant time. In this way, we obtain an $\mathcal{O}(|N|^2)$ pricing algorithm.

An overview of the algorithm is given in Algorithm 4.1. It starts by computing a list of all possible combinations of pre-break and post-break task using the `BreakPairs` method. This pre-processing step can be performed before entering the column generation algorithm. The algorithm proceeds by computing a distance matrix $\mathbf{D}$, containing information on the shortest paths between tasks that will be used in the final two steps of the algorithm. In the third step, the `ComputeMatrix` function is called to compute an auxiliary distance matrix $\mathbf{E}$, which contains the minimum distance

from a given post-break task to any end task that ends at a particular end station before a given time. In the final step, the function `ComputeMinimumReducedCost` takes **D** and **E** as input and returns the reduced cost of the duty with minimum reduced cost.

Algorithm 4.1 `PricingAlgorithm`

```
Compute list of break pairs BreakPairs                          ▷ Section 4.4.1
Compute distance matrix D (Section 4.4.2)
E ← ComputeMatrix(BreakPairs, D, D, N)                          ▷ Section 4.4.3
RC* ← ComputeMinimumReducedCost(BreakPairs, D, E, N)            ▷ Section 4.4.4
return RC*
```

In Sections 4.4.1-4.4.4, we describe the four steps of the algorithm in more detail and discuss their time complexity. In Section 4.4.5, we show that the time complexity of `PricingAlgorithm` is $\mathcal{O}(|N|^2)$. Throughout these sections we will assume that the tasks N have been topologically sorted, and use $i \in N$ interchangeably for task i and the index of task i in the topological sorting.

4.4.1 Pre-Processing Break Combinations

A meal break can take place between any pair of pre-break and post-break task that can be performed consecutively and that allow for the minimum break duration B, giving rise to a potentially quadratic number of meal break candidates. For a given post-break task, however, it suffices to consider only a single pre-break task. Let $k \in N$ be a post-break task. Any pre-break task $j \in N$ must satisfy $t_e^j + B < t_s^k$ and $s_e^j = s_s^k$. In the time-space network, it suffices to consider only the last pre-break task, as indicated by the topological ordering, satisfying these conditions. After all, it is reachable from all earlier candidate tasks through waiting arcs. As such, we do not restrict the set of feasible duties.

Denote by `BreakPairs` the list of pairs of pre-break and post-break tasks:

Lemma 4.1. `BreakPairs` *can be constructed in* $\mathcal{O}(|N|^2)$ *time.*

Proof. Each task $k \in N$ can occur as post-break task in at most one pair (j, k), and does so if and only if a suitable pre-break task exists. This pre-break task, if any, can be retrieved by backtracking from the node of k along waiting arcs of G, and stopping at the first task meeting all break conditions. Since there are $\mathcal{O}(|N|)$ waiting

arcs in G, and we perform this procedure for each $k \in N$, the time complexity of constructing `BreakPairs` becomes $\mathcal{O}(|N|^2)$. □

Note that `BreakPairs` has size $\mathcal{O}(|N|)$ and can be pre-computed before entering the column generation algorithm.

4.4.2 Distance Matrix

The last two steps of the pricing algorithm, described in Sections 4.4.3 and 4.4.4, make use of a distance matrix $\mathbf{D}$. Entry $\mathbf{D}_{i,j}$ equals the distance of the shortest path, of duration at most L, from i to j, and equals $+\infty$ if no such path exists.

Lemma 4.2. $\mathbf{D}$ *can be computed in* $\mathcal{O}(|N|^2)$ *time.*

Proof. We compute $\mathbf{D}$ by solving one SSSPP for each task $i \in N$ and subsequently filling the entries of row $\mathbf{D}_i$. Note that in each SSSPP we enforce the maximum duration without break constraint using the method described in Section 4.3.2. Since we solve $\mathcal{O}(|N|)$ SSSPPs, and solving a single a single SSSPP on the time-space network takes $\mathcal{O}(|N|)$ time, the total time complexity becomes $\mathcal{O}(|N|^2)$. □

4.4.3 Matrix Computation

In the third step of the algorithm, we compute the matrix $\mathbf{E}$. Recall that entry $\mathbf{E}_{k,l}$ of this matrix equals the distance of the shortest path, of duration at most L, from k to any task with topological index smaller than or equal to l, and with the same end station as l. It equals $+\infty$ if no such task can be reached from k. Algorithm 4.2 presents `ComputeMatrix`, a procedure to compute $\mathbf{E}$ from distance matrix $\mathbf{D}$. For each possible post-break task k, the procedure initialises an auxiliary vector $\mathbf{E}^D$, storing the incumbent minimal distances to all different crew bases. It then iterates over all tasks in topological order to fill the entries of row $\mathbf{E}_k$.

Algorithm 4.2 `ComputeMatrix`(`BreakPairs`, $\mathbf{D}$, D, N)

$\mathbf{E} \leftarrow |\texttt{BreakPairs}| \times |N|$ matrix of $+\infty$'s
for $(j, k) \in \texttt{BreakPairs}$ **do**
  $\mathbf{E}^D \leftarrow 1 \times |D|$ matrix of $+\infty$'s
  for $l \in N$ **do**
    $\mathbf{E}^D_{s_e^l} \leftarrow \min\{\mathbf{D}_{k,l}, \mathbf{E}^D_{s_e^l}\}$
    $\mathbf{E}_{k,l} \leftarrow \mathbf{E}^D_{s_e^l}$
  end for
end for
return E

The following result follows directly from Algorithm 4.2, observing that both $|D|$ and $|\texttt{BreakPairs}|$ are $\mathcal{O}(|N|)$:

Lemma 4.3. *The time complexity of* `ComputeMatrix` *is* $\mathcal{O}(|N|^2)$.

4.4.4 Duty Generation

In the final step of the algorithm, we find the duty with minimum reduced cost by iterating over all combinations of start and post-break task. For each combination (i, k) we can look up the minimum distance required to complete this duty, i.e., the distance obtained by choosing an optimal feasible end task, in matrix $\mathbf{E}$. In particular, we know that any feasible end task must (i) end at most W time units after the start time t_s^i of the start task and (ii) be reachable from the post-break task k within L time units. We encode the first condition using the indices of the tasks given by the topological ordering of G. By construction, the topological ordering of G respects the ordering of the end times. As such, for each start task i there exists an index $latestEnd(i)$ such that all tasks with topological index smaller then or equal to $latestEnd(i)$ end at most W time units after t_s^i. This index can be computed for each start task in a pre-processing step of $\mathcal{O}(|N|^2)$ time complexity. By construction of $\mathbf{E}$, the distance from k to the optimal feasible end task is now given by $\mathbf{E}_{k,latestEnd(i)}$. Note that this distance is $+\infty$ when no feasible end task exists, in which case the reduced cost will evaluate to $+\infty$ too. The complete algorithm of the final step is given in Algorithm 4.3.

Algorithm 4.3 `ComputeMinimumReducedCost`(`BreakPairs`, $\mathbf{D}$, $\mathbf{E}$, N)

$RC^* \leftarrow +\infty$
for $i \in N$ **do**
  for $(j,k) \in$ `BreakPairs` **do**
    $RC \leftarrow c_F + c_V \times (t_e^i - t_s^i) - \lambda_i + \mathbf{D}_{i,j} + c_V \times (t_e^k - t_e^j) + \mathbf{E}_{k,latestEnd(i)}$
    $RC^* \leftarrow \min\{RC, RC^*\}$
  end for
end for
return RC^*

Lemma 4.4. *The time complexity of* `ComputeMinimumReducedCost` *is* $\mathcal{O}(|N|^2)$.

While Algorithm 4.3 only returns the minimum reduced cost, the corresponding duty can be easily retrieved as follows. First, in `ComputeMatrix` one would construct an auxiliary matrix storing the indices of the end tasks that yield the distances in $\mathbf{E}$. This can be done while maintaining the current time complexity. Given the start, post-break, and end tasks of the optimal duty, the full set of tasks in the duty can then be obtained by backtracking in $\mathcal{O}(|N|)$. Moreover, in practice one would like to add more than one column to the RMP in each iteration. Similar to the literature benchmark algorithm, `ComputeMinimumReducedCost` identifies at most $\mathcal{O}(|N|^2)$ negative reduced cost duties per iteration.

4.4.5 Time Complexity

Combining the results of the previous sections, the main result of this chapter reads:

Theorem 4.5. *The time complexity of* `PricingAlgorithm` *is* $\mathcal{O}(|N|^2)$.

Proof. `PricingAlgorithm` sequentially performs the four steps outlined in Sections 4.4.1-4.4.4. Since none of these steps has complexity beyond $\mathcal{O}(|N|^2)$, the time complexity of the full algorithm becomes $\mathcal{O}(|N|^2)$. □

4.5 Computational Experiments

We perform computational experiments to compare the computation times of the proposed algorithm to those of the benchmark algorithm of Huisman (2007). More

specifically, we set up a column generation framework to solve the RMP of several randomly generated instances to optimality. In this framework we solve each pricing problem twice: once using the benchmark algorithm and once using our proposed algorithm. Since both pricing algorithms solve identical pricing problems with identical dual values, we can directly compare their running times. The software used to conduct these experiments is publicly available (Van Rossum, 2022b).

In Section 4.5.1, we describe the instance generation procedure, and in Section 4.5.2, we describe the set-up of the column generation algorithm and the computational settings. We discuss the results in Section 4.5.3.

4.5.1 Instances

We create an instance of size $|N|$ by randomly generating $|N|$ tasks. For each task, we randomly select a start and end station out of a set of size $|S|$. Next, a random start time is drawn from the interval $[0, T_e]$. The end time is defined as the sum of the start time and the task duration, where the duration is drawn from the interval $[T^-, T^+]$. In our experiments, we use $|S| = |D| = 5$, $T_e = 24 \times 60$, $T^- = 20$, and $T^+ = 90$.

The remaining problem parameters are based on realistic values and are chosen as follows: $c_F = 900$, $c_V = 60$, $W = 9 \times 60 + 30$, $L = 5 \times 60$, and $B = 30$.

4.5.2 Set-up

For each instance we initialise a restricted master problem with a slack variable for each task. In each iteration, we solve the RMP with CPLEX 20.1.0 and then solve the pricing problem using both pricing algorithms. We add the columns generated by our algorithm to the RMP. To limit the number of columns in the RMP, similar to Breugem et al. (2022a) we only add columns to the RMP that are sufficiently task-disjoint. In particular, two columns may not have more than 50% of their tasks in common. All experiments are conducted using four cores of an Intel Xeon Gold 6130 processor.

4.5.3 Results

Figure 4.2 displays the average solution time per pricing iteration of each pricing algorithm for randomly generated instances of size $|N|$, for $|N|$ ranging from 100 to 1,250. As expected, the proposed algorithm is significantly faster than the benchmark

algorithm on all instance sizes, achieving a relative speed-up of up to 95% for $|N| = 1,250$. Moreover, the relative speed-up increases rapidly in $|N|$ as the ratio $|N|^2/|N|^3$ tends to zero.

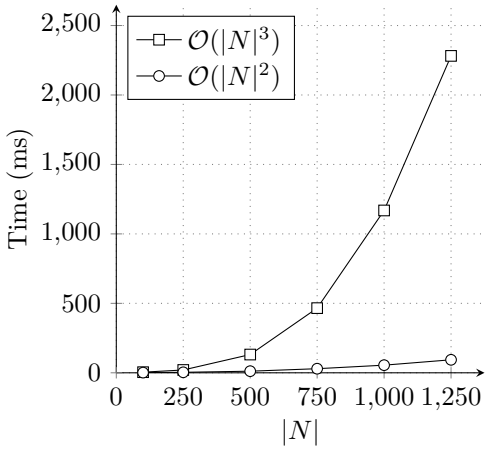


Figure 4.2: Average computation time per pricing iteration of each pricing algorithm, averaged over 10 randomly generated instances of size $|N|$, for different values of $|N|$.

4.6 Conclusion

In this chapter, we propose an $\mathcal{O}(|N|^2)$ algorithm for the pricing problem that is encountered when solving railway crew scheduling problems using column generation. In computational experiments on randomly generated instances, we find that running times decrease by up to 95% as compared to the literature benchmark algorithm with $\mathcal{O}(|N|^3)$ complexity.

We now discuss several assumptions of the crew scheduling problem that can be relaxed without compromising the $\mathcal{O}(|N|^2)$ complexity. First, the simple cost structure currently used can be expanded to incorporate source and sink costs, waiting costs (see, e.g., Hanafi and Kozan (2014)), paid break time, and task-specific costs. We refer to Heil et al. (2020) for an overview of commonly used duty cost structures. It is only when costs are incurred, that are dependent on the connections between specific pairs of tasks, that the current framework is no longer applicable. Second, it is possible to let the maximum duty length depend on the start time of the first task (as in, e.g., Breugem et al. (2022a)), a constraint that can easily be incorporated in

the duty generation procedure of Section 4.4.4. Third, one might wish to restrict the number of stations at which a meal break can take place, e.g. to large stations with a canteen only. This can easily be incorporated by restricting the set of post-break tasks in the pre-processing phase of Section 4.4.1.

Several interesting directions for future research remain. While in this chapter we have only compared exact algorithms for the pricing problem, in practice one might use heuristic pricing algorithms, such as labelling algorithms with strong dominance rules, to speed up the column generation algorithm. It would be interesting to compare the solution times of heuristic procedures with those of the newly proposed exact algorithm. In addition, one could attempt to further reduce the solution times of the proposed algorithm by incorporating completion bounds.

Part II

Fairness-Oriented Optimisation

Chapter 5

Optimising Fairness over Time with Homogeneous Workers

This chapter is based on B.T.C. van Rossum, R. Chen and A. Lodi (2023). 'Optimizing Fairness over Time with Homogeneous Workers'. 23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2023). *Schloss-Dagstuhl-Leibniz Zentrum für Informatik.*

5.1 Introduction

While most optimisation literature focuses on minimising costs or maximising efficiency, there is a growing interest in including fairness in optimisation models. In particular, fairness over time is desirable for sequential decision making. It relates to the situation where a central decision maker is faced with a multi-period optimisation problem involving multiple agents, and the goal is to maximise fairness (minimise unfairness) of the (dis)utilities that agents gain from the solution. A mixed-integer programming framework has been proposed for the offline version of the problem, in which the optimisation problems to be solved in all time periods are known in advance (Lodi et al., 2024a; Lodi et al., 2024b). The online version of the problem, where instances are revealed in a dynamic fashion, has been studied for, among other things, resource allocation (Si Salem et al., 2022) and railway crew planning (Chapter 3).

In this chapter, we restrict our attention to fairness over time in the context of online assignment problems. We define assignment problems as combinatorial problems that involve dividing the solution into blocks of work to be assigned to workers. Many real-life problems, including vehicle routing and crew scheduling, can be modeled in this way. In each time period, a new problem instance is revealed, and a solution must be determined with a cost that is within a pre-determined factor of the minimum cost. Next, the solution is assigned to a fixed group of workers whose utilities are updated accordingly. The goal is to determine solutions and assignments such that the unfairness of the workers' (dis)utilities is minimised.

The contributions of this chapter are as follows. We formally introduce the online assignment problem with fairness over time, and derive some theoretical results. Next, we propose a simple and interpretable assignment policy and prove some desirable properties. Finally, we perform a case study on the capacitated vehicle routing problem, where we provide an exact branch-and-price algorithm for the problem of route balancing subject to a budget constraint. Empirically, we show that the most efficient solution usually results in unfair assignments while much more fair solutions can be attained with minor efficiency loss using our approach.

5.2 The Online Assignment Problem with Fairness over Time

We consider a sequential decision making problem with a planning horizon of length T. In each period $t \in T$, the decision maker receives an instance $(c^t(\cdot), \mathcal{X}^t)$ of a combinatorial optimisation problem, where $c^t(\cdot)$ and $\mathcal{X}^t$ denote its cost objective and its feasible region, respectively. The decision maker has to choose a solution $\boldsymbol{x}^t \in \mathcal{X}^t_\alpha$. Here, $\mathcal{X}^t_\alpha$ denotes the set of acceptable solutions to the problem of period t, parameterised by α. In principle, the definition of the set of acceptable solutions can be very general. In this chapter, we focus on cost-efficient solutions to avoid pathological cases of perfectly fair but highly inefficient solutions. Specifically, we define $\mathcal{X}^t_\alpha$ to be the set of feasible solutions whose costs are within a factor $(1+\alpha)$ of the minimum cost, i.e.,

$$\mathcal{X}^t_\alpha = \Big\{ \boldsymbol{x} \in \mathcal{X}^t : c^t(\boldsymbol{x}) \leq (1+\alpha) \min_{\boldsymbol{z} \in \mathcal{X}^t} c^t(\boldsymbol{z}) \Big\}. \tag{5.1}$$

Varying α allows us to characterise the trade-off between efficiency and equity.

We assume that each solution $\boldsymbol{x}^t \in \mathcal{X}^t$ can be partitioned into n pieces of work $(x^t_1, \ldots, x^t_n)$ that must be assigned to n homogeneous workers. As such, the work assignment can be freely permuted without changing its cost. With a slight abuse of notation, for each permutation $\pi \in \Pi^n$ of $\{1, \ldots, n\}$, we define the mapping $\pi(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ by $\pi(y_1, \ldots, y_n) = (y_{\pi_1}, \ldots, y_{\pi_n})$. We assume that $\boldsymbol{x}^t \in \mathcal{X}^t$ if and only if $\pi(\boldsymbol{x}^t) \in \mathcal{X}^t$. We do not distinguish between a permutation of subvectors of $(x^t_1, \ldots, x^t_n)$ and a permutation of coordinates of an n-dimensional vector.

Each assignment of $\boldsymbol{x}^t$ yields a payoff vector $\boldsymbol{p}(\boldsymbol{x}^t) \in \mathbb{R}^n$, representing the payoffs to the n workers. Since the workers are homogeneous, we assume that the same piece of work x^t_i yields an identical payoff to each worker, i.e., $\boldsymbol{p}(\pi(\boldsymbol{x}^t)) = \pi(\boldsymbol{p}(\boldsymbol{x}^t))$ for all permutations π. Payoffs are aggregated in a linear fashion, such that the current utility vector equals the sum of all previous payoff vectors, i.e., $\boldsymbol{u}^t = \sum_{\tau=1}^t \boldsymbol{p}(\boldsymbol{x}^\tau)$. We let $\phi(\cdot)$ denote an unfairness measure of the worker's utilities satisfying the definition of the inequity measure in Tsang and Shehadeh (2024), e.g., the difference between the largest and smallest utilities. Our goal is to determine online a sequence of solutions $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^T) \in \prod_{t=1}^T \mathcal{X}^t_\alpha$ such that $\phi(\boldsymbol{u}^T)$ is minimised. The offline version

of the online assignment problem with fairness over time (OAPFoT) reads as

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \phi(\boldsymbol{u}^T) \tag{5.2a}$$

$$\text{s.t.} \quad \boldsymbol{u}^t = \boldsymbol{u}^{t-1} + \boldsymbol{p}(\boldsymbol{x}^t) \qquad t = 1, \dots, T \tag{5.2b}$$

$$\boldsymbol{x}^t \in \mathcal{X}_\alpha^t, \qquad t = 1, \dots, T, \tag{5.2c}$$

$$\boldsymbol{u}^0 = \boldsymbol{0}. \tag{5.2d}$$

In brief, in each time period t, the problem can be split into the following three steps:

1. *Defining the set of acceptable solutions.* In the case of $\mathcal{X}_\alpha^t$ being cost-efficient solutions as defined in (5.1), we must compute the optimal cost $\min_{\boldsymbol{z}\in\mathcal{X}^t} c^t(\boldsymbol{z})$ to explicitly define $\mathcal{X}_\alpha^t$. We assume that a suitable algorithm is available for this problem, and consider this step to be outside the scope of this chapter.

2. *Picking an acceptable solution.* A criterion, which may or may not depend on $\boldsymbol{u}^{t-1}$, has to be decided in order to pick a solution from $\mathcal{X}_\alpha^t$. Then, potentially we need to solve an optimisation problem associated with that criterion, which we refer to as the α-*subproblem*, to obtain a particular acceptable solution $\boldsymbol{z}^t$.

3. *Assigning the solution to workers.* If the criterion we use in Step 2 has not yet taken $\boldsymbol{u}^{t-1}$ into account, we may need to determine how to assign the solution $\boldsymbol{z}^t$ obtained in Step 2 to workers to obtain $\boldsymbol{x}^t = \pi(\boldsymbol{z}^t)$. In other words, we decide upon a permutation π.

The latter two steps can be used to minimise ϕ. Proposing good strategies for picking and assigning solutions is the main goal of this chapter. In the remainder of this chapter, we analyse the performance of the following combination of strategies. In Step 2, we propose to pick the solution whose payoffs minimise the inequity function, i.e., we pick $\boldsymbol{z}^t \in \arg\min_{\boldsymbol{z}\in\mathcal{X}_\alpha^t} \phi(\boldsymbol{p}(\boldsymbol{z}))$. The rationale behind this strategy is that solutions with equitable payoffs, when properly assigned, lead to equitable utilities. In Step 3, we make use of a simple policy that assigns work, in increasing order of payoffs, to workers, in decreasing order of current utility. Some theoretical justifications for this assignment policy are provided in Section 5.4.

5.3 Problem Complexity

We first present some results regarding the complexity of OAPFoT. A simple reduction from the partition problem yields the following proposition.

Proposition 5.1 ($\mathcal{NP}$-hardness of the Offline OAPFoT)**.** *The offline version of OAPFoT is $\mathcal{NP}$-hard even if the original cost minimisation problem is solvable in polynomial time and $T = 1$, and even if each $\mathcal{X}_\alpha^t$ contains only solutions identical up to a permutation and $n = 2$.*

The above results indicate that even the offline assignment problem is hard to solve. We now turn our attention to online assignment policies.

Definition 5.2 (Online assignment policy)**.** *An online assignment policy is a function $\tau(\cdot, \cdot)$ that maps any combination of a utility vector $\boldsymbol{u} \in \mathbb{R}^n$ and a payoff vector $\boldsymbol{p} \in \mathbb{R}^n$ to a permutation $\pi \in \Pi^n$.*

Informally, each policy determines how the next payoffs should be assigned to workers based on their current utilities and the next payoffs. Due to the online nature of the problem, it is easy to see that no such policy can always attain minimum unfairness.

