Exact and heuristic methods for optimization in distributed logistics

Schrotenboer, Albert

DOI:
10.33612/diss.112911958

Exact and Heuristic Methods for Optimization in Distributed Logistics

Albert Harm Schrotenboer

Publisher: University of Groningen
Groningen, The Netherlands

Printed by: Ipskamp Printing
Enschede, The Netherlands

ISBN: 978-94-034-2290-9 (printed version)
978-94-034-2289-3 (electronic version)

university of groningen

Exact and Heuristic Methods for Optimization in Distributed Logistics

PhD thesis

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus Prof. C. Wijmenga
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

Thursday 6 February 2020 at 14:30 hours

by

Albert Harm Schrotenboer

born on 7 July 1991
in Hoogeveen

Supervisor
Prof. I.F.A. Vis

Co-supervisor
Dr. E. Ursavas

Assessment committee
Prof. R. H. Teunter
Prof. M. W. P. Savelsbergh
Prof. R. Dekker

To Aukje

You will always leave a light on in our hearts

Contents

Chapter 1

Introduction

The embracement of technological innovations by leading businesses in the field of (distributed) logistics increases the importance of efficient and effective planning and control of operations, which is more than ever determining whether or not businesses will survive in this field of fierce competition. Next to this, increasing awareness for sustainability has emerged among businesses and society, which requires a paradigm shift with regards to the current operational, tactical, and strategic decision making. Both the technological innovation and the need for sustainable practices have led to a transformation of the logistics sector in the last (few) decade(s). This transformation is not limited to specific subfields; it can clearly be observed throughout society with examples including, but not limited to, the construction of large offshore wind farms causing the need for advanced maintenance service logistics concepts (Shafiee 2015), renewed thinking within warehouse fulfillment operations on how to deal with the typically high number of returned products in the e-commerce era (Boysen, De Koster, and Weidinger 2019), and the need for robust last-mile delivery operations in city logistics (Savelsbergh and Van Woensel 2016).

The basis of distributed logistics is the transportation of what is commonly called *commodities*, a general descriptor for freight, parcels, tools, people, technicians, and inventory (see, e.g., Savelsbergh and Sol 1995, Crainic 2000). The specific characteristics of the commodities determine, to a large extent, how efficient operations should be organized efficiently. For instance, when looking at the characteristics of the involved vehicles, freight transportation typically involves a combination of full-truckload transportation for the long-haul, and less-than-truckload transportation for the short-haul. But trucks used for freight transportation are generally not suitable for last-mile parcel delivery operations in inner cities. Furthermore, other requirements

resulting from the specific problem context can be restricted delivery times (e.g., in the case of elderly transportation) or inaccessibility restrictions in case of offshore wind technician transportation. Hence the design of efficient and effective planning and control of operations depends predominantly on the specific context.

However, there is common ground between all distributed logistics operations which justifies studying, from an Operations Research perspective, distributed logistics problems on a higher, abstract level of modeling and contextualization. Namely, from this Operations Research perspective, we consider *distributed logistics* as the process in which we employ strategic, tactical, operational, and real-time decision making to transport commodities from one or multiple origin locations to one or multiple destination locations while controlling for problem-specific restrictions, in order to provide viable solutions, either in cost, time or other objectives, for the business problem at hand.

Here, Operations Research refers to the art of identifying today's essential business practices and translating them into mathematical optimization models that grasp the crucial aspects that impact decision making. Consequently, it entails the design of solution approaches to solve these mathematical optimization models efficiently and effectively, and using that, to provide guidance and support for improved decision making. Contributions within Operations Research can reflect any part of this process, from identifying crucial aspects within current business practices to new or enhanced solution methodologies to provide insights for complex optimization models that were thought to be unsolvable and uninterpretable before.

In this thesis, we study mathematical optimization problems inspired by new operations and renewed thinking in two areas of distributed logistics. We first present three studies on maintenance service logistics for offshore wind farms, and second, we present three studies that are inspired on emerging applications of e-commerce investigating last-mile delivery network design and order fulfillment operations within warehouses. All the studies' underlying optimization problems are addressed from a Mixed Integer Programming (MIP) point of view. That is, each problem considered in this thesis is formulated as an MIP, possibly with exponentially many constraints and variables. Afterward, suitable solution methods are developed to solve the optimization problems efficiently. The major contribution of each chapter is, therefore, the development of sophisticated solution approaches to solve the associated optimization problems. Such approaches are exact algorithms (e.g., branch-and-price), MIP-based reformulations (e.g., MIPs to obtain upper bounds), or metaheuristic methods (e.g., adaptive large neighborhood search). This allows us to perform exhaustive numerical experiments, which provide insights on the problem dynamics and can be used to

provide decision support for practitioners. In the following, we give a brief outline of each chapter's main contributions, the corresponding solution methods, and the overall relation between the chapters.

In the first part of this thesis, concerning Chapters 2-4, we study the maintenance planning problem at offshore wind farms from three different perspectives.

We first focus on a single wind farm scenario for which we develop exact solution methods to determine an optimal short-term maintenance planning. We provide the first column-generation based, exact solution method in this area and enhance its efficiency by including novel valid inequalities. Due to the nature of the problem, established methods based on dominance criteria and labeling algorithms do not suffice to create an efficient exact algorithm. We, therefore, develop a tailored method that is not based on dominance criteria to generate columns. Exhaustive numerical experiments confirm the efficiency of this method. In addition, we show that we can add the novel valid inequalities *while* we generate new columns just as efficiently as adding these valid inequalities *after* generating new columns. In the end, we demonstrate that we can solve problems of a practical size that were deemed as unsolvable before.

In Chapter 3, we extend this view towards a multiple wind farm setting. We provide a solution for the short-term maintenance planning problem and evaluate the cost-saving potential of coordinated operations in the case of multiple wind farms. As the underlying optimization model is computationally intractable with exact solution methods, we provide a metaheuristic approach based on adaptive large neighborhood search. Using exhaustive numerical experiments, we show that the developed approach provides high-quality solutions. In addition, coordinated operations result in using the scarcely available technicians more efficiently, which leads to fewer vessel trips while decreasing the mean time to maintenance.

Both studies mentioned above consider deterministic settings, which is a reasonable choice for short-term decisions. Chapter 4 takes a broader view on maintenance planning by studying tactical decision making under uncertainty. In a setting of multiple wind farms, we study how to allocate a given fleet to minimize total costs. It has two main contributions. First, we discuss the impact of modeling assumptions commonly used in the literature on decision making at an operational level and show how it affects the computational tractability. We show that established modeling techniques, although being computationally efficient, lead to overestimations of the total maintenance costs. Second, we consider the most important stochastic elements in offshore wind maintenance planning, namely, the uncertainty of the maintenance tasks and the uncertainty of weather conditions. Using Sample Average Approximation, we solve a large-scale, scenario-based reformulation of the two-stage stochastic mixed

integer programming model. Extensive numerical experiments show that the value of the stochastic solution is large, and henceforth stochastic elements should be considered in tactical or strategic decision making for offshore wind maintenance service logistics.

The second part of this thesis, concerning Chapters 5-7, is inspired by new developments in e-commerce logistics. We investigate two major problems from an operational and a network design point of view. These are discussed in three distinct chapters.

First, in Chapter 5, we study order-picker operations in picker-to-parts warehouses. We incorporate the (re)stocking of (returned) products in the traditional warehouse order-picking problem. Whereas the order-picking problem in isolation is a tractable optimization problem, the incorporation of the restocking returned products makes it hard to solve. Next to that, we discuss how multiple order pickers can be routed in such a way that their interaction (i.e., the number of times they cross) is minimized. For both the case with and without order-picker interaction, we develop an efficient genetic algorithm that provides high-quality solutions. We show that incorporating the restocking of returned products is efficient. Moreover, we show that there exist multiple structurally different near-optimal solutions so that interaction can be kept at a minimum by only increasing travel costs slightly.

In Chapter 6, we extend the setting studied in the previous chapter by considering integrated warehouse order-processing operations in picker-to-parts warehouse. Namely, we consider a joint order-picker routing, batching, and scheduling problem. That is, we design order-picker batches, route these batches, and assign and sequence these batches to order pickers. To solve practically sized instances of up to 8000 order lines, we develop a parallel adaptive large neighborhood search. By performing exhaustive numerical experiments, it is shown that incorporating product returns in rich warehouse operations still has the potential to reduce overall costs significantly. In addition, we investigate the benefit of splitting-up order lines belonging to a single customer order among multiple order picking batches. This shows promising results, with potential cost-savings around 45%.

Whereas we studied optimization problems arising in warehouse fulfillment operations in the previous two chapters, we take a more strategic look in Chapter 7. Namely, we investigate how to design robust city logistics distribution networks. We provide a general model of commodity streams between city distribution centers, which can be interpreted as a network design problem for organizing last-mile delivery operations. To the best of the authors' knowledge, we are the first to consider two-stage robust solutions within this context. Based on this robust-optimization paradigm (see, e.g. Ben-Tal and Nemirovski 2002, Gorissen, Yanıkoğlu, and Den Hertog 2015) in which one aims to minimize worst-case realizations, we consider taking strategic network

design decisions in such a way that the worst-case operational costs are minimized. We introduce the concept of time-invariant vehicle paths, where a sequence of locations to be visited is determined in the first stage while the associated departure times and the commodities to be transported are determined after observing uncertain parameters. We provide a general two-stage robust integer programming formulation and a large-scale, scenario-based reformulation. We compare this with a single-stage static variant that serves as an upper bound. We show that robust networks are obtained using dynamic decision making with time-invariant vehicle paths.

1.1 The Operations Research perspective

Having discussed the main contributions of this thesis in general terms, we now discuss the common principles from a theoretical Operations Research perspective.

The transportation of commodities between origin and destination locations is the subject of study in two classical Operations Research Problems. First, we have the Pickup and Delivery Problem (PDP), which consists in finding cost-minimizing vehicle tours so that all commodities are transported between their corresponding origin and destination locations (see, e.g., Savelsbergh and Sol 1995, Ropke, Cordeau, and Laporte 2007). The structure between the commodities' pickup and delivery locations can take many forms, as is described by the overview articles by Berbeglia et al. (2007); Parragh, Doerner, and Hartl (2008); and Battarra, Cordeau, and Iori (2014). In Chapters 2 and 3, we study the so-called Maintenance Service Planning and Routing Problem (MSPRP) and the Technician Allocation and Routing Problem (TARP), respectively. Following the classification by Battarra, Cordeau, and Iori (2014), these are a multi-period and multi-commodity variant of the PDP (Chapter 1) and a multi-period, multi-commodity, multi-depot variant of the PDP (Chapter 2). Both the MSPRP and the TARP have pickup-and-delivery structures which can be categorized as a novel mix of traditional pickup-and-delivery structures. Namely, technicians are on a daily basis picked up from a depot, delivered (and transported between) wind turbines, and brought back to the depot. We refer to Chapters 2 and 3 for a detailed classification. In the context of warehouse fulfillment operations, we study two variants of a PDP as well. In Chapter 5 we consider an order-picker routing problem with product returns, where in Chapter 6 we extend this by considering batching customer orders and scheduling them. These problems can be considered as a one-to-many-to-one PDP (Chapter 6), and a one-to-many-to-one, multi-trip, PDP with deadlines (Chapter 7). We refer to the actual chapters for a detailed explanation of the pickup-and-delivery structure.

In Chapters 3 and 7, we focus on (multi)commodity network design problems. In such problems, we need to open a set of cost-minimizing links in a network to transport commodities from origin to destination locations over those opened links. The main difference with the field of PDPs is that network design problems typically model more high-level, or strategic, decisions so that the day-to-day operations can be performed efficiently within the designed networks. In other words, whereas in PDPs we model the detailed movement of the vehicles (or any other mode of transportation) operating on arcs in the graph (we need to *route* the vehicles), in classical network design problems we model that vehicles are available to operate a link in the network only and do not detail (and model) further operational characteristics (i.e., no *routing* of vehicles).

In Chapter 3 we design optimal networks for tactical planning problems in offshore wind maintenance service logistics. The second-stage problem of the stochastic optimization model we propose is a network design problem with side constraints. Opening a link in the considered (time-expanded) network indicates to perform maintenance with a particular vessel at a particular wind farm. In Chapter 7, we consider a classical multi-commodity network design problem with temporal characteristics, that is, the commodities to be transported from origin to destination can only be transported within an uncertain delivery window.

It is clear that studying emerging distributed logistics problems inspired by the embracement of new technology by the logistics sector, and studying new distributed logistics problems in novel application areas such as offshore wind energy, is both practically relevant as well as theoretically challenging. All the discussed optimization problems contained in this thesis are inspired by observations in practice that did not receive the required research attention before. Although we will describe the practical relevance in detail in Sections 1.2 and 1.3, let us shortly summarize it here as well.

Offshore wind maintenance service logistics has particular characteristics for which the extant literature did not provide solutions on how to design a short-term and medium-term maintenance planning efficiently. Compared to onshore operations, vessels are more costly and structurally different than the vehicles (e.g., vans) deployed for onshore maintenance operations (see, e.g., Irawan et al. 2017). Regarding e-commerce applications, many new developments take place including the large stream of product returns and the design of distribution networks to enable high customer satisfaction in dense inner-cities. We propose novel concepts in this area to help the sector stay efficient and sustainable. Namely, we study the integrated processing of product returns and customer orders in warehouses as well as the design of robust logistics networks to circumvent daily uncertainty in operations (see, e.g., Boysen,

De Koster, and Weidinger 2019, Sampaio et al. 2019).

Hence, novel ideas and efficient and effective solution methodologies can have a real impact on society. We, therefore, conclude that it is an exciting era to work on problems in transportation and logistics, a field where the embracement of innovation in technology is yet only at the beginning. This provides researchers and practitioners numerous challenges where the extent to which those are dealt with efficiently will determine whether or not the involved businesses will still be known to the public in a couple of years.

In the remainder of the introduction, we will discuss both application areas (i.e., offshore wind and e-commerce logistics) in more detail. There, we discuss from a practical and theoretical point of view our contributions in more detail. We conclude the introduction with a brief overview of the manuscripts that form the basis of the remaining chapters in this thesis as well as other related manuscripts.

1.2 Offshore wind maintenance logistics

The offshore wind service logistics sector is challenging, risky, and expensive, with issues related to service logistics and maintenance accounting for 25-30% of the costs incurred during the operational phase (Röckmann, Lagerveld, and Stavenuiter 2017). In the Netherlands, four offshore wind farms are currently operational (Offshore WindPark Egmond aan Zee, WindturbinePark Prinses Amalia, Luchterduinen, and Gemini). The total installed capacity in the European Union is expected to increase by 19.1% yearly, increasing the current installed capacity of 6.5 GW to a total of 150GW in 2030 (GL Garrad Hassan 2013). However, the costs of offshore wind energy are higher than the costs of energy produced by traditional power suppliers based on coal and gas. Numerous initiatives have taken place to become competitive, for example, the green deal offshore wind Topteam Energie (2012), EU funded projects (see, e.g., EY 2015), and research projects funded by the Dutch Organization for Scientific Research (NWO). Only recently, the first wind farms will be built without direct subsidies that guarantee minimum energy prices. However, this is partly driven by low interest rates and cheap steel prices (Schrotenboer 2019). Still, more sustainable cost reductions are required, both during the installation phase and the operational phase. We will focus on the latter in this thesis, and refer to papers by Vis and Ursavas (2016) and Fischetti and Fraccaro (2019) for information on the installation phase.

A promising way to reduce the costs of offshore wind energy is the optimization of its logistics network and the operations that take place within the network at the operational phase, i.e., maintenance service logistics (Shafiee 2015, Shafiee and

Sørensen 2017). The operations that take place within the logistics network should be neatly coordinated with respect to the onshore and offshore flow of tools, modes of transportation, spare parts, and technicians (see, e.g. Irawan et al. 2017). Consequently, the logistics network design must offer opportunities for effective, efficient, and robust maintenance service logistics under any circumstance. For example, sophisticated workforce scheduling approaches are not useful if suitable modes of transportation are not available at the right spot and at the right time. Hence, it is of utmost importance that optimization models in the context of offshore wind maintenance service logistics grasp the essential ingredients regarding the coordination of the different material and technician flows. This will help the offshore wind sector to become truly competitive with the traditional energy suppliers and will make it an attractive, sustainable energy source for the coming decades (Welte et al. 2018).

In offshore wind maintenance service logistics, traditional onshore operations are mixed with novel offshore operations. These offshore operations are barely studied from an Operations Research perspective. The onshore operations include aspects that are visible in any supply-chain with examples including factories of original equipment manufacturers, warehouses of spare parts, and the distribution of parts surrounding those. On the offshore part, however, we observe new and comprehensive logistics optimization problems involving the transportation and routing of differently skilled technicians by a wide variety of vessels from the Operations and Maintenance Base (typically a port) to the offshore wind farms on a daily or (bi)weekly basis (see, e.g., Dai, Stålhane, and Utne 2015, Welte et al. 2018).

These operations have four major challenges which are structurally different from traditional maintenance service logistics operations. First, performing maintenance at offshore wind farms requires the chartering of expensive vessels and helicopters to transport the materials and technicians demanded for performing maintenance tasks, whereas in the traditional (onshore) applications service vehicles are relatively affordable. In addition, whereas in offshore operations such service vehicles are simply assigned to technicians at the beginning of each day, a considerable effort must be made to coordinate the vessels' movements with the technicians' movements to perform maintenance efficiently. Third, in addition to difficulties with regards to transportation, the coordination of technicians and spare parts is crucial; delivering a technician to a wind turbine without having brought the required supplies causes major disruptions of the operations as going back to the Operating and Maintenance base is expensive. Fourth, even if suitable modes of transportation are arranged, and different workflows are coordinated well, the daily maintenance activities are due to safety reasons affected by the weather conditions. Wind speed, wave height, and fog determine the extent to

which maintenance and transportation are allowed.

The attention to optimizing operations in the aforementioned context of offshore wind maintenance service logistics is scarce. The proposed solution methods for determining, for example, short-term maintenance planning problems are typically simulation oriented resulting in practically oriented decision support tools (see, e.g. Hofmann 2011), or focus on reliability engineering aspects such as condition monitoring (Uit het Broek et al. 2019). Although a recent increase in attention to optimizing underlying logistics operations has been observed (Dai, Stålhane, and Utne 2015, Stålhane, Hvattum, and Skaar 2015, Irawan et al. 2017), this can only be seen as a beginning of a new and exciting research field where benefits of many advanced methods (e.g., column generation or metaheuristics) are still unknown. We, therefore, continue with the development of new Operations Research models and accompanying methods to capture the above-defined challenges in the design of efficient and effective logistics network and maintenance service logistics design. We believe this will contribute to obtaining a thorough understanding of the crucial aspects that, if dealt with efficiently, will help the offshore wind sector to reduce costs and become competitive with the traditional energy suppliers.

In Chapters 2-4, we study optimization problems inspired by maintenance operations in offshore wind, aiming to obtain a thorough understanding of its dynamics.

In Chapters 2 and 3, we consider the trade-off between transportation costs, technician costs, and maintenance costs. In Chapter 2, we then consider a single wind farm setting for which we provide optimal solutions, and in Chapter 3, we consider a multiple wind farm setting and allow for collaborative operations between these wind farms. The developed algorithms can readily be applied to determine the short-term maintenance planning for offshore wind farms. This will reduce the total costs of the daily maintenance operations, and thereby will help the offshore wind sector to become competitive with traditional energy suppliers.

In Chapter 4, as mentioned before, we study a more tactical decision model. In addition to the model and the proposed solution approach, we evaluate the impact that assumptions on an operational level might have. This is of utmost importance for designing decision support tools to estimate the total maintenance costs on a medium-term horizon. Moreover, we take the viewpoint of a single maintenance provider that is solely responsible for performing maintenance according to prespecified conditions in so-called maintenance service requirements. By studying multiple variants of these requirements, we resemble different incentive schemes observed in practice, something which by the best of the author's knowledge has not been investigated before in this area.

1.3 E-commerce operations

The ever-increasing growth of e-commerce companies on the one hand, and the continuation of urbanization on the other hand, led to high pressure on organizing city logistics operations in such a way that acceptable levels of congestion, safety and sustainability are guaranteed (Savelsbergh and Van Woensel 2016). The total worldwide e-commerce sales in the business-to-customer segment have raised to 2304 billion dollars in 2017 and are expected to be 4878 billion dollars in 2020. This is only a small part of the total amount of internet-based sales, which equal an estimated 29000 billion dollars in 2017 (United Nations 2019). Regarding the business-to-customer segment, a large part of the sales is made by a few established companies such as Amazon or Alibaba. This has led to a high degree of competition amongst e-commerce companies, both established and upcoming businesses, of which the profit margins are under pressure (Cardona et al. 2015). To stay competitive in this field of fierce competition, efficient and effective planning and control of operations are required that fully utilize the opportunities this evolving landscape of e-commerce offers.

Due to the adoption of new technologies, novel business models are developed that compete on the offered customer services (Morganti et al. 2014). Examples include same-day delivery (Ulmer and Thomas 2018), crowd-sourced delivery (Sampaio et al. 2019), the use of parcel lockers (Enthoven et al. 2019, Faugère and Montreuil 2018), and renewed thinking concepts such as the Physical Internet (Montreuil 2011). What is clear from most new business models is that they consist of operations with a relatively high level of dynamism compared to the classical (and often static) freight or parcel transportation. For such classical logistics operators, it is, therefore, of utmost importance to change their operations so that these dynamic operations can be dealt with efficiently. If this change cannot be done efficiently, consequences on two distinct levels will be observed. First, on an individual firm level, problems with the firms' profitability are to be expected, and they will go out of business with high probability. Second, on a broad-society level, it will lead to more congestion and less safety and sustainability in large and dense inner cities.

When focusing on particular operations, many new developments can be observed that are worth the attention of researchers and practitioners. For instance, the large number of sales related to e-commerce also has a downside: Many of the ordered products are being returned to their seller, leading to a large stream of product returns upward the supply chain (see, e.g., Boysen, De Koster, and Weidinger 2019). Although this phenomenon is widely observed in the last 20 years, no sustainable solutions have been designed yet that completely resolves these overhead costs or led to a decrease in

returns. Namely, 52% of the Dutch population returned a product in 2018[1], and this fraction equals 45% for the French and 40% for the British.

At some point in the supply chain these products need to be incorporated in the regular (i.e., downstream) supply chain operations. The fraction of returns observed at e-commerce companies is on average 30%[2], compared to only 8.8% for brick-and-mortar stores. In addition, there is a trend that customer orders consists typically of only a few products (Chen, Wei, and Wang 2018). Both characteristics, i.e, the product returns and the large number of small-sized customer orders, are of key-importance when designing future warehouse order processing (and product return) operations.

Taking a bird's eye of view, this increase of e-commerce activities and especially the increasing popularity of same (or next) day delivery services causes high pressure on operations between distribution centers in large and dense inner cities. Such a city logistics network should be designed in such a way that it is efficient and effective for daily fluctuating demand patterns between city distribution centers. The design of such robust operations within a city distribution network is very complicated, as this network is only a small part of the overall supply chain. On a lower level, the last-mile delivery from city distribution centers to the customer's preferred pickup location needs to take place within promised delivery windows (see, e.g. Campbell and Savelsbergh 2006). On a higher level, parcels might be required on the same or next day in cities elsewhere, requiring long-haul transportation that might be planned separately from the city logistics operations (see, e.g., Crainic 2000). The daily planning of these external operations might differ, and the temporal characteristics (e.g., the earliest possible pickup time of parcels) is therefore typically uncertain. The extent to which such operations are dealt with efficiently is of key importance to achieve a high level of satisfied customers.

The aforementioned two phenomena, i.e., the need for robust city-logistics networks and the change in customer order characteristics, are the motivation for studying three distinct optimization problems that aim to provide decision support for the discussed challenges and opportunities.

In Chapters 5 and 6, we investigate the cost-savings potential of incorporating the restocking of product returns in regular order-picker operations. In Chapter 5, we study this problem in isolation for a single order picker. There we show that this can be done efficiently, and it is, therefore, of interest for operations managers in e-commerce warehouses to exploit if the quantified cost-savings will suffice in their actual operations. In Chapter 6, we extend this setting by considering multiple order

[1] https://www.statista.com/chart/16615/e-commerce-product-return-rate-in-europe/
[2] https://www.invespcro.com/blog/ecommerce-product-return-rate-statistics/

pickers for which we need to design batches, route the batches, assign the batches to order pickers and sequence the batches for each order picker. There it is shown that the cost-savings potential advocated in Chapter 5 is still attainable in richer, and more practical, settings.

In addition, we study two fundamental questions inspired on the developments in e-commerce. First, in Chapter 5, we develop a method that is capable of avoiding any interaction (e.g., picker blocking) between the deployed order pickers. We show that there are many structurally different optimal solutions that allow for the incorporation of additional objectives while order-picker routes increase in length only slightly. Second, in Chapter 6, as customer orders typically consist of only a few order lines, we study whether or not splitting-up order lines of the same customer exhibits enough cost-savings potential to compensate for additional handling operations further downstream the operations. We show that remarkable cost-savings can be obtained if the unavoidable additional expenses due to splitting up such customer orders (e.g., multiple deliveries to the customer or additional sorting efforts) are not too large. Hence, when designing future warehouses of e-commerce companies, both the inclusion of product returns in the regular order picking processes and (to a certain extent) the possibility to split up customer orders within the warehouse operations should be considered carefully.

In Chapter 7, we consider the design of robust city logistics networks while we control for uncertain temporal aspects. We study the composition of robust city networks and show that additional flexibility is obtained by considering two-stage robust solutions compared to static, single-stage robust solutions. The concept of time-invariant vehicle paths is shown to be efficient, thereby providing a practical way to organize future city-logistics operations.

1.4 Overview of manuscripts

Between September 2015 - August 2019, I have been pursuing my PhD Degree in Operations Research at the University of Groningen. This led to a wide variety of manuscripts that are under final preparation, under review, revised and resubmitted, or accepted at international scientific journals.

What follows is a brief overview of the manuscripts that I have worked on in the last four years. Not all manuscripts are part of this thesis, but they are closely related to distributed logistics as well. Namely, these additional manuscripts concern robust reserve-crew scheduling for airlines, multi-depot asymmetric vehicle routing, freight transportation network design, two-echelon vehicle routing, and continuous-

time network design for city logistics. Each manuscript is either accessible online or available upon request.

Manuscripts related to this thesis' chapters

1. Schrotenboer AH, Ursavas E, Vis IFA, 2019a *A branch-and-price-and-cut algorithm for solving resource constrained pickup and delivery problems. Transportation Science* 53(4):1001–1022

   (Chapter 2)

2. Schrotenboer AH, Uit het Broek MA, Jargalsaikhan B, Roodbergen KJ, 2018a *Coordinating technician allocation and maintenance routing for offshore wind farms. Computers & Operations Research* 98:185–197

   (Chapter 3)

3. Schrotenboer AH, Ursavas E, Vis IFA, 2019b *Mixed integer programming models for maintenance planning at offshore wind farms under uncertainty. Transportation Research Part C: Emerging Technologies* In press

   (Chapter 4)

4. Schrotenboer AH, Wruck S, Roodbergen KJ, Veenstra M, Dijkstra AS, 2017 *Order picker routing with product returns and interaction delays. International Journal of Production Research* 55(21):6394–6406

   (Chapter 5)

5. Schrotenboer AH, Wruck S, Vis IFA, Roodbergen KJ, 2019b *Integrating product returns and decomposition of customer orders in e-commerce warehouses.* Submitted

   (Chapter 6)

6. Schrotenboer AH, Ursavas E, Vis IFA, 2019c *Two-stage robust network design with temporal characteristics.* Working paper

   (Chapter 7)

Other manuscripts

7. Schrotenboer AH, Ursavas E, Zhu SX, Wenneker R, 2018b *Robust reserve crew recovery in air transportation: Reserve-crew scheduling to mitigate risks.* Submission in preparation
8. Uit het Broek MAJ, Schrotenboer AH, Jargalsaikhan B, Roodbergen KJ, Coelho LC, 2019 *Valid inequalities and a branch-and-cut algorithm for asymmetric multi-depot routing problems. CIRRELT, 2019-02* Revised and Resubmitted
9. Schrotenboer AH, Phoa TA, van der Heide G, Kilic OA, Buijs P, 2019a *A resource-efficient freight transportation network inspired by public transport.* Submitted

10. Enthoven DLJU, Jargalsaikhan B, Roodbergen KJ, Uit het Broek MAJ, Schrotenboer AH, 2019 *The two-echelon vehicle routing problem with covering options.* Revised and Resubmitted
11. Schrotenboer AH, Savelsbergh M, 2019 *Service network design for city logistics.* Working paper

Part I

Offshore wind logistics

Chapter 2

A branch-and-price-and-cut algorithm for resource constrained pickup and delivery problems

Abstract. *We study a multi-commodity multi-period resource constrained pickup-and-delivery problem inspired by the short-term planning of maintenance services at offshore wind farms. In order to begin a maintenance service, different types of relatively scarce servicemen need to be delivered (transported) to the service locations. We develop resource-exceeding route (RER) inequalities, which are inspired by knapsack cover inequalities, in order to model the scarcity of servicemen. In addition to a traditional separation approach, we present a column-dependent constraints approach so as to include the RER inequalities in the mathematical formulation. An alternative pricing strategy is developed to correctly include the column-dependent constraints. The resulting approach is broadly applicable to any routing problem that involves a set of scarce resources. We present a branch-and-price-and-cut algorithm to compare both approaches that include RER inequalities. The branch-and-price-and-cut algorithm relies on efficiently solving a new variant of the Elementary Resource Constrained Shortest Path Problem, using a tailored pulse algorithm developed specifically to solve it. Computational experiments show that the RER inequalities significantly tighten the root node relaxations. The column-dependent constraints approach searches then the branch and bound tree more effectively and appears to be competitive with the traditional separation procedure. Both approaches are able to solve instances of up to 92 nodes over 21 periods to optimality.*

This chapter is based on Schrotenboer, Ursavas, and Vis (2019a):
Schrotenboer AH, Ursavas E, Vis IFA, 2019a *A branch-and-price-and-cut algorithm for solving resource constrained pickup and delivery problems. Transportation Science* 53(4):1001–1022

2.1 Introduction

The short-term planning of maintenance services at geographically scattered locations is a frequently encountered optimization problem in maintenance service logistics. At the core of these optimization problems lies the daily planning and routing of a scarce set of resources in order to peform maintenance services. We will study such a problem at offshore wind farms, a fairly new area of optimization that has received the attention of researchers lately, and refer to the problem as the Multi-period Service Planning and Routing Problem (MSPRP) (Dai, Stålhane, and Utne 2015, Stålhane, Hvattum, and Skaar 2015). Typical for these optimization problems is the restricted availability of differently skilled servicemen, which may be viewed as a scarce set of resources needed for performing maintenance services. In this paper, we present effective valid inequalities to model the scarcity of resources, and, based on those inequalities, we develop an exact solution approach that relies on a new variant of column-dependent constraints. The resulting approach is broadly applicable to routing problems that consume such a scarce set of resources (e.g., the MSPRP). In addition, we will develop the first sophisticated exact solution approach for short-term maintenance planning at offshore wind farms. This enables us to study a setting without predefined planning restrictions, as opposed to current approaches in the literature.

In the MSPRP, a service is begun if the right amount of spare parts and the right number of differently skilled servicemen are delivered to the service location. After completion of the service, the servicemen need to be picked up again to be delivered to their next-scheduled service. These delivery and pickup tasks are performed by a heterogeneous fleet consisting of vessels and helicopters, each capacitated for the total weight of the spare parts and the number of servicemen. We will refer to this fleet by using the general term "vehicle". Note that the vehicles are not dedicated to a single serviceman, whereas, in onshore operations, vehicles are often "owned" by the servicemen: see, for example, Zamorano and Stolletz (2017); and Chen, Thomas, and Hewitt (2016). In addition, we let the travel costs and travel time be arbitrarily given for each vehicle and period, allowing the modeling of a wide variety of application dependent characteristics in a unified manner. For example, the different cost structures of corrective and preventive maintenance services (Stålhane, Hvattum, and Skaar 2015), as well as the influence of weather conditions on the maximum allowed travel time in each period (Kerkhove and Vanhoucke 2017), can be characterized in this way.

We assume that every service can be started and completed within a single period. Vehicles are allowed to continue with the remaining delivery and pickup tasks following the delivery of the servicemen at a service location, but they remain responsible

for the pickup of the servicemen delivered. Service times and designated maximum daily working hours of the servicemen need to be respected. The number of available servicemen is restricted in every period, that is, it can be considered as a resource whose total consumption among different vehicle routes must respect its limited availability.

We model the MSPRP as a multi-commodity, multi-period pickup-and-delivery problem. We aim to develop cost-minimizing routes such that spare parts and servicemen are picked up and delivered between service locations, for each vehicle in each period, assuring the start and completion of all maintenance services. We develop a new mathematical formulation based on Resource-exceeding Route (RER) inequalities that model the scarcity of servicemen (resources). The RER inequalities are included by means of column-dependent constraints. We will prove that the new formulation is stronger than a standard set-covering formulation for a broad class of instance characteristics. Its use is not restricted to the MSPRP; it is broadly applicable for routing problems that involve a scarce set of restricted resources. In order to test the competitiveness of the column-dependent constraints approach, a traditional separation procedure for including RER inequalities is presented as well.

To include the column-dependent constraints in a branch-and-price (or branch-and-price-and-cut) framework, an alternative, and optimal, pricing strategy is proposed. The general performance of the branch-and-price-and-cut algorithm relies on efficiently solving pricing problems that are obtained by decomposing the problem for each vehicle and period. The pricing problems are a new variant of the Elementary Resource Constrained Shortest Path Problem (Irnich and Desaulniers 2005), and are solved by a tailored pulse algorithm (Lozano, Duque, and Medaglia 2015). We propose efficient lower bounds that are exploited in the pulse algorithm. Finally, the strength of the branch-and-price-and-cut algorithm is shown by solving a case for maintenance service logistics at offshore wind farms, which is a newly created situation and one practically inspired.

The remainder of this section will review the relevant literature and highlight the paper's contributions. First, we discuss recent developments in algorithms to solve mathematical formulations with column-dependent constraints. Second, we discuss some closely related pickup-and-delivery problems and their most recent exact solution approaches. Finally, we review recent work on short-term planning for maintenance services at offshore wind farms.

The first contribution of this paper is the formulation and use of a new variant of column-dependent constraints, that is, the RER inequalities. Column-dependent constraints are constraints that are generated for every column or variable (Feillet et al. 2010). Its use in column generation applications expressly reveals these difficulties; the

number of constraints grows with the number of columns, thereby causing identification issues when generating new columns. We are able to overcome this difficulty through the development of an alternative pricing strategy.

A framework for handling column-dependent constraints with a decomposition into two subproblems is developed in Muter, Birbil, and Bülbül (2013). This work has recently been extended to an arbitrary number of subproblems in Maher (2015). Unlike these studies, the structure that we study exhibits interaction between the variables generated in the different subproblems. By exploiting this specific problem structure, we are able to develop a general and optimal pricing strategy that is broadly applicable to resource-constrained routing problems.

Our second contribution is the development of a branch-and-price-and-cut algorithm for the MSPRP, a problem that exhibits a combination of multiple traditional pickup and delivery structures. Pickup and delivery problems, as reviewed in Berbeglia et al. (2007); Parragh, Doerner, and Hartl (2008); and Battarra, Cordeau, and Iori (2014), involve finding cost-minimizing routes to satisfy transportation requests between pickup and delivery locations. A particular class of pickup-and-delivery problem is the vehicle routing problem with pickups and deliveries, where a one-to-one relation between pickup and delivery nodes exists (Dumas, Desrosiers, and Soumis 1991, Savelsbergh and Sol 1995). In the MSPRP, a one-to-one delivery and pickup structure exists between nodes that represent the start and completion of a service. State-of-the-art solution approaches are developed in Ropke, Cordeau, and Laporte (2007), and Ropke and Cordeau (2009). The former introduced many valid inequalities and tested the performance in a branch-and-cut algorithm, where the latter developed a branch-and-price-and-cut algorithm. Another exact algorithm, based on dual ascent heuristics and a cut-and-column generation procedure, is developed by Baldacci, Bartolini, and Mingozzi (2011). The most recent work is presented by Gschwind et al. (2018), where the authors present new dominance rules that allow for a bidirectional labeling algorithm.

Another class of pickup and delivery problem exhibits a many-to-many pickup and delivery structure, that is, locations demand or supply one or multiple commodities and picked-up supply may be used to satisfy demand. This pickup and delivery structure is encountered in the MSPRP between different maintenance services, that is, the picked-up servicemen can then be used to begin a new maintenance service. A branch-and-cut approach for a single vehicle is presented in Hernández-Pérez and Salazar-González (2004). This has recently been extended to multiple commodities in Hernández-Pérez and Salazar-González (2014). Looking into the multiple types of servicemen and the heterogeneous fleet being present in the MSPRP, another

observation is made. If we consider a single servicemen type and a homogeneous fleet, the MSPRP can be categorized as a pickup and delivery problem with maximum travel time (Subramanian and Cabral 2008, Polat et al. 2015).

Summarizing, the MSPRP mixes a one-to-one pickup and delivery structure (between nodes representing the same service) with a many-to-many pickup and delivery structure (between different services). In addition, respecting the service times of the maintenance services yields so-called delayed precedence constraints between the delivery and pickup of servicemen, that is, the earliest possible departure time at the pickup location depends on the arrival time at the corresponding delivery location. This pickup and delivery relationship is typical for offshore applications, see, for example, Irawan et al. (2017).

The mix of traditional pickup and delivery structures leads to a new variant of the Elementary Resource Constrained Shortest Path Problem as a pricing problem. Because both travel costs and servicemen costs are being minimized, no efficient dominance criteria can be developed. We therefore propose an efficient pulse algorithm (Lozano, Duque, and Medaglia 2015) to solve the pricing problems, since that approach does not depend on dominance criteria. It relies on calculating lower bounds instead, which appears effective for the MSPRP.

The paper's third contribution is the development of a branch-and-price-and-cut algorithm in the area of offshore wind maintenance service logistics, which is the first sophisticated exact solution method in the setting we are studying. In the MSPRP we are studying a general, new setting of a single large offshore wind farm that is operated from a single depot without predefined planning restrictions. Some related studies exist, however; offshore wind farm maintenance service logistics was first encountered in Dai, Stålhane, and Utne (2015), and a follow up was presented by Stålhane, Hvattum, and Skaar (2015). They proposed a set covering formulation with a heuristic labeling algorithm to solve the pricing problems, but restricted it to a single period, whereas the MSPRP is situated in a multi-period setting. A first attempt to exactly solve realistically sized instances is presented by Irawan et al. (2017). They propose a route-enumeration strategy to solve up to eight maintenance services for three wind farms operated from two depots in a three-period planning horizon. The restriction that a route only contain services from a single wind farm reduces the complexity of the problem drastically, only at the expense of a slight increase in complexity due to the inclusion of multiple depots. In point of fact, only with a heuristic approach were they able to solve instances of up to 12 services per wind farm. Inherently, such a route enumeration approach is deemed impossible for the MSPRP, since we have no restrictions on the planning of services. With the branch-and-price-and-cut algorithm

we propose optimal solutions for instances of up to 45 services in a single wind farm.

More generally speaking, offshore wind maintenance service logistics, and thus the MSPRP, falls into a particular stream of the technician routing and scheduling literature (Pillac, Gueret, and Medaglia 2013, Pillac, Guéret, and Medaglia 2018). It entails the design of routes and schedules for technicians such that a set of services is performed in a cost-minimizing way. The main difference between the MSPRP and onshore applications (Paraskevopoulos et al. 2017) is how vehicles are operated; vehicles in offshore applications are flexibly deployed to satisfy transportation requests throughout the time horizon, whereas vehicles are typically assigned upfront to servicemen in onshore applications. Recently, technician's ability to become more experienced in an activity is discussed by Chen, Thomas, and Hewitt (2016). This is extended to the stochastic case in which activities are uncertain (Chen, Thomas, and Hewitt 2017). Another recent work discusses the combined maintenance and routing problem (López-Santana et al. 2016), in which machines deteriorate stochastically over time. We acknowledge that those innovations in onshore applications may be of relevance for offshore wind maintenance service logistics. However, since offshore operations differ structurally from onshore operations, and we present the first sophisticated approach for solving a large-scale maintenance service logistics problem in offshore wind farms, we leave it for further research to assess the impact of incorporating the earlier described onshore innovations.

The remainder of this paper is as follows. We give a mathematical description of the MSPRP and, by means of a Danzig-Wolfe reformulation, a set covering formulation in Section 2.2. In Section 2.3, we describe valid inequalities for the MSPRP. In particular, we discuss the RER inequalities, the new formulation based on column-dependent constraints, and the accompanying optimal pricing strategy. Sections 2.4 and 2.5 discuss the pulse algorithm developed for solving the pricing problems and the overall structure of the branch-and-price-and-cut algorithm, respectively. Computational experiments showing the performance of the branch-and-price-and-cut algorithm and the impact of the valid inequalities are presented in Section 2.6. We conclude the paper in Section 2.7.

2.2 Problem description

In this section, we will provide a mathematical representation of the Multi-period Service Planning and Routing Problem (MSPRP) in the form of a Mixed Integer Program (MIP). After this, a standard set-covering formulation will be presented.

2.2.1 Mixed integer programming formulation

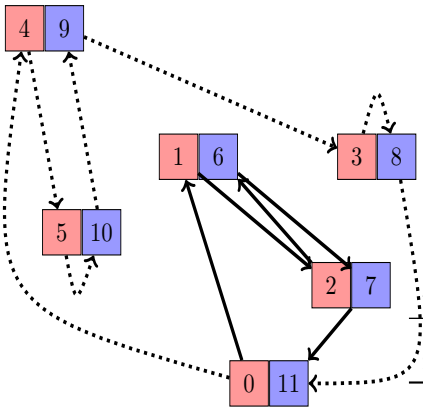
Let $G = (N, A)$ be a directed graph with a set of nodes N and a set of arcs $A = \{(i, j) \mid i, j \in N, i \neq j\}$. The node set N consists of delivery nodes $N_d = \{1, \ldots, n\}$, pickup nodes $N_p = \{n + 1, \ldots 2n\}$, and the origin and destination depot $\{0, 2n + 1\}$. Every delivery node i has a corresponding pickup node $n + i$, which represent the start and the completion of service i, respectively.

Let $\mathcal{T} = \{1, \ldots, T\}$ be the given time horizon in which every $t \in \mathcal{T}$ represents a single period. We consider a set of different type of servicemen $\mathcal{L} = \{1, \ldots, L\}$. The demand for service-person type $\ell \in \mathcal{L}$ at node $i \in N$ is given by $Q_{i\ell} \geq 0$, and it holds that $Q_{n+i,\ell} = -Q_{i\ell}$. The number of available servicemen is restricted; there are $\tilde{Q}_\ell$ servicemen of type ℓ available in each period. The fixed costs of using a service-person of type ℓ equals $\tilde{c}_\ell$ for each period. A precedence constraint exists between node i and $n + i$; node $n + i$ can be visited, at the earliest, s_i time after visiting node i, where s_i denotes the duration of service i. The weight of the demanded spare parts is given by $\widehat{Q}_i > 0$ for each $i \in N_d$.

A heterogeneous set of capacitated vehicles $\mathcal{K} = \{1, \ldots, K\}$ is available to deliver and pickup the required servicemen in each period. As is typical for offshore operations, we assume that all vehicles are different. For each arc $(i, j) \in A$, the costs incurred of traversing it with vehicle k in period t equals c_{ij}^{kt} and the corresponding travel time equals t_{ij}^{kt}. Maintenance costs are included in the travel costs c_{ij}^{kt}, and we do not pose any restrictions on its modeling. This flexibility has two aims. First, it allows us to make a distinction between preventive maintenance tasks, in which maintenance costs are constant over the periods, and corrective maintenance tasks, in which maintenance costs increase over the periods. Second, we can model the relative urgency of the maintenance services, that is, higher costs reflect a greater urgency to perform a particular maintenance service. Both aims are easily achieved by introducing exogenously given penalty costs for not performing a maintenance service in a particular period (which could be zero).

Each vehicle $k \in K$ is capacitated in the total number of servicemen $\bar{Q}_k^1$ and the total amount of spare parts $\bar{Q}_k^2$ it can transport. The maximum travel time of vehicle k in period t equals ω_{kt}. This reflects the restrictions on performing offshore maintenance services due to weather conditions.

Let x_{ij}^{kt} be a binary decision variable that equals 1 if if vehicle k traverses arc (i, j) in period t, and 0 otherwise. Let $q_{i\ell}^{kt}$ be a nonnegative decision variable that indicates the number of servicemen of type $\ell \in \mathcal{L}$ in vehicle k in period t upon leaving node i. Finally, let z_i^{kt} be a nonnegative decision variable that equals the time at which vehicle k leaves node i at period t.



(a) Network and route visualization

$i, n+i$	s_i	Q_{i1}	Q_{i2}
1, 6	2.0	3	2
2, 7	3.0	1	2
3, 8	1.5	2	1
4, 9	1.5	3	1
5, 10	3.5	1	2

(b) Overview of instance characteristics

t	route	q_{01}^{1t}	q_{02}^{1t}	Dur.
1	$0-1-2-6-7-11$	4	4	4.5
2	$0-4-5-10-9-3-8-11$	4	3	7.5

(c) Route characteristics

Figure 2.1: An illustrative example of the MSPRP.

To highlight the complexity of the MSPRP, an illustrative example is provided in Figure 2.1.

Example 2.1. Let $n = 5$, $T = 2, K = 1$, and $L = 2$. Spare parts demand $\widehat{Q}_i$ equals 0 and servicemen availability $\tilde{Q}_\ell = 4$ for all $\ell \in \{1, 2\}$. Maximum travel time ω_{1t} equals 6 and 12 for $t = 1$ and $t = 2$, respectively. In Figure 2.1(b), characteristics of the services are provided, and in Figure 2.1(c), a feasible solution is depicted. "Dur." indicates the travel time of the corresponding route. We assume that all the arcs' travel times equal 0.5, except for the edges between corresponding delivery and pickup locations, those are assumed to take zero time in this example. Some calculations are as follows: 1) Regarding the duration of the route in Period 1. Let $a := t_{01} + t_{12} + \max\{t_{01} + s_1, t_{01} + t_{12} + t_{26}\}$ be the earliest possible departure from Node 6. The earliest possible departure from Node 7 is then $\max\{a + t_{67}, t_{01} + t_{12} + s_2\} := b$. The route duration is then equal to $b + t_{7,11}$, which equals 4.5 in this example. 2) The servicemen use in Period 2 is the sum of the servicemen used for Services 4 and 5, since Service 3 is supplied with servicemen that become available after having finished Services 4 and 5. 3). Note that a single route in Period 2 $(0-1-2-6-7-4-5-10-9-3-8-11)$ is a feasible solution as well, as its duration is less than 12 and the servicemen use equals the maximum of both individual routes. ◁

The following MIP models the MSPRP.

$$\min \quad \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \sum_{(i,j)\in A} c_{ij}^{kt} x_{ij}^{kt} + \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \sum_{\ell\in\mathcal{L}} q_{0\ell}^{kt} \tilde{c}_\ell \tag{2.1}$$

subject to

$$\sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}}\sum_{j:(i,j)\in A} x_{ij}^{kt} = 1 \qquad \forall i \in N_d, \tag{2.2}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{kt} - \sum_{j:(j,i)\in A} x_{ji}^{kt} = 0 \qquad \forall\, i \in N_d \cup N_p, k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.3}$$

$$\sum_{j:(0,j)\in A} x_{0j}^{kt} = 1 \qquad \forall\, k \in K, t \in \mathcal{T}, \tag{2.4}$$

$$\sum_{j:(j,2n+1)\in A} x_{j,2n+1}^{kt} = 1 \qquad \forall\, k \in K, t \in \mathcal{T}, \tag{2.5}$$

$$\sum_{j:(i,j)\in A} x_{ji}^{kt} - \sum_{j:(n+i,j)\in A} x_{n+i,j}^{kt} = 0 \qquad \forall\, i \in N_d,\ k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.6}$$

$$t_{ij}^{kt} x_{ij}^{kt} - M(1 - x_{ij}^{kt}) \le z_j^{kt} - z_i^{kt} \qquad \forall\, (i,j) \in A,\ k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.7}$$

$$z_i^{kt} + s_i \le z_{i+n}^{kt} \qquad \forall\, i \in N_d,\ k \in \mathcal{K},\ t \in \mathcal{T}, \tag{2.8}$$

$$Q_{j\ell} x_{ij}^{kt} - M(1 - x_{ij}^{kt}) \le q_{i\ell}^{kt} - q_{j\ell}^{kt} \qquad \forall\ (i,j) \in A,\ \ell \in \mathcal{L},\ k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.9}$$

$$\max\{0, -Q_{i\ell}\} \le q_{i\ell}^{kt} \qquad \forall\, i \in N_d \cup N_p,\ \ell \in \mathcal{L}, k \in \mathcal{K},\ t \in \mathcal{T}, \tag{2.10}$$

$$\sum_{\ell\in\mathcal{L}} q_{j\ell}^{kt} \le \min\{\bar{Q}_k^1, \bar{Q}_k^1 + \sum_{\ell\in\mathcal{L}} Q_{i\ell}\} \qquad \forall\, j \in\in N_d \cup N_p, k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.11}$$

$$\sum_{(i,j)\in A: j\in N_d} x_{ij}^{kt} \widehat{Q}_j \le \bar{Q}_k^2 \qquad \forall\, k \in \mathcal{K},\ t \in \mathcal{T} \tag{2.12}$$

$$z_{2n+1}^{kt} \le \omega_{kt} \qquad \forall\, k \in \mathcal{K}, t \in \mathcal{T}, \tag{2.13}$$

$$\sum_{k\in\mathcal{K}} q_{0\ell}^{kt} \le \tilde{Q}_\ell \qquad \forall\, \ell \in \mathcal{L},\ t \in \mathcal{T}, \tag{2.14}$$

$$x_{ij}^{kt} \in \{0,1\} \qquad \forall\, (i,j) \in A,\ k \in \mathcal{K},\ t \in \mathcal{T}, \tag{2.15}$$

$$q_{i\ell}^{kt} \ge 0 \qquad \forall\, i \in N,\ \ell \in \mathcal{L},\ k \in \mathcal{K},\ t \in \mathcal{T}, \tag{2.16}$$

$$z_i^{kt} \ge 0 \qquad \forall\, i \in N,\ k \in \mathcal{K},\ t \in \mathcal{T}. \tag{2.17}$$

Objective (2.1) minimizes the costs of traveling and for servicemen usage. The travel costs may include maintenance costs or penalty costs. Constraints (2.2) ensure that every node is visited only once and constraints (2.3) are the traditional flow conservation constraints. Constraints (2.4) and (2.5) ensure that every route starts and end at the origin and destination depot, respectively. The vehicle that delivers the servicemen must also pickup the servicemen, as denoted by Constraints (2.6).

Constraints (2.7) and (2.8) model travel and service times, respectively. In (2.7), M denotes a big enough number so that the constraints are redundant if they need to be. A valid value of M is ω_{kt}. Constraints (2.9) model the servicemen demand at every node. With Constraints (2.10) and (2.11) we strengthen the lower bound and upper bound of $q_{j\ell}^{kt}$, respectively. The maximum capacity for spare parts is respected due to Constraints (2.12). Finally, Constraints (2.13) limit the maximum travel time of a vehicle and Constraints (2.14) ensure feasibility with respect to the limited availability of servicemen.

The MIP formulation exhibits an interesting structure. It is decomposable for every vehicle $k \in \mathcal{K}$ and period $t \in \mathcal{T}$. The constraints that link the decisions among the (k,t)-subproblems are Constraints (2.2) and (2.14). We, therefore, apply a Dantzig-Wolfe reformulation resulting into a set-covering formulation as presented in the following section.

2.2.2 Set covering formulation

Let $\mathcal{R}$ be the set of all feasible routes that can be constructed in the MSPRP. A route's costs and feasibility may differ between vehicles and periods, since vehicles are heterogeneous and arc costs, as well as maximum travel times, differ among periods. Therefore, let $\mathcal{R} = \cup_{k\in\mathcal{K},t\in\mathcal{T}}\mathcal{R}_{kt}$, where $\mathcal{R}_{kt}$ denotes the set of feasible routes of vehicle k in period t. For notational convenience, let $\mathcal{R}_k = \cup_{t\in\mathcal{T}}\mathcal{R}_{kt}$ and $\mathcal{R}_t = \cup_{k\in\mathcal{K}}\mathcal{R}_{kt}$. For each route $r \in \mathcal{R}_{kt}$, let y_r be a binary decision variable that equals 1 if route r is chosen and 0 otherwise. In addition, let c_r be the corresponding costs, β_i^r be the number of times node $i \in N_d$ is visited, and γ_ℓ^r be the number of servicemen of type $\ell \in \mathcal{L}$ used by route $r \in \mathcal{R}_{kt}$.

A set-covering formulation is then given by:

$$\min \quad \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_{kt}} c_r y_r \tag{2.18}$$

$$\text{s.t.} \sum_{t\in\mathcal{T}}\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_{kt}} y_r\beta_i^r \geq 1 \qquad \forall\, i \in N_d, \tag{2.19}$$

$$\sum_{r\in\mathcal{R}_{kt}} y_r \leq 1 \qquad \forall\, k \in \mathcal{K},\ t \in \mathcal{T}, \tag{2.20}$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}_{kt}} y_r\gamma_\ell^r \leq \tilde{Q}_\ell \qquad \forall\, \ell \in \mathcal{L},\ t \in \mathcal{T}, \tag{2.21}$$

$$y_r \in \{0,1\} \qquad \forall r \in \mathcal{R}_{kt}, k \in \mathcal{K},\ t \in \mathcal{T}. \tag{2.22}$$

The objective (2.18) minimizes the costs for using the selected routes from each subset

$\mathcal{R}_{kt}$. Constraints (2.19) ensure that every node is visited at least once. Constraints (2.20) ensure that every vehicle in every period is used at most once, which is necessary due to the heterogeneity of vehicles and periods. Constraints (2.21) ensure that the maximum number of servicemen used in every period does not exceed the servicemen availability. We will refer to the model described by equations (2.18)-(2.22) as the Integer Programming Master (IPM) problem. Its linear relaxation, obtained by replacing Constraints (2.22) with $y_r \geq 0$, is referred to as the Linear Programming Master (LPM) problem.

Due to the exponential size of $\mathcal{R}$, solutions to LPM are usually obtained by column generation (Barnhart et al. 1998). To that extent, consider restricted route sets $\overline{\mathcal{R}}_{kt} \subset \mathcal{R}_{kt}$. Notice that by a Dantzig-Wolfe decomposition we arrive at $K \cdot T$ subproblems, that is, for vehicle k and period t we obtain the (k,t)-pricing problem. We iteratively solve LPM subject to $\overline{\mathcal{R}}_{kt}$ and generate new routes for each $\overline{\mathcal{R}}_{kt}$ by solving the (k,t)-pricing problems. Model LPM is solved if no route of negative reduced cost can be found for any (k,t)-pricing problem. Then, a dual optimal solution is found and by strong duality it is a primal optimal solution as well.

To formulate the (k,t)-pricing problems, let μ_i, λ_{kt}, and π_ℓ^t be dual variables corresponding to Constraints (2.19) - (2.21), respectively. Let d_{ij}^{kt} be the costs of traversing arc (i,j) in some (k,t)-pricing problem. We define

$$d_{ij}^{kt} = \begin{cases} c_{ij}^{kt} - \mu_j & \text{if } j \in N_d, \\ c_{ij}^{kt} & \text{otherwise.} \end{cases} \tag{2.23}$$

Similarly, let $\tilde{d}_\ell^t = \tilde{c}_\ell - \pi_\ell^t$ be the reduced servicemen costs in an arbitrary (k,t)-pricing problem. Then the (k,t)-pricing problems are given by

$$\min_{r \in \mathcal{R}_{kt}} \left\{ \sum_{(i,j)\in A} d_{ij}^{kt} r_{ij} + \sum_{\ell \in \mathcal{L}} \tilde{d}_\ell^t \gamma_\ell^r - \lambda_{kt} \right\}, \forall\ k \in \mathcal{K},\ t \in \mathcal{T}, \tag{2.24}$$

where $r_{ij} = 1$ for all arcs (i,j), $j \neq n+i$, which are used by path $r \in \mathcal{R}_{kt}$, and is 0 otherwise.

An illustrative example of a route in an arbitrary (k,t) pricing problem is presented in Figure 2.2.

Example 2.2. Let $K = T = L = 1$ and let $n = 5$. Consider three Services $1, 2$ and 3 that demand $2, 2$, and 3 servicemen, respectively. As observed from Figure 2.2, only $\tilde{c}_1 - \pi_1^1$ reduced servicemen costs are incurred when visiting Service 3 (instead of $3\tilde{c}_1 - 3\pi_1^1$) since 2 servicemen just become available after visiting node 7. Furthermore,

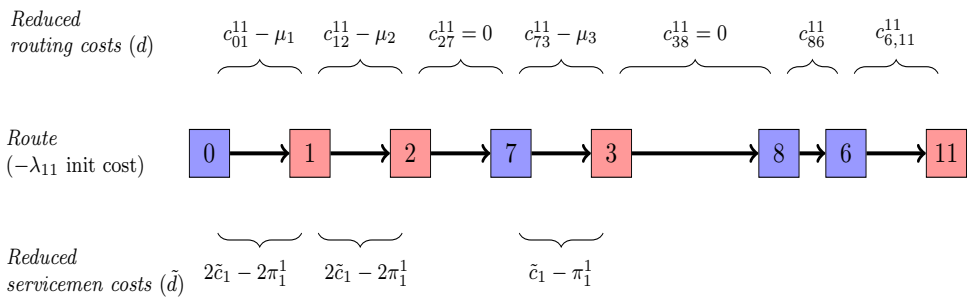


Figure 2.2: An illustrative example of a route in a (k, t) pricing problem.

note that the traveling costs between corresponding delivery and pickup nodes equals zero, and observe that only reduced servicemen costs are considered when visiting a delivery node. ◁

The (k, t)-pricing problems are new variants of the Elementary Resource Constrained Shortest Path Problem. In line with Irnich and Desaulniers (2005), it can be categorized as a multi-commodity resource constrained shortest path problem with delayed paired precedence constraints, a problem that, to the best of the authors' knowledge, has not been solved before. The delayed paired precedence constraints refer to the one-to-one relationship between pickup and delivery nodes in combination with the corresponding service time constraint. For simplicity, we refer to this new variant of the Elementary Resource Constrained Shortest Path Problem by using the general term Pricing Problem (PP). The main complicating factor that appears in (2.24) are the servicemen costs $\tilde{d}^t_\ell$, as is indicated in Figure 2.2 as well. These cannot be incorporated directly into individual edge costs, henceforth leading to algorithmic difficulties, as will be explained in detail in Section 4.

2.3 Valid inequalities

Model IPM, presented in Section 2.2.2, is a set covering formulation as encountered in many branch-and-price (and branch-and-price-and-cut) approaches. It is, although valid for solving the MSPRP, relatively weak due to the inclusion of knapsack-type Constraints (2.21). These constraints cannot be taken into account in the pricing problems directly, thereby reducing the overall efficiency, since it weakens the LP relaxation. In this section, we will develop *Resource-exceeding Route* (RER) inequalities, which are a specialized form of knapsack cover inequalities (Gu, Nemhauser, and Savelsbergh 1999,?). They are applied in branch-and-price algorithms for the generalized

assignment problem (Savelsbergh 1997) and multicommodity flow problems (Barnhart, Hane, and Vance 2000), for instance. The RER inequalities very efficiently restrict the number of servicemen that routes can use, since this is a resource "consumed" by vehicles whose its availability at the depot is restricted. What differs here from the current applications of cover inequalities is that the sequence of visits within a tour may change the resource consumption. This causes difficulties during the pricing of routes. The inclusion of multiple resource types complicates the construction of RER inequalities further, which can be observed in the remainder of this section.

We present two approaches for including RER inequalities that are tractable in pricing problems. In the first approach, we develop a new formulation of the MSPRP in which we replace Constraints (2.21) with column-dependent Constraints (Muter, Birbil, and Bülbül 2013). These constraints are generated when the restricted route set (as used in column generation) is enlarged. Its applicability is not restricted to this case only: The proposed reformulation and corresponding solution approach are applicable for any linear program being solved with column generation in which a system of constraints as described by (2.21) is present. The approach is easy to understand and it provides interesting theoretical insights into the inclusion of valid inequalities. In order to test its competitiveness, we propose a second approach that separates the RER inequalities by a traditional separation procedure, which is called upon dynamically during the branch-and-price-and-cut algorithm. So, where the first approach adds the RER inequalities *while* generating new routes, the second approach adds RER inequalities *after* generating new routes.

In the following, we will first develop the alternative formulation for the MSPRP by using column-dependent constraints. The column-dependent constraints are problem defining, and a valid formulation of the MSPRP is obtained by replacing the knapsack-type Constraints (2.21). We refer to this alternative formulation as the Alternative Integer Programming Master Problem (AIPM). We analyze the strength of the formulation and prove that for special cases its LP relaxation is stronger than LPM. We present an alternative pricing strategy and prove its optimality. After this, a description of a traditional separation procedure of RER inequalities is given, in order to be able to judge the computational performance of the column-dependent constraints approach. We will end this section with some well-known valid inequalities that are included in our exact algorithm as well.

2.3.1 Column-dependent constraints approach

By exploiting the notion of *resource-exceeding routes*, that is, routes that cannot simultaneously enter a feasible integer solution due to their servicemen use, we will replace the relatively weak Constraints (2.21) by a set of column-dependent constraints that restricts the use of resource-exceeding routes. Consider the following example. It illustrates the concept of resource-exceeding routes and of the corresponding valid inequalities.

Example 2.3. Let $L, T = 1$ and $\tilde{Q}_\ell = 6$. Consider routes $r_i \in \mathcal{R}_{it}, i \in \{1, 2, 3\}$ with $\gamma_1^{r_i} = 3$ such that $r_i = (0, i, n + i, 2n + 1)$. In other words, Vehicle 1 performs Service 1, Vehicle 2 performs Service 2, and Vehicle 3 performs Service 3. Then the constraint $\sum_{i\in\{1,2,3\}} y_{r_i} \leq 2$ is valid, whereas no restriction should be put on any subset of these routes. Consider now a fourth route r_4 with $\gamma_1^{r_4} = 4$. Then there are i valid inequalities of the form $y_{r_4} + y_{r_i} \leq 1$ for all $i \in \{1, 2, 3\}$. ◁

The valid inequalities shown in Example 1 are problem-defining for the MSPRP. A valid formulation of the MSPRP is obtained if we include those valid inequalities in IPM and leave out Constraints (2.21). In the following, we present an approach that generates such valid inequalities for any subset of routes from different vehicles, of size at least 1 and at most $K - 1$. For every subset, we include column dependent constraints that restrict the maximum number of allowed routes, so that the servicemen availability is respected.

2.3.1.1 Resource-exceeding route inequalities

Let $k(r)$ and $t(r)$ be the vehicle and period index of route $r \in \mathcal{R}_{kt}$, respectively. The complement route set R_r^C of route r is defined as the set of routes from different vehicles in the same period as r, that is, $\mathcal{R}_r^C := \{\tilde{r} \in \mathcal{R} \mid t(\tilde{r}) = t(r), k(\tilde{r}) \neq k(r)\}$. We call a set of routes a *partial solution* if it contains at most a single route for each vehicle and if it does not contain a route for all the vehicles. The collection of partial solutions is formally defined as follows:

Definition 2.1 (partial solution)**.** The collection $\mathbb{S}_t$ of partial solutions is defined as

$$\mathbb{S}_t := \{S \subseteq \mathcal{R}_t \mid |S_k| \leq 1 \ \forall k \in K, 1 \leq |S| \leq K - 1\}, \tag{2.25}$$

with $S_k = \{r \in S \mid k(r) = k\}$.

Now the collection of route sets $\mathbb{S}_t$ contains all partial solutions, and for each of those partial solutions we need to check how many of those routes can be present in a

feasible integer solution. Therefore, we let ϕ_S be the maximum number of allowed routes from $S \in \mathbb{S}_t$ in a feasible integer solution, that is.,

$$\phi_S := \max \left\{ |v| : \sum_{v \in V \subseteq S} \gamma_\ell^v \leq \tilde{Q}_\ell \ \forall \ell \in \mathcal{L} \right\} \tag{2.26}$$

We will continue with a brief example to clarify the intuition behind ϕ_S and $\mathbb{S}_t$.

Example 2.4. Consider the setting as in example 1 with routes r_1, r_2, r_3 and r_4. Two partial solutions are given by $S_1 = \{r_1, r_4\}$ and $S_2 = \{r_1, r_2, r_4\}$. Here $\phi_{S_1} = 1$ as only r_1 or r_4 could be present in a feasible integer solution. On the other hand, $\phi_{S_2} = 2$ as r_1 and r_2 can both be present in a feasible integer solution. ◁

The concept of RER inequalities is as follows. Suppose we select a partial solution, the subset of whose routes is part of a feasible integer solution. Since it is a partial solution, some routes for other vehicles (of which no routes are contained in the selected partial solution) could be part of the optimal solution as well. We can, however, put restrictions on the use of those routes for other vehicles in combination with a partial solution, depending on ϕ_S.

To model these restrictions neatly, we introduce $\gamma_\ell^{\phi_S}$ as the minimum use of servicemen over the sets V that results in ϕ_S, see (2.26). Consider the following three special cases for the value of $\gamma_\ell^{\phi_S}$. First, for $|S| = 2$ and $\phi_S = 1$, $\gamma_\ell^{\phi_S}$ equals the minimum use of servicemen type ℓ among the routes from S. Second, for $|S| = 2$ and $\phi_S = 2$ it results in the sum of servicemen types ℓ among the routes from S. Finally, for $|S| = 3$ and $\phi_S = 2$, $\gamma_\ell^{\phi_S}$ equals the minimum sum of servicemen use among two routes of S, for each ℓ independently.

For any feasible integer solution we ensure that at most ϕ_S routes are selected from a partial solution $S \in \mathbb{S}_t$. Let the complement route sets of S be given by $R_S^C = \{r \in R \mid \forall s \in S,\ t(s) = t(r),\ k(r) \in k^C(S)\}$, where $k^C(S) = \{C \subset \mathcal{K} \mid k' \neq k(s), \forall s \in S, k' \in C\}$. It reflects all routes belonging to vehicles for which no routes are contained in the partial solution S but are in the same period as the routes of S. Then we can formally define a resource-exceeding route as:

Definition 2.2 (Resource-exceeding Route)**.** A route $r \in \mathcal{R}_{kt}$ is resource-exceeding with respect to some partial solution $S \in \mathbb{S}_t$ if it belongs to the set E_S of resource-exceeding routes with respect to S. The set E_S is defined as:

$$E_S := \left\{ \tilde{r} \in \mathcal{R}_S^C \mid \exists\ \ell \in \mathcal{L}\ : \gamma_\ell^{\tilde{r}} + \gamma_\ell^{\phi_S} > \tilde{Q}_\ell \right\} \tag{2.27}$$

Hence a route is resource-exceeding with respect to a partial solution S if it cannot

be added to the partial solution without changing the value of ϕ_S.

To model RER inequalities, let $\Upsilon_{S,r} = 1$ if $r \in E_S$, and 0 otherwise. In addition, let $\Gamma_{S,r} = 1$ if there exists an $s \in S$ such that $k(s) = k(r)$ and for all $\ell \in \mathcal{L}$ it holds that $\gamma_l^s \geq \gamma_\ell^r$, and 0 otherwise. Then constraints (2.21) can be rewritten as the complete set of RER inequalities

$$\sum_{r \in \mathcal{R}_t} \Gamma_{S,r} y_r + \sum_{r \in \mathcal{R}_S^C} \Upsilon_{S,r} y_r \leq \phi_S \quad \forall\ C \in k^C(S),\ S \in \mathbb{S}_t,\ t \in \mathcal{T}. \tag{2.28}$$

We call S the constraint-generating subset of routes. We refer to RER inequalities of size s as the constraints (2.28) when $|S| = s$.

The model described by equations (2.18)-(2.20), (2.22) and (2.28) is a valid description of the MSPRP, and we refer to it as the Alternative Integer Programming Master Problem (AIPM). Its Linear Relaxation is referred to as ALPM. Since we will work with only a subset $\overline{\mathcal{R}} \subset \mathcal{R}$ in our pricing procedure, let $\overline{\mathbb{S}}_t$ denote $\mathbb{S}_t$ with respect to the restricted route set $\overline{\mathcal{R}}$. The resulting restricted integer and linear programming master problems are then denoted by RAIPM and RALPM, respectively.

An extensive comparison between the different models is given in Section 2.6. We will especially focus on the strength of the root node relaxations of models IPM and AIPM. In addition, we will investigate the effect of extending IPM with (2.28) and of extending AIPM with (2.21).

2.3.1.2 Properties of resource exceeding route inequalities

First, we show that AIPM has stronger LP relaxations then IPM, for $L = 1$. This is characterized by the following proposition.

Proposition 2.1 (Improved relaxation)**.** *Assume that $L = 1$. Let $c(ALPM)$ and $c(LPM)$ be the objective values of model ALPM and LPM, respectively. Then for any given $\overline{\mathcal{R}} \subseteq \mathcal{R}$, it holds that $c(ALPM) \geq c(LPM)$*

Proof. We show that the constraints generated by (2.21) are a subset of the constraints generated by (2.28). Without loss of generality, assume that $T = 1$, $y_r > 0$ for all $r \in \bar{\mathcal{R}}$ and that $\gamma_\ell^r \neq \gamma_\ell^{r'} (r \neq r')$ for any $r, r' \in \bar{\mathcal{R}}_k$. Consider an arbitrary constraint (2.21) that is violated. It implies that there is a minimum set of routes U such that $\sum_{u \in U} y_u \gamma_u > \tilde{Q}_\ell$; otherwise, constraint (2.21) cannot be violated. We show that there will always exist a RER inequality (2.28) that is violated if the following procedure is followed. For every vehicle K whose routes are included in U, let γ_ℓ^k be the minimum servicemen use of that vehicle. Let $U' \subset U$ be the collection of routes with that minimum resource usage among the K vehicles. Then, for at least a single

subset $U'' \subset U'$, an RER inequality (2.28) with generating subset of routes U'' will be violated. This follows directly from the definition of $\Upsilon(\cdot)$ and $\Gamma(\cdot)$. Hence the constraints generated by (2.21) are a subset of the constraints generated by (2.28), and therefore the objective value of ALPM is at least the objective value of LPM, that is, $c(\text{ALPM}) \geq c(\text{LPM})$. □

Hence, for a single resource (i.e., servicemen) type, model ALPM provides us with tighter LP relaxations. However, the differences between AIPM and IPM become less clear when multiple resource types are considered. The following propositions, therefore, provide insights into the differences between the models for practical scenarios.

Proposition 2.2. *Let $t \in \mathcal{T}$ be arbitrarily given and assume that $L = 1$. Let $S \subset \mathbb{S}_t$ be an arbitrary partial solution such that $\sum_{s \in S} \gamma_\ell^s > Q_\ell$. Then for any solution y with $\sum_{r \in S} y_r > \phi_S$ and $\sum_{r \in R_t} y_r \gamma_\ell^r \leq Q$ it follows that $y \notin ALPM$ while $y \in LPM$.*

Proposition 2.3. *Let $t \in \mathcal{T}$ be arbitrarily given. Suppose that for all $r \in \mathcal{R}$ it holds that $\gamma_\ell^r = Q_\ell$ for all $\ell \in \mathcal{L}$. Then $y \in (LPM)$ implies that $y \in (ALPM)$ and therefore $Conv(ALPM) = Conv(LPM)$.*

We briefly sketch the proofs of Propositions 2 and 3, as they follow trivially by the definitions of $\Upsilon(\cdot)$ and $\Gamma(\cdot)$. For Proposition 2, notice that $\sum_{r \in R_t} y_r \gamma_\ell^r \leq Q$ implies that constraints (2.21) are satisfied, while constraints (2.28) are not satisfied since $\sum_{s \in S} \gamma_\ell^s > Q_\ell$. For Proposition 3, all servicemen are transported by a single vehicle, and therefore constraints (2.21) and (2.28) coincide.

Propositions 2.2 and 2.3 tell us that AIPM is especially beneficial compared with IPM if the number of servicemen is restrictive but not fully utilized by single vehicles, that is, servicemen are maximally deployed but among different vehicles. In addition, both propositions made clear that for many (practical) route sets $\bar{\mathcal{R}}$, it may hold that $c(\text{ALPM}) > c(\text{LPM})$. This especially holds for the MSPRP, where a shared resource (servicemen) is completely utilized among multiple vehicles.

2.3.1.3 Pricing of resource exceeding route inequalities

Difficulties arise during the pricing step of the branch-and-price-and-cut algorithm. Normally, the added columns are found by solving the pricing problem, and we continue doing that until no new columns are found. However, when a new column is added to ALPM, new constraints must be included as well (constraints (2.28)). These constraints are, however, not yet known when the pricing problem is solved and columns might, therefore, be priced incorrectly. For general linear programs, this could lead to incorrectly priced columns and non-terminating pricing procedures. In

the next subsection, we will describe an alternative pricing strategy that will provide optimal solutions to ALPM and is guaranteed to terminate.

Let d_{ij}^{kt}, r_{ij} and λ_{kt} be as defined in Section 2.2.2 and let ψ_S be the dual costs corresponding to constraints (2.28). For readability, let $\mathbb{S} = \cup_{t\in\mathcal{T}}\mathbb{S}_t$ and let $\overline{\mathbb{S}}$ be the set $\mathbb{S}$ with respect to the restricted route set $\overline{\mathcal{R}}$. The (reduced) servicemen costs in the (k,t)-pricing problem corresponding to some route r equals

$$\bar{d}_r = \sum_{\ell\in\mathcal{L}} \tilde{c}_\ell \gamma_\ell^r - \sum_{S'\in A(S), S\in\mathbb{S}_t} (\Gamma_{S,r} + \Upsilon_{S,r})\psi_{S'} - \sum_{S'\in B(S), S\in\mathbb{S}_t} (\Gamma_{S,r} + \Upsilon_{S,r})\psi_{S'}, \quad (2.29)$$

where $A(S)$ is the set of constraints (2.28) generated by route set S already existing when route r is being generated, and $B(S)$ is the set of constraints (2.28) being generated by all generating subsets S that include r. That is, the constraints in $A(S)$ can be automatically priced , since r can only enter as a resource exceeding route in those inequalities, whereas the constraints in $B(S)$ do not exist at the moment of pricing, since they are generated due to generating r.

The first term of equation (2.29) consists of the primal service-person costs for the generated route r, the second term consists of the dual costs corresponding to constraints (2.28) where route r enters, and the third term consists of the dual costs corresponding to constraints (2.28) that are not yet in RALPM at the moment of generating r. The completely specified (k,t)-pricing problem then equals

$$\min_{r\in\overline{\mathcal{R}}_{kt}} \hat{c}_r := \sum_{(i,j)\in A} d_{ij}^{kt} r_{ij} + \bar{d}_r - \lambda_{kt}, \quad (2.30)$$

In (2.29), the dual values corresponding to the constraints generated by including the new columns are unknown, and therefore (2.29) cannot be determined at the moment of generating new routes. In the remaining part of this section, we will develop an alternative pricing strategy that provides us with an optimal solution to ALPM and is guaranteed to terminate as well.

For readability, let $\Delta(r) = \sum_{S'\in B(S), S\in\overline{\mathbb{S}}_t} (\Gamma_{S,r} + \Upsilon_{S,r})\psi_{S'}$, and let $\hat{\bar{d}}_r := \bar{d}_r + \Delta(r)$. Since $\psi_{S'} \leq 0$ for all S', it follows that $\Delta(r) \leq 0$ and subsequently that $\hat{\bar{d}}_r \leq \bar{d}_r$ for all $r \in \mathcal{R}$. By replacing $\bar{d}_r$ with $\hat{\bar{d}}_r$ in (2.29), we arrive at the alternative (k,t)-th pricing problem:

$$\min_{r\in\overline{\mathcal{R}}_{kt}} \hat{\hat{c}}_r := \sum_{(i,j)\in A} d_{ij}^{kt} r_{ij} + \hat{\bar{d}}_r - \lambda_{kt}. \quad (2.31)$$

All dual values corresponding to RER inequalities are nonnegative and, consequently,

Algorithm 2.1: Column and Row Generation procedure (CRG)

```
while true do
    LP ← SolveRLPM();
    for k ∈ 𝒦 do
        for t ∈ 𝒯 do
            Set of columns 𝒮 ← SolveAlternativePricing(k, t, LP);
            if S = empty then
                goTo line 2;
            end
        end
    end
    break;
end
```

$\hat{\hat{c}}_r \leq \hat{c}_r$. Hence the alternative pricing problem will correctly identify routes that cause dual infeasibility after being included in the restricted route set, which is needed for a correct column generation approach. The following proposition summarizes the above reasoning.

Proposition 2.4. *Let $k \in \mathcal{K}$ and $t \in \mathcal{T}$ be arbitrarily given. Consider the (k,t)-pricing problem* (2.29) *and the alternative (k,t)-pricing problem* (2.31)*. Consider an arbitrarily dual solution and corresponding to this dual solution, let $\hat{R} := \{r \in \mathcal{R}_{kt} \mid \hat{c}_r < 0\}$ and $\hat{\hat{R}} := \{r \in \mathcal{R}_{kt} \mid \hat{\hat{c}}_r < 0\}$. Then $\hat{R} \subseteq \hat{\hat{R}}$.*

From this proposition, we conclude that using the alternative pricing problem will not result in any suboptimal solution of ALPM. What remains to be shown is that using (2.31) instead of (2.29) will result in a column generation procedure that terminates.

The complete column and row generation procedure (CRG) is outlined in Algorithm 2.1. It consists of iteratively solving the alternative (k,t)-th pricing problem and solving RALPM. If no negative reduced cost routes are found in some (k,t)-th pricing problem, we continue searching for negative reduced cost routes in the next (k,t)-th pricing problem. If negative reduced cost routes are found by some (k,t)-pricing problem, we add those to RALPM and generate constraints (2.28). We then solve RALPM and restart the procedure. The procedure terminates if there is not a single route of negative reduced cost for any (k,t)-pricing problem.

Proposition 2.5. *Let $\hat{c}_r$ and $\hat{\hat{c}}_r$ be the reduced costs according to* (2.29) *and* (2.31)*, respectively. Let $\hat{R} := \{r \in \mathcal{R}_{kt} \mid \hat{c}_r < 0\}$ and $\hat{\hat{R}} := \{r \in \mathcal{R}_{kt} \mid \hat{\hat{c}}_r < 0\}$. Then $\hat{\hat{R}} \backslash \hat{R} = \emptyset$.*

Proof. We show that there is no route r such that $\hat{c}_r > 0$ while $\hat{\hat{c}}_r$ is negative. Let $k \in \mathcal{K}$ and $t \in \mathcal{T}$ be arbitrarily given. There are two cases that we need to consider.

1. Consider the possibility that we generate a route r that is already included in $\overline{\mathcal{R}}_{kt}$. Now assume $\hat{\hat{c}}_r < 0$. We know that there exists a route r' already included in $\overline{\mathcal{R}}_{kt}$ before generating r. That implies that $\hat{c}_{r'} < 0$, since $\Delta(r')$ is already known. However, $\hat{c}_{r'} > 0$ since it is contained in the restricted route set before obtaining the optimal solution to RALPM. Hence, such a route r cannot be generated.

2. Assume we generate route $r \in \mathcal{R}_{kt} \backslash \overline{\mathcal{R}}_{kt}$ with $\hat{\hat{c}}_r < 0$ and $\hat{c}_r > 0$. If $\hat{\hat{c}} < 0$, this implies that route r cuts off the current dual solution, if we ignore the constraints (2.28) generated by r. However, including those constraints results in a larger dual feasible region. It automatically follows that route r still cuts off the same dual solution in the enlarged dual space. Hence $\hat{c}_r < 0$ as well. As a result, we have shown that there are no such routes r.

By Proposition 2.5 and the above results, it follows that $\hat{\hat{R}} \backslash \hat{R} = \emptyset$. □

Theorem 2.1. *The procedure (CRG) terminates and solves ALPM to optimality.*

Proof. Follows directly from Propositions 4 and 5. □

2.3.2 Separating resource exceeding route inequalities

The concept of resource-exceeding route inequalities is explained in depth in Section 2.3.1. A novel method of adding the inequalities based on column-dependent rows has been discussed above. In order to compare its computational efficiency, we present a traditional separation procedure for adding the resource-exceeding route inequalities in this section.

A resource-exceeding route inequality is determined by a subset $S \in S_t$, a set $C \subseteq K^C(S)$ and the resource consumption level ϕ_ℓ^S. Instead of taking the perspective from individual routes, we now take the perspective of a resource consumption level $u^k = (u_1, u_2, \ldots, u_L)$ of vehicle k. A resource level u^k and a subset of vehicle indices $\overline{\mathcal{K}} \subset \mathcal{K}$, where $k \in \overline{\mathcal{K}}$, defines a single resource-exceeding route inequality in the following way:

Let resource consumption levels $u^k = (u_1, \ldots, u_L)$, $k \in \overline{\mathcal{K}}$ be given. Consider the set of dummy routes $S' = \{r_k\}_{k \in \overline{\mathcal{K}}}$ such that route r_k has resource consumption level u^k. Then, resource-exceeding inequalities are defined as inequalities (2.28) for $S = S'$.

The two approaches for adding resource exceeding-route inequalities have their benefits and drawbacks. The column-dependent constraints approach does not rely on a separation procedure; it adds resource exceeding route inequalities *while* generating the routes. This may become less efficient when K becomes large, since the number of added inequalities will then quickly increase. On the other hand, it remains very efficient when the number of servicemen type L becomes large. In addition, the column-dependent constraints approach does not rely on the exact defintion of resource-exceeding, that is, the approach remains valid even if other applications require nonlinear relations to determine whether or not routes are resource-exceeding. The separation approach becomes relatively inefficient for larger L due to the increasing number of combinations of resource levels. In addition, the separation approach will not result in a problem-defining set of constraints, whereas the column dependent constraints approach will.

2.3.3 Other valid inequalities

We will continue by discussing two well-known valid inequalities that are included in our algorithm for solving the MSPRP. For a set of nodes $S \subseteq N$, let $\delta^+(S) := \{(i,j) \in A \mid i \in S, j \notin S\}$.

2.3.3.1 2-path inequalities

Since the MSPRP inherits aspects from the one-to-one pickup and delivery problem, we have included so-called 2-path inequalities that have been shown to be effective in a set-covering formulation (Ropke, Cordeau, and Laporte 2007, Ropke and Cordeau 2009). They are formulated as follows:

Let $S \subseteq (N_d \cup N_p)$ be such that it cannot be visited by a single vehicle k in some period t. Then the following inequality is valid for the MSPRP,

$$\sum_{y_r \in \delta^+(\mathcal{S})} y_r \geq 2. \tag{2.32}$$

The 2-path inequalities are separated by means of a greedy heuristic and an exact labeling algorithm, as is discussed in Ropke and Cordeau (2009). Since a 2-path inequality is a single cut on the y_r variables, corresponding dual costs can be incorporated in the pricing problem by subtracting it from the arc costs that leave the set S.

2.3.3.2 Subset-row inequalities

We include subset-row inequalities, in the form of Chvátal-Gomory rank 1 cuts on Constraints (2.19). They are defined as follows:

For any $S \subseteq N_d$ and $k \in \mathbb{N}$ such that $0 < k \leq |S|$,

$$\sum_{r \in \overline{\mathcal{R}}} \Big\lfloor \frac{1}{k} \sum_{i \in S} \beta_i^r \Big\rfloor y_r \leq \Big\lfloor \frac{|S|}{k} \Big\rfloor. \tag{2.33}$$

The dual values corresponding to the subset-row inequalities (2.33) are incorporated in the pricing problem as discussed by Jepsen et al. (2008). We adopted their separation procedure as well. Initial experiments have shown that including subset-row inequalities for $k = 2$ and $|S| = 3$ is computationally efficient, whereas inequalities for other values of k and $|S|$ are not effective and are therefore not taken into account.

2.4 Pricing problems

The previous section defined (k, t)-pricing problems for both LPM and ALPM. Since the structure of each pricing problem is similar, we describe how to solve an arbitrarily given (k, t)-pricing problem. We first motivate the choice for developing a new variant of the pulse algorithm. Then, we discuss how to incorporate the dual values into the pricing problem, especially those corresponding to the RER inequalities (2.28). After this, we present the pulse algorithm in detail and prove its correctness.

2.4.1 The pricing problem

The pricing problem is a new variant of the Elementary Resource Constrained Shortest Path Problem (ERCSPP), introduced by Desrochers (1987) and discussed in Feillet et al. (2004) and Irnich and Desaulniers (2005), for instance. It can be classified as a Multi-Commodity Elementary Resource Constrained Shortest Path Problem with Delayed Paired Precedence Constraints, which has, to the best of the authors' knowledge, not been solved before. For readability, we refer to it by the term Pricing Problem (PP). Let us briefly summarize the problem characteristics included in the PP. The multi-commodity part refers to the different types of servicemen demanded at each service. Each service has a pickup and a delivery node, where the delivery node needs to be visited before the pickup node (precedence relation), the pickup node needs to be visited if the delivery node is visited (pairing relation), and the corresponding service time needs to be respected between the delivery and pickup

node ("delayed time" relation).

Traditionally, ERCSPPs are solved with labeling algorithms (Desrochers, Desrosiers, and Solomon 1992). These are dynamic programming approaches in which non-dominated partial paths are extended with nodes until the optimal solution is found. The extent to which non-dominated partial paths can be identified determines the labeling algorithm's computational efficiency. Dominance criteria are often based on the validity of the triangle inequality regarding the costs of traversing a path: It is more expensive to make a detour instead directly traversing an arc.

The nature of the MSPRP complicates the construction of efficient dominance criteria, since there are costs $\tilde{c}_\ell$ for using servicemen of type ℓ. This destroys the validity of the triangle inequality, i.e., it can be cheaper to make a detour instead of directly traversing an arc. The method proposed by Ropke and Cordeau (2009) to restore the, in their case, triangle inequality for the pickup nodes, can be used to partially restore the triangle inequality for the delivery nodes. In particular, travel costs (including possible penalty or maintenance costs) could be restructured as they depend solely on the arcs traversed, but the servicemen costs incurred could not.

A major consequence of this is that the dominance criteria developed by Ropke and Cordeau (2009) are not valid for solving our PP, and, henceforth, cannot be used for the PP. This leads to less label dominance and to a slower labeling algorithm. Therefore, we need to resort to another algorithm to solve PP efficiently. We develop a new variant of the pulse algorithm (Lozano, Duque, and Medaglia 2015) tailored for solving the PP without relying on dominance criteria. The pulse algorithm is shown to be competitive with the state-of-the-art labeling algorithm of Baldacci, Bartolini, and Mingozzi (2011) for solving the ERCSPP. The general overview of the pulse algorithm for solving the PP is given in Algorithms 2.3 and 2.4.

2.4.2 Incorporating the dual values

Recall that μ_i, λ_{kt}, and π_ℓ^t are the dual variables corresponding to constraints (2.19)-(2.21), and let their value be zero if one of the corresponding constraints is not included in the problem formulation. For example, π_ℓ^t equals zero in the PP of ALPM, since constraints (2.21) are replaced by constraints (2.28).

Let the initial costs of any partial path be $-\lambda_{kt}$. In order to incorporate the dual values corresponding to visiting nodes, we let the costs of traversing an arc $(i,j) \in A$ be d_{ij}^{kt} and let the costs for using a service-person of type ℓ be $\tilde{d}_\ell^t$, as defined in Section 2.2. Dual values corresponding to other cuts or valid inequalities (as long they correspond to visiting nodes or a set of nodes only) can be included in the travel

Algorithm 2.2: Construction of $\hat{C}$.

Data: Matrix $\hat{C}$, set of RER inequalities $\mathcal{S}$ with for each $S \in S$, corresponding $C \subset K^C(S)$, period t and vehicle k of the current (k,t)-th PP
for *RER inequality* $S \in \mathcal{S}$ *with corresponding set* $K^C(S)$ **do**
  for *route* $r \in S$ **do**
    if $k(r) = k$ **then**
      u = findResourceLevel(r);
      setMatrixSameVehicle($u, \tilde{Q}_1, \ldots, \tilde{Q}_L$)
    end
  end
  if $k \in k^C(S)$ **then**
    u = findResourceLevel(r);
    setMatrixOtherVehicle($u, \tilde{Q}_1, \ldots, \tilde{Q}_L$)
  end
end

costs by subtracting them from their corresponding arcs.

To incorporate the dual costs ψ of every RER inequality generated by all subsets $S \subset \mathcal{R}$, we define an $L-$dimensional array $\hat{C}$ of size $(\tilde{Q}_1+1) \times (\tilde{Q}_2+1) \times \cdots \times (\tilde{Q}_L+1)$. Entry $Q_{u_1 \ldots, u_L}$ contains the dual costs corresponding to the RER inequalities generated by subsets S where a generated route r using $u_1, \ldots, u_L$ servicemen will enter.

The algorithm to construct $\hat{C}$ is given in Algorithm 2.2. Here, the function 'findResourceLevel(r)' returns the servicemen use u of the route r in the inequality generating subset S. Then, if $k(r) = k$, we subtract the dual cost for every entry $Q_{\tilde{u}_1,\ldots,\tilde{u}_L}$ for which $\tilde{u}_\ell \geq u_\ell$ for all $\ell \in \mathcal{L}$. However, if $k \in k^C(S)$ we subtract the dual costs for every entry $Q_{\tilde{u}_1,\ldots,\tilde{u}_L}$ if there exists an $\ell \in \mathcal{L}$ such that $\tilde{u}_\ell + \gamma_\ell^{\phi_S} > Q_\ell$, that is, the servicemen-use is such that it is resource exceeding with respect to S. All dual values corresponding to RER inequalities are non-positive. This leads to the following observation:

Proposition 2.6. *Let* $u = (u_1, \ldots, u_L)$ *be an arbitrarily given resource level. Then for any* $u' = (u'_1, \ldots, u'_L)$*, such that* $u'_\ell \geq u_\ell$ *for all* $\ell \in \mathcal{L}$*, it holds that* $\hat{C}_{u_1,\ldots,u_L} \leq \hat{C}_{u'_1,\ldots,u'_L}$.

Proof. Let r and r' be routes consuming u and u' resources, respectively. Then the set of constraints where r enters is a subset of the set of constraints where r' enters. Since all dual values are negative, it follows that $\hat{C}_{u_1,\ldots,u_L} \leq \hat{C}_{u'_1,\ldots,u'_L}$. □

Algorithm 2.3: pulse main

```
Data: Time t, time step δ.
κ = createLowerBounds(t, δ);
z̄ = ∞;
Set of negative reduced cost routes S;
pulse(L, {0}, z̄);
return S;
```

Algorithm 2.4: pulse(L, n, $\bar{z}$)

```
Data: Label L, node n, best objective z̄
if feasible(L, n) then
    if checkBounds(L, n) then
        Label L' = extend(L, n);
        if d(L') + d̃(L') < z̄ then
            z̄ = d(L') + d̃(L');
        end
        if d(L') + d̃(L') < 0 then
            S = S ∪ {L'}
        end
        for n' ∈ N do
            pulse(L', n', z̄);
        end
    end
end
```

2.4.3 The pulse algorithm for PP

As can be seen from Algorithms 2.3 and 2.4, the pulse algorithm uses a depth-first search for exploring the solution space. Pruning of partial paths is performed by a lower bound criterion, initialized in the "createLowerBounds" function. Then the "pulse" procedure is called upon recursively, in which the "feasible" and "checkBounds" procedures prune partial paths based on feasibility and a lower bound criterion, respectively. All procedures will be explained in detail subsequently.

We store the information of a partial path as a label L, consisting of the following elements:

- The corresponding partial path $\vec{\eta}(L)$;
- The vector of departure times $\vec{t}(L)$ corresponding to $\vec{\eta}(L)$,
- The travel costs $d(L)$;

- The servicemen costs $\tilde{d}(L)$ including the dual costs from $\hat{C}(L)$;
- The number of servicemen $\zeta_\ell(L)$ of type ℓ currently working at delivery nodes;
- The maximum number of servicemen $\bar{\zeta}_\ell(L)$ of type ℓ used so far;
- The cumulative weight $\xi(L)$ of spare parts delivered so far;
- The set of pickup nodes $\mathcal{S}(L)$ whose delivery node is visited and the pickup node is still unvisited;
- And a set of delivery nodes $\mathcal{U}(L)$ that have already been visited.

The notation $d(L)$ and $\vec{\eta}(L)$ is used to denote the costs and the partial path of label L, respectively. This notation is used consistently for referring to the elements of label L. For a partial path of size s, the vectors $\vec{\eta}(L)$ and $\vec{\eta}(t)$ are of dimension $1 \times s$. With $t(L)$ and $\eta(L)$ we denote the last element of $\vec{t}(L)$ and $\vec{\eta}(L)$, respectively.

We introduce time windows $[a_i, b_i]$ for nodes $i \in N$. For $i \in N_d$, time windows are set as $[t_{0i}, \omega - s_i - t_{i,2n+1}]$, and for $i \in N_p$ they are set as $[t_{0i} + s_i, \omega - t_{i,2n+1}]$. In each call of the 'pulse' function, we extend a label L with some node $i \in N$, resulting in a new label L' that is constructed as follows.

$$\vec{\eta}(L') = (\vec{\eta}(L), i), \tag{2.34}$$

$$\zeta_\ell(L') = \zeta_\ell(L) + Q_{i\ell} \quad \forall \ell \in \mathcal{L}, \tag{2.35}$$

$$\xi(L') = \xi(L) + Q_i, \tag{2.36}$$

$$\vec{t}(L') = (\vec{t}(L), \max\{t(L) + t^{kt}_{\eta(L),i},\ a_i\}), \tag{2.37}$$

$$\bar{\zeta}_\ell(L') = \max\{\zeta_\ell(L) + Q_{i\ell}, \bar{\zeta}_\ell(L)\} \quad \forall \ell \in \mathcal{L}, \tag{2.38}$$

$$\tilde{d}(L') = \sum_{\ell \in \mathcal{L}} \bar{\zeta}_\ell(L') \cdot \tilde{d}^t_\ell - \hat{C}_{\bar{\zeta}(L)} + \hat{C}_{\bar{\zeta}(L')}, \tag{2.39}$$

$$d(L') = d(L) + d^{kt}_{\eta(L),i}, \tag{2.40}$$

$$\mathcal{S}(L') = \begin{cases} \mathcal{S}(L) \cup \{n+i\} & \text{if } i \in N_d \\ \mathcal{S}(L) \backslash \{i\} & \text{if } i \in N_p \end{cases}, \tag{2.41}$$

$$\mathcal{U}(L') = \begin{cases} \mathcal{U}(L) \cup \{i\} & \text{if } i \in N_d \\ \mathcal{U}(L) & \text{if } i \in N_p \end{cases}. \tag{2.42}$$

A label is pruned if it appears to be infeasible or if it cannot improve the current best solution. Both pruning criteria will be discussed next.

2.4.3.1 Feasibility

In the "feasible" procedure, we consider a label L that we extend with some node i. Similarly as in Ropke and Cordeau (2009), we only need to consider nodes i that satisfy

$$i \notin U \text{ if } i \in N_d, \quad i \in S \text{ if } i \in N_p, \quad \mathcal{S} = \emptyset \text{ if } i = 2n+1. \tag{2.43}$$

These indicate that a delivery node can be visited once at most; a pickup node can only be visited if the corresponding delivery node is visited, and visiting the destination depot is only feasible if there are no pickup nodes unvisited whose corresponding delivery node is visited.

If a node i satisfies the precedence and pairing relationships (2.43), we need to check the feasibility of the resource constraints:

$$\begin{aligned} &\xi(L) + Q_i \leq \bar{Q}^2, \quad \sum_{\ell \in L} \max\{\bar{\zeta}(L), \zeta_\ell(L) + Q_{il}\} \leq \bar{Q}^1, \\ &\zeta_\ell(L) + Q_{il} \leq \tilde{Q}_\ell \ \forall \ell \in \mathcal{L}, \quad t(L) + t_{\eta(L),i} \leq b_i. \end{aligned} \tag{2.44}$$

These model the spare part capacity as well as the servicemen capacity restrictions of the vehicle, the restricted availability of servicemen, and feasibility with respect to the time windows introduced. To conclude, the "feasible" procedure returns false if one of the conditions (2.43) or (2.44) is violated; otherwise true is returned.

2.4.3.2 Lower Bounds

The "createLowerBounds" procedure constructs a series of lower bounds κ_i^t for visiting node i at discrete time steps $t \in [\underline{t},\ \underline{t}+\delta,\ \underline{t}+2\delta+, \ldots, \omega-\delta]$ for all $i \in N_d$, where δ is a parameter determining the discrete time steps. The κ_i^t can be interpreted as a lower bound on the maximum possible gain for a path starting at i at time t. Let $\bar{z}$ denote the best solution to PP so far, that is, a valid upper bound on the optimal solution of the PP.

Consider a feasible label L with i being the last-added node to $\vec{\eta}$, and let t_i be departure time of i. The "checkBounds" procedure prunes a Label L if $d(L) + \tilde{d}(L) + \kappa_i^{\tilde{t}} > \bar{z}$, where $\tilde{t} \in [\underline{t},\ \underline{t}+\delta,\ \underline{t}+2\delta+, \ldots, \omega-\delta]$ and $\tilde{t} \leq t_i$. We choose $\tilde{t}$ as large as possible, since this will give the tightest lower bound at time t_i.

The lower bounds are calculated before solving the PP. They are based on the validity of the *pickup-triangle inequality* with respect to the incurred *travel* costs d, that is, the travel costs cannot decrease due to visiting an additional pickup location.

Initially, this property does not hold, since dual costs corresponding to valid inequalities and branching decisions may be included in the travel cost.

The following procedure restores the pickup-triangle inequality with respect to the travel costs: Assume that the dual costs of the pickup-triangle inequality breaking constraints are already incorporated in $\bar{d}_{ij}$. We search for the largest violation v_j of the pickup-triangle inequality, that is, $v_j := \max_{i,k \in N}\{\bar{d}_{ik} - (\bar{d}_{i,j+n} + \bar{d}_{j+n,k})\}$ for all $j \in N_d$. For all $j \in N_d$, we subtract v_j from $\bar{d}_{ij}$ and add v_j to $\bar{d}_{i,j+n}$ for all $i \in N$. For a proof of the correctness of this, we refer to Ropke and Cordeau (2009).

The following proposition describes how a valid lower bound is obtained.

Proposition 2.7 (Lower Bounds)**.** *A valid lower bound κ_i^t for all $i \in N_d$, $t \in [\underline{t},\ \underline{t}+\delta,\ \underline{t}+2\delta+, \ldots, \omega-\delta]$ is obtained by solving the pulse algorithm starting with label L, where L is defined as*

$$\vec{\eta}(L) = (i),\ d(L) = 0,\ \tilde{d}(L') = 0,\ \vec{t}(L) = (t),\ \xi(L) = Q_i$$

$$\zeta_\ell(L) = Q_{i\ell},\ \bar{\zeta}_\ell(L) = \tilde{Q}_\ell\ \forall \ell \in \mathcal{L}$$

$$\mathcal{S}(L) = \{n+i\},\ \mathcal{U}(L) = \{i\},$$

In addition, we assume that $\hat{C} = 0$ during the construction of κ_i^t.

Proof. Let $L^\star$ be the solution of the pulse algorithm if it starts with label L as defined above. Then the following two properties of $L^\star$ hold:

1. $\tilde{d}(L^\star) = 0$, since $\hat{C} = 0$ and $\bar{\zeta}_\ell(L^\star) = \tilde{Q}_\ell\ \forall \ell \in \mathcal{L}$. Hence $\bar{z} = d(L^\star)$ after running the pulse algorithm.

2. For any label $\tilde{L}$ starting at $\{0\}$ and ending at i, it holds that $\mathcal{S}(L) \subseteq \mathcal{S}(\tilde{L})$ and $\mathcal{U}(L) \subseteq \mathcal{S}(\tilde{L})$.

Then, let $\tilde{L}^\star$ be the solution of the pulse algorithm if we start with some label $\tilde{L}$ at time t, and let $d(\tilde{L}^\star) + \tilde{d}(\tilde{L}^\star)$ be its costs. By Property 2 and the pickup triangle inequality, it follows that $d(\tilde{L}^\star) \geq d(L^\star)$. By Property 1, and non-negativity of the servicemen costs, it follows that $d(L^\star) \geq d(\tilde{D}^\star)$.

Since $\kappa_i^t = \bar{z} = d(L^\star)$ after running the pulse algorithm starting with label L, it follows that $d(\tilde{L}) + \tilde{d}(\tilde{L}) + \kappa_i^t \leq d(\tilde{L}^\star) + \tilde{d}(\tilde{L}^\star)$ for any label $\tilde{L}$ starting at $\{0\}$ arriving at node i at time $\tilde{t} \geq t$. Hence κ_i^t is a valid lower bound for any $i \in N_d, t \in [\underline{t},\ \underline{t}+\delta,\ \underline{t}+2\delta+, \ldots, \omega-\delta]$. □

2.5 Branch-and-price-and-cut algorithm

We will now continue by presenting the general outline of the branch-and-price-and-cut algorithm. Recall that IPM is defined by equations (2.18)-(2.22), whereas AIPM is defined by equations (2.18)-(2.20), (2.22), and (2.28). We will elaborate on a simple heuristic for the construction of an initial route set, and we will discuss the branching and node selection strategy used.

Algorithm 2.5: Randomized search procedure

Data: Set of services $\mathcal{S}$
randomSort($\mathcal{S}$);
$s, s', s'' =$ CheapestFeasibleInsertion($\mathcal{S}$);
$k = 1$;
while $k < k_{\max}$ **do**
  $s'' = s'$;
  $\mathcal{S}' =$ removeVessel(s'', i);
  $s'' =$ CheapestFeasibleInsertion($s'', \mathcal{S}'$);
  $\mathcal{S}' =$ removeJobs(s'', j);
  $s'' =$ CheapestFeasibleInsertion($s'', \mathcal{S}'$);
  $k = k + 1$;
  if *accept(*s''*)* **then**
    $s' = s''$
  end
  if $s'' < s$ **then**
    $s = s''$
  end
end

2.5.1 Initial solution

Starting with a high-quality set of initial solutions may help to prune nodes of the branch-and-bound tree at an earlier stage. To that extent, we have used a small randomized search strategy inspired by the Two-stage Adaptive Large Neighbourhood Search in Schrotenboer et al. (2018a). It consists of randomly sequencing all services and inserting them by Cheapest Feasible Insertion. After this, it consists of an iterative procedure of two main steps: first, removing all services from a randomly selected number of routes and reinserting them in a random order with cheapest feasible insertion, and, second, removing some random services from the solution and reinserting them in a random order by cheapest feasible insertion. The new solution is accepted based on some simple simulated annealing criteria, see Schrotenboer et al. (2018a) for details. The general outline of this procedure is given in Algorithm 2.5.

Table 2.1: Benchmark characteristics.

Bm.	ω_t	$Q_{i\ell}$	$\tilde{Q}_\ell$	$\tilde{Q}_i$	s_i	p_i	$\hat{p}$	Vehicle	$\bar{Q}^1$	$\bar{Q}^2$	speed	cost
A	$[6, 10]$	$[1, 3]$	6	$[400, 800]$	$[2, 5]$	$[50, 1000]$	0.25	1	9	2000	60	20
								2	12	3000	35	32
B	$[6, 12]$	$[0, 4]$	7	$[400, 800]$	$[3, 6]$	$[50, 1000]$	0	1	9	2000	8	80
								2	12	3000	35	128
C	$[6, 10]$	$[1, 3]$	6	$[400, 800]$	$[2, 5]$	$[50, 1000]$	0.25	1	8	2000	60	30
								2	8	2000	50	25
								3	10	2000	50	20

2.5.2 Branching and node selection strategy

Branching rules and node selection rules should be selected with care in column-generation applications. We apply branching inspired by the approach in Naddef and Rinaldi (2001). Before explaining the exact branching procedure, notice that we are branching on edges in the original formulation. We need to define the values of x_{ij}^{kt} for every $(i, j) \in A, k \in \mathcal{K}, t \in \mathcal{T}$. These can easily be obtained from the values of the LP solution $\bar{y}_r, r \in \overline{\mathcal{R}}$, that is, $x_{ij}^{kt} = \sum_{r \in \overline{\mathcal{R}}_{kt}} r_{ij} \bar{y}_r$, where r_{ij} equals 1 if edge (i, j) is visited in route r.

The branching rule is simple, but effective. We search for a set $S \subseteq (N_d \cup N_p)$, for every period t and vehicle k, such that $x^{kt}(\delta^+(S))$ is as fractional as possible. Here, $x^{kt}(\delta^+(S)) := \sum_{(i,j) \in \delta^+(S)} x_{ij}^{kt}$. A simple greedy procedure is used to determine suitable candidates S for every combination of k and t. Preliminary experiments have shown that often a set of two nodes is found for which it holds that $x^{kt}(\delta^+(S))$ - $\lfloor x^{kt}(\delta^+(S)) \rfloor = 0.5$.

When a suitable candidate set S, a period index t and a vehicle index k, are found, branching is performed by imposing the constraint $\sum_{r \in \bar{\mathcal{R}}_{kt}} \sum_{(i,j) \in \delta^+(S)} r_{ij} y_r \leq \lfloor x^{kt}(\delta^+(S)) \rfloor$ on one child node, and $\sum_{r \in \bar{\mathcal{R}}_{kt}} \sum_{(i,j) \in \delta^+(S)} r_{ij} y_r \geq \lceil x^{kt}(\delta^+(S)) \rceil$ on the other child node. The branching constraints impose single cuts on (A)IPM, whose corresponding dual values should be considered in the pricing problem. This is done by incorporating the dual values into the costs of traversing arcs $\delta^+(S)$ in the (k, t)-pricing problem. The node selection strategy, i.e., which nodes of the branch and bound tree to explore first, is based on a best first search strategy. Initial experiments have shown that this works well.

2.6 Computational experiments

The goal of this section is twofold. First, we will provide insights into the efficiency of RER inequalities for different sizes of the corresponding generating subsets. Recall that that the maximum size of the RER inequalities equals the number of vehicles minus one. Second, we will show that the column dependent approach (Section 3.1) is, besides being theoretically interesting, competitive with a traditional separation method (Section 3.2). Recall that model formulation AIPM inherently uses the column-dependent constraints approach, since the knapsack-type constraints (2.21) are replaced with RER inequalities (2.28). On the other hand, we are making use of the separation procedure when we include RER inequalities into model IPM. In particular, we refer to IPM if we use the traditional set-covering formulation without RER inequalities. If we include the RER inequalities in IPM, we will refer to it as IPM + RERx, where x is the size of the included RER inequalities.

All experiments are conducted on three newly created benchmark sets of practically inspired instances, described below. We implemented the branch-and-price-and-cut algorithm with the framework for constraint programming SCIP 3.2.1 (Gamrath et al. 2016) in combination with CPLEX 12.6.3 as an LP-solver. The overall program is coded in C++. All experiments are performed on a Xeon E5 2680v3 CPU (2.5 GHz) processor with 16 GB of RAM. The implementation is completely single-threaded. The maximum calculation time is set to 10800 seconds or the time that 16GB of RAM is used, whichever comes first.

The benchmark sets are constructed based on a practical setting of offshore wind maintenance service logistics off the coast of the Netherlands. The instance characteristics are in line with recent work in offshore wind maintenance service logistics (Dai, Stålhane, and Utne 2015, Irawan et al. 2017) and follow from interviews with stakeholders in maintenance service logistics for offshore wind farms in the Netherlands. Benchmarking our branch-and-price-and-cut algorithm with existing approaches and benchmarks in this area is either not possible due to the inclusion of multiple depots and of multiple distinct wind farms that led to restrictions in the planning of the maintenance services (Irawan et al. 2017), or the benchmark instances are too small (solved within a second) to provide additional insights (Dai, Stålhane, and Utne 2015). We would like to stress that the approach of Irawan et al. (2017), in which the complete solution space is enumerated, has its limits at 8 services per wind farm. The instance sets that we propose, and are able to solve, contain up to 45 services for a single wind farm.

The lower bound procedure incorporated in the pulse algorithm for solving the

pricing problem is run with $\delta = 1$ and $\underline{t} = \lfloor 0.4\omega_t \rfloor$ for all $t \in \mathcal{T}$. The separation procedure for RER inequalities (model IPM) is a simple enumeration based on the current route set. It is run once every five branch and bound nodes, but only if the current branch and bound node provides the smallest lower bound on the optimal solution. The subset-row and 2-path inequalities are only included in the root node. This appears computationally to be most efficient.

2.6.1 Benchmark characteristics

We created three benchmark sets (A, B and C) that each entail different problem characteristics. A detailed description of the parameters is given in Table 2.1. Benchmark Set A resembles a practical situation with two vessels, a relatively fast and cheap but small vessel and a larger but slower and more expensive vessel. It consists of relatively short time horizons ($T \leq 10$). Benchmark Set B shares the same vessel characteristics as Benchmark A, but travel costs are increased. In addition, time horizons are larger ($T > 10$) and the number of services of the instances is larger. Finally, Benchmark Set C resembles a practical situation with three relatively small vessels with small differences regarding their capacity, travel costs, and travel speed. We would like to stress that the benchmarks are created such that vehicles will not be deployed in every period, since weather conditions are such that it is more profitable to cluster services in particular periods, instead of visiting the wind farm with each vehicle on a daily basis.

For each benchmark, the coordinates of the depot are fixed at $(0, 30)$, and the maintenance jobs are drawn in a box with lower left corner $(20, 20)$ and upper right corner $(40, 40)$. We consider three different types of servicemen, resembling servicemen with mechanical, electrical and electromechanical specialties. Their costs $\hat{c}_\ell$ equal $300, 325$ and 375, respectively. The demand for service-person type ℓ, as well as the weight of the spare parts needed for each service, is independent of other services' demands and is (uniformly) randomly drawn as specified in Table 2.1. This table also specifies the service times. In order to make a distinction between services, we consider a constant, per period, penalty p_i for not performing service i. The probability of a job having $p_i = 0$ equals $\hat{p}$. This resembles preventive maintenance tasks, while higher values of p_i resemble the relative urgency of performing a service and can be interpreted as corrective maintenance tasks.

The maximum driving time (in hours) of the vehicles is uniformly drawn as denoted by ω_t in Table 2.1. This resembles practical situations at offshore wind farms where daily working hours are limited due to weather conditions (e.g., wind speeds or fog). In

Table 2.2: Optimality gaps (%) of the root node relaxations for the inclusion of different valid inequalities.

		AIPM					IPM					dif.	
Bm.	n	none	2p	ss	full	t_{root}	none	2p	ss	full	t_{root}	full	t_{root}
A	10	1.14	1.05	0.73	0.73	0.12	2.46	2.34	2.16	2.16	0.17	65.96	1.35
	15	0.76	0.76	0.70	0.70	0.74	2.69	2.69	2.61	2.60	0.53	73.08	0.72
	20	1.13	1.13	1.02	1.02	4.09	2.65	2.62	2.42	2.42	6.95	57.41	1.70
	23	0.79	0.79	0.66	0.64	5.59	2.43	2.43	2.29	2.29	4.97	72.24	0.89
	26	1.33	1.33	1.28	1.27	13.36	3.27	3.27	3.23	3.23	8.14	60.66	0.61
	29	1.19	1.19	1.18	1.16	11.95	2.63	2.63	2.58	2.57	21.46	54.89	1.80
	32	1.16	1.16	1.14	1.12	30.56	2.96	2.94	2.95	2.93	48.75	61.70	1.59
	35	0.98	0.98	0.95	0.96	53.46	2.74	2.74	2.72	2.72	59.42	64.57	1.11
	38	1.00	0.99	0.99	0.98	77.97	3.25	3.25	3.24	3.24	130.41	69.81	1.67
B	30	0.65	0.65	0.64	0.62	6.36	1.42	1.42	1.40	1.40	9.85	55.74	1.55
	40	0.73	0.72	0.70	0.70	24.41	1.60	1.58	1.59	1.55	31.52	54.41	1.29
	45	0.65	0.64	0.61	0.64	46.57	1.61	1.61	1.61	1.59	57.01	59.80	1.22
Avg.		0.96	0.95	0.88	0.88	22.93	2.48	2.46	2.40	2.39	31.60	63.20	1.38

addition, it specifies the vessels' capacity for servicemen $\bar{Q}^1$ and the vessels' maximum allowed spare parts weight $\bar{Q}^2$. The vessels' cost parameter specifies the costs per unit of Euclidean distance traveled, and their speed parameter specifies the units of distance that can be traveled in an hour. For Benchmark A, we let the second vehicle have maximum driving time at least as large as the first vehicle, while for Benchmark B and C we did not make that distinction between the vehicles. Finally, it takes 0.25 time to transfer between a vessel and an offshore location. We incorporated this by increasing the travel time of every arc by 0.25. The instances from each benchmark set are named after the number of services n and time periods T that they consist of, followed by an index, for example, instance An20T5-1 and An20T5-2 are, respectively, the first and second instance with 20 services and 5 time periods in Benchmark Set A. For every combination of the number of services n and the number of periods T, we generated 10 instances and included the first three feasible instances in the benchmarks. The total number of instances is 49 for Benchmark Set A, 18 for Benchmark Set B and 26 for Benchmark Set C.

We tried to solve the instances by directly plugging the compact model formulation (Section 2.1) into CPLEX without including RER inequalities. This led to far worse results than the results presented here; CPLEX was only able to solve the very small instances in reasonable computation times.

2.6.2 Root node relaxations

We study the effect of the RER inequalities of size 1, the subset-row inequalities, and the 2-path inequalities by comparing the optimality gap of the root node relaxations on the instances of Benchmarks A and B. Recall that RER inequalities are bounded in size by the number of vehicles minus one. The impact of RER inequalities of size 1 and 2 is shown by comparing the root node relaxations of Benchmark C for different model formulations. If one of the instances is not solved to optimality by any of the models, the best upper bound is used (see Tables 2.4 - 2.6) to compute the corresponding root node optimality gaps.

The results are given in Tables 2.2 and 2.3. The results are presented according to the number of services the corresponding instances consist of, that is, each row represents averages of the benchmark instances with the corresponding number of services. The results in Table 2.2 below "IPM" are obtained without including RER inequalities of size 1, whereas the results below "AIPM" include RER inequalities of size 1 as model AIPM is based on the column dependent constraints approach. The columns indicated by "none" provide the root node optimality gap in percentages without using additional valid inequalities. The root node optimality gaps are calculated as $(UB - LB_{\text{root}})/LB_{\text{root}} \times 100\%$, where LB_{root} is the lower bound after processing the root node and UB equals the best upper bound (see Tables 4-6). Columns "2p" and "ss" denote the root node optimality gaps with 2-path inequalities and subset-row inequalities, respectively. The root node optimality gap of the full model specification, including both subset-row inequalities and 2-path inequalities, is given in the column "full". Next to that, t_{root} denotes the root node computation time of the full model specification. Differences between IPM and AIPM are given in the columns under "dif.". Here, "full" denotes the percentage decrease of the optimality gap resulting from using AIPM instead of IPM (i.e., the difference between AIPM's and IPM's root node optimality gaps as a percentage of IPM's root optimality gap), and t_{root} indicates the relative speed increase (i.e., t_{root} of IPM divided by t_{root} of AIPM).