Proposition 5.3 (Non-existence)**.** *For $n \geq 2$ and $t \geq 3$, there does not exist an online assignment policy that attains perfect fairness on all instances that admit perfect fairness.*

Proof. Let τ be any assignment policy, $n = 2$, and $t = 3$. We will construct two instances that admit a perfectly fair solution and show that τ cannot attain perfect fairness on both.

The first instance is characterised by payoff vectors $\boldsymbol{p}_1^1 = (0, 1)$, $\boldsymbol{p}_1^2 = (0, 2)$, and $\boldsymbol{p}_1^3 = (0, 3)$. The second instance differs in the third payoff vector only: $\boldsymbol{p}_2^1 = (0, 1)$, $\boldsymbol{p}_2^2 = (0, 2)$, and $\boldsymbol{p}_2^3 = (0, 1)$. Note that both instances admit perfectly fair allocations with aggregate utility vectors $(3, 3)$ and $(2, 2)$, respectively.

Since both instances are identical up to $t = 2$, policy τ will attain identical aggregate utilities $\boldsymbol{u}^2$ on both instances. Depending on the assignment for $t = 2$, this aggregate vector equals either $(1, 2)$ or $(0, 3)$. In the first case, it becomes impossible to attain perfect fairness on the first instance. In the second case, it becomes impossible to

attain perfect fairness on the second instance. This shows that τ does not always attain minimum unfairness.

We can extend the two instances to $n > 2$ by repeating the positive entries of the payoff vectors. Similarly, we can extend to all $t > 3$ by adding all-zero payoff vectors for later time periods. This concludes the proof. □

5.4 A Simple Online Assignment Policy

We no present some positive results for one particular policy that is rather intuitive and simple to implement. This policy, which we refer to as the *best-to-worst policy* (BTW) and denote by τ^{BTW}, assigns payoffs, in increasing order, to workers, in decreasing order of current utility. More formally, $\pi^{\text{BTW}} := \tau^{\text{BTW}}(\boldsymbol{u}, \boldsymbol{p})$ satisfies $\pi_i^{\text{BTW}}(\boldsymbol{p}) < \pi_j^{\text{BTW}}(\boldsymbol{p})$ if $\boldsymbol{u}_i > \boldsymbol{u}_j$ and only if $\boldsymbol{u}_i \geq \boldsymbol{u}_j$ with ties broken arbitrarily. We next show that this policy is always locally optimal in the unfairness measure ϕ.

Proposition 5.4 (Local optimality of BTW)**.** *Let $\boldsymbol{u}, \boldsymbol{p} \in \mathbb{R}^n$ and ϕ be an inequity measure. It holds that $\pi^{BTW} = \tau^{BTW}(\boldsymbol{u}, \boldsymbol{p}) \in \arg\min_{\pi \in \Pi^n} \phi(\boldsymbol{u} + \pi(\boldsymbol{p}))$.*

Proof. Assume without loss of generality that $p_1 \leq p_2 \leq \ldots \leq p_n$, $u_1 \leq u_2 \leq \ldots \leq u_n$, and $\pi_i^{\text{BTW}} = n - i + 1$ for $i = 1, \ldots, n$. Let $\pi \in \arg\min_{\pi \in \Pi^n} \phi(\boldsymbol{u} + \pi(\boldsymbol{p}))$, and let i be the smallest index for which $\pi_i < n - i + 1$, i.e., for which the assignments of π and π^{BTW} differ. This implies that there exists a $j > i$ for which $\pi_j = n - i + 1 > \pi_i$. Recall that, by construction, $p_{\pi_i} \leq p_{\pi_j}$. If $p_{\pi_i} = p_{\pi_j}$, then we can reverse the assignments of i and j without affecting the resulting utilities. Otherwise, we have $p_{\pi_i} < p_{\pi_j}$, and reversing the assignments of i and j constitutes a Pigou-Dalton transfer that can only decrease unfairness (Tsang and Shehadeh, 2024). Note that the smallest index satisfying our condition is now increased by at least one. Hence, in at most $n - 1$ of such transfers we can convert the assignment determined by π into that determined by π^{BTW}. Since each transfer can only decrease unfairness, it holds that $\phi(\boldsymbol{u} + \pi^{BTW}(\boldsymbol{p})) \leq \phi(\boldsymbol{u} + \pi(\boldsymbol{p}))$. Therefore, $\pi^{BTW} \in \arg\min_{\pi \in \Pi^n} \phi(\boldsymbol{u} + \pi(\boldsymbol{p}))$. □

In addition, under the BTW policy the unfairness at any stage is bounded by the maximum unfairness of any set of payoffs. We present a simplified version of this result for range unfairness, defined as the largest difference between payoffs/utilities.

Proposition 5.5 (Bounded range unfairness)**.** *Under the best-to-worst policy and with* $\phi(\boldsymbol{u}) = \max_{i=1,\dots,n} u_i - \min_{i=1,\dots,n} u_i$*, it holds that* $\phi(\boldsymbol{u}^t) \leq \max_{\tau=1,\dots,t} \phi(\boldsymbol{p}^\tau)$ *for all* $t = 1, \dots, T$*.*

The above result is the main motivation for minimising the unfairness of the payoffs in our proposed strategy, as it further minimises an upper bound of $\phi(\boldsymbol{u}^t)$ for all t.

5.5 Case Study: Capacitated Vehicle Routing Problem

We perform a case study on the capacitated vehicle routing problem (CVRP) to test the effectiveness of our proposed strategies and to analyse the role of the budget parameter α. We define the payoff of a route in terms of either its distance or its load, i.e., we use both variable-sum and constant-sum payoffs (Matl et al., 2019). As the unfairness measure ϕ, we use the range, defined as the largest difference in payoffs between any two routes. Solving the α-subproblem now corresponds to selecting a set of routes for which the range, in terms of either distance or load, is minimised, subject to the α-budget constraint. This problem strongly relates to the CVRP with route balancing, for which no efficient exact algorithms are known.

We now introduce the necessary notations for the α-subproblem, omitting time indices t for brevity. Let N denote the set of customers, K denote the number of vehicles, i.e., workers, and B denote the cost of the most cost-efficient solution. We denote by R the set of all feasible routes and we introduce a binary variable x_r, $\forall r \in R$ that takes value 1 if route r is selected. The cost and payoff of route r are denoted by c_r and p_r, respectively. Binary parameter a_{ir} indicates whether customer $i \in N$ is visited by route r. We model the range using variables η and γ representing the maximum payoff and the minimum payoff, respectively. Building on the formulation of Bianchessi et al. (2023), we model the maximum and minimum based on the last customer of the route. Binary parameter b_{ir} indicates whether customer i is the last on route r. Finally, let M be an upper bound on the minimum payoff of any route. Our formulation for the α-subproblem now reads as

$$\min \quad \eta - \gamma \tag{5.3a}$$

$$\text{s.t.} \quad \sum_{r \in R} a_{ir} x_r = 1, \qquad \forall i \in N, \tag{5.3b}$$

$$\sum_{r \in R} p_r b_{ir} x_r \leq \eta, \qquad \forall i \in N, \tag{5.3c}$$

$$M \left(1 - \sum_{r \in R} b_{ir} x_r \right) + \sum_{r \in R} p_r b_{ir} x_r \geq \gamma, \qquad \forall i \in N, \tag{5.3d}$$

$$\sum_{r \in R} c_r x_r \leq B(1 + \alpha) \tag{5.3e}$$

$$\sum_{r \in R} x_r = K \tag{5.3f}$$

$$\boldsymbol{x} \in \{0, 1\}^R. \tag{5.3g}$$

We solve the above program using branch and price, in which we solve a pricing problem for each possible last customer at each node of the search tree. The pricing problems are solved using bidirectional labelling and the ng-route relaxation (Baldacci et al., 2011). Since we consider symmetric instances of the CVRP, we strengthen the formulation by enforcing that the index of the last customer along a route is at least that of the first customer. We branch on the last customer first, followed by arcs, and separate rounded capacity inequalities at the root node of the branching tree.

The set-up of our experiments is as follows. Similar to Matl et al. (2019), we generate a sequence of 20 daily instances of $N = 15$ customers by taking disjoint subsets of instance X-n641-k35 (Uchoa et al., 2017). This yields a single online instance of $T = 20$ time periods. We use $K = 5$ vehicles (one for each worker, i.e., $n = 5$), and set the vehicle capacity to $Q = \lceil \frac{1}{K-1} \sum_{i=1}^{N} q_i - 1 \rceil$. We consider values of α in $\{0, 1\%, \ldots, 10\%\}$, and use the best-to-worst policy to assign routes to workers. We solve the LP-relaxation of the restricted master problem using CPLEX 22.1.0.

Results of our case study for both the distance and load resource are summarised in Figure 5.1. For each value of α, we present the range of the payoffs ($\boldsymbol{p}$), utilities ($\boldsymbol{u}$), and the computing time. Results are averaged over all time periods. In line with Proposition 5.5, we find that the unfairness of the utilities is generally well below that of the payoffs, showing the effectiveness of the best-to-worst policy. Both ranges of payoffs and utilities decrease as α grows, though the effect is more pronounced for payoffs. While we observe similar patterns for distance and load, we note that the range of the load displays a stronger reduction as a function of α. In addition, the range of the utilities is closer to that of the payoffs for load than for distance.

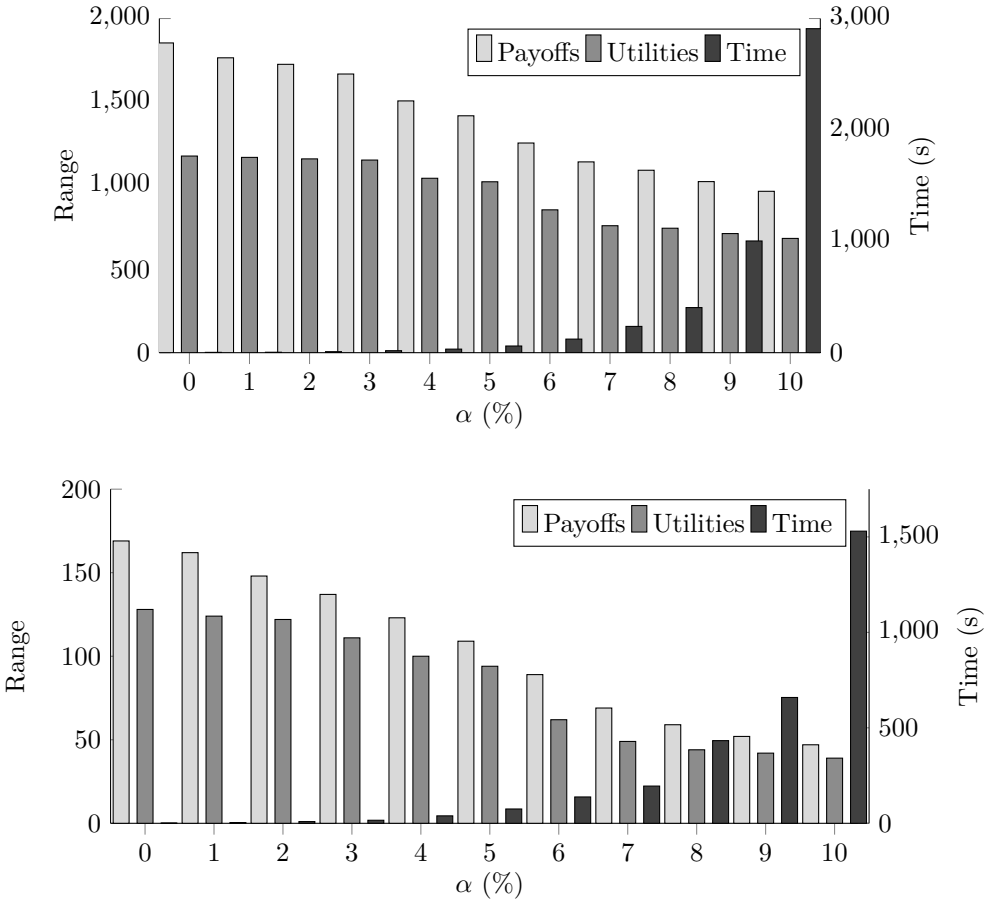


Figure 5.1: Results for the distance (top) and load (bottom) resources for different values of α.

The computing times grow rapidly in α. We remark that the α-subproblem appeared to be a lot harder to solve than the cost minimisation problem. While an optimal solution to the latter was always obtained within seconds, the former could take multiple hours for $\alpha = 10\%$. This discrepancy can be attributed to the large integrality gap of our formulation, resulting in large branch-and-bound trees, which can have more than 100,000 nodes.

5.6 Future Research

We consider several directions for future research. First, we aim to study the performance of using a different criterion for picking acceptable solutions. In particular, we will base the selection on current utilities $\boldsymbol{u}^{t-1}$ and select the solution $\boldsymbol{z}^t = \arg\min_{\boldsymbol{y}^t \in \mathcal{X}^t_\alpha} \phi(\boldsymbol{p}(\boldsymbol{y}^t) + \boldsymbol{u}^{t-1})$. This would effectively eliminate the need for an assignment policy, and would also require a slight reformulation of the α-subproblem. Surprisingly, preliminary experiments indicate that this approach might be counterproductive in some cases. Second, we aim to further explore why the α-subproblem appears to be much harder than the original cost-efficient problem. Hopefully, the resulting insights can be used towards the development of more efficient solution methods. We experimented with the use of cutting planes from the vehicle routing literature but these attempts proved ineffective, indicating the need for developing other techniques. Chapter 6 provides a first step in this direction. Finally, it would be interesting to perform a case study on a large-scale crew scheduling problem, to analyse the performance of our approach in settings with a larger number of workers.

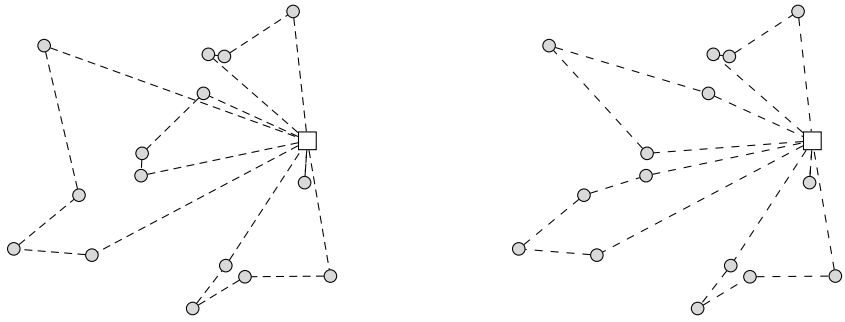
Chapter 6

Efficient Branching Rules for Optimising Range and Order-Based Objective Functions

This chapter is an extended version of B.T.C. van Rossum, R. Chen and A. Lodi (2024a). 'A New Branching Rule for Range Minimization Problems'. Integer Programming and Combinatorial Optimization. *Ed. by J. Vygen and J. Byrka. Cham: Springer Nature Switzerland, pp. 433–445.*

6.1 Introduction

A growing stream of literature focuses on fairness-oriented optimisation (Chen and Hooker, 2023; Jozefowiez et al., 2009; Karsu and Morton, 2015; Lodi et al., 2024a; Matl et al., 2018; Matl et al., 2019; Tsang and Shehadeh, 2024), where not only the efficiency but also the fairness of solutions are taken into account. A popular fairness measure is the range, defined as the maximum minus the minimum of the quantities of interest (Matl et al., 2018; Matl et al., 2019; Tsang and Shehadeh, 2024). In many practical applications, the range and hence the fairness of the solution can be significantly improved with a minor loss in efficiency. Relevant examples include vehicle routing (Matl et al., 2019, Chapter 5), crew rostering (Breugem et al., 2022b), order picking (Vanheusden et al., 2020), and the p-median problem (Ljubić et al., 2024; Marín et al., 2020). Indeed, Figure 6.1 presents an example where the range between the distances of vehicle routes can be reduced by 26.2% by tolerating an efficiency loss of 0.05% compared to the most cost-efficient solution. In addition to fairness applications, range minimisation is often encountered as one of the objectives in bi-objective optimisation (Jozefowiez et al., 2009) or even as a stand-alone problem (Puerto et al., 2012).



(a) Total distance: 6,178. Range: 2,170. **(b)** Total distance: 6,181. Range: 1,601.

Figure 6.1: Efficient (left) and fair (right) vehicle routing solutions.

We focus on range minimisation problems whose formulations feature a number of columns that is prohibitively large for mixed-integer linear programming solvers. This is commonly observed in railway crew scheduling, vehicle routing, and airline crew pairing problems (Desaulniers et al., 2005). Here, formulations with a huge number of variables are preferred as they typically display a tighter linear programming (LP) relaxation and a less symmetric structure than their compact counterparts (Barnhart

et al., 1998). When a classical cost-based objective is used, branch and price is the state-of-the-art solution approach for this type of problems. At each node of the branch-and-bound tree, the LP relaxation is solved through column generation. Here, only a small subset of all possible columns is initially considered, and a pricing algorithm is used to solve the pricing problem of identifying columns that potentially improve the objective value. A branching scheme is used to obtain integer solutions, and all branching decisions must be compatible with the pricing algorithm. We refer to Barnhart et al. (1998) for a detailed introduction to branch and price.

Unfortunately, formulations for range minimisation problems often suffer from poor LP relaxations that drastically impair the performance of classical branch-and-price algorithms. Consider, for example, the capacitated vehicle routing problem (CVRP) with route balancing, which can be phrased as a bi-objective problem where range minimisation is one of the objectives. While the branch-and-price (-and-cut) scheme is highly effective for the classical cost-oriented CVRP (Costa et al., 2019), the range minimisation problem is known to suffer from very weak dual bounds, as observed in Chapter 5.

The phenomenon of weak dual bounds has also been repeatedly observed in min-max routing problems, where the goal is to minimise the length of the longest route. In a sense, these problems are special cases of range minimisation problems, with a zero objective coefficient on the length of the shortest route. Examples include the min-max multiple travelling salesman problem (França et al., 1995) and the min-max open vehicle routing problem (Applegate et al., 2002; Lysgaard et al., 2020; Marinho Damião, 2021). For min-max routing problems, the bisection method (a form of dichotomous search) can be efficiently applied to obtain optimal solutions on instances with up to 150 nodes (França et al., 1995; Marinho Damião, 2021). Unfortunately, this approach cannot be generalised to range minimisation: since the solution to range minimisation problems is characterised by both a minimum and maximum payoff, binary search along one dimension is not possible. As a result, enumeration schemes (Matl et al., 2019) and a wide range of heuristics have been proposed for the fairness-oriented CVRP (see Matl et al. (2018) for an overview).

Our main contribution is to propose an efficient exact branch-and-price framework for large-scale range minimisation problems. In particular, we propose a specialised branching rule, which we refer to as *range branching*, that directly tackles the poor LP relaxation of range minimisation formulations. It can be used on top of problem-specific branching schemes, and allows one to effectively leverage branch-

and-price techniques commonly applied to cost-oriented optimisation problems. We show some promising properties of range branching, and evaluate its effectiveness on a series of fairness-oriented capacitated vehicle routing and generalised assignment problems. Range branching significantly improves the performance of multiple classical branching schemes in terms of computing time, optimality gap, and size of the branch-and-bound tree, allowing us to solve many more instances to optimality and outperform a standard MILP-solver. This result appears to be mainly driven by strongly improved dual bound behaviour. Finally, we successfully show how range branching can be generalised to order-based minimisation problems, and use this result to minimise the Gini deviation, defined as the sum of absolute pairwise differences, in CVRP.

The remainder of this chapter is structured as follows. We formally define the range minimisation problem in Section 6.2 and present range branching in Section 6.3. In Sections 6.4 and 6.5, we present the results of computational experiments on the fair capacitated vehicle routing problem and fair generalised assignment problem, respectively. We extend our range branching rule to the case of general order-based objective functions in Section 6.6, and conclude in Section 6.7.

6.2 Range Minimisation Problem

We formally define the range minimisation problem in Section 6.2.1, and discuss mathematical formulations for this problem in Section 6.2.2.

6.2.1 Problem Description

Throughout this chapter, we consider problems with the following structure. Given a set of N columns (implicitly), the problem requires to select a subset of them, with decision variables $\boldsymbol{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$ being the indicator variables of the selected columns.[2] We assume that some linear constraints $A\boldsymbol{x} \leq \boldsymbol{b}$ must be respected. These constraints can include efficiency lower bounds, constraints on the number of selected columns, and other problem-specific constraints (e.g., in the case of the CVRP, regular customer partition constraints). To summarise, the feasible region of the decision variables is given by $\mathcal{X} = \{\boldsymbol{x} \in \{0,1\}^N : A\boldsymbol{x} \leq \boldsymbol{b}\}$, and the feasible region of its linear programming relaxation is given by $\mathcal{X}_{LP} = \{\boldsymbol{x} \in [0,1]^N :$

[2]For simplicity, we do not assume additional auxiliary variables. However, these can be easily incorporated into our approach.

$A\boldsymbol{x} \leq \boldsymbol{b}\}$.

The objective is to minimise the range of the payoffs of selected columns. In particular, we assume that each column i has an associated payoff value p_i. Without loss of generality, we assume nonnegative and bounded payoffs, i.e., for all i we have $p_i \in [0, M]$ for some (large) constant M. For notational convenience, given $\boldsymbol{x} \in [0,1]^N$ and $\boldsymbol{p} \in [0, M]^N$, define $p_{\min}(\boldsymbol{x}) = \min_{i:x_i>0} p_i$ and $p_{\max}(\boldsymbol{x}) = \max_{i:x_i>0} p_i$ to be the minimum and maximum payoff of any (possibly fractional) solution $\boldsymbol{x}$, respectively. We use a definition that covers fractional solutions because these are frequently encountered during the course of branch and price. The goal of the range minimisation problem is to determine a binary solution $\boldsymbol{x} \in \mathcal{X}$ minimising the difference between the largest and smallest payoffs, i.e., minimising $p_{\max}(\boldsymbol{x}) - p_{\min}(\boldsymbol{x})$.