Replacing the knapsack type constraints (2.21) with RER inequalities of size 1, that is, using AIPM instead of IPM, results in an average decrease of 63.20% of the root node optimality gap. The time needed for computing the root node relaxations differs significantly among the instances, but on average the root node relaxations of AIPM take 22.93 seconds, whereas the IPM's root node relaxations are obtained in 31.60 seconds on average. The effect of the subset-row inequalities on both AIPM and IPM is noticeable, whereas 2-path inequalities seem less effective.

The results in Table 2.3 show root node optimality gaps for the instances of benchmark C. We compare model formulation IPM (without RER inequalities), IPM

Table 2.3: Optimality gaps (%) of the root node relaxations of benchmark C with different RER inequalities included.

		IPM		IPM + RER1		IPM + RER2		IPM + RER1+2	
n	T	Opt. gap	t_{root}	Opt. gap	t_{root}	Opt. gap	t_{root}	Opt. gap	t_{root}
10	2	1.04	0.08	1.04	0.07	0.61	0.13	0.61	0.13
10	3	0.60	0.06	0.60	0.07	0.57	0.08	0.57	0.08
14	3	1.68	0.21	1.64	0.21	1.40	0.26	1.27	0.27
14	4	1.05	0.23	1.05	0.25	0.94	0.30	0.94	0.29
18	3	1.23	0.63	1.23	0.60	0.63	0.75	0.63	0.75
18	4	1.28	0.45	1.28	0.44	0.84	0.73	0.84	0.69
22	4	1.50	0.95	1.50	0.93	1.22	1.37	1.22	1.35
22	5	1.23	0.96	1.22	0.96	1.04	1.47	1.03	1.55
26	5	2.23	1.25	2.16	1.29	1.53	2.01	1.50	1.54
avg.		1.31	0.54	1.30	0.54	0.98	0.79	0.96	0.74

with RER inequalities of size 1 ("IPM + RER1"), IPM with RER inequalities of size 2 ("IPM + RER2"), and IPM with RER inequalities of size 1 and 2 ("IPM + RER1+2"). All experiments include both 2-path inequalities and subset-row inequalities. Table 2 presents for each of the four formulations, the root node optimality gap (as calculated in Table 2) and the corresponding calculation time for processing the root node. The formulation "IPM + RER 1+2" provides the best root node relaxations. With an average root node optimality gap of 0.96%, it closes IPM's root node optimality gap with 26.71%. This is mainly due to the inclusion of RER inequalities of size 2, since they individually close IPM's optimality gap with 25.19%.

2.6.3 Full model comparison

The solutions to Benchmark sets A, B, and C are given in Tables 2.4 - 2.6, respectively. Each instance is solved with AIPM, which relies on the column dependent constraints approach, and with IPM + RER1 (Benchmark Set A and B) or IPM + RER1+2 (Benchmark Set C) in which the RER inequalities are separated in a traditional way. We included subset-row and 2-path inequalities in all the presented results. The columns "UB", and "gap" present the best upper bound found and the corresponding optimality gap in percentages, respectively. The optimality gap is calculated as $(UB - LB)/LB \times 100\%$, where LB is the best lower bound. Next, the column "nodes" denotes the number of explored nodes of the branch-and-bound tree, and the column "Sec." denotes the total runtime. Finally, the percentage difference in runtime is denoted in the column "Δ Sec.", calculated as [Sec.(AIPM) - Sec.(IPM + RER1)]/[Sec. (IPM + RER1)]$\times$ 100 %, and the root node optimality gaps are provide in the last column.

Table 2.4: Solutions to instances benchmark set A with IPM + RER1 and AIPM. RER inequalities of size 1 are included in IPM.

	IPM + RER1				AIPM					
Instance	UB	Gap	Nodes	Sec.	UB	Gap	Nodes	Sec.	Δ Sec.	Gap (root)
An10T3-1	28087.55	0.00	99	2	28087.55	0.00	11	0	-76.70	1.54
An10T3-2	20970.66	0.00	7	0	20970.66	0.00	3	0	-34.88	0.29
An10T3-3	26617.57	0.00	5	0	26617.57	0.00	5	0	-48.98	0.26
An10T4-1	27098.09	0.00	19	0	27098.09	0.00	11	0	-38.89	0.62
An10T4-2	28152.48	0.00	112	2	28152.48	0.00	27	1	-75.00	1.04
An10T4-3	23789.15	0.00	19	0	23789.15	0.00	9	0	-21.95	0.65
An15T4-1	43793.93	0.00	46	3	43793.93	0.00	15	3	-3.23	0.81
An15T4-2	43458.52	0.00	11	1	43458.52	0.00	5	1	32.95	0.43
An15T4-3	48748.17	0.00	182	8	48748.17	0.00	27	4	-47.49	2.23
An15T5-1	43025.34	0.00	41	3	43025.34	0.00	55	4	39.33	0.67
An15T5-2	47227.70	0.00	1	1	47227.70	0.00	1	0	-43.64	0.00
An15T5-3	51964.74	0.00	3	1	51964.74	0.00	3	0	-46.74	0.05
An20T5-1	63872.94	0.00	1824	145	63872.94	0.00	337	126	-13.30	2.07
An20T5-2	38780.62	0.00	1	6	38780.62	0.00	1	11	73.50	0.00
An20T5-3	49958.59	0.00	164	52	49958.59	0.00	51	23	-56.43	1.10
An20T6-1	69517.70	0.00	216	11	69517.70	0.00	68	5	-51.21	1.04
An20T6-2	65297.03	0.00	1	2	65297.03	0.00	1	1	-42.54	0.00
An20T6-3	64001.64	0.00	567	155	64001.64	0.00	201	188	21.19	1.97
An23T6-1	68871.16	0.00	301	92	68871.16	0.00	82	80	-13.60	0.84
An23T6-2	59077.44	0.00	147	43	59077.44	0.00	161	139	221.40	0.79
An23T6-3	62875.43	0.00	27	16	62875.43	0.00	11	15	-5.26	0.38
An23T7-1	68931.47	0.00	164	22	68931.47	0.00	123	32	44.86	0.73
An23T7-2	70931.07	0.00	508	94	70931.07	0.00	145	56	-40.67	0.78
An23T7-3	62269.57	0.00	12	12	62269.57	0.00	9	13	12.04	0.29
An26T7-1	84364.62	0.00	34	24	84364.62	0.00	3	41	73.01	0.12
An26T7-2	72722.87	0.00	7537	2506	72722.87	0.00	1548	1081	-56.85	1.64
An26T7-3	73208.39	0.00	1989	660	73208.39	0.00	864	780	18.04	1.95
An26T8-1	72019.28	0.00	787	128	72019.28	0.00	63	50	-60.88	0.47
An26T8-2	92919.51	0.00	10465	1799	92919.51	0.00	2951	1534	-14.75	1.97
An26T8-3	76139.09	0.00	1602	375	76139.09	0.00	528	401	6.79	1.48
An29T8-1	78543.35	0.00	4247	3157	78543.35	0.00	2167	4010	27.02	1.36
An29T8-2	96981.64	0.00	2488	438	96981.64	0.00	874	492	12.35	0.93
An29T8-3	114259.36	0.00	1591	466	114259.36	0.00	476	469	0.61	1.14
An29T9-1	78185.67	0.00	233	549	78185.67	0.00	117	394	-28.13	1.10
An29T9-2	93382.71	0.00	579	249	93382.71	0.00	141	290	16.28	1.20
An29T9-3	85954.50	0.00	2025	850	85954.50	0.00	953	778	-8.43	1.22
An32T9-1	101388.52	0.00	8447	3352	101388.52	0.00	1542	1078	-67.85	1.37
An32T9-2	111095.36	0.00	1669	2727	111095.36	0.00	1281	4518	65.69	1.38
An32T9-3	100752.91	0.00	798	289	100752.91	0.00	124	186	-35.74	0.95
An32T10-1	84453.77	0.00	3208	6130	84453.77	0.00	987	3554	-42.03	1.43
An32T10-2	123195.44	0.00	3613	3491	123195.44	0.00	1085	1273	-63.55	0.55
An32T10-3	89877.16	0.00	805	2773	89877.16	0.00	443	2607	-6.00	1.05
An35T10-1	119355.72	0.00	1082	1812	119355.72	0.00	442	1144	-36.85	0.60
An35T10-2	115930.11	0.00	1170	1603	115930.11	0.00	501	2193	36.82	0.92
An35T10-3	120197.78	0.00	1435	2757	120197.78	0.00	747	2880	4.45	1.37
An38T10-1	135434.78	0.00	2629	4893	135434.78	0.00	682	2632	-46.20	1.00
An38T10-2	128006.98	0.00	1395	1482	128006.98	0.00	554	2591	74.81	0.97
An38T10-3	135708.68	0.00	1650	9896	135708.68	0.00	760	8794	-11.13	0.96
An41T10-1	196225.73	1.43	6767	10800	196225.73	1.40	2241	10800	0.00	2.97
An41T10-2	133740.76	0.60	4116	10800	134029.73	0.82	967	10800	0.00	1.36
Avg.		0.04	1506.65	1465		0.04	478.51	1296	-5.75	1.00

As can be seen from Table 2.4 and Table 2.5, both methods of handling the RER inequalities provide competitive results. The optimality gaps of both methods are comparable and the computation time of AIPM is slightly lower than the computation time of IPM + RER1, that is, from 1464.80 to 1296.35 seconds for Benchmark A and from 4031.17 to 3132.50 seconds for Benchmark B. This difference is explained by the decrease of the searched branch and bound nodes in the column dependent constraints approach. This is probably caused by a better structured solution space as RER inequalities are generated earlier, which then may increase the efficiency of branching. Furthermore, both methods can solve instances of up to 45 services over a time horizon of 21 days to optimality, showing the strength of the branch-and-price-and-cut algorithm developed. Out of the 117 instances, only five instances remain unsolved within the time limit. Nevertheless, resulting optimality gaps for the 5 unsolved instances are small, that is, on average 0.676 % for AIPM. The root node optimality gaps are on average 1.00% and 0.68% for the instances of Benchmark Sets A and B, respectively. It can be observed that instances that are more difficult to solve suffer indeed from a larger root node optimality gap. However, all resulting gaps are relatively small which can be contributed to the effectiveness of the RER inequalities.

A typical optimal solution of the MSPRP consists of a mixture of routes in which jobs are scheduled simultaneously, sequentially, or a combination of both (see, e.g., Table 7). This is caused by several factors driving the structure of the optimal solutions: 1) minimizing travel costs, 2) minimizing servicemen costs, 3) capacitated vehicles in spare parts and servicemen, and 4) for each job and period exogenous costs are present reflecting the services' relative urgency. This results in a small fraction of the routes being devoted to a single service, i.e., the number of such scheduled services is 8% and 11% for benchmark sets A and B, respectively. The remaining routes' lengths, in the number of visited nodes, equals 4.8 on average over all instances, and routes consisting of 10 nodes are typically part of the optimal solution. This implies that in 40% of the periods a vehicle is not used due to being suboptimal. In addition, we like to stress that during the execution of the branch-and-price-and-cut algorithm routes of large sizes need to be generated, otherwise no optimality guarantee can be provided. These observations and statistics indicate that an enumerative approach, as is taken in Irawan et al. (2018), is indeed not able to solve the MSPRP to optimality.

Table 2.6 presents the result for solving Benchmark Set C. The average number of explored branch and bound nodes is almost four times smaller for AIPM (1004.58 vs. 252.92). Nevertheless, the number of constraints included in the column-dependent constraints approach causes nodes to be processed more slowly, and, consequently, the overall computation time becomes greater in some instances. Finally, all instances

could be solved to optimality with both AIPM and IPM + RER1+2.

2.6.4 Ignoring technician costs

Looking into the structure of the optimal solutions, it should be noted that routes of length up to 5 services are observed among the optimal solutions, which is in line with the results presented in Irawan et al. (2017). When technician costs are ignored, the driving factor that disperses services over the time horizon is removed. To assess the performance of the branch-and-price-and-cut algorithm under such circumstances we resolved the instances of Benchmark Set A by removing technician costs. The average optimality gap equals 0.19% on average with an average computation time of 1331 seconds. Comparing withe results in Table 2.4, it is observed that the performance is similar. Hence we can conclude that the branch-and-price-and-cut algorithm developed is able to solve the variant of the MSPRP in which servicemen costs are ignored.

2.6.5 Short service times

To test the performance of the branch-and-price-and-cut algorithm developed, we resolved Benchmark Set B with service times cut in half using AIPM. Although this may not reflect current practices in offshore wind, it might become a reality in a more mature offshore wind industry, in which short inspections and minor repairs might become reality (Willis et al. 2018). We increased the computation time allowed to 24 hours and the maximum memory usage to 64 GB. Average optimality gap after termination equalled 1.10 %, mainly due to instance 2n45T14 (6.8%) for which the memory limit was reached. We clearly observe that only a limited number of periods are being utilized by the vehicles. The average number of routes among the best upper bounds found equals 10.68, with a maximum of 13 routes and a minimum of 8 routes. In Table 7, we provide an example solution (Bn40T21-2) consisting of 11 routes, out of a total possibility of 42 routes (21 periods times 2 vehicles). Looking into the structure of the solution in Table 7, once could clearly observe the mixture of routes in which jobs are scheduled simultaneously (e.g., in Table 7, the route of Vehicle 1 in Period 1) and routes in which jobs are scheduled sequentially (e.g., in Table 7, the route of vehicle 0 in Period 2). This is caused by the several characteristics of the MSPRP influencing the structure of the optimal solution, as is discussed in Section 6.3.

Table 2.5: Solutions to instances benchmark set B with IPM + RER1 and AIPM.

	IPM + RER1				AIPM					
Instance	UB	Gap	Nodes	Sec.	UB	Gap	Nodes	Sec.	Δ Sec.	Gap (root)
Bn30T14-1	255600.40	0.00	3	7	255600.40	0.00	5	6	-13.63	1.00
Bn30T14-2	164668.12	0.00	84	50	164668.12	0.00	44	59	17.04	0.01
Bn30T14-3	229575.13	0.00	546	93	229575.13	0.00	261	80	-13.22	0.22
Bn30T21-1	223300.89	0.00	5613	2255	223300.89	0.00	849	432	-80.82	0.71
Bn30T21-2	198642.60	0.00	85	95	198642.60	0.00	57	44	-53.51	0.96
Bn30T21-3	187745.43	0.00	2282	1741	187745.43	0.00	738	924	-46.93	0.60
Bn40T14-1	274446.27	0.12	13736	10800	274446.27	0.08	6289	10800	0.00	1.22
Bn40T14-2	266032.66	0.00	396	498	266032.66	0.00	179	382	-23.33	1.27
Bn40T14-3	259001.17	0.00	2508	2999	259001.17	0.00	894	2019	-32.67	0.28
Bn40T21-1	293918.58	0.00	1894	1304	293918.58	0.00	1307	1186	-9.00	0.72
Bn40T21-2	316379.51	0.00	82	64	316379.51	0.00	11	31	-51.14	0.96
Bn40T21-3	265363.92	0.17	6281	10800	265363.92	0.00	3106	7727	-28.46	0.33
Bn45T14-1	316761.50	0.73	5003	10801	317716.30	0.80	2922	10800	-0.01	0.68
Bn45T14-2	367451.65	0.00	3004	1629	367451.65	0.00	3475	3733	129.15	1.27
Bn45T14-3	309185.47	0.00	1778	2160	309185.47	0.00	336	961	-55.52	0.57
Bn45T21-1	311858.05	0.00	4220	6466	311858.05	0.00	931	2164	-66.53	0.27
Bn45T21-2	326823.04	0.00	12704	7951	326823.04	0.00	4307	4235	-46.73	0.42
Bn45T21-3	313399.50	0.32	12342	10800	313638.42	0.28	6184	10800	0.00	0.67
Avg.		0.07	4031.17	3917.46		0.06	1771.94	3132.50	-20.85	0.68

Table 2.6: Solutions to instances benchmark set C with IPM and AIPM. RER inequalities of size 1 and 2 included in IPM.

	IPM + RER1+2				AIPM					
Instance	UB	Gap	Nodes	Sec.	UB	Gap	Nodes	Sec.	Δ Sec.	Gap (root)
Cn10T2-1	30463.24	0	23	0	30463.239	0	31	0	81.25	1.60
Cn10T2-2	25253.40	0	3	0	25253.396	0	1	0	-46.15	0.20
Cn10T2-3	29919.01	0	8	0	29919.01	0	1	0	-61.54	0.03
Cn10T3-1	27419.56	0	3	0	27419.559	0	1	0	-41.18	0.00
Cn10T3-2	27250.62	0	1	0	27250.615	0	1	0	-33.33	0.00
Cn10T3-3	29503.38	0	36	0	29503.382	0	20	0	31.25	1.72
Cn14T3-1	38699.91	0	61	1	38699.914	0	41	3	226.83	0.98
Cn14T3-2	41705.84	0	11	0	41705.835	0	3	0	88.89	0.05
Cn14T3-3	41317.37	0	414	2	41317.365	0	143	9	276.76	2.79
Cn14T4-1	40873.76	0	95	1	40873.758	0	26	1	-36.46	0.87
Cn14T4-2	36218.82	0	30	1	36218.821	0	50	5	590.91	1.35
Cn14T4-3	37699.02	0	3	0	37699.022	0	7	0	53.33	0.62
Cn18T3-1	51376.73	0	13	1	51376.725	0	1	2	77.45	0.25
Cn18T3-2	49947.32	0	87	4	49947.321	0	11	5	45.79	1.02
Cn18T4-1	58605.42	0	15	1	58605.42	0	23	1	106.15	0.71
Cn18T4-2	50437.17	0	163	8	50437.17	0	59	8	4.96	1.06
Cn18T4-3	57105.47	0	37	1	57105.47	0	30	2	26.61	0.74
Cn22T4-1	61700.09	0	35	4	61700.09	0	17	5	14.29	0.33
Cn22T4-2	56387.57	0	1489	26	56387.57	0	1964	6028	23428.81	1.91
Cn22T4-3	69951.25	0	6191	370	69951.25	0	879	1277	244.64	1.44
Cn22T5-1	69566.61	0	1233	84	69566.61	0	196	55	-34.43	1.38
Cn22T5-2	68890.12	0	654	15	68890.12	0	219	44	187.17	1.21
Cn22T5-3	75468.34	0	99	3	75468.34	0	101	8	148.84	0.48
Cn26T5-1	81605.927	0	209	11	81605.927	0	35	30	172.94	0.67
Cn26T5-2	80033.842	0	1018	48	80033.842	0	219	282	490.28	1.88
Cn26T5-3	86571.154	0	14188	512	86571.154	0	2497	4860	848.92	1.96
Avg.		0.00	1004.58	42.08		0.00	252.92	485.58	1034.35	0.97

Table 2.7: Solution of instance Bn40T21-2 with small service times.

Vehicle	period	route
0	0	0 25 16 56 65 28 68 81
0	2	0 8 48 35 75 11 51 81
0	4	0 10 50 26 66 27 67 81
0	6	0 12 52 23 63 29 69 81
0	7	0 4 44 5 45 32 72 13 53 81
0	13	0 40 80 34 74 39 79 81
1	0	0 31 71 18 33 73 58 81
1	1	0 24 14 54 64 1 20 60 17 41 2 42 57 81
1	2	0 6 15 55 46 9 49 21 61 81
1	4	0 7 47 38 37 77 78 22 30 62 70 81

2.7 Conclusions

In this paper, we introduced resource exceeding route inequalities, a specialized form of knapsack cover inequalities. These resource-exceeding route inequalities are applicable for any routing problem involving the consumption of a scarce set of resources. Two different approaches for including the resource-exceeding route inequalities were presented. In the first approach, we make use of column-dependent constraints to replace knapsack-type inequalities that model the resource scarcity. We showed that the convex hull of this new formulation is contained in the convex hull of a traditional set-covering formulation for particular cases, and we provided insights in instance characteristics that benefit the most from the new formulation. In order to use the new formulation in a column-generation approach, we formulated an alternative pricing procedure and proved that it provides optimal solutions. A traditional separation procedure for RER inequalities is also presented, in order to assess the computational performance of the column-dependent constraints approach.

The strength of the resource-exceeding route inequalities and the effectiveness of the column-dependent constraints approach has been tested on a new problem in the area of offshore wind maintenance service logistics: the Multi-period Service Planning and Routing Problem. This is the first problem in this area without predefined planning restrictions. A branch-and-price-and-cut algorithm, which is the first sophisticated exact solution approach in the area of offshore wind maintenance service logistics, has been developed to solve the Multi-period Service Planning and Routing Problem. A tailored pulse algorithm with a novel lower bounding procedure was developed to solve the pricing problems.

Computational experiments on a practically inspired set of benchmark instances showed the strength of RER inequalities. Optimality gaps of the root node relaxations

on instances with two vehicles were reduced, on average, by 63.20%, a scenario widely encountered throughout the literature. The column-dependent constraints approach appears very effective in searching the branch and bound tree, as compared to a standard separation procedure. Overall, both methods of including RER inequalities are competitive in terms of computational efficiency. Instances of up to 92 nodes and 21 time periods could be solved to optimality in reasonable computation times.

A promising direction for further research is the inclusion of stochastic elements in the Multi-Period Service Planning and Routing Problem. For example, one could model uncertain weather conditions and thereby uncertainty in travel times. Another promising direction is to include what are known as experience-based service times. By doing so, service times become more predictable when servicemen become more experienced in their work.

Chapter 3

Coordinating technician allocation and maintenance routing for offshore wind farms

Abstract. *A maintenance activity at offshore wind farms requires a combination of technicians with different skills. At an operational level, it is important to fully utilize and coordinate technicians in order to increase efficiency of the short-term maintenance planning. In this paper, we investigate sharing of technicians between wind farms over multiple periods, while determining per period vessel routes delivering and picking up technicians. The problem can be considered as a novel variant of the multi-period multi-commodity pick up and delivery problem. We develop an adaptive large neighborhood search heuristic which achieves high-quality, and often optimal, solutions on benchmark instances from the literature. The heuristic is used to explore the benefits of different sharing policies. By sharing technicians, both the flexibility of the daily planning is improved and the expected maintenance costs are reduced. In addition, the increased flexibility results in fewer vessel trips and increases the decision maker's ability to cope with extreme scenarios encountered in the short-term maintenance planning.*

This chapter is based on Schrotenboer et al. (2018a):
Schrotenboer AH, Uit het Broek MA, Jargalsaikhan B, Roodbergen KJ, 2018a *Coordinating technician allocation and maintenance routing for offshore wind farms. Computers & Operations Research* 98:185–197

3.1 Introduction

Resource sharing has the potential to improve the efficiency of the overall supply chain. To incorporate resource sharing into a firm's daily practice, supply chain integration is required on both a strategic and a tactical level of supply chain management. On an operational level, supply chain integration is likely to reduce the waste of valuable resources and to increase profitability of the business (Leuschner, Rogers, and Charvet 2013). Examples of such practice include inventory pooling (Berman, Krass, and Mahdi Tajbakhsh 2011, Zhong et al. 2017) and flexibly allocating workforce to multiple sites (Bhandari, Scheller-Wolf, and Harchol-Balter 2008). In this study, we analyse different policies to allocate the workforce in the context of maintenance routing and scheduling at offshore wind farms, and their consequences on an operational level. We, thereby, generalize existing routing and scheduling problems in this area (e.g., Dai, Stålhane, and Utne (2015), Stålhane, Hvattum, and Skaar (2015), Irawan et al. (2017)), and show that a flexible workforce policy is a natural way to improve the efficiency of the supply chain.

The total number of offshore wind farms has increased significantly in the last years, and will increase even more in the coming decades[1]. This results in a situation in which maintenance providers are becoming responsible for maintaining multiple wind farms. It is clear that operating each wind farm in isolation may result in an incomplete utilization of the available resources, which is the main challenge of the short-term maintenance planning at offshore wind farms. A typical feature of this problem is the scarcity of technicians with different specialties that restricts the flexibility, and thereby the efficiency, of the short-term maintenance planning. A maintenance activity requires, besides required spare parts, a combination of technicians with different skills. Therefore, a coordinated approach in which the available technicians are jointly utilized between wind farms may increase the overall efficiency of the planning, for example, by reducing idle time of technicians (Shafiee 2015, Irawan et al. 2017). Such insights on an operational level offer opportunities to increase sustainability of the offshore wind sector, and increases competitiveness with traditional energy suppliers.

Usually, technicians are assigned on a long-term basis to one of the Operating and Maintenance (O&M) bases, from which the daily operations are coordinated. From this O&M base, the technicians are deployed to perform maintenance activities at one or multiple wind farms. In order to improve efficiency, the maintenance provider may reassign technicians between O&M bases in the short-term. For example, technicians may be assigned completely flexible on a daily basis or for multiple days to a single O&M

[1] European Wind Energy Association. *Wind energy scenarios for 2030*, EWEA, 2015.

base depending on the geographical restrictions. Once the allocation of technicians is decided upon, the construction of a short-term maintenance planning itself presents several challenges. Generally, the time frame of a short-term planning is a week as the maintenance activities are heavily dependent on the weather conditions. Each maintenance activity requires an activity-specific set of technicians. Thus, technicians with a suitable skill set for an activity are chosen and transported to an offshore location to perform maintenance activities. After completion of the maintenance, the set of technicians may change to start other maintenance activities, i.e., technicians do not work in fixed teams throughout the period. Summarizing, a set of vessel routes needs to be determined for each period that describes which technicians are transferred to which offshore location, such that all maintenance activities are scheduled throughout the time horizon.

The allocation of technicians to different O&M bases classifies as tactical decision making. Embedding vehicle routing problems into optimization problems of a tactical or a strategic nature are practically relevant but challenging. Inherently, such problems are often intractable to solve exactly and, therefore, metaheuristics are often employed to provide insights on the problem. Ciancio et al. (2018) developed a heuristic solution approach for scheduling bus lines, and afterwards assigning crew to these lines. Technician allocation touches the field of workforce management, of which its influence in routing problems is studied by Smilowitz, Nowak, and Jiang (2013) as well. They study the efficiency of vehicle routing solutions in the presence of workforce management constraints, and show that properly including workforce management conditions into the optimization slightly increases overall costs. Many rich routing problems are found in health or home care applications. For example, Detti, Papalini, and de Lara (2017) have developed a hybrid heuristic combining variable neighborhood search and tabu search for a healthcare routing problem with many side constraints.

In this paper, we introduce the Technician Allocation and Routing Problem for offshore wind farms (TARP). Its goal is to jointly determine the daily allocation of differently skilled technicians to multiple O&M bases and the accompanying daily vessel routes used to perform the maintenance activities. Daily vessel routes should depart and return to their corresponding O&M bases every day, assuring the pickup and delivery of technicians and the transport of spare parts, to the offshore locations. By incorporating the technician allocation, the TARP is a generalization of the short-term maintenance planning and routing problem at offshore wind farms, see Dai, Stålhane, and Utne (2015), Stålhane, Hvattum, and Skaar (2015), Irawan et al. (2017). In the TARP, technicians are *flexibly* allocated on a daily basis; to analyze different practices at offshore wind farms, we introduce two variants of the TARP. The first

variant, called TARP-F, allocates technicians at the beginning of the time horizon and the allocation is *fixed* over time. The second variant, called TARP-G, assumes that the allocation of technicians to O&M bases is *given*, i.e., technicians are not shared between O&M bases. The TARP-G serves as a benchmark to obtain insights on the impact of technician sharing.

We develop a Two-Stage Adaptive Large Neighborhood Search (2-ALNS) heuristic to solve the TARP and its two variants. Our 2-ALNS achieves high-quality (and often optimal) solutions on a set of benchmark instances available in the literature for the TARP-G. Since we are the first to consider large-scale instances for the TARP and its two variants, a set of new benchmark instances is provided. Via simulation, we provide insights on the impact of technician sharing under the different technician allocation policies.To be specific, we varied parameters of the geographical layout of the wind farms, O&M bases, and the maintenance characteristics by performing a Monte-Carlo simulation which embeds the 2-ALNS for the estimation of the expected maintenance and routing costs. It is shown that reassigning technicians increases the robustness of the short-term planning, i.e., extreme scenarios are handled more efficiently. On top of reducing expected maintenance and routing costs, sharing technicians leads to fewer vessel routes on average.

In the remainder of this section, we discuss the literature related to the TARP. Note that TARP-G and TARP-F are special cases of the TARP and the statements regarding the TARP hold for its two variants as well, unless explicitly mentioned. We first consider the general pickup and delivery literature as the TARP can be modeled as a novel variant of a Pickup and Delivery Problem (PDP). Second, we discuss articles that study related problems on maintenance routing and scheduling at offshore wind farms. Finally, we review miscellaneous results related to our heuristic approach and computational analysis.

The TARP can be classified as a new variant of the multi-depot multi-period pickup and delivery problem with multiple commodities. Pickup and delivery problems (Dumas, Desrosiers, and Soumis 1991, Savelsbergh and Sol 1995) consider the fulfillment of transportation requests by developing cost-minimizing vehicle routes. Following the classifications by Berbeglia et al. (2007) and Parragh, Doerner, and Hartl (2008), the TARP combines a one-to-one with a many-to-many pickup and delivery structure. The former structure emerges between nodes representing the start and completion of a maintenance activity; a vessel delivering technicians to a maintenance location needs to pick up the technicians after completion. The latter structure, many-to-many pickup and delivery, is present between nodes representing different maintenance activities; after completion of a maintenance activity, technicians with different skills become

available to start the next activity.

Both pickup-and-delivery structures have been extensively studied. Sophisticated exact solution approaches have been proposed for the one-to-one PDP with time windows, e.g., see Ropke, Cordeau, and Laporte (2007), Ropke and Cordeau (2009), Baldacci, Bartolini, and Mingozzi (2011). An Adaptive Large Neighborhood Search (ALNS) is introduced by Ropke and Pisinger (2006), involving larger instances up to 500 nodes. Given the characteristics of our problem, ALNS is well suited for TARP as well. The many-to-many PDP is studied in several variants. We refer to Hernández-Pérez and Salazar-González (2004) and Hernández-Pérez, Rodríguez-Martín, and Salazar-González (2009) for the single vehicle case, and Hernández-Pérez and Salazar-González (2014) and Hernández-Pérez, Rodríguez-Martín, and Salazar-González (2016) for the multi-commodity case. One of the main areas of application is the so-called bike repositioning problem; a many-to-many pickup and delivery problem where one needs to find cost-minimizing routes for multiple vehicles that re-balance the inventory of bike rental systems (Dell et al. 2016, Bulhões et al. 2018).

Besides the transportation of technicians, spare parts need to be moved as well. This aspect shares characteristics with the fixed multi-compartment vehicle routing problem, in which multiple product types need to be stored separately in capacitated vehicles (see, e.g., Alinaghian and Shokouhi 2018). Instead of a regular vehicle routing structure, as encountered in multi-compartment vehicle routing problems, the TARP has a pickup and delivery structure. To the best of the authors' knowledge, such a structure is only examined in maintenance and routing problems at offshore wind farms.

Maintenance routing and scheduling at offshore wind farms, with a given set of technicians to a single O&M base, is first introduced by Dai, Stålhane, and Utne (2015): A mixed integer programming model is presented and solved with standard commercial solvers. An exact decomposition method for a single period presented by Stålhane, Hvattum, and Skaar (2015). Recently, an extension to multiple wind farms and multiple depots (TARP-G) is discussed by Irawan et al. (2017). Optimal solutions to instances of 24 maintenance activities were found by means of a set partition formulation. Their approach is not applicable for the TARP since the inclusion of technician allocation in the optimization problem increases the solution space exponentially. We, therefore, focus on a heuristic approach capable of finding high-quality solutions for large instances of the TARP.

We embed the 2-ALNS in a Monte-Carlo simulation for an in-depth analysis of the impact of technician sharing for different underlying networks. Such a simulation to obtain insights on complex routing problems are common yet crucial for the analysis.

For example, an analysis of periodic routing problems is given by Palhazi Cuervo, Vanovermeire, and Sörensen (2016). By a full-grid Monte-Carlo simulation on parameters describing the transportation network, they provided insights when it is beneficial for individual logistics carriers to cooperate. A similar simulation approach is taken by Post et al. (2018), incorporating routing heuristics to provide an in-depth analysis of an LNG transportation network.

The remainder of this paper is structured as follows. The complete problem description is given in Section 3.2. In Section 3.3, we introduce the 2-ALNS proposed to solve the TARP and its two variants. In Section 3.4, we show the performance of the 2-ALNS by solving benchmark instances from the literature, and secondly, we solve a new benchmark set consisting of computationally challenging instances. Managerial insights, including the circumstances in which technician sharing is beneficial, are provided in Section 3.5. We conclude and provide suggestions for future research in Section 6.6.

3.2 Problem description

In this section, we give a mathematical description of the Technician Allocation and Routing Problem (TARP). In the TARP, one needs to jointly determine the allocation of technicians to depots (i.e., O&M bases) and the accompanying short-term maintenance planning. After the mathematical description of the TARP, we present additional constraints that lead to the two variants TARP-F and TARP-G (with a fixed sharing policy and without sharing, respectively). At the end of the section, we restructure the travel costs and times to obtain a more efficient and simpler problem structure. This is especially useful for the computational performance of the 2-ALNS. A summary of the used notation can be found in Table 3.1.

3.2.1 The Technician Allocation and Routing Problem (TARP)

The TARP is defined on a graph $G = (\mathcal{N}, \mathcal{A})$, where the node set $\mathcal{N} = \mathcal{N}_d \cup \mathcal{N}_p \cup \mathcal{N}_0$ consists of delivery nodes $\mathcal{N}_d = \{1, \ldots, n\}$, pickup nodes $\mathcal{N}_p = \{n+1, \ldots, 2n\}$, and depots $\mathcal{N}_0 = \{2n+1, \ldots, 2n+B\}$. Every delivery node $i \in \mathcal{N}_d$ has a corresponding pickup node $i + n \in \mathcal{N}_p$, i.e., every delivery and pickup pair represents a single geographical location as they indicate the start and end of a single maintenance activity. Every node $i \in \mathcal{N}_d$ and $(i + n \in \mathcal{N}_p)$ belongs to a wind farm $w_i \in \mathcal{W}$, where $\mathcal{W} = \{1, \ldots, W\}$ is the set of wind farms. Let $\mathcal{T} = \{1, \ldots, T\}$ be the finite time horizon in which the nodes may be visited and let $\mathcal{V}_b = \{1, \ldots, V_b\}$ be the set of

Table 3.1: Summary of sets, parameters and decision variables for the TARP.

Sets	
$\mathcal{N}_d, \mathcal{N}_p, \mathcal{N}_0$	Set of delivery nodes ($\mathcal{N}_d$), pickup nodes ($\mathcal{N}_p$) and depots ($\mathcal{N}_0$).
$\mathcal{W}$	Set of wind W farms.
$\mathcal{T}$	Set of T periods.
$\mathcal{L}$	Set of L technician types.
$\mathcal{V}$	Set of all vessels.
$\mathcal{V}_b$	Set of vessels stationed at depot b.
$\mathcal{R}_{vt}$	Set of all feasible routes of vessel v in period t.
Parameters	
$\widehat{q}_\ell$	Number of available technicians of type ℓ.
q_i^ℓ	Demand of technician type ℓ at node i.
s_i	The service time at node i.
d_i	Due date of maintenance activity i.
p_i	Penalty for exceeding due date of activity i
e_i	Equals 1 if a vessel needs to wait during maintenance activity i.
t_{ij}^v, c_{ij}^v	Travel time and travel costs between nodes i and j by vessel v.
$\tilde{t}_{ij}^v, \tilde{c}_{ij}^{vt}$	Modified travel time and travel costs.
$\hat{c}_\ell$	Costs of using a technician of type ℓ.
$\bar{q}_v^1$, $\bar{q}_v^2$	The spare parts capacity ($\bar{q}_v^1$) and technician capacity ($\bar{q}_v^2$) of vessel v.
ω_{vt}	Maximum duration of vessel v in period t.
τ	Transfer time from a vessel to a node, and vice versa.
c_r	Costs of route r.
θ_i^r	Equals 1 if route r visits node i, 0 otherwise.
γ_ℓ^r	The number of technicians of type ℓ used in route r.
Decision variables	
$x_{bt\ell}$	Number of technicians of type ℓ at depot b in period t.
y_r	Equals 1 if route r is selected in a solution, and 0 otherwise.

heterogeneous vessels at depot $b \in \mathcal{N}_0$, so that the complete set of vessels is given by $\mathcal{V} = \cup_{b \in \mathcal{N}_0} \mathcal{V}_b$. Traversing arc $(i, j) \in \mathcal{A}$ with vessel $v \in \mathcal{V}$ takes t_{ij}^v time.

We consider a set of technicians' types $\mathcal{L} = \{1, \ldots, L\}$. The demand for technician type $\ell \in \mathcal{L}$ at node i equals q_i^ℓ so that $q_i^\ell = -q_{i+n}^\ell$ for all $i \in \mathcal{N}_d$. Each maintenance activity has a service time $s_i \geq 0, i \in \mathcal{N}_d$, i.e., if node $i \in \mathcal{N}_d$ is visited at time $a_i \geq 0$, node $i + n$ cannot be left before time $a_i + s_i$. Every job is scheduled such that it is completed in a single period, and the vessel that delivers technicians to node i is responsible for picking them up at node $i + n$. In addition, a maintenance activity may require a vessel to wait during the performance of the maintenance activity. We denote this with the parameter e_i, equaling 1 if a vessel needs to wait, and 0 otherwise. For notational convenience, we let $e_i = 0$ for all $i \in \mathcal{N}_p$.

The total number of available technicians of type ℓ equals $\widehat{q}_\ell$. For every period $t \in \mathcal{T}$, one needs to determine $x_{bt\ell}$, the number of technicians of type ℓ designed to depot b in period t. Hence for every period t, $x_{bt\ell}$ technicians of type ℓ must be available at the start and end of period t. In addition, each vessel v has a limited capacity $\bar{q}_v^1$ to transport spare parts, a limited capacity $\bar{q}_v^2$ to transport technicians, and transferring technicians from a vessel to any node i takes τ time. The maximum time a vessel v can be utilized in period t equals ω_{vt}. This models restrictions in working hours due to weather conditions.

Each maintenance activity i has a due date $d_i \in \mathcal{T}$. If a maintenance activity is scheduled in some period $t' > d_i$, a penalty p_i per period exceedance is incurred. Using technicians to perform maintenance activities comes at a per period cost of $\hat{c}_\ell$. Utilizing edge $(i,j) \in A$ with vessel $v \in \mathcal{V}$ comes at travel cost c_{ij}^v.

A solution to the TARP consists of an allocation $x_{bt\ell}$ of technicians and accompanying vessel routes, for every vessel $v \in \mathcal{V}_b$, depot $b \in \mathcal{N}_0$, period $t \in \mathcal{T}$, and all $\widehat{q}_\ell$ technicians of type $\ell \in \mathcal{L}$. Let $\mathcal{R}_{vt}$ denote the complete set of feasible vessel routes of vessel v in period t, satisfying the constraints described in the previous paragraphs. Let θ_i^r be equal to 1 if route $r \in \mathcal{R}_{vt}$ visits node $i \in \mathcal{N}_d$ and 0 otherwise. Similarly, let γ_ℓ^r equal the number of technicians of type ℓ used in route r. The costs of a route are given by c^r, consisting of travel costs c_{ij}^v for every edge (i,j) used in route r, and of technician costs $\hat{c}_\ell$ for all technicians of type ℓ deployed in r. Besides the continuous decision variable $x_{bt\ell}$, we let y_r be a binary decision variable equaling 1 if route $r \in \mathcal{R}_{vt}$ is part of the solution, and 0 otherwise.

The following set partitioning formulation models the TARP:

$$\min \quad \sum_{t\in\mathcal{T}} \sum_{b\in\mathcal{N}_0} \sum_{v\in\mathcal{V}_b} \sum_{r\in\mathcal{R}_{vt}} y_r c^r \tag{3.1}$$

$$\text{s.t.} \quad \sum_{r\in R_{vt}} y_r \leq 1 \qquad \forall\, v \in \mathcal{V}_b, b \in \mathcal{N}_0, t \in \mathcal{T} \tag{3.2}$$

$$\sum_{b\in\mathcal{N}_0} \sum_{v\in\mathcal{V}_b} \sum_{t\in\mathcal{T}} \sum_{r\in\mathcal{R}_{vt}} y_r \theta_i^r = 1 \qquad \forall\, i \in \mathcal{N}_d \tag{3.3}$$

$$\sum_{v\in\mathcal{V}_b} \sum_{r\in R_{vt}} y_r \gamma_\ell^r \leq x_{bt\ell} \qquad \forall\, \ell \in \mathcal{L}, b \in \mathcal{N}_0, t \in \mathcal{T} \tag{3.4}$$

$$\sum_{b\in\mathcal{N}_0} x_{bt\ell} \leq \widehat{q}_\ell \qquad \forall\, \ell \in \mathcal{L}, t \in \mathcal{T} \tag{3.5}$$

$$y_r \in \{0,1\}, \quad x_{bt\ell} \geq 0 \qquad \forall\, r \in \mathcal{R}_{vt}, v \in \mathcal{V}_b, b \in \mathcal{N}_0, t \in \mathcal{T} \tag{3.6}$$

The Objective (3.1) minimizes the costs of the chosen routes. Constraint (3.2) models

that each vessel can only perform a single trip per period, and Constraint (3.3) ensures that all nodes are visited exactly once throughout the time horizon. Constraint (3.4) limits the use of technicians of the selected vessel routes to the number of technicians being allocated, and Constraint (3.5) ensures that technician allocation obeys the availability of technicians.

3.2.2 The Technician Allocation and Routing Problem with Fixed Allocations (TARP-F)

In this variant, we study a scenario in which wind farms and depots are located such that changing depots in every period is not practical for the technicians. In the TARP-F, we adhere a fixed allocation policy: The technicians are assigned to depots and stay at their depots over the time horizon $[1, \dots, T]$. This variant arises when the wind farms and depots are located such that period changing of depots is not practical for the technicians. A set partitioning formulation for the TARP-F is obtained by extending formulation (3.1) - (3.6) with

$$x_{bt\ell} - x_{b\tilde{t}\ell} = 0 \quad \forall\, t, \tilde{t} \in \mathcal{T}, b \in \mathcal{N}_0, \ell \in \mathcal{L}. \tag{3.7}$$

3.2.3 The Technician Allocation and Routing Problem with Given Allocations (TARP-G)

To investigate the effects of efficiently coordinating technicians between the depots and wind farms, we consider the Technician Allocation and Routing Problem with Given Allocations (TARP-G). It is derived from the TARP by taking $x_{bt\ell}$ as given, i.e., it is not longer a decision variable but a parameter. The TARP-G is referred to as the Maintenance Routing and Scheduling Problem at offshore wind farms, and an exact solution approach based on a set partitioning formulation has been proposed by Irawan et al. (2017). However, an explicit enumeration of the route sets $\mathcal{R}_{vt}$ is already complex and time consuming for a given technician allocation, and becomes intractable when the technician allocation is included as a decision.

3.2.4 Preprocessing: cost and travel restructuring

Rich routing problems are often characterized with fluctuating travel costs or travel times due to a variety of side constraints. By letting travel costs be period dependent, i.e., the costs of traversing arc (i, j) with vessel v in period t equals $\tilde{c}_{ij}^{vt}$, we can incorporate the due date penalties p_i into the cost function. The modified cost of

traversing edge (i, j) with vessel v in period t are defined as $\tilde{c}_{ij}^{vt} = c_{ij}^{v} + \max\{t - d_i, 0\} \cdot p_i$.

Recall that t_{ij}^{v} is the travel time of vessel v between nodes i and j. Both the transfer time and any required waiting time at location i during maintenance (if $e_i = 1$) are incorporated into the travel time function t_{ij}^{v}. Let $\tilde{t}_{ij}^{v} = t_{ij}^{v} + \tau + e_j s_j$, be the modified travel time function for all $i, j \in N$. To be precise, we add the transfer times to every incoming edge, and the service times to the incoming edges of the delivery nodes if needed. Then, for delivery nodes i with $e_i = 1$, the corresponding pickup can be performed without additional costs after the delivery, since they resemble the same location.

At first, it may seem that the restructuring invalidates well-studied characteristics of routing problems such as the triangle inequality. However, it does not introduce additional complexities for the TARP and its two variants, as the triangle inequality is already violated by the costs for using technicians. To be specific, as technicians can be reused from the pickup nodes at no additional expenses, it may be cheaper to visit a delivery location via a pickup location, instead of directly traveling to the delivery location.

3.3 Adaptive Large Neighborhood Search

We develop a Two-Stage Adaptive Large Neighborhood Search (2-ALNS) to solve the TARP and its two variants. We first give a brief overview of the 2-ALNS procedure. Afterwards, we provide the details of the 2-ALNS and the operators used therein. We conclude this section with some minor differences between solving the TARP-G instances on the one hand, and the TARP-F and TARP instances on the other hand.

Adaptive Large Neighborhood Search (ALNS), as first introduced by Ropke and Pisinger (2006), is a metaheuristic often used to solve complex routing problems, see, for example, Mancini (2016) and Ghilas, Demir, and Woensel (2016). ALNS is an iterative procedure relying on a set of destroy operators, removing parts of a solution, and a set of repair operators, inserting the removed parts into the solution. The successive appliance of destroy operators and a repair operator is called a *move*. A move consists of independently selected destroy and repair operators according to a continuously updated probability distribution. If a move appeared to be successful, the probability of selecting the corresponding operators will slightly increase. Thereby, effective operators are more likely to be chosen in future iterations of the ALNS. As often encountered in current metaheuristics, we embed the iterative destroying and repairing of the solution in a simulated annealing environment. A move is accepted with a probability that depends on the number of iterations and the difference in

Algorithm 3.1: ALNS Procedure

```
s_cur, s_best, s_glob ← initialSolution();
n_1 ← 0 and n_2 ← 0;
while n_1 < N^1_iter do
    s_cur, s_best ← s_glob;
    while n_2 < N^2_iter do
        s_cur ← s_best;
        destroyRepair(s_cur);
        updateProb(s_cur);
        if acceptSolution(s_cur, s_best, n_2) then
            s_best ← s_cur;
        end
        if c(s_cur) < c(s_glob) then
            s_glob ← s_cur;
        end
        n_2 ← n_2 + 1;
    end
    n_1 ← n_1 + 1;
    Shake(s_best, n_1);
end
```

objective value. To widen the search space, a shaking procedure is started when the above procedure terminates (after a fixed number of iterations). The shaking procedure is especially included to cope with the dependency between the allocation of technicians and the optimization of the routes. This relation between the allocation and routes increases the difficulty of escaping local optima. The general outline of the ALNS procedure is given in Algorithm 3.1.

The ALNS procedure as described above is embedded in a two-stage procedure. In the first stage, we start with ten initial solutions and perform, on each of the ten initial solutions, a single run of the ALNS without the shaking procedure. In the second stage, we select the four best solutions from the first stage procedure and improve those four solutions with the full ALNS procedure. This two-stage procedure is the complete 2-ALNS. In the following, a complete description of the ALNS (as part of the 2-ALNS) is presented.

3.3.1 Destroy operators

A destroy operator's goal is to remove potentially suboptimal parts of a solution. This can either be achieved by destroying random parts of the solution, or one could provide more guidance in destroying the solution. Throughout this section, we will use the term *job* to indicate a pickup and delivery node corresponding to the same maintenance activity.

We divide the destroy operators into two classes. The first class, *random destroy operators*, destroys the solution by randomly removing jobs while taking the actual solution structure into account. The second class, *guided destroy operators*, are more sophisticated destroy operators. In the following, we discuss both classes of destroy operators.

3.3.1.1 Random destroy operators

By exploiting the problem structure, four random destroy operators are constructed, called *randomJob*, *randomVessel*, *randomDepot*, and *randomPeriod*. The first one removes a random job, the second operator removes all jobs from a random vessel in a random period, the third one removes all jobs from a random depot in a random period, and the fourth operator removes all jobs in a random period.

3.3.1.2 Guided destroy operators

This class consists of six destroy operators. The first guided destroy operator is the *worstRemoval* operator (e.g., Ropke and Pisinger (2006)). It removes the job that results in the largest cost decrease, i.e., the most expensive job is removed from the solution. When multiple jobs are removed by this operator, it iteratively removes single jobs and recalculates the cost decreases after each removal.

The five other guided destroy operators are based on the Shaw removal heuristic (see, e.g., Ropke and Pisinger (2006)) which tries to identify jobs that have similar characteristics. The main difference with the classical use of the Shaw removal heuristic is that the similarity of jobs does not depend on the actual solution, which improves the computational efficiency of the ALNS. Namely, the similarity of the jobs is computed before starting the ALNS instead of being computed every iteration.

To remove two jobs simultaneously from the solution, we define $S(i,j)$ as the similarity score for two jobs i and j measuring the degree of similarity of jobs i and j. The higher the score, the higher the probability of being selected simultaneously. It is defined in the following way,

$$\begin{aligned} S(i,j) = \hat{\alpha}\left(1 - \frac{\text{dist}(i,j)\chi_{i,j}}{\max_{a,b\in\mathcal{N}}\text{dist}(a,b)}\right) + \hat{\beta}\left(1 - \frac{\sum_{\ell=1}^{L}|q_{i\ell} - q_{j\ell}|}{\sum_{\ell=1}^{L}\max\{q_{i\ell}, q_{j\ell}\}}\right) \\ + \hat{\gamma}\left(1 - \frac{|s_i - s_j|}{\max\{s_i, s_j\}}\right) + \hat{\delta}\left(1 - \frac{|d_i - d_j|}{\max\{d_i, d_j\}}\right), \end{aligned} \tag{3.8}$$

where $\text{dist}(i,j)$ equals the Euclidean distance between nodes i and j, and $\chi_{i,j}$ equals 1 if $w_i = w_j$ and 0 otherwise. The four terms of (3.8) measure for jobs i and j: The relative

Algorithm 3.2: Random Cheapest Feasible Insertion (RCFI)

```
Data: Solution s, repair operator op, set of nodes U
shuffle(U);
foreach v ∈ U do
    insertBestLocation(s, v, op);
end
```

difference in distance, the relative difference in technician demand, the similarity in service times, and the difference in due dates. Let $P(i,j) = S(i,j)/\sum_{a \in \mathcal{N}} S(i,a)$ be the probability of selecting job j while job i is already selected. In other words, job i is randomly selected whereas job j is selected with probabilities $P(i,j)$.

We enhance the use of five destroy operators based on this similarity criteria. For each operator, different weights are attached to the individual parts of $S(i,j)$. The *shaw* operator has positive weight on every individual part, i.e., $\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta} > 0$. The *shaw-distance*, *shaw-technician*, *shaw-service*, and *shaw-penalty* guided destroy operators only have a non-zero value for $\hat{\alpha}, \hat{\beta}, \hat{\gamma}$, and $\hat{\delta}$, respectively.

3.3.2 Repair operators

In total, six repair operators are defined and divided into two classes, namely, the regular *RCFI* operators and the guided *RCFI* operators. The repair operators are variants of the so-called Random Cheapest Feasible Insertion (RCFI). The general structure of RCFI can be found in Algorithm 3.2. It consists of two main procedures. First, the set of to be insterted jobs is randomly sequenced. Second, we insert the jobs one by one on their 'best' location, based on an objective defined by the different repair operators. A single call to RCFI is of order $\mathcal{O}(n^2)$.

Some repair operators evaluate a job's insert location with so-called noise, characterized by the noise parameter $0 \leq \Omega \leq 1$. Noise is a simple, but effective, way to include randomness in the ALNS. The costs used to compare insertion locations is obtained by multiplying the actual insertion costs with a uniformly drawn random number between $1 - \Omega$ and $1 + \Omega$.

3.3.2.1 Regular *RCFI* operators

Regular RCFI operators are straightforward insertion operators based on the total costs of the repaired solution. The first regular RCFI operator, *RCFI-cost*, finds the cheapest location for the job to be inserted. As *RCFI-cost* consideres the jobs in a random sequence, applying it more than once may give better results. The second operator, *RCFI-k-cost*, evaluates the *RCFI-cost* operator k times, and applies the one

that results in the lowest costs. Both operators do not include noise in the objective functions, i.e., $\Omega = 0$. The *RCFI-noise-cost* is similar to the *RCFI-cost* operator, but Ω is strictly positive for this operator.

3.3.2.2 Guided *RCFI* operators

Guided RCFI operators are tailored operators to steer the solution into a direction that uses more or fewer technicians or vessel trips. The main purpose is to quickly escape local optima during the search. The fourth repair operator, *RCFI-noise-tech*, only considers the use of technicians by assuming zero travel costs, and includes noise. The fifth repair operator, *RCFI-vessel*, favors insertion in unused vessels by ignoring the travel costs if a job is inserted in an empty vessel. For the rest, it is similar to *RCFI-noise-cost*. The last repair operator, *RCFI-alloc*, penalizes the use of technicians that cause an increase in the allocation of that particular technician type to a depot. To be precise, a penalty $\eta > 0$ per unit increase of the allocation is added to the costs of the particular insertion location. In addition, it does not consider travel costs.

3.3.3 Destroy and repair procedure

At the core of the ALNS lies the Destroy and Repair Procedure. Its general overview is presented in Algorithm 3.3. The repair and destroy operators deployed are the ones described in the previous sections. The Destroy and Repair Procedure consists of n_{d} times applying a randomly selected destroy operator, and afterwards repairing the solution by one of the repair operators. The number n_d is randomly drawn between $\underline{n}_d$ and $\overline{n}_d$ in each iteration.

Each of the n_d destroy operators is applied a random number of times. We call this the destroy intensity: The number of entities a destroy operator removes from the solution. For destroy operators that consider individual jobs, it equals the number

Algorithm 3.3: destroyRepair(s)

```
Node set U ← ∅;
k ← 0;
while k < N_init do
    op ← chooseRandomDestroy();
    v ← destroy(s, op);
    U ← U ∪ v;
    k ← k + 1;
end
op ← chooseRandomRepair;
repair(s, op);
```

of removed jobs. For destroy operators that consider vessels, depots or periods, it denotes the number of those entities. The destroy intensity is a randomly drawn number between $\underline{I}_i$ and $\overline{I}_i$, where i refers to the class the destroy belongs to, i.e., $i = 1$ refers to the random destroy operators and $i = 2$ to the guided destroy operators. The actual destroy intensity is determined each time an operator is applied.

Besides operator settings, ALNS adapts the probability of the operators being selected depending on a move being successful or not. To do so, we keep an integer value $p_j > 0$ for every operator j individually. Their initial values are set as part of the operator group they belong to. These initial values are given by p_i^{init}, where $i = 1$ and $i = 2$ are defined as above, $i = 3$ refers to the class of regular RCFI operators, and $i = 4$ to the class of guided RCFI operators. If a move is accepted, the values p_j of the operators involved in the move are increased by $\Delta > 0$. Thus, drawing a random number between 0 and $\sum_j p_j$ is sufficient to select an operator. Finally, if the shaking procedure is started, the operator values p_j are reset to the initial value p_i^{init} of their corresponding operator group i.

3.3.4 Initial solution

Due to the multiple constraints and aspects of the TARP and its two variants, we first generate a set of N_{init} random solutions. Each random solution is generated by calling the *RCFI-Cost* repair operator on an empty solution, i.e., all jobs are inserted in a random order on their best location into the solution. We select the solution with the lowest cost. Notice that in the complete 2-ALNS procedure, we start with 10 initial solutions, each being generated as described in the former.

3.3.5 Acceptance criterion

We use a classical simulated annealing criterion to determine the acceptance of a move. It is defined by a cooling parameter $c_{\text{sa}} > 0$ and a weight parameter $w_{\text{sa}} > 0$. Let u_{old} and u_{new} be the objective before and after the move, and let n_{iter} be the current iteration. If $u_{\text{new}} < u_{\text{old}}$, we always accept the move. If a move results in a higher objective value, we calculate the acceptance probability of the move as

$$\exp\left(\frac{-w_{\text{sa}}\hat{u}}{\hat{n}^{c_{\text{sa}}}}\right),$$

where $\hat{u} = (u_{\text{new}} - u_{\text{old}})/u_{\text{old}}$, and $\hat{n} = 1 - n_{\text{iter}}/N_{\text{iter}}^2$. A higher weight parameter steers towards a more deterministic search, i.e., the probability of accepting a move that results in a higher objective value is lower. In addition, a higher cooling parameter

results in lower acceptance probabilities for the first few iterations.

3.3.6 Shaking procedure

When N_{iter}^2 iterations of applying the destroy and repair procedure have passed (see Algorithm 3.1), a shaking procedure is started to escape from a possible local optimum. It is a destroy and repair procedure as indicated by Algorithm 3.3, although two major changes are made. First, only destroy operators of the random destroy operators class are used. Second, only repair operators *RCFI-noise-vessel* and *RCFI-noise-alloc* are used in the shaking procedure, being chosen at random. In addition, its intensity depends on the number of passed shaking procedures n_{fail} that did not yield a new best solution: It is applied n_{fail} times, with a minimum of 1 and a maximum of $n_{\text{fail}}^{\max}$ times. Preliminary experiments have shown that this works well.

3.3.7 Differences between TARP, TARP-F, and TARP-G

It is clear that the TARP and TARP-F have exponentially larger solution spaces than the TARP-G. The freedom of deviating from a given allocation of technicians causes the solution space to be rather unstructured. To avoid being trapped in local optima, we change the first stage of the two-stage ALNS when we are solving TARP(-F) instances: We perform the first stage of the ALNS two times for each of the 10 instances. First, we run it as if it was a TARP-G instance, thereby taking the given allocation of technicians randomly around the expected number of technicians. Then we take this solution as an initial solution for another round of the first stage, but now solving the instances as they are. The second stage is similar for each problem type.

3.4 Two-stage ALNS performance

We evaluate the performance of the 2-ALNS by solving benchmark instances from the literature, obtained from Irawan et al. (2017). In addition, we generate a new set of computationally challenging benchmark instances. The 2-ALNS is implemented in *C++* and the experiments are performed on an Intel Xeon E5 2680v3 CPU (2.5 GHz). We made use of four parallel running threads on a single core, a setup available on any common desktop machine nowadays. In the following, we first describe our benchmark instances in more detail. We follow with a description of the 2-ALNS' parameter tuning and conclude with its computational performance.

Table 3.2: The effect of the destroy operators when solving TARP instances. For each operator, we denoted the maximum average objective increase if that particular operator is excluded from the 2-ALNS

Operator	Description	max increase (%)
randomJob	removes random jobs	1.22
randomVessel	removes random vessels	0.68
randomDepot	removes random depots	1.47
randomPeriod	removes random periods	1.62
worstRemoval	removes jobs leading to largest objective decrease	1.17
shaw	removes jobs based ons shaw criteria	0.75
shaw-distance	removes jobs based on distance	0.73
shaw-technician	removes jobs based on technician use	0.58
shaw-service	removes jobs based on service time similarity	1.07
shaw-penalty	removes jobs based on due date similarity	0.72

3.4.1 Benchmark instances

Two existing sets of benchmark instances (G1 and G2) are obtained from Irawan et al. (2017). Since their problem is equivalent to the TARP-G in our setting, we compared the performance of the 2-ALNS with the exact approach by Irawan et al. (2017). Both benchmark sets comprise of two depots and three wind farms, with two available vessels per depot. The number of technician types L equals three. We refer to their paper for a complete description of parameter settings. Benchmark set G1 consists of 10 instances with 24 maintenance activities over a planning horizon of 3 days, whereas set G2 consists of 10 instances with 36 maintenance activities over a planning horizon of 7 days.

In addition to benchmark sets G1 and G2, we created a new benchmark set H. The instances herein vary in size and in the length of the time horizon, see Table 3.4. The parameters regarding the vessels, depots, technicians, and the location of the turbines and wind farms are the same as of those in G1 and G2. The actual maintenance activities in each instance are obtained in the following way: For the first two wind farms, we randomly draw the number of maintenance activities between 15 and 35 percent of the total number of activities of that particular instance. The third wind farm takes the remaining number of maintenance activities. Due dates are uniformly drawn over the time horizon, service times are uniformly drawn between 2 and 6 hours, and spare part weights are uniformly drawn between 300kg and 900kg. The number of required technicians is randomly between 1 and 3 for each technician type. This results in benchmark instances similar to the instances of G1 and G2. However, as opposed to the instances of G1 and G2, the number of jobs per wind

farm may vary. Preliminary experiments showed that this increases the complexity of the instances, which is in line with our aim of to create computationally challenging instances reflecting practical scenarios in offshore wind maintenance.

3.4.2 Parameter tuning

An overview of the parameter tuning of the 2-ALNS is provided in Appendix A. It describes an extended grid search through the main parameters. The parameter values used for benchmarking the 2-ALNS are given in Appendix A, Table 3.A.1.

3.4.3 The impact of the destroy operators

We continue by studying the impact of the destroy operators on the performance of the 2-ALNS. For this purpose, we solve a randomly generated set of instances (constructed as in dataset H) with 40, 45, 50, 55 and 60 maintenance activities. For each destroy operator, two experiments are performed. First, all the instances are solved with the 2-ALNS in which only that particular destroy operator is included, and none of the others. Second, we solve the instances while excluding that particular destroy operator and including all other destroy operators. This led to 21 experiments, i.e., two experiments for each operator and the complete 2-ALNS for comparison. All other elements of the 2-ALNS (e.g., the repair operators) are as described in Section 3.3.

Running the complete 2-ALNS with a single destroy operator is on average 3.6% worse than including all the destroy operators. This is mainly due to the *worstRemoval* operator, which is on its own 21.9 % worse. All other destroy operators, when included in isolation, result in average cost increases up to 7%.

The results for excluding a particular operator are presented in Table 3.2. For each of the destroy operators it holds that when excluded, one or more instances showed a deterioration in solution quality of at least 0.58%. This is on average 0.99% for the random destroy operators and 0.84% for the guided destroy operators. Especially excluding the *randomDepot* operator seems to harm the solution quality, as it leads to a 1.46% decrease in the average solution quality for particular instances. Moreover, it must be noted that the number of operators is not a significant factor in the calculation times, as the total number of iterations is fixed and the operators are randomly selected. We therefore include all the destroy operators into the 2-ALNS.

Table 3.3: Performance of the two-stage ALNS for benchmark sets G1 and G2 from Irawan et al. (2017)

	Irawan et al. (2017)		TARP-G (min)		TARP-G (avg)			TARP-F (min)		TARP-F (avg)			TARP (min)		TARP (avg)		
Instance	obj.	t(s)	obj.	gap %	obj.	gap %	t(s)	obj.	gap %	obj.	gap %	t(s)	obj.	gap %	obj.	gap %	t(s)
G1-01	41839.74	244.69	41839.74	0.00	41839.74	0.00	4.17	40422.90	-3.39	40432.48	-3.36	4.06	40168.21	-4.00	40168.21	-4.00	8.11
G1-02	33425.30	542.18	33425.30	0.00	33425.30	0.00	3.61	33425.30	0.00	33425.30	0.00	3.72	33425.30	0.00	33425.30	0.00	6.86
G1-03	36501.28	317.69	36501.28	0.00	36502.35	0.00	3.47	36038.08	-1.27	36177.04	-0.89	3.54	36038.08	-1.27	36038.08	-1.27	6.60
G1-04	31316.73	2356.46	31316.31	0.00	31316.31	0.00	3.53	31316.31	0.00	31316.31	0.00	3.86	31316.31	0.00	31316.31	0.00	6.79
G1-05	31147.73	549.59	31147.73	0.00	31147.73	0.00	3.89	30961.79	-0.60	31047.65	-0.32	4.15	30961.79	-0.60	30961.79	-0.60	7.38
G1-06	36141.68	134.34	36141.68	0.00	36141.68	0.00	3.76	36141.68	0.00	36141.68	0.00	4.33	36043.37	-0.27	36043.37	-0.27	7.68
G1-07	29995.24	1779.76	29995.24	0.00	29995.24	0.00	3.99	29995.24	0.00	29995.24	0.00	4.43	29995.24	0.00	29995.24	0.00	7.58
G1-08	37242.48	534.70	37242.48	0.00	37242.48	0.00	4.16	37242.48	0.00	37242.48	0.00	4.36	37242.48	0.00	37242.48	0.00	7.91
G1-09	30779.64	3465.71	30779.64	0.00	30779.64	0.00	4.10	30779.64	0.00	30779.64	0.00	4.29	30779.64	0.00	30779.64	0.00	7.66
G1-10	33030.87	674.37	33030.87	0.00	33030.87	0.00	4.23	33030.87	0.00	33030.87	0.00	4.28	33030.87	0.00	33030.87	0.00	8.33
G2-01	50835.10	1775.46	50896.33	0.12	51117.82	0.56	6.15	50835.10	0.00	51009.10	0.34	6.92	50814.94	-0.04	51030.02	0.38	7.17
G2-02	49047.09	3369.74	49047.09	0.00	49055.58	0.02	6.17	49047.09	0.00	49078.06	0.06	6.99	49047.09	0.00	49056.31	0.02	7.12
G2-03	44026.05	3518.18	44049.88	0.05	44050.40	0.06	6.86	44026.05	0.00	44048.02	0.05	7.79	44026.05	0.00	44045.37	0.04	7.93
G2-04	54139.22	1505.83	54139.22	0.00	54139.22	0.00	6.55	53564.54	-1.06	53564.54	-1.06	7.10	52755.73	-2.56	52755.73	-2.56	7.36
G2-05	53733.12	3549.63	53733.12	0.00	53733.12	0.00	6.35	52751.86	-1.83	52751.86	-1.83	6.92	52725.82	-1.87	52725.82	-1.87	7.32
G2-06	49156.52	3456.92	49156.52	0.00	49230.68	0.15	6.53	49156.52	0.00	49299.51	0.29	7.38	49156.52	0.00	49190.27	0.07	7.61
G2-07	49315.18	3436.71	49315.18	0.00	49327.49	0.02	6.35	49315.18	0.00	49315.18	0.00	7.24	49315.18	0.00	49323.39	0.02	7.37
G2-08	51051.50	1422.22	51051.50	0.00	51055.03	0.01	6.26	51051.50	0.00	51060.71	0.02	7.30	51051.5	0.00	51070.91	0.04	7.14
G2-09	57920.34	907.24	57920.34	0.00	57927.26	0.01	6.67	57920.34	0.00	57928.91	0.01	7.21	57920.34	0.00	57932.11	0.02	7.62
G2-10	52668.30	2866.38	52668.30	0.00	52668.30	0.00	6.73	52367.14	-0.57	52446.52	-0.42	7.55	52264.39	-0.77	52326.04	-0.65	7.80
Avg.	42665.66	1820.39	42669.89	0.01	42686.31	0.04	5.18	42469.48	-0.44	42504.56	-0.36	5.67	42403.94	-0.57	42422.86	-0.53	7.47

Table 3.4: Performance of the 2-ALNS for Benchmark Set H. For four instances, no feasible solution is found with the TARP-F and TARP-G, as indicated with a '-'. The percentage transportation costs, personnel costs and penalty costs are denoted in the columns headed by d_{cost}

			TARP				TARP-F					TARP-G				
Instance	n	T	$z_{\min}$	d_{cost}	z_{avg}	t(s)	$z_{\min}$	% dif.	d_{cost}	z_{avg}	t(s)	$z_{\min}$	% dif.	d_{cost}	z_{avg}	t(s)
H-01	20	5	41009.20	(32, 60, 8)	41009.20	6.61	41719.16	1.73	(33, 59, 8)	41719.16	3.86	44010.35	7.32	(32, 56, 12)	44010.35	3.87
H-02	20	5	40200.06	(30, 67, 4)	40200.06	8.81	41460.18	3.13	(32, 65, 3)	41460.18	4.79	42007.17	4.50	(30, 64, 6)	42007.17	3.35
H-03	20	5	46161.78	(30, 55, 14)	46161.78	5.72	46178.70	0.04	(28, 55, 16)	46178.7	3.57	47885.12	3.73	(30, 55, 14)	47885.12	3.39
H-04	25	5	50886.58	(30, 63, 6)	50886.58	7.41	50886.58	0.00	(30, 63, 6)	50886.58	4.6	54518.67	7.14	(29, 56, 14)	54518.67	4.39
H-05	25	5	64242.77	(27, 58, 15)	64242.77	6.53	68058.22	5.94	(23, 53, 24)	68062.35	3.98	70921.57	10.40	(25, 52, 23)	70921.57	3.76
H-06	25	5	44104.52	(33, 67, 0)	44104.52	7.12	44104.52	0.00	(33, 67, 0)	44574.35	4.26	45262.92	2.63	(31, 61, 7)	45262.92	4.21
H-07	30	6	61915.15	(28, 65, 8)	61942.24	9.08	64414.22	4.04	(30, 59, 11)	64439.29	5.08	65119.43	5.18	(31, 59, 11)	65119.43	5.50
H-08	30	6	55523.02	(28, 63, 9)	55523.02	8.88	55800.53	0.50	(29, 63, 8)	55803.13	5.50	56719.30	2.15	(29, 63, 8)	56719.30	5.54
H-09	30	6	69244.24	(28, 62, 11)	69244.24	8.64	73072.19	5.53	(26, 56, 18)	75481.43	4.43	76635.92	10.67	(27, 55, 18)	76635.92	4.36
H-10	35	6	86464.86	(25, 58, 17)	86492.27	19.36	-	-	-	-	-	-	-	-	-	-
H-11	35	6	62763.19	(31, 66, 3)	62782.29	10.54	63586.56	1.31	(31, 66, 3)	64913.34	6.04	66962.10	6.69	(32, 63, 5)	67012.53	6.11
H-12	35	6	66825.21	(29, 61, 10)	66904.33	9.64	66875.44	0.08	(29, 61, 10)	67313.31	5.46	67526.70	1.05	(30, 60, 10)	67526.70	5.47
H-13	40	7	80437.54	(29, 66, 6)	80518.61	27.14	-	-	-	-	-	-	-	-	-	-
H-14	40	7	76387.97	(29, 64, 7)	76421.25	10.25	78681.03	3.00	(29, 62, 9)	78681.03	6.04	79466.96	4.03	(29, 62, 9)	79584.81	6.21
H-15	40	7	83268.38	(28, 59, 13)	83556.70	12.20	89594.48	7.60	(29, 56, 15)	89866.18	6.99	90758.93	9.00	(27, 54, 19)	90805.34	7.21
H-16	45	8	87730.87	(28, 62, 10)	87987.42	13.37	95973.41	9.40	(29, 56, 15)	96972.31	7.17	99592.69	13.52	(26, 53, 21)	100118.49	7.06
H-17	45	8	84580.57	(30, 62, 8)	84891.96	20.14	90163.54	6.60	(29, 60, 11)	91611.49	15.74	91792.10	8.53	(30, 57, 13)	91863.09	7.42
H-18	45	8	88896.50	(25, 58, 17)	89014.79	22.74	93168.44	4.81	(27, 57, 17)	94486.24	14.41	98603.87	10.92	(24, 52, 24)	99024.75	9.23
H-19	50	9	83282.72	(31, 67, 2)	83736.41	19.33	85752.39	2.97	(31, 67, 2)	86163.71	8.49	85752.39	2.97	(31, 67, 2)	85791.57	8.72
H-20	50	9	90408.42	(30, 66, 5)	91681.52	16.18	93881.95	3.84	(28, 64, 7)	95178.30	8.56	96703.78	6.96	(28, 62, 10)	97112.26	8.93
H-21	50	9	86663.27	(31, 66, 3)	86947.44	17.70	90015.99	3.87	(30, 67, 3)	91917.10	10.43	91234.05	5.27	(33, 63, 4)	91497.11	8.59
H-22	55	10	108868.20	(28, 66, 6)	109478.67	49.02	-	-	-	-	-	-	-	-	-	-
H-23	55	10	94744.55	(29, 69, 2)	95217.99	17.63	98036.58	3.47	(31, 66, 3)	99442.43	9.16	98541.14	4.01	(32, 65, 3)	98861.88	9.23
H-24	55	10	124547.25	(23, 56, 21)	125697.84	35.28	130733.18	4.97	(25, 53, 22)	134571.77	23.9	137397.52	10.32	(24, 49, 27)	138366.20	13.29
H-25	60	10	117622.65	(27, 64, 9)	118752.65	67.36	-	-	-	-	-	-	-	-	-	-
H-26	60	10	101444.19	(29, 71, 0)	102188.35	24.28	103394.53	1.92	(33, 67, 0)	104629.78	11.41	103486.01	2.01	(33, 67, 0)	104526.42	12.39
H-27	60	10	102830.53	(33, 66, 1)	104249.67	22.00	108044.30	5.07	(33, 64, 3)	110170.64	10.07	108249.10	5.27	(32, 64, 4)	109361.19	10.31
Avg.			77816.82		78142.02	17.89	77112.87	3.47		78022.73	8.00	79093.38	6.27		67575.29	5.87

3.4.4 Computational results

We solve the TARP-G instances of benchmarks G1 and G2 and compare these with exact solutions of the literature. Results are presented in Table 3.3. we solve the benchmark instances as if they were TARP(-F) instances as well. The instances from benchmark H are solved as TARP-G, TARP-F, and TARP instances. The results are presented in Table 3.4.

The results of solving the instances from benchmarks G1 and G2 are presented in Table 3.3. The first two columns represent the objective (obj.) and computation time in seconds of the exact approach of Irawan et al. (2017). The next two columns provide the minimum solution found by the 2-ALNS over ten runs, solving the instances as the TARP-G, and the corresponding gap with the best-known solution. The average values corresponding to the ten runs are provided afterwards. The columns headed by 'TARP-F (min)' and 'TARP (min)' present the minimum objective value and the corresponding gap to the best-known solution when we solve the instances as the TARP-F and TARP, respectively. We included the average results over 10 runs of the TARP and TARP-F as well.

It is clear that the 2-ALNS provides high-quality, and often optimal, solutions to the instances of Irawan et al. (2017).Computation times are on average 5.25 seconds, a decrease of factor 347 compared with the exact approach, at the expense of an average optimality gap of only 0.04%. Although the objective values of the instances of benchmark set G2 are not proven to be optimal by Irawan et al. (2017), the 2-ALNS obtaines similar results. Moreover, expirements have shown that by increasing the 2-ALNS' run time, it is possible to close the gap to the best-known solutions. The results indicate that often the complete symmetric allocation of technicians is best for the instances of Irawan et al. (2017), as solving the instances as TARP and TARP-F only leads to slight improvements. We provided more insights regarding the differences between the TARP, TARP-F and TARP-G with the instances of benchmark set H, and in Section 5.

Table 3.4 presents the results of our benchmark set H. Every instance is solved by the 2-ALNS as a TARP, TARP-F, and a TARP-G instance, as the column headings indicate. For each problem type, we report the minimum objective, the average objective, and the average computation time over 10 runs. In addition, we provide the distribution of the costs of the minimum solution under 'd_{cost}'. It indicates the percentage of the total costs belonging to transportation costs, personnel costs and penalty costs. The column '% dif.' indicates the increase of the minimum objective when compared to the TARP. For four instances, no feasible solutions can be found if they are treated as TARP-G or TARP-F, as marked with a '-'.

The merits of the flexible allocation policy of the TARP can be observed, as the TARP-F and TARP-G give average objective increases of 3.47% and 6.27%, respectively. The impact of the fixed allocation policy (TARP-F) is instance dependent, and will be further investigated in the simulation study in Section 5. Finally, computation times of the TARP and TARP-F are somewhat larger than of the TARP-G, because the TARP and TARP-F have exponentially larger solutions spaces than the TARP-G.

3.5 Managerial insights

The 2-ALNS provides optimal or near-optimal solutions to the short-term maintenance planning problem, as shown in Section 3.4. In benchmark set H, it is observed that sharing technicians, according to both the flexible and fixed allocation rule, reduces the overall maintenance costs. Recall that in the TARP, technicians are completely shared between O&M bases, whereas in the TARP-F, technicians are assigned to O&M bases but they cannot change between O&M bases over time. In the TARP-G, we take the number of technicians at each O&M base equal to the expected number of needed technicians, i.e., we do not include the assignment as a decision in the TARP-G.

We study in more detail the impact of technician sharing. We are especially interested in the characteristics of the underlying network in which technician sharing is most promising. We do so by an extensive Monte-Carlo simulation, in which we vary both the instance characteristics (e.g., number of technicians, number of jobs) and layout characteristics (e.g., the distance between wind farms and depots). First, we describe the set-up of the Monte-Carlo simulation. Second, we provide managerial insights on technician sharing based on the outcomes of the simulation. In particular, we answer the following questions:

1. Does technician sharing contribute to sustainability by reducing the waste of valuable resources such as technicians and vessels?
2. What is the added value of a completely flexible allocation rule over a fixed allocation rule.
3. Which geographical settings offer opportunities for cost-savings by means of technician sharing?

3.5.1 Simulation set-up

We summarize the simulation set-up here, and refer to Appendix B for the specifics. Throughout all experiments, we consider three O&M bases that are α kilometer apart.

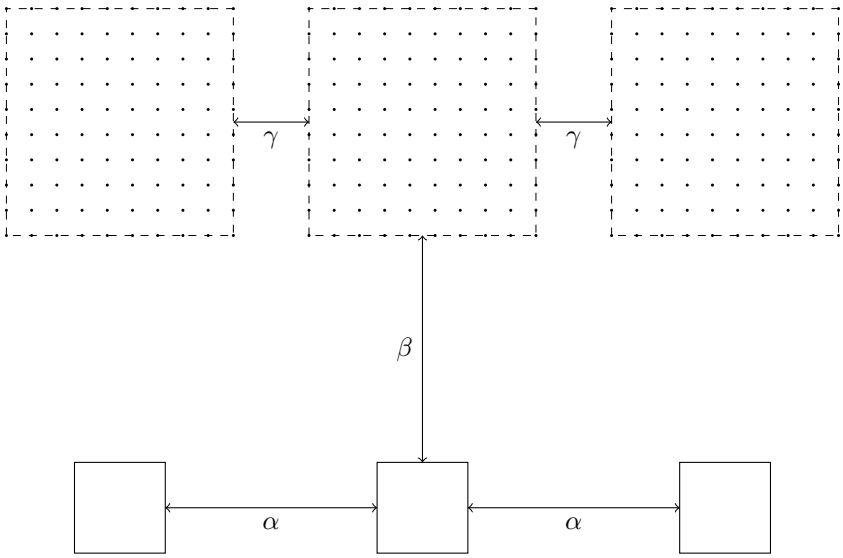


Figure 3.1: General layout used in the simulation study. The three squares at the bottom represent the O&M bases and the grids of dots represent the wind farms. The distances α, β, and γ are given parameters and are varied in the simulation study.

The distance between the O&M bases and the offshore wind farms equals β kilometer, and the distance between the wind farms is γ kilometer. We refer to α, β and γ as the *layout parameters*, see Figure 3.1.

The *instance parameters* of the simulation study are the number of available technicians $\widehat{q}_\ell$, the probability p of a turbine requiring maintenance, and the time horizon T. The fleet of vessels is kept fixed throughout all experiments, of which the specifics can be found in Appendix B. Other parameters, including service times and technician costs, are chosen in accordance with those in dataset H, as described in Section 3.4.1.

A single simulation experiment is defined by the instance and layout parameters described above. For each simulation experiment, we generate N_{sim} instances. Those are solved as the TARP, TARP-F and the TARP-G, where, in the latter one, we allocate technicians equally among the O&M bases as this coincides with the expected number of jobs serviced from the O&M bases. The 2-ALNS is deployed to approximate the expected cost of the short-term maintenance planning in each instance, under the instance and layout characteristics of that particular experiment. Each of the N_{sim} instances is generated by randomly selecting jobs for each wind farm, i.e., a turbine needs maintenance within the considered time horizon with probability p. As

Table 3.5: The mean and 90%, 95% and 99% quantiles of the performance measures. Given values are averages over all experiment runs for $T = 5$.

Cost Measure	Problem type	mean	90%	95%	99%
Total Costs	TARP-G	48427.56	66674.88	72284.24	76588.63
	TARP-F	47568.13	65788.28	70797.22	75178.79
	TARP	44879.77	61390.62	66309.56	70631.05
Penalty Costs	TARP-G	3649.70	7649.71	9040.22	10187.87
	TARP-F	3107.83	7020.74	7957.02	9191.37
	TARP	1466.98	3193.96	3800.00	4476.93
Number of trips	TARP-G	12.62	15.84	16.34	16.64
	TARP-F	12.24	15.46	16.08	16.40
	TARP	10.92	13.59	13.96	14.19
Mean time to maintenance	TARP-G	1.30	2.21	2.29	2.34
	TARP-F	1.16	2.15	2.27	2.31
	TARP	0.85	1.68	2.02	2.15

Table 3.6: Overview of the percentage feasible instances with $T = 5$ for different combinations of α and β.

	$\alpha, \beta \leq 30$	$\beta \leq 30, \alpha > 30$	$\beta > 30, \alpha \leq 30$	$\alpha, \beta > 30$
TARP-G	83.13	80.18	74.97	70.35
TARP-F	83.79	80.40	75.67	71.09
TARP	89.58	89.50	84.20	84.27

a result, the number of maintenance activities of the instances differs within a single experiment.

A complete overview of the instance and layout parameters can be found in Appendix B. We performed a complete grid search over the instance and layout characteristics space, obtaining cost estimations of the short-term maintenance planning for a total of 1500 scenarios. Common random numbers are used in every experiment to reduce the variance in our results. All experiments consist of $N_{\text{sim}} = 100$ instances, unless stated otherwise. Preliminary experiments have shown that this provides stable simulation outcomes.

3.5.2 Simulation outcomes

Four performance measures are used to assess the performance of the different technician allocation policies: The resulting total costs, the penalty costs, the number of needed visits to the wind farms (called trips), and the mean time to maintenance. First, we provide an in-depth analysis of the effect of technician sharing. Second, results on the relation between the layout parameters and the different sharing policies are presented.

3.5.2.1 Technician sharing analysis

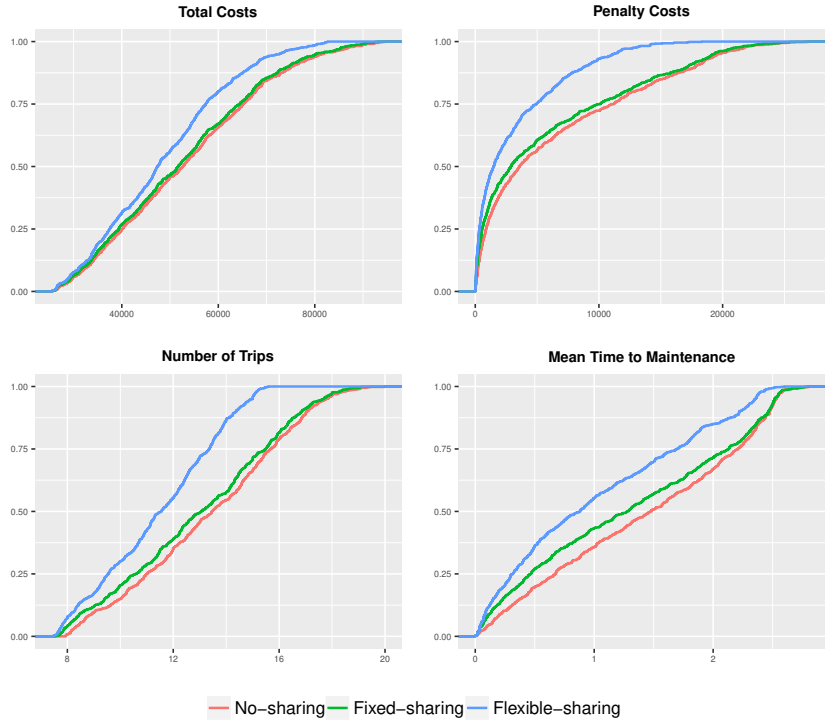


Figure 3.2: Empirical distribution functions of the Total Costs, Penalty Costs, Number of Trips and the Mean Time to Maintenance for all experiments with $T = 6$.

The mean and the 90%, 95%, and 99% quantiles of the four performance measures are summarized in Table 3.5. It shows the average outcomes of all experiments with $T = 5$. Instances that are infeasible by the TARP-G are excluded for the TARP-F and TARP as well. Analyzing the effect of technician sharing in general, it is seen that the number of trips decreases without increasing the mean time to maintenance. As a matter of fact, both the mean and the high quantiles of the mean time to maintenance are reduced if technicians are shared. The average cost increase of not allowing technicians to be shared equals 1.81% and 7.91% compared to the TARP-F and TARP, respectively. The main reason is the increase in penalty costs, which are 17.44% and 148.79% compared with the TARP-F and TARP, respectively. Summarizing, technician sharing leads to a decrease in the mean time to maintenance and the penalty costs, hence more maintenance activities are performed within their deadline. This allows the decision maker to cope more easily with future disturbances.

As mentioned before, not all the problem instances included in the experiments are feasible. Table 3.6 shows the percentage of instances that could be solved by each of the three problem variants, for different combinations of α and β. The TARP-F slightly increases feasibility of problem instances when $T = 5$, whereas the TARP shows a substantial increase of the feasibility of the instances at hand (e.g., from 70.35% to 84.27% for higher values of α and β). Thus, sharing of technicians is a natural way to cope with difficult short-term maintenance planning problems.

For the experiments with $T = 6$, empirical distribution functions of the four measures are given in Figure 3.2. With flexible sharing, 55% of the solutions use fewer than 12 trips while with a given allocation (no sharing) only 27% of the solutions use at most 12 trips. The maximum number of trips is reduced from 19 to 15. We conclude that fexibly sharing technicians effectively reduces the number of trips, while, at the same time, the mean time to maintenance is reduced.

Based on the simulation results presented so far, we can answer the first research question, posed at the beginning of Section 5, with the affirmative. Indeed, technician sharing contributes to sustainability as vessels and technicians are used more efficiently. We argue that sharing of technicians can be used as a tool by the operational manager to increase its ability to cope with extreme scenarios. In particular, flexibly allocating technicians clearly has its merits compared with the other allocation rules. For offshore wind operations that are relatively nearby, but operated in isolation, this can function as a a simple but effective way to reduce costs.

3.5.2.2 Experiment layout impact

The Monte-Carlo simulation allows us to study the effect of the *layout parameters* on the efficiency of technician sharing. We present the results of the experiments with $T = 6$, as few instances are infeasible. We excluded the extreme scenarios with a minimum number of technicians and the highest job probability, as few comparisons could be made due to the high number of infeasible instances for the TARP-G.

In Figure 3.3, the effect of α (the distance between wind farms) on the performance of the different sharing policies is depicted. We compare the number of trips and the incurred penalty costs (as a proxy of the total costs) for two scenarios in which the wind farms are either relatively close, or relative far away, from the O&M bases. From Figure 3.3, one can observe that if O&M bases and wind farms are relatively far apart (higher values of α), the penalty costs can be reduced by allowing technician sharing. In addition, the decrease in the number of trips is slightly decreasing for higher values of α, meaning that the number of trips is most efficiently reduced when O&M bases are relatively close.

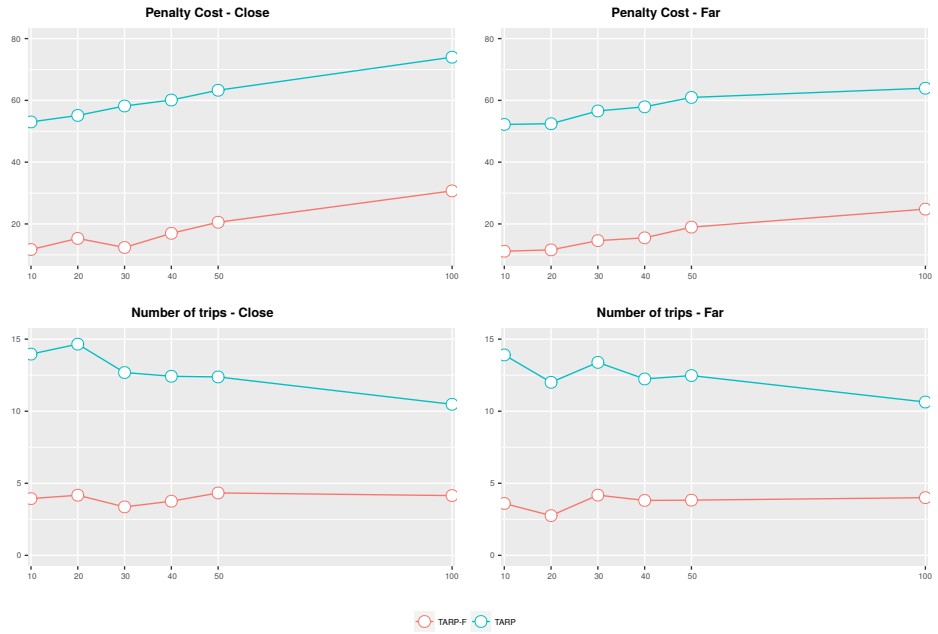


Figure 3.3: Percentage decrease in penalty costs and number of trips for the fixed (TARP-F) and the flexible (TARP) technician allocation policies for $\alpha \in \{10, 20, 30, 40, 50, 100\}$. The left two plots are for $\beta = 20$ and the right two plots are for $\beta = 40$

Reflecting on the second and third research question, we conclude that a flexible allocation policy increases robustness significantly compared to a fixed allocation policy. Additional experiments show that, for specific layouts, the fixed allocation performs significantly better than the no-sharing policy. This is especially true for wind farms operated in relative isolation, as we allow every wind farm to be served from every O&M Base.

3.6 Conclusions

In this paper, we introduced the Technician Allocation and Routing Problem for offshore wind farms (TARP). Its goal is to jointly optimize the daily allocation of technicians to O&M bases and the daily vessel routes transporting those technicians to offshore wind farms, in order to perform all maintenance activities throughout a given time horizon. We developed a Two-Stage Adaptive Large Neighborhood Search heuristic to solve the TARP and two variants. The first variant restricts the allocation of technicians to O&M bases to be constant throughout time, whereas the second

variant takes an allocation as given. We showed that the Two-Stage Adaptive Large Neighborhood Search provides high-quality, and often optimal, solutions on benchmark instances from the literature. In addition, we provided a new set of benchmark instances that are computationally challenging.

We embedded the Two-Stage Adaptive Large Neighborhood Search in a Monte-Carlo simulation to study the impact of the technician sharing in different practical scenarios of offshore wind maintenance service logistics. It is found that an increase in technician sharing reduces the number of vessel trips due to a more efficient use of the scarcely available technicians, which contributes to a more sustainable short-term maintenance planning. In addition, it is shown that the increased flexibility of technician sharing allows the short-term maintenance planner to cope with extreme scenarios more efficiently, leading to average cost-savings around 7%.

The opportunities for further research are numerous. To evaluate the effectiveness of the proposed solutions, one could could impose stochastic travel times or stochastic weather conditions via a rolling horizon framework. Another possibility is the inclusion of strategic decisions regarding the composition of the fleet. Especially the decrease in vessel trips when technician are shared may contribute to significant cost-savings for such strategic decisions.

Appendices

3.A Parameter calibration

In this appendix, we present the details of the parameter calibration, and the resulting parameter values for benchmarking the 2-ALNS. The performance of the ALNS is crucial for the performance of the 2-ALNS, as it is the main ingredient of the 2-ALNS procedure. We therefore tuned the parameters of the ALNS by means of a grid search, as specified in Table 3.A.1.

The experiments are conducted on 90 instances with sizes varying between 20 and 60, similarly constructed as the instances in Benchmark Set H. As a measure of the ALNS' performance we take the average minimum objective value over 10 runs over all instances. The parameter values that gave the highest quality solutions are presented in Table 3.A.1. The column indicated by 'Benchmark' represents parameter settings used for benchmarking purposes in Section 4 (for all problem variants). We only considered parameter values that had an average run time of at most 20 seconds. For the simulation study in Section 5, we considered the best parameter values within a time limit of 150 seconds. They are given in the columns marked by 'TARP', 'TARP-F', and 'TARP-G'.

Table 3.A.1: Parameter calibration. Operator values $(p_1, \ldots, p_4)$ are taken equal to 1000. Shaw operator values $\hat{\alpha}, \hat{\beta}, \hat{\gamma}, \hat{\delta}$ are set equal to 5. The number of initial solution N_{init} is set equal to 1000

Parameter	Interpretation	Range	Benchmark	TARP-G	TARP-F	TARP
N^1_{iter}	Number of shaking procedure iterations	{20, 30, 50}	20	30	20	20
N^2_{iter}	Number of iterations Destroy and Repair procedure	{2000, 4000}	2000	4000	4000	4000
$n^{\text{fail}}_{\max}$	Maximum shaking intensity	{3, 5, 8}	5	8	5	8
c_{sa}	Cooling parameter of SA	{0.5, 1.0, 1.5}	1	0.5	1	0.5
w_{sa}	Weight parameter of SA	{50, 100, 150}	100	150	100	150
$(\underline{n}_d, \overline{n}_d)$	Number of destroy operators in a single destroy	{(1,2), (1,3)})	(1,2)	(1,3)	(1,3)	(1,3)
Δ	Change in operator probability after successful move	{1, 50}	1	1	50	50
$(\underline{I}_1, \overline{I}_1)$	Destroy operator class 1 intensities	$\{(1,1),(1,2),(1,3)\}$	(1,1)	(1,1)	(1,2)	(1,2)
$(\underline{I}_2, \overline{I}_2)$	Destroy operator class 2 intensities	$\{(1,2),(1,3)\}$	(1,3)	(1,3)	(1,3)	(1,3)

3.B Simulation study set-up

In this appendix, we present additional details to the simulation set-up summary as presented in Section 5. We made use of a fixed vehicle fleet as specified in Table 3.B.1. Every depot has two vessels: A relatively small though fast and cheap vessel, and a larger though slower and more expensive vessel. This is in line with practical scenarios of offshore wind as encountered in the literature (see, e.g., Stålhane, Hvattum, and Skaar (2015) and Irawan et al. (2017)).

In total 1500 scenarios are considered in the simulation study. Every scenario consists of a unique combination of instance and layout parameters, of which the considered values are denoted in Table 3.B.2.

Table 3.B.1: Overview of vessel characteristics in simulation. Distances are euclidean. Travel times are obtained by dividing travel distance with speed parameter. Travel costs are obtained by multiplying travel distance with the cost parameter.

Depot	Personnel cap.	Spare parts cap.	Speed	Cost
1	15	2000	45	225
1	25	25000	25	300
2	12	2000	45	225
2	20	25000	25	300
3	8	2000	45	225
3	16	25000	24	300

Table 3.B.2: Considered values of instance and layout parameters. For each combination of parameters 100 instances are solved as a TARP, TARP-G, and TARP-F.

Parameter	Interpretation	Considered Values
α	Distance between O&M Bases	$\{10, 20, 30, 40, 50, 100\}$
β	Distance between wind farms and O&M Bases	$\{10, 20, 30, 40, 50\}$
$\widehat{q}_\ell$	Number of technicians of each type	$\{9, 12, 15, 18, 21\}$
p	Probability of maintenance for each turbine	$\{0.08, 0.10, 0.12, 0.14, 0.16\}$
T	Length of time horizon	$\{5, 6\}$

Chapter 4

Mixed Integer Programming models for planning maintenance at offshore wind farms under uncertainty

Abstract. *We introduce the Stochastic Maintenance Fleet Transportation Problem for Offshore wind farms (SMFTPO), in which a maintenance provider determines an optimal, medium-term planning for maintaining multiple wind farms while controlling for uncertainty in the maintenance tasks and weather conditions. Since the maintenance provider is typically not the owner of a wind farm, it needs to adhere minimum service requirements that specify the required service. We consider three of such settings: 1) perform all maintenance tasks, 2) allow for a fraction of unscheduled tasks, and 3) incentivize to perform maintenance rather quickly. We provide a two-stage stochastic mixed integer programming model for the three SMFTPO settings, and solve it by means of Sample Average Approximation. In addition, we provide an overview of the, what we discovered, non-aligned modeling assumptions in the literature regarding operational decisions. By providing a series of special cases of the second-stage problem resembling the different modeling assumptions, we aim to establish a common consensus regarding the key modeling decisions to be taken in maintenance planning problems for offshore wind farms. We provide newly constructed, and publicly available, benchmark sets. We extensively compare the different SMFTPO settings and its special cases on those benchmark sets, and we show that the special case reformulations are very effective for solving the second-stage problems. In addition, we find that for particular cases, established modeling techniques result in overestimations and increased running times.*

This chapter is based on Schrotenboer, Ursavas, and Vis (2019b):
Schrotenboer AH, Ursavas E, Vis IFA, 2019b *Mixed integer programming models for maintenance planning at offshore wind farms under uncertainty. Transportation Research Part C: Emerging Technologies* In press

4.1 Introduction

Successful offshore wind maintenance service logistics requires a thought-out maintenance strategy describing service-vessel[1] utilization strategies for the medium- and long-term. Such strategies need to consider a highly stochastic, operational environment, in which the turbines' failure behavior is difficult to predict and in which weather conditions determine the wind farm's daily accessibility (Shafiee 2015, Shafiee and Sørensen 2017). In addition, misaligned objectives between the wind farm owner (i.e., profit maximization) and the maintenance service provider (i.e., cost minimization) has led to current practices where operations are streamlined by imposing *minimum service requirements* (Ferreira, Feinstein, and Barroso 2014). These are contractually binding requirements specifing the required performance of the maintenance provider. In this context, we study a stochastic maintenance planning problem for offshore wind farms, controlling for uncertain maintenance tasks and weather conditions, in which we explicitly take the viewpoint of a maintenance service provider that is subject to such minimum service requirements. We refer to this problem as the Stochastic Maintenance Fleet Transportation Problem at Offshore wind farms (SMFTPO).

When considering decisions on a tactical level as in the SMFTPO, simplifications on the underlying operational planning problem are insurmountable as the operational planning is already shown to be computationally challing in a deterministic setting (see, e.g., Irawan et al. 2017, Schrotenboer et al. 2018a, Schrotenboer, Ursavas, and Vis 2019a). However, the extent of such simplifications affects the medium-term vessel utilization and thereby long-term vessel charter strategies. A wide variety of approximations (and underlying assumptions) have been made in the literature, and we show, by means of a thorough review, that there is no common consensus on those assumptions. We present four categories of modeling decisions that are crucial for the resulting complexity and practicality of the optimization problems. We aim to provide more insight into the impact of those assumptions on the tractability of the resulting optimization problems, and consequently, how this affects the computational efficiency. This structural overview of the impact of such assumptions on the underlying optimization problem, both computationally and mathematically, will contribute to the development of a common consensus on key modeling decisions in offshore wind maintenance logistics.

Inspired by offshore wind practices observed in the Netherlands, and contrary to previous work in offshore wind maintenance service logistics (see e.g., Stålhane et al. 2016), we take the perspective of a single, large maintenance service provider

[1]For readability, we use the term 'vessel' to indicate both helicopters and vessels

that is responsible for the maintenance of one or multiple offshore wind farms. This maintenance service provider is not the owner of the wind farms and, therefore, does not bear the risk of uncertain production revenues due to the highly volatile energy prices and the risk of production losses due to downtime of the turbines. The maintenance service provider's sole priority is adhering the *minimum service requirements* specified in a service contract between the wind farm owner and the maintenance service provider, typically resulting in a misalignment of objectives between wind farm owner (production maximization) and the maintenance service provider (maintenance cost minimization). This is in contrast with the current literature, in which it is assumed that objectives are aligned and the sum of total costs will be minimized, without imposing hard constraints on the service requirements.

In this paper, we introduce the Stochastic Maintenance Fleet Transportation Problem for Offshore wind farms (SMFTPO). Its goal is to develop a cost-minimizing, medium-term maintenance planning by assigning vessels to depots (O&M bases) in the first stage, and after the uncertain maintenance tasks and weather conditions are revealed, to assign maintenance tasks to the vessels in the second stage. This is, to the best of the authors' knowledge, the first study on a tactical level of decision making in offshore wind maintenance service logistics. We consider three settings of the SMFTPO, each describing a different variant of minimum service requirements. These settings are 1) to perform all maintenance tasks, 2) to allow for a fraction of unscheduled tasks, and 3) to incentivize performing maintenance rather quickly. We provide a general two-stage stochastic mixed integer programming formulation and its scenario-based large scale representation which we solve by Sample Average Approximation (see, e.g., Kleywegt, Shapiro, and Homem-de Mello 2002, Santoso et al. 2005). The second-stage is modelled by using decomposed and time-expanded networks, which are commonly used in network design applications (see, e.g., Crainic 2000, Andersen et al. 2011) and for which sophisticated solution approaches have been developed (see, e.g., Boland et al. 2017). We provide insights in the value of the stochastic solution and the effect of the different service requirements, both from a computational and a managerial point of view.

In addition, we exemplify the impact of different decisions regarding the identified modeling categories by studying five special cases of the second-stage problem of the SMFTPO, resulting in a series of reformulations for each SMFTPO setting. The special cases are inspired on practical observations in offshore wind, on the need for further mathematical insights as identified in our literature review, or on both. The special cases elaborate on differences between single and multiple wind farm settings, maintenance task pre-processing techniques, and the level of detail of the operational

problem. A numerical investigation on the computational tractability of the special cases is performed to assess the trade-off between the level of modeling detail and the computational performance. The formulations of the special cases are shown to be very efficient in terms of computational efficiency. Moreover, it is numerically shown that pre-processing maintenance tasks into bundles (see, e.g., Gundegjerde et al. 2015) will result in an overestimation of the total incurred maintenance costs, although it is computationally attractive to do so. To foster future research in this area, we made our set of benchmark instances publicly available[2].

4.1.1 Literature Review

The SMFTPO relates to so-called fleet size and mix problems in offshore wind, which focus on strategic decision making (buying or chartering a vessel) as opposed to the tactical decision making (assigning vessels to wind farms and maintenance tasks) in the SMFTPO. Halvorsen-Weare et al. (2013) presents the first application of fleet size and mix problems in the context of offshore wind. They propose a MIP formulation, which formed the basis of the 3-stage stochastic programming approach by Gundegjerde et al. (2015). It includes stochastic vessel spot rates, weather conditions, electricity prices, and turbine failures.

Motivated by this seminal work, a number of papers have studied slight variants of this optimization problem or introduced new solution approaches.The work by Stålhane, Halvorsen-Weare, and Nonås (2016) considers a similar setting as Gundegjerde et al. (2015). By advanced preprocessing of maintenance tasks, they provide the first sophisticated solution approach in this context. Its mathematical model forms the basis of three other works. First, Stålhane et al. (2016) use the same concepts to study optimal fleet size and mix over the complete lifetime of a wind farm. Second, Gutierrez-Alcoba et al. (2017) provide additional insights by means of a sophisticated case-study. Third, Halvorsen-Weare et al. (2017) developed a metaheuristic approach in which uncertainty is assessed by means of simulation. Slightly different works are those by Stålhane et al. (2017) studying optimal jack-up vessel chartering strategies and the work of Sperstad et al. (2017) on the robustness of different decision support tools.

Since the accurateness of the modeling of operational activities such as vessel routing and short-term maintenance planning is crucial for the difficulty of fleet size and mix problems, we shortly review the most important contributions in that area. The short-term planning of maintenance tasks in offshore wind farms has become

[2]https://sites.google.com/rug.nl/albertschrotenboer

increasingly popular among researchers. A series of research articles (Dai, Stålhane, and Utne 2015, Stålhane, Hvattum, and Skaar 2015, Irawan et al. 2017, Raknes et al. 2017, Schrotenboer, Ursavas, and Vis 2019a, Schrotenboer et al. 2018a) discuss this problem, leading to a branch-and-price-and-cut method for the single wind farm case (Schrotenboer, Ursavas, and Vis 2019a), and a metaheuristic approach for the multiple wind farm case (Schrotenboer et al. 2018a).

In addition, a number of articles are written about decisions support tools for offshore wind maintenance logistics (see, e.g., Stålhane, Hvattum, and Skaar 2015, Stålhane, Halvorsen-Weare, and Nonås 2016). The interested reader is referred to the work of Hofmann (2011) for a review of papers on decisions support tools for offshore wind (maintenance) logistics. For interested readers on the installation phase of offshore wind farms, see the papers by Vis and Ursavas (2016) and Ursavas (2017). For all contributions on fleet size and mix problems in onshore logistics, we refer the interested reader to the recent review by Braekers, Ramaekers, and Van Nieuwenhuyse (2016). Finally, for all other aspects related to offshore wind maintenance logistics that are not stringently relevant to the SMFTPO, we refer the reader to the excellent review by Shafiee and Sørensen (2017).

The notion of minimum service requirements is, to the best of the authors' knowledge, not considered before in maintenance planning optimization problems focussing on offshore wind. However, it is well-known that maintenance performance should be measured and the right incentives should be given (Parida et al. 2015). Especially in more mature industries that include applications in rail (see, e.g., Lidén 2015) and power systems (see, e.g., Froger et al. 2016) the use of minimum service requirements is common. However, the impact of stochastic weather conditions that limit the maximum working hours in a period is typical for offshore wind, and can not directly be related to other applications. We, therefore, introduce three basic minimum service requirements and investigate their impact on the maintenance planning problem as discussed in this paper.

4.1.1.1 Modeling decisions and assumptions.

From the literature review it is clear that a series of research articles has been devoted to offshore wind maintenance planning problems. However, those studies are typically not aligned with regards to their modeling decisions. We discuss, what we call, four *modeling categories*, and we detail how the existing studies have taken decisions within.

First, there is a difference between maintenance tasks that take more than a single period, and less than a single day. The papers by Halvorsen-Weare et al. (2013) and Gundegjerde et al. (2015) partition the set of maintenance tasks based on their

duration (whether they take a single period or more). However, the impact of this on the computational tractability is not discussed by the authors. In Stålhane et al. (2016) and Stålhane, Halvorsen-Weare, and Nonås (2016), task durations are assumed to be at most a single period, i.e., tasks are not scheduled among multiple periods.

Second, taking the perspective of a single, large maintenance provider in offshore wind (as in the SMFTPO), the question how to jointly use multiple depots (ports or harbours) is of utmost importance. In Gundegjerde et al. (2015), they assume vessels can switch depots, however, they consider only a single wind farm in their experiments. Hence it is not clear what the impact of this assumption is. In Stålhane et al. (2016), vessels may switch depots, however as the period length resembles several months, this does not have a great influence. In Stålhane, Halvorsen-Weare, and Nonås (2016) and Halvorsen-Weare et al. (2017) it is assumed that vessels are associated with a single base.

The third modeling category is the modeling and categorization of maintenance tasks. Halvorsen-Weare et al. (2013) and Gundegjerde et al. (2015) consider every task on its own, but belonging to different categories that define whether or not a vessel is eligible to perform the maintenance task. Moreover, they assume that a task can only be completed by a single vessel. In Stålhane et al. (2016), the considered time horizon length equals the wind farm lifetime and the length of single period is in the order of several months. Tasks of the same category are assumed to demand a number of technicians, and a piecewise linear relation between vessel fleet capacity and incurred downtime costs is proposed. The same holds for, Stålhane, Halvorsen-Weare, and Nonås (2016) and Halvorsen-Weare et al. (2017) though they have a similar focus as Ahmadi-Javid and Seddighi (2012), i.e., they consider a time horizon of a year and periods reflect days. In this paper, we are especially interested in the impact of assuming that a single job is performed by one vessel only.

The final modeling category is the extent to which minimum service requirements are imposed, as we detailed for the SMFTPO in the former. Halvorsen-Weare et al. (2013) and Gundegjerde et al. (2015) consider a penalty cost for not scheduling maintenance tasks. Before solving actual instances, penalty costs need to be set so that enough tasks are completed in order to reflect practical situations. This is structurally different from the SMFTPO, as we model minimum service requirements with hard constraints. The papers by Stålhane et al. (2016), Stålhane, Halvorsen-Weare, and Nonås (2016), and Halvorsen-Weare et al. (2017) model the assigned vessel's capacity to wind farms and penalize under- and over-coverage of the expected number of needed technicians.

4.1.2 Contributions and outlook

Summarizing, four key differences of this paper with the current literature are observed. First, all the above-mentioned papers considered the wind farm as a single entity of which the total costs are to be minimized. We, however, take the viewpoint of a maintenance service provider subject to *minimum service requirements*. Such a maintenance service provider is not concerned with production losses but minimizes their own costs so that the maintenance service requirements are met. Second, all the above-mentioned papers on optimizing maintenance planning at offshore wind farms did not provide mathematical insights into the underlying MIP formulations and its relation to the observed computational performance. In this paper, we provide such insights in Section 4, by discussing a series of reformulations for special cases in offshore wind. Third, we study tactical decision making, i.e., we allocate an already existing vessel fleet to wind farms and depots instead of focusing on strategic or operational decisions, allowing us to elaborate further on those mathematical insights. Fourth, by taking decisions on a tactical level, the SMFTPO is the first model that allows vessels to be utilized from multiple depots.

The remainder of this paper is structured as follows. In Section 4.2, we provide, next to a formal problem statement, the two-stage stochastic mixed integer programming formulation and its scenario-based large-scale monolithic formulation. In Section 4.3, we present five special cases of the second-stage problem of the SMFTPO that focus on the modeling categories and assumptions discussed in Section 4.1.1.1. A numerical analysis of all the special cases and the SMFTPO is provided in Section 4.4. We conclude our work in Section 4.5, where we provide numerous avenues for further research as well.

4.2 Problem Formulation

In this section, we present the Mixed Integer Programming (MIP) formulation for the Stochastic Maintenance Fleet Transportation Problem for Offshore wind farms (SMFTPO). We first describe the system upon which the SMFTPO is based, and discuss the first and second stage decisions to take. After that, we provide a time expanded and decomposed network formulation upon which we model the second-stage decisions. We discuss three distinct second-stage optimization models each of which is tailored towards a particular setting of minimum service requirements. In the first setting, all maintenance tasks need to be scheduled. In the second setting, a fraction α of so-called technician hours can be left unscheduled. The third setting

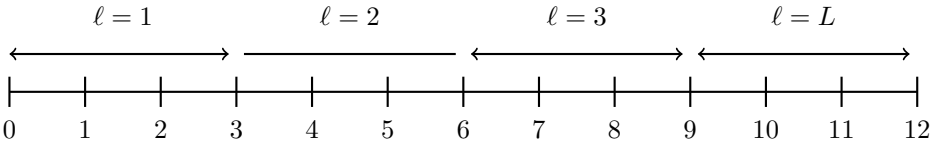


Figure 4.1: Example planning horizon with $T = 12$ periods devided into $L = 4$ leaseterms each with a length of 3 periods.

restricts the maximum fraction β of periods on which turbines are left unrepaired. Those three settings, although stylized, will provide realistic cost estimations of the medium-term maintenance planning accounting for different incentives; the setting with α unscheduled technician hours is commonly encountered and results in delaying tasks that are most unprofitable, and the setting with at most β downtime periods resembles a maintenance contract giving incentives to perform maintenance tasks rather quickly.

We end this section by providing the two-stage stochastic programming model and its scenario-based monolithic formulation. In the remainder, we denote uncertainty by the set Ξ. Dependency on this set is denoted by the general descriptor $\cdot(\xi)$, where $\xi \in \Xi$.

4.2.1 System description

An overview of the notation used for describing the system is given in Table 4.1. We consider a time horizon $\mathcal{T} = \{1, \dots, T\}$ that is partitioned in L lease terms of equal size. We refer to each $t \in \mathcal{T}$ as a period. Each period has a maximum number of working hours. The corresponding periods of each lease term $\ell \in \mathcal{L} = \{1, 2, \dots, L\}$ are given by $\mathcal{T}^\ell = \{(T/L)(\ell - 1) + 1, \dots, (T/L)\ell\}$. In Figure 4.1, an example planning horizon with $T = 12$ periods and $L = 4$ leaseterms is presented. Throughout the paper we will assume that $T/L \in \mathbb{N}$.

The set of wind farms is denoted by $\mathcal{W}$. Each wind farm $w \in \mathcal{W}$ consists of N^w turbines and is geographically represented by a single set of coordinates (X^w, Y^w). All distances within a wind farm will be ignored. Let $\mathcal{D}$ be the set of depots (O&M bases). Each depot $d \in \mathcal{D}$ is geographically placed at (X^d, Y^d). Distances between depots and wind farms are assumed to be Euclidean.

The set of stochastic maintenance tasks is denoted by $\mathcal{M}(\xi) = \{1, \dots, M(\xi)\}$. A maintenance task $m \in \mathcal{M}(\xi)$ is part of a particular wind farm F^m. The set of maintenance tasks at wind farm $w \in \mathcal{W}$ is defined as $\mathcal{M}^w(\xi) := \{m \in \mathcal{M}(\xi) : F^m = w\}$. Every maintenance task m has a number of consecutive periods $\mathcal{T}^m(\xi)$ in which it

Table 4.1: Overview of the main sets, parameters, and decision variables

Determenistic sets and parameters	
$\mathcal{T} = \{1, \dots, T\}$	The complete time horizon
$\mathcal{W} = \{1, \dots, W\}$	Set of wind farms
$\mathcal{D} = \{1, \dots, D\}$	Set of depots
L	Number of leaseterms with equal length $T/L \in \mathbb{N}$
$\mathcal{V} = \{1, \dots, V\}$	Set of vessels
S_1^v	Number of technicians on board vessel $v \in \mathcal{V}$
S_2^v	Total per period technician working hours on vessel $v \in \mathcal{V}$
$\hat{C}_{d\ell}^v$	Costs of assigning vessel v to depot d in leaseterm ℓ.
$\hat{P}$	Costs of changing depot in the second stage
$\mathcal{L}^{\text{M}}$	Set of maintenance categories
$\mathcal{L}^{\text{V}}$	Set of vessel types
$\Theta(\ell^{\text{V}})$	Maintenance categories that can be performed by vessel type ℓ^{V}
$\Theta(\ell^{\text{M}})$	Set of vessel types that can perform maintenance category ℓ^{M}
$\theta(\ell^{\text{M}}, \ell^{\text{V}})$	Equals 1 if vessel type ℓ^{V} can perform maintenance category ℓ^{M}
Stochastic sets and parameters	
$\mathcal{M}(\xi) = \{1, \dots, M(\xi)\}$	Set of maintenance tasks
$M^w(\xi)$	Number of maintenance tasks at wind farm w
$M^{wt}(\xi)$	Number of maintenance tasks at wind farm w in period t
$S_m(\xi)$	First period in which maintenance task m can be scheduled
$E_m(\xi)$	The latest period in which maintenance task m can be scheduled
$\mathcal{T}^m(\xi)$	The set of periods in which maintenance task m can be scheduled
$F^m(\xi)$	Wind farm in which maintenance task m is located
$D_1^m(\xi)$	Number of technicians demanded for maintenance task m
$D_2^m(\xi)$	Number of hours of work (for each of the demanded technicians) $v \in \mathcal{V}$
$H_{\ell^{\text{V}}}^{tw}(\xi)$	Total hours vessel type ℓ^{V} can be offshore in period t at wind farm w
Decision variables	
$y_{d\ell}^v$	1st-stage variable equalling 1 if vessel v is assigned to depot d in period ℓ.
$x_{ij}^v(\xi) \in \{0, 1\}$	2nd-stage variable whether arc $(i, j) \in \mathcal{A}^{\mathcal{T}}$ is traversed by vessel v
$z_{ij}^v(\xi)$	total technician hours send along arc $(i, j) \in \mathcal{A}^{\mathcal{T}}$ by vessel v

can be scheduled, the first period being denoted by S_m, and the latest period is denoted by E_m. We call the periods between $S_m(\xi)$ and $E_m(\xi)$ the maintenance window. Then, the set of maintenance tasks of windfarm w at period t is defined as $\mathcal{M}^{wt}(\xi) := \{m \in \mathcal{M}^w(\xi) : S_m(\xi) \leq t \leq E_m(\xi)\}$.