Finally, we assume that the large number of columns N prohibits the out-of-the-box use of standard mixed-integer programming solvers, and that an exact branch-and-price algorithm is used to solve the problem. In particular, we assume that a problem-specific pricing algorithm and a problem-specific branching scheme, i.e., set of branching rules, are available. The range branching rule we propose will operate *on top of* the problem-specific branching scheme. In other words, the range branching rule is to be invoked first at any node in the branch-and-price tree, and the regular branching scheme is applied only if range branching does not detect any branching opportunities. Since the range branching decisions induce constraints involving the payoff value p_i, as well as constraints limiting the domain of p_i, we assume that these constraints are amenable to the pricing algorithm. We will show that this is the case for the vehicle routing and generalised assignment applications under consideration.

6.2.2 Mathematical Formulation

We now discuss a general class of formulations for the range minimisation problem. To this end, we first introduce the concept of *range-respecting solutions*.

Definition 6.1 (Range-Respecting Solution)**.** *Let a vector $(\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma}) \in [0,1]^N \times \mathbb{R}^2$ be given. We say this vector is* range-respecting *(with respect to $\boldsymbol{p}$) if $\bar{\eta} \geq p_{\max}(\bar{\boldsymbol{x}})$ and $\bar{\gamma} \leq p_{\min}(\bar{\boldsymbol{x}})$, and* range-violating *(with respect to $\boldsymbol{p}$) otherwise.*

The range of a solution $\bar{\boldsymbol{x}} \in \mathcal{X}$ is equal to $\bar{\eta} - \bar{\gamma}$ with the smallest $\bar{\eta}$ and largest $\bar{\gamma}$ that make $(\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma})$ range-respecting. We are now ready to define a class of formulations for the range minimisation problem.

Definition 6.2 (Valid Formulation)**.** *Let $A, \boldsymbol{b}$ be given. Let $\boldsymbol{w} = (C, \boldsymbol{d}, \boldsymbol{f}, \boldsymbol{g})$ be a tuple of constraint matrices and coefficient vectors, and consider the following mixed-binary linear program:*

$$\min_{\boldsymbol{x},\eta,\gamma} \quad \eta - \gamma \tag{6.1a}$$

$$s.t. \quad A\boldsymbol{x} \leq \boldsymbol{b} \tag{6.1b}$$

$$C\boldsymbol{x} + \eta\boldsymbol{d} + \gamma\boldsymbol{f} \leq \boldsymbol{g} \tag{6.1c}$$

$$\boldsymbol{x} \in \{0,1\}^N. \tag{6.1d}$$

Auxiliary constraints (6.1c), as defined by $\boldsymbol{w}$, model the values of η and γ representing the maximum and minimum payoff. We say program (6.1) is a valid formulation *for the range minimisation problem when $\boldsymbol{z} = (\boldsymbol{x}, \eta, \gamma)$ is feasible in (6.1) if and only if $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{z}$ is range-respecting. We say $\boldsymbol{w}$ is valid when it defines a valid formulation. We denote by $\mathcal{Z}_{LP}(\boldsymbol{w})$ the feasible region of the LP relaxation of (6.1), obtained by relaxing constraints (6.1d) to $\boldsymbol{x} \in [0,1]^N$.*

Without loss of generality, we can assume that, for any solution in the LP relaxation $\mathcal{X}_{LP}$ of feasible region $\mathcal{X}$, its range-respecting counterpart is contained in the LP relaxation of any valid formulation.

Assumption 6.3 (Fractional Range-Respecting Solutions)**.** *Let $\boldsymbol{w}$ be valid. Then, for any $\boldsymbol{x} \in \mathcal{X}_{LP}$, the range-respecting solution $\boldsymbol{z} = (\boldsymbol{x}, p_{max}(\boldsymbol{x}), p_{min}(\boldsymbol{x}))$ is contained in $\mathcal{Z}_{LP}(\boldsymbol{w})$.*

Suppose, to the contrary, that the auxiliary constraints modelling the range tighten the description of the feasible region and cut off some fractional $\boldsymbol{x}$. Then, these auxiliary constraints could have been included as valid inequalities in the set $A\boldsymbol{x} \leq \boldsymbol{b}$.

Ideally, we wish to minimise over the set of range-respecting solutions only. Unfortunately, this goal is well out of reach. Since the operators $p_{\max}$ and $p_{\min}$ are non-convex and non-concave, any valid formulation of a nontrivial range minimisation problem contains infinitely many range-violating fractional solutions, as demonstrated in our next result.

Proposition 6.4 (Fractional Range-Violating Solutions)**.** *Let $\mathcal{X}$ be such that there exist distinct $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ for which $p_{max}(\boldsymbol{x}_1) \neq p_{max}(\boldsymbol{x}_2)$ or $p_{min}(\boldsymbol{x}_1) \neq p_{min}(\boldsymbol{x}_2)$. Then, for any valid $\boldsymbol{w}$, it holds that $\mathcal{Z}_{LP}(\boldsymbol{w})$ contains infinitely many range-violating*

fractional solutions.

Proof. Let a valid $\boldsymbol{w}$ be given, and suppose that $p_{\max}(\boldsymbol{x}_1) > p_{\max}(\boldsymbol{x}_2)$ for some $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$. Since $\boldsymbol{w}$ is valid, we know that the range-respecting integral solutions $\boldsymbol{z}_1 = (\boldsymbol{x}_1, p_{\max}(\boldsymbol{x}_1), p_{\min}(\boldsymbol{x}_1))$ and $\boldsymbol{z}_2 = (\boldsymbol{x}_2, p_{\max}(\boldsymbol{x}_2), p_{\min}(\boldsymbol{x}_2))$ are contained in $\mathcal{Z}_{LP}(\boldsymbol{w})$. Now, for $\lambda \in (0,1)$, define $\boldsymbol{z}(\lambda) = (\boldsymbol{x}(\lambda), \eta(\lambda), \gamma(\lambda)) = \lambda \boldsymbol{z}_1 + (1-\lambda)\boldsymbol{z}_2$. By convexity of $\mathcal{Z}_{LP}(\boldsymbol{w})$, we know that $\boldsymbol{z}(\lambda) \in \mathcal{Z}_{LP}(\boldsymbol{w})$. By construction, $p_{\max}(\boldsymbol{x}_\lambda) = p_{\max}(\boldsymbol{x}_1) > \lambda p_{\max}(\boldsymbol{x}_1) + (1-\lambda) p_{\max}(\boldsymbol{x}_2) = \eta(\lambda)$ for $\lambda \in (0,1)$. In other words, $\boldsymbol{z}(\lambda) \in \mathcal{Z}_{LP}(\boldsymbol{w})$ is range-violating for $\lambda \in (0,1)$. The case $p_{\min}(\boldsymbol{x}_1) > p_{\min}(\boldsymbol{x}_2)$ is treated similarly. □

The admission of many range-violating fractional solutions can result in poor dual bounds, thereby hindering the performance of branch-and-price algorithms. As we will see, range branching provides a way of eliminating range-violating solutions from valid formulations.

While constructing a tractable valid formulation is highly problem-dependent, we like to make some general remarks on how to formulate constraints that correctly link $\boldsymbol{x}$ to η and γ. First, constraints per individual column of the type $p_i x_i \leq \eta$ are valid for *any* range minimisation problem, but are highly impractical. The use of such constraints would require simultaneous column-and-row generation, severely complicating the pricing problem. Instead, one typically resorts to some form of aggregated constraints. A notable case arises when the set of columns can be written as some union of subsets, and exactly one column is selected from each subset in any feasible solution. This situation is frequently encountered in agent-based problems, where each agent is always assigned exactly one 'job' (column). Formally, let I denote the set of column indices and suppose that $I = \bigcup_{j \in [m]} I_j$, where the I_j need not be pairwise disjoint. If, for any feasible $\boldsymbol{x} \in \mathcal{X}$ and all $j \in [m]$, it holds that $\sum_{i \in I_j} x_i = 1$, the following constraints correctly model the maximum and the minimum:

$$\sum_{i \in I_j} p_i x_i \leq \eta \qquad \forall j \in [m] \tag{6.2a}$$

$$\sum_{i \in I_j} p_i x_i \geq \gamma \qquad \forall j \in [m]. \tag{6.2b}$$

In some applications, we might only be able to show that *at most* one column is selected from each subset. If, for any feasible $\boldsymbol{x} \in \mathcal{X}$ and all $j \in [m]$, it holds that $\sum_{i \in I_j} x_i \leq 1$, we must replace constraints (6.2) by the following, slightly weaker set

of constraints:

$$\sum_{i \in I_j} p_i x_i \leq \eta \qquad \forall j \in [m] \tag{6.3a}$$

$$M + \sum_{i \in I_j} (p_i - M) x_i \geq \gamma \qquad \forall j \in [m]. \tag{6.3b}$$

Recall that M is an upper bound on the payoff of any column. All valid formulations presented in this chapter make use of some version of either constraints (6.2) or (6.3). Of course, identifying sets I_j requires problem-specific knowledge and might not always be possible. Finally, problem-specific valid inequalities are unlikely to be of use, as they are typically aimed at tightening the formulation of the problem-specific feasible region $\mathcal{X}$ and do not necessarily cut off fractional range-violating solutions.

We finish the discussion on valid formulations with a small example, illustrating how a nontrivial gap between the optimal solution and the LP relaxation bound can exist even when $\mathcal{X}_{LP}$ equals the convex hull of $\mathcal{X}$.

Example 6.5. *Consider three columns x_1, x_2, and x_3 with payoffs $p_1 = 1$, $p_2 = 2$, and $p_3 = 3$, respectively. In any feasible solution, two out of three columns are selected, i.e., $\mathcal{X} = \{(1,1,0), (1,0,1), (0,1,1)\}$. Clearly, this problem has two optimal solutions, $(1,1,0)$ and $(0,1,1)$, both of which have a range of one.*

We now analyse the dual bound of a valid formulation for this problem. Inspired by constraints (6.3), we consider the following mixed-binary linear program:

$$\min_{\boldsymbol{x},\eta,\gamma} \quad \eta - \gamma \tag{6.4a}$$

$$s.t. \quad x_1 + x_2 + x_3 = 2 \tag{6.4b}$$

$$p_i x_i \leq \eta \qquad \forall i \in [3] \tag{6.4c}$$

$$3 + (p_i - 3) x_i \geq \gamma \qquad \forall i \in [3] \tag{6.4d}$$

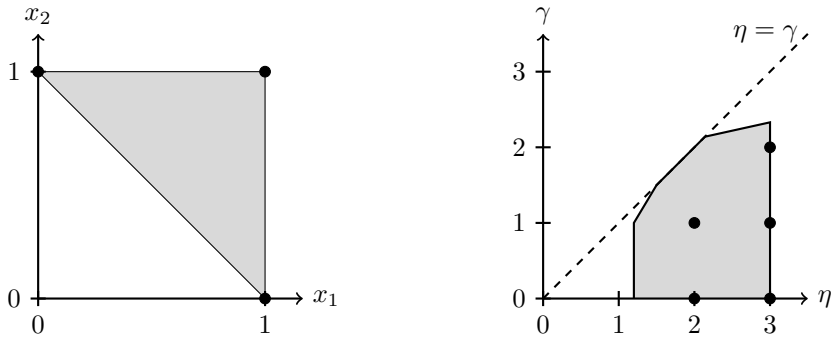
$$\eta \geq \gamma \tag{6.4e}$$

$$x_1, x_2, x_3 \in \{0, 1\} \tag{6.4f}$$

$$\eta, \gamma \in [0, 3]. \tag{6.4g}$$

Note that we have bounded the domain of η and γ and added the valid inequality $\eta \geq \gamma$.

Figure 6.2 illustrates the feasible region of model (6.4). As seen in Figure 6.2(a), the formulation is a perfect formulation for $\mathcal{X}$, i.e., the set of $\boldsymbol{x} \in \mathcal{X}_{LP}$ are exactly those solution vectors in the convex hull of solutions in $\mathcal{X}$. Nonetheless, the set of feasible pairs of (η, γ), as shown in Figure 6.2(b), contain a solution with $\eta = \gamma$. In particular, solution $(x_1, x_2, x_3, \eta, \gamma) = (1/2, 1, 1/2, 2, 2)$ is feasible and has range zero.



(a) Projection onto space of (x_1, x_2). **(b)** Projection onto space of (η, γ).

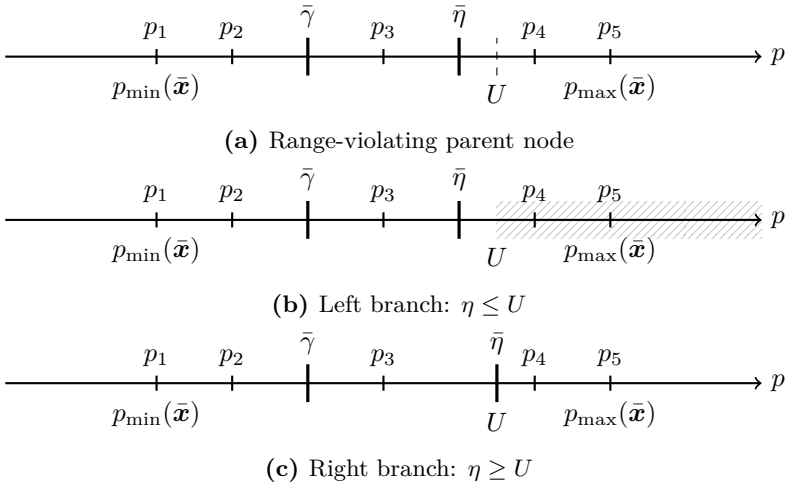
Figure 6.2: Feasible region in the space of (x_1, x_2) (left) and (η, γ) (right). Integer-feasible points are indicated by black dots, and the feasible region of the LP relaxation is indicated in grey.

6.3 Range Branching

We propose *range branching*, a generic branching strategy for range minimisation problems. We define the range branching rule, to be used on top of problem-specific branching schemes, in Section 6.3.1, and show some desirable properties of range branching in Section 6.3.2. In Section 6.3.3, we discuss some practical considerations regarding the implementation of range branching.

6.3.1 Range Branching Rule

The goal of range branching is to eliminate all range-violating fractional solutions from consideration and thereby improve the dual bound in branch-and-price algorithms for range minimisation problems. To this end, we propose to (i) branch on the values of η and γ and (ii) make use of a combination of variable bounding and variable fixing.



(a) Range-violating parent node

(b) Left branch: $\eta \leq U$

(c) Right branch: $\eta \geq U$

Figure 6.3: Example of range branching applied to a range-violating solution. The striped rectangle indicates columns, whose payoffs are above the threshold, that have been fixed to zero.

We now formally define the range branching rule. Consider a fractional solution $(\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma})$ encountered at any node in the branch-and-price tree. Recall that $p_{\min}(\bar{\boldsymbol{x}}) = \min_{i:\bar{x}_i>0} p_i$ and $p_{\max}(\bar{\boldsymbol{x}}) = \max_{i:\bar{x}_i>0} p_i$. When this solution is range-respecting, i.e., $\bar{\gamma} \leq p_{\min}(\bar{\boldsymbol{x}}) \leq p_{\max}(\bar{\boldsymbol{x}}) \leq \bar{\eta}$, our branching rule does not create any branches. Instead, the problem-specific branching scheme is invoked to construct branches that cut-off the incumbent fractional solution. Otherwise, suppose that the incumbent fractional solution $(\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma})$ is range-violating, for example, $\bar{\eta} < p_{\max}(\bar{\boldsymbol{x}})$. We cut off this solution by branching on η. In particular, we select a cutoff value $U \in (\bar{\eta}, p_{\max}(\bar{\boldsymbol{x}}))$ and create two child nodes. In the left child node, we impose $\eta \leq U$. In addition, in the left child node and its descendants, we apply variable fixing by setting to zero all x_i for which $p_i > U$ and prevent the generation of columns with payoffs $p_i > U$ in the pricing problem. In other words, we apply the locally valid inequality $\sum_{i:p_i>U} x_i = 0$ in the left node. In the right child node, we only impose $\eta \geq U$. We have experimented with enforcing the locally valid inequality $\sum_{i:p_i\geq U} x_i \geq 1$, but this did not improve the computational performance. The case $\bar{\gamma} > p_{\min}(\bar{\boldsymbol{x}})$ is treated similarly. This branching rule preserves all range-respecting solutions, but cuts off range-violating incumbent solutions. Note that range branching is to be applied at every node in the branch-and-price tree, even those where problem-specific branching decisions are already being enforced.

Figure 6.3 illustrates the range branching rule with a simple example. Figure 6.3(a) presents a feasible solution consisting of five columns with payoffs p_1 to p_5, respectively. This solution is range-violating since $p_4, p_5 > \bar{\eta}$, although a similar violation can be detected for $\bar{\gamma}$. As such, range branching is applied with cutoff value $U \in (\bar{\eta}, p_4)$. The left and right branches created through range branching are presented in Figures 6.3(b) and 6.3(c), respectively. In the left branch, we impose $\eta \leq U$, and forbid the use of columns with payoffs above U, as indicated with the striped rectangle. In particular, columns with payoff p_4 and p_5 are fixed to zero. In the right branch, we impose $\eta \geq U$, ensuring that the objective value of this child node is strictly larger than that of its parent.

With proper choices of the cutoff values (see Section 6.3.3), the use of this branching rule has several effects on the branch-and-price algorithm. First, we typically observe that the branch in which variable fixing is applied becomes infeasible and can be pruned, especially in early branching iterations. The branch where only variable bounding on η and γ is used encounters a higher objective value and thereby drives up the overall LP bound. Second, the variable fixing has an ambiguous effect on the efficiency of the pricing problems. On the one hand, incorporating the constraints $p_i \leq U$ or $p_i \geq L$ restricts the feasible region of the pricing problem, which can yield a speed-up. On the other hand, actually enforcing this constraint can complicate the pricing problems, e.g., by prohibiting the use of efficient dominance rules in labelling algorithms. Which effect dominates the other is highly specific to the problem and the payoff structure under consideration.

6.3.2 Properties of Range Branching

We define a *range-branching-free (RBF) node solution* to be any solution vector $\boldsymbol{z} = (\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma})$ encountered at any node in the branch-and-bound tree for which the range branching rule does not detect any branching opportunities. We refer to the corresponding node as *RBF node*. We say $\boldsymbol{z}$ is a RBF node solution with respect to $\boldsymbol{w}$ when it is obtained by solving the range formulation defined by $\boldsymbol{w}$. By construction, all RBF node solutions are range-respecting.

Proposition 6.6 (RBF Node Solutions are Range-Respecting)**.** *Let $\boldsymbol{z} = (\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma})$ be a RBF node solution with respect to some valid $\boldsymbol{w}$. Then, $\boldsymbol{z}$ is range-respecting.*

Proof. Suppose that $\boldsymbol{z}$ is not range-respecting. Then, range branching can be invoked. This contradicts the fact that the node is RBF. Hence, $\boldsymbol{z}$ must be range-

respecting. □

It follows that any RBF node solution is feasible in the LP relaxation of any valid formulation for the range minimisation problem.

Corollary 6.7 (Formulation-Independence)**.** *Let $\boldsymbol{z} = (\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma})$ be a RBF node solution with respect to some valid $\boldsymbol{w}$. Then, $\boldsymbol{z} \in \mathcal{Z}_{LP}(\boldsymbol{w}')$ for all valid $\boldsymbol{w}'$ that satisfy Assumption 6.3.*

Proof. Let $\boldsymbol{w}'$ be valid. By Definition 6.2 and Assumption 6.3, the LP-relaxation of the formulation defined by $\boldsymbol{w}'$ admits all range-respecting $\boldsymbol{z} = (\boldsymbol{x}, \eta, \gamma)$ for which $\boldsymbol{x} \in \mathcal{X}_{LP}$. Solution $\boldsymbol{z}$ is range-respecting by Proposition 6.6, and $\boldsymbol{x} \in \mathcal{X}_{LP}$ holds since $\boldsymbol{w}$ is valid. Hence, $\boldsymbol{z} \in \mathcal{Z}_{LP}(\boldsymbol{w}')$. □

As a result, the objective value at any RBF node is at least as high as the best possible root node bound of any valid formulation that admits all fractional range-respecting solutions for range minimisation.

Proposition 6.8. *Let $\boldsymbol{z} = (\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma})$ be a RBF node solution with respect to some valid $\boldsymbol{w}$, and denote its objective value by $v_{RBF} = \bar{\eta} - \bar{\gamma}$. Moreover, let $v^*_{LP}(\boldsymbol{u}) = \min_{(\boldsymbol{x},\eta,\gamma)\in\mathcal{Z}_{LP}(\boldsymbol{u})}(\eta - \gamma)$ denote the LP bound of the valid formulation defined by $\boldsymbol{u}$. Then $v_{RBF} \geq v^*_{LP}(\boldsymbol{w}')$ for all valid $\boldsymbol{w}'$ that satisfy Assumption 6.3.*

Proof. The result directly follows from Corollary 6.7. □

Recall that the LP lower bound in branch and price is defined as the minimum objective over all leaf nodes in the branch-and-bound tree. At some point in the course of the branch-and-price algorithm, all leaf nodes in the branch-and-bound tree will be RBF nodes or descendants of RBF nodes. From this point onward, the LP lower bound will be at least as high as the best possible root node bound of any valid formulation. The rationale behind range branching is that this tighter dual bound will reduce the number of nodes to explore in the branch-and-price tree and therefore result in an overall reduction in computing time.

6.3.3 Practical Considerations

Cutoff Values As discussed in Section 6.3.1, the success of range branching can be partially explained by its differential effect on the child branches it creates. In one

branch, we apply both variable bounding and fixing, after which we can typically prune by infeasibility. In the other branch, we apply variable bounding, thereby driving up the LP bound of this branch. These effects are likely to be small, however, when cutoff values close to $\bar{\gamma}$ and $\bar{\eta}$ are used.