We consider a given fleet of vessels $\mathcal{V}$ that are deployed for performing maintenance tasks. Each vessel $v \in \mathcal{V}$ transports S_1^v technicians that can work S_2^v hours in each period. Each maintenance task requires D_1^m technicians to work for D_2^m hours to be completed. We will model the demand (resp. supply) of technicians in terms of *technician hours*: The number of technicians multiplied with the number of hours they are required (resp. available) to work. The supply of technician hours can be restricted to reflect travel time, unloading time, or severe weather conditions.

Each maintenance task m can be categorized to a maintenance category $\ell^M \in \mathcal{L}^M$,

where the set $\mathcal{L}^M = \{1^M, \ldots, L^M\}$ contains all maintenance categories. Similarly, each vessel v can be categorized to a vessel type $\ell^V \in \mathcal{L}^V$, where the set $\mathcal{L}^V = \{1^V, \ldots, L^V\}$ is the set of vessel types. A vessel cannot perform all maintenance categories. We use $\ell^M(m)$ and $\ell^V(v)$ to denote the category of a maintenance task m and the type of a vessel v, respectively. We let $\theta(\ell^M, \ell^V) = 1$ if maintenance category ℓ^M can be performed by vessel type ℓ^V. For notational convenience, we let $\Theta(\ell^m) := \{\ell^V \in \mathcal{L}^V \mid \theta(\ell^M, \ell^V) = 1\}$, and $\Theta(\ell^V) := \{\ell^M \in \mathcal{M} \mid \theta(\ell^M, \ell^V) = 1\}$. In other words, $\Theta(\ell^M) \subseteq \mathcal{L}^V$ are all the vessel types that can perform maintenance category ℓ^M, and $\Theta(\ell^V) \subseteq \mathcal{L}^M$ are all the maintenance categories that can be performed by vessel type ℓ^V.

We define $H^{wt}_{\ell^V}(\xi)$ as the number of hours that a vessel of type ℓ^V can perform maintenance tasks in period t at wind farm w. This reflects the impact of weather conditions on the daily operations. We, hereby, imply that vessels cannot change wind farms within a period. However, wind farms being close are likely to be exposed to similar weather conditions, and such wind farms will be modeled as a single wind farm.

Then, the SMFTPO is a two-stage stochastic optimization problem. In the first stage, we need to assign vessels to depots for each lease period $l \in \mathcal{L}$. Then, in the second stage, after the set of maintenance tasks and the weather conditions are revealed, we need to assign the vessels to the maintenance tasks. We allow a vessel to change depot in the second stage with a penalty cost $\hat{P}$. The first-stage decisions are modelled by binary decision variables $y^v_{d\ell}$ equalling 1 if vessel $v \in \mathcal{V}$ is assigned to depot $d \in \mathcal{D}$ in leaseterm $\ell \in \mathcal{L}$, and 0 otherwise.

4.2.2 The second stage problem

We model the second-stage problem, i.e., given a first-stage decision and after observing uncertain parameters $\xi \in \Xi$, as a network design problem on a decomposed and time-expanded network. An overview of the notation used is provided in Table 4.2. The time expansion is made on the period level, i.e., nodes encode a maintenance task in a particular period. The decomposition is made in the vessel dimension, as we assume their movements are independent. To enhance readability, we omit the dependency on $\cdot(\xi)$ in this subsection.

We first consider a flat network (i.e., no time expansion or decomposition) that forms the basis of the decomposed and time-expanded formulation. Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be this network, where the node set $\mathcal{N}$ consists of a node for each wind farm $w \in \mathcal{W}$, depot $d \in \mathcal{D}$, and maintenance task $m \in \mathcal{M}$. The arc set $\mathcal{A}$ consists of two types of

Table 4.2: Overview of node and arc sets in the decomposed and time-expanded formulation

Node Sets	
$\mathcal{G}^{\mathcal{T}}$	The time expanded network
$\mathcal{N}^{\mathcal{T}}$	Set of nodes in the time expanded graph.
$\mathcal{N}_{\mathcal{D}}^{\mathcal{T},v}$	Set of nodes representing depots (for vessel v).
$\mathcal{N}_{\mathcal{W}}^{\mathcal{T},v}$	Set of nodes representing wind farms (for vessel v).
$\mathcal{N}_{\mathcal{M}}^{\mathcal{T},v}$	Set of nodes representing maintenance tasks (for vessel v).
$\mathcal{N}_{\text{ART}}^{\mathcal{T},v}$	Set of nodes representing artificial source and sink nodes (for vessel v).
Arc Sets	
$\mathcal{A}^{\mathcal{T},v}$	Set of arcs in the time expanded graph.
$\mathcal{A}_1^{\mathcal{T},v}$	Set of arcs from depot nodes to wind farm nodes.
$\mathcal{A}_2^{\mathcal{T},v}$	Set of arcs from wind farm nodes to depot nodes.
$\mathcal{A}_3^{\mathcal{T},v}$	Set of arcs from depot nodes to depot nodes.
$\mathcal{A}_4^{\mathcal{T},v}$	Set of arcs from wind farm nodes to task nodes, and vice versa.
$\mathcal{A}_5^{\mathcal{T},v}$	Set of arcs connecting artificial (source and sink) nodes to depots, and vice versa.
$\mathcal{A}_4^{\mathcal{T},v}(m)$	Set of incoming arcs into nodes representing maintenance task m.
$\mathcal{A}_4^{\mathcal{T},v}(w)$	Set of incoming arcs into nodes representing maintenance tasks at windfarm w.

arcs:

(1) We create arcs (i,j) for each $i,j \in (\mathcal{D} \cup \mathcal{W})$, i.e., arcs between depots and reachable wind farms.

(2) We create arcs (i,w) and (w,i) for each $i \in M^w$ and for all $w \in \mathcal{W}$, i.e., arcs between maintenance tasks and their corresponding wind farms.

4.2.2.1 Decomposed and time-expanded network.

Let $\mathcal{T}$ and $v \in \mathcal{V}$ be given. The corresponding, decomposed and time expanded graph is then defined as $\mathcal{G}_v^{\mathcal{T}} = (\mathcal{N}_v^{\mathcal{T}}, \mathcal{A}_v^{\mathcal{T}})$ for each vessel $v \in \mathcal{V}$. The node set is defined as $N_v^{\mathcal{T}} := \mathcal{N}_{\mathcal{D}}^{\mathcal{T},v} \cup \mathcal{N}_{\mathcal{W}}^{\mathcal{T},v} \cup \mathcal{N}_{\mathcal{M}}^{\mathcal{T},v} \cup \mathcal{N}_{\text{ART}}^{\sqcup,v}$. Here $\mathcal{N}_{\mathcal{D}}^{\mathcal{T},v} := \{(d,t) \mid d \in \mathcal{D}, t \in \mathcal{T}\}$, $\mathcal{N}_{\mathcal{W}}^{\mathcal{T},v} := \{(w,t) \mid w \in \mathcal{W}, t \in \mathcal{T}\}$, $\mathcal{N}_{\mathcal{M}}^{\mathcal{T},v} := \{(m,t) \mid m \in \mathcal{M} : \ell^V(v) \in \Theta(\ell^M(m)), t \in (\mathcal{T}^m \cap \mathcal{T})\}$, and $\mathcal{N}_{\text{ART}}^{\mathcal{T},v} := \{\ell \mid \ell \in \mathcal{L}\}$. In other words, those sets contain node copies for each $t \in \mathcal{T}$ representing the depots, wind farms, the eligible maintenance tasks for vessel v, and artificial nodes modeling the availability of vessels.

The arc set $\mathcal{A}^{\mathcal{T},v}$ is partitioned into the sets $\mathcal{A}_1^{\mathcal{T},v}$, $\mathcal{A}_2^{\mathcal{T},v}$, $\mathcal{A}_3^{\mathcal{T},v}$, $\mathcal{A}_4^{\mathcal{T},v}$, and $\mathcal{A}_5^{\mathcal{T},v}$, which are constructed as follows:

(1) Arcs $((d,t),(w,t)) \in \mathcal{A}_1^{\mathcal{T},v}$ for each $d \in \mathcal{D}$, $j \in \mathcal{W}$, $t \in \mathcal{T}$. Note that we only consider arcs from a depot to a windfarm if it is reachable from that depot. The costs of these arcs represent the daily traveling costs.The capacity of this arc represents the realization of $H_{\ell^V}^{t\omega}$.

(2) Arcs $((w,t),(d,t+1)) \in \mathcal{A}_2^{\mathcal{T},v}$ for each $w \in \mathcal{W}, d \in \mathcal{D}, t \in \mathcal{T}\backslash T$. Here $t+1$ refers to the first period that follows t in $\mathcal{T}$. Again, we only consider wind farm to depot arcs if the wind farm can be reached from the depot. The costs represent the traveling costs.

(3) Arcs $((d,t),(d',t+1)) \in \mathcal{A}_3^{\mathcal{T},v}$ for each $d, d' \in \mathcal{D}, t \in \mathcal{T}\backslash T$. These arcs between depots represent either no maintenance (if they connect the same depot) or the recourse action that can be taken to change the allocation of the vessel to a different depot. In case of the latter, travel costs and the penalty $\hat{P}$ are incurred.

(4) Arcs $((i,t),(w,t)) \in \mathcal{A}_4^{\mathcal{T},v}$ and $((w,t),(i,t)) \in \mathcal{A}_4^{\mathcal{T}}$ for each $i \in \mathcal{M}^{wt}$, $t \in \mathcal{T}$, $w \in \mathcal{W}$. These arcs represent performing a particular maintenance task in a wind farm. The costs represent maintenance specific costs. The capacity of those arcs resemble the demand for technician hours by the maintenance task.

(5) Arcs $(\ell,(d,t'))$ and $((d,t'),\ell+1) \in \mathcal{A}_5^{\mathcal{T},v}$, with t' being the earliest period t in lease period ℓ and t'' the latest period t in lease period ℓ, for all $\ell \in \mathcal{L}\backslash L$. These arcs model the inflow of vessels in a lease term, as depicted by the first-stage solution. The costs are already included in the first-stage decision. The capacity of this arc equals the supply of technician hours of vessel type ℓ^{V}

Example 4.1. An illustrative example of a time-expanded network $\mathcal{G}_v^{\mathcal{T}}$ for an arbitrary vessel $v \in \mathcal{V}$ is presented in Figure 4.2. Using this graph, we can model the second-stage problem of the SMFTPO as a network design problem. In the example, we included two wind farms 'wf1' and 'wf2', two depots 'D1' and 'D2'. In this particular example, wf1 is only reachable from D1, and wf2 is only reachable from D2. On the left, the node 'ART' is a artificial node acting as source (and sink) of the vehicle flow. In the example, six jobs 'j1'-'j6' are depicted. It is seen that jobs 'j1'-'j3' are located in the first wind farm, and the remaning jobs in the second wind farm. Each job is only present in its maintenance window, e.g., job six is not present in period 2. Finally, note the (red) arcs between the depots, which model either a period in which no maintenance is performed (an arc between the same depot) or a period where the vessel changes depot and incures the penalty $\hat{P}$. ◁

4.2.2.2 Second-stage mixed integer programming formulation.

We can model the second-stage problem of the SMFTPO on the decomposed and time-expanded networks $\mathcal{G}^{\mathcal{T}}$. Let x_{ij}^v be a binary decision variable equaling 1 if arc $(i,j) \in \mathcal{A}^{\mathcal{T},v}$ is traversed by vessel $v \in \mathcal{V}$. Let z_{ij}^v be a continuous decision variable

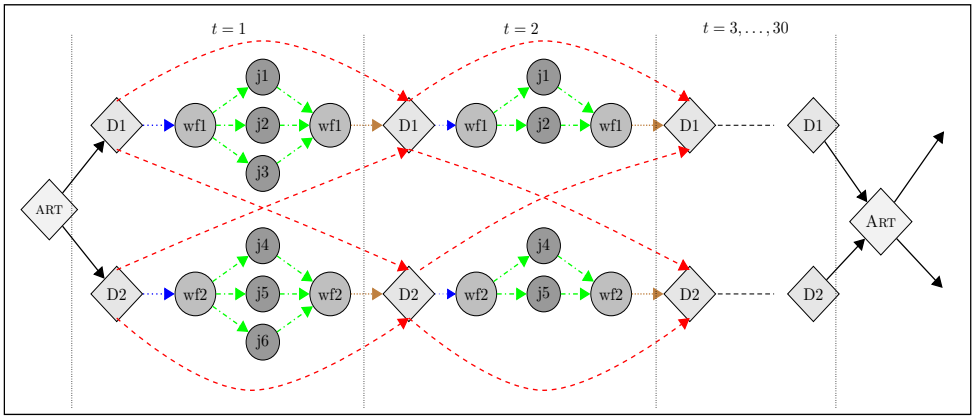


Figure 4.2: (Color online) Graph $G_v^{\mathcal{T}}$ corresponding to Example 4.1 with in blue (and dotted) the set $\mathcal{A}_1^{\mathcal{T},v}$, in brown (and densily dotted) the set $\mathcal{A}_2^{\mathcal{T},v}$, in red (and dashed) the set $\mathcal{A}_3^{\mathcal{T}}$, in green (and dashdotted) the set $\mathcal{A}_4^{\mathcal{T},v}$, and in black (and solid) the set $\mathcal{A}_5^{\mathcal{T},v}$. For illustrative purposes, only a single vessel is included in the example

indicating the number of technician hours sent along arc $(i,j) \in \mathcal{A}_v^{\mathcal{T}}$. For each arc $(i,j) \in \mathcal{A}_v^{\mathcal{T}}$, we let U_{ij}^v be the total number of technician hours (or capacity) that can be sent along (i,j) with vessel v, as described in the former.

We define C_{ij}^v as the costs of traversing arc (i,j) and we let F_{ij}^v be the task-specific maintenance cost per supplied technician hour. Note that these costs are exogenously given and might be used to make a distinction between corrective or preventive maintenance costs. We elaborate more on the actual construction of the arc capacities and costs in the numerical results in Section 4.4.

Some additional notation is required in order to obtain a concise formulation. Let $\delta^+(S) := \{(i,j) \in \mathcal{A}_v^{\mathcal{T}} \mid i \in S, j \notin S\}$ and $\delta^-(S) := \{(i,j) \in \mathcal{A}_v^{\mathcal{T}} \mid i \notin S, j \in S\}$ for any $S \subseteq N_v^{\mathcal{T}}$. In addition, we denote with δ^n the difference in incoming and outgoing technician hours. This equals 0 for each node except the artificial nodes $\mathcal{N}_{\text{ART}}^{\mathcal{T},v}$, in which δ^n models the availability of vessels and their corresponding supply of technician hours (i.e., the first-stage decision). Finally, we refer to the complete vectors of decision variables by denoting them in bold, e.g., with $\mathbf{y}$ we denote $y_{d\ell}^v$ for all $d \in \mathcal{D}, \ell \in \mathcal{L}$, and $v \in \mathcal{V}$. Then, the second-stage problem of the SMFTPO asks for solving

$$Q(\mathbf{x}, \mathbf{z} \mid \xi) := \min \quad \sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}^{\mathcal{T},v}} C_{ij}^v x_{ij}^v + \sum_{v \in \mathcal{V}} \sum_{m \in \mathcal{M}} \sum_{(i,j) \in \mathcal{A}_4^{\mathcal{T},v}(m)} F_{ij}^v z_{ij}^v \tag{4.1}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(n)} x_{ij}^v \leq 1 \qquad \forall\, n \in \mathcal{N}_{\mathcal{D}}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.2}$$

$$\sum_{(i,j)\in\delta^-(n)} x_{ij}^v \leq 1 \qquad \forall n \in \mathcal{N}_{\mathcal{D}}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.3}$$

$$\sum_{(i,j)\in\delta^-(n)} z_{ij}^v - \sum_{(i,j)\in\delta^+(n)} z_{ij}^v = \delta^n \qquad \forall n \in \mathcal{N}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.4}$$

$$z_{ij}^v \leq U_{ij}^v x_{ij}^v \qquad \forall (i,j) \in \mathcal{A}^{\mathcal{T},v} \tag{4.5}$$

$$\sum_{v\in\mathcal{V}} \sum_{(i,j)\in\mathcal{A}_4^{\mathcal{T},v}(m)} z_{i,j} \geq D_1^m D_2^m \qquad \forall m \in \mathcal{M} \tag{4.6}$$

$$x_{ij}^v \in \{0,1\},\ z_{ij}^v \in \mathbb{R}_+ \qquad \forall\ (i,j) \in \mathcal{A}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.7}$$

The Objective (4.1) minimizes the costs over the arcs in the network. Constraints (4.2) and (4.3) ensure that a vessel cannot be split to multiple wind farms in the same period and that it returns to a single depot, respectively. Constraints (4.4) ensure the flow conservation of technicians, and Constraints (4.5) ensure that only a positive flow of technician hours can be sent along an arc if it is traversed. Constraints (4.6) ensure that every task is being performed and the variable domains are denoted by (4.7).

Example 4.2. In Figure 4.3, we provide two feasible example flows of technician hours (of vessel v) through the same graph as in Example 4.1. In black, we depict a solution that visits windfarm 1 and works on job 1 and job 3 in period 1, and switch depots in period 2. In blue, we work in wind farm 2 on job 6 in period 1, and switch depots in period 2. ◁

4.2.2.3 The three SMFTPO settings.

The minimum service requirements, as modeled by Constraints (4.6), impose that all the tasks should be scheduled within their imposed time windows. *This is the first setting* of the SMFTPO. However, this might not reflect the practical incentives of a maintenance service provider that does not bear any risk of incurred downtime costs.

In the second setting, we allow that a fraction α^w of demanded technician hours in wind farm $w \in \mathcal{W}$ is left unassigned. The values of α^w are typically small (e.g., 0.10 or 0.05). We model this as follows: The total supply of technician hours to maintenance tasks should be at least $(1-\alpha^w)$ times the total demand of technician hours of all the maintenance tasks. The second setting asks for solving

$$Q^\alpha(\mathbf{x},\mathbf{z} \mid \xi) := \min \quad \sum_{v\in\mathcal{V}} \sum_{(i,j)\in\mathcal{A}^{\mathcal{T},v}} C_{ij}^v x_{ij}^v + \sum_{v\in\mathcal{V}} \sum_{m\in\mathcal{M}} \sum_{(i,j)\in\mathcal{A}_4^{\mathcal{T},v}(m)} F_{ij}^v z_{ij}^v \tag{4.8}$$

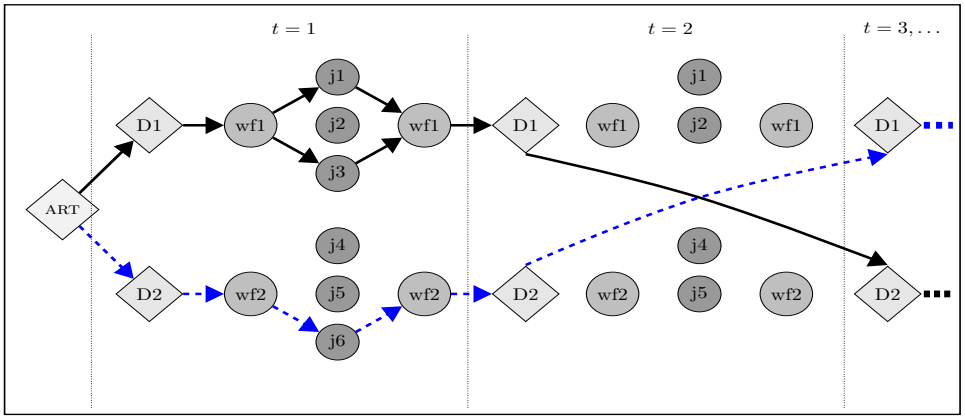


Figure 4.3: (Color online) Illustration accompanying Example 4.2 that shows two feasible flows (a dashed and blue flow, and a solid black flow) of technician hours through the time expanded network. For illustrative purposes, only a single vessel is included.

$$\text{s.t.} \quad \sum_{v \in V} \sum_{(i,j) \in \mathcal{A}_4^{\mathcal{T},v}(w)} z_{ij}^v \geq (1 - \alpha^w) \sum_{w \in \mathcal{M}^w} D_1^w D_2^w \qquad \forall\, w \in \mathcal{W} \qquad (4.9)$$

Constraints (4.2) - (4.5), (4.7)

Constraints (4.9) ensure that, for each wind farm w, the fraction of technician hours supplied by the vessels is at least $(1 - \alpha^w)$ of the total number of required technician hours. Constraints (4.2) - (4.5), and (4.7) are unchanged, as no other modeling restrictions need to be taken into account.

In the third setting, we consider a maximum fraction β^w of so-called downtime periods. These are defined as the difference between the first possible period of scheduling a task and the latest period in which a task is actually performed. We introduce the variable η_m, being equal to the latest period in which task m is scheduled. This is required since tasks could be split up among different periods and vessels. The third setting of the SMFTPO asks for solving

$$Q^\beta(\mathbf{x}, \mathbf{z}, \boldsymbol{\eta} \mid \xi) := \min \quad \sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}^{\mathcal{T},v}} C_{ij}^v x_{ij}^v + \sum_{v \in \mathcal{V}} \sum_{m \in \mathcal{M}} \sum_{(i,j) \in \mathcal{A}_4^{\mathcal{T},v}(m)} F_{ij}^v z_{ij}^v \qquad (4.10)$$

$$\text{s.t.} \quad \sum_{m \in \mathcal{M}^w} \eta_m - S_m \leq \beta^w N^w T \qquad \forall w \in \mathcal{W} \tag{4.11}$$

$$\eta_m \geq t x_{ij}^v \qquad \forall\ (i,j) \in \mathcal{A}_4^{\{t\},v}(m), m \in M, t \in \mathcal{T}, v \in \mathcal{V} \tag{4.12}$$

$$S_m \leq \eta_m \leq E_m \qquad \forall\ m \in \mathcal{M} \tag{4.13}$$

$$\eta_m \in \mathbb{R}_+ \qquad \forall\ m \in \mathcal{M} \tag{4.14}$$

Constraints (4.2) - (4.7)

Constraints (4.11) ensure that the number of downtime periods is, for each windfarm w, at most β^w times the maximum number of production periods. Constraints (4.12) ensure that the variables η_m are equal to the last period in which maintenance task m is scheduled, and Constraints (4.13) impose a trivial upper and lower bound on the variables η_m. Finally, Constraints (4.14) indicate that η_m is a continuous variable.

4.2.3 Two-stage stochastic programming formulation

In order to provide a concise two-stage mixed integer stochastic programming formulation for each of the three SMFTPO variants, we introduce the following notation. We gather feasibility of the second stage variables $\mathbf{x}, \mathbf{z}, \boldsymbol{\eta}$ in the feasibility sets X, Z so that we can denote $\mathbf{x} \in X$ to ensure feasibility of the second stage decision. Similar notation is used for $\mathbf{z}$ and $\boldsymbol{\eta}$. Let $\phi_{ij}^{dv\ell}$ be equal to 1 if arc $(i,j) \in \mathcal{A}^{\mathcal{T},v}$ connects the the artifical source node at lease term ℓ to depot d. Then, the SMFTPO asks for solving

$$z := \min \quad \sum_{d \in \mathcal{D}} \sum_{\ell \in \mathcal{L}} \sum_{v \in \mathcal{V}} \hat{C}_{d\ell}^v y_{d\ell}^v + E_\xi[Q(\mathbf{x}, \mathbf{z}, \xi)] \tag{4.15}$$

$$\text{s.t.} \quad \sum_{d \in \mathcal{D}} y_{d\ell}^v \leq 1 \qquad \forall \ell \in \mathcal{L}, v \in \mathcal{V} \tag{4.16}$$

$$\sum_{(i,j) \in \mathcal{A}_5^{\mathcal{T},v}} y_{ij}^v \phi_{ij}^{d\ell} - x_{d\ell}^v \leq 0 \qquad \forall d \in \mathcal{D}, v \in \mathcal{V}, \ell \in \mathcal{L} \tag{4.17}$$

$$\mathbf{x} \in Y, \mathbf{z} \in Z \tag{4.18}$$

Here, Objective (4.15) minimizes the sum of first-stage vessel assignments and the expected costs of the second-stage decisions. Constraints (4.16) ensure that a vessel is assigned at most once every period. Constraints (4.17) are the non-anticipativity constraints linking the first and second stage decisions. Finally, with Constraints (4.18) we indicate second-stage feasibility of $\mathbf{x}$ and $\mathbf{z}$.

The above formulation is for the first setting of the SMFTPO, i.e., when all maintenance tasks need to be scheduled. Replacing $Q(\mathbf{x}, \mathbf{z}, \xi)$ with $Q^{\alpha}(\mathbf{x}, \mathbf{z}, \xi)$ or $Q^{\beta}(\mathbf{x}, \mathbf{z}, \boldsymbol{\eta}, \xi)$ results in the second SMFTPO setting in wich we can leave a fraction α^w of demanded technician hours unscheduled (denoted by z^{α}) and the third SMFTPO setting where the fraction of downtime periods is at most β (denoted by z^{β}), respectively.

4.2.3.1 Monolithic formulation by scenario generation.

Instead of directly working with the two-stage stochastic mixed integer programming formulation above, we will consider set of generated scenarios $\overline{\Xi}$ of size N and its accompanying monolithic formulation. This formulation can directly be solved with commercial MIP solvers and is provided in Appendix A.

We will employ a Sample Average Approximation (SAA) approach to solve this monolithic formulation (see, e.g., Kleywegt, Shapiro, and Homem-de Mello 2002, Santoso et al. 2005). In order to detail this procedure, we write solving the scenario-based formulation of the SFTMPO as

$$z^{\mathrm{SAA}} := \min_{\mathbf{y} \in Y} \left[\hat{\mathbf{C}}\mathbf{y} + \sum_{\xi \in \overline{\Xi}} \frac{1}{N} \left[Q(\mathbf{x}^{\xi}, \mathbf{z}^{\xi}, \xi \mid \mathbf{x}^{\xi} \in X^{\xi}, \mathbf{z}^{\xi} \in Z^{\xi}) \right] \right] \tag{4.19}$$

Here, the superscript ξ in the second-stage decision variables $\mathbf{x}^{\xi}$ and $\mathbf{z}^{\xi}$ denotes that we explicitly define the variables for each scenario $\xi \in \overline{\Xi}$ (see also appendix A).

Then, SAA consists of the following two steps:

1. We take M samples of N scenarios. Let z_i^{SAA} denote the solution of the monolithic MIP imposed by the N scenarios in sample $i \leq M$. Then $\widehat{LB} := \frac{1}{M} \sum_{i=1}^{M} z_i^{\mathrm{SAA}}$ is an estimation of a lower bound on an optimal solution of the two-stage stochastic mixed integer program.

2. Let $j := \arg\min_i z_i^{\mathrm{SAA}}$ the sample with the lowest objective value. Let $\hat{\mathbf{y}}$ be the corresponding first-stage solution. To estimate an upper bound, we take this first-stage solution and evaluate it on M' scenarios. That is, for each scenario ξ^i, $1 \leq i \leq M'$, we calculate $z_{\xi^i}^{\mathrm{SAA}} = \hat{\mathbf{C}}\hat{\mathbf{y}} + Q(\mathbf{x}^{\xi^i}, \mathbf{z}^{\xi^i}, \xi^i \mid \hat{\mathbf{y}})$. Then, we estimate an upper bound for the two-stage stochastic mixed integer program by $\widehat{UB} := \frac{1}{M'} \sum_{i=1}^{M'} z_{\xi^{\hat{y}}}^{\mathrm{SAA}}$.

Once again, we can replace $Q(\cdot)$ with $Q^{\alpha}(\cdot)$ or $Q^{\beta}(\cdot)$ to solve the monolithic formulations of the second and third setting of the SMFTPO, respectively.

Table 4.1: Overview of special cases and relation to modeling categories. Note that Special Case IV equals the SMFTPO for a single scenario ξ, $L = 1$ and $\hat{P} = 0$

	1) task dur.	2) vessel all.	3) maintenance mod.	4) serv. req.
Special Case I	≥ 1 period	1 depot	free	all, α, β
Special Case II	≥ 1 period	1 depot	no split jobs	all, α, β
Special Case III	≤ 1 period	1 depot	no split jobs & bundles	all, α, β
Special Case IV	≥ 1 period	≥ 2 depots	free	all, α, β
Special Case V	≤ 1 period	≥ 2 depots	no split jobs & bundles	all, α, β

4.3 Modeling decisions for the second-stage problem

In this section, we present a series of reformulation for the second-stage problem of the SMFTPO (for each setting) by imposing additional assumptions. We refer to these formulations as special cases for the second-stage problem of the SMFTPO, or in short *special cases*. As stated in Section 4.1.1.1, we identified four *modeling categories* that are both relevant from an optimization perspective and a practical point of view. These modeling categories are (1) the duration of maintenance tasks, (2) the allocation of vessels to depots, (3) the modeling and categorization of maintenance tasks, and (4) the induced minimum service requirements.

Table 4.1 provides an overview of the special cases and the corresponding modeling decisions. With regards to the task duration ('task dur.'), a distinction is made based on wether they take more than 1 period, or not. A preprocessing step is required that splits the tasks that take more than 1 period into tasks that take at most a single day. The vessel allocation ('vessel all.') indicates if chartered vessels are allowed to change depots in their lease term. Special Cases I-III are single windfarm, single depot cases, and inherently assume it is not possible to change depots. Special Cases IV and V consider multiple depots, and we allow vessels to change depots during the lease term. The maintenance modeling('maintenance mod.') has two distinct assumptions that we investigate. First, whether jobs can be completed by more than a single vessel (referred to as 'free'), and second, whether the preprocessing step is taken (referred to as 'bundles'). Finally, we specify the special cases for each of the service requirements('serv. req.') as discussed in Section 4.2, which we refer to as 'all', 'α', and 'β'. Note that this leads us to fifteen distinct models (fifteen special cases with each three different service requirements).

In the following, we describe these special cases, show how they reduce to classic problems in the field of Operations Research, and provide an outlook of their com-

putational efficiency. Note that, without additional assumptions, maintenance tasks might take any amount of time, vessels are allowed to use all the depots available, and all the maintenance tasks are treated individually.

Finally, to not distract the reader from our goal to highlight the impact of different modeling assumptions and to keep the exposition concise, we will assume that there are no costs attached for assigning vesssels to depots for the special cases. Therefore, we will assume that the complete planning horizon is comprised of a single leaseterm for the special cases and that $\hat{P} = 0$.

4.3.1 Special Case I: Single wind farm

In the case of a single depot, there is no need to track the vessels' depot and the vessels' wind farm location over time. Hence, constraints ensuring that vessels can only depart from, and work in, a single wind farm or depot are abundant. The analysis in the following will also hold for multiple wind farms when vessels cannot change depot, since, then, the problem can be decomposed in a straightforward way.

Assuming a single wind farm allows the reformulation of the second-stage problem of the SMFTPO into (variants of) the *capacitated facility location problem*. In facility-location terms, we need to open capacitated facilities (a vessel traveling to the wind farm in a particular period), and assign customers' demand (technician hours) to the opened facility. Let $\hat{y}^v_{tm}$ be the fraction of the demanded technician hours of task m fulfilled by vessel v in period t. Let u^v_t be a binary decision variable whether vessel v is deployed in period t. Let U^v_t be the total technician hours that can be supplied from vessel v in period t. Let C_{tm} be the costs of maintaining task m in period t, and let G^v_t be the fixed vessel deployment costs.

We need to impose upper bounds $\overline{Y}^v_{tm}$ on the variable y^v_{tm} in the following way. First, if task m does not exist in period t, we set $\overline{Y}^v_{tm} = 0$ for all $v \in \mathcal{V}$. Second, the fraction $(D^1_m S^1_v)/(D^1_m D^2_m)$ indicates the maximum fraction of technician hours that can be sent from vessel v to task m in period t. This ensures, for example, that if a task demands 3 technicians for 16 hours, no supply of 48 (3×16) technician hours in a single period can be sent to satisfy the tasks' demand. Hence we set $\overline{Y}^v_{tm} = \min\{1, (D^1_m S^2_v)/(D^1_m D^2_m)\}$. Notice that this still allows for multiple vessels supplying a particular task with more technicians than possible in a single period. However, this is typically not observed in the resulting optimal solutions due to the set-up costs of visiting a wind farm, and we, therefore, do not include additional

constraints to cover those scenarios. Then, this special case is solved by finding

$$Q_{\text{F1}}(\hat{\mathbf{y}}, \mathbf{u} \mid \xi) := \min \quad \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} \sum_{v\in\mathcal{V}} C_{tm}\hat{y}^v_{tm} + \sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{T}} G^v_t u^v_t \tag{4.20}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}} D^m_1 D^m_2 \hat{y}^v_{tm} \leq U^v_t u^v_t \qquad \forall\ t \in \mathcal{T},\ v \in \mathcal{V} \tag{4.21}$$

$$\sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{T}} \hat{y}^v_{tm} \geq 1 \qquad \forall\ m \in \mathcal{M} \tag{4.22}$$

$$0 \leq \hat{y}^v_{tm} \leq \overline{Y}^v_{tm}, u^v_t \in \{0,1\} \qquad \forall\ t \in \mathcal{T}, m \in \mathcal{M}, v \in \mathcal{V} \tag{4.23}$$

Objective (4.20) minimizes the fixed costs of deploying a vessel and the task specific costs. Constraints (4.21) ensure that no more technicians are deployed than possible in each period. Constraints (4.22) ensure that all tasks are performed, and Constraints (4.23) indicate the domain of the decision variables. The formulations $Q^{\alpha}_{\text{F1}}(\hat{\mathbf{y}}, \mathbf{u} \mid \xi)$ and $Q^{\beta}_{\text{F1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\eta}^1, \boldsymbol{\eta}^2 \mid \xi)$ are similar as discussed in Section 4.2, i.e., they model the minimum service requirements in which at most α unscheduled technician hours are allowed and in which the fraction of downtime periods is at most β, respectively. To keep our exposition concise, we provide these formulations in Appendix 4.B.

4.3.2 Special Case II: Single wind farm and dedicated vessels

An often encountered constraint is that a single maintenance task should completely be performed by a single vessel. We refer to this assumption as 'dedicated vessels'. This implies that the we need additional binary variables λ^v_m equaling 1 if vessel $v \in \mathcal{V}$ is assigned to task m. This special case then reduces to solving

$$Q_{\text{F1-V1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\lambda} \mid \xi) := \min \quad \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} \sum_{v\in\mathcal{V}} \hat{y}^v_{tm} C_{tm} + G^v_t u^v_t \tag{4.24}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}} D^m_1 D^m_2 \hat{y}^v_{tm} \leq U^v_t u^v_t \qquad \forall\ t \in \mathcal{T} v \in \mathcal{V} \tag{4.25}$$

$$\sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{T}} \hat{y}^v_{tm} \geq 1 \qquad \forall\ m \in \mathcal{M} \tag{4.26}$$

$$\sum_{t\in\mathcal{T}} \hat{y}^v_{tm} \leq \lambda^v_m \qquad \forall\ m \in \mathcal{M}, v \in \mathcal{V} \tag{4.27}$$

$$\sum_{v\in\mathcal{V}} \lambda^v_m \leq 1 \qquad \forall\ m \in \mathcal{M} \tag{4.28}$$

$$0 \leq \hat{y}^v_{tm} \leq \overline{Y}^v_{tm},\ \lambda^v_t, u^v_m \in \{0,1\} \qquad \forall\, t \in \mathcal{T}, m \in \mathcal{M}, v \in \mathcal{V} \tag{4.29}$$

Constraints (4.27) and (4.28) ensure that each maintenance task is only performed by a single vessel. Hence, finding $Q_{\text{F1-V1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\lambda} \mid \xi)$ is equal to solving a facility location problem in which demand of a 'customer' can only be assigned to a, to be determined, subset of opened facilities. Similar to the previous exposition, the models for this special case with the generalized minimum service requirements ($Q^{\alpha}_{\text{F1-V1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\lambda} \mid \xi)$ and $Q^{\beta}_{\text{F1-V1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\eta}^1, \boldsymbol{\eta}^2 \mid \xi)$) are provided in Appendix 4.B.

4.3.3 Special Case III: Single wind, dedicated vessels, and bundle preprocessing

We can preprocess the maintenance tasks such that each vessel is assigned to a so-called bundle of tasks in every period (Gundegjerde et al. 2015). These bundles are sets of maintenance tasks that will be performed on a single day. Inherently, it is assumed that tasks will take less than a period. One could argue that tasks can be split up into smaller pieces that fit into a period, a strategy we will follow in our numerical analysis in Section 5. The major concern is the number of bundles being generated. For the z_{basic} variants one can assume that each period contains at most ϕ tasks, the total number of bundles will be approximately $V \sum_{i=1}^{\phi} \binom{M^{vt}}{i}$. The number of bundles might be reduced in two ways. First one could provide sophisticated enumeration algorithms in which particular bundle compositions are suboptimal. Second, one could use a dynamic discretization discovery algorithm (Boland et al. 2017) which iteratively enlarges the set $\mathcal{T}$.

For the single wind farm case, with bundle preprocessing, we need to solve a set covering problem, i.e., one needs to assign preprocessed task bundles to a vessel in a particular period. We only need a single set of variables, as technician hours cannot be split between tasks as in the previous models. We let $\mathcal{B}$ the set of all possible bundles of maintenance tasks. The binary parameter ϕ^m_b equals 1 if maintenance task m is contained in bundle $b \in \mathcal{B}$. Other notation for bundles $b \in \mathcal{B}$ is straightforwardly obtained from the notation on maintenance tasks: D^b_1 and D^b_2 denote the demanded technicians and the demand working hours of those technicians for bundle $b \in \mathcal{B}$, respectively. Binary variables $\tilde{y}^v_{tb}$ indicate whether vessel v in period t is assigned to bundle b. The upper bounds on the $\tilde{y}^v_{tb}$ variables are similarly obtained as their task counterpart. Then, this variant asks for solving:

$$Q_{\text{F1-B}}(\tilde{\mathbf{y}} \mid \xi) := \min \quad \sum_{t \in \mathcal{T}} \sum_{b \in \mathcal{B}} \sum_{v \in \mathcal{V}} \tilde{y}^v_{tb} C^v_{tb} \tag{4.30}$$

$$\text{s.t.} \quad \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} \tilde{y}^v_{tb} \phi^m_b \geq 1 \qquad \forall\, b \in \mathcal{B} \tag{4.31}$$

$$\sum_{b \in \mathcal{B}} \tilde{y}^v_{tb} \leq 1 \qquad \forall\, t \in \mathcal{T}, v \in \mathcal{V} \tag{4.32}$$

$$0 \leq \tilde{y}^v_{tb} \leq \overline{Y}^v_{tb} \qquad \forall\, t \in \mathcal{T}, v \in \mathcal{V}, b \in \mathcal{B} \tag{4.33}$$

The Objective (4.30) minimizes the costs of assigning bundles to vessels, and Constraints (4.31) ensure that each maintenance task is being completed. The generalizations to the other minimum service requirements ($Q^{\alpha}_{\text{F1-B}}(\tilde{\mathbf{y}} \mid \xi)$ and $Q^{\beta}_{\text{F1-B}}(\tilde{\mathbf{y}} \mid \xi)$) are provided in Appendix 4.B. The above formulation classifies as a traditional set-covering formulation.

4.3.4 Special Cases IV and V: Multiple farms and bundle preprocessing

We refer to the second-stage problem of the SMFTPO (see (4.1) - (4.14)) with $\hat{P} = 0$ and $L = 1$ as Special Case IV. We like to impose the idea of bundle preprocessing on the general second-stage problem of the SFTMPO, and refer to that as Special Case V.

To consider job bundles instead of individual tasks, we need to merge the task nodes, as illustrated in Figure 4.2, into sets of tasks. We thereby implicitly assume that only complete tasks are part of a bundle, otherwise the number of generated bundles becomes too large, or one should allow a separate flow for each task in the bundle thereby undoing the whole benefit of introducing bundles. We do so by preprocessing the maintenance tasks in tasks of at most a single day, see Section 4.4.

Let us revise some of the notation. First, we disregard the flow variables z^v_{ij}, since selecting an arc entering a bundle immediately implies performing all the tasks within the bundle. This implies that the flow conservation (i.e., Constraints (4.4)) are modelled in terms of the x^v_{ij} variables. Moreover, we assume that the task-specific maintenance costs F^v_{ij} are incorporated in the C^v_{ij}, as we only have binary decision variables. Let $\mathcal{A}^{\mathcal{T},v}_4(b)$ be the set of incoming arcs of bundle $b \in \mathcal{B}$. Then the basic formulation with bundle preprocessing reduces to finding

$$Q_{\text{B}}(\mathbf{x} \mid \xi) := \min \quad \sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}^{\mathcal{T},v}} C^v_{ij} x^v_{ij} \tag{4.34}$$

$$\text{s.t.} \quad \sum_{(i,j) \in \delta^+(n)} x^v_{ij} \leq 1 \qquad \forall\, n \in \mathcal{N}^{\mathcal{T},v}_{\mathcal{D}}, v \in \mathcal{V} \tag{4.35}$$

$$\sum_{(i,j)\in\delta^-(n)} x_{ij}^v \leq 1 \qquad \forall n \in \mathcal{N}_{\mathcal{D}}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.36}$$

$$\sum_{(i,j)\in\delta^-(n)} x_{ij}^v - \sum_{(i,j)\in\delta^+(n)} x_{ij}^v = \delta^n \qquad \forall n \in \mathcal{N}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.37}$$

$$\sum_{v\in\mathcal{V}} \sum_{(i,j)\in\mathcal{A}_4^{\mathcal{T},v}(b)} x_{ij}^v \phi_b^m \geq 1 \qquad \forall m \in \mathcal{M} \tag{4.38}$$

$$x_{ij}^v \in \{0,1\} \qquad \forall\ (i,j) \in \mathcal{A}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.39}$$

Objective (4.34) minimizes the costs of traveling along edges in the network. Constraints (4.35) and (4.36) ensure that vessels cannot be split between different depots and wind farms. Constraints (4.37) ensure that flow is conserved in all nodes but the source and sink nodes. Constraints (4.38) ensure that all tasks are being performed, and Constraints (4.39) indicate the domain of the $\mathbf{x}$ variables.

As in the previous expositions, the formulations for the other minimum service requirements ($Q_{\mathrm{B}}^{\alpha}(\mathbf{x} \mid \xi)$ and $Q_{\mathrm{B}}^{\beta}(\mathbf{x} \mid \xi)$) are provided in Appendix 4.B.

4.4 Numerical Results

The goal of this section is twofold. First, we provide a numerical analysis of the second-stage special cases of the SMFTPO as discussed in Section 4.3. We assess their computational tractability on a set of newly created benchmark instances. Second, we provide an analysis of the three different settings of the SMFTPO by solving the two-stage stochastic optimization models, using the the monolithic formulations and an SAA approach.

In the following, we first describe how we constructed the benchmark instances for the second-stage models. Afterward, we show the performance of the different special cases on the benchmark instances. We analyze the proposed solutions by the different formulations and assess which formulations are the most suitable for particular offshore wind scenarios. Then, we discuss the implications for managers and scientists in the field of offshore wind maintenance service logistics, in which we focus on how the identified modeling categories (Sections 4.1.1.1 and 4.3) relate to the presented results. Then, we discuss the benchmark instances used for testing the two-stage stochastic models, and we finally present results on that.

4.4.1 Benchmark instances for the second-stage models

The newly constructed benchmark instances are inspired on the works of Gundegjerde et al. (2015) and Stålhane, Halvorsen-Weare, and Nonås (2016). In addition, information and knowledge gathered from industry partners participating in our research project on "Sustainable service logistics for offshore wind farms"[3] is used as well. The analysis consists of two parts, one for comparing the different models and assumptions for the single wind farm case (Benchmark Set A - Special Cases I-III), and one for comparing it in the context of multiple wind farms (Benchmark Set B - Special Cases IV-V).

Within the benchmark sets, the instances differ in the number of turbines in the wind farm(s) and in the total number of periods. Three instances are randomly constructed, as described below, for each combination of the number of wind farms and the number of periods. This results in 30 instances for Benchmark Set A and 24 instances for Benchmark Set B. See Table 4.2 and 4.3 for an overview of the instances and the corresponding solutions.

The maintenance tasks are generated as follows. We consider five different maintenance categories: (i) small preventive maintenance, requiring 2 - 4 technicians for 2 - 12 hours; (ii) large preventive maintenance, requiring 2 - 6 technicians for 12 - 24 hours; (iii) small corrective maintenance, requiring 2 - 3 technicians for 2 - 6 hours; (iv) severe corrective maintenance, requiring 3 - 5 technicians for 12 - 36 hours; (v) Lifting operations, requiring 3 technicians for 2 hours. Each turbine has a tuple $p =< p_1, p_2, p_3, p_4, p_5 >$ denoting the probability of a maintenance event of each type. Technician demand and required hours are then drawn uniformly from the intervals as specified in the former. We ensure that the number of lifting operations is smaller or equal to the number maintenance tasks performed. We ignore any precedence relations between maintenance tasks (see, e.g., Gundegjerde et al. 2015).

An overview of the considered transportation modes is provided in Table 4.1. We assume all the transportation modes are available in each period. Although varying fleets over time can be handled by the special cases, we choose to keep the fleet composition constant to not overly complicate the analysis.

Weather conditions limit the vessel's daily availability. For this, historical data (see, e.g., uit het Broek et al. 2019) is used similar to Gundegjerde et al. (2015). A Weibull distribution with shape 2.17 and scale $1.128\overline{\varphi}$, where $\overline{\varphi}$ is the mean wind speed of an arbitrarily drawn month from the historical data, is used to generate wind speeds for all the periods. We assume all the drawings are independent, similar to

[3] https://www.rug.nl/cope/projecten/servicelogistiek-windmolens

Table 4.1: Characteristics of the vessels present in the special case experiments. Speeds and fuel cost are in unit distances.

Type	# tech (S_v^1)	work. hours (S_v^2)*	Travel speed	Fuel cost	max wind
Small CTV	10	10 h	50	20	20 m/s
Large CTV	15	12 h	35	25	25 m/s
Helicopter	4	10 h	200	30	20 m/s
Supply Vessel	20	12 h	35	20	35 m/s

Gundegjerde et al. (2015).

The vessels' daily working hours are affected by the observed wind speed φ_t in period t in the following way: If φ_t is larger than the maximum allowed wind speed φ_v^{SAFE} than no operation are allowed. If $\varphi^{\text{SAFE}} > \varphi_t$, the working hours of vessel v in period t are reduced to $\min\{S_2^v, \varphi_v^{\text{SAFE}} - \varphi_t\}$ hours. For the single wind farm case, we directly incorporated further travel and transfer times between depot and wind farm into the vessels' daily working hours.

4.4.2 A comparison of second-stage special cases

We provide an overview of the computational performance of the different formulations in two parts. First, we consider Benchmark Set A and provide an overview of the three variants of minimum service requirements models for the single wind farm case, with and without dedicated vessels, with and without bundle preprocessing. In other words, we obtain the Special Case I-III solutions for each of the three settings regarding minimum service requirements. The results are presented in Table 4.2. Next, we solve the instances of Benchmark Set B as Special Cases IV and V, i.e., with and without bundle preprocessing. These results are provided in Table 4.3. We will omit the arguments of the $Q(\cdot)$ functions to enhance readability.

All the instances are solved by means of CPLEX 12.8.0 via its callable library in C++. The runtime is limited to 10800 seconds or when 24gb of RAM is used. At most four parallel threads are exploited by CPLEX. In the following, we will discuss the results presented in Tables 4.2 and 4.3. Initial experiments have shown that the results are robust with respect to the values of α and β, and we, therefore, fixed those on 0.0175 and 0.02, respectively.

4.4.2.1 The single wind farm case.

We compare solving the Special Case models assuming a single wind farm ($Q_{\text{F1}}(\cdot)$, $Q_{\text{F1}}^{\alpha}(\cdot)$, and $Q_{\text{F1}}^{\beta}(\cdot)$), assuming a single wind farm with dedicated vessels ($Q_{\text{F1-V1}}(\cdot)$,

$Q^{\alpha}_{\text{F1-V1}}(\cdot)$, and $Q^{\beta}_{\text{F1-V1}}(\cdot)$), and assuming a single wind farm with bundle-preprocessing ($Q_{\text{F1-B}}(\cdot)$, $Q^{\alpha}_{\text{F1-B}}(\cdot)$, and $Q^{\beta}_{\text{F1-B}}(\cdot)$). Recall that, the Special Cases I and IIs are (variants of) the capacitated facility location problem, whereas the Special Case III models are (variants of) a set-covering formulation (see Section 4.3 for further details).

The models with bundled maintenance tasks assume that all tasks can be performed within a single period, hence a preprocessing step is done to convert the instances so that the models $Q_{\text{F1-B}}(\cdot)$, $Q^{\alpha}_{\text{F1-B}}(\cdot)$, and $Q^{\beta}_{\text{F1-B}}(\cdot)$ can be solved: Every task lasting more than 8 hours is partitioned into tasks of 8 hours and a task that takes the remaining hours. In this way, every task can be performed within a period. Preliminary experiments have shown that this provides us with the most convenient and comparable instances, i.e., a significant amount of instances becomes infeasible if we increase the maximum task duration to more than 8 hours. In addition, we ensure that there is no bundle containing multiple tasks that correspond to the same original task.

In Table 4.2, the results of solving all the models is provided. Some instances of the Special Case III models ($Q_{\text{F1-B}}$) were not solved to optimality (indicated with an asterisk), but their final optimality gaps were so small (0.20% or smaller) that reporting it is not relevant. It is observed that the runtime of all the models become larger if the instance sizes grow, which is expected. Significant differences are observed between instances of the same size, which is typically caused by differing distances between wind farm and depot. Larger distances inherently complicate the planning process, which is expressed by the runtime for the particular instances.

A few observations stand out. First, averaging over all the results, the special cases with the β downtime setting take on average 969 seconds compared to on average 25 and 140 for the second setting where α technician hours are unmet and the first setting where all jobs need to be scheduled, respectively. Second, although α is only small (0.02), the decrease in cost-estimation between the first and second setting is on average 10.13%. Third, the differences between whether or not vessels are dedicated (Special Case II) or not have their influence on both the running time and on the objective values. The objective values raise on average with 5.10%, 4.34%, and 5.43% for the three settings of the SMFTPO (i.e, $Q_{\text{F1-V1}}(\cdot)$, $Q^{\alpha}_{\text{F1-V1}}(\cdot)$, and $Q^{\beta}_{\text{F1-V1}}(\cdot)$), respectively. The runtime remarkably decreases when vessels are dedicated, which might be explained by a more restricted solution space. In other words, branch and bound is more efficient which decreases the runtime of the instances that are difficult to solve, e.g., the runtime of instance 25 reduces from 6623 seconds to 2757 seconds.

Fourth, the effect of β on the resulting objective values is not very large, i.e., the objective value increases with 1.27%, 1.59%, and 0.80% for the single wind farm formulation, the single wind farm with dedicated vessels formulation, and the bundled

formulation, respectively. Finally, the average increase in objective value that is observed when using the bundled formulations (Special Cases III) instead of the non-bundled formulations (Special Cases I and II) is remarkable. This increase equals 9.91%, 9.03%, and 9.40% for the three SMFTPO settings, respectively. Comparing the runtime, those differences are 10779%, 7444% and -98% for the three settings, respectively.

The last result has severe practical implications for operations managers in offshore wind maintenance service logistics. Although the bundled formulations have their popularity (see, e.g., Gundegjerde et al. 2015), and are quite intuitive to incorporate and model, they come at an overestimation of the actual maintenance planning costs. The major difficulty is how the bundles are generated, and how tasks that take more than a single period are split up over multiple tasks. Summarizing, taking into account the cost estimation increase and the differences in runtime, only for the third SMFTPO setting (with at most β downtime periods) one might prefer the use of a bundled formulation. For the other minimum service requirements (all the tasks or the at least α technician hours variant), we advise to using the non-bundled formulations.

4.4.2.2 The multiple wind farm case.

The results of solving the Special Case IV models $Q(\cdot), Q^{\alpha}(\cdot)$, and $Q^{\beta}(\cdot)$ and their bundle-preprocessed counterparts (the Special Case V models) $Q_{\mathrm{B}}(\cdot), Q^{\alpha}_{\mathrm{B}}(\cdot)$, and $Q^{\beta}_{\mathrm{B}}(\cdot)$ are given in Table 4.3. The columns headed with "UB", "LB", and "Sec" denote the best upper bound, the best lower bound, and the runtime of the solver, respectively.

What stands out is the difference in computational efficiency between the Special Case IV and V models. The average optimality gap for the Special Case IV models equals 10.00 % against 0.07% for their bundled counterparts formulations. Since the lower bounds of the basic formulations are significantly lower than the optimal solutions found by the bundled formulations, we infer that the bundled formulations are overestimating the costs similarly as in the single wind farm case, but no hard conclusions cab be drawn from this

Regarding the difference between the cost estimations for different minimum service requirement policies, we focus on the models with bundled tasks (Special Case V). Average cost differences of -10.18% and 6.10% are observed for the $Q^{\alpha}(\cdot)$ and $Q^{\beta}(\cdot)$ settings over the $Q(\cdot)$ setting, respectively. The 6.10% increase of incorporating the at most β downtime periods constraints stands out when compared with the increases around 2 % in case of a single wind farm. Finally, similar to the single wind farm case, the second setting of the SMFTPO (α technician hours unscheduled) is relatively difficult to solve compared to the other two minimum service requirement settings.

Table 4.2: Computation performance of the different models and formulations to the instances of Benchmark Set A. Time (t) is measured in seconds.

			Special Case I						Special Case II						Special Case III					
			$Q_{\text{F1}}(\cdot)$		$Q^{\alpha}_{\text{F1}}(\cdot)$		$Q^{\beta}_{\text{F1}}(\cdot)$		$Q_{\text{F1-V1}}(\cdot)$		$Q^{\alpha}_{\text{F1-V1}}(\cdot)$		$Q^{\beta}_{\text{F1-V1}}(\cdot)$		$Q_{\text{F1-B}}(\cdot)$		$Q^{\alpha}_{\text{F1-B}}(\cdot)$		$Q^{\beta}_{\text{F1-B}}(\cdot)$	
Inst.	L	N	Obj.	t	Obj.	t	Obj.	t	Obj.	t	Obj.	t	Obj.	t	Obj.	t	Obj.	t	Obj.	t
1	2	50	46264	0	42951	0	47789	33	47789	0	43828	1	49313	20	49301	0	45538	0	50825	0
2	2	50	51261	0	46625	0	52029	11	52414	0	47710	0	53950	14	55846	0	51618	1	56614	0
3	2	50	47238	0	43048	0	47722	1	51111	0	45911	0	53047	14	53991	0	48257	3	53991	0
4	2	60	55457	0	50399	0	55457	1	56994	0	51981	1	56994	1	60424	0	55059	0	60808	0
5	2	60	38812	0	34554	2	38812	2	39178	0	34800	1	39178	1	43548	0	39164	1	43548	0
6	2	60	54348	0	50202	0	54348	0	57846	0	53313	0	57846	0	63077	0	58836	0	63077	0
7	2	70	60250	0	54850	0	60734	3	68481	0	61843	1	69449	14	68934	1	62908	1	68934	1
8	2	70	58325	0	53412	0	58325	0	60657	0	55814	0	60657	0	68219	0	62985	0	68219	0
9	2	70	65274	0	59305	5	69588	8108	69156	1	62059	3	75627	10800	73850	2352	65724	91	79024	42
10	2	80	48651	0	43404	1	48651	1	49750	0	44112	1	49750	0	54114	1	48724	2	54114	2
11	2	80	74622	1	68087	1	77967	7387	78525	4	72257	6	80755	2913	85746	1	77050	5	87976	4
12	2	80	73294	0	66644	1	74842	84	76391	1	69323	1	78455	308	80486	0	73846	7	81518	1
13	2	90	64307	0	57587	1	64307	1	68524	0	60981	1	68524	2	70359	1	64271	3	70359	1
14	2	90	74414	0	67355	1	74955	76	76037	1	68711	5	76037	8	82487	3	74168	8	83027	55
15	2	90	78066	12	69665	3	79200	10811	81468	1	72823	7	84493	7340	87102	19	76700	5227	87858	11
16	3	50	48570	0	43173	0	48570	0	49802	0	43583	0	49802	0	53872	0	46899	0	53872	0
17	3	50	49290	0	43783	1	49290	0	50071	0	43992	0	50071	0	53564	0	47795	33	53564	0
18	3	50	45418	0	41913	1	45418	0	45790	0	42017	1	45790	0	49498	0	45136	0	49498	0
19	3	60	58879	0	52344	1	58879	1	59269	0	52625	1	59269	1	64319	0	57526	0	64319	0
20	3	60	75774	0	67478	1	75774	5	78658	1	71090	2	78658	11	88358	1	77832	114	88358	1
21	3	60	72236	0	64853	1	72236	0	79376	0	66648	1	79376	1	83186	0	73879	1	83186	0
22	3	70	62337	0	55897	0	62337	0	64942	0	57732	1	64942	1	67895	0	60602	1	67895	0
23	3	70	83947	0	76543	1	83947	0	92734	0	81254	1	92734	0	94344	0	84042	0	94344	1
24	3	70	89487	0	79487	1	92392	1570	93845	0	83672	1	96750	1708	101552	1	90243	10	102520	4
25	3	80	94442	1	84256	23	95163	6623	98408	3	87757	3	99129	2757	107376	14	94828	6440	108096	47
26	3	80	101258	0	92557	1	104306	473	105322	0	95735	1	108878	438	118497	4	104630	12	118497	8
27	3	80	106455	0	95489	1	107728	175	111547	0	100936	2	112819	90	119973	4	105276	6	122093	710
28	3	90	125062	1	113918	18	127892	10800	135250	1	121354	9	140911	7765	141998	25	126697	40	143696	79
29	3	90	117250	3	105147	101	119094	4849	127391	9	112632	12	128774	954	-	-	-	-	-	-
30	3	90	90029	0	81986	1	90029	6	91919	1	83254	2	91919	37	100958	1	89613	6	100958	3
	Average		70367	1	63564	5	71259	1701	73955	1	66325	2	75130	1173	77340	84	69305	414	77958	34

The entries marked with '-' are infeasible due to bundle preprocessing. The $z_{\text{BASIC-2}}$ formulations of instance 9 are not solved to optimality (but with optimality gaps around 1 %)

Table 4.3: Computation performance of the different models and formulations to the instances of Benchmark Set B. Time (t) is in seconds.

				Special Case IV									Special Case V								
				$Q(\cdot)$			$Q^{\alpha}(\cdot)$			$Q^{\beta}(\cdot)$			$Q_{\mathrm{B}}(\cdot)$			$Q_{\mathrm{B}}^{\alpha}(\cdot)$			$Q_{\mathrm{B}}^{\beta}(\cdot)$		
Inst.	W	L	N^w	UB	LB	t	UB	LB	t	UB	LB	t	UB	LB	t	UB	LB	t	UB	LB	t
1	2	1	30	20868	20314	4919	19308	18524	4485	20868	20314	4962	21498	21498	0	20225	20225	1	22595	22595	0
2	2	1	30	36888	35287	6076	31741	31143	10814	36888	35287	5997	38277	38277	1	34749	34749	2	43982	43982	1
3	2	1	30	23389	22174	10800	20061	19458	4388	23389	22173	10800	23269	23269	0	20769	20769	1	24687	24687	0
4	2	1	45	43581	40033	3492	39519	35643	3870	43581	40033	3408	42001	42001	1	38401	38397	10	47978	47978	3
5	2	1	45	35680	35676	9461	32300	30766	10800	35680	35676	9370	36835	36835	3	33032	33030	48	41585	41582	2
6	2	1	45	30799	29481	2946	29298	27554	3749	30799	29481	2910	32786	32786	0	30712	30712	1	34628	34628	0
7	2	2	30	46569	43693	10800	43172	39335	10800	46569	43681	10800	48160	48160	0	44782	44782	2	48826	48826	0
8	2	2	30	57055	52956	10800	48447	45600	10800	57055	52958	10800	56749	56749	13	49186	49186	37	63307	63307	16
9	2	2	30	55974	51641	10800	52158	45815	2926	56361	51641	10800	56092	56092	0	51224	51224	1	58739	58739	0
10	2	2	45	69378	67079	10800	62504	57756	8205	69378	67081	10800	73003	73003	50	64296	63353	10800	75177	75177	100
11	2	2	45	78176	71039	3468	68947	63480	6866	78176	71039	3504	78773	78773	1	69681	69681	4	84667	84667	2
12	2	2	45	84779	77169	10800	74234	66361	4018	84696	77169	10800	79117	79117	1	69382	69171	2814	86763	86763	2
13	3	1	30	36437	33646	10800	33351	30192	10800	36354	33645	10800	36806	36806	1	35001	35001	4	40956	40956	1
14	3	1	30	42060	37181	2290	38623	32960	9352	41356	37414	5136	41951	41951	0	37694	37694	3	45403	45403	1
15	3	1	30	38071	33788	5864	32782	29321	10800	38071	33788	5932	36800	36800	1	32646	32645	6	39265	39265	1
16	3	1	45	47726	44964	4035	41960	39717	8696	47726	44964	4032	50097	50097	1	44607	44604	6	52897	52897	2
17	3	1	45	47669	44748	3852	42122	39357	10800	47669	44748	3910	49694	49694	4	43956	43777	7003	50760	50756	8
18	3	1	45	50155	46635	10808	45125	40662	9744	50101	46635	10808	50428	50428	0	46425	46425	1	51306	51303	0
19	3	2	30	76509	69907	10800	72566	60325	10801	76916	69907	10800	75438	75438	0	66540	66533	16	77558	77556	1
20	3	2	30	63822	59187	10800	57886	52498	10800	63822	59184	10800	64016	64016	1	57420	57415	44	66850	66850	3
21	3	2	30	74793	67691	10800	99011	60953	6832	74793	67691	10800	74654	74654	0	67851	67851	1	76066	76066	0
22	3	2	45	94307	85376	10800	86961	75648	10800	94307	85376	10800	93768	93768	3	83337	83107	5111	98861	98861	4
23	3	2	45	109801	100524	10800	100017	88250	10800	109801	100523	10800	110765	110765	8	99005	98269	2514	117641	117630	7
24	3	2	45	109988	99292	10800	141207	87692	3156	109624	99292	10800	108229	108229	1	97911	97385	5526	112740	112730	9
	Average			57270	52895	8234	54721	46625	8129	57249	52904	8349	57467	57467	4	51618	51499	1415	60968	60967	7

4.4.2.3 Implications.

The analysis of the results in Tables 4.2 and 4.3 made clear that it is important to have a good understanding of the different models and the underlying assumptions since it severely impacts the accuracy of the cost-estimations for tactical maintenance planning at offshore wind farms. What stands out is that the effect of bundling maintenance tasks. It essentially assumes that the tasks will be performed in a single period, by a single vessel, and this simplifies the underlying optimization problem at the expense of a quite expensive preprocessing step (of which the computation times are not taken into account in the results). This relates to the first three Modeling Categories, as identified in Sections 4.1.1.1 and 4.3. Especially when multiple wind farms are included and when the necessary skills to develop sophisticated solution approaches are lacking, the bundled tasks are efficient and effective for obtaining quick cost-estimations for medium-term maintenance planning problems.

The effect of the different minimum service requirement settings (Modeling Category 4) is also noteworthy. Operations managers should be aware of the fact that, when a fraction of jobs can be left unscheduled in the mathematical model, a multiplying effect is observed with regards to the cost estimations, i.e., not satisfying 2% of the demanded technician hours will result in cost-decreases of more than 2%. This is explained by noticing that in such cases the most expensive tasks will be left (partly) unscheduled. However, it might result in a more realistic description of the actual behavior of the maintenance service provider, as such minimum service requirements are commonly encountered. Hence, if the models as presented in this paper will be used for predicting the behavior of maintenance providers, it is advisable to include such minimum service requirements in the mathematical model formulation.

Furthermore, when a restriction in downtime periods is included in the minimum service requirements, it can be seen that the resulting estimated cost-increases will slightly increase. If such requirements are part of a service contract between wind farm owner and maintenance provider, it is likely that the maintenance provider will ask a higher price for the maintenance operations. The results have shown, however, that this increase should only be slight. This holds as well for the assumption that a maintenance task cannot be performed by multiple vessels, although this assumption reflects upon the operations of the maintenance provider and is less likely to be part of the service contract negotiations.

Finally, the implications are not only of interest for practitioners (e.g., operations managers at wind farms), but also provide guidance for scientists to build upon the models we propose for the SMFTPO settings and its special cases.

4.4.3 Computational results of the SMFTPO

In the following, we present experiments on the three settings of the SMFTPO in its two-stage stochastic optimization model. We briefly discuss how we generated a suitable set of benchmark instances and the configuration of the Sample Average Approximation algorithm. Consequently, we analyze the obtained solutions and provide insights in the computational difficulty.

As as indicated by the results of the Special Case IV and V models, the preprocessed maintenance tasks (i.e., considering bundles) is computationally the most efficient. We, therefore, apply this preprocessing and solve the instances as such in the following.

4.4.3.1 Benchmark instances for the SMFTPO.

To evaluate the three settings of the SMFTPO in the full two-stage stochastic optimizatino model by means of Sample Average Approximation (SAA), two main ingredients are required. First, we need to be able to draw scenarios in an independent and identical fashion, and second, we need a series of benchmark instances with varying characteristics of the base system. The scenarios are identically generated as described for the previous benchmark instances with the major distinction that the previous benchmark instances were in fact single scenario realizations, and we use them in a sampling context as part of the SAA approach.

We differed the base system by the number of considered lease terms and the number of periods per lease term, see Table 4.4 for detailed characteristics. We considered three windfarms that each can be served from a single depot. The wind farm sizes are set to 30 turbines per farm. We consider the same vessel types as denoted in Table 4.1. The fleet is of a relatively large size: The number of small CTVs equals equals twice the number of depots, the number of medium sized CTV's equals the number of depots, we considered two helicoptors and a single supply vessel. Their charter prices per period equal 3500, 5000, 10000, and 25000 respectively. The penalty $\hat{P}$ for changing depots within a lease-term equals 2500.

4.4.3.2 Sample Average Approximation, EVPI, and VSS.

The SAA algorithm, as explained in Section 4.2, is used to evaluate the three settings of the SMFTPO on the benchmark instances described in the former. The lower bound estimates are obtained by averaging the lower bounds of $M = 96$ samples of $N = 25$ scenarios. Preliminary experiments have shown that this ensures a relatively low variance of the obtained estimators. Then, an upper bound estimation is obtained

by selecting the first-stage solution corresponding to the solution with lowest cost from the $M = 96$ samples. This first-stage solution is then evaluated on $M' = 1000$ scenarios. The average objective from these $M = 1000$ evaluations is an upper bound estimation of the SMFTPO.

Except for the lower and upper bound estimations, we are interested in two additional estimations. First, we are interested in the so-called *Expected Value of Perfect Information* (EVPI), which expresses the amount by which costs are reduced in a perfect information situation. We calculate it by solving the complete two-stage stochastic optimization model for each of the $M' = 1000$ scenario's obtained for estimating the upper bound. The average is then an estimator for the perfect information solution. The difference between this perfect information solution and the estimated upper bound is an estimation of the EVPI.

In addition, we are interested in the so-called *Value of the Stochastic Solution* (VSS). It expresses the reduction in costs if one considers the two-stage optimization model instead of using simply the expected values of the uncertain parameters. This expected value solution is obtained by solving the two-stage optimization model with a single scenario where all parameters are equal to their expected value. We evaluate this solution on $M' = 1000$ individually drawn scenarios and average the obtained objective values. The VSS is then the difference between this average and the estimated upper bound.

Finally, for computational reasons we terminate the MIP solver if the optimality gap is smaller than 2%. The experiments are performed on two Intel Xeon E5 2680v3 CPUs, totalling 24 cores. Each MIP is solved by four dedicated threads. We allow for a maximum of three days to calculate lower bound, upper bound, EVPI and VSS. Finally, to ensure relatively complete recourse decisions we include artificial binary variables (to perform an individual maintenance task) that we penalize at 15000 each.

4.4.3.3 Results of the SAA.

The results of solving all the benchmark instances as each of the three SMFTPO settings are given in Table 4.4. For each setting, we provide the estimated lower bound (in $\cdot 10^3$), upper bound (in $\cdot 10^3$), and the EVPI (in $\cdot 10^3$). We further provide the difference $\Delta(z, z^\alpha)$ in percentages of using z^α instead of z, and similarly $\Delta(z, z^\beta)$ denotes this difference for using z^β instead of z. We do no provide the VSS in this table as its values are very large, i.e, the estimated expected value solution is 2-5 times as large as the estimated upper bound. The explanation is simple; the difficulty of the SFMTPO is caused by the high variability in weather conditions and in maintenance activities which is not reflected by the 'nicely' behaving expected value solution.

Table 4.4: Overview of the solutions of the two-stage optimization model. $\widehat{LB}$, $\widehat{UB}$, and $\widehat{EVPI}$ are given in $\cdot 10^3$. The variance of the estimators in the order of 500 - 2000, and is not presented to enhance readibility reasons. Instances 9-11 and 16-18, setting z^α, are solved with 18 scenarios instead of 24 to estimate the lower bound.

					all jobs (z)			α uptime (z^α)			β downtime (z^β)				
Inst	L	T/L	D	rep	$\widehat{LB}$	$\widehat{UB}$	$\widehat{EVPI}$	$\widehat{LB}$	$\widehat{UB}$	$\widehat{EVPI}$	$\widehat{LB}$	$\widehat{UB}$	$\widehat{EVPI}$	$\Delta(z, z^\alpha)$	$\Delta(z, z^\beta)$
1	5	5	3	1	175.6	187.0	27.3	159.0	175.3	27.0	192.1	197.1	20.8	-6.3	5.4
2	5	5	3	2	171.8	179.9	26.7	155.5	165.7	22.6	188.3	195.0	21.5	-7.9	8.4
3	5	5	3	3	187.1	196.3	24.6	169.6	182.2	21.3	204.0	209.3	19.8	-7.2	6.6
4	5	7	3	1	222.8	234.5	24.3	189.8	225.4	37.1	241.9	257.4	25.9	-3.9	9.8
5	5	7	3	2	221.2	234.3	25.5	188.3	211.7	25.8	240.3	253.1	25.8	-9.7	8.0
6	5	7	3	3	208.0	216.4	24.9	176.6	196.7	23.1	227.4	249.2	32.9	-9.1	15.1
7	8	5	3	1	289.3	303.1	37.2	245.5	270.9	37.2	310.1	320.0	30.9	-10.6	5.6
8	8	5	3	2	275.4	287.5	38.3	233.1	257.9	40.7	295.9	306.6	32.5	-10.3	6.6
9	8	5	3	3	283.9	295.2	35.7	241.1	261.4	32.8	304.5	316.4	31.4	-11.4	7.2
10	8	7	3	1	368.9	390.3	36.4	297.5	351.3	52.3	391.5	415.6	36.7	-10.0	6.5
11	8	7	3	2	349.6	368.0	38.5	279.8	316.4	36.3	371.7	394.7	38.8	-14.0	7.3
12	8	7	3	3	371.0	390.9	35.9	299.4	335.8	35.6	393.5	413.7	33.0	-14.1	5.8
13	10	5	3	1	335.1	350.5	49.4	274.3	305.5	48.0	358.7	373.5	42.9	-12.8	6.6
14	10	5	3	2	339.8	358.5	48.7	279.2	306.8	43.6	362.1	379.3	40.3	-14.4	5.8
15	10	5	3	3	356.9	374.3	46.2	295.1	340.4	60.5	381.0	399.6	44.8	-9.1	6.7
16	10	7	3	1	418.1	443.0	50.9	328.4	389.3	63.6	444.6	485.0	62.7	-12.1	9.5
17	10	7	3	2	416.8	436.7	48.6	328.1	380.5	55.2	444.5	489.2	64.1	-12.9	12.0
18	10	7	3	3	433.5	458.3	47.9	342.7	412.9	71.0	460.6	490.8	48.1	-9.9	7.1

First it should be noted that the estimated lower and upper bounds for all the SFMTPO settings, and all the instances, are rather tight. As we terminated the MIP solver at 2%, this gap should is at least 2%, and the differences are only slightly larger. Second, we see that the estimated differences between scheduling all tasks and leaving α technician hours unscheduled (i.e., $\Delta(z, z^{\alpha})$) is becoming larger for longer planning horizons, whereas for $\Delta(z, z^{\beta})$ no such trend is visible. This is explained by the fact that for longer planning horizons more variability in the uncertainty realizations is observed, which increases the importance to anticipate upon the uncertain demand by not scheduling maintenance tasks.

The EVPI is, as expected, increasing for instances with longer planning horizons. This is trivial, as a longer planning horizon includes more uncertainty to deal with, and therefore a larger variability (in absolute value) in the uncertainty realizations. We furthermore observe that the EVPI is rather small for a tactical planning problem as the one we propose. Namely, although vessels will be allocated more in the solution corresponding to the estimated upper bound (i.e., the stochastic programming solution), the effect on individual perfect information solutions is not devestating. Still, the EVPI is around 10%, which might motivate further research in this area for more dynamic planning algorithms.

4.5 Conclusion

In this paper, we introduced the Stochastic Maintenance Fleet Transportation Problem for Offshore wind farms (SMFTPO). In the SMFTPO, we aim to find a cost-minimizing assignment of maintenance tasks to vessels while controlling for uncertain maintenance tasks and weather conditions. We take the viewpoint of a single maintenance provider which is responsible for the maintenance at multiple wind farms, a situation often encountered in practice. This is in high contrast with existing research on tactical and strategic decisions in offshore wind maintenance service logistics; current research does not make the distinction between the maintenance provider and wind farm owner.

We introduced the notion of minimum service requirements, in the context of offshore wind, describing the contractual obligations of the logistics provider to the wind farm owner. Three settings of those minimum service requirements were considered. We then modeled those settings of the SMFTPO as a generic two-stage stochastic mixed integer programming model. The second-stage problem is modelled on a decomposed and time expanded-network, and Sample Average Approximation was used to solve a scenario-based large-scale mixed integer programming model.

In addition, by means of a thorough literature review, we provided an overview

of the key-modeling decisions in related work by presenting four modeling categories in the context of offshore wind maintenance service logistics. To gain insights into the (computational) tractability of those key-modeling decisions, we presented five (re)formulations of the second-stage problem of each of the three SMFTPO settings.

We provided a new set of benchmark instances, which we made publicly available, and used those to assess the computational performance of all the models. First, focussed on the (re)formulations of the second-stage problems of the SMFTPO. It is shown, amongst others, that an established method for bundling maintenance tasks results in overestimating medium-term maintenance costs but is a computationally attrictive. Moreover, it is shown that incorporating additional constraints incentivizing quickly scheduling maintenance tasks is especially costly in a multiple wind farm setting. Second, we studied the solutions of the two-stage stochastic optmization model for each of the three SMFTPO settings. It is clearly shown that considering uncertainty is a must, as the value of the stochastic solution is large, while on the other hand, the expected value of perfect information is rather small.

The directions for further research are numerous. A natural extension of this exploratory work is the incorporation of the notion of minimum service requirements in a multi-stage stochastic programming model, instead of a two-stage stochastic programming model. The development of such multi-stage models goes hand-in-hand with the need for advanced solution algorithms (e.g., integer L-shaped algorithms). Up until now, no such advanced algorithms exist in the context of strategic and tactical offshore wind maintenance service logistics. Such algorithms are required for solving practical cases in a more dynamic fashion since the increasing number of offshore wind farms and increased collaboration between the wind farms complicates the optimization problem significantly.

Furthermore, building upon the increasing popularity of approximate dynamic programming (Powell 2007), a fundamentally different approach might be interesting to pursue. Namely, to model the SMFTPO by means of a stochastic dynamic program, instead of a two- or multi-stage stochastic mixed integer program. A comparison of the required level of dynanism in such optimization problems is interesting for researchers and practicioners.

A different, but not less interesting extension could be the development of polynomial approximation algorithms to obtain cost-estimations for particular offshore wind scenarios. Then tactical planning models, as the ones developed in this paper, might be incorporated in real-time decision making which requires instant calculation of cost-estimations.

Appendices

4.A Monolithic formulation for solving the SMFTPO

Here we detail the complete MIP for solving the scenario-based formulation of the SMFTPO. We only provide this for the first setting of the SMFTPO (i.e., where all the jobs need to be performed).

$$\min \sum_{d\in\mathcal{D}}\sum_{\ell\in\mathcal{L}}\sum_{v\in\mathcal{V}} \hat{C}^{v}_{d\ell} y^{v}_{d\ell} + \frac{1}{N}\sum_{\xi\in\overline{\Xi}}\left[\sum_{v\in\mathcal{V}}\sum_{(i,j)\in\mathcal{A}^{\mathcal{T},v}} C^{v\xi}_{ij} x^{v\xi}_{ij} + \sum_{v\in\mathcal{V}}\sum_{m\in\mathcal{M}}\sum_{(i,j)\in\mathcal{A}^{\mathcal{T},v}_4(m)} F^{v\xi}_{ij} z^{v\xi}_{ij}\right] \tag{4.40}$$

$$\text{s.t.} \quad \sum_{d\in\mathcal{D}} y^{v}_{d\ell} \leq 1 \qquad \forall \ell\in\mathcal{L}, v\in\mathcal{V} \tag{4.41}$$

$$\sum_{(i,j)\in\mathcal{A}^{\mathcal{T},v,\xi}_5} y^{v}_{ij}\phi^{d\ell}_{ij} - x^{v\xi}_{d\ell} \leq 0 \qquad \forall d\in\mathcal{D}, v\in\mathcal{V}, \ell\in\mathcal{L}, \xi\in\overline{\Xi} \tag{4.42}$$

$$\sum_{(i,j)\in\delta^{+}(n)} x^{v\xi}_{ij} \leq 1 \qquad \forall\, n\in\mathcal{N}^{\mathcal{T},v,\xi}_{\mathcal{D}}, v\in\mathcal{V}, \xi\in\overline{\Xi} \tag{4.43}$$

$$\sum_{(i,j)\in\delta^{-}(n)} x^{v\xi}_{ij} \leq 1 \qquad \forall n\in\mathcal{N}^{\mathcal{T},v,\xi}_{\mathcal{D}}, v\in\mathcal{V}, \xi\in\overline{\Xi} \tag{4.44}$$

$$\sum_{(i,j)\in\delta^{-}(n)} z^{v\xi}_{ij} - \sum_{(i,j)\in\delta^{+}(n)} z^{v\xi}_{ij} = \delta^{n\xi} \quad \forall n\in\mathcal{N}^{\mathcal{T},v,\xi}, v\in\mathcal{V}, \xi\in\overline{\Xi} \tag{4.45}$$

$$z^{v\xi}_{ij} \leq U^{v\xi}_{ij} x^{v\xi}_{ij} \qquad \forall (i,j)\in\mathcal{A}^{\mathcal{T},v,\xi}, \xi\in\overline{\Xi} \tag{4.46}$$

$$\sum_{v\in\mathcal{V}}\sum_{(i,j)\in\mathcal{A}^{\mathcal{T},v,\xi}_4(m)} z^{v\xi}_{ij} \geq D^{m}_{1} D^{m}_{2} \qquad \forall m\in\mathcal{M}^{\xi}, \xi\in\overline{\Xi} \tag{4.47}$$

$$x^{v\xi}_{ij} \in \{0,1\} \qquad \forall\, (i,j)\in\mathcal{A}^{\mathcal{T},v,\xi}, v\in\mathcal{V}, \xi\in\overline{\Xi} \tag{4.48}$$

$$z_{ij}^{v\xi} \in \mathbb{R}_+ \qquad \forall\ (i,j) \in \mathcal{A}^{\mathcal{T},v,\xi}, v \in \mathcal{V}, \xi \in \overline{\Xi} \qquad (4.49)$$

$$y_{d\ell}^{v} \in \{0,1\} \qquad \forall d \in \mathcal{D}, \ell \in \mathcal{L}, v \in \mathcal{V} \qquad (4.50)$$

The above formulation is similar to the stochastic mixed integer programming formulation presented in Section 2. Only difference is that all second-stage decision variables are indexed by ξ, and in the objective we take the average costs of all the second-stage decisions.

4.B Additional MIP formulations of special cases

This appendix provides the MIP formulations referred to in Section 3.

4.B.1 Special Case I: Single wind farm

No additional variables or notation is required for finding $Q^{\alpha}_{\text{F1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\lambda} \mid \xi)$,, i.e, the generalization of the single wind farm case to include the α-uptime minimum service requirements. Notice we drop the indices $w \in \mathcal{W}$, since we assume that there is only a single wind farm. This special case asks for solving

$$Q^{\alpha}_{\text{F1}}(\hat{\mathbf{y}}, \mathbf{u} \mid \xi) := \min \quad \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} \sum_{v\in\mathcal{V}} C_{tm} \hat{y}^{v}_{tm} + \sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{T}} G^{v}_{t} u^{v}_{t} \qquad (4.51)$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}} D^{m}_{1} D^{m}_{2} \hat{y}^{v}_{tm} \leq U^{v}_{t} u^{v}_{t} \qquad \forall\ t \in \mathcal{T}, v \in \mathcal{V} \qquad (4.52)$$

$$\sum_{m\in\mathcal{M}^w} \sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{T}} \hat{y}^{v}_{tm} D^{1}_{m} D^{2}_{m} \geq (1-\alpha) \sum_{m\in\mathcal{M}} D^{1}_{m} D^{2}_{m} \qquad (4.53)$$

$$\sum_{t\in\mathcal{T}} \sum_{v\in\mathcal{V}} \hat{y}^{v}_{tm} \leq 1 \qquad \forall m \in \mathcal{M} \qquad (4.54)$$

$$0 \leq \hat{y}^{v}_{tm} \leq \overline{Y}^{v}_{tm},\ u^{v}_{t} \in \{0,1\} \qquad \forall\ t \in \mathcal{T}, m \in \mathcal{M}, v \in \mathcal{V} \qquad (4.55)$$

Objective (4.51) minimizes the fixed costs of deploying a vessel and the task specific costs. Constraints (4.52) ensure that the supplied technicians do not exceed the availability. Constraint (4.53) ensures that at least $(1-\alpha)$ of the total technician hours are supplied. Since a task cannot be performed more than once, Constraints (4.54) are required. Finally, Constraints (4.55) denote the domain of the decision variables.

To model $Q^{\beta}\text{F1}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\eta\lambda} \mid \xi)$ (at most β downtime periods with a single wind farm), additional variables are needed to track the latest period in which flow is send to a maintenance task. We define the continues variables η^{1}_{m} and the binary variables

η^2_{tm} to indicate the latest period in which task m is maintained and whether or not task m is performed in period t, respectively. Then, it boils down to finding

$$Q^{\beta}_{\text{F1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\eta}^1, \boldsymbol{\eta}^2 \mid \xi) := \min \quad \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} \sum_{v\in\mathcal{V}} C_{tm}\hat{y}^v_{tm} + \sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{T}} G^v_t u^v_t \tag{4.56}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}} D^m_1 D^m_2 \hat{y}^v_{tm} \leq U^v_t u^v_t \qquad \forall\, t \in \mathcal{T} v \in \mathcal{V} \tag{4.57}$$

$$\sum_{m\in\mathcal{M}^w} \eta_m - S_m \leq \beta NT \tag{4.58}$$

$$\eta^2_{tm} \geq \hat{y}^v_{tm} \qquad \forall\, m \in \mathcal{M}, t \in \mathcal{T}, v \in \mathcal{V} \tag{4.59}$$

$$\eta^1_m \geq t\eta^2_{tm} \qquad \forall m \in \mathcal{M}, t \in \mathcal{T} \tag{4.60}$$

$$0 \leq \hat{y}^v_{tm} \leq \overline{Y}^v{}_{tm}, S_m \leq \eta^1_m \leq E_m$$

$$u^v_t, \eta^2_{tm} \in \{0,1\} \qquad \forall\, t \in \mathcal{T}, m \in \mathcal{M}, v \in \mathcal{V} \tag{4.61}$$

Objective (4.56) minimizes the fixed costs of deploying a vessel and the task specific costs. Constraints (4.57) ensure that no more technicians are deployed than possible in each period. Constraints (4.58) ensure that at most β downtime periods are observed in any feasible solution. Constraints (4.59)-(4.13) ensure that the η_m variables are modeled correctly. Finally, the domain of the variables is indicated by Constraints (4.61).

4.B.2 Single wind farm case with dedicated vessels

We extend $Q^{\text{F1-V1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\lambda} \mid \xi)$ to the case in which at least $(1-\alpha)$ technician hours are supplied of the total demanded technician hours. It asks for solving

$$Q^{\alpha}_{\text{F1-V1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\lambda} \mid \xi) := \min \quad \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} \sum_{v\in\mathcal{V}} C_{tm}\hat{y}^v_{tm} + \sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{T}} G^v_t u^v_t \tag{4.62}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}} D^m_1 D^m_2 \hat{y}^v_{tm} \leq U^v_t u^v_t \qquad \forall\, t \in \mathcal{T}, v \in \mathcal{V} \tag{4.63}$$

$$\sum_{m\in\mathcal{M}^w} \sum_{v\in\mathcal{V}} \sum_{t\in\mathcal{T}} \hat{y}^v_{tm} D^1_m D^2_m \geq (1-\alpha) \sum_{m\in\mathcal{M}} D^1_m D^2_m \tag{4.64}$$

$$\sum_{t\in\mathcal{T}} \sum_{v\in\mathcal{V}} \hat{y}^v_{tm} \leq 1 \qquad \forall m \in \mathcal{M} \tag{4.65}$$

$$\sum_{t\in\mathcal{T}} \hat{y}^v_{tm} \leq \lambda^v_m \qquad \forall\, m \in \mathcal{M}, v \in \mathcal{V} \tag{4.66}$$

$$\sum_{v\in\mathcal{V}} \lambda_m^v \leq 1 \qquad \forall\, m \in \mathcal{M} \tag{4.67}$$

$$0 \leq \hat{y}_{tm}^v \leq \overline{Y}_{tm}^v \text{ and } \lambda_t^v, u_m^v \in \{0,1\} \qquad \forall\, t \in \mathcal{T}, m \in \mathcal{M} v \in \mathcal{V} \tag{4.68}$$

Objective (4.62) minimizes the fixed costs of deploying a vessel and the task specific costs. Constraints (4.63) ensure that no more technicians are deployed than possible in each period. Constraints (4.64) make sure that the fraction of not supplied technician hour is at most α, and Constraints (4.65) ensure that at task is not served more than once. Constraints (4.66) and (4.67) ensure that a each task is at most assigned to a one vessel. Finally, the variables' domains are modeled via Constraints (4.68).

The extension to at most β downtime periods asks for solving

$$Q^{\beta}_{\text{F1-V1}}(\hat{\mathbf{y}}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\eta}^1, \boldsymbol{\eta}^2 \mid \xi) := \min \quad \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} \sum_{v\in\mathcal{V}} C_{tm}\hat{y}_{tm}^v + G_t^v u_t^v \tag{4.69}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}} D_1^m D_2^m \hat{y}_{tm}^v \leq U_t^v u_t^v \qquad \forall\, t \in \mathcal{T} v \in \mathcal{V} \tag{4.70}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}^w} \eta_m^1 - S_m \leq \beta^w N^w T \qquad \forall w \in \mathcal{W} \tag{4.71}$$

$$\eta_m^1 \geq t\eta_{tm}^2 \qquad \forall\, m \in M, t \in \mathcal{T} \tag{4.72}$$

$$\eta_{tm}^2 \geq \hat{y}_{tm}^v \qquad \forall\, m \in M, t \in \mathcal{T}, v \in \mathcal{V} \tag{4.73}$$

$$\sum_{t\in\mathcal{T}} \hat{y}_{tm}^v \leq \lambda_m^v \qquad \forall\, m \in \mathcal{M}, v \in \mathcal{V} \tag{4.74}$$

$$\sum_{v\in\mathcal{V}} \lambda_m^v \leq 1 \qquad \forall\, m \in \mathcal{M} \tag{4.75}$$

$$0 \leq \hat{y}_{tm}^v \leq \overline{Y}_{tm}^v, S_m \leq \eta_m^1 \leq E_m$$

$$\eta_{tm}^2, \lambda_t^v, u_m^v \in \{0,1\} \qquad \forall\, t \in \mathcal{T}, m \in \mathcal{M}, v \in \mathcal{V} \tag{4.76}$$

Objective (4.69) minimizes the fixed costs of deploying a vessel and the task specific costs. Constraints (4.70) ensure that no more technicians are deployed than possible in each period. Constraints (4.71) ensure that at most β^w downtime periods are observed in any feasible solution. Constraints (4.72)- (4.75) ensure that the η_m, z_{tm}, and λ_m^v variables are modeled correctly. Finally, Constraints (4.76) indicate the decision variables' domains.

4.B.3 Bundle Preprocessing

The extension of $Q_{\text{F1-B}}(\tilde{\mathbf{y}} \mid \xi)$ to the α and β minimum service requirement variants is presented in the following. The variant in which at most a fraction $(1-\alpha)$ of the total technician hours is left unassigned asks for solving

$$Q^{\alpha}_{\text{F1-B}}(\tilde{\mathbf{y}} \mid \xi) := \min \quad \sum_{t\in\mathcal{T}}\sum_{b\in\mathcal{B}}\sum_{v\in\mathcal{V}} \tilde{y}^v_{tb} C^v_{tb} \tag{4.77}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}}\sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}} \tilde{y}^v_{tb} D^1_b D^2_b \geq (1-\alpha) \sum_{m\in\mathcal{M}} D^1_m D^2_m \tag{4.78}$$

$$\sum_{b\in\mathcal{B}} \tilde{y}^v_{tb} \leq 1 \qquad \forall\, t\in\mathcal{T}, v\in\mathcal{V} \tag{4.79}$$

$$\sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}}\sum_{b\in\mathcal{B}} \tilde{y}^v_{tb}\phi^m_b \leq 1 \qquad \forall\, m\in\mathcal{M} \tag{4.80}$$

$$\tilde{y}^v_{tb} \in \{0,1\} \qquad \forall\, t\in\mathcal{T}, v\in\mathcal{V}, b\in\mathcal{B} \tag{4.81}$$

Objective (4.77) minimizes the costs of assigning bundles to vessels. Constraints (4.78) ensures that at least $(1-\alpha)$ technician hours are supplied. Moreover, Constraints (4.79) ensure that each bundle is assigned only once, and Constraints (4.80) ensure that a maintenance task is not scheduled more than once. The domains of the decision variables are given in Constraints (4.81).

To model the $Q^{\beta}_{\text{F1-B}}(\tilde{\mathbf{y}} \mid \xi)$ variant, we introduce the integer parameter ξ_b indicating the number of downtime periods incurred when assigning bundle b . The formulation then becomes

$$Q^{\beta}_{\text{F1-B}}(\tilde{\mathbf{y}} \mid \xi) := \min \quad \sum_{t\in\mathcal{T}}\sum_{b\in\mathcal{B}}\sum_{v\in\mathcal{V}} \tilde{y}^v_{tb} C^v_{tb} \tag{4.82}$$

$$\text{s.t.} \quad \sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}} \tilde{y}^v_{tb}\phi^m_b \geq 1 \qquad \forall\, m\in\mathcal{M} \tag{4.83}$$

$$\sum_{b\in\mathcal{B}} \tilde{y}^v_{tb} \leq 1 \qquad \forall\, t\in\mathcal{T}, v\in\mathcal{V} \tag{4.84}$$

$$\sum_{m\in\mathcal{M}}\sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}} \tilde{y}^v_{tb}\xi_b \leq \beta NT \tag{4.85}$$

$$\tilde{y}^v_{tb} \in \{0,1\} \qquad \forall\, t\in\mathcal{T}, v\in\mathcal{V}, b\in\mathcal{B} \tag{4.86}$$

Objective (4.82) minimizes the costs of assigning bundles to vessels. Constraints (4.83) ensures that each task is performed, while not assigning more bundles to vessels on a

single period than possible (Constraints (4.84)). Constraints (4.85) ensures that the downtime constraints are respected. Finally, the domain of the decision variables is indicated via Constraints (4.86).

4.B.4 Bundle selection in basic formulation

Introducing bundles in the $Q^{\alpha}(\mathbf{x} \mid \xi)$ formulation leads to finding

$$Q^{\alpha}_{\mathrm{B}}(\mathbf{x} \mid \xi) := \min \quad \sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}^{\mathcal{T},v}} C^v_{ij} x^v_{ij} \tag{4.87}$$

$$\text{s.t.} \quad \sum_{(i,j) \in \delta^+(n)} x^v_{ij} \leq 1 \qquad \forall\, n \in \mathcal{N}^{\mathcal{T},v}_{\mathcal{D}}, v \in \mathcal{V} \tag{4.88}$$

$$\sum_{(i,j) \in \delta^-(n)} x^v_{ij} \leq 1 \qquad \forall n \in \mathcal{N}^{\mathcal{T},v}_{\mathcal{D}}, v \in \mathcal{V} \tag{4.89}$$

$$\sum_{(i,j) \in \delta^-(n)} x^v_{ij} - \sum_{(i,j) \in \delta^+(n)} x^v_{ij} = \delta^n \qquad \forall n \in \mathcal{N}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.90}$$

$$\sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}^{\mathcal{T},v}_4(b)} x^v_{ij} \phi^m_b \leq 1 \qquad \forall m \in \mathcal{M} \tag{4.91}$$

$$\sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}^{\mathcal{T},v}_4(b,w)} x^v_{ij} D^1_b D^2_b \geq (1 - \alpha^w) \sum_{m \in \mathcal{M}^w} D^1_m D^2_m \qquad \forall w \in \mathcal{W} \tag{4.92}$$

$$x^v_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in \mathcal{A}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.93}$$

Here, we let $\mathcal{A}^{\mathcal{T},v}_4(b,w)$ be the set of arcs incoming at bundle $b \in \mathcal{B}$ at windfarm $w \in \mathcal{W}$. Objective (4.87) minimizes the transportation and maintenance costs. Constraints (4.88) - (4.90) ensure that vessels are not split among wind farms and depots and ensure flow conservation of technician hours. Constraints (4.91) models that a task can only be performed once, and Constraints (4.92) model that the fraction of supplied technician hours is at least $(1 - \alpha^w)$. Finally, the domain of the decision variables is denoted by Constraints (4.93).

The variant in which the fraction of downtime periods is at most β^w asks for finding

$$Q^{\beta}{}_{\mathrm{B}}(\mathbf{x} \mid \xi) := \min \quad \sum_{v \in \mathcal{V}} \sum_{(i,j) \in \mathcal{A}^{\mathcal{T},v}} C^v_{ij} x^v_{ij} \tag{4.94}$$

$$\text{s.t.} \quad \sum_{(i,j) \in \delta^+(n)} x^v_{ij} \leq 1 \qquad \forall\, n \in \mathcal{N}^{\mathcal{T},v}_{\mathcal{D}}, v \in \mathcal{V} \tag{4.95}$$

$$\sum_{(i,j)\in\delta^-(n)} x_{ij}^v \leq 1 \qquad \forall n \in \mathcal{N}_{\mathcal{D}}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.96}$$

$$\sum_{(i,j)\in\delta^-(n)} x_{ij}^v - \sum_{(i,j)\in\delta^+(n)} x_{ij}^v = 0 \qquad \forall n \in \mathcal{N}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.97}$$

$$\sum_{v\in\mathcal{V}} \sum_{(i,j)\in\mathcal{A}_4^{\mathcal{T},v}(b)} x_{ij}^v \phi_b^m \geq 1 \qquad \forall m \in \mathcal{M} \tag{4.98}$$

$$\sum_{v\in\mathcal{V}} \sum_{(i,j)\in\mathcal{A}_4^{\mathcal{T},v}(b,w)} x_{ij}^v \xi_b \leq \beta N^w T \qquad \forall w \in \mathcal{W} \tag{4.99}$$

$$x_{ij}^v \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}^{\mathcal{T},v}, v \in \mathcal{V} \tag{4.100}$$

Constraints (4.94) - (4.98) are similar to the $z_{\text{BASIC}}^{\text{B}}$ formulation provided in Section 4. We let ξ_b be the number of downtime periods when bundle b is chosen to be performed. Then constraints (4.99) model the minimum service requirements. Finally, the domain of the decision variables is denoted by Constraints (4.100).

Part II

E-commerce logistics

Chapter 5

Order picker routing in the e-commerce era

Abstract. *E-commerce companies often use manual order-picking systems in their warehouses since these systems can provide the required flexibility and scalability. Manual systems have been widely studied, but the operating policies may require significant changes for e-commerce settings. First, to maintain consumers' loyalty, it is important to maintain delivery reliability even on the busiest days. When the number of order pickers in an area increases, however, more delays due to interactions may occur. For example, travel speed may need to be lowered when order pickers pass each other in narrow aisles. Second, many products sold through e-commerce are returned by consumers. Before these returned products can be sold again, they must be reintegrated in the stock. This paper presents hybrid genetic algorithms to determine routes for simultaneous pickup of products in response to consumers' orders and delivery of returned products to storage locations. Furthermore, interactions between the order pickers are considered in the routing decisions. The developed algorithms use specific warehouse problem characteristics. We identify the mix of pickups and deliveries to realize the highest savings in practice. It is shown that order-picker interactions can be a significant cause for delay and should be accounted for in the routing.*

This chapter is based on Schrotenboer et al. (2017):
Schrotenboer AH, Wruck S, Roodbergen KJ, Veenstra M, Dijkstra AS, 2017 *Order picker routing with product returns and interaction delays. International Journal of Production Research* 55(21):6394–6406

5.1 Introduction

Often flexible and complimentary return options are offered to consumers in e-commerce, which allow consumers to order more products than are actually needed. Consumers then make the actual purchase decision at home after the order is delivered. The remainder of the products is returned to the e-commerce company's warehouse. Return rates of up to 74% occur, as noticed by Mostard, De Koster, and Teunter (2005), and many of those products can be resold after inspection and repackaging. This return flow leads to an additional cost and labor effort in the warehouse, since the returned products have to be reintegrated in the stock before they are available for reselling.

Online retailers also face significant variations in demand. Especially in December, many e-commerce warehouses are challenged to keep up with demand for seasonal gifts. Although manual order picking systems are flexible and scalable, a doubling of the workforce does not necessarily lead to a doubling in throughput. With an increase in the number of order pickers in any area, more interactions between the order pickers arise, causing lower productivity. For example, aisles are typically so narrow that when two vehicles need to pass, careful maneuvering is required. Furthermore, one order picker may be blocking access to a location from which another order picker needs to retrieve products.

E-commerce is thus redefining the requirements for the operation of warehouses, see De Koster, De Brito, and de Vendel (2002) and Stock and Mulki (2009). The warehouse process of retrieving products from storage in response to customers' orders is known as order picking. This process is generally thought to be the most costly and labor-intensive part in warehouse operations. It can contribute to 55% of the overall warehouse operation costs, see Tompkins et al. (2010). Furthermore, the largest portion of an order picker's time in manual picking systems is spent on traveling between locations. Product returns and order-picker interactions only add to these travel costs. The high costs involved in order picking and the challenges of product returns and interactions in busy e-commerce environments have motivated us to revisit the warehouse routing problem.

Our first goal is to incorporate the restocking of returned products in the order-picking routes. It is important to realize that the restocking of returned products is similar to the order-picking process, and quite different from regular stock replenishments. For regular stock replenishments, a vehicle typically replenishes only one or a few locations per trip with large quantities of the product. Restocking of returned items, on the other hand, requires visiting many locations while restocking only a

single item per location, which results in a significant amount of travel. Thus for order picking, an order picker starts with an empty vehicle and gradually fills the vehicle to its capacity by retrieving products from storage locations in response to customers' demand. While for restocking of returned products, an order picker starts with a full vehicle, which is gradually emptied by bringing returned products to their designated storage locations. As with the general Traveling Salesman Problem with Pickup and Delivery (TSPPD), which is NP-hard, see Mosheiov (1994), it seems advantageous for travel distances to integrate the two processes.

Our second goal is to give insights in the effects of order-picker interactions on efficiency and to incorporate interaction avoidance strategies as an integral part of the routing method. Order-picker interactions are especially of concern when designing routes that combine the restocking of returned products with the order picking of customers' orders, as is our first goal. Due to the capacity restriction of the vehicle, an order picker may not always be able to pick products when it is most logical from a routing point of view, since it may be necessary to first free capacity in the vehicle by dropping off returned products. This will cause the routes to be more complex and to include some backtracking. Furthermore, for any given capacity of the vehicle, more locations can be visited in a single route with restocking than in a route without restocking. Both the increased probability of backtracking and the increased number of stops per route, will increase the probability of order-picker interactions. Hence the need to study return handling and order-picker interactions simultaneously.

Next to routing, there are often also other control methods involved in operating a picking area, see De Koster, Le-Duc, and Roodbergen (2007). For example, batching methods aim at combining (parts of) several orders into a single picking route (e.g., Hong, Johnson, and Peters (2012)). Our focus on routing can be explained from the fact that product returns and picker interactions must be included in the routing method, since the routing method serves to verify feasibility and route length. Furthermore, the use of additional methods is not precluded by our approach, since the products considered in the routing can be the result of a batching method.

Order-picker routing in warehouses, for picking activities only, is a well studied topic in research. We refer to Gu, Goetschalckx, and McGinnis (2007), De Koster, Le-Duc, and Roodbergen (2007) and Gong and De Koster (2011) for general warehouse literature and order picking in specific. Recently, Theys et al. (2010) consider multi-aisle warehouse layouts and concluded that the inclusion of local-search techniques in the warehouse routing problem is promising. To exploit the benefit of the inclusion of sophisticated search methods in the solution procedure, we propose a hybrid genetic algorithm (HGA), i.e., a genetic algorithm with local search aspects for solving the

warehouse routing problem with pickups and deliveries (warehouse TSPPD). HGAs for general pickup and delivery problems already exist, see Zhao et al. (2009a) and Zhao et al. (2009b). We however construct a heuristic that is inspired by more sophisticated HGAs as presented by Vidal et al. (2012, 2013). Furthermore, we use specific characteristics of the warehouse routing problem in the algorithmic design.

Also an extended version of this HGA is presented that can reduce order-picker interactions by quantifying and penalizing them. Interaction between order pickers was first investigated by Pan and Shih (2008) and Parikh and Meller (2009) from a queuing theory perspective. Recently, Chen et al. (2013, 2016) looked into a related problem problem for order-picking without product returns. Our method allows for a trade-off between delays caused by interactions and the time required for interaction avoidance strategies, while the approach of Chen et al. (2013, 2016) requires all interactions to be avoided.

Our two HGAs can identify routes for combined order picking and restocking in low computation times that are acceptable for real-time applications. Moreover, they are suitable for various warehouse layouts and can therefore be widely applied. For situations without order-picker interactions, we demonstrate that near-optimal, and often optimal, solutions are obtained by the HGA. Furthermore, we perform an analysis to identify the best way of mixing restocking and order-picking requests in the routes. Finally, we demonstrate that significant improvements are achieved by explicitly considering order-picker interactions in the algorithmic procedure.

The structure of the paper is as follows. In Section 6.3 we give a detailed problem description and introduce an ILP formulation of the problem under study. Section 5.3 is dedicated to the explanation of the HGA. A description of the extensions made to the HGA to account for order-picker interactions is given in Section 5.4. We discuss the results of our numerical experiments in Section 5.5, and conclude the paper in Section 5.6.

5.2 Problem description

This paper considers the warehouse pickup and delivery problem with order picker interaction. We first present a description of the warehouse pickup and delivery problem without interactions (warehouse TSPPD) for which an Integer Linear Program (ILP) is constructed as well. After that, order picker interaction is defined in the general setting of multiple order pickers. We consider a warehouse consisting of two cross aisles that are connected by n_{aisle} parallel aisles of length a_{length} and width a_{width}. A graphical representation of this system is given in Figure 7 in De Koster, Le-Duc, and

Roodbergen (2007). All routes start and end at the central depot.

5.2.1 Single order picker

The warehouse TSPPD is defined on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = (\mathcal{P} \cup \mathcal{D} \cup \{0\})$ the set of vertices representing the products to be picked ($\mathcal{P}$), delivered ($\mathcal{D}$), and the central depot $\{0\}$, and $\mathcal{E}$ is the set of edges connecting all locations $v \in \mathcal{V}$. The orders used can be the result of batching methods. Let $n = |V| - 1$ be the order size and let $c : \mathcal{E} \to \mathbb{R}$ be the cost or distance function. For $i, j \in \mathcal{V}$, c_{ij} is defined as the shortest distance between i and j, see Theys et al. (2010). The order picker transport capacity $q > 0$ is defined as the maximum number of products an order picker can transport at any time.

We present an ILP formulation for the single order-picker case, based on the model in Mosheiov (1994). For $(i, j) \in \mathcal{V}$, let x_{ij} be a binary variable equaling 1 if the order picker travels along edge (i, j) and 0 otherwise. Furthermore, let $y_{ij} \geq 0$ be the total load already picked and transported along edge (i, j), and let $z_{ij} \geq 0$ be the total load to be delivered and transported along edge (i, j).

The warehouse TSPPD can now be formulated as:

$$\min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \tag{5.1}$$

subject to

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad \forall\, j \in \mathcal{V} \tag{5.2}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad \forall\, i \in \mathcal{V} \tag{5.3}$$

$$\sum_{j=0}^{n} y_{ij} - \sum_{j=0}^{n} y_{ji} = \begin{cases} p_i & i \in \mathcal{P} \\ -\sum_{j=0}^{n} p_i & i = 0 \\ 0 & i \in \mathcal{D} \end{cases} \tag{5.4}$$

$$\sum_{j=0}^{n} z_{ij} - \sum_{j=0}^{n} z_{ji} = \begin{cases} -d_i & i \in \mathcal{D} \\ \sum_{j=0}^{n} d_i & i = 0 \\ 0 & i \in \mathcal{P} \end{cases} \tag{5.5}$$

$$y_{ij} + z_{ij} \leq q x_{ij} \qquad \forall\, i, j \in \mathcal{V} \tag{5.6}$$

$$x_{ij} \in \{0, 1\}, \quad y_{ij},\ z_{ij} \geq 0 \qquad \forall\, i, j \in \mathcal{V} \tag{5.7}$$

The objective function (5.1) represents the total travel costs to be minimized when traveling a complete route. The Constraints (5.2) and (5.3) assure that each location is visited exactly once. With constraints (5.4) and (5.5) we control for the currently transported volume between any pair of locations i and j. Constraint (5.4) requires that the volume of any location i is picked up, if $i \in \mathcal{P}$, and that all products were picked up, when the order picker returns to the depot ($i = 0$). Constraint (5.5) states that the entire volume to be delivered to location i is delivered , if $i \in \mathcal{D}$, and all items were delivered at the end of the route. Constraint (5.6) restricts the volume of the total currently transported load between each pair of locations i and j to the transport capacity of the order picker.

5.2.2 Multiple order pickers and interaction effects

We consider two interaction events that cause delays for order pickers. First, if order pickers are in close proximity, both order pickers are assumed to incur a delay. Such delay may be caused by various reasons, including blocking of access to pick locations or simply slowing down for safety reasons. Second, two order pickers traveling in opposite directions slow down when passing each other, which is also registered as a delay. For brevity we will say that order pickers are *close* and that order pickers *cross*, for these two events respectively. To be able to quantify the interaction effects, we first extend our description of the problem by introducing multiple order pickers and by adding a trace of the route each order picker travels. That is, at some well-defined moments in time, the location of the order pickers has to be known in order to detect order picker interactions. To do so, it is assumed that order pickers travel at constant speed s_{cross} and s_{par} through cross aisles and parallel aisles, respectively. Product picking time and delivery time are constant at s_{pick}, and order pickers are assumed to be stationary during this time.

Let $\mathcal{M} = \{1, \ldots, m\}$ be the set representing all order pickers. Then for some order picker $a \in \mathcal{M}$, let $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a)$ be a complete graph with $\mathcal{V}_a = (\mathcal{P}_a \cup \mathcal{D}_a \cup \{0\})$ the set of vertices representing the products to be picked ($\mathcal{P}_a$), delivered ($\mathcal{D}_a$), and the central depot ($\{0\}$), and $\mathcal{E}_a$ is the set of edges connecting all locations $v \in \mathcal{V}_a$. The order size n, the cost function c and the transport capacity q are defined as before. Furthermore, let $\mathcal{S}_a$ be the set consisting of all solutions to the order picker routing problem on $\mathcal{G}_a$ and $\mathcal{S} = \cup_{a \in \mathcal{M}} \mathcal{S}_a$.

For some order picker $a \in \mathcal{M}$ and some solution $\sigma_a \in \mathcal{S}_a$, the interaction costs $\mathcal{I}(\sigma_a)$ are defined as the delay order picker a incurs due to interaction with the other $m - 1$ order pickers. The time horizon is given by $\mathcal{T} = \{0, \delta_{\text{time}}, 2\delta_{\text{time}}, 3\delta_{\text{time}}, \ldots, T\}$,

where δ_{time} is a properly chosen step size transforming the continuous time horizon into a discrete one and T is the latest time an order picker is finished. The function $\tau : \mathcal{S}_a \times \mathcal{T} \to \mathbb{R}^2$ maps σ_a and time t to the location in the warehouse of order picker a at time t while traveling according to solution σ_a. Such a location is represented by a pair of coordinates, i.e. $\tau(\sigma_a, t) = (x_a, y_a) \in \mathbb{R}^2$. Then for some order pickers $a, b \in \mathcal{M}$ and corresponding solutions $\sigma_a \in \mathcal{S}_a, \sigma_b \in \mathcal{S}_b$ and time $t \in \mathcal{T}$, order pickers are said to be *close* if $|\tau(\sigma_a, t) - \tau(\sigma_b, t)| \leq \delta_{\text{loc}}$, where $|\tau(\sigma_a, t) - \tau(\sigma_b, t)|$ is defined as the warehouse distance between $\tau(\sigma_a, t)$ and $\tau(\sigma_b, t)$. Then let $N_{\text{loc}}^{(\sigma_a, \sigma_b)}$ be the number of times order pickers a and b are close when they travel according to solution σ_a and σ_b respectively. It is defined as

$$N_{\text{loc}}^{(\sigma_a, \sigma_b)} = \sum_{t \in \mathcal{T}, t \neq 0} I\{|\tau(\sigma_a, t) - \tau(\sigma_b, t)| \leq \ \delta_{\text{loc}}\}, \tag{5.8}$$

where $I\{\cdot\}$ is an indicator function returning 1 if the condition between parentheses is true and 0 otherwise.

Let $h(\sigma_a, \sigma_b, t) = \big(\tau(\sigma_a, t) - \tau(\sigma_b, t)\big) \odot \big(\tau(\sigma_a, t - \delta_{\text{time}}) - \tau(\sigma_b, t - \delta_{\text{time}})\big) \in \mathbb{R}^2$, where $\odot$ is the Hadamard product, i.e., element-wise multiplication. Then the number of times order pickers *cross* for solutions σ_a, σ_b, denoted by $N_{\text{cross}}^{(\sigma_a, \sigma_b)}$, is

$$N_{\text{cross}}^{(\sigma_a, \sigma_b)} = \sum_{t \in \mathcal{T}, t \neq 0} \sum_{i=1}^{2} I\Big\{h(\sigma_a, \sigma_b, t)_i < 0 \wedge h(\sigma_a, \sigma_b, t)_1 \cdot h(\sigma_a, \sigma_b, t)_2 = 0\Big\},$$

where $h(\sigma_a, \sigma_b, t)_i$ refers to the i-th element of the vector $h(\sigma_a, \sigma_b, t)$. The order picker crossings are penalized with a delay of d_c for both order pickers that cross, while the delay for being close is equal to d_l for both order pickers. Then the interaction costs for some solution $\sigma_a \in \mathcal{S}_a$ are defined as

$$\mathcal{I}(\sigma_a) = \sum_{b \in \mathcal{M} \setminus \{a\}} \left[d_c \cdot N_{\text{cross}}^{(\sigma_a, \sigma_b)} + d_l \cdot N_{\text{loc}}^{(\sigma_a, \sigma_b)}\right]. \tag{5.9}$$

Finally, let $\mathcal{L} = \{\sigma_1, \ldots \sigma_m\},\ \sigma_i \in \mathcal{S}_i,\ \forall i \in \mathcal{M}$ be a set of solutions to m warehouse TSPPD problems with order picker interaction. Then $\mathcal{I}(\mathcal{L})$ is the sum of all pairwise interaction costs between the solutions in $\mathcal{L}$,

$$\mathcal{I}(\mathcal{L}) = \sum_{a \in \mathcal{M}} \mathcal{I}(\sigma_a). \tag{5.10}$$

Notice that the interaction costs are a summation of the pairwise occurred delays

between order pickers. For events when two order pickers interact, this is straightforward and exact. However, there may occasionally be events when three or more order pickers interact simultaneously. This may cause additional delays beyond those accounted for when summing the interactions between each pair of order pickers. Thus our formula for interaction delays may provide an underestimation in some situations. However, an event with simultaneous interactions between three or more order pickers already accounts for more delays than an interaction between two order pickers. Thus, our solution procedure, which aims to reduce interactions, will try to address these events with priority and therefore further minimize the occurrence of these already rare events.

5.3 Hybrid Genetic Algorithm

Hybrid genetic algorithms (HGAs), rely on the repeated generation of sets of solutions, starting from an initial population, which is iteratively altered by crossover, mutation and education functions. At each step the quality of a solution is evaluated and influences the probability of its attributes to survive in the next generation. Considering recent successful applications of HGAs (e.g., Vidal et al. (2012, 2013)) in related studies, hybrid genetic algorithms appear to be a well-suited tool for our problem. Mutation of individuals, i.e., single solutions in a population, allow us to incorporate warehouse-specific characteristics by developing mutation operators that make use of the warehouse layout information. In addition, HGAs allow for the inclusion of sophisticated search methods, as suggested in Theys et al. (2010). A schematic overview of the general procedure of the HGA we developed is given in Figure 5.1. The flow of the HGA can, after creating an initial population, be divided into four parts, which we will describe in detail in the following sections. Extensive preliminary experiments have shown that the actual construction of the initial population has no significant influence on the solution quality. We therefore employ a variant of the nearest neighbor heuristic with some degree of randomness in the sense that subsequent locations are selected randomly, but based on probabilities according to their distance from the current location, i.e., parameterized regret based random sampling.

5.3.1 Update Phase

One of the major concerns in hybrid genetic algorithm design is convergence to local optima. It is therefore essential that promising attributes survive, whether they belong to feasible or to infeasible solutions. To achieve this, we allow a fraction p_{mut} of the

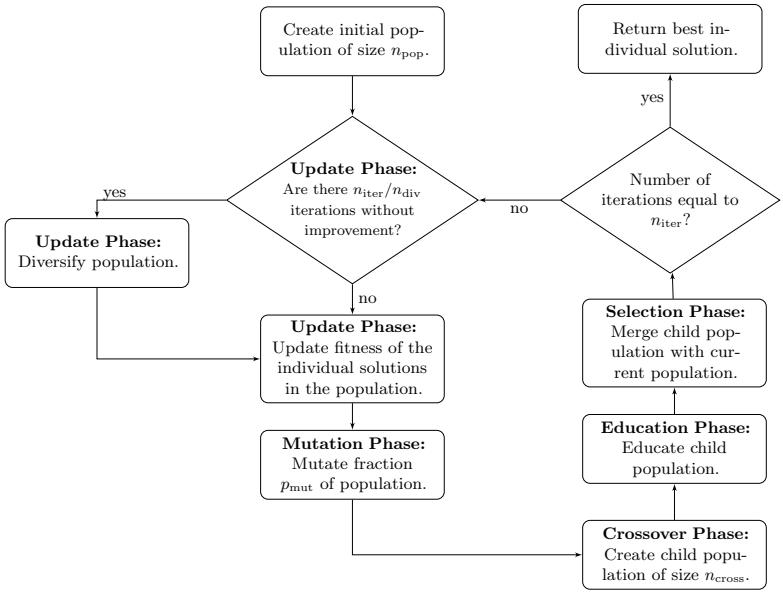


Figure 5.1: Schematic overview of the general flow of the Hybrid Genetic Algorithm.

population to be infeasible.