In practice, we find that the following relaxation of range branching works well. Let $\alpha \in [0, 1)$. At a node with incumbent solution $(\bar{\boldsymbol{x}}, \bar{\eta}, \bar{\gamma})$, set $U_\alpha = (1 + \alpha)\bar{\eta}$ and $L_\alpha = (1 - \alpha)\bar{\gamma}$. We check whether columns with payoff above U_α or below L_α are used, i.e., when range violations of more than a factor α occur. In this case, we perform range branching with U_α and L_α as cutoff values. Otherwise, we invoke the problem-specific branching scheme. A value of $\alpha = 0.025$ seems to perform well in our experiments. The results in Section 6.3.2 hold only for the case of $\alpha = 0$, since only then all range-violating solutions are removed through range branching. From a computational point of view, however, it is interesting to consider layered branching schemes with decreasing values of α.

Further Improvements The default range branching scheme can be refined in several ways. First, it is clear that the branching scheme can be tightened when all p_i are integer-valued (e.g., branch by creating nodes $\eta \leq U$ and $\eta \geq U + 1$ with integer-valued U). Second, one could incorporate a restricted master heuristic throughout the branch-and-price algorithm and use valid upper bounds on the range to locally restrict the domain of η and γ. Third, if one uses big-M constraints to model the minimum, as in (6.3), this value of M can be tightened throughout the course of the algorithm based on the variable domain of γ. Fourth, our theoretical analysis suggests that it is possible to solve range minimisation problems through range branching without the use of auxiliary constraints of the type (6.1c), i.e., without an explicit valid formulation.

6.4 Fair Capacitated Vehicle Routing Problem

In this section, we introduce an application of the range minimisation problem which we call the *fair capacitated vehicle routing problem (F-CVRP)*, a fairness-oriented version of the capacitated vehicle routing problem.

Definition 6.9 (Fair Capacitated Vehicle Routing Problem)**.** *Let C be a set of customers, where customer $i \in C$ has integer demand $d_i > 0$. Let $G = (\{0\} \cup C, A)$ be the complete directed graph on the set of customers and a central depot, and let c_a*

and p_a denote the cost and distance of traversing arc $a \in A$, respectively. Moreover, let K homogeneous vehicles with capacity Q be given, and let B be an upper bound on the total cost of all routes. A set of K routes is feasible when each route starts and ends at the depot, each customer is visited by exactly one route, the sum of customer demands along each route does not exceed Q, and the total cost of traversed arcs does not exceed B. The goal of the fair capacitated vehicle routing problem is to select a feasible set of routes that minimises the range of the route distances.

We include an upper bound on the overall route cost to ensure that solutions do not deteriorate too much compared to the cost-efficient solution. Without such a bound, the length of the shortest route can be artificially lengthened to increase the minimum and thereby reduce the range. In practice, a tight efficiency bound leaves little room for such routes, but there is no guarantee that selected routes are TSP-optimal: shorter routes visiting the same set of customers in different order might exist.

We present two mathematical formulations of F-CVRP in Section 6.4.1 and present a branch-and-price algorithm with range branching in Section 6.4.2. We describe the set-up of our experiments in Section 6.4.3 and discuss the results in Section 6.4.4. We empirically study the computational cost of enforcing TSP-optimality in the F-CVRP, and analyse whether our method can effectively solve this problem variant in Section 6.4.5.

6.4.1 Mathematical Formulation

We present two formulations for the F-CVRP. The vehicle-index formulation explicitly assigns routes to vehicles. The last-customer formulation does not require a vehicle index, but relies on the observations that (i) each route has a unique last customer and (ii) no customer can appear as last customer on more than one route (Bianchessi et al., 2023). We observe in preliminary experiments that the direct branch and price implementation of both formulations suffers from poor LP relaxations.

Vehicle-Index Formulation Let R denote the index set of feasible routes. For each $r \in R$, let $c_r = \sum_{(i,j)\in r} c_{ij}$ and $p_r = \sum_{(i,j)\in r} p_{ij}$ denote the cost and the distance of route r, respectively, and let binary parameter a_{ir} indicate whether customer i is visited by route r. We introduce the binary decision variable x_{rk}, indicating whether or not route r is assigned to vehicle k. Then, the following mixed-binary linear

program is a valid formulation of F-CVRP:

$$\begin{aligned}
\min \quad & \eta - \gamma && && (6.5a)\\
\text{s.t.} \quad & \sum_{k\in[K]} \sum_{r\in R} a_{ir} x_{rk} = 1 && \forall i \in C && (6.5b)\\
& \sum_{k\in[K]} \sum_{r\in R} c_r x_{rk} \leq B && && (6.5c)\\
& \sum_{r\in R} x_{rk} = 1 && \forall k \in [K] && (6.5d)\\
& \sum_{r\in R} p_r x_{rk} \leq \eta && \forall k \in [K] && (6.5e)\\
& \sum_{r\in R} p_r x_{rk} \geq \gamma && \forall k \in [K] && (6.5f)\\
& x_{rk} \in \{0,1\} && \forall k \in [K],\ r \in R. && (6.5g)
\end{aligned}$$

Here, constraints (6.5e)-(6.5f) are essentially special cases of constraints (6.2), based on the fact that each vehicle always performs exactly one route. Due to the symmetry of the formulation in index k, this formulation always admits a fractional solution with objective value 0 (by setting $x_{r1} = x_{r2} = \ldots = x_{rK}$ for all $r \in R$). There exist techniques that partially break the symmetry among vehicles, but completely eliminating this symmetry in branch and price remains challenging (Darvish et al., 2020).

Last-Customer Formulation We use the same parameters as in the vehicle-index formulation. In addition, for each route $r \in R$, we let the binary parameter b_{ir} indicate whether customer i is the last customer on route r. We introduce the binary decision variable x_r, indicating whether or not route r is selected. Then, the following mixed-binary linear program is a valid formulation of F-CVRP:

$$\begin{aligned}
\min \quad & \eta - \gamma && && (6.6a)\\
\text{s.t.} \quad & \sum_{r\in R} a_{ir} x_r = 1 && \forall i \in C && (6.6b)\\
& \sum_{r\in R} c_r x_r \leq B && && (6.6c)\\
& \sum_{r\in R} x_r = K && && (6.6d)\\
& \sum_{r\in R} p_r b_{ir} x_r \leq \eta && \forall i \in C && (6.6e)
\end{aligned}$$

$$M\left(1-\sum_{r\in R} b_{ir}x_r\right)+\sum_{r\in R} p_r b_{ir}x_r \geq \gamma \qquad \forall i \in C \tag{6.6f}$$

$$x_r \in \{0,1\} \qquad \forall r \in R. \tag{6.6g}$$

Constraints (6.6e)-(6.6f) are similar in spirit to constraints (6.3), based on the observation that each customer can appear as the last customer on at most one route. Big-M constraints (6.6f) in particular contribute to poor LP bounds. In addition, fractional solutions with low objective value can be obtained by selecting two copies of each route with equal value: one in forward orientation, and one in backward orientation (visiting all customers in reversed order). This issue, however, can be largely mitigated by enforcing that the index of the last customer exceeds that of the first customer within the pricing problem.

It is readily seen that both formulations are valid.

Proposition 6.10 (Valid F-CVRP Formulations)**.** *Formulations (6.5) and (6.6) are valid formulations of F-CVRP.*

6.4.2 Branch and Price

In the following, we present a branch-and-price algorithm for solving the vehicle-index formulation, pointing out differences with the last-customer formulation whenever applicable. We refer to Costa et al. (2019) for a detailed introduction to branch and price for vehicle routing problems.

Pricing Problem Let $\boldsymbol{\kappa}, \lambda, \boldsymbol{\mu}, \boldsymbol{\nu}$, and $\boldsymbol{\pi}$ denote the dual variables of constraints (6.5b)-(6.5f), respectively, of the vehicle-index formulation (6.5). The reduced cost of route r of vehicle k is then given by

$$RC(x_{rk}) = -\mu_k - \sum_{i\in C} \kappa_i a_{ir} - \lambda c_r - (\nu_k + \pi_k)p_r \tag{6.7a}$$

$$= -\mu_k - \sum_{(i,j)\in r} (\kappa_j + \lambda c_{ij} + (\nu_k + \pi_k)p_{ij}). \tag{6.7b}$$

The pricing problem consists of finding a feasible route with negative reduced cost. The reduced cost can be easily decomposed on the arcs of G, and hence the pricing problem can be formulated as a series of elementary shortest path problems with resource constraints. There is a single pricing problem per vehicle, and resource constraints model the vehicle capacity constraint. We can partly break symmetry by

enforcing that customer i is served by a vehicle with index $k \leq i$, i.e., by removing customer i from all pricing problems where $k > i$. In the last-customer formulation, there is a single pricing problem per customer.

Pricing Algorithm As typically done in branch-and-price algorithms for vehicle routing problems, we solve the pricing problem using a labelling algorithm (Costa et al., 2019). Each label represents a partial path from the depot. The algorithm is initialised with a single dummy label at the depot. For each remaining label, its partial path is extended in all possible directions. For each extended label, we check whether it corresponds to a potentially feasible route completion. To reduce the number of labels, we check whether an extended label is dominated by some existing label, or whether it dominates an existing label. Dominated labels are removed from consideration. Labels returning to the depot with negative reduced cost are stored in memory. We now discuss the different components of the labelling algorithm in more detail.

A label $L = (v(L), c(L), q(L), \Pi(L))$ is a tuple representing a partial path from the depot to customer $v(L)$, with reduced cost $c(L)$, load $q(L)$, and set of previously visited customers $\Pi(L)$. In practice, each label also stores its predecessor to enable backtracking. It is feasible to extend label L along arc $(v(L), j)$ if $j \notin \Pi(L)$ and $q(L) + d_j \leq Q$, i.e., when customer j has not yet been visited and there is sufficient remaining capacity to cover demand d_j. In this case, the extended label L' is given by $L' = (j, c(L) - (\kappa_j + \lambda c_{ij} + (\nu_k + \pi_k)p_{ij}), q(L) + d_j, \Pi(L) \cup \{j\})$. A label corresponds to a route with negative reduced cost when $v(L) = 0$ and $c(L) < \mu_k$.

Dominance rules are applied to reduce the number of labels to consider. Label L_1 is said to dominate L_2 when any feasible extension of L_2 can be used to extend L_1 to a feasible route with lower or equal reduced cost. The following dominance checks form a set of sufficient conditions:

$$v(L_1) = v(L_2) \ \wedge \ q(L_1) \leq q(L_2) \ \wedge \ c(L_1) \leq c(L_2) \ \wedge \ \Pi(L_1) \subseteq \Pi(L_2). \tag{6.8}$$

We use several well-known acceleration techniques to speed-up the labelling algorithm. First, to allow for efficient dominance checks, we store labels in buckets based on their final customer and demand, and only perform intra-bucket dominance checks. Second, we use bi-directional labelling, i.e., extend partial paths both starting from and ending at the depot. Finally, we use *ng*-path relaxation, where the customer memory Π is truncated to only store nodes in a local neighborhood around

the current node (Baldacci et al., 2011). While this allows for non-elementary routes, elementarity is restored through branching and the number of non-dominated labels is significantly reduced. We refer to Costa et al. (2019) for more information on these acceleration techniques.

Branching Rules For the vehicle-index formulation (6.5), we first branch on the most fractional assignment of a customer to a vehicle, i.e., on the value $\sum_{r \in R} a_{ir} x_{rk}$. When a customer is assigned to a vehicle, it is removed from the pricing problem of all other vehicles. Not assigning a customer to a vehicle is processed similarly. In case of the last-customer formulation, we instead branch on whether a customer is last on a route or not. Second, we branch on the most fractional arc. A forbidden arc is removed from all pricing problems. Arc (i, j) can be forced in a solution by removing all other arcs leaving customer i or entering customer j, with special attention paid to the case where i or j equals the depot. Our range branching rule acts on top of these problem-specific branching rules, i.e., the problem-specific branching rules are only invoked when range branching does not detect any branching opportunities.

Range Branching Under the range branching scheme, we need to be able to prevent the generation of routes with distance below L or above U in the pricing problem. As such, we must extend each label L to also store the distance $d(L)$ of the partial path. To check whether label L_1 dominates label L_2, one of the following additional dominance rules must be satisfied:

$$\begin{aligned}(L = 0 \ \wedge \ U < \infty \ \wedge \ d(L_1) \leq d(L_2)) \ \vee \ (L > 0 \ \wedge \ U = \infty \ \wedge \ d(L_1) \geq d(L_2)) \\ \vee \ (L > 0 \ \wedge \ U < \infty \ \wedge \ d(L_1) = d(L_2)) \,.\end{aligned} \tag{6.9}$$

These additional dominance rules can significantly slow down the labelling algorithm, as they potentially cause a strong increase in the number of non-dominated labels.

6.4.3 Set-Up

Similar to Matl et al. (2019), we construct instances based on CVRP instance `X64` (Uchoa et al., 2017). For each number $|C| \in \{15, 20, 25\}$, we construct 20 instances by selecting $|C| + 1$ customer locations from the original instance, where the first customer location serves as depot. Each instance has $K = 5$ vehicles with capacity

Q chosen such that all vehicles are needed to serve the demand. We set the budget B equal to 110% of the cost of the cost-efficient solution. The payoff of a route is equal to its distance, i.e., we aim to balance the length of the routes.

We consider the vehicle-index and last-customer formulations for the F-CVRP presented in Section 6.4.1, and solve both formulations using both the classical and the range branching scheme. We provide the range of the efficient solution as an upper bound to the branch-and-price algorithm, impose a time limit of one hour, and solve up to 8 pricing problems in parallel. We return at most 20 columns per pricing problem, and remove columns that have been inactive, i.e., not appeared in an optimal basis, for 20 iterations. Based on preliminary computational experiments, we implement range branching with cutoff values defined by $\alpha = 0.025$. All linear programs are solved using `CPLEX 22.1.0`.

6.4.4 Results

Table 6.1 presents the results for the three sets of instances. For each combination of number of customers, formulation, and branching scheme, we present the number of instances solved to optimality within the time limit, the computing time, the optimality gap, the number of branch-and-bound nodes, and the percentage-wise reduction in range of the best solution found compared to the most cost-efficient solution (Δ). For each number of customers, the results are averaged over all 20 instances. In case not all instances are solved, the computing times and number of branch-and-bound nodes provide lower bounds on the true values.

The results in Table 6.1 show that range branching outperforms classical branching in all cases. For all sizes and formulations, range branching leads to a strong reduction in computing time, optimality gap, and number of branch-and-bound nodes. In addition, it allows us to solve 14 more instances with $|C| = 25$ customers, the largest instances we consider. The last-customer formulation seems to particularly benefit from range branching, an effect that appears to be largely driven by a huge reduction in the size of the branch-and-bound tree. While it is not competitive with the vehicle-index formulation under the classical branching scheme, it dominates this formulation on all instance sets when range branching is used. Finally, we note that fairness gains up to 54% can be attained compared to the most cost-efficient solution.

Figure 6.4 displays the progression of the lower and upper bounds, for each combination of number of customers, formulation, and branching scheme, throughout

Table 6.1: Computational performance of vehicle-index and last-customer formulations with classical and range branching scheme.

$\|C\|$	Formulation	Branching	# Solved	Time (s)	Gap (%)	# Nodes	Δ (%)
15	Vehicle	Classical	20/20	207.1	0	6,619.4	49.4
		Range	20/20	175.4	0	5,687.8	49.4
	Customer	Classical	17/20	916.9	4.0	15,726.8	48.7
		Range	20/20	3.6	0	385.8	49.4
20	Vehicle	Classical	9/20	2,769.5	33.3	14,446.9	34.2
		Range	16/20	1,099.0	12.3	6,386.6	44.8
	Customer	Classical	1/20	3,494.3	67.4	21,279.1	25.4
		Range	19/20	290.6	0.0	3,430.9	53.5
25	Vehicle	Classical	0/20	3,600.0	94.0	11,349.4	8.6
		Range	6/20	3,053.2	25.2	983.8	38.6
	Customer	Classical	0/20	3,600.0	94.1	13,220.6	5.1
		Range	14/20	1,741.1	5.8	1,728.1	54.0

the course of the branch-and-price algorithm. These relative values are expressed in terms of the best known lower bound, and subsequently averaged over all instances. We find that the performance of range branching is largely driven by its ability to quickly drive up the lower bound to near its final value. This is well in line with the theoretical properties of range branching derived in Section 6.3.2. On the larger instances, the lower bounds of the classical branching scheme are still slowly increasing after an hour of computation, whereas the bounds with range branching reach near-optimal values early in the process. This is especially true for the last-customer formulation that attains near-optimal lower bounds in the first few iterations of the algorithm. This might explain why it outperforms the vehicle-index formulation. In addition to improved lower bounds, range branching also has a positive effect on the upper bound evolution throughout the course of branch and price. Hence, range branching appears to be the method of choice even on large instances that are not solved to optimality within the time limit.

6.4.5 TSP-Optimal Routes

In our setup, the range of a solution can potentially be reduced by (i) exchanging customers across routes or (ii) changing the customer order along a route. The latter can be undesirable in practice, as the resulting routes might violate TSP-optimality. That is, the routes might deviate from the shortest tour visiting all customers and the depot, resulting in an unnecessarily large distance to be travelled. In this section, we conduct additional experiments to show that directly enforcing TSP-optimality

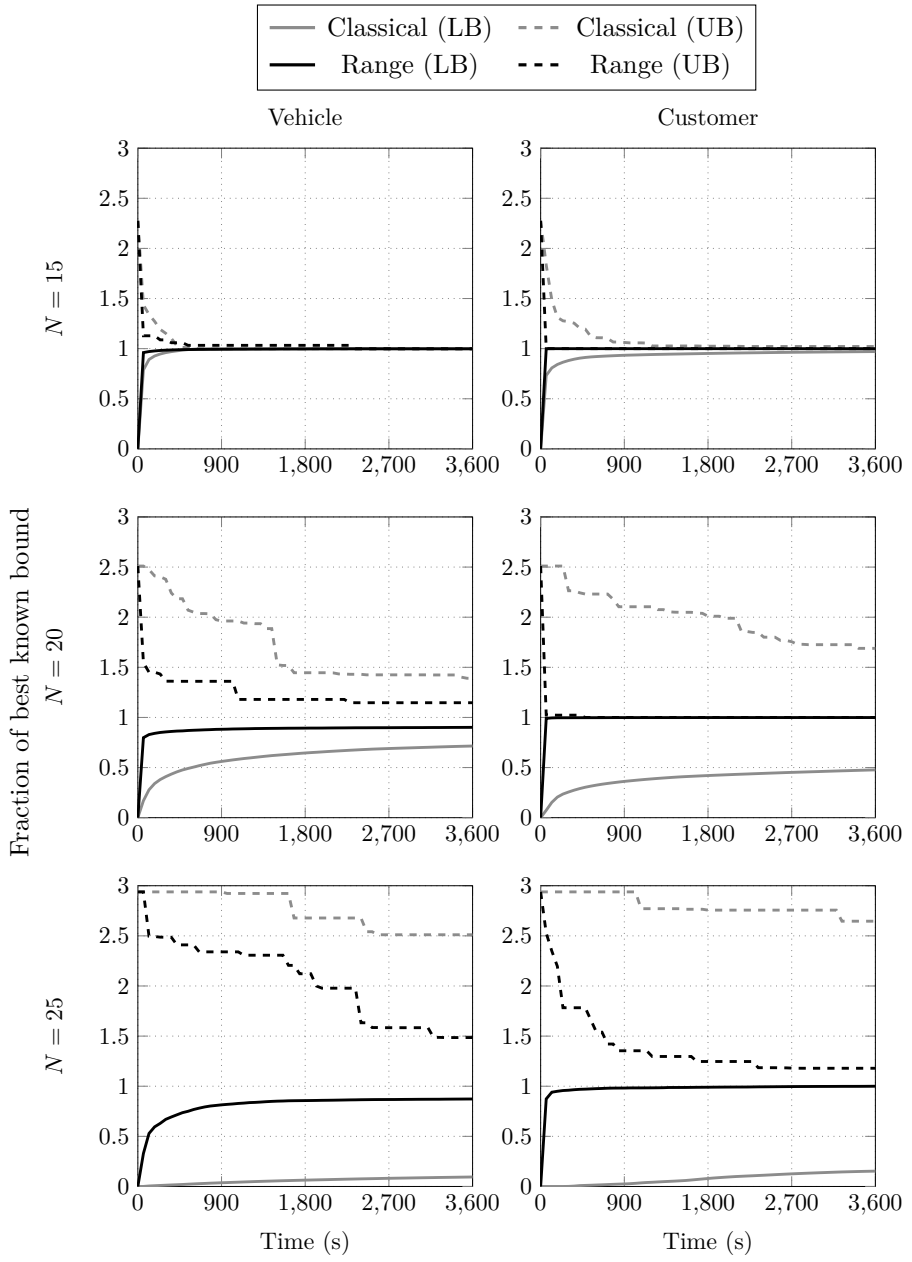


Figure 6.4: Progression of lower and upper bound of vehicle-index formulation (left) and last-customer formulation (right) with classical and range branching scheme.

is computationally demanding, but a small postprocessing step is sufficient for our method to perform well on the problem restricted to TSP-optimal routes.

We start by comparing the performance of range branching on the standard F-CVRP and the F-CVRP restricted to TSP-optimal routes. Since the impact of range branching for the TSP-optimal case is similar to the general case of Section 6.4.4, we do not consider the classical branching scheme here. The first algorithm, denoted by 'General', is the range-B&P algorithm applied to the last-customer formulation. This algorithm allows TSP-violating routes. The second algorithm, denoted by 'TSP', is range-B&P applied to the vehicle-index formulation. Here, we modify the dominance rules to ensure that only TSP-optimal routes are generated by the pricing algorithm. In particular, one partial route dominates another when it covers the same customers in a shorter or equal distance. Note that this introduces a significant weakening of the dominance rules. We compare the algorithms using the same instances and parameter settings as in Section 6.4.3. Since the degree to which routes violate TSP-optimality is likely to depend on the total route budget, we vary the budget B to be 101%, 105%, and 110% of the cost-efficient solution.

Table 6.2 presents the computational results of these experiments. For each combination of number of customers, route budget, and algorithm, we present the number of instances solved to optimality within the time limit, the computing time, and the optimality gap. For each combination of number of customers and budget, the results are averaged over all 20 instances. In case not all instances are solved, the computing times and number of branch-and-bound nodes provide lower bounds on the true values. A dash indicates that the root node relaxation could not be solved to optimality within the time limit.

The results in Table 6.2 show the additional computational complexity that arises when enforcing TSP-optimality. The performance of the range-B&P algorithm with general routes is in line with the results of Section 6.4.4. A majority of the instances is solved to optimality, and the route budget does not appear to have any significant effect on computational performance. However, enforcing TSP-optimal routes comes at a severe computational cost. Only a handful of 20-customer instances is solved to optimality, and all of the 25-customer instances exceed the time limit before solving the root node. This is not surprising, given that the modified dominance rules imply a near-enumeration of all possible routes by the labelling algorithm. We conclude that directly enforcing TSP-optimality is intractable for all non-trivial instance sizes.

Table 6.2: Computational performance of range branching with only TSP-optimal routes and with general routes.

		General			TSP		
$\|C\|$	B (%)	# Solved	Time (s)	Gap (%)	# Solved	Time (s)	Gap (%)
15	101%	20/20	8.0	0	17/20	824.4	0.9
	105%	20/20	70.6	0	16/20	956.9	3.6
	110%	20/20	3.6	0	20/20	476.6	0
20	101%	18/20	576.9	1.1	3/20	3,479.4	10.0
	105%	19/20	294.0	0.0	0/20	3,600.0	27.5
	110%	19/20	290.6	0.0	0/20	3,600.0	41.5
25	101%	13/20	1,865.1	7.1	-	-	-
	105%	12/20	1,841.4	14.2	-	-	-
	110%	14/20	1,741.1	5.8	-	-	-

Fortunately, the range-B&P algorithm with general routes provides an alternative way of solving the F-CVRP with only TSP-optimal routes. In particular, any lower bound for the problem with general routes is a valid lower bound for the problem with only TSP-optimal routes. Second, a solution containing non-TSP routes can be easily converted to a feasible one by rearranging the order of customers in each route, for example, by using a compact MILP formulation of the TSP. This yields a valid upper bound.

Table 6.3 compares the bounds obtained in this way with the bounds from the TSP-algorithm. For all number of customers and budgets, it reports the number of instances for which the converted solutions are provably optimal, the relative percentage by which the range was underestimated by allowing non-TSP routes (Δ_R), the relative percentage increase in lower bound compared to the TSP-algorithm (Δ_{LB}), and the relative percentage reduction in upper bound compared to the TSP-algorithm (Δ_{UB}). Finally, it reports the optimality gap of the converted solutions, computed using the best-known lower bound from both methods. A dash indicates that the TSP-algorithm did not compute any valid bounds within the time limit.

The results of Table 6.3 show that the proposed method forms an effective heuristic for the F-CVRP with only TSP-optimal routes, and even solves many of the problem instances to optimality. It provides improved bounds, indicated by positive values of Δ_{LB} and Δ_{UB}, on nearly all instances. The only exception is the case with 15 customers and a large route budget of 110%, in which (i) the instance is sufficiently small as to be solved to optimality by the TSP-algorithm and (ii) the budget is

Table 6.3: Analysis of bounds obtained by converting general routes to TSP-optimal routes.

$\|C\|$	B (%)	# TSP-Optimal	Δ_R (%)	Δ_{LB} (%)	Δ_{UB} (%)	Gap (%)
15	101%	19/20	0.0	0.3	0.6	0.0
	105%	18/20	1.2	1.4	1.7	0.3
	110%	13/20	7.7	-4.8	-3.8	3.2
20	101%	17/20	0.3	6.6	4.1	1.2
	105%	8/20	4.1	5.9	20.7	3.1
	110%	4/20	13.1	1.3	28.1	10.6
25	101%	9/20	0.4	-	-	7.6
	105%	4/20	2.7	-	-	16.9
	110%	3/20	14.9	-	-	21.3

sufficiently large as to allow for many TSP-violated routes in the general model. The increase in range after converting routes to TSP-optimal ones, i.e., the degree to which TSP-violating routes are used, is generally negligible for smaller route budgets, but becomes sizeable as the route budget reaches 110% of the cost-efficient solution. Similarly, the optimality gaps are well below 10% except for the larger instances with high route budgets. Given the inefficiency of the modified labelling algorithm, we conclude that it is preferable to use general routes and convert them to TSP-optimal routes in a post-processing step, than to directly enforce TSP-optimality in the pricing problem. Of course, the possibility of devising an entirely new algorithm for the TSP-optimal case remains open.

6.5 Fair Generalised Assignment Problem

In this section, we introduce an application of the range minimisation problem that we call the *fair generalised assignment problem (F-GAP)*, a fairness-oriented version of the generalised assignment problem.

Definition 6.11 (Fair Generalised Assignment Problem)**.** *Consider m jobs to be assigned to n agents. Assigning job $j \in [m]$ to agent $i \in [n]$ yields a profit p_{ij} and consumes c_{ij} units of the resource of agent i. The resource capacity of agent i equals C_i. All costs and capacities are integer-valued. An assignment is feasible when the total resource consumption of each agent does not exceed his capacity, and the overall profit of all jobs is at least P. The goal of the fair generalised assignment problem is to find a feasible assignment that minimises the range of the resource consumption*

of different agents.

Branch and price is not necessarily the preferred solution method for the generalised assignment problem, as commercial solvers are typically faster in solving the regular, cost-oriented problem. We show that, despite this fact, branch and price is several factors faster than solving a standard compact formulation for F-GAP. In other words, when minimising range it can be beneficial to switch to a Dantzig-Wolfe reformulation as compared to a standard compact formulation.

We present two mathematical formulations of F-GAP in Section 6.5.1 and present a branch-and-price algorithm with range branching in Section 6.5.2. We describe the set-up of our experiments in Section 6.5.3 and discuss the results in Section 6.5.4.

6.5.1 Mathematical Formulation

Compact Formulation We introduce the binary decision variable x_{ij} to indicate whether or not job j is assigned to agent i. A compact formulation of F-GAP reads as

$$\begin{aligned}
\min \quad & \eta - \gamma && && \text{(6.10a)}\\
\text{s.t.} \quad & \sum_{i\in[n]} \sum_{j\in[m]} p_{ij}x_{ij} \geq P && && \text{(6.10b)}\\
& \sum_{i\in[n]} x_{ij} = 1 && \forall j \in [m] && \text{(6.10c)}\\
& \sum_{j\in[m]} c_{ij}x_{ij} \leq C_i && \forall i \in [n] && \text{(6.10d)}\\
& \sum_{j\in[m]} c_{ij}x_{ij} \leq \eta && \forall i \in [n] && \text{(6.10e)}\\
& \sum_{j\in[m]} c_{ij}x_{ij} \geq \gamma && \forall i \in [n] && \text{(6.10f)}\\
& x_{ij} \in \{0,1\} && \forall i \in [n],\ j \in [m]. && \text{(6.10g)}
\end{aligned}$$

Dantzig-Wolfe Reformulation We define an assignment as a set of jobs that can be performed by a single agent without violating its resource constraint. Formally, an assignment $a = (i, S)$ is a tuple denoting that agent i performs all the jobs in set $S \subseteq \{1, \ldots, m\}$ with $\sum_{j\in S} c_{ij} \leq C_i$. Let A be the set of feasible assignments, with

$A_i \subseteq A$ denoting the set of feasible assignments to agent i. For each $a \in A$, let p_a denote the profit of assignment a, c_a the resource consumption of assignment a, and let k_{aj} be a binary parameter indicating whether job j is included in assignment a. We introduce the binary decision variable y_a indicating whether or not assignment a is selected. A Dantzig-Wolfe reformulation of F-GAP then reads as

$$\begin{aligned}
\min \quad & \eta - \gamma && && \text{(6.11a)}\\
\text{s.t.} \quad & \sum_{a \in A} p_a y_a \geq P && && \text{(6.11b)}\\
& \sum_{a \in A} k_{aj} y_a = 1 && \forall j \in [m] && \text{(6.11c)}\\
& \sum_{a \in A_i} y_a \leq 1 && \forall i \in [n] && \text{(6.11d)}\\
& \sum_{a \in A_i} c_a y_a \leq \eta && \forall i \in [n] && \text{(6.11e)}\\
& \sum_{a \in A_i} c_a y_a \geq \gamma && \forall i \in [n] && \text{(6.11f)}\\
& y_a \in \{0,1\} && \forall a \in A. && \text{(6.11g)}
\end{aligned}$$

Constraints (6.11e)-(6.11f) are formulated in line with general constraints (6.2), following from the fact that each agent is always assigned exactly one assignment. The reformulation contains an exponential number of variables, but can be solved efficiently by branch and price (Savelsbergh, 1997). It is easy to see that the reformulation is a valid formulation of F-GAP.

Proposition 6.12 (Valid F-GAP Formulation)**.** *Formulation (6.11) is a valid formulation of F-GAP.*

Recall that the concept of a valid formulation does not apply to the compact formulation (6.10), since it is not a column-based formulation and therefore does not meet the conditions of Definition 6.2.

6.5.2 Branch and Price

We largely follow the branch-and-price implementation of Savelsbergh (1997), and show how to augment it with our range branching rule.

Pricing Problem and Algorithm Let $\kappa, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\nu}$, and $\boldsymbol{\pi}$ denote the dual variables of constraints (6.11b)-(6.11f), respectively. The reduced cost of assignment $a \in A_i$ is

then given by

$$RC(y_a) = -\mu_i - \kappa p_a - \sum_{j\in[m]} \lambda_j k_{aj} - c_a(\nu_i + \pi_i) \tag{6.12a}$$

$$= -\mu_i - \sum_{j\in[m]} (\lambda_j + \kappa p_{ij} + (\nu_i + \pi)c_{ij})k_{aj}. \tag{6.12b}$$

The pricing problem consists of finding a feasible assignment with a negative reduced cost. It can be modeled as a series of knapsack problems, one per agent. Here, the knapsack problem associated with agent i has capacity C_i, and job j has weight c_{ij} and value $\lambda_j + \kappa p_{ij} + (\nu_i + \pi)c_{ij}$. We solve each knapsack problem using a custom implementation of the well-known pseudo-polynomial dynamic programming (DP) algorithm. A DP table of size $\mathcal{O}(mC_i)$ is constructed, where entry (k, c) corresponds to the maximum value that can be attained by filling a knapsack of size c with items $\{1, \ldots, k\}$. The value of the optimal knapsack is given by entry (m, C_i).

Branching Rule We branch on the most fractional assignment of a job to an agent, i.e., on the value of $\sum_{a\in A_i} k_{aj} y_a$. Note that this corresponds to branching on x_{ij} in the compact formulation. If a job is assigned to an agent, all assignments of this agent not containing the job are removed from the model and the remaining resource capacity in the pricing problem is adjusted downwards. Assignments to other agents containing this job are removed from the problem and the job is removed from the pricing problem of these agents. Not assigning a job to an agent is processed similarly. Our range branching rule acts on top of this problem-specific branching rule and is always invoked first.

Range Branching To adhere to the variable fixing strategy of the range branching rule, we need to be able to prevent the generation of assignments with resource consumption above U or below L in the pricing problem. The former is readily achieved by artificially reducing the knapsack capacity from C_i to U. The latter requires a slight adjustment of the DP table. We re-define entry (k, c) to be equal to the maximum value of a knapsack of weight *exactly* c with items $\{1, \ldots, k\}$. These entries can be computed with the same recursive relation as in the original dynamic program after a minor change in the initialisation procedure. The value of the optimal knapsack is obtained by taking the maximum over entries from (m, L) up to (m, U).

6.5.3 Set-Up

We evaluate the performance of range branching by comparing it with a classical branching scheme as well as a MILP-solver on the compact formulation. We use benchmark instances for the generalised assignment problem proposed by Chu and Beasley (1997). In particular, we consider the instances in classes A, B, and C with 100 jobs and 5, 10, and 20 agents, respectively. We compute the profit-optimal solution value P^* to these instances and require a minimum profit of $(1 - \theta)P^*$, where θ ranges from 0.25% up to 1% in steps of 0.25%. We impose a time limit of one hour, return at most one column per pricing problem, and remove columns that have been inactive for 30 iterations. We use range-branching with cutoff values corresponding to $\alpha = 0$, i.e., we branch on all solutions that are not strictly range-respecting. All linear programs, as well as the compact formulation, are solved with `CPLEX 22.1.0`.

6.5.4 Results

Table 6.4 presents the results for all instance configurations. For each combination of number of customers, profit lower bound, and solution method, we present the number of instances solved to optimality, the computing time, optimality gap, and the percentage-wise reduction in range of the best solution found compared to the most cost-efficient solution (Δ). Per row, the results are averaged over the three instances of type A, B, and C, and the final row averages over all instance configurations. In case not all instances are solved to optimality, the computing times provide lower bounds on the true values.

The results in Table 6.4 once again show the drastic improvement in computational performance following from the use of range branching. The classical branch-and-price implementation is able to solve only 7 out of 36 instances, all of which with 5 agents. When augmenting the branching scheme with our range branching rule, all instances can be solved to optimality and the average computing time is reduced by a factor larger than 20. Range-B&P also outperforms the commercial MILP-solver applied to the compact formulation: 11 more instances are solved to optimality, and the average computing time decreases by a factor of 8. The average reduction in range is two percentage points higher for range-B&P, suggesting that strictly better integral solutions are found by this method. This result is mainly driven by the instances with 10 or 20 agents. On instances with 5 agents, the MILP-solver is on average faster than range-B&P, although computing times are of the same order of

magnitude. Across all methods, we observe that a tighter profit lower bound generally leads to a reduction in computing times, logically following from the fact that the feasible region becomes smaller. Interestingly, we observe that the computing time of range-B&P often decreases as the number of agents grows. This phenomenon can largely be attributed to the lower maximum resource capacity per agent in these instances. As a result, the number of potential assignments is significantly reduced, which in turn decreases the time required to solve each node in the branch-and-price tree. Finally, we note that a large reduction in range can be attained at the cost of only a 1% profit reduction.

Table 6.4: Computational performance of compact formulation and Dantzig-Wolfe reformulation with classical and range branching scheme.

# Agents	$1-\theta$	Method	# Solved	Time (s)	Gap (%)	Δ (%)
5	0.99	MILP	3/3	117.0	0	99.0
		B&P	1/3	2,446.3	66.7	0
		Range-B&P	3/3	166.9	0	99.0
	0.9925	MILP	3/3	85.4	0	96.1
		B&P	1/3	2,487.2	63.8	0
		Range-B&P	3/3	144.8	0	96.1
	0.995	MILP	3/3	67.8	0	91.0
		B&P	2/3	1,930.5	33.3	57.8
		Range-B&P	3/3	76.3	0	91.0
	0.9975	MILP	3/3	34.6	0	70.1
		B&P	3/3	974.6	0	70.1
		Range-B&P	3/3	106.2	0	70.1
10	0.99	MILP	1/3	2,716.4	66.7	91.7
		B&P	0/3	3,600.0	100.00	1.9
		Range-B&P	3/3	52.0	0	99.7
	0.9925	MILP	2/3	1,325.2	33.3	93.4
		B&P	0/3	3,600.0	100.0	1.9
		Range-B&P	3/3	90.5	0	97.1
	0.995	MILP	2/3	1,262.4	33.3	85.5
		B&P	0/3	3,600.0	94.0	1.9
		Range-B&P	3/3	132.8	0	89.2
	0.9975	MILP	3/3	113.2	0	72.5
		B&P	0/3	3,600.0	77.6	1.9
		Range-B&P	3/3	22.8	0	72.5
20	0.99	MILP	0/3	3,600.0	100.0	88.9
		B&P	0/3	3,600.0	100.0	0
		Range-B&P	3/3	758.6	0	96.2
	0.9925	MILP	0/3	3,600.0	83.3	88.9
		B&P	0/3	3,600.0	100.0	0
		Range-B&P	3/3	164.5	0	91.3
	0.995	MILP	2/3	1,506.6	33.3	82.2
		B&P	0/3	3,600.0	98.0	0
		Range-B&P	3/3	56.7	0	83.0
	0.9975	MILP	3/3	166.7	0	63.0
		B&P	0/3	3,600.0	75.3	0
		Range-B&P	3/3	10.8	0	63.0
Average		MILP	25/36	1,220.4	29.2	85.2
		B&P	7/36	3,094.4	75.7	11.3
		Range-B&P	36/36	148.6	0	87.4

6.6 Order-Based Minimisation

Up to now, we have focused on range minimisation problems, where the goal is to minimise the difference between the largest and smallest payoff among a set of columns. From a slightly more general perspective, we can see this as minimising a linear combination of the ordered payoffs, in which we only place non-zero weight on the first and last ordered value (i.e., 1 and -1). Such order-based objective functions are frequently encountered in fairness-oriented optimisation (Tsang and Shehadeh, 2024). For example, the Gini deviation is an important fairness measure that penalises deviations between all pairs of payoffs, not just between the largest and smallest. It generalises the range and can be written as an order-based objective function. In this section, we show how the range branching rule can be generalised to order-based minimisation problems.

The remainder of this section is structured as follows. We provide a formal definition of the order-based minimisation problem as well as a mathematical formulation in Sections 6.6.1 and 6.6.2, respectively. We propose order branching, a generalisation of range branching, in Section 6.6.3, and evaluate its effectiveness by minimising the Gini deviation in a CVRP application in Section 6.6.4.

6.6.1 Problem Description

We formally define the order-based minimisation problem by making some additional assumptions on top of the setting described in Section 6.2.1. As before, we consider the problem of selecting a feasible subset of columns satisfying some linear constraints, i.e., the feasible region is contained in $\mathcal{X} = \{\boldsymbol{x} \in \{0,1\}^N : A\boldsymbol{x} \leq \boldsymbol{b}\}$. In addition, to ensure that the order-based objective is well-defined, we assume that a fixed number of exactly K columns is to be selected in any solution, where we require $K \geq 2$. This could be, for example, the number of available drivers (one per vehicle) in a vehicle routing application, or the number of agents in a generalised assignment problem. The feasible region is now given by $\mathcal{X}' = \{\boldsymbol{x} \in \mathcal{X} : \mathbf{1}^\top \boldsymbol{x} = K\} \subseteq \mathcal{X}$. For any $\boldsymbol{x} \in \mathcal{X}'$, let $\boldsymbol{z}(\boldsymbol{x}) \in \mathbb{R}^K$ denote the vector of ordered payoffs, such that $z_k(\boldsymbol{x})$ is the value of the k-th largest payoff of columns in $\boldsymbol{x}$. Let $\boldsymbol{v} \in \mathbb{R}^K$ be a weight vector on the order variables. The goal of order-based minimisation is to select a set of columns $\boldsymbol{x} \in \mathcal{X}'$ that minimises $\boldsymbol{v}^\top \boldsymbol{z}(\boldsymbol{x})$. The class of objective functions considered here encapsulates a large set of fairness measures (Tsang and Shehadeh, 2024). In particular, range minimisation corresponds to the special case with $\boldsymbol{v} = (1, 0, \ldots, 0, -1)^\top$.

6.6.2 Mathematical Formulation

We propose a simple general formulation of the order-based minimisation problem. Recall that I denotes the set of column indices. For each $i \in I$ and $k \in [K]$, introduce the auxiliary binary variable y_{ik} that equals one if column i is selected and yields the k-th largest payoff, and zero otherwise. Moreover, let continuous variable z_k denote the value of the k-th largest payoff. The order-based minimisation problem can then be formulated as follows:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \quad \boldsymbol{v}^\top \boldsymbol{z} \tag{6.13a}$$

$$\text{s.t.} \quad A\boldsymbol{x} \leq \boldsymbol{b} \tag{6.13b}$$

$$\sum_{k=1}^{K} y_{ik} = x_i \qquad \forall i \in I \tag{6.13c}$$

$$\sum_{i \in I} y_{ik} = 1 \qquad \forall k \in [K] \tag{6.13d}$$

$$\sum_{i \in I} p_i y_{ik} = z_k \qquad \forall k \in [K] \tag{6.13e}$$

$$z_k \geq z_{k+1} \qquad \forall k \in [K-1] \tag{6.13f}$$

$$\boldsymbol{x} \in \{0,1\}^N \tag{6.13g}$$

$$\boldsymbol{y} \in \{0,1\}^{NK}. \tag{6.13h}$$

Constraints (6.13c) and (6.13d) ensure that each selected column takes exactly one position in the list of ordered payoffs, and that each position in the order corresponds to exactly one column. As a result, precisely K columns are selected. Constraints (6.13d)-(6.13e) ensure that the values of z_k are correctly set, while constraints (6.13f) guarantee a consistent ordering.

We make some brief remarks on formulation (6.13). First, we are not aware of any other formulation for the order-based minimisation problem that does not require additional binary variables or big-M constraints. Second, while $\mathcal{O}(NK)$ auxiliary variables are required, this formulation is still relatively compact in the sense that only $\mathcal{O}(N+K)$ extra linear constraints are introduced (aside from bound and integrality constraints). A smaller formulation can be obtained by projecting out $\boldsymbol{x}$ and dropping constraints (6.13c). In fact, solving model (6.13) with constraints (6.13c) would necessitate simultaneous column-and-row generation. Third, it is possible to strengthen this formulation by adding valid inequalities. In particular, assum-

ing ordered payoffs $p_1 \geq p_2 \geq \ldots \geq p_N$, one can add $\mathcal{O}(NK)$ constraints of the type

$$\sum_{i=1}^{l} y_{ik} \geq \sum_{i=1}^{l} y_{i,k+1} \tag{6.14}$$

for all $k \in [K-1]$ and $l \in [N]$. In a branch-and-price algorithm, however, the use of such inequalities would imply the generation of additional rows when pricing out columns, severely complicating the pricing problem.[1] Moreover, our focus is on branching strategies rather than the strength of particular formulations. As such, we consider these inequalities to be outside the scope of this chapter.

It is clear that formulation (6.13) can be used for range minimisation by setting $\boldsymbol{v} = (1, 0, \ldots, 0, -1)^\top$. In fact, the vehicle-index formulation (6.5) and agent-based formulation (6.11) can be seen as special cases of (6.13) where $\boldsymbol{x}$ is projected out.

6.6.3 Order Branching

We present an extension of range branching to the case of order-based minimisation, which we call *order branching*. As before, we first formalise the notion of *order-violating* solutions, after which we show how to eliminate these through branching.