Besides allowing for infeasible solutions, also a strong diversification procedure is integrated in the HGA. Inspired by the approach of Vidal et al. (2012), if there are $n_{\text{iter}}/n_{\text{div}}$ iterations without improvement, the 90% worst solutions are removed from the population and replaced by newly generated initial solutions.

5.3.1.1 Fitness function

Although we allow for infeasible solutions, infeasibility is penalized. A similar approach is used in most modern genetic algorithms, see Zhao et al. (2009a), Zhao et al. (2009b), Vidal et al. (2012, 2013).

For a solution $\sigma \in \mathcal{S}$, we define $q^{\sigma}_{\max}$ as the maximum load the order picker a any point transports in solution σ. Hence the maximum transport capacity exceedance is given by $(q^{\sigma}_{\max} - q)^{+} = \max\{0, q_{\max} - q\}$. A linear increasing infeasibility penalty $p(i)$, where i denotes the iteration, is used to guide the search. The fitness of a solution $\sigma \in \mathcal{S}$ is the sum of the route length $\ell(\sigma)$ and the penalized transport capacity exceedance, and is given by

$$F(i, \sigma) = (q^{\sigma}_{\max} - q)^{+} \cdot p(i) + \ell(\sigma). \tag{5.11}$$

5.3.2 Mutation Phase

The role of mutation operators for our solution approach is twofold. First, mutations add diversity to the search space of hybrid genetic algorithms. It prevents that the procedure stops at local minima, as crossover alone mostly facilitate the inheritance of attributes which are already available in the population. Second, in the mutation phase we also consider the warehouse-specific characteristics when defining mutation operators, such as sorting partial sequences in an intuitive order with respect to the warehouse layout. This facilitates the creation of attributes which might not be contained in the solutions of the population yet. Next to that, the mutation operators, as described below, exhibit a kind of local focus within single aisles to create potentially promising attributes.

We apply a mutation rate $p_m \in (0, 1)$, i.e., a percentage of the population size, to control the number of solutions which are altered by mutation operators in each iteration. A mutated individual solution thereby replaces the current worst not mutated individual solution from the population, if it is unique. A prioritization of solutions to be selected for mutation is not made; each individual is equally likely to be selected. For the selected individuals one out of three different mutation operators is randomly selected and applied a total of n_{mut} times to the individual. Clearly, a low values for n_{mut} cause less diversity in the population, which may tighten the search. In contrast, any too large value for n_{mut} might cause that mutated solutions are less likely to be selected for crossover, as in the crossover phase solutions are selected based on their fitness value.

With the first mutation operator any aisle that contains locations to be visited is randomly selected. Next, all locations in this aisle are sequenced by their position in the aisle. The direction of sequencing is randomly selected. The resulting partial sequence containing all locations of one aisle is inserted in the complete routing sequence at any point of the original solution at which the corresponding aisle was visited before. By implication, all locations within this aisle are removed from the sequence at all other positions. The second mutation operator is designed in a similar manner. Here, again one aisle that contains locations to be visited is randomly selected. All locations in this aisle are sorted in such a way that the aisle is entered from one side by the picker, all deliveries are performed on the picker's way into the aisle, the picker turns at the farthest pickup or delivery location, and all pickup requests are made on the picker's way back. Again, the side on which the picker enters and leaves the aisle is randomly selected with a uniform distribution. Finally, the third mutation operator randomly selects a visited location. It searches for all locations that are adjacent in the solution and located in the same aisle. This sequence of locations is removed from its current

position in the solution and inserted at a new random position.

5.3.3 Crossover Phase

The Crossover Phase aims to inherit well-performing attributes from the previous generation. solutions of lower fitness value are more likely to contain well-performing attributes. To select such solutions with higher probability, parents are selected by means of a binary tournament selection. It is a frequently used parent selection method, see Vidal et al. (2012, 2013). It consists of choosing two pairs of two solutions at random, thereby selecting from each pair the individual solution with the lowest fitness value as a parent for the crossover.

The creation of a new individual solution, referred to as a child solution, from two parents can be described as follows. One location to be visited is randomly selected. Up to this location all locations are inserted in the new individual in the same sequence as in the first parent. All remaining locations are added to the new individual in the sequence in which they appear in the second parent. The crossover procedure is illustrated in Figure 5.2. Doing so, two potentially well-performing individuals are re-combined in a way that maintains advantageous partial sequences. Obviously, the crossover as explained so far would allow only little variation in the beginning of sequences, as the first part (of random length) of an individual would always be adopted by the children from the first parent. To prevent this effect we apply the crossover procedure either by starting at the beginning or at the end of the routing sequence. The selection of the direction is determined randomly beforehand for each newly created child.

This crossover procedure is used to construct a population of child solutions, called

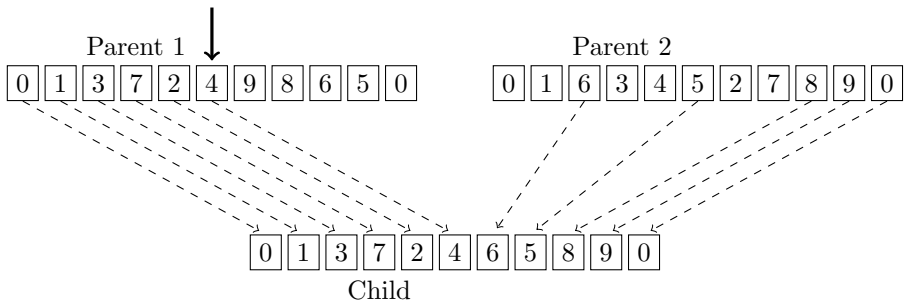


Figure 5.2: Illustration of the crossover operator. The solutions are represented by an array of product indices. The first 6 elements are selected from the first parent. Then from left to right, the products not already copied to the child are added from parent two.

child population. Its size is controlled by the crossover rate p_{cross}, i.e. the number of new solutions entering the child population in each iteration.

5.3.4 Education and Selection Phase

The child solutions are at this stage typically of relative low quality although they may contain promising attributes. To increase the probability that these attributes survive, all solutions of the child population are educated. We thereby increase the quality of the child solutions, which is relative beneficial for child solutions that contain promising attributes. Increasing child solutions' quality by means of local search seems to be standard today, see, among others, Vidal et al. (2012). We therefore choose to let each child solution be subject to a fixed number n_{educ} of randomly chosen education operators, i.e. classical local search operators.

The first education operator is *Swap*. It considers in random order the exchange of two products, i.e. a possible move, in some child solution. As soon the operator finds a possible move that results in a beneficial change of the child solution's fitness, it is applied and the operator terminates. The second and third education operators are *Relocate* and *2-Opt*. They respectively relocate a single location and reverse a part of the sequence of location visits. Again, if such a randomly chosen possible move is beneficial, it is applied and the operator terminates. Since the operators apply moves based on changing fitness values, a child solution may still be infeasible, or may become infeasible, at the end of the education phase. This is no issue, since infeasibility is penalized more for increasing iterations. Therefore the education phase will produce relatively more feasible child solutions towards the end of the HGA.

Finally, the child solutions need to be merged with the parent solutions. There are two guidelines for merging these solutions. First, the fraction of infeasible solutions should not exceed its maximum p_{inf} and second, the fittest n_{pop} solutions should be selected to form the population for a new iteration.

5.4 Hybrid Genetic Algorithm with Interaction Effects

Order picker routing and order picker interaction, as defined in this paper, are not simultaneously studied before. By adopting the general flow of the HGA, as presented in Section 5.3, and extending it by solving multiple warehouse TSPPDs simultaneously, thereby including order picker interaction, a new HGA is developed, called Hybrid Genetic Algorithm with Interaction Effects (HGA-I).

Order picker interaction is already quantified in Section 5.2.2. To allow for simultaneous solving of m warehouse TSPPDs, we store the information of a set of solutions $\mathcal{L} = \{\sigma_1, \ldots, \sigma_m\},\ \sigma_i \in \mathcal{S}_i$. The population maintained by the HGA-I therefore consists of n_{pop} sets of solutions. To calculate the interaction costs $\mathcal{I}(\mathcal{L})$, the routes corresponding to the solutions need to be traced.

The array $\rho(\sigma_i)$ is filled with location-time pairs. Each pair consists of a location in the warehouse and the actual time it is visited by order picker $i \in \mathcal{M}$, if the order picker travels according to solution $\sigma_i \in S_i$. All relevant locations to detect order picker interaction are contained by $\rho(\sigma_i)$. To be precise, all product visits and locations where the order picker turns, i.e. leaving an aisle or turning within the same aisle, are stored. In addition, every δ_{time} between two products visits or turning points the location and actual time are stored in $\rho(\sigma_i)$ as well. However, this implies that the location-time pairs for different order pickers can slightly differ in their time dimension. To overcome that problem, a bandwidth δ_{band} is introduced. If for some $a \in \rho(\sigma_a)$ and $b \in \rho(\sigma_b)$, the difference in time is less than δ_{band}, a and b are assumed to happen simultaneously.

The fitness function for a set of solutions $\mathcal{L}$ becomes

$$F(i, \mathcal{L}) = \sum_{\sigma \in \mathcal{L}} \left[(q^{\sigma}_{\max} - q)^{+} \cdot p(i) + \ell(\sigma) \right] + \mathcal{I}(\mathcal{L}), \tag{5.12}$$

where $I(\mathcal{L})$ is as given by Equation 5.10 and $p(i), q^{\sigma}_{\max}$, and q are as given in Section 5.3.

The initial population generation is slightly changed to handle sets of solutions. Solutions are produced identically as in the HGA. After creation they are grouped to form a set of solutions and interaction costs are calculated afterwards. The Mutation Phase is unchanged. If a set of solutions is selected to be mutated all solutions are subject to mutation.

Parent selection is still performed according to binary tournament selection, although the flow of the Crossover Phase itself is adapted. Crossovers between two parent sets of solutions only alter one of the solutions in the set; all other solutions in the set remain the same. Two parents may generate two unique child solutions. See Figure 5.1 for an illustration.

The Education Phase is slightly adapted compared to the case without interaction costs. Education is applied to every solution from a particular set of solutions. The education operators itself are not changed, implying that interaction costs are not updated during the exploration of the neighborhood. Finally, the Selection Phase is left unchanged.

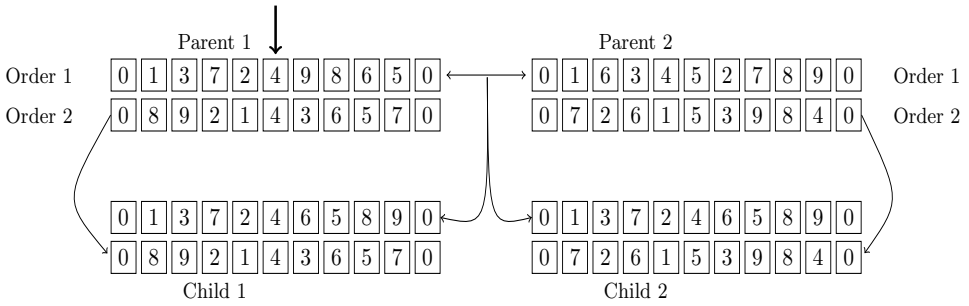


Figure 5.1: Illustration of a crossover between two sets of solutions of size 2. The first order is selected for actual crossover, thereby replacing the 'old' solutions of the first order, while the second order is simply copied to the child solutions.

5.5 Numerical experiments

First, we provide insights in the performance and applicability of the HGA by comparing its computational results with optimal solutions and a simple local improvement heuristic. Lastly, we compare the HGA solutions (without taking interaction effects into account) to the HGA-I solutions (taking interaction effects into account). This will show that order picker interaction is significant and should be taken into consideration when determining routes.

Our solution approach is applicable to a variety of warehouse layouts. Particularly, the HGA is independent of the length, alignment, and number of storage aisles. We conduct our experiments in either a *small* warehouse, with $n_{\text{aisle}} = 7, a_{\text{length}} = 12, a_{\text{width}} = 2.5$, or a *large* warehouse, with $n_{\text{aisle}} = 15, a_{\text{length}} = 32, a_{\text{width}} = 2.5$. Pick and delivery locations are assumed to be solely in the parallel aisles and not in the cross aisles. Locations in the aisles are assigned with a 0.1 meters accuracy and random storage is assumed. Capacity and transported loads are measured for unit-size products. For each location there is 1 unit of products to be picked or delivered. Extensive parameter calibration experiments showed that the following parameter values produce the highest quality solutions: $n_{\text{iter}} = 1500, n_{\text{pop}} = 100, n_{\text{cross}} = 100, p_{\text{surv}} = 0.3, p_{\text{mut}} = 0.05, n_{\text{mut}} = 1, p_{\text{inf}} = 0.05, n_{\text{educ}} = 6$ and $n_{\text{div}} = 5$.

In our preliminary experiments we looked at the effect of allowing infeasible solutions at a penalty cost against not allowing them. Allowing infeasibility results in a bigger search space and consequently in slower convergence. Compared to not considering infeasible solutions the solution quality is on average not significantly different for the majority of the instances. We however found particular difficult instances of which the solution quality improved strongly by considering penalized

infeasible solutions.

5.5.1 Performance of the Hybrid Genetic Algorithm

The performance of the HGA is tested by using 18 instance sets that consist of 100 randomly generated instances each. These can be interpreted as either the result from batching methods or as completely new customer orders. The instance sizes vary from 20 to 100 and have an equal number of pickups and deliveries, since this results in instances that are hardest to solve. For instances until size 40, optimal solutions could be obtained by means of CPLEX 12.5. A local search heuristic starting from a S-shape solution, called SLS, serves as upper bound in the experiments. This heuristic works as follows. Using S-shape routing the picker traverses each aisle containing pickups or deliveries entirely. Pickup locations where in this route the capacity constraint would be violated, are skipped and the order picker returns to collect these pickups as soon as transport capacity suffices. After a solution is constructed it is improved by applying the *2-Opt* operator, see Section 5.3.4, until convergence. It can be seen as the pickup-and-delivery counterpart of the 'S-shape + 2-Opt' heuristic as presented by Theys et al. (2010).

All methods are coded in C++11 and experiments are performed on an Intel Core i5-2400, 3.10 GHz processor. Solutions of the SLS are obtained within a second. The GA delivered results within 10 seconds for the smaller instances and within 2 minutes for the larger instances. Note that the final solutions are often reached within several seconds; a tuning of parameters to the specific warehouse layout at hand will therefore suffice to ensure the algorithm is fast enough for practical purposes. The calculation times for optimal solutions obtained through CPLEX varied from a few seconds for the instance sets 1 and 2 until 15 hours for instance set 7. All results are presented in

Table 5.1: Performance of the HGA heuristic, compared with optimal solutions and the SLS heuristic.

Set	n_{aisle}	a_{length}	n	q	OPT	HGA	difference HGA - OPT	SLS	difference SLS - HGA
1	7	12	20	10	103.05	103.12	0.07%	112.07	8.68%
2	7	12	20	15	101.51	101.58	0.07%	104.85	3.29%
3	7	12	30	15	111.06	111.08	0.01%	123.85	11.50%
4	7	12	30	20	109.32	109.32	0.00%	111.12	1.65%
5	7	12	40	20	113.71	113.76	0.05%	130.69	14.88%
6	7	12	40	30	112.15	112.15	0.00%	114.55	2.14%

Tables 5.1 and 5.2.

The numerical results give insights in the performance of the HGA. As can be seen in Table 5.1, the HGA has an average gap to optimality of less than 0.1%, and can therefore be considered competitive by the current standard for meta-heuristics. The results are based on a single run of the HGA, and instances not solved to optimality with the HGA could typically be solved to optimality with just one extra run (not shown in the tables). The results for larger instances can be found in Table 5.2. The performance of the SLS heuristic is twofold. First, for instances where the transport capacity is not very restrictive, it is at most 3.29% worse than the HGA. Second, the instances with more restrictive transport capacity show an average gap to the HGA between 8% and 16%. It is noticeable that the gap between the SLS and the HGA is comparable for small and large instances. We therefore are inclined to conclude that the HGA continues to produce high quality solutions for larger instances.

5.5.1.1 Practical implications

In order to determine the best possible practical application of our solution approach we performed a second set of experiments to analyze the most suitable composition of pickup and delivery requests in the routes. Clearly, the total number of delivery requests will typically be lower in e-commerce settings than the number of picking request. Warehouses in online retailing are facing return rates ranging from 18 to 74%, see Mostard, De Koster, and Teunter (2005), depending on the product category

Table 5.2: Performance of the HGA heuristic in comparison with the SLS heuristic.

Set	n_{aisle}	a_{length}	n	q	HGA	SLS	difference SLS - GA
7	7	12	50	25	117.08	135.84	16.02%
8	7	12	50	35	114.51	117.36	2.48%
9	15	32	60	30	502.60	555.55	10.53%
10	15	32	60	40	498.91	510.53	2.33%
11	15	32	70	35	514.73	580.23	12.73%
12	15	32	70	45	511.06	523.11	2.36%
13	15	32	80	45	528.76	590.65	11.70%
14	15	32	80	50	524.96	537.15	2.32%
15	15	32	90	45	537.80	609.97	13.41%
16	15	32	90	55	533.92	545.53	2.17%
17	15	32	100	50	543.20	623.68	14.82%
18	15	32	100	60	539.54	551.69	2.25%

Table 5.3: Cases for the route composition of deliveries versus pickups.

Case	Total Number of batches	Pick Batches	Delivery Batches	# Mixed Batches	Mix Batch #pickup - # deliveries
1	160	80	0	80	30 - 30
2	160	64	0	96	30 - 25
3	160	40	0	120	30 - 20
4	160	0	0	160	30 - 15
5	176	0	16	160	30 - 12
6	208	0	48	160	30 - 6
7	240	160	80	0	-

and return opportunities. The dataset that was used for the following experiments consists of 7200 locations to be visited in total, of which one third (i.e., 2400) are delivery requests. The goal of these experiments is to determine the best combinations of pickup and delivery requests in routes. We aim to find out whether the delivery requests should be distributed only over a few routes, or whether an even distribution of deliveries over all picking routes is more advantageous. For these experiments the large warehouse layout was used. The locations of the 7200 pickup and delivery requests were assigned randomly with a uniform distribution. Here, the capacity of the picking device is set to 30. Seven cases for the composition of deliveries versus pickups were computed, which are characterized in Table 5.3. The cases vary between a full integration of pickup and delivery requests in one half of all routes, while the second half of the routes contains pickup requests only (case 1) and a completely separated processing of pickup and delivery requests in 240 routes, each containing 30 requests (case 7).

The results of these experiments are presented in Figure 5.1. Full integration of pickups and deliveries resulted in the shortest total travel distance (case 1, 70.30km). However, we observe that a distribution of delivery requests over more routes does not affect the results significantly, as for case 2 (70.87 km), case 3 (71.13 km), and case 4 (71.86 km). For practical reasons a distribution of deliveries over more routes might still be advantageous to allow for some flexibility for the order picker in sorting products in the picking cart. The use of the HGA contributes to these observations, since it is able to come up with high quality routing solutions with product returns. In contrast, we find significant differences in the resulting travel distance for all cases in which (part of the) deliveries are processes separately. While in case 5 (76.72 km) and case 6 (84.52 km) still some pickup and delivery requests are performed in the

same routes, in case 7 deliveries and pickup requests are entirely separated which leads to an overall travel distance of 91.87 km. This clearly shows that the integration of pickup and delivery requests can significantly reduce travel distance. Our experiments show savings of 23.48% between the cases 1 and 7.

5.5.2 Performance of the Hybrid Genetic Algorithm with interaction effects

The HGA provides us with high quality solutions and showed its practical relevance. However, interaction effects were not yet taken into account. We now set out to test whether interaction effects can be mitigated by means of our HGA-I heuristic. To be able to quantify interaction effects, additional assumptions about warehouse properties must first be made. These are presented in Table 5.4. We use the same 18 instance sets as before, each consisting of 50 or 25 problems for respectively $m = 2$ or $m = 4$ order pickers. Parameters are again calibrated, but no significant differences with the results of the earlier calibration are obtained. We therefore use the same parameter settings.

We first determine order-picker interaction using the HGA for the 18 instance sets. This will provide a reference point that shows the amount of interactions when routing decisions are not yet adjusted to avoid interactions. Each problem is solved by the HGA, and afterwards the solutions are merged into groups of size m, the number of order pickers that simultaneously start processing their orders. For each set of solutions, the interaction delays that arise from synchronously processing m orders are calculated afterwards. These are added to the original route durations of the m order pickers, resulting in a single objective. These results are presented as "HGA" in Table 5.5. Secondly, we run our HGA-I to simultaneously determine routes for all m order pickers, while aiming to minimize total route time, including interaction delays.

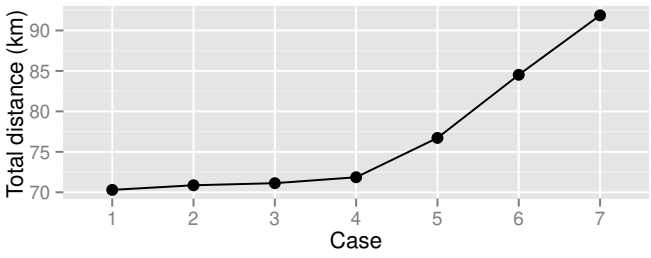


Figure 5.1: The resulting total traveled distance for the different batch composition cases.

These results are presented as "HGA-I" in Table 5.5. Note that, though the instances are the same, the objective values do not directly correspond to the values in Tables 5.1 and 5.2 since these are given in meters, whereas the results in Table 5.5 are for obvious reasons presented in seconds.

When determining routes in isolation, the percentage of total route time that is due to interaction delays varies between 0.5% and 3.4% in the $m = 2$ order pickers case, and between 6.3% and 15.8% in the $m = 4$ order pickers case. It is therefore evident that interaction effects increase strongly with an increase in the number of order pickers. The smaller warehouse (cases 1-8) gives higher interaction delays (13.4-15.8% for $m = 4$) than the larger warehouse (6.3-7.9% for $m = 4$). This is not surprising, since the probability that order pickers interact in a large warehouse is lower than in a small warehouse.

As can be seen from Table 5.5, the HGA-I is capable of decreasing the total route time significantly (refer to column "difference"). For two simultaneously starting order pickers, including order picker interaction considerations in the solution procedure results in up to 2.58% lower total route times. The relative decrease in objective value for the instances located in the larger warehouse is less than 1%, which again shows that in a relatively large warehouse with only two order pickers the interactions are not that significant. However, in the smaller warehouse with four order pickers, the HGA-I reduces total route time by up to 14.4%, which is due to a large decrease in interaction delays at the expense of only a slight increase in travel time. In depth analysis of individual routes showed that many of the HGA-I solutions are constructed without any conflicts between order pickers. This shows that in an environment with multiple pickers it may be very useful to use a routing method that can take order-picker interactions into account.

Table 5.4: Warehouse properties for situations with interactions between order pickers.

Travel speed cross-aisle in m/s	s_{cross}	=	0.7
Travel speed pick-aisle in m/s	s_{par}	=	1.3
Item processing time in seconds	s_{pick}	=	20
Penalty for order pickers' crossing in seconds	d_{c}	=	2
Penalty for order picker's being close in seconds	d_{l}	=	1
Bandwidth for space in meters	δ_{loc}	=	1
Bandwidth for time in seconds	δ_{band}	=	1
Step size for discrete approximation in seconds	δ_{time}	=	1

Table 5.5: Total route times including interaction delays when making routing decisions without taking interaction effects into account (HGA) and with taking interaction effect into account (HGA-I) for $m = 2$ and $m = 4$ order pickers. The total route times (obj.) and the total interaction delays (ID), which are part of total route times, are given for each instance set. The column "dif." indicates the improvement in total route time when taking interaction effects into account for determining routes.

	$m = 2$					$m = 4$				
	HGA		HGA-I			HGA		HGA-I		
Inst.	obj.	ID	obj.	ID	dif.	obj.	ID	obj.	ID	dif.
1	204.30	5.80	200.05	0.68	2.08%	458.20	61.20	412.58	9.76	9.96%
2	200.94	5.24	195.74	0.00	2.58%	451.88	60.48	398.71	6.24	11.76%
3	216.61	5.80	213.19	0.68	1.58%	492.88	71.20	440.86	9.52	10.55%
4	212.62	4.84	207.84	0.00	2.25%	481.17	65.60	422.99	5.68	12.09%
5	221.73	7.08	217.52	1.20	1.90%	504.40	74.88	448.56	9.60	11.07%
6	218.84	6.72	212.15	0.00	3.06%	497.69	73.44	431.45	5.28	13.31%
7	225.16	5.84	221.53	0.76	1.61%	526.23	87.68	456.99	9.04	13.16%
8	223.38	7.56	215.83	0.00	3.38%	512.84	81.20	438.76	4.96	14.44%
9	873.91	4.80	872.21	0.44	0.19%	1857.97	119.52	1772.20	15.12	4.62%
10	868.43	6.80	861.63	0.00	0.78%	1848.22	124.96	1730.84	6.32	6.35%
11	895.66	7.80	892.07	1.20	0.40%	1899.72	124.56	1815.68	16.24	4.42%
12	888.16	8.52	879.67	0.00	0.96%	1880.73	121.44	1767.80	7.36	6.00%
13	914.24	5.80	912.51	1.28	0.19%	1958.18	140.48	1862.68	18.56	4.88%
14	906.14	5.00	901.14	0.00	0.55%	1938.36	136.08	1808.84	6.32	6.68%
15	929.85	6.40	929.39	1.04	0.05%	1990.38	144.56	1893.81	17.76	4.58%
16	921.16	6.32	916.84	0.00	0.47%	1953.68	124.00	1837.85	6.48	5.93%
17	937.99	6.20	936.69	1.24	0.14%	2009.73	146.16	1910.31	17.12	4.95%
18	930.41	7.24	923.23	0.00	0.77%	2005.77	159.44	1854.54	6.48	7.54%

5.5.2.1 Sensitivity analysis

The interaction delays that the solutions of the HGA comprise are influenced by the specific characteristics of the order picking system in use. To give more insights in order picker interaction and the effects of specific order picker characteristics on the order picker interaction, a sensitivity analysis is conducted. At first, the effect of the item processing time is surprising. One would expect that a higher item processing time increases the probability that order pickers cross, since order pickers are stationary for a longer time giving more opportunity for other order pickers to cross. The results are, however, the opposite; higher item processing times leads to less order picker interaction. For item processing times $s_{\text{pick}} = 5, 10, 15, \ldots, 40$, the resulting order picker interaction delays between 4 order pickers are plotted in Figure 5.2. To obtain these results we used the problems from instance set 7.

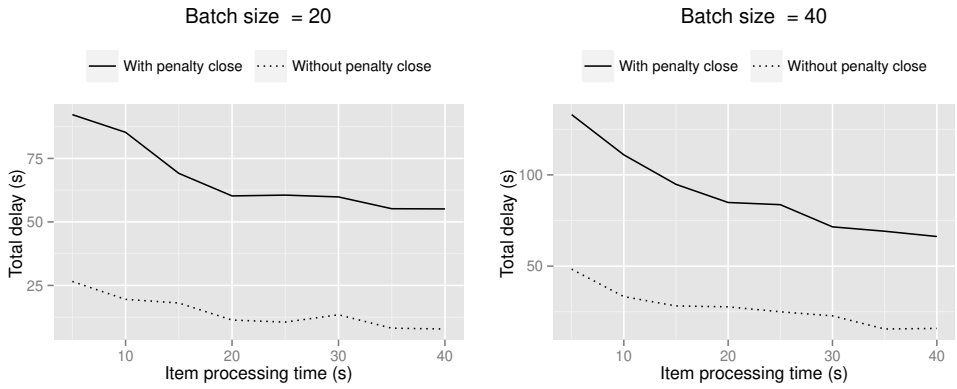


Figure 5.2: Sensitivity analysis of order picker interaction for changing values of the item processing time. For batch sizes of 20 and 40, the results - with and without taking 'close' delays into account - are presented for various values of the item processing time.

To investigate the composition of the order picker interaction delays, we have performed these experiments with and without accounting for delays due to proximity (i.e., order pickers being 'close'). Thus the results without 'close' only show delays for order picker crossing. Thirdly, the same experiments are performed for order picker problems that consists of smaller order sizes. We used the problems from instance set 1. The results are also presented in Figure 5.2. They show the same effect of decreasing order picker interaction for an increasing item processing time.

Concluding, it is shown that order picker interaction needs to be considered in devising routes and that the developed heuristic is capable of constructing routes that are nearly conflict free.

5.6 Conclusions

In this paper we consider a routing problem in e-commerce warehouses for two types of jobs, order picking and restocking of returned products. The inclusion of products that are returned to the warehouse by customers, required reconsideration of the classical warehouse order-picker routing problem. We propose a hybrid genetic algorithm to identify routes by which product returns can be returned to their storage locations, while simultaneously customer orders are picked. In numerical experiments we demonstrate the performance of the solution approach and evaluate the potential gains in travel distance in comparison with a local search heuristic and with optimal solutions. It is shown that the hybrid genetic algorithm yields near-optimal, and

mostly optimal, solutions, and that travel distances are decreased by up to 23.48% when product returns are included in the picking routes, instead of processing product returns separately. In addition, we explored the most suitable manner to integrate product returns in the picking routes and found that an incorporation of many returns in fewer picking routes is slightly preferable over an even distribution of returns over all picking routes.

Since e-commerce warehouses tend to be faced with periods of extreme work loads, which result in a high number of order pickers being employed in the same area, we also set out to investigate the effects of order picker interactions. Furthermore, the integration of restocking activities of returned products in the order picking routes will increase order picker interactions as well, which brought us to investigate both aspects in conjunction. To this end, the hybrid genetic algorithm is extended to include order-picker interaction effects. It is shown that order-picker interaction significantly contributes to the total route time and should be accounted for in solution approaches, especially when product returns are integrated in the regular order picking process.

There are some opportunities for further research. We have discovered some instances in which allowing for multiple depot visits in a single route yielded shorter route lengths, since the picker can drop already picked products at the depot, thus increasing capacity. Regarding order picker interaction, it may be of interest to investigate how storage location assignments influences the significance of order picker interaction. Finally, the possibilities for combining batching methods with our routing methods seems promising. Most of the batching literature assumes simple constructive heuristics for order picker routing, reasoning that this prevents order pickers from interacting. We, however, showed that near-optimal routes can be constructed that are almost interaction free.

Chapter 6

Integration of returns and decomposition of customer orders in e-commerce warehouses

Abstract. *In picker-to-parts warehouses, order picking is a cost- and labor-intensive operation that must be designed efficiently. It comprises the construction of order batches and the associated order picker routes, and the assignment and sequencing of those batches to multiple order pickers. The ever-increasing competitiveness among e-commerce companies has made the joint optimization of this order picking process inevitable. Inspired by the large number of product returns and the many but small-sized customer orders, we address a new integrated order picking process problem. We integrate the restocking of returned products into regular order picking routes and we allow for the decomposition of customer orders so that multiple batches may contain products from the same customer order. We thereby generalize the existing models on order picking processing. We provide Mixed Integer Programming (MIP) formulations and a tailored adaptive large neighborhood search heuristic that, amongst others, exploits these MIPs. We propose a new set of practically-sized benchmark instances, consisting of up to 5547 to be picked products and 2491 to be restocked products. On those large-scale instances, we show that integrating the restocking of returned products into regular order picker routes results in cost-savings of 10 to 15%. Allowing for the decomposition of the customer orders' products results in cost savings of up to 44% compared to not allowing this. Finally, we show that on average cost-savings of 17.4% can be obtained by using our ALNS instead of heuristics typically used in practice.*

This chapter is based on Schrotenboer et al. (2019b):
Schrotenboer AH, Wruck S, Vis IFA, Roodbergen KJ, 2019b *Integrating product returns and decomposition of customer orders in e-commerce warehouses.* Submitted

6.1 Introduction

Technological innovations, high competition, and increasing service requirements necessitate the design and maintenance of efficient customer order processing, preferably with short response times and minimal effort costs (Xu, Allgor, and Graves 2009, Boysen, De Koster, and Weidinger 2019). For the warehouses of leading e-commerce companies such as Alibaba, Amazon or JD, efficient customer order processing reveals the need to adapt order picking activities to be able to handle a large number of typically small-sized customer orders (Boysen, Stephan, and Weidinger 2019). Besides, e-commerce warehouses face a considerable number of product returns, which must be restocked into the warehouse and create additional labor demands (De Koster, De Brito, and Van De Vendel 2002, Mostard, De Koster, and Teunter 2005, Su 2009, Schrotenboer et al. 2017). In this paper, we study a comprehensive order processing problem to provide insights on how picker-to-parts warehouse operations are impacted by these developments. That is, we consider a joint order-picker batching, assignment, sequencing and routing model (see, e.g., Scholz, Schubert, and Wäscher 2017), in which we exploit 1) the small-sized customer orders by allowing the decomposition of customer orders so that the products of a customer order may be picked in multiple batches, and 2) we integrate the restocking of return products in the regular order picking operations. We refer to this practically relevant problem as the Generalized Joint Order Batching, Assignment, Sequencing and Routing Problem (G-JOBASRP).

The two distinct features that we incorporate in current warehouse operations aim to increase its flexibility. First, it is well-known that splitting up customer orders within distribution networks reduces the transportation costs in the complete supply chain (Zhang et al. 2019). In the context of warehouse operations, separating customer orders (i.e., the contained order lines) amongst multiple order picking batches may similarly increase the flexibility of the order picking process. Namely, travel times (within the warehouse) can be shortened leading to a larger warehouse capacity to process customer orders. This may especially be true for e-commerce businesses that face large numbers of relatively small customer orders. We, therefore, investigate if allowing for splitting up customer orders among multiple batches has a cost-saving potential large enough to compensate for the inevitable additional handling operations.

Such additional handling operations due to splitting up customer orders amongst multiple batches are either sorting operations further downstream within the warehouse or additional operations in the last-mile due to multiple shipments to the same customer. However, the split-up of customer orders within picker-to-parts warehouses has not been investigated in the literature. Hence, the trade-off between the gained efficiency

and the required additional handling operations is not known. The model that we present aims to make this quantification by the inclusion of an order split-up cost that represents the costs of additional handling operations in a unified way. That is, the split-up costs can be interpreted as the costs of recollecting the split-up order within the warehouse or as the extra shipping costs incurred by sending multiple shipments to the same customer.

The second distinct feature we add is the integration of the restocking of returned products in the regular customer order processing operations. Prior research has shown that the efficiency of the order picking processes is increased by integrating individual processes. For instance, integrating order-picker routing and batching (Valle, Beasley, and da Cunha 2017), as well as integrating the assignment and sequencing of those batches to multiple order pickers (Henn 2015, Scholz, Schubert, and Wäscher 2017). Jointly studying such processes is also advocated by the recent review on warehouse systems by Boysen, De Koster, and Weidinger (2019). We, therefore, investigate whether the recognized cost-saving potential of incorporating the restocking of returned products in case of a single order picker, as detailed in Schrotenboer et al. (2017), also suffices in comprehensive environments that include order batching, sequencing and assignment.

The integration of both distinct features might increase the efficiency of warehouse operations even further. However, the operational circumstances under which this integration is most profitable are not known. For instance, the split-up of customer orders will simplify the actual routing of order pickers, since batches can consist of products that are in relative proximity of each other. This may lead to a reduction in travel costs (inside the warehouse) and thereby simplifies the integration of the restocking of returned products. On the other hand, the complexity of the order picker routing and its associated travel costs will increase due to the presence of customer order deadlines and the stylized split-up costs. The construction of efficient order picker routes will be a balancing act between the travel costs inside the warehouse on the one hand, and the extent to wich orders are split-up and the associated split-up costs incurred outside our scope. This increased complexity might reduce the efficiency of integrating product returns in the regular order picking operations. Hence, experiments and insights are needed in order to accomplish a successful implementation of such integrated policies in future warehouses.

In this paper, we generalize the Joint Order Batching, Assignment, Sequencing, and Routing problem (JOBASRP) introduced by Scholz, Schubert, and Wäscher (2017). We incorporate the restocking of returned products into regular order picking routes and allowing customer orders to be split-up at the price of incurring split-up costs.

For sufficiently high split-up costs, all order lines of a customer order are assigned to a single batch. Setting the split-up costs to zero equals handling each order line of a customer order individually. We refer to our problem as the Generalized Joint Order Batching, Assignment, Sequencing, and Routing problem (G-JOBASRP). Its goal is to find a cost-minimizing solution for processing a pool of customer orders and product returns with multiple order pickers. Each customer order has a specific deadline before it needs to be processed. The solution determines, for each order picker, a sequence of batches consisting of the products to be picked and returned, while taking into account the actual order picker routes of each batch.

We formalize the G-JOBASRP by developing a compact Mixed Integer Programming (MIP) formulation, which is independent of the actual warehouse layout. We also provide two additional MIP formulations that result from the compact MIP formulation by applying two different Dantzig-Wolfe reformulations. We aim to solve practically sized instances, which typically consists of 1000's of order lines per day for a warehouse (De Koster, De Brito, and Van De Vendel 2002). Therefore, we develop a tailored Adaptive Large Neighborhood Search (ALNS) heuristic (see, e.g., Ropke and Pisinger 2006). It consists of a carefully selected set of operators that build upon concepts from ALNS, Iterative Local Search (see, e.g., Lourenço, Martin, and Stützle 2003), and uses the extended MIP formulations.

Unfortunately, the benchmark instances of Scholz, Schubert, and Wäscher (2017) are, after contacting the authors, not available anymore. We, therefore, provide a new set of practically inspired benchmark instances that are publically available [1]. The benchmark instances consist of up to 5547 to be picked order lines and 2491 to be restocked products. On those instances, the ALNS provides on average 17.4% percent better than an intuitive and practically often observed heuristic. Besides, those instances show that incorporating the restocking of returned products into regular order picker routes results in cost-savings of up to 19%.

Moreover, we analyze the effect of splitting up customer orders among multiple batches at the price of incurring additional split up costs. Maximum cost savings around 45% are obtained due to the increased flexibility while creating order picking batches. Besides, we show that the impact of split-up costs is indirectly observed by an increase in the travel costs associated with picking orders inside the warehouse, as customer order split-ups will then typically be avoided.

The remainder of the paper is structured as follows. We provide a review of selected literature in Section 6.2 and present the mixed integer programming formulations of the G-JOBASRP in Section 6.3. In Section 6.4, we describe the adaptive large

[1] www.albertschrotenboer.com

neighborhood search heuristic in detail. In Section 6.5, we summarize and discuss our numerical results and we conclude and provide suggestions for future research in Section 6.6.

6.2 Literature Review

Warehouse operations, including order picking processes, have received substantial research attention (Rouwenhorst et al. 2000, Gu, Goetschalckx, and McGinnis 2007, De Koster, Le-Duc, and Roodbergen 2007, Roodbergen and Vis 2009, Gong and De Koster 2011, Marchet, Melacini, and Perotti 2015, Staudt et al. 2015, Davarzani and Norrman 2015, Van Gils et al. 2018). Van Gils et al. (2018) offer an extensive overview of research on combining planning problems within warehouse operations. On an operational level, they urge researchers to continue the studying of integrated decision making, and advise the further incorporation of aspects related to e-commerce and globalization. We believe that the G-JOBASRP follows this spirit of incorporating e-commerce aspects in integrated decision making. Namely, we incorporate the restocking of returned products into traditional order picking routes, and we allow customer orders to be split up amongst order picking batches. In the following, we review selected work being relevant or related to the G-JOBASRP. We first review work on (integrated) batching and routing, which is a subproblem of major relevance for the G-JOBASRP. Although its stems from the early 20000's, we judge the discussion to be essential since it forms the basis for recent papers on integrated decision problems within the order picking process.

When order picker routing is the focal problem, the underlying assumption is that the batch (i.e., the set of locations to be visited) is known beforehand. In this case, the problem forms a special case of the traveling salesman problem. Ratliff and Rosenthal (1983) were the first to design an optimal solution approach for the case of a rectangular warehouse with parallel aisles and two cross aisles. Roodbergen and De Koster (2001) extend their approach for warehouses with a middle cross-aisle. However, product returns make an extension of these optimal approaches to more general warehouse situations cumbersome. Hence, also the warehouse routing problem has been approached mostly with heuristic methods. For example, Roodbergen and De Koster (2001) provide a discussion of several routing heuristics for warehouses with more cross aisles. The authors also propose the combined and combined+ heuristics and demonstrate that the latter yields good results compared with S-shape and largest gap routing. Theys et al. (2010) also consider routing methods in multi-parallel-aisle warehouses and opportunities to decrease travel distances by using meta-heuristic

approaches rather than constructive heuristics. Recently, a metaheuristic approach is developed in Schrotenboer et al. (2017) for solving the order picker routing problem with product returns. Their work shows that incorporating the restocking of product returns in the regular order picking routes results in substantial cost-savings. To the best of the authors' knowledge, it is the only work that incorporated the restocking of return products in regular order picking routes. In this paper, we investigate its cost-savings potential in the richer setting of the G-JOBASRP.

Whereas the order picker routing literature assumes predetermined batches, the order batching literature typically assumes fixed routing policies (e.g., S-shape or largest-gap). The design of such order batching policies usually relies on heuristic approaches because of the high combinatorial complexity of this problem. Gademann, Van Den Berg, and Van Der Hoff (2001) prove that the batching problem is NP-hard in the strong sense when the number of products in a batch is greater than two. Gademann and Van De Velde (2005) use a branch-and-price algorithm to solve the batching problem for up to 300 order lines. The batch size, however, was limited to 10 order lines in their experiments, which might be restrictive for large warehouses. Chen et al. (2005) and Ho, Su, and Shi (2008) show that that there is a large potential for sophisticated (metaheuristic) search methods. The so-called savings heuristics form another stream of solution methods for solving the batching problem. They rely on iteratively improving the solution, for instance, by merging batches until no improvement can be found. De Koster, Van Der Poort, and Wolters (1999) provide an overview of such methods and also show that they substantially outperform myopic policies. The integration of product returns in order batching is first discussed by Wruck, Vis, and Boter (2013). In their work, the authors considered this integrated process while assuming fixed routing policies. To the best of the author's knowledge, this is the only work in the batching literature that considers product returns and customer orders simultaneously. It is, therefore, interesting to study this in the richer environment of the G-JOBASRP.

The ever-increasing competitiveness of e-commerce companies causes an increased interest in integrated decision making within the order picking process. Tsai, Liou, and Huang (2008) offer a first step toward the integration of the batching and routing problem for a specific warehouse layout by developing a genetic algorithm (GA). To find an efficient batch partitioning, they use an outer GA to identify the batch formation, and then an inner GA to evaluate the quality of these batches by optimizing the route. An exact algorithm for solving the joint order picker routing and batching problem has recently been proposed by Valle, Beasley, and da Cunha (2017).

In Henn and Schmid (2013), the joint order batching and sequencing problem

To be picked:

Cust.	Deadline	order line(s)	batch
1	10:15	6 × i-23	**A**
	10:15	2 × i-24	**A**
	10:15	1 × i-08	**B**
	10:15	9 × i-26	**A**
2	9:45	1 × i-09	**B**
	9:45	3 × i-04	**B**
...	...	...	...
8	11:00	6 × i-98	**I**
8	11:00	3 × i-22	**F**
8	11:00	4 × i-46	**F**

To be returned:

order line(s)	assigned batch
1 × i-09	**E**
2 × i-56	**C**
1 × i-49	**A**

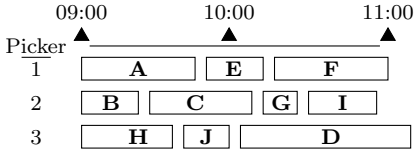


Figure 6.1: Illustrative example of the G-JOBSARP with three order pickers. On the left, a list of customer orders consisting of multiple order lines and the assigned batches. On the right, a possible schedule of batches to be processed by the order pickers. The processing time of each batch depends on the order picker route within the batch.

is studied that penalizes both travel costs and tardiness (collecting customer orders after their corresponding deadline). Metaheuristic solution approaches have shown that cost-savings up to 46% can be attained compared to constructive heuristics. An improvement upon this work is presented by Menéndez et al. (2017). Multiple order pickers are then considered by Henn (2015), and the authors solve the resulting optimization problem with a variable neighborhood search. Instead of considering multiple order pickers, Chen et al. (2015) consider a generalization of the joint order batching and sequencing problem by including the optimization of order picker routes. Scholz, Schubert, and Wäscher (2017) then provide a unified generalization, with both multiple pickers and order picker routing, what they call the Joint Order Batching, Sequencing, Assignment, Routing Problem (JOBASRP). The authors propose a metaheuristic solution procedure and use it to obtain insights into the resulting efficiency of joint optimization. In this paper, we intend to improve this efficiency even further by accounting for two generalizing aspects. First, we include the restocking of return products, which is already shown to be efficient for subproblems of the JOBASRP (Wruck, Vis, and Boter 2013, Schrotenboer et al. 2017). Second, we allow customer orders to be split up amongst batches at the price of a split-up cost. We are not aware of any studies that studied the latter trade-off between splitting up customer orders and the resulting cost increase.

6.3 Problem Definition

In this section, we present a Mixed Integer Programming (MIP) formulation of our Generalized Joint Order Batching, Assignment, Sequencing and Routing Problem (G-JOBASRP).

In the following, we consider *customer orders* that consists out of multiple *order lines*. Each order line consists of a number of the same *products*. A product is assumed to be stored at a single location in the warehouse. The weight of an order line equals the weight of the associated product weight multiplied with the number of times the product is ordered on that order line. We assume that each order line is picked by at most a single order picker, i.e., the products associated to a single order line (that are all equal) are picked in the same batch. We allow the order lines associated to the same customer order to be split-up amongst multiple batches, but incur split-up costs if we do so. To keep notation concise, products that need to be restocked in the warehouse are defined as customer orders of a single order line (possibly consisting of multiple products) with a negative weight.

Customer order deadlines apply to all order lines contained. The deadlines for order lines consisting of returned products are set to occur at the end of the time horizon. The transport capacity of an order picker is limited and cannot be exceeded at any point of a route. Note that due to the inclusion of product returns it is required to keep track of the order picker's capacity along the route. We assume that the travel speed of all order pickers and the time required to pick or return an order line are constant.

The goal of the G-JOBASRP is then to process all the order lines by creating order picking batches and its associated routing, as well as to assign those create batches to an order picker and sequence the assigned batches for each order picker. The objective consists of travel costs (the routing of the batches), tardiness costs incurred if an customer order deadline is exceeded, and split-up costs (representing costs incurred outside our operations) due to splitting the order lines of the same customer amongst multiple batches.

The MIP formulation of the G-JOBASRP is independent of most specific warehouse characteristics, including the layout; only the distances between locations need to be specified. We consider a single depot where all routes of the pickers begin and end. All order lines picked are delivered to the depot and all order lines returned are picked from the depot. The number of available order pickers is given. We detail our notations for the main parameters and decision variables in Tables 6.1 and 6.2.

Before detailing the problem mathematically, let us illustrate the main characteris-

tics and terminology used. We provide an illustrative example of the G-JOBASRP in Figure 6.1. Here, three elements of the G-JOBASRP are presented. On the left, we see a list of *customer orders* (Cust.), with associated *order lines* to be picked (e.g., 6 times product 'i-23'). We assign these order lines to different batches (called A, B, ..., F). It is not allowed to assign, for example, the six times product 'i-23' (order line 1) to multiple batches. On the top right, we see several order lines that require restocking (e.g., one times product 'i-09'). A solution to this example is provided on the bottom right, where we scheduled the batches A until F among three available order pickers. The length of each batch in this schedule is determined by the actual route the order picker traverses through the warehouse to process the assigned batches.

6.3.1 Mixed Integer Programming Formulation

We model the G-JOBASRP on a undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where the node set $\mathcal{N}$ represents all locations in the warehouse and is denoted by $\mathcal{N} = \{0, \ldots, N\}$, where $0 \in \mathcal{N}$ represents the depot. To be independent from the actual warehouse layout, we let the edge set $\mathcal{A}$ be complete and symmetric.

We consider a set of customer orders $\mathcal{C} = \{1, \ldots, C\}$. Each customer orders $c \in \mathcal{C}$ consists of a number of order lines. We denote the set of all order lines by $\mathcal{I} = \{1, \ldots, I\}$, and we let $\mathcal{I}^c \subset \mathcal{I}$ be the order lines corresponding to customer request $c \in \mathcal{C}$. It holds that $\cup_{c \in \mathcal{C}} \mathcal{I}^c = \mathcal{I}$ and $\mathcal{I}^c \cap \mathcal{I}^{c'} = \emptyset$ for all $c \neq c' \in \mathcal{C}$.

Each customer order $c \in \mathcal{C}$ consists of either products to be returned or products to be picked. Let q_i^r and q_i^p denote the quantity to pick and the quantity to return for each order line $i \in \mathcal{I}$. The weight of the products associated with order line i are denoted by w_i. The distance between any pair of locations $n_1, n_2 \in \mathcal{N}$ is denoted by $d_{n_1 n_2}$. We let $n(i)$ denote the location in the warehouse of the product(s) associated with order line $i \in \mathcal{I}$.

A set of order pickers $\mathcal{E} = \{1, \ldots, , E\}$ is available for processing the customer orders. We consider a continuous time horizon of length T. Order pickers can perform multiple routes through the warehouse, and the order lines processed in a route will be called a batch. The set of batches of order picker $e \in \mathcal{E}$ is then given by $\mathcal{H}^e = \{1, \ldots, H_e\}$. All order pickers need to have finished their batches before the end of the time horizon. The capacity of the device of each order picker equals Q, and will be referred to as the order pickers' capacity. A break between the subsequent routes of an order picker is required, and equals c_{break} minutes. Order pickers are assumed to have a constant travel time v, and processing an order line includes a constant pick (or return) time s_{PICK}.

Table 6.1: Overview of the main parameters of the G-JOBASRP

$\mathcal{C} = \{0, \ldots, C\}$	Set of all customer orders	w_i	Weight of one product in order line i
$\mathcal{I} = \{1, \ldots, I\}$	Set of all order lines	v	Travel speed
$\mathcal{I}^c$	Order lines of customer order c	β	Customer order split-up costs
$\mathcal{E} = \{1, \ldots, E\}$	Set of order pickers	s_{break}	Break length between routes
$\mathcal{H}^e = \{1, \ldots, H^e\}$	Set of batches of picker e	s_{pick}	Pick time per product
$\hat{\mathcal{B}}$	Set of assigned routes	s	Travel cost per time unit
$\hat{\mathcal{S}}$	Set of assigned schedules	l_i	Number of products of order line i
q_i^{P}	Quantity of $i \in \mathcal{I}$ to pick	$d_{n_1 n_2}$	Distance between $n_1, n_2 \in \mathcal{N}$
q_i^{R}	Quantity of i to return	α	Delay penalty per product and time unit
$\bar{d}_i$	Deadline of customer order c	Q	Capacity of picking device
T	Time horizon length		

Table 6.2: Overview of decision variables and accompanying parameters

Compact MIP Formulation (Section 3.1)	
$b_i^{he} \in \{0,1\}$	Equals 1 if order line $i \in \mathcal{I}$ is contained in the h-th batch of picker e
$\tilde{b}_c^{he} \in \{0,1\}$	Equals 1 if order line i is contained in the h-th batch of picker e
$\chi_{ij}^{he} \in \{0,1\}$	Equals 1 if order line j is "visited" after order line i in the h-th batch of picker e
$\psi_{ij}^{he} \in \mathbb{R}_+$	Total load of already picked order lines, transported along arc (i,j)
$\omega_{ij}^{he} \in \mathbb{R}_+$	Total load of order lines to be delivered, transported along this arc (i,j)
$\text{TD}^{he} \in \mathbb{R}_+$	Travel distance of the route of the h-th batch of picker e
$\tau_i \in \mathbb{R}_+$	Tardiness of order line i
$c_i \in \mathbb{R}_+$	Completion time of order line i
$c_i^{eh} \in \mathbb{R}_+$	Modeling variable; completion time of order line i of picker e if it is contained in batch h, 0 otherwise
$\kappa_{ihe}^1,\ \kappa_{ihe}^2 \in \mathbb{R}_+$	Modeling variables for linearizing purposes
$\text{ST}^{he}, \text{CO}^{he} \in \mathbb{R}_+$	Start and completion time of batch h of picker e
Extended MIP formulations (Section 3.2)	
$x_{eh}^{\hat{b}} \in \{0,1\}$	Variable equals 1 if $\hat{b}$ is assigned to batch h of picker e
$\xi_i^{\hat{b}} \in \{0,1\}$	Parameter equals 1 if i is in batch $\hat{b} \in \hat{\mathcal{B}}$
$y^{\hat{s}} \in \{0,1\}$	Equal to 1 if schedule $\hat{s} \in \hat{\mathcal{S}}$ is selected
$\eta_c^{\hat{s}} \in \mathbb{N}_+$	The number of distinct batches with products of customer c
$\xi_i^{\hat{s}} \in \{0,1\}$	Parameter equals 1 if i is in schedule $\hat{s} \in \hat{\mathcal{S}}$.

Each order line $i \in \mathcal{I}$ has a deadline $\bar{d}_i$ before its need to be processed. Deadlines of the order lines consisting of products to be returned are set equal to T. If an order line is processed after the deadline, tardiness costs α are incurred per order line per time unit. If order lines belonging to the same customer order are assigned to multiple batches, split-up costs β are incurred per additional split-up. For example, if three order lines are each processed in a different batch, two times β split-up costs are incurred as the order is split up twice more than minimally needed.

We introduce decision variables χ_{ij}^{he}, ψ_{ij}^{he}, and ω_{ij}^{he} to model the routing decisions, comparable to the formulation provided by Mosheiov (1994). Thus, χ_{ij}^{he} is a binary variable that expresses whether the edge between $i, j \in N$ is traveled by picker $e \in \mathcal{E}$ in batch $h \in \mathcal{H}^e$. In turn, ψ_{ij}^{he} denotes the load that has already been picked, and ω_{ij}^{he} refers to the load of order lines that still have to be returned in this route. Both loads are carried along the arc (i, j); the sum must not exceed the transport capacity Q. The binary variable b_i^{he} describes whether order line i is contained in batch h of picker e, and the binary variable $\tilde{b}_c^{he}$ describes whether an order line of customer c is contained in batch h of picker e.

Then the G-JOBASRP asks for solving,

$$\min \quad s \cdot \sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}} \left(\frac{1}{v} d_{n(i)n(j)} + s_{\text{PICK}}\right) \chi_{ij}^{he} + \alpha \sum_{i\in\mathcal{I}} \tau_i + \beta \sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} (\tilde{b}_c^{he} - 1) \tag{6.1}$$

$$\text{s.t.} \quad \sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} b_i^{he} = 1 \qquad \forall i \in \mathcal{I}, \tag{6.2}$$

$$\tilde{b}_c^{he} \geq b_i^{he} \qquad \forall (c,i) \in (\mathcal{C}, \mathcal{I}^c), h \in \mathcal{H}, e \in \mathcal{E}, \tag{6.3}$$

$$\sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} \sum_{i\in\mathcal{I}} \chi_{ij}^{he} = 1 \qquad \forall j \in \mathcal{I}\backslash\{0\}, \tag{6.4}$$

$$\sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} \sum_{j\in\mathcal{I}} \chi_{ij}^{he} = 1 \qquad \forall i \in \mathcal{I}\backslash\{0\}, \tag{6.5}$$

$$\sum_{j\in\mathcal{I}} \chi_{0j}^{he} \leq 1 \qquad \forall h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.6}$$

$$\sum_{j\in\mathcal{I}} \psi_{ij}^{he} - \sum_{k\in\mathcal{I}} \psi_{ki}^{he} = \begin{cases} q_i^p \cdot w_i \cdot b_i^{he} & \text{if } i \neq 0 \\ -\sum_{l\in\mathcal{I}} q_l^p \cdot w_l \cdot b_l^{he} & \text{if } i = 0 \end{cases} \qquad \forall h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.7}$$

$$\sum_{j\in\mathcal{I}} \omega_{ij}^{he} - \sum_{k\in\mathcal{I}} \omega_{ki}^{he} = \begin{cases} -q_i^d \cdot w_i \cdot b_i^{he} & \text{if } i \neq 0 \\ \sum_{l\in\mathcal{I}} q_l^d \cdot w_l \cdot b_l^{he} & \text{if } i = 0 \end{cases} \qquad \forall h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.8}$$

$$\psi_{ij}^{he} + \omega_{ij}^{he} \leq Q \chi_{ij}^{he} \qquad \forall i, j \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.9}$$

$$\chi_{ij}^{he} \leq b_i^{he} \qquad \forall i, j \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.10}$$

$$\chi_{ij}^{he} \leq b_j^{he} \qquad \forall i,j \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.11}$$

$$\mathrm{ST}^{he} = 0 \qquad h = 1, \forall e \in \mathcal{E}, \tag{6.12}$$

$$\mathrm{CO}^{he} = \mathrm{ST}^{he} + \frac{\mathrm{TD}^{he}}{v} \qquad \forall h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.13}$$

$$st^{he} = co^{(h-1)e} + s_{\mathrm{BREAK}} \qquad \forall h \in \mathcal{H}^e \backslash \{1\}, e \in \mathcal{E}, \tag{6.14}$$

$$c_i = \sum_{e \in \mathcal{E}} \sum_{h \in \mathcal{H}^e} c_i^{he} \qquad \forall i \in \mathcal{I}, \tag{6.15}$$

$$\mathrm{CO}^{he} - c_i^{he} = \kappa_{ihe}^1 \qquad \forall i \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.16}$$

$$c_i^{he} = \kappa_{ihe}^2 \qquad \forall i \in \mathcal{I}, h \in \mathcal{H}^e \forall e \in \mathcal{E}, \tag{6.17}$$

$$\kappa_{ihe}^1 \leq (1 - b_i^{he})M \qquad \forall i \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.18}$$

$$\kappa_{ihe}^2 \leq b_i^{he} M \qquad \forall i \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.19}$$

$$\tau_i \geq c_i - \bar{d}_i \qquad \forall i \in \mathcal{I}, \tag{6.20}$$

$$\chi_{ij}^{he},\ b_i^{he},\ \tilde{b}_c^{he} \in \{0,1\} \qquad \forall i,j \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.21}$$

$$c_i, c_i^{he}, \psi_{ij}^{he}, \omega_{ij}^{he}, \kappa_{ihe}^1, \kappa_{ihe}^2, \tau_i \geq 0 \qquad \forall i,j \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}. \tag{6.22}$$

The Objective (6.1) includes the total travel and pick time costs of all order pickers, as well as the penalty cost α for order lines fulfilled too late. Constraint (6.2) ensures that each order line is contained in exactly one batch. With Constraints (6.3), we ensure the correct modeling of $\tilde{b}_c^{he}$. With Constraints (6.4) - (6.5) we confirm that each location at which order lines need to be picked or returned gets visited exactly once in one route. Constraints (6.6) prohibits multiple visits to the depot within one route, to avoid that the capacity restriction can be eluded. With Constraints (6.7) and (6.8), we keep track of the currently transported load during each tour. This is required as we have both pickups and deliveries which cause the order picker's load to fluctuate within a single tour. Constraints (6.9) limits the transported load at any point of an order picker's route to the maximum transport capacity Q. Constraints (6.10) and (6.11) are required to express that an arc (i,j) can only be traveled if i and j are contained in the same batch. In Constraints (6.12) - (6.14), we express the relationships of the start and completion times of the batches addressed by same order picker. Constraints (6.15) define the completion time of an order line, and Constraints (6.16) - (6.19) are linearized constraints ensuring that the κ variables are correct. Finally, Constraints (6.20) model the tardiness variable τ_i.