For notational convenience, given $\boldsymbol{y} \in [0,1]^{NK}$ and $\boldsymbol{p} \in [0, M]^N$, for each $k \in [K]$ define $p^k_-(\boldsymbol{y}) = \min_{j=1,\ldots,k} \min_{i: y_{ij}>0} p_i$ and $p^k_+(\boldsymbol{y}) = \max_{j=k,\ldots,K} \max_{i: y_{ij}>0} p_i$ to be the minimum and maximum payoff of any column assigned to order at most or at least k, respectively. In this definition, we look at the payoffs of columns beyond index k since a solution can only be order-respecting when the payoffs are in non-increasing order. By definition, we have $p^k_+(\boldsymbol{y}) \geq p^k_-(\boldsymbol{y})$. In fact, equality must hold for any feasible solution $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ to (6.13), i.e., it must hold that $z_k = p^k_-(\boldsymbol{y}) = p^k_+(\boldsymbol{y})$ for all k. As for range minimisation, we can formalise this idea by introducing the notion of *order-respecting solutions*.

Definition 6.13 (Order-Respecting Solution)**.** *Let a vector* $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{z}}) \in [0,1]^N \times [0,1]^{NK} \times \mathbb{R}^K$ *be given. We say this vector is* order-respecting *(with respect to* $\boldsymbol{p}$*) if, for all* $k \in [K]$*, it holds that* $\bar{z}_k = p^k_+(\bar{\boldsymbol{y}}) = p^k_-(\bar{\boldsymbol{y}})$*, and* order-violating *(with respect to* $\boldsymbol{p}$*) otherwise.*

[1] In the vehicle routing literature, valid inequalities that make the pricing problem significantly harder are referred to as 'non-robust cuts'.

In range-minimisation, by optimality it holds that range-violation occurs due to η being under- or γ being over-estimated. In contrast, order-violation can occur due to z_k being either under- or over-estimated, depending on the sign of entries of $\boldsymbol{v}$. In other words, a solution can be order-violating due to $\bar{z}_k > p^k_-(\bar{\boldsymbol{y}})$ or $\bar{z}_k < p^k_+(\bar{\boldsymbol{y}})$ for some k.

Similar to range branching, order branching eliminates order-violating solutions by branching on the values of z_k. Consider a fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{z}})$ encountered at any node of the branch-and-price tree. When this solution is order-respecting, our branching rule does not create any branches, but proceeds to invoke the problem-specific branching scheme. Suppose now that the solution is order-violating, i.e., there exists some index $k \in [K]$ for which, for example, $z_k > p^k_-(\boldsymbol{y})$. We choose a cutoff value $Z_k \in (p^k_-(\boldsymbol{y}), z_k)$ and create two child nodes. In the left child node, we impose $z_k \leq Z_k$, while, in the right child node, we require $z_k \geq Z_k$. Unlike in range branching, we can now enforce strong locally valid inequalities that forbid columns with payoffs above or below Z_k in *both* child nodes. In particular, in the left child node, we can enforce $\sum_{j \geq k} \sum_{i:p_i > Z_k} y_{ij} = 0$, while, in the right child node, we can impose $\sum_{j \leq k} \sum_{i \in I:p_i < Z_k} y_{ij} = 0$. As before, the valid inequalities are implemented by fixing existing columns to zero and preventing the generation of new columns in the pricing problem. The case $z_k < p^k_+(\boldsymbol{y})$ is treated similarly.

6.6.4 Computational Experiments: Gini Deviation CVRP

An important example of an order-based objective function is the Gini deviation, a frequently used inequity measure (Tsang and Shehadeh, 2024). The Gini deviation is defined as $\sum_i \sum_{j>i} |z_i - z_j|$, and penalises the difference between not only the largest and smallest payoffs, but also between intermediate payoffs. It can be represented as an order-based objective function with weight vector entries given by $v_k = (K - 2k + 1)$ for $k = 1, \ldots, K$. We evaluate the effectiveness of order branching by minimising the Gini deviation among the distances of routes in the CVRP.

The set-up of our experiments is highly similar to that of Section 6.4.3. We consider the same instances and parameter settings as before, but focus only on the vehicle-indexed formulation. This formulation naturally lends itself to the order-based objective setting, as it is essentially formulation (6.13) with $\boldsymbol{x}$ projected out. Moreover, it is not immediately clear how to correctly model the Gini deviation in the last-customer formulation. Since all vehicles are homogeneous, without loss of generality, we can assume that vehicle k performs the route with k-th largest dis-

tance. The pricing problem and algorithm described in Section 6.4.2 remain valid, although the symmetry-breaking technique presented there can no longer be used. For the sake of completeness, we present the full resulting formulation here:

$$\min \quad \boldsymbol{v}^\top \boldsymbol{z} \tag{6.15a}$$

$$\text{s.t.} \quad \sum_{k\in[K]} \sum_{r\in R} a_{ir} y_{rk} = 1 \qquad \forall i \in C \tag{6.15b}$$

$$\sum_{k\in[K]} \sum_{r\in R} c_r y_{rk} \leq B \tag{6.15c}$$

$$\sum_{r\in R} y_{rk} = 1 \qquad \forall k \in [K] \tag{6.15d}$$

$$\sum_{r\in R} p_r y_{rk} = z_k \qquad \forall k \in [K] \tag{6.15e}$$

$$z_k \geq z_{k+1} \qquad \forall k \in [K-1] \tag{6.15f}$$

$$y_{rk} \in \{0,1\} \qquad \forall k \in [K],\ r \in R. \tag{6.15g}$$

The results in Table 6.5 show that the order branching scheme significantly outperforms the classical branching scheme. For all instance sizes, the number of instances solved to optimality grows, and the computing time and optimality gap are reduced, when adding the order branching rule to the classical branching scheme. As for range branching, we observe that this effect is driven by a large reduction in the size of the branch-and-price tree. Moreover, we find that the Gini deviation is reduced by up to 50% compared to the cost-efficient solution. Interestingly, comparing the results of the vehicle-index formulation for range minimisation and order-based minimisation, we observe that it is more effective for the latter problem, being able to solve more instances to optimality with the improved branching scheme.

Table 6.5: Computational performance of vehicle-index formulation for Gini deviation with classical and order branching scheme.

$\|C\|$	Branching	# Solved	Time (s)	Gap (%)	# Nodes	Δ (%)
15	Classical	17/20	1,173.7	8.5	35,079.7	39.6
	Order	20/20	2.9	0	86.2	46.5
20	Classical	3/20	3,452.3	76.4	43,244.9	7.5
	Order	20/20	55.7	0	221.9	50.7
25	Classical	0/20	3,600.0	99.6	19,211.8	0.0
	Order	12/20	2,078.3	18.3	340.1	38.7

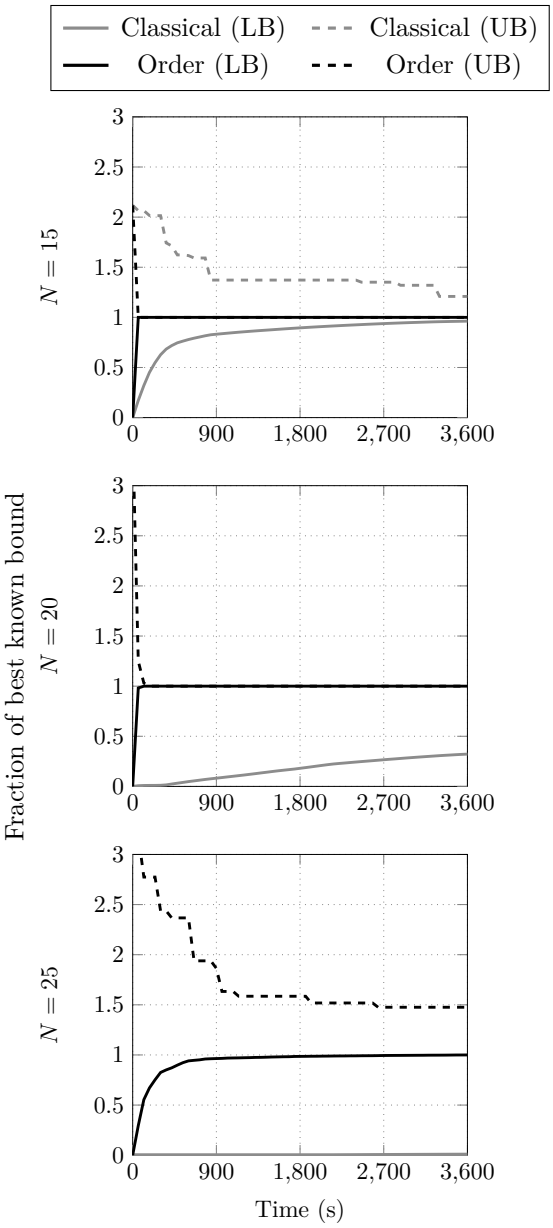


Figure 6.5: Progression of lower and upper bound of vehicle-index formulation for Gini deviation with the classical and order branching scheme.

Figure 6.5 presents the progression of upper and lower bounds throughout the course of the branch-and-price algorithm. As with range branching, we find that order branching is extremely effective at quickly driving up the lower bound as compared to a classical branching scheme. In addition, we are able to identify high-quality integer solutions earlier on, leading to an improved upper bound development.

We have also experimented with using formulation (6.15) to solve the range minimisation problem. Interestingly, the order-based formulation seems to slightly outperform the vehicle-index formulation (6.5), but is not as good as the last-customer formulation (6.6).

6.7 Conclusion

In this chapter, we proposed a generic branch-and-price approach for range minimisation problems. Our approach relies on range branching, a branching rule that effectively cuts off fractional nodes underestimating the true range of the solution. We have shown several desirable properties of the branching rule, and confirmed its strong computational performance in extensive experiments on instances of the fair capacitated vehicle routing problem and fair generalised assignment problem. Using our new branching rule, we were able to solve many more large instances to optimality, compared to using classical branching strategies. Finally, we successfully generalised our approach to the case of order-based objective functions, such as the Gini deviation.

Depending on the problem at hand, some critical remarks on range branching are in place. First, minimising the range can have unintended negative consequences in the form of 'wasteful' columns being selected in optimal solutions. For example, in our capacitated vehicle routing application we observed that the range was reduced through the use of TSP-suboptimal routes. The degree to which this is problematic and can be avoided is problem-specific. Second, imposing the range branching decisions can make the pricing problem degenerate. This was observed in the capacitated vehicle routing problem, where imposing our branching strategy caused a weakening of label dominance rules. The issue did not occur in the generalised assignment problem, where the efficiency of the pricing algorithm was not affected by range branching decisions. Moreover, the increased computing time per branch-and-price node might still be compensated by a large reduction in the number of nodes. Once again, the net effect of applying range branching is largely problem-specific.

Several promising directions for future research remain. First, it would be interesting to apply our approach to other problems, such as railway crew scheduling or airline crew pairing. Second, Section 6.3.3 contains many suggestions for different ways of implementing range branching, including, for example, layered branching schemes with increasingly tight cutoff values. Third, our approach can be readily applied to problems where the objective consists of minimising the maximum payoff, or maximising the minimum payoff. Fourth, ensuring TSP-optimality of selected routes in the CVRP remains an interesting open problem. Finally, while range branching focuses on improving the lower bound throughout the branch-and-price algorithm, it seems promising to combine this with heuristics providing tight upper bounds.

Part III

Summary and Conclusions

Chapter 7

Summary and Conclusions

The workers in the vineyard from Chapter 1 received equal pay for different amounts of labour, frustrating those who had worked since the dawn of day. To avoid more workplace disputes, this thesis examined optimisation techniques for fair work allocation. The first part of this thesis focused on optimisation problems arising in the newly proposed crew planning process of Netherlands Railways (NS), which has the goal of constructing a more robust, fair, and flexible crew plan. The second part of this thesis studied models and algorithms for generic fairness-oriented assignment problems.

In Chapter 2, we proposed a Benders decomposition heuristic for robust tactical railway crew scheduling. Here, a set of robust templates must be selected based on historical planning information. This problem arises in the proposed planning process of NS, where the operator wishes to inform crew early on about their work schedules while also maintaining the ability to respond to changes in the daily timetables. Computational experiments on real-life instances from NS, featuring up to 948 tasks per day, demonstrated that our method solved three times more instances to optimality compared to a benchmark algorithm from the literature. Historical planning information was successfully leveraged to generate robust templates, strongly reducing the need for excess duties. Moreover, sparse solutions were obtained at negligible extra costs, and our results provided practical insights on the effect of the template length and the number of reserve templates.

In Chapter 3, we studied railway crew planning with fairness over time imposed by

individual Sharing-Sweet-and-Sour rules. This problem is faced in the operational phase of the new planning process, where an equitable distribution of sweet and sour work among individual crew members must be ensured over a planning period of several months. To solve this problem, we developed a rolling horizon approach with a penalty-based feedback mechanism and a column generation heuristic. Applied to real-life instances from NS, including up to 572 unique guards, our method successfully satisfied the individual Sharing-Sweet-and-Sour rules for on average 95.2% of the employees. Our method was able to balance multiple Sharing-Sweet-and-Sour attributes over different crew bases, displaying rapid convergence to high satisfaction levels.

In Chapter 4, we proposed an efficient exact pricing algorithm to be used in column generation approaches for railway crew scheduling problems. The algorithm is applicable to a large class of basic scheduling problems. By using a tailored time-space network whose size is linear in the number of tasks to be scheduled, our method achieves a quadratic time complexity. This significantly improves upon the cubic time complexity of the previously best-known algorithm from literature. Experiments on randomly generated instances confirmed the efficacy of our method, showing a reduction in computing time of up to 95%.

In Chapter 5, we investigated fairness over time in online optimization problems involving the assignment of work to homogeneous workers. This setting models numerous real-life applications, including vehicle routing and crew scheduling variants. Given the problem's complexity, we proposed an intuitive and interpretable assignment policy with desirable theoretical properties. Since this policy allocates the most desirable piece of work to the worker with the historically highest workload, we refer to it as 'best-to-worst' policy. The best-to-worst policy was evaluated in a case study on the capacitated vehicle routing problem. Our empirical analysis revealed that the most cost-efficient solutions usually result in unfair assignments, while significantly fairer solutions could be attained with minor efficiency loss by applying our policy.

In Chapter 6, we explored range minimisation problems featuring exponentially many variables, as frequently arising in fairness-oriented or bi-objective optimization. Although branch and price is successful at solving cost-oriented problems with many variables, the performance of classical branch-and-price algorithms for range minimisation is drastically impaired by weak linear programming relaxations. We proposed range branching, a generic branching rule directly tackling this issue. Computational

experiments on the fair capacited vehicle routing problem and fair generalised assignment problem demonstrated that range branching allowed us to solve a larger number of challenging instances compared to classical methods. In addition, we successfully generalised range branching to order-based objective functions like the Gini deviation.

Implications

Our analyses in Part I confirm the practical feasibility of the proposed crew planning process at NS, confirming its strong potential benefits to crew members. However, some serious hurdles must be taken to reach full-scale implementation. These include the need to incorporate a wider range of Sharing-Sweet-and-Sour attributes, the issue of strong heterogeneity across crew bases, and the necessity to scale our methods to the full Dutch railway network. It is encouraging to see that NS has already taken serious steps in this direction. Company experts are experimenting with data-driven rules of thumb for selecting templates. Such rules are less computationally demanding than the method of Chapter 2, but still capable of leveraging historical information towards the construction of a robust crew plan. Moreover, the algorithm of Chapter 3 is used to conduct extensive simulation experiments, and its core ideas are being implemented in a commercial crew scheduling solver.

The methods proposed in Part II increase the applicability of fairness-oriented optimisation techniques. The best-to-worst policy of Chapter 5 can be readily applied by practitioners, who can specify the maximum efficiency loss they are willing to incur in return for fairer solutions. As shown in our experiments, a minor efficiency loss typically suffices to attain satisfactory fairness levels. Similarly, the flexible branch-and-price framework of Chapter 6 yields an immediate speed-up on a large class of problems, i.e., problems for which efficient branch-and-price implementations already exist for the cost-oriented version. Our approach increases the size of fairness-oriented instances for which the trade-off between fairness and efficiency can be successfully studied.

Future Research

The research of this thesis provides several promising avenues for future work. Given the focus on large-scale optimisation problems in Part I, scalability of our algorithms remains a critical concern. While the methods of Chapters 2 and 3 are heuristic

in nature, their computational performance does not yet enable one to apply these methods to large-scale instances like the full Dutch railway network. This is especially stringent for the Benders decomposition method of Chapter 2, which cannot yet be reliably applied to instances featuring more than one large crew base. Successfully extending the methods beyond the current instances remains challenging. At the same time, the feasibility of exact methods for railway crew scheduling remains an open question. The field of vehicle routing appears to be more developed in this regard, offering several promising ideas that might be successfully adapted to crew scheduling applications. The work of Chapter 4 can also be expected to play a role here.

The chapters in Part II take some initial steps towards large-scale fairness-oriented optimisation, but generally consider relatively simplistic settings. As such, many real-life problem characteristics can still be incorporated to increase the realism of our models. For example, it would be interesting to examine the concept of fairness among groups of workers, or among heterogeneous workers. Moreover, it is not clear what an efficient assignment policy should look like when fairness is measured along multiple payoff attributes, as is the case with the Sharing-Sweet-and-Sour rules at NS. From a theoretical perspective, an informed way of choosing among formulations for range minimisation problems is desired. Given the strong difference in computational performance among the formulations of Chapter 6, this certainly warrants more attention.

References

Abbink, E., Albino, L., Dollevoet, T., Huisman, D., Roussado, J. and Saldanha, R. (2011). 'Solving large scale crew scheduling problems in practice'. *Public Transport* 3.2, pp. 149–164.

Abbink, E., Fischetti, M., Kroon, L., Timmer, G. and Vromans, M. (2005). 'Reinventing crew scheduling at Netherlands Railways'. *Interfaces* 35.5, pp. 393–401.

Abbink, E., Huisman, D. and Kroon, L. (2018). 'Railway Crew Management'. *Handbook of Optimization in the Railway Industry*. Ed. by R. Borndörfer, T. Klug, L. Lamorgese, C. Mannino, M. Reuther and T. Schlechte. Cham: Springer International Publishing, pp. 243–264.

Albers, M. and Nandramn, A. (Sept. 2022). 'Vrijdag in heel Nederland geen NS-treinen wegens staking, bonden onvermurwbaar'. *De Volkskrant*.

Amaya, J. and Uribe, P. (2018). 'A model and computational tool for crew scheduling in train transportation of mine materials by using a local search strategy'. *Top* 26, pp. 383–402.

Antunes, D., Vaze, V. and Antunes, A.P. (2019). 'A robust pairing model for airline crew scheduling'. *Transportation Science* 53.6, pp. 1751–1771.

Applegate, D., Cook, W., Dash, S. and Rohe, A. (2002). 'Solution of a min-max vehicle routing problem'. *INFORMS Journal on Computing* 14.2, pp. 132–143.

Audibert, J., Bubeck, S. and Lugosi, G. (2014). 'Regret in online combinatorial optimization'. *Mathematics of Operations Research* 39.1, pp. 31–45.

Balakrishnan, A., Mirchandani, P. and Lin, S. (2022). 'Crew Assignment with Duty Time Limits for Transport Services: Tight Multicommodity Models'. *Operations Research* 70.2, pp. 690–714.

Baldacci, R., Mingozzi, A. and Roberti, R. (2011). 'New route relaxation and pricing strategies for the vehicle routing problem'. *Operations Research* 59.5, pp. 1269–1283.

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

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M. and Vance, P. (1998). 'Branch-and-price: Column generation for solving huge integer programs'. *Operations Research* 46.3, pp. 316–329.

Bertsimas, D., Farias, V.F. and Trichakis, N. (2011). 'The price of fairness'. *Operations Research* 59.1, pp. 17–31.

Bešinović, N. (2020). 'Resilience in railway transport systems: A literature review and research agenda'. *Transport Reviews* 40.4, pp. 457–478.

Bianchessi, N., Tilk, C. and Irnich, S. (2023). 'On solving the minmax multiple traveling salesman problem by column generation'. Presented at Column Generation 2023, Montréal.

Borndörfer, R., Reuther, M., Schlechte, T., Schulz, C., Swarat, E. and Weider, S. (2015). *Duty Rostering in Public Transport - Facing Preferences, Fairness, and Fatigue*. Tech. rep. ZIB.

Borndörfer, R., Schulz, C., Seidl, S. and Weider, S. (2017). 'Integration of duty scheduling and rostering to increase driver satisfaction'. *Public Transport* 9.1, pp. 177–191.

Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M. and Schlechte, T. (2018). *Handbook of optimization in the railway industry*. Vol. 268. Springer.

Breugem, T., Schlechte, T., Schulz, C. and Borndörfer, R. (2023). 'A three-phase heuristic for the Fairness-Oriented Crew Rostering Problem'. *Computers & Operations Research* 154, p. 106186.

Breugem, T., van Rossum, B.T.C., Dollevoet, T. and Huisman, D. (2022a). 'A column generation approach for the integrated crew re-planning problem'. *Omega* 107, p. 102555.

Breugem, T. (2020). *Crew Planning at Netherlands Railways: Improving Fairness, Attractiveness, and Efficiency*. EPS-2020-494-LIS.

Breugem, T., Dollevoet, T. and Huisman, D. (2022b). 'Is equality always desirable? Analyzing the trade-off between fairness and attractiveness in crew rostering'. *Management Science* 68.4, pp. 2619–2641.

Caprara, A., Fischetti, M., Guida, P., Toth, P. and Vigo, D. (1999). 'Solution of large-scale railway crew planning problems: The Italian experience'. *Computer-Aided Transit Scheduling*. Springer, pp. 1–18.

Caprara, A., Toth, P., Vigo, D. and Fischetti, M. (1998). 'Modeling and solving the crew rostering problem'. *Operations Research* 46.6, pp. 820–830.

Chen, V. and Hooker, J. (2023). 'A guide to formulating fairness in an optimization model'. *Annals of Operations Research* 326.1, pp. 581–619.

Chu, P.C. and Beasley, J.E. (1997). 'A genetic algorithm for the generalised assignment problem'. *Computers & Operations Research* 24.1, pp. 17–23.

Cordeau, J.F., Stojković, G., Soumis, F. and Desrosiers, J. (2001). 'Benders decomposition for simultaneous aircraft routing and crew scheduling'. *Transportation Science* 35.4, pp. 375–388.

Cormen, T., Leiserson, C., Rivest, R. and Stein, C. (1990). *Introduction to Algorithms*. MIT Press.

Costa, L., Contardo, C. and Desaulniers, G. (2019). 'Exact branch-price-and-cut algorithms for vehicle routing'. *Transportation Science* 53.4, pp. 946–985.

Darvish, M., Coelho, L.C. and Jans, R. (2020). *Comparison of symmetry breaking and input ordering techniques for routing problems*. Tech. rep. CIRRELT.

Desaulniers, G., Desrosiers, J. and Solomon, M. (2002). 'Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems'. *Essays and surveys in metaheuristics*. Springer, pp. 309–324.

Desaulniers, G., Desrosiers, J. and Solomon, M.M., eds. (2005). *Column generation*. New York, NY: Springer Science & Business Media.

Dunbar, M., Froyland, G. and Wu, C.-L. (2014). 'An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing'. *Computers & Operations Research* 45, pp. 68–86.

Ernst, A., Jiang, H., Krishnamoorthy, M., Nott, H. and Sier, D. (2001). 'An integrated optimization model for train crew management'. *Annals of Operations Research* 108.1, pp. 211–224.

Ernst, A., Jiang, H., Krishnamoorthy, M. and Sier, D. (2004). 'Staff scheduling and rostering: A review of applications, methods and models'. *European Journal of Operational Research* 153.1, pp. 3–27.

França, P.M., Gendreau, M., Laporte, G. and Müller, F.M. (1995). 'The m-traveling salesman problem with minmax objective'. *Transportation Science* 29.3, pp. 267–275.

Fuentes, M., Cadarso, L. and Marín, Á. (2019). 'A hybrid model for crew scheduling in rail rapid transit networks'. *Transportation Research Part B: Methodological* 125, pp. 248–265.

Gattermann-Itschert, T., Poreschack, L. and Thonemann, U. (2023). 'Using Machine Learning to Include Planners' Preferences in Railway Crew Scheduling Optimization'. *Transportation Science* 57.3, pp. 796–812.

Grötschel, M., Krumke, S., Rambau, J. and Zimmermann, T.W.U. (2001). *Combinatorial online optimization in real time*. Springer.

Hanafi, R. and Kozan, E. (2014). 'A hybrid constructive heuristic and simulated annealing for railway crew scheduling'. *Computers & Industrial Engineering* 70, pp. 11–19.

Hartog, A., Huisman, D., Abbink, E. and Kroon, L. (2009). 'Decision support for crew rostering at NS'. *Public Transport* 1.2, pp. 121–133.

Heil, J., Hoffmann, K. and Buscher, U. (2020). 'Railway crew scheduling: Models, methods and applications'. *European Journal of Operational Research* 283.2, pp. 405–425.

Huisman, D. (2007). 'A column generation approach for the rail crew re-scheduling problem'. *European Journal of Operational Research* 180.1, pp. 163–173.

Joncour, C., Michel, S., Ruslan, S., Sverdlov, D. and Vanderbeck, F. (2010). 'Column generation based primal heuristics'. *Electronic Notes in Discrete Mathematics* 36, pp. 695–702.

Jozefowiez, N., Semet, F. and Talbi, E.-G. (2009). 'An evolutionary algorithm for the vehicle routing problem with route balancing'. *European Journal of Operational Research* 195.3, pp. 761–769.

Jütte, S., Müller, D. and Thonemann, U. (2017). 'Optimizing railway crew schedules with fairness preferences'. *Journal of Scheduling* 20.1, pp. 43–55.

Jütte, S. and Thonemann, U. (2012). 'Divide-and-price: A decomposition algorithm for solving large railway crew scheduling problems'. *European Journal of Operational Research* 219.2, pp. 214–223.

Karsu, Ö. and Morton, A. (2015). 'Inequity averse optimization in operational research'. *European Journal of Operational Research* 245.2, pp. 343–359.

Kohl, N. and Karisch, S. (2004). 'Airline crew rostering: Problem types, modeling, and optimization'. *Annals of Operations Research* 127.1, pp. 223–257.

Liebchen, C., Lübbecke, M., Möhring, R. and Stiller, S. (2009). 'The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications'. *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*. Ed. by R.K. Ahuja, R.H. Möhring and C.D. Zaroliagis. Berlin, Heidelberg: Springer Berlin Heidelberg. Chap. 1, pp. 1–27.

Ljubić, I., P, M.A., Puerto, J. and Torrejón, A. (2024). 'Benders decomposition for the discrete ordered median problem'. *European Journal of Operational Research*.

Lodi, A., Olivier, P., Pesant, G. and Sankaranarayanan, S. (2024a). 'Fairness over time in dynamic resource allocation with an application in healthcare'. *Mathematical Programming* 203.1, pp. 285–318.

Lodi, A., Sankaranarayanan, S. and Wang, G. (2024b). 'A framework for fair decision-making over time with time-invariant utilities'. *European Journal of Operational Research* 319.2, pp. 456–467.

Lübbecke, M. (2010). 'Column generation'. *Wiley Encyclopedia of Operations Research and Management Science. Wiley, New York*, pp. 1–14.

Lusby, R.M., Larsen, J. and Bull, S. (2018). 'A survey on robustness in railway planning'. *European Journal of Operational Research* 266.1, pp. 1–15.

Lysgaard, J., López-Sánchez, A.D. and Hernández-Díaz, A.G. (2020). 'A matheuristic for the minmax capacitated open vehicle routing problem'. *International Transactions in Operational Research* 27.1, pp. 394–417.

Maenhout, B. and Vanhoucke, M. (2010). 'A hybrid scatter search heuristic for personalized crew rostering in the airline industry'. *European Journal of Operational Research* 206.1, pp. 155–167.

Magnanti, T.L. and Wong, R.T. (1981). 'Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria'. *Operations Research* 29.3, pp. 464–484.

Marín, A., Ponce, D. and Puerto, J. (2020). 'A fresh view on the discrete ordered median problem based on partial monotonicity'. *European Journal of Operational Research* 286.3, pp. 839–848.

Marinho Damião, C. (2021). 'The capacitated vehicle routing problem with alternative objective functions'. MA thesis. Universidade Federal Fluminense.

Matl, P., Hartl, R. and Vidal, T. (2018). 'Workload equity in vehicle routing problems: A survey and analysis'. *Transportation Science* 52.2, pp. 239–260.

Matl, P., Hartl, R. and Vidal, T. (2019). 'Workload equity in vehicle routing: The impact of alternative workload resources'. *Computers & Operations Research* 110, pp. 116–129.

Mercier, A., Cordeau, J.F. and Soumis, F. (2005). 'A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem'. *Computers & Operations Research* 32.6, pp. 1451–1476.

Mesquita, M., Moz, M., Paias, A. and Pato, M. (2013). 'A decomposition approach for the integrated vehicle-crew-roster problem with days-off pattern'. *European Journal of Operational Research* 229.2, pp. 318–331.

Muter, I., Birbil, Ş.İ., Bülbül, K., Şahin, G., Yenigün, H., Taş, D. and Tüzün, D. (2013). 'Solving a robust airline crew pairing problem with column generation'. *Computers & Operations Research* 40.3, pp. 815–830.

Nederlandse Spoorwegen (2022). *NS Personeelstekort & Dienstregeling november 2022*. Accessed November 22, 2024. `https://open.overheid.nl/documenten/ronl-48c1ba92b7eca801b3b10619f6038aeae581363e/pdf`.

Nederlandse Spoorwegen (2023). *NS Jaarverslag 2023*. Accessed November 22, 2024. `https://www.nsjaarverslag.nl/external/asset/download/project/f9822a0a-03e8-0000-607c-d0e51dab3a1d/name/NS-Jaarverslag-2023.pdf`.

Nishi, T., Sugiyama, T. and Inuiguchi, M. (2014). 'Two-level decomposition algorithm for crew rostering problems with fair working condition'. *European Journal of Operational Research* 237.2, pp. 465–473.

Papadakos, N. (2008). 'Practical enhancements to the Magnanti-Wong method'. *Operations Research Letters* 36.4, pp. 444–449.

Potthoff, D., Huisman, D. and Desaulniers, G. (2010). 'Column generation with dynamic duty selection for railway crew rescheduling'. *Transportation Science* 44.4, pp. 493–505.

Puerto, J., Ricca, F. and Scozzari, A. (2012). 'Range minimization problems in path-facility location on trees'. *Discrete Applied Mathematics* 160.15, pp. 2294–2305.

Quesnel, F., Desaulniers, G. and Soumis, F. (2020). 'Improving air crew rostering by considering crew preferences in the crew pairing problem'. *Transportation Science* 54.1, pp. 97–114.

Rählmann, C., Wagener, F. and Thonemann, U. (2021). 'Robust Tactical Crew Scheduling Under Uncertain Demand'. *Transportation Science* 55.6, pp. 1392–1410.

Rahmaniani, R., Crainic, T.G., Gendreau, M. and Rei, W. (2017). 'The Benders decomposition algorithm: A literature review'. *European Journal of Operational Research* 259.3, pp. 801–817.

Rawls, J. (1991). 'Justice as fairness: Political not metaphysical'. *Equality and Liberty: Analyzing Rawls and Nozick*. Springer, pp. 145–173.

Er-Rbib, S., Desaulniers, G., Elhallaoui, I. and Munroe, P. (2021). 'Preference-based and cyclic bus driver rostering problem with fixed days off'. *Public Transport* 13.2, pp. 251–286.

Saddoune, M., Desaulniers, G., Elhallaoui, I. and Soumis, F. (2012). 'Integrated airline crew pairing and crew assignment by dynamic constraint aggregation'. *Transportation Science* 46.1, pp. 39–55.

Saddoune, M., Desaulniers, G. and Soumis, F. (2009). 'A rolling horizon solution approach for the airline crew pairing problem'. *2009 International Conference on Computers & Industrial Engineering*. IEEE, pp. 344–347.

Savelsbergh, M. (1997). 'A branch-and-price algorithm for the generalized assignment problem'. *Operations Research* 45.6, pp. 831–841.

Shebalov, S. and Klabjan, D. (2006). 'Robust airline crew pairing: Move-up crews'. *Transportation Science* 40.3, pp. 300–312.

Si Salem, T., Iosifidis, G. and Neglia, G. (2022). 'Enabling long-term fairness in dynamic resource allocation'. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6.3, pp. 1–36.

Stojković, M., Soumis, F. and Desrosiers, J. (1998). 'The operational airline crew scheduling problem'. *Transportation Science* 32.3, pp. 232–245.

Topham, G. (June 2022). 'Great Britain faces biggest rail strike in 30 years'. *The Guardian*.

Tsang, M.Y. and Shehadeh, K.S. (2024). *A Unified Framework for Analyzing and Optimizing a Class of Convex Fairness Measures*.

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T. and Subramanian, A. (2017). 'New benchmark instances for the capacitated vehicle routing problem'. *European Journal of Operational Research* 257.3, pp. 845–858.

Van Rossum, B.T.C. (2022a). 'A fast exact pricing algorithm for the railway crew scheduling problem'. *Operations Research Letters* 50.6, pp. 707–711.

Van Rossum, B.T.C. (2022b). 'Code: A Fast Exact Pricing Algorithm for the Railway Crew Scheduling Problem'.

Van Rossum, B.T.C., Chen, R. and Lodi, A. (2023). 'Optimizing Fairness over Time with Homogeneous Workers'. *23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

Van Rossum, B.T.C., Chen, R. and Lodi, A. (2024a). 'A New Branching Rule for Range Minimization Problems'. *Integer Programming and Combinatorial Optimization*. Ed. by J. Vygen and J. Byrka. Cham: Springer Nature Switzerland, pp. 433–445.

Van Rossum, B.T.C., Dollevoet, T. and Huisman, D. (2024b). 'Railway crew planning with fairness over time'. *European Journal of Operational Research* 318.1, pp. 55–70.

Van den Hurk, V.E., van den Broek, E.L., van den Akker, M. and Abbink, E. (2024). 'Template and constraint-based models for the optimization of schedules, applied to Netherlands Railways (NS)' crew planning'. *Proceedings of the 36th BNAIC/BeNeLearn 2024 Joint International Scientific Conferences on AI and Machine Learning*.

Vanheusden, S., Van Gils, T., Caris, A., Ramaekers, K. and Braekers, K. (2020). 'Operational workload balancing in manual order picking'. *Computers & Industrial Engineering* 141, p. 106269.

Veelenturf, L.P., Potthoff, D., Huisman, D., Kroon, L.G., Maróti, G. and Wagelmans, A.P. (2016). 'A quasi-robust optimization approach for crew rescheduling'. *Transportation Science* 50.1, pp. 204–215.

Wang, J., Gronalt, M. and Sun, Y. (2017). 'A two-stage approach to the depot shunting driver assignment problem with workload balance considerations'. *PloS one* 12.7, e0181165.

Weide, O., Ryan, D. and Ehrgott, M. (2010). 'An iterative approach to robust and integrated aircraft routing and crew scheduling'. *Computers & Operations Research* 37.5, pp. 833–844.

Wolbeck, L. (2019). *Fairness aspects in personnel scheduling*. Diskussionsbeiträge.

Zeighami, V., Saddoune, M. and Soumis, F. (2020). 'Alternating Lagrangian decomposition for integrated airline crew scheduling problem'. *European Journal of Operational Research* 287.1, pp. 211–224.

Zeighami, V. and Soumis, F. (2019). 'Combining Benders decomposition and column generation for integrated crew pairing and personalized crew assignment problems'. *Transportation Science* 53.5, pp. 1479–1499.

Samenvatting

In de bekende parabel over arbeiders in de wijngaard ontvangen de werkers gelijk loon voor een ongelijk aantal gewerkte uren, tot grote onvrede van hen die al vroeg in de ochtend waren begonnen met werken. Om dergelijke oneerlijke situaties te voorkomen bestudeert dit proefschrift verschillende optimalisatietechnieken om werk eerlijk te verdelen. Deel I van dit proefschrift onderzoekt optimalisatieproblemen in het beoogde nieuwe personeelsplanningsproces van NS, met als doel het ontwikkelen van een robuuster, eerlijker, en flexibeler personeelsplan. Deel II van dit proefschrift beschouwt modellen en algoritmes voor generieke toewijzingsproblemen met eerlijkheidsoverwegingen.

In hoofdstuk 2 bestuderen we een robuust optimalisatieprobleem in de tactische fase van het beoogde personeelsplanningsproces van NS, enkele maanden voor de dag van uitvoering. Hierin moet een robuuste verzameling *blokken*, sjablonen met tijdsintervallen voor diensten, worden gekozen op basis van historische planningsinformatie. Dit stelt het bedrijf in staat om enerzijds personeel tijdig te informeren over hun werktijden, en anderzijds te reageren op operationele wijzigingen in de dagelijkse dienstregeling. We lossen dit probleem op met een Benders-decompositieheuristiek, die we evalueren op praktijkinstanties van NS met tot wel 948 taken per dag. Onze methode is in staat driemaal zoveel instanties optimaal op te lossen als een benchmarkalgoritme uit de literatuur. Door historische planningsinformatie te benutten kunnen robuustere blokken worden gekozen en zijn er minder reservediensten nodig. Bovendien is het mogelijk om eenvoudige oplossingen te vinden die slechts een klein aantal unieke blokken bevatten, tegen een kleine toename in personeelskosten. Tot slot bieden onze resultaten nuttige inzichten in het effect van de bloklengte en het aantal blokken.

In hoofdstuk 3 bekijken we vervolgens het genereren van gepersonaliseerde diensten

voor spoorwegpersoneel in de operationele fase, enkele weken voor de dag van uitvoering. In dit probleem moet worden voldaan aan zogenaamde individuele lusten-en-lastenregels. Deze regels hebben als doel dat het werk gedurende de planningsperiode eerlijk verdeeld wordt over alle werknemers. Zo dient elk personeelslid minstens 35% van zijn tijd door te brengen op een moderne type A-Trein, en maximaal 30% van zijn tijd op een trein met een hoog risico op passagiersagressie. Om dit probleem op te lossen gebruiken we een kolomgeneratieheuristiek met een feedbackmechanisme. We testen deze methode op praktijkinstanties van NS met tot wel 572 verschillende conducteurs. Onze methode is in staat om diensten te genereren waarmee voor gemiddeld 95.2% van de werknemers aan alle regels wordt voldaan. Daarnaast is onze methode in staat om aantrekkelijk en minder aantrekkelijk werk te verdelen over de diverse standplaatsen, en stabiliseren de individuele lusten-en-lastenscores binnen enkele weken op hun doelwaarden.

In hoofdstuk 4 ontwikkelen we een efficiënt algoritme om het zogenaamde *pricing problem* binnen kolomgeneratiealgoritmes voor spoorwegplanningsproblemen op te lossen. Het algoritme maakt gebruik van een specifiek tijd-ruimte netwerk, waarvan de grootte lineair schaalt in het aantal taken dat gepland moet worden. Hierdoor bereikt het algoritme een kwadratische tijdscomplexiteit. Dit betekent een aanzienlijke verbetering ten opzichte van de kubische complexiteit van de snelste algoritmes uit de literatuur. Op synthetische instanties reduceert onze methode de rekentijd met tot wel 95%.

In hoofdstuk 5 bestuderen we eerlijkheid door de tijd heen in *online* optimalisatieproblemen waarin werk aan een groep identieke arbeiders toegewezen moet worden. Een grote groep praktische problemen, waaronder voertuigroutering en personeelsplanning voor spoorwegen en de luchtvaart, kan op deze manier gemodelleerd worden. Intuïtief zou men zeggen dat deze problemen het best opgelost kunnen worden door het leukste werk toe te wijzen aan de werknemers die tot op heden het minst leuke werk hebben verricht. We bewijzen dat deze beslissingsregel, die we 'best-naar-slechtst-regel' noemen, inderdaad over gunstige theoretische eigenschappen beschikt. Daarnaast blijkt uit onze testen met voertuigrouteringsproblemen dat deze regel inderdaad tot eerlijke oplossingen leidt. De eerlijkheid van de toewijzingen kan nog verder worden vergroot in ruil voor een beperkte kostenstijging.

In hoofdstuk 6 kijken we tot slot naar het minimaliseren van het bereik in problemen met exponentieel veel variabelen. Het bereik is een veelgebruikte eerlijkheidsmaatstad, en wordt gedefinieerd als het verschil tussen de hoogste en laagste waarde.

Een voorbeeld hiervan is het voertuigrouteringsprobleem, waarin enorm veel mogelijke routes bestaan en men, vanuit eerlijkheidsoverwegingen, het verschil in afstand tussen de kortste en langste route zou willen minimaliseren. Hoewel geavanceerde *branch-and-price*-algoritmes succesvol worden toegepast om de kosten te minimaliseren in dergelijke problemen, presteren deze algoritmes ondermaats wanneer ze gebruikt worden om het bereik te minimaliseren. Dit is te wijten aan de zwakke wiskundige formuleringen voor dit type probleem. We introduceren *range branching*, een generieke *branching rule* gericht op het aanpakken van deze zwakke formuleringen. Door range branching te gebruiken kunnen we een groter aantal instanties van voertuigrouteringsproblemen en algemene toewijzingsproblemen met eerlijkheidsoverwegingen oplossen dan klassieke methodes. Daarnaast generaliseren we range branching naar algemenere doelfuncties, zoals de Gini-afwijking.

Implicaties

De analyses in deel I bevestigen de praktische haalbaarheid van het beoogde personeelsplanningsproces van NS, en laten zien dat dit grote voordelen zou bieden voor het personeel. Zo zouden de geplande roosters in het nieuwe planningsproces een stuk betrouwbaarder worden, en zouden de individuele lusten-en-lastenregels een aantrekkelijk werkpakket voor elke werknemer garanderen. Er dienen echter nog wat hordes genomen te worden voordat een grootschalige implementatie in de praktijk mogelijk is. Hieronder vallen de eis om een groter aantal lusten-en-lastenregels te modelleren, het probleem van grote geografische verschillen tussen standplaatsen, en de uitdaging om onze methodes op te schalen naar het volledige Nederlandse spoornetwerk. Het is echter bemoedigend om te zien dat NS al enkele serieuze stappen in deze richting heeft gezet. Zo experimenteren analisten van NS momenteel met vuistregels om blokken te kiezen, waardoor minder zware rekenmiddelen nodig zijn dan onze methode uit hoofdstuk 2. Daarnaast wordt het algoritme uit hoofdstuk 3 veelvuldig gebruikt om uitgebreide simulatie-experimenten mee uit te voeren. De belangrijkste ideeën uit dit algoritme worden bovendien geïmplementeerd in commerciële personeelsplanningssoftware, wat een grootschalige toepassing mogelijk zou moeten maken.

De methoden uit deel II zorgen ervoor dat optimalisatietechnieken met eerlijkheidsoverwegingen breder toepasbaar worden. De best-naar-slechtst-regel uit hoofdstuk 5 kan direct worden toegepast door planners, die zelf kunnen specificeren hoeveel extra kosten ze willen maken in ruil voor eerlijkere oplossingen. Vaak volstaat een klein efficiëntieverlies om acceptabele eerlijkheidsniveaus te behalen. Het flexibele frame-

work uit hoofdstuk 6 zorgt ervoor dat een flink aantal problemen sneller opgelost kan worden. De uitruil tussen eerlijkheid en efficiëntie kan daarmee ook voor grotere probleeminstanties in kaart gebracht worden.

About the author

Bart van Rossum was born in Nijmegen, the Netherlands, in 1997. Bart holds Bachelor degrees, summa cum laude, in Econometrics and Operations Research and Economics and Business Economics, as well as a Master's degree, cum laude, in Econometrics and Management Science with a specialisation in Operations Research and Quantitative Logistics. All degrees were obtained at Erasmus University Rotterdam. Bart's research interests include railway crew planning, fairness-oriented optimisation, and combinatorial optimisation in general. In his research, Bart employs a wide range of mathematical programming techniques, with a strong emphasis on large-scale column generation.

In 2020, Bart started his PhD candidacy at the Econometric Institute, Erasmus University Rotterdam, as part of the PhD program of the Erasmus Research Institute of Management. Bart's work has been published in scientific journals such as *European Journal of Operational Research*, *Omega*, and *Operations Research Letters*. In 2022, Bart was awarded the INFORMS Railway Applications Section Best Student Paper Award for his work on individual fairness over time in railway crew scheduling. Together with Rick Willemsen, he received the first prize in the junior category at the ROADEF 2022 challenge. During his PhD, Bart collaborated closely with the Digitalization of Operations department at Netherlands Railways. In Spring 2023, he engaged in a research visit to prof.dr. Andrea Lodi at Cornell Tech, New York City. Bart currently works as a postdoctoral researcher at the Eindhoven University of Technology within the Operations, Planning, Accounting and Control Group.

Portfolio

Publications

Breugem, T., van Rossum, B.T.C., Dollevoet, T. and Huisman, D. (2022a). 'A column generation approach for the integrated crew re-planning problem'. *Omega* 107, p. 102555.

Van Rossum, B.T.C. (2022a). 'A fast exact pricing algorithm for the railway crew scheduling problem'. *Operations Research Letters* 50.6, pp. 707–711.

Van Rossum, B.T.C., Chen, R. and Lodi, A. (2023). 'Optimizing Fairness over Time with Homogeneous Workers'. *23rd Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

Van Rossum, B.T.C., Chen, R. and Lodi, A. (2024a). 'A New Branching Rule for Range Minimization Problems'. *Integer Programming and Combinatorial Optimization*. Ed. by J. Vygen and J. Byrka. Cham: Springer Nature Switzerland, pp. 433–445.

Van Rossum, B.T.C., Dollevoet, T. and Huisman, D. (2024b). 'Railway crew planning with fairness over time'. *European Journal of Operational Research* 318.1, pp. 55–70.

Van Rossum, B. and Frasincar, F. (2019). 'Augmenting LOD-Based Recommender Systems Using Graph Centrality Measures'. *Web Engineering*. Ed. by M. Bakaev, F. Frasincar and I.-Y. Ko. Cham: Springer International Publishing, pp. 19–31.

Research visits

Research visit to Cornell Tech to collaborate with prof.dr. Andrea Lodi and dr. Rui Chen *(February-June 2023)*.

Awards

Best Junior Team ROADEF Challenge 2022 *(with Rick Willemsen)*
First Prize Railway Applications Section Student Paper Award, INFORMS 2022

PhD courses

Interior Point Methods
Robust Optimisation
Integer Programming Methods
Randomized Algorithms
Convex Analysis for Optimization
Networks and Polyhedra
Networks and Semidefinite Programming
Hacking the Code of Publishing
Presentation Skills
Academic Writing
Open Science & Scientific Integrity
Topics in the Philosophy of Science
Thesis Supervision

Teaching activities

Lecturer for the *Econometrics and Operations Research* bachelor course *Advanced Programming (Minor)*, 2023-2024.
Supervisor and co-reader for master theses in *Operations Research and Quantitative Logistics*, 2021-2024.
Supervisor and co-reader for bachelor theses in *Econometrics and Operations Research*, 2020-2024.
Teaching assistant for the *Operations Research and Quantitative Logistics* master course *Transportation and Scheduling*, 2021-2024.
Teaching assistant for the *Econometrics and Operations Research* bachelor courses *Academic Skills* and *Analysis*, 2021-2023.
Teaching assistant for the *Economics and Business Economics* bachelor courses *Mathematics and Game Theory* and *Introduction to Mathematics*, 2021-2023.

Referee activities
Computers & Operations Research
European Journal of Operational Research
INFORMS Journal on Computing
Journal of Rail Transport Planning & Management
Transportation Research: Part B
Transportation Research: Part C
Transportation Science

Conference presentations
INFORMS 2024, Seattle, USA
IPCO 2024, Wroclaw, Poland
EURO 2024, Copenhagen, Denmark
ATMOS 2023, Amsterdam, The Netherlands
OR 2023, Hamburg, Germany
Column Generation 2023, Montréal, Canada
LNMB 2023, Soesterberg, The Netherlands
TRISTAN XI, Island of Mauritius
INFORMS 2022, Indianapolis, USA
EIPC 2022, Rotterdam, The Netherlands
EURO 2021, Athens, Greece
ODS 2021, Rome, Italy

Other presentations
Business Analytics Seminar, University of Amsterdam, September 2024, Amsterdam, The Netherlands
EI-ERIM Operations Research Seminar, Econometric Institute, 2021-2024, Rotterdam, The Netherlands

The ERIM PhD Series

The ERIM PhD Series contains PhD dissertations in the field of Research in Management defended at Erasmus University Rotterdam and supervised by senior researchers affiliated to the Erasmus Research Institute of Management (ERIM). Dissertations in the ERIM PhD Series are available in full text through: https://pure.eur.nl. ERIM is the joint research institute of the Rotterdam School of Management (RSM) and the Erasmus School of Economics (ESE) at the Erasmus University Rotterdam (EUR).

Dissertations in the last four years

Abdelwahed, A., *Optimizing Sustainable Transit Bus Networks in Smart Cities*, Supervisors: Prof. W. Ketter, Dr P. van den Berg & Dr T. Brandt, EPS-2022-549-LIS

Alkema, J., *READY, SET, GO(AL)! New Directions in Goal-Setting Research*, Supervisors: Prof. H.G.H. van Dierendonck & Prof. S.R. Giessner, ESP-2022-555-ORG

Alves, R.A.T.R.M, *Information Transmission in Finance: Essays on Commodity Markets, Sustainable Investing, and Social Networks*, Supervisors: Prof. M.A. van Dijk & Dr M. Szymanowska, EPS-2021-532-LIS

Anantavrasilp, S., *Essays on Ownership Structures, Corporate Finance Policies and Financial Reporting Decisions*, Supervisors: Prof. A. de Jong & Prof. P.G.J. Roosenboom, EPS-2021-516-F&E

Ansarin, M., *The Economic Consequences of Electricity Pricing in the Renewable Energy Era*, Supervisors: Prof. W. Ketter & Dr Y. Ghiassi-Farrokhfal, EPS-2021-528-LIS

Aydin Gökgöz, Z. *Mobile Consumers and Applications: Essays on Mobile Marketing*, Supervisors: Prof. G.H. van Bruggen & Dr B. Ataman, EPS-2021-519-MKT

Azadeh, K., *Robotized Warehouses: Design and Performance Analysis*, Supervisors: Prof. M.B.M. de Koster & Prof. D. Roy, EPS-2021-515-LIS

Badenhausen, K., *IoT – Inducing Organizational Transformation?* Supervisors: Prof. R.A. Zuidwijk & Dr M. Stevens, ESP-2022-559-LIS

Balocco, F.A.M., *Asymmetric information in programmatic advertising: Three studies on adverse selection, mechanism choices, and fee structures*, Supervisors: Prof. T. LI & Prof. E. van heck, EPS-2023-565-LIS

Bilgin, B., *Visionary Leadership and the Pursuit of Organizational Visions*, Supervisors: Prof. D.L. van Knippenberg & Dr I.J. Hoever, EPS-2024-583-ORG

Breet, S., *A Network Perspective on Corporate Entrepreneurship: How workplace relationships influence entrepreneurial behavior*, Supervisors: Prof. J.J.P. Jansen, Prof. J. Dul & Dr L. Glaser, EPS-2022-545-S&E

Bunders, D., *Gigs of their Own: can platform cooperatives become resilient*? Supervisors: Prof. T. de Moor, Prof. A. Akkerman & Prof. P. Dykstra, EPS-2024-580-ORG

Chung, Y.S., *Valorizing Innovation through Imaginativeness in Business Venturing*, Supervisors: Prof. PH.B.F. Franses & Prof. H.P.G. Pennings, EPS-2022-561-MKT

Dijkstra, N., *Statistical Modeling Approaches for Behavioral and Medical Sciences*, Supervisors: Prof. P.J.F Groenen, Prof. H. Tiemeier & Prof. A.R Thurik, EPS-2022-539-S&E

Duijsters, J., *Change in Inter-Organizational Relationship Portfolios and Social Networks in the Context of Corporate Venturing*, Prof. V.J.A. van de Vrande, Prof. J.J.P. Jansen & Prof. P.P.M.A.R. Heugens, EPS-2024-504-S&E

Fu, G., *Agency Problems in the Mutual Fund Industry*, Supervisors: Prof. M.J.C.M. Verbeek & Dr E. Genc, EPS-2022-526-F&A

Genc, A., *The Triad of Innovation: Exploring management innovation, business model innovation, and ownership in unlocking firms' innovative potential*, Supervisors: Prof. H. W. Volberda & Prof. J. S. Sidhu, EPS-2024-579-S&E

Geradts, T. *Moving Beyond the Why and What Question: How Corporations Achieve Sustainable Development*, Supervisors: Prof. J.J.P. Jansen, Prof. J.P. Cornelissen & Prof. A. Nijhof, EPS-2021-521-S&E

Giessen, M. van der, *Co-creating Safety and Security: Essays on bridging disparate needs and requirements to foster safety and security*, Supervisors: Prof. G. Jacobs, Prof. J.P. Cornelissen & Prof. P.S. Bayerl, EPS-2022-542-ORG

Giudici, A., *Cooperation, Reliability, and Matching in Inland Freight Transport*, Supervisors: Prof. R.A. Zuidwijk, Prof. C. Thielen & Dr T. Lu, EPS-2022-533-LIS

Gunadi, M.P., *Essays on Consumers and Numbers*, Supervisors: Prof. S. Puntoni & Dr B. Van Den Bergh, EPS-2022-558-MKT

Harter, C., *Vulnerability through vertical collaboration in transportation: A complex networks approach*, Supervisors: Prof. R.A. Zuidwijk & Dr O.R. Koppius, EPS-2023-560-LIS

Hartleb, J., *Public Transport and Passengers: Optimization Models that Consider Travel Demand,* Supervisors: Prof. D. Huisman, Prof. M. Friedrich & Dr M.E. Schmidt, EPS-2021-535-LIS

Hoogendoorn, Y.N., *Vehicle Routing with Varying Levels of Demand Information*, Supervisors: Prof. A.P.M. Wagelmans, Prof. R. Dekker & Dr R. Spliet, EPS-2024-578-LIS

Hoogervorst, R., *Improving the Scheduling and Rescheduling of Rolling Stock: Solution Methods and Extensions*, Supervisors: Prof. D. Huisman & Dr T.A.B. Dollevoet, EPS-2021-534-LIS

Hulsen, M., *Wait for others? Social and intertemporal preferences in allocation of healthcare resources*, Supervisors: Prof. K.I.M. Rohde & Prof. N.J.A. van Exel, EPS-2023-556-MKT

Jellema, S.F., *Brace for Impact: Good intentions, unintended consequences, and the role of performative micro-processes in organized efforts for societal change*, Supervisors: Prof. J.P. Cornelissen & Prof. T.H. Reus, EPS-2024-587-ORG

Kalvapelle, S.G., *Breaking the Conduit: A Relational Approach to Communication in Management and Entrepreneurship*, Supervisors: Prof. J.P. Cornelissen & Prof. P.P.M.A.R. Heugens, EPS-2023-575-ORG

Kanellopoulos, I., *Navigating Digital Platform Challenges: Three Studies on Blockchain-based Platform Competition, Platform Subsidies for Non-Fungible Token Creation, and Ghosting in the Dating Market*, Supervisors: Prof. T. Li & Prof. H.W.G.M. van Heck, EPS-2024-585-LIS

Karaca, U., *Privacy in Resource Allocation Problems*, Supervisors: Prof. Ş.İ. Birbil & Prof. A.P.M. Wagelmans, EPS-2023-567-LIS

Koritarov, V.D., *The Integration of Crisis Communication and Regulatory Focus: Deconstructing and Optimizing the Corporate Message*, Promotors: Prof. C.B.M. van Riel, Dr G.A.J.M. Berens & Prof. P. Desmet, EPS-2021-522-ORG

Korman, B., *Leader-Subordinate Relations: The Good, the Bad and the Paradoxical*, Promotors: S.R. Giessner & Prof. C. Tröster, EPS-2021-511-ORG

Lam, H.H.W., *Lonely-ship:The emergence and experience of leader loneliness*, Supervisors: Prof. S.R. Giessner, Prof. M. Shemla & Dr M.D. Werner, EPS-2023-525-ORG

Legierse, W., *The Timing and Pricing of Initial Public Offerings: Evidence from the Low Countries*, Supervisors: Prof. A. de Jong & Prof. P.G.J. Roosenboom, EPS-2022-543-F&A

Leung, Y.K., *The Mind with a Touch of Madness? Mental health and well-being of entrepreneurs*, Supervisors: Prof. A.R. Thurik & Prof. I.H.A. Franken, EPS-2022-546-S&E

Li, Wei., *Competition in the Retail Market of Consumer Packaged Goods*, Supervisors: Prof. D. Fok & Prof. Ph.H.B.F. Franses, EPS-2021-503-MKT

Liu, M., Essays on Financial Disclosure and Innovation, Supervisors: Prof. J.P.M. Suijs & Dr P.Y.E. Leung, EPS-2023-582-F&A

Lieshout, R. van, *Integration, Decentralization and Self-Organization Towards Better Public Transport*, Supervisors: Prof. D. Huisman, Dr P.C. Bouman & Dr.ir J.M. van den Akker, EPS-2022-547-LIS

Liu, W., *An indigenous perspective on institutions for sustainable business in China*, Supervisors: Prof. P.P.M.A.R. Heugens & Dr. F.H. Wijen, EPS-2023-568-S&E

Mazzola, F., *Externalities in Economics and Finance: Essays on spillover effects and economic decisions*, Supervisors: Prof. W.B. Wagner & Prof. D.G.J. Bongaerts, EPS-2023-573-F&A

Mobini Dehkordi, Z., *Supply Chain Coordination with Separating and Pooling Contracts*, Supervisors: Prof. A.P.M Wagelmans, Dr W. van den Heuvel, EPS-2022-554-LIS

Musa, SM., *Making a Life on the Margins: An ethnographic account from Kutupalong*, Supervisors: Prof. P.P.M.A.R. Heugens & Prof. L. Berchicci, EPS-2024-586-S&E

Nikulina, A., *Interorganizational Governance in Projects: Contracts and collaboration as alignment mechanisms*, Supervisors: Prof. J.Y.F. Wynstra & Prof. L. Volker, EPS-2021-523- LIS

Novales Uriarte, A., *Thriving with Digitized Products: How Firms Leverage their Generative Capacity via Experimentation, Learning, and Collaboration*, Supervisors: Prof. H.W.G.M. van Heck & Prof. M.Mocker, EPS-2022-544-LIS

Orlandi, I., *Unringing the stigma bell: Investigating informational and social mechanisms behind boards of directors' appointments*, Supervisors: Prof. P.M.A.R. Heugens & Prof. V. F Misangyi, EPS-2023-564-S&E

Osrouti, F. El, *Norms versus Organizational Practices: Essays in the context of innovation, standardization, and religion*, Supervisors: Prof. H.J. de Vries & Prof H.W. Volberda, EPS-2024570-LIS

Pasparakis, A.M., *Leader-Follower Relationships in Technologically Advanced Operations*, Supervisors: Prof. M.B.M. de Koster & Dr J. de Vries, EPS-2023-571-LIS

Paul, J., *Online Grocery Operations in Omni-channel Retailing: Opportunities and Challenges*, Supervisors: Prof. M.B.M de Koster & Dr N.A.H. Agatz, EPS-2022-541-LIS

Paundra, J., *The Search for Alternatives to Private Vehicles The multi-perspective investigation of transportation mode choice preference based on policy, ICT development, and psychology*, Supervisors: Prof. W. Ketter, Dr J. van Dalen & Dr L. Rook, EPS-2023-496-LIS

Pocchiari, M., *Managing Successful and Resilient Shared-Interest Communities: The Role of Digitization Technologies and Disruptive Events*, Supervisors: Prof. G.H. van Bruggen & Dr J.M.T. Roos, EPS-2022-552-MKT

Ralcheva, A., *Crowdfunding: A Disruptive Innovation in Entrepreneurial Finance?* Supervisors: Prof. P.G.J. Roosenboom & Dr T. Lambert, EPS-2023-581-F&A

Ramezan Zadeh, M.T., *How Firms Cope with Digital Revolution: Essays on Managerial and Organizational Cognition*, Supervisors: Prof. H.W. Volberda & Prof. J.P. Cornelissen, EPS-2021-508-S&E

Ratara, C., *Behavioural and Neural Evidence for Processes Underlying Biases in Decision-Making*, Supervisors: Prof. A. Smidts & Dr M.A.S. Boksem, EPS-2022-548-MKT

Sluga, A., *Hour of Judgment: On judgment, decision making, and problem solving under accountability*, Supervisors: Prof. F.G.H. Hartmann & Dr M.A.S. Boksem, EPS-2021-520-F&A

Slob, E., *Integrating Genetics into Economics*, Supervisors: Prof. A.R. Thurik, Prof. P.J.F. Groenen & Dr C.A. Rietveld, EPS-2021-517-S&E

Speer, S.P.H., *The (Dis)Honest and (Un)Fair Brain: Investigating the Neural Underpinnings of Moral Decisions*, Supervisors: Prof. A. Smidts & Dr M.A.S. Boksem, EPS-2021-537-MKT

Stet, V.C., *(In)flexibility in Power Markets with Supply from Variable Renewable Sources*, Supervisors: Prof. J.T.J. Smit & Dr R. Huisman, EPS-2021-527-F&A

Stirnkorb, S., *Changes in the Information Landscape and Capital Market Communication*, Supervisors: Prof. E. Peek & Prof. M. van Rinsum, EPS-2021-536-F&A

Tierean, S.H., *Mind the Gap: The role of psychic distance and supplier's reputation in international buyer-supplier relationships*, Supervisors: Prof. C.B.M. van Riel, Prof. G. van Bruggen & Dr G.A.J.M. Berens, EPS-2022-551-ORG

Truong, H.M., *The Effect of Posted Prices on Sequential Auctions in B2B Multi-channel Markets*, Supervisors: Prof. H.W.G.M. van Heck, Prof. W. Ketter & Prof. A. Gupta, EPS-2021-539-LIS

Turturea, R., *Overcoming Resource Constraints: The Role of Creative Resourcing and Equity Crowdfunding in Financing Entrepreneurial Ventures,* Supervisors: Prof. P.P.M.A.R Heugens, Prof. J.J.P. Jansen & Dr I. Verheuil, EPS-2019-472-S&E

Vafa Arani, H., *Creating Shared Value: An operations and supply chain management perspective*, Supervisors: Prof. M.B.M. de Koster & Dr H. de Vries, EPS-2023-576-LIS

Waltré, E., *Leading for Performance in Adversity: Managing Failure, Negative Emotions, and Self-Threats*, Supervisors: Prof. D.L. van Knippenberg & Dr H.M.S. Dietz, EPS-2022-513-ORG

Wismans, A., *Behavioural Insights from the COVID-19 Pandemic Studies on Compliance, Vaccination, and Entrepreneurship*, Supervisors: Prof. A.R. Thurik & Prof. I.H.A. Franken, EPS-2023-562-S&E

Xiao, J., *Coordination & Control in Contemporary Organizations*, Supervisors: Prof. J. Cornelissen & Prof. D.A. Stam, EPS-2021-531-LIS

Yalcin, G., *Consumers in the Age of AI: Understanding Reactions Towards Algorithms and Humans in Marketing Research*, Supervisors: Prof. S. Puntoni & Dr A. Klesse, EPS-2022-550-MKT

Yang, A., *Corporate Bond Markets: Investor Preferences and Intermediary Frictions*, Supervisors: Prof. P. Verwijmeren & Prof. S. van Bekkum, EPS-2022-553-F&A

Zhang, Q., *Making sense of sensitivity in the workplace: Coping with contextual information in innovation and social networks*, Supervisors: Prof. D. Stam & Dr S. Tasselli, EPS-2023-574-LIS

Zhu, S., *Spare Parts Demand Forecasting and Inventory Management: Contributions to Intermittent Demand Forecasting, Installed Base Information and Shutdown Maintenance*, Supervisors: Prof. R. Dekker & Dr W.L. van Jaarsveld, EPS-2021-538-LIS

Zhou C., *Exploring The Role of Context and Interpretative Dynamics in Large-Scale Cross-Cultural Collaborations*, Supervisors: Prof. T. Simons & Dr M.D. Werner, EPS-2023-572-ORG

Zon, M. van, *Cost Allocation in Collaborative Transportation*, Supervisors: Prof. A.P.M. Wagelmans, Dr R. Spliet & Dr W. van den Heuvel, EPS-2021-530-LIS

In many real-life settings, the size and complexity of work allocation problems necessitate the use of operations research (OR) techniques. Traditionally, OR models have prioritised efficiency objectives, such as cost minimisation. However, there is a growing interest in methods that ensure fair work allocations. This thesis applies OR techniques to design models and methods that achieve fairness in work assignments.

The first part of this thesis focuses on railway crew planning, addressing practical problems that arise in the proposed crew planning process of Netherlands Railways. The first study considers tactical crew scheduling, introducing a Benders decomposition approach for robust template selection. The second study examines fair operational crew scheduling under the assumption that template-based rosters have already been constructed. It proposes a tailored column generation heuristic to construct individual crew schedules that are fair over time. The third study presents an efficient exact pricing algorithm to accelerate column generation algorithms for basic railway crew scheduling problems.

The second part of this thesis takes a more theoretical perspective, developing general optimisation methods for fairness-oriented work allocation. The first study centers on fairness over time in settings where work must be assigned online to homogeneous workers. It provides theoretical and experimental justifications for using an intuitive work allocation policy. The second study investigates branch-and-price methods for minimising the range and other order-based objective functions, introducing a generic branching rule that enables the use of classical, efficient branch-and-price methods for this type of problem.

Erasmus Research Institute of Management
Erasmus University Rotterdam
Burgemeester Oudlaan 50
Mandeville (T) Building
3062 PA Rotterdam, The Netherlands

P.O. Box 1738
3000 DR Rotterdam, The Netherlands
T: +31 10 408 1182
E: info@erim.eur.nl
W: www.erim.eur.nl
www.eur.nl/ese