To provide insights into the problem complexity, let us first consider the order picker routing problem with integrated order picking and return handling alone. For one order picker and one batch, this problem reduces to a classical Traveling Salesman Problem

(TSP), if the transport capacity of the order picker is large enough (Hernández-Pérez and Salazar-González 2004). Concerning TSP problems for warehouse routing, some previous research studies propose polynomial-time solution approaches for specific layouts (Ratliff and Rosenthal 1983, Roodbergen and De Koster 2001). However, Theys et al. (2010) argue that the problem becomes rather intractable for warehouse layouts with more than three cross aisles. Namely, the time complexity is polynomial in the 'given' number of cross ailes. The inclusion of deadlines does not complicate the decision variant of our optimization problem since deadline exceedances are only penalized. Thus, the decision variant of our optimization program combines several NP-hard routing problems with simultaneous order picking and return processing.

6.3.2 Extended MIP formulations

We employ two different Dantzig-Wolfe reformulations on the above compact formulation (with a polynomial number of variables and constraints) in order to provide a formulation more suitable for solving large-scale instances. Both will be used in the ALNS, as described in Section 6.4.

In the first formulation, we consider the exponentially large sets of order picker routes $\hat{\mathcal{B}}$, which we need to assign to and sequence for the multiple order pickers. Note that with a route $\hat{b} \in \mathcal{B}$ we denote the complete description of an order picker's route, which we need to assign to a batch $h \in \mathcal{H}^{]}$ for some order picker $e \in \mathcal{E}$.

In the second formulation, we consider order picker schedules $\hat{\mathcal{S}}$. A schedule $s \in \hat{\mathcal{S}}$ is a completely described sequence of order picker routes that should be assigned to an available order picker. It is clear that $\hat{\mathcal{S}}$ is of an even larger size than $\hat{\mathcal{B}}$, as each schedule $\hat{s} \in \hat{\mathcal{S}}$ is a feasible assignment of routes $\hat{b} \in \hat{\mathcal{B}}$ to an order picker.

A route $\hat{b} \in \hat{\mathcal{B}}$ has an associated travel time $\text{TT}^{\hat{b}}$. Let $x_{eh}^{\hat{b}}$ be a binary decision variable indicating whether route $\hat{b}$ is scheduled as the h-th batch of order picker e. The parameter $\xi_i^{\hat{b}}$ equals 1 if order line i is processed by route b and equals 0 otherwise. The first dantzig-wolfe reformulation then replaces constraints (6.3) - (6.11) by the integer hull described by $\{x_{eh}^{\hat{b}} \mid \hat{b} \text{ is a feasible route}\}$. That is, we can solve the G-JOBASRP by the following MIP formulation:

$$\min \quad \sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} \sum_{\hat{b}\in\hat{\mathcal{B}}} c^{\hat{b}} x_{eh}^{\hat{b}} + \alpha \sum_{i\in\mathcal{I}} \tau_i + \beta \sum_{i\in\mathcal{I}} \left(\sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} \sum_{\hat{b}\in\hat{\mathcal{B}}} \xi_i^{\hat{b}} x^{\hat{b}e} \; - \; 1 \right) \tag{6.23}$$

$$\text{s.t.} \quad \sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} \sum_{\hat{b}\in\hat{\mathcal{B}}} \xi_i^{\hat{b}} x_{he}^{\hat{b}} \geq 1 \qquad \forall i \in \mathcal{I}\backslash\{0\}, \tag{6.24}$$

$$\sum_{\hat{b}\in\hat{\mathcal{B}}} x_{he}^{\hat{b}} \leq 1 \qquad \forall h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.25}$$

$$\text{ST}^{1,e} = 0 \qquad \forall e \in \mathcal{E}, \tag{6.26}$$

$$\text{CO}^{he} = \text{ST}^{he} + \sum_{\hat{b}\in\hat{\mathcal{B}}} x_{he}^{\hat{b}} \text{TT}^{\hat{b}} \qquad \forall h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.27}$$

$$\text{ST}^{he} = \text{CO}^{h-1,e} + s_{\text{BREAK}} \qquad \forall h \in \mathcal{H}^e \backslash \{1\}, e \in \mathcal{E}, \tag{6.28}$$

$$c_i = \sum_{e\in\mathcal{E}} \sum_{h\in\mathcal{H}^e} c_i^{he} \qquad \forall i \in \mathcal{I}, \tag{6.29}$$

$$\text{CO}^{he} - c_i^{he} = \kappa_{ihe}^1 \qquad \forall i \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.30}$$

$$c_i^{he} = \kappa_{ihe}^2 \qquad \forall i \in \mathcal{I}, h \in \mathcal{H}^e \forall e \in \mathcal{E}, \tag{6.31}$$

$$\kappa_{ihe}^1 \leq \left(1 - \sum_{\hat{b}\in\hat{\mathcal{B}}} \xi_i^{\hat{b}} x_{eh}^{\hat{b}}\right) M \qquad \forall i \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.32}$$

$$\kappa_{ihe}^2 \leq M \sum_{\hat{b}\in\hat{\mathcal{B}}} \xi_i^{\hat{b}} x_{eh}^{\hat{b}} \qquad \forall i \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}, \tag{6.33}$$

$$\tau_i \geq c_i - d_i \qquad \forall i \in \mathcal{I}, \tag{6.34}$$

$$x_{eh}^{\hat{b}} \in \{0,1\} \qquad \forall \hat{b} \in \hat{\mathcal{B}}, e \in \mathcal{E}, h \in \mathcal{H}^e, \tag{6.35}$$

$$c_i, c_i^{he}, \kappa_{ihe}^1, \kappa_{ihe}^2, \tau_i \geq 0 \qquad \forall i, j \in \mathcal{I}, h \in \mathcal{H}^e, e \in \mathcal{E}. \tag{6.36}$$

Here, Constraints (6.24) ensure that all order lines are scheduled and Constraints (6.25) ensures that not more than a single route is assigned at the h-th batch of order picker e. The remaining constraints are similar as in the compact MIP formulation, i.e., they ensure the correct modeling of the start and end times of the assigned routes to the batches.

We perform a further Dantzig-Wolfe reformulation of the above formulation by describing a sequence of routes, or schedule $\hat{s} \in \hat{\mathcal{S}}$. We let $y^{\hat{s}}$ be the binary variable equalling 1 if schedule $\hat{s} \in \hat{\mathcal{S}}$ is selected, and 0 otherwise. The parameter $\eta_c^{\hat{s}}$ indicates how many routes in $\hat{s}$ contain order lines of customer order c, and similarly to the previous formulation, we let $\xi_i^{\hat{s}}$ be equal to 1 if order line i is contained in schedule s, and 0 otherwise. We denote with $\hat{c}^{\hat{s}}$ the costs of selecting schedule $\hat{s}$. These costs include the costs for tardiness and the travel costs,

$$\min \quad \sum_{s\in\mathcal{S}} c^{\hat{s}} y^{\hat{s}} + \beta \sum_{c\in\mathcal{C}} \left(\sum_{\hat{s}\in\hat{\mathcal{S}}} \eta_c^{\hat{s}} y^{\hat{s}} \; - \; 1 \right) \tag{6.37}$$

$$\text{s.t.} \quad \sum_{\hat{s} \in \hat{\mathcal{S}}} \xi_i^{\hat{s}} y^{\hat{s}} \geq 1 \qquad \forall i \in \mathcal{I} \backslash \{0\} \tag{6.38}$$

$$\sum_{\hat{s} \in \hat{\mathcal{S}}} y^{\hat{s}} \leq K \tag{6.39}$$

$$y^{\hat{s}} \in \{0, 1\} \qquad \forall \hat{s} \in \hat{\mathcal{S}} \tag{6.40}$$

In the above formulation, Objective (6.37) minimizes the costs of the selected schedules and customer order split-up costs. Constraints (6.38) ensure that all order lines are contained in the solution. With Constraints (6.39), we ensure that at most K schedules are selected. Finally, Constraints (6.40) model the domain of the variables.

Both Dantzig-Wolfe reformulations presented above can, in principle, be solved with column-generation methods. Namely, we can consider subsets of the set routes and schedules and iteratively create routes and schedules that have a negative reduced cost. When no more batches and schedules of negative reduced cost can be found, we have an optimal solution of the linear relaxation of both the Dantig-Wolfe reformulations. However, considering the practically-sized instance sizes we are interested in (1000's of order lines), it is deemed impossible to solve such formulations to optimality.

However, we will employ both formulations in the Adaptive Large Neighborhood Search, as presented in Section 6.4. Namely, we will store in memory all the promising batches and schedules that are found during the search, and try to solve MIPs for those fixed sets of batches and schedules. Crucial is the selection of batches and schedules during the ALNS run, on which we will elaborate in Section 6.4.

6.4 Adaptive Large Neighborhood Search

Adaptive Large Neighborhoud search, first introduced by Ropke and Pisinger (2006), consists of iterative destroying and repair parts of a solution smartly and efficiently. Ater creating an intial solution, we destroy a solution using so-called destroy operators, according to criteria inspired on the G-JOBASRP structure. The removed parts of the solution, i.e., the order lines to be scheduled and assigned to pickers, need to be reinserted. This is done by repair operators, that typically work on a greedy basis. By smartly combining destroy and repair operators, we aim to find an improved solution.

Only destroying and repairing the solution is insufficient to reach high-quality solutions. After a single destroy and repair operation, the solution is either worsened or improved. In the case of the latter, we consider this solution. When the solution is worsened, it is determined by a simulated annealing criterium (Kirkpatrick, Gelatt, and Vecchi 1983) whether or not we accept this worsened solution. In the early stages

of the heuristic, it is allowed to move to (slightly) worse solutions, while towards the end it is ensured that solutions can only improve.

A traditional ALNS implementation considers destroy and repair operators independently from each other. As the G-JOBASRP is a complex problem with many facets on which improvements are possible, not all the repair and destroy operators can be combined logically with each other. We, therefore, consider fixed pairs of repair and destroy operators that focus on improving one of the many aspects of the G-JOBASRP. We call such a fixed combination of destroy and repair operators an 'operator'.

Second, the ALNS is programmed in parallel to enhance consistency of the outcomes. We employ four different threads, each starting with a different initial solution. Then, the threads run the ALNS (as will be described in the following) independently, except that every n^2_{ITER} iterations all the threads move to the best solution found so far.

During the ALNS, we allow for infeasible solutions with respect to the capacity of the pickers. We penalize capacity violations with a cost that increases over the run of the algorithm, ensuring that at the end, it is sufficiently high so that the final solution is feasible.

Finally, we adaptively select operators. We keep track of the performance of the operators by increasing their so-called score if applying the operator resulted in an improved solution. The adaptive mechanism ensures that operators that have been proven successful are selected more frequently, which is particularly useful if we have high split-up costs.

In the following, we first discuss the basis of our generic insertion heuristic. After that, we discuss the construction of initial solutions, the operators included, and the details on the overall structure. The general flow is depicted in Algorithm 6.2, which is discussed in detail at the end of this section. We give a high-level description of the heuristic and its operators; for implementation details and actual parameter values, we refer to Appendix A.

6.4.1 Generic repair heuristic (CI heuristic)

The basis of every repair operator is a classic Cheapest Insertion (CI) heuristic. Classic CI repairs a solution by iteratively inserting the order line resulting in the lowest cost-increase. As this operation is of time complexity $O(|\mathcal{I}|^3)$, it becomes somewhat limiting for large instances or a large number of order lines to be (re)inserted.

We, therefore, take a similar approach to Schrotenboer et al. (2018a). Namely, we insert the order lines in a fixed sequence. This sequence is determined depending

Algorithm 6.1: The generic cheapest insertion (CI) heuristic.

```
Data: Solution s, sorting criteria c, insertion criteria i, set of order lines U
U ← SortSequenceByCriteria(U, c) ;
foreach v ∈ U do
  | InsertOrderLineByCriteria(s, v, i, op);
end
```

on the goal of the operator. Examples include sorting the order lines to be inserted lexicographically on first their aisle and second their location in the aisle or sequencing them in random order. Given this sorted (on any specified criteria) sequence of order lines, inserting them one-one-by-one is of time complexity $O(|\mathcal{I}|^2)$ plus some fixed (negligible) time for generating the sorted sequence. The pseudocode of this procedure is presented in Algorithm 6.1.

In addition, it is well-known (see, e.g., Ropke and Pisinger 2006) that additional randomness (or penalizing of particular characteristics) during the repair of the solution is suitable for diversification of the search. Therefore, as will be detailed in the following, evaluating insertion locations for the order lines is not for each operator solely based on the actual costs in the resulting solution. Some random costs (called noise) might be included, or particular characteristics of the solution might be penalized. We indicate this clearly when detailing the operators.

In the remainder of this section, we will refer to this generic repair operator as the 'CI heuristic'.

6.4.2 Initial Solution

For each employed thread, we construct two types of starting solutions. We first sort the order lines randomly to ensure ties are broken arbitrarily. Subsequently, we sort the order lines lexicographically on their corresponding deadlines and aisles. Then, we perform the CI heuristic on this list of sorted order lines while we assure that the first batches of all order pickers are filled before the order lines are included in the second batch of any order picker. This results in the first solution.

The procedure for the second solution is equal, except that the lexicographical sort is extended by first sorting on the customer id of each order line. This is motivated by the fact that order lines associated with a customer order are less likely to be split-up amongst multiple batches. This is especially useful if the split-up costs are relatively large.

The initial solution from which the further ALNS algorithm departs is the solution with the lowest objective, for each thread. Preliminary experiments have shown that

the initial solutions differ significantly between the threads, so no additional effort is undertaken to diversify the ALNS at this point.

6.4.3 Operators

Each iteration of the ALNS procedure consists of adaptively choosing an operator. Each operator starts with an equal probability of being selected. If an operator improves the current solution, we slightly increase its probability of being selected in subsequent moves. If the probability exceeds a certain threshold, we reset its probability to its initial value. In this way, we ensure diversity in the search

We employ 11 different operators, each of which address different attributes of the solution, which together cover improvement in each aspect of the multi-faceted objective of the G-JOBASRP. Some of the operators change focus during the ALNS procedure by adjusting their corresponding search region, while the other operators' search space is kept constant. In the following, we discuss each operator

6.4.3.1 Similarity of Batches (Operator 1)

For any pair of batches $\hat{b}_1$ and $\hat{b}_2$, we call these batches similar if the following condition is satisfied:

$$\Big[|\mathrm{ST}_{B_1} - \mathrm{ST}_{B_2}| \leq \mathrm{TOL}_{\mathrm{SIM}} \wedge |\mathrm{CO}_{B_1} - \mathrm{CO}_{B_2}| \leq \mathrm{TOL}_{\mathrm{SIM}}\Big], \tag{6.41}$$

where $\mathrm{TOL}_{\mathrm{sim}} \geq 0$ is a tolerance parameter that is high in the beginning the ALNS and decreases during execution. Thus, any pair of batches are similar if they have similar start times and similar completions times according to the current tolerance value $\mathrm{TOL}_{\mathrm{SIM}}$.

This operator then considers, at random, each combination of two batches (ensuring each batch is selected at most once). If two batches are similar according to the definition mentioned before, we perform a so-called single linkage cluster merging of both batches.

In single linkage cluster merging, we iteratively group the order lines of the two similar batches to arrive at two new batches. We start with the complete set of order lines where each order line defines a group by itself. We search for two groups of order lines $k_1 \in \hat{\mathcal{B}}, k_2 \in \hat{\mathcal{B}}$ that satisfies one of the following conditions (in the presented order)

$$\{k_1, k_2\} = \{k_1,\ k_2 \subseteq \mathcal{K} \mid \exists\ i_1 \in k_1, i_2 \in k_2, c \in \mathcal{C} : i_1, i_2 \in \mathcal{I}^c\},\ \text{or ,}$$

$$\{k_1, k_2\} = \arg \min_{k_1,\ k_2 \subseteq \mathcal{K}} \left\{ d_{n(i_1),n(i_2)} |\ i_1 \in k_1,\ i_2 \in k_2,\ k_1 \neq k_2 \right\},$$

where $\mathcal{K}$ denotes the set of all clusters, $i_1 \in k_1$ and $i_2 \in k_2$ denote an order line in group k_1 and k_2, respectively, and $n(i_1)$ and $n(i_2)$ denote the location of order line i_1 and i_2, respectively. In words, we start with each order line of the two similar batches as a cluster. Then, we iteratively merge two clusters of which the order line are the 'closest' to each other. The cluster merging procedure stops if two clusters are left or if the total load of order lines or returns reaches the transport capacity. In the latter cases, all remaining clusters are merged to a single second cluster. In earlier iterations of the ALNS, this might be impossible as the capacity violation penalty is not sufficiently high yet. Possible remaining order lines are inserted using the CI heuristic.

To obtain a new solution, we specify the sequence of locations to be visited in an S-shape routing plan while respecting the capacity restriction. That is, the locations are visited in an S-shaped route through the warehouse and the orders that would exceed the capacity get skipped, to be picked on the order picker's way back to the depot.

6.4.3.2 Outlier of Order Lines (Operator 2)

This operator focusses on removing outliers from the solution. For each batch in the current solution, we define an outlier as an order line for which long detours are made. A long detour is quantified in the following way. We define a threshold $0 \leq \text{OP}^2_{\text{TOL}} \leq 1$, and count the number of order lines processed in any of the aisles in the current batch. If the fraction of order lines in an aisle is smaller than OP^2_{TOL}, all order lines in the corresponding aisle are marked as outliers.

The operator then searches randomly through all order lines in the current solution. At most $\text{OP}^2_{\text{INT}} * N$ order lines are checked on being an outlier and subsequently removed if it qualifies as an outlier. Here, $0 \leq \text{OP}^2_{\text{INT}} \leq 1$. The insertion is done using the CI heuristic. Here, the sorting criterium is based on the associated aisle of the order lines to be reinserted, and the insertion criterium is the actual insertion costs in the resulting solution.

6.4.3.3 Order line destroy and repair (Operator 3)

Operators 3 is based on classical ALNS moves. First, the operator selects a random number of order lines to remove from the current solutions, uniformly distributed between $\text{OP}^{3a}_{\text{INT}} \cdot N$ and $\text{OP}^{3b}_{\text{INT}} \cdot N$. The removed order lines are then placed in a random

order, and inserted using the CI heuristic.

6.4.3.4 Batch destroy and repair (Operator 4)

Operator 4 works in a similar fashion as Operator 3, only here we remove complete batches from a solution. The number of removed batches is uniformly drawn between 1 and $\text{OP}^4_{\text{INT}} \cdot E \cdot H$. All the order lines associated with the removed batches are placed in a random order, and inserted using the CI heuristic.

6.4.3.5 Aisle destroy and repair (Operator 5)

This classical ALNS move aims to improve the solution by focussing on order lines that are located in the same aisle. We select a random number of batches, uniformly drawn between 1 and $\text{OP}^5_{\text{INT}} \cdot E \cdot H$. We insert the customers based on the CI heuristic. As a sorting criterium, we use the aisle associated with the order line (ties are broken randomly).

6.4.3.6 Customer order destroy and repair (Operator 6)

Operator 5 removes a uniformly randomly drawn number between $\text{OP}^{6a}_{\text{INT}} \cdot C$ and $\text{OP}^{6b}_{\text{INT}} \cdot C$ complete customer orders from the current solution. We then randomly sort the customer orders (with associated order lines grouped), and insert them using the CI heuristic. We ensure to insert all the order lines associated with the same customer order in a single batch. We do not allow for capacity violations while inserting the customer orders, which can lead to an unrepaired final solution. If this is the case, we insert the remaining customersusing the CI heuristic in the complete solution without grouping together the order lines associated with the same customer order.

6.4.3.7 Split customer order destroy and repair (Operator 7)

This operator is identical to Operator 5, but only removes the associated order lines of customer orders that are split among multiple batches. The maximum of such removed customer orders from the current solution is uniformly drawn between $\text{OP}^{7a}_{\text{INT}} \cdot C$ and $\text{OP}^{7b}_{\text{INT}} \cdot C$.

6.4.3.8 Batch variable neighborhood decent (Operator 8)

Operator 8 focusses solely on improving the order picker routes. It employes the relocate, swap, and 2-opt local search operators part of the genetic algorithm by Schrotenboer et al. (2017) in a variable neighborhood decent way. First, relocate

moves are applied until no improvement is found. Then, swap moves are applied until no improvement is found. Finally, 2-opt moves are applied until no improvement is found. If either swap or 2-opt finds an improvement, we restart from the beginning. The operator terminates if all operators are unable to find an improvement. We refer to Schrotenboer et al. (2017) for the details on the implementation of the relocate, swap, and 2-opt local search operators.

6.4.3.9 Tardines (Operator 9)

The Tardiness operator focusses on improving the solution concerning the current tardiness of the solution. Here, tardiness equals the absolute difference between the time the order line is dropped at the depot and its deadline. We randomly check each order line (with a maximum of $\text{OP}^9_{\text{INT}} \cdot N$) for its so-called tardiness. We remove the order line from the solution if the deadline is exceeded or if it is dropped at the depot at least OP^9_{TOL} before its deadline.

We then sort the orders based on their deadline and insert the sorted sequence of customers via the CI heuristic. We additionally penalize insertion in the solution based on the difference in ready-time and the orders corresponding deadline. Namely, if the deadline is respected, we add OP^9_{PEN} times the difference between depot drop time and the deadline to the actual insertion costs. If the deadline is exceeded, we add a sufficiently big penalty to prefer any insertion that respects the deadline. In this way, both an early and a late pick are penalized. Preliminary experiments have shown that penalizing early picks is helpfull for a faster convergence of the ALNS.

6.4.3.10 Random customer removal, sorted insert (Operator 10)

This operator works similarly to Operator 6. We remove a random number of customer, drawn between the same bounds as described for Operator 6. Then, instead of assuring that the customers can be inserted in the same batch, we use as a sorting criterium for the CI heuristic the customer index associated to the order lines.

6.4.3.11 Random batch removal, sorted insert (Operator 11)

This operator is a blend of Operator 10 and Operator 2. We first select several batches uniformly within the same bounds as indicated for Operator 2. Then, we sort the removed order lines based on their customer index and apply our CI insertion heuristic.

Algorithm 6.2: General flow of the ALNS

```
p ← initalizeAdaptiveProb()
s_cur, s_best ← initialSolution()
n_1, n_2, it ← 0
while n_1 < N^1_iter do
    while n_2 < N^2_iter do
        s_cur ← s_best
        applyRandomOperator(p, s_cur)
        updateProb(s_cur, s_best, p)
        if acceptSolution(s_cur, s_best, it) then
            s_best ← s_cur
        end
        n_2 ← n_2 + 1
        it ← it + 1.
    end
    n_1 ← n_1 + 1
    ExchangeInformationBetweenThreads(s_best, p)
    MipBasedImprovements()
end
```

6.4.4 MIP-based improvements

Next to the metaheuristic type of operators as discussed in Section 4.2, we employ two different Mixed Integer Programming based improvement operators based on the Dantzig-Wolfe decomposed models presented in Section 3.

The first MIP operator solely focusses on an optimal assignment of the batches currently part of the solution. It is simply obtained by solving the extended formulation (6.23) - (6.36) with the set of batches $\hat{\mathcal{B}}$ equal to the batches currently in the solution.

The second MIP operator uses information of the complete ALNS run so far. While executing the ALNS, many different solutions with different batches are encountered. We store all the unique batches in the set of batches $\hat{\mathcal{B}}$. For this set of batches, we can solve the extended formulation of the G-JOBASRP. However, preliminary experiments have shown that solving this for the complete set of batches is computationally expensive. We, therefore, select an arbitrary number of batches $\text{OP}^{\text{MIP}}_{\text{INT1}} \cdot E \cdot H$ which we fix in the current schedule. We then solve the extended formulation, which will then assign batches (and start and end times) to the non-fixed batches. This is repeated $\text{OP}^{\text{MIP}}_{\text{INT2}} \cdot E \cdot H$ times.

We perform these MIP based operators in the order presented above. If an operator improves the solution we terminate the execution of all MIP-based operators.

Finally, as will be detailed in Section 6.5, we will use the extended MIP formulations presented in Section 3 to validate the performance of the ALNS. That is, we store all batches and schedules found during execution of the ALNS, and solve the complete extended MIP formulations based on batches and schedules.

6.4.5 Overal heuristic structure

The complete ALNS Procedure is outlined Algorithm 6.2. On lines 1-3, we initialize a vector of adaptive probabilities $\mathbf{p}$, we compute the initial solutions by calling the procedure "initialSolution()", and we initalize some iterators to zero. Four procedures are detailed in Algorithm 6.2. First, the "applyRandomOperator($\mathbf{p}$, s_{cur})" selects an operator based on the probability vector $\mathbf{p}$ and applies that operator on the solution s_{cur}. Second, the "updateProb($s_{\text{cur}}, s_{\text{best}}, \mathbf{p}$)" updates the probability vector based on the result of the operator being applied. The "acceptSolution($s_{\text{cur}}, s_{\text{best}}, it$)" procedure contains the simulated annealing criterion (detailed below) that either accepts or denies a new solution. Fourth, the "exchangeInformationBetweenThreads($s_{\text{best}}, \mathbf{p}$)" procedure exchanges information between the running threads, i.e., it updates the best solution and the adaptive probability vector. Notice that this function is called only if all the threads arrive at it simultaneously.

As denoted in Algorithm 6.2, a simulated annealing criterion surrounds the ALNS iterations. Let it be the current iteration, and let $N_{\text{iter}}^{\max} := N_{\text{iter}}^1 N_{\text{iter}}^2$. Moreover, let z_{best} be the objective value before applying an operator and let z_{cur} be the objective value after applying. If $z^* < z$, we accept the new solution. Otherwise, we accept it with probability

$$\exp\left(-1\frac{(z_{\text{cur}} - z_{\text{best}})/z_{\text{best}} \cdot \alpha}{[1-(i/N_{\text{iter}}^{\max})]^{\beta}}\right).$$

Here, α denotes how much objective values may become worse in order to still are likely to be accepted, and β denotes how fast we aim to let the ALNS converge. Moreover, $N_{\text{iter}}^1 \times N_{\text{iter}}^2$ number of iterations are being run in the ALNS. If N_{iter}^2 iterations have passed, the running threads exchange information: First, each thread continues with the solution that has the lowest objective. Second, the adaptive probabilities are updated by averaging the probabilities of each thread with the probabilities of the thread that provided the best solution.

6.5 Numerical Results

In this section, we will provide insights into two main points. First, we investigate the cost-savings potential of integrating the restocking of (returned) products in the regular order picking process. Second, we investigate the efficiency of allowing the split-up of customer orders among multiple batches. We do this by solving the G-JOBASRP on a carefully constructed testbed of instances.

In the following, we first detail the construction of the benchmark instances used.

We then comment on our preliminary computational campaign to find suitable control parameters of the ALNS. We then solve the new G-JOBASRP instances and provide some comparison with common heuristics (for example assuming fixed routing policies). We then focus on providing insights on the two main points as mentioned before.

6.5.1 Benchmark data sets

We perform our experiments in a rectangular warehouse layout that is commonly used to study order picking performance (Gademann, Van Den Berg, and Van Der Hoff 2001, Gong and De Koster 2009, Schrotenboer et al. 2017). The parallel aisles have length a_{length} and width a_{width}, along with two cross aisles at the beginning and the end of the parallel aisles. Products are stored only in the parallel aisles, not in the cross aisles. We assume that order pickers can reach locations on the left and the right side within an aisle without traveling additional distances. The distance between any two points in the warehouse is then trivially calculated as the warehouse distance between those points.

Unfortunately, the benchmark instances of Scholz, Schubert, and Wäscher (2017) are not available anymore so we could not test the performance of the ALNS on those instances. We therefore, carefully constructed 12 generated sample data sets, that serve as the order and return arrival process of a single working shift of 8 hours for multiple order pickers.

The data sets vary in the number of order lines, the warehouse size, and the number of products to be restocked, as is detailed in Table 6.1. In column 4 in Table 6.1, we denote the percentage of customer returns with respect to the total number of requests. We ensure that a return amount of 30% indicates that 30 of 100 customer requests are product returns, which corresponds to a $(30/70 =)43\%$ actual return rate. Pickers move through the warehouse with a constant pace of 0.7 meters per second, and the time to pick or return one order line equals 8 seconds. In all experiments, travel costs equal 0.54 EUR per minute per order picker, which equates to approximately 32 EUR for an order picker in each hour. Delay costs equal 0.06 EUR per minute per delayed product (similar to Tsai, Liou, and Huang 2008). The capacity of the picking device is 80 kg for all order pickers. Finally, we set the break between two subsequent routes of the same order picker to equal $c_{break} = 5$ minutes.

The number of parallel aisles in data sets $1 - 9$ is 35, with an aisle length of 30 meters. For data sets $10 - 12$, we considered 50 aisles, with a length of 40 meters. The aisle width equals 2.5 meters in all experiments. Deadlines for returned products are the end of the work shift (i.e., 8 hours). Deadlines for customer orders might

Dataset	C	I	Return %	Total weight (kg)	Orders (kg)	Returns (kg)
1	195	598	10%	641.0	578.1	62.9
2	227	640	20%	714.0	579.4	134.6
3	292	864	30%	904.8	528.6	376.2
4	566	1737	10%	1893.3	1764.2	129.1
5	672	2028	20%	2191.9	1766.0	425.9
6	920	2786	30%	2988.8	2097.9	890.9
7	1348	4000	10%	4299.8	3848.7	451.1
8	1706	5049	20%	5647.1	4597.3	1049.8
9	1817	5431	30%	6092.8	4099.4	1993.4
10	2029	6059	10%	6711.6	5947.0	764.6
11	2341	6995	20%	7654.3	6082.5	1571.8
12	2673	8038	30%	8897.9	6181.9	2716.0

Table 6.1: Sample Data Sets

occur during the shift, at every full hour; we assigned them randomly with a uniform distribution. The number of products in an order line can be a maximum of 4, being 1 for 40% of the order lines, 2 for 30%, 3 for 20%, and 4 for 10%, resulting in a mean quantity of two products per order line. The weight of a product is accurate to 0.1 kg with a uniform distribution and a maximum of 1 kg. Similarly, the storage locations of order lines, given by the aisle number and the location in the aisle, accurate to 0.1 meters, are assigned with a uniform distribution. We assumed random storage assignment as it performs well when the number of order lines to be fulfilled is large (Chan and Chan 2011). Finally, the datasets cover product return rates that are typical for e-commerce businesses (Mostard, De Koster, and Teunter 2005, De Koster, De Brito, and Van De Vendel 2002).

6.5.2 Parameter tuning

The performance of the ALNS appears rather robust concerning the set paramaters. We, therefore, briefly describe the parameters used for the results described in the following sections.

We allow $N^1_{iter} = 50$ outer iterations of $N^2_{iter} = 100$ inner iterations, see Algorithm 2. Note that this is predominantly oriented so that the largest instances still have reasonable computation times. For instance, they can be solved in a single night before operations start in the warehouse in the morning. The TOL_{SIM} equals 2000 at the start, and is lowered linearly to 500 at the end of the algorithm. An order line is called an outlier if it is part of an aisle that is used less then 20 % (i.e. $\text{OP}^2_{\text{TOL}} = 0.2$) in a route. Consequently, OP^2_{INT} equals 0.15, indicating that at most 15% of the solution is destroyed.

Then, Operator 3 removes a random number of order lines with $\text{OP}^{3a}_{\text{INT}} = 0.05$ and

$\text{OP}_{int}^{3b} = 0.15$. For Operators 4 and 5 (that remove batches and aisles, respectively), $\text{OP}_{\text{INT}}^{4} = 0.25$ and $\text{OP}_{\text{INT}}^{5} = 0.15$, respectively.

Furthermore, Operator 6 removes a randomly number of customers uniformly according to $\text{OP}_{\text{INT}}^{6a} = 0.05$ and $\text{OP}_{\text{INT}}^{6b} = 0.15$. For Operator 7, this is 0.05 for $\text{OP}_{\text{INT}}^{7a}$ and 0.10 for $\text{OP}_{\text{INT}}^{7b}$.

Tardiness Operator 9 checks at most $\text{OP}_{\text{INT}}^{9} = 0.15$ of the order lines for its tardiness contribution. We set $\text{OP}_{\text{TOL}} = 1000$ and $\text{OP}_{\text{PEN}}^{9} = 0.01$. Then, Operator 10 removes as many customers as Operator 6, and, finally, Operator 11 removes at most the fraction of $\text{OP}_{\text{INT}} = 0.25$ of the batches.

To imitate the probabilities guiding the adaptive aspect of the ALNS, we initialize each operator with a 'score' equal to 1. The distribution of this scores among the operators then guides the operator selection. If a succesfull move is made, we increase the score of the associated operator and the 4 previous operators that resulted into accepted moves with 0.25. If after the inner (and parallel) part of the algorithm a score is larger than 5, we reset it to 1 so as to keep diversity in the choosen operators.

Finally, the results presented in this section are obtained without using the MIP operators. We discuss the (dis)adventages of using these operators in Subsection 6.5.10.

6.5.3 Computation times

To not disturb the reader from the essence of our contribution, namely to show the dynamics of the G-JOBASRP, we do not accompany the detailed results with the computation times of the ALNS. We will, however, give a brief description of the computational aspects in the following.

The instance sizes range from 598 order lines to 8038 order lines. This is significantly larger compared to existing problems in related settings, as for instance, the newest proposed datasets for the (much more stylized and less constrained) capacitated vehicle routing problem range from 100 to 1000 customer (Uchoa et al. 2017). As stated before, we callibrated our parameters so that the largest instances are still solvable in reasonable time, i.e., they can be solved overnight. The ALNS is programmed in C++ and all experiments are run on a Xeon E5 2680v3 2.5 GHz CPU.

Regarding the smaller instances, the computation times range from 313 seconds for Dataset 1 to 1.5 hours for Dataset 6. Datasets 7, 8 and 9 then respectively ask approximately 4.5, 7, and 8 hours for solving the associated G-JOBASPR. Finally, Dataset 12 took around 11.5 hours to solve, which we deemed feasible for practical purposes. If this is, due to the practical circumstances, deemed infeasible, one could

Table 6.2: Results of comparison ALNS with BM1 and BM2 with $\beta = 0$.

Dataset	E	BM1	BM2	ALNS	Dif(%)
1	3	102.7	98.9	66.0	-33.3
	4	102.7	98.5	65.9	-33.1
	5	102.6	98.1	66.0	-32.7
2	3	105.6	102.4	70.4	-31.2
	4	106.3	102.8	69.7	-32.2
	5	106.3	102.6	69.7	-32.0
3	4	120.4	117.9	84.5	-28.4
	5	120.3	117.9	84.2	-28.5
	8	120.3	117.8	84.1	-28.7
4	7	229.2	223.2	175.6	-21.3
	8	230.2	224.0	173.2	-22.7
	9	229.1	223.8	173.1	-22.7
5	8	250.0	244.5	195.6	-20.0
	9	250.4	244.4	195.9	-19.9
	10	250.4	244.7	194.5	-20.5
6	8	312.4	308.6	267.3	-13.4
	9	312.7	308.7	264.6	-14.3
	10	313.4	308.5	264.0	-14.4
7	12	443.0	437.8	387.4	-11.5
	13	441.1	436.1	390.1	-10.5
	14	440.4	435.7	386.9	-11.2
8	14	538.0	531.3	481.7	-9.3
	15	536.2	531.3	481.9	-9.3
	16	537.0	531.8	482.2	-9.3
9	14	549.5	543.7	498.2	-8.4
	15	550.2	544.7	499.0	-8.4
	16	549.3	543.6	501.3	-7.8
10	20	751.5	742.7	661.0	-11.0
	21	749.2	739.9	653.6	-11.7
	22	751.5	742.3	658.1	-11.3
11	20	823.2	813.2	725.2	-10.8
	21	823.2	812.8	730.9	-10.1
	22	822.4	812.6	732.0	-9.9
12	20	899.2	889.7	815.3	-8.4
	21	901.0	889.2	810.0	-8.9
	22	898.0	889.6	813.7	-8.5
Avg.		426.9	421.0	368.7	-17.4

reduce the number of iterations to achieve only slightly worse solutions in shorter computation times.

6.5.4 Benchmark models

In practice, instances of the G-JOBASRP of the size we propose (i.e., up to 8038 order lines) are solved with intuitive constructive heuristics. We, therefore, propose two of such methods to compare the performance of ALNS.

The first benchmark model (BM1) uses a variation of the earliest deadline first (EDF) job scheduling heuristic and organizes the route of each batch using cheapest

feasible insertion. In contrast with job scheduling problems, the G-JOBASRP considers no single jobs (i.e., a batch equals 'a job' in scheduling terminology). Thus, when applying EDF, we need to make an additional decision when an order line is included in an already existing (non-empty) batch or in a new empty batch. This decision involves the consideration of the earliest deadline that occurs in the non-empty batch among already included order lines. Therefore, we consider the order lines to be sorted lexicographically according to first their deadlines and second their aisle. The first order line is included in the first batch of the first order picker. This batch is filled up with the next order lines. We determine a preliminary route by inserting the order lines using cheapest feasible insertion in the sequence in which the order lines are included in the batch.

As soon as an order line is included in a particular batch, we also allow insertion in the same batch of the next order picker. If no such batch is available, we allow insertion of the order line in a new batch of the first order picker.

The second benchmark model (BM2) improves upon BM1 by afterward performing a local search on the created batches. All ties while sorting the sequence of order lines are broken arbitrarily. We, therefore, perform both benchmark models 100 times and select the solution with the lowest objective.

6.5.5 Solutions to the G-JOBASPR instances

We solve the constructed datasets 1-12 with the ALNS as described in Section 6.4. We create three G-JOBASRP instances from each dataset, each having a different number of order pickers. These numbers and the maximum number of 8 batches for each order picker resulted from a preliminary analysis of the instances.

We first compare the performance of the ALNS with the performance of BM1 and BM2, when $\beta = 0$. The results are provided in Table 6.2. A few observations stand out. First, the average cost reduction from using ALNS instead of BM2 equals 17.4%. Second, the instances with a higher fraction of product returns show slightly smaller cost reductions. This decrease is due to dedicated routes with only deliveries becoming efficient for an increasing number of deliveries. Third, the cost-reduction that results from using the ALNS decreases for larger instances. This decrease is, again, due to the structure of the batches for the final solutions. If there are many order lines to pick, the average number of aisles an order picker traverses is rather small. Due to sorting on aisle number and customer deadlines in BM1 and BM2, the structure of the proposed solutions of both benchmark models is rather similar. Hence, for the scenarios where the warehouse is working at its full capacity (e.g., just

Table 6.3: Results of solving the G-JOBASRP benchmark instances

		$\beta = 0$ (split-ups)				$\beta = 10000$ (no split-ups)					dif.	
Set	E	int.	orders	ret.	dif.	int.(1)	int.(2)	orders	ret.	dif.	int.	sep.
1	3	66.0	61.6	12.4	-10.8	106.7	107.5	103.4	12.4	-7.2	-38.6	-36.1
	4	65.9	61.8	12.4	-11.2	107.8	107.5	103.2	12.4	-7.0	-38.7	-35.8
	5	66.0	61.8	12.4	-11.2	107.0	107.5	103.1	12.4	-7.0	-38.6	-35.7
2	3	70.4	60.0	24.0	-16.1	108.3	107.5	96.5	24.0	-10.8	-34.4	-30.3
	4	69.7	59.9	24.1	-17.0	108.0	107.1	95.9	24.0	-10.6	-34.9	-29.9
	5	69.7	59.7	23.9	-16.6	106.2	106.4	94.9	24.0	-10.5	-34.5	-29.7
3	4	84.5	60.5	43.7	-19.0	126.5	126.8	96.8	43.9	-9.9	-33.4	-25.9
	5	84.2	60.5	43.7	-19.1	127.0	124.5	95.8	43.8	-10.8	-32.4	-25.4
	8	84.1	60.3	43.6	-19.1	123.0	123.7	96.7	43.6	-11.9	-32.0	-26.0
4	7	175.6	167.0	23.6	-7.9	335.2	329.7	322.3	23.6	-4.7	-46.7	-44.9
	8	173.2	163.0	23.7	-7.3	332.3	331.8	318.2	23.7	-2.9	-47.8	-45.4
	9	173.1	163.1	23.7	-7.3	331.7	334.6	320.5	23.8	-2.8	-48.3	-45.7
5	8	195.6	165.7	48.6	-8.7	366.0	368.3	331.1	48.7	-3.0	-46.9	-43.6
	9	195.9	166.6	48.7	-9.0	366.6	368.2	327.2	48.5	-2.0	-46.8	-42.7
	10	194.5	165.1	48.5	-9.0	371.2	362.3	329.3	48.6	-4.1	-46.3	-43.5
6	8	267.3	199.8	89.2	-7.5	470.3	462.4	396.0	89.8	-4.8	-42.2	-40.5
	9	264.6	198.1	89.1	-7.9	463.7	460.6	390.2	89.9	-4.0	-42.6	-40.2
	10	264.0	198.1	89.7	-8.3	466.7	464.4	391.5	89.1	-3.4	-43.1	-40.1
7	12	387.4	359.8	50.7	-5.6	740.5	764.0	740.5	50.7	-3.4	-49.3	-48.1
	13	390.1	359.0	50.7	-4.8	764.2	765.8	730.5	50.8	-2.0	-49.1	-47.5
	14	386.9	358.2	50.4	-5.3	770.2	767.8	736.5	50.9	-2.5	-49.6	-48.1
8	14	481.7	413.4	99.9	-6.1	946.0	925.5	857.0	100.0	-3.3	-48.0	-46.4
	15	481.9	413.2	99.4	-6.0	932.8	928.7	857.3	99.3	-2.9	-48.1	-46.4
	16	482.2	413.7	99.9	-6.1	943.1	936.6	854.4	100.3	-1.9	-48.5	-46.2
9	14	498.2	372.0	175.5	-9.0	906.6	909.9	754.9	179.3	-2.6	-45.2	-41.4
	15	499.0	372.1	175.6	-8.9	914.6	901.1	747.5	179.7	-2.8	-44.6	-40.9
	16	501.3	371.1	175.2	-8.2	903.4	906.8	751.0	178.1	-2.4	-44.7	-41.2
10	20	661.0	609.6	91.1	-5.7	1773.2	1606.5	1681.6	91.6	-9.4	-58.9	-60.5
	21	653.6	608.0	90.9	-6.5	1666.3	1582.5	1575.1	91.1	-5.0	-58.7	-58.1
	22	658.1	610.4	91.1	-6.2	1644.1	1553.1	1553.2	90.9	-5.5	-57.6	-57.3
11	20	725.2	623.8	168.1	-8.4	1847.0	1676.1	1674.7	172.2	-9.3	-56.7	-57.1
	21	730.9	621.1	168.3	-7.4	1783.2	1669.9	1625.3	171.8	-7.1	-56.2	-56.1
	22	732.0	624.8	167.7	-7.6	1724.9	1638.7	1585.8	171.7	-6.8	-55.3	-54.9
12	20	815.3	626.2	275.6	-9.6	1948.2	1941.6	1658.3	290.0	-0.3	-58.0	-53.7
	21	810.0	627.1	276.8	-10.4	1889.7	1847.3	1599.0	290.8	-2.2	-56.2	-52.2
	22	813.7	626.9	276.6	-9.9	1850.3	1819.6	1564.5	285.8	-1.7	-55.3	-51.2
Avg.					-9.6					-5.1	-46.3	-43.5

before Christmas), using an intuitive heuristic already performs quite well. However, warehouses are typically not working at their full capacity. Then, using the ALNS instead of the intuitive heuristics will improve efficiency significantly (20-30% cost reductions).

In Table 6.3, we present the results for solving the instances associated with the datasets for $\beta = 0$ and $\beta = 10000$ with the ALNS. Here, the former indicates a situation where customer orders might be split-up without any costs, whereas the latter indicates

a situation in which customer orders cannot be split. All the presented results are the minimum values over ten repetitions. The columns headed by 'int.' denote the costs associated with solving the problem with the product returns integrated with the order picking process. For the case with $\beta = 10000$, we provide two integrated objectives. The costs 'int.(1)' are associated with solving the G-JOBASRP instances with the number of order pickers increased for the integrated problem by the fraction of product returns in the associated instance. The costs 'int.(2)' denote the minimum of 1)solving the G-JOBASRP with the specified number of order pickers and of 2) solving the G-JOBASRP completely separately. The motivation for both cost measures is due to the observation that tardiness costs are included in the larger instances if $\beta = 10000$. In such a case, incorporating a delivery (i.e., an order line of product returns) results in longer travel times and hence more tardiness costs due to its associated product drop time. Hence, for the larger instances (datasets 10-12), it is particularly difficult to find the benefit of the integrated solution (if there is any). Moreover, as we give the same number of iterations to both settings, and the integrated setting consists of significantly larger instances, it also provides a more fair comparison in terms of the quality of the ALNS. In the final two columns, we measure the difference between $\beta = 0$ and $\beta = 10000$, where column 'int.' denotes the differences between the integrated problems (int.(2) for $\beta = 10000$) and column 'sep'. denotes the difference between the separated problems.

The columns headed by 'orders' and 'returns' denote the costs of solving only the order picking problem and the product return problem, respectively. In other words, we solve the G-JOBASRP separately for the pickups and the deliveries. The differences between the integrated and separated problem are denoted (in %) in the column 'dif'.

From the results in Table 6.3, it is observed that splitting up customer orders results into an average cost-saving of 47.7% when product returns are integrated and 43.7 % when product returns are handled separately. Hence, the opportunity to improve efficiency by separating customer orders is very likely for practical situations. This will be evaluated further in Section 6.5.8. The effect of integrating product returns leads to average cost-savings of 9.2 % and 1.9% when order split-ups are not penalized ($\beta = 0$) and are prohibited ($\beta = 10000$), respectively. The former cost-savings are in line with the results for the single picker, single tour scenario as studied by Schrotenboer et al. (2017), whereas the latter cost-savings seem to be marginalized. The performance on the larger instances is driving this marginalization since it is impossible to process all customer orders before their associated deadlines in these instances.

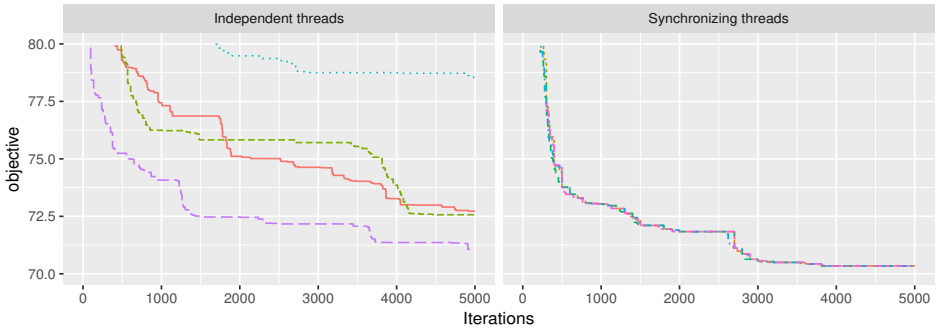


Figure 6.1: Solution trajectory of the ALNS on dataset 2 with 3 order pickers with four threads exchanging information (on the left) and four threads working indepedently (on the right). Each line in each figure corresponds to the current best-known solution found by a single thread.

6.5.6 The effect of multiple threads

As explained in Section 6.4, the ALNS is programmed in parallel using four threads. A simple, though computationally useful feature is that we exchange information between the threads every 100 iterations. In this section, we briefly illustrate the impact of this on the observed solution trajectory.

We compare the ALNS with threads exchanging information and the best-known solution with the same ALNS but now run by four threads on isolation. We focus on Dataset 2 with three order pickers and Dataset 6 with eight order pickers. Here, the former resembles a relatively small G-JOBASRP instance and the latter a relatively large G-JOBASRP instance.

In Figures 6.1 and 6.2, we depict two example trajectories of the best-known solution found while solving these particular instances. The graphs show that using independent threads result in widely dispersed outcomes. For instance, in Figure 2, the final objectives range between 71 and 79 for the independent threads, while the synchronized threads result in a final objective just above 70. This behavior is the primary benefit of using this simple parallel approach; the variance in the outcomes is reduced sharply, which also results in a faster converging of the best-known solution.

At the same time, as is advocated by the results in Figure 3 as well, it is observed that the synchronized threads find a better solution than the best solution among the four independent threads. This improvement feels natural, as more computer power will be spent on current solutions with high potential. For instance, the worst-performing thread in Figure 3 still obtained a quarter of the devoted computer power.

It is, however, likely that the worst-performing thread is trapped in a local optimum and will not reach a new best-known solution.

Hence, we advocate the use of this parallel technique in any metaheuristic. It does not require more resources than there are readily available at any modern computer, and it is easy to implement. It does, however, reduce the variability and increase the quality of the results.

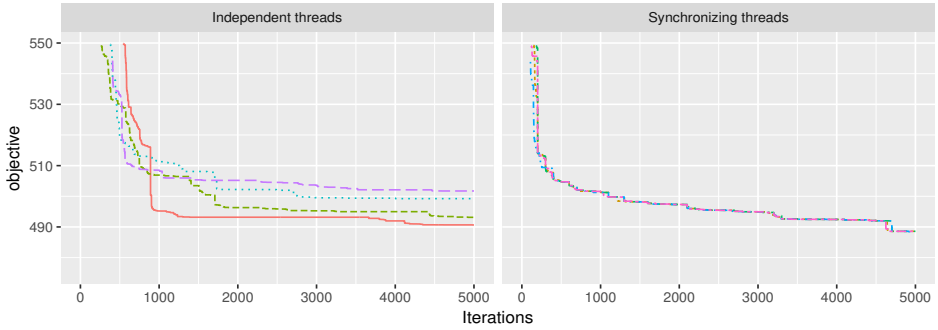


Figure 6.2: Solution trajectory of the ALNS on dataset 6 with 8 order pickers with four threads exchanging information (on the right) and four threads working indepedently (on the left). Each line in each figure corresponds to the current best solution found by a single thread.

6.5.7 Sensitivity of the order picker's capacity

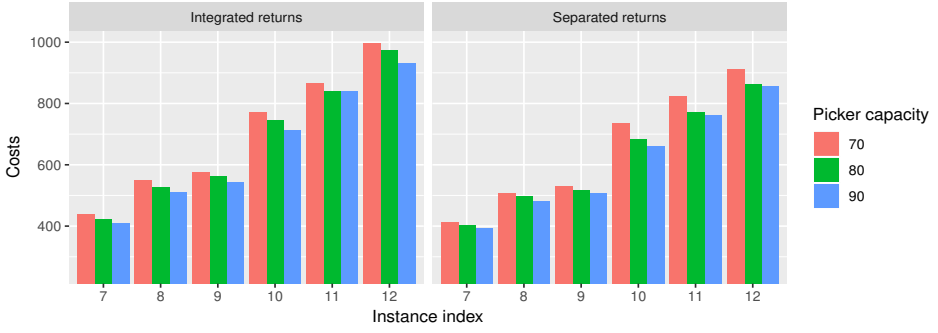


Figure 6.3: Comparison of total costs of the first JOBASPR instances of datasets 7-12 for varying capacity levels. On the left, when product returns are not integrated. On the right, when product returns are integrated.

We furthermore study the effect of reducing and increasing the capacity of the

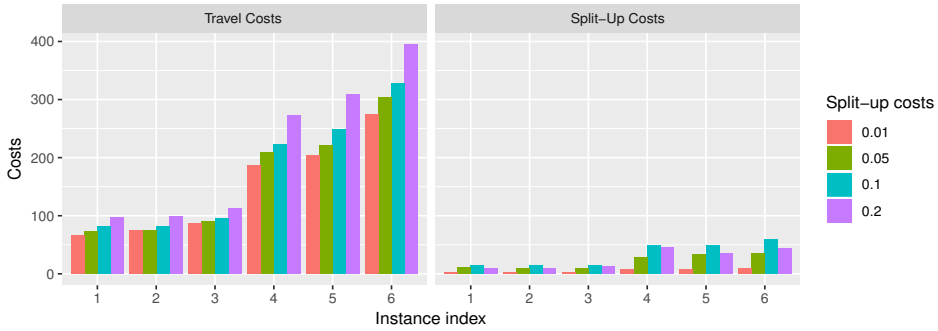


Figure 6.4: The travel and split-up costs for different values of the split-up costs (β). The instance indices 1-6 refer to the first G-JOBASRP instances of dataset 1-6 respectively

order pickers by 10, and its impact on the integration of product returns in the regular order picking process. In Figure 6.3, we depict for the first G-JOBASPR instances of Datasets 7-12, the effect of increasing picker capacity both when product returns are integrated (on the left) and when products returns are handled separately.

First, the total costs decrease (slightly) when the picker capacity increases, which is as expected. However, it appears that increasing picker capacity is slightly more useful for the larger instances, as can be observed in Figure 6.3. This effect seems more pressing for handling the returned products in separation.

Second, the on average cost reduction for integrated processing of product returns equals 6.5%, 8.3%, and 7.4% for an order picker capacity of 70, 80 and 90, respectively. Hence, no conclusion can be drawn on the impact of a smaller or larger picker capacity on the efficiency of handling the product returns integrated with the regular order processing.

6.5.8 Impact of the order split-up costs

In order to provide practical insights under which circumstances it is valuable to organize customer order processing in warehouses with order split-ups, we studied the impact of varying the customer order split-up costs β on dataset 1-6, again on the first instances of these datasets only. The results are presented in Figure 6.4.

What stands out is that split-up costs have a large impact on the total costs of the proposed solutions. However, this is not solely caused by the fraction of costs due to splitting up customer orders. Increasing the order split-up costs result in higher travel costs since orders are increasingly assigned to a single batch if split-up costs increase. We observe in Figure 6.4, that changing the customer order split-up costs from 0.10

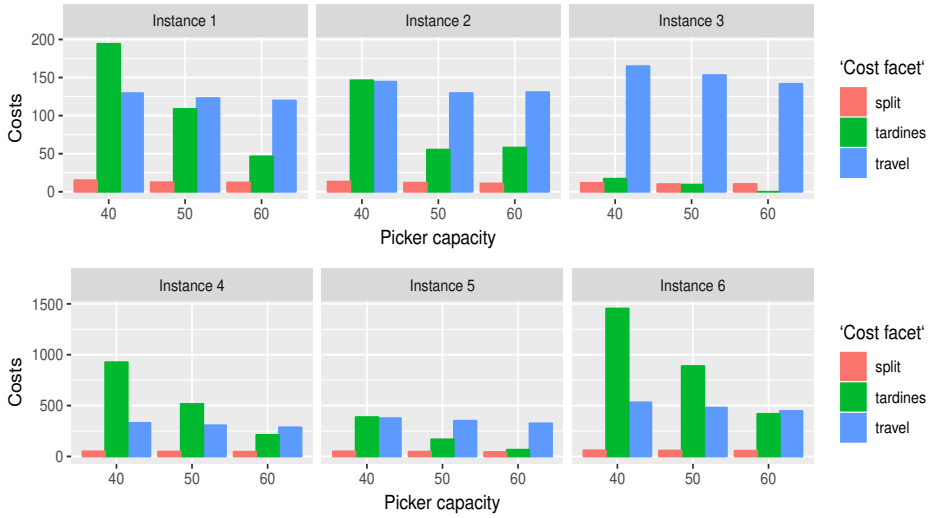


Figure 6.5: The devision of costs caused by travelling, tardines and split-up costs in for the first instances of Datasets 1-6. Customer order deadlines are cut into half and the pickers capacity is varied between 40, 50 and 60.

to 0.20 leads to a decrease in the total incurred split-up costs. A sharp increase in travel costs accompanies this.

As a final note, the fraction of split-up costs compared to the travel costs is relatively small. For small split-up costs, the split-up costs are incurred, but they do not form a significant fraction of the total costs. When split-up costs are large, solutions tend to not split-up customer orders, resulting again in a relatively small share of the total costs associated to splitting up customer orders. Concluding, splitting-up customer orders amongst multiple batches have a significant but indirect effect on the total efficiency.

6.5.9 Cost partitioning with tight deadlines

The previous analyses are performed in settings where deadline violations did not occur. This is a practically realistic scenario, as violating deadlines might disrupt operations further downstream the supply chain. However, we are interested in the performance of the ALNS and the cost composition of the proposed solutions, in case customer order deadlines are violated. We, therefore, solved the first instances of datasets 1-6 where we cut the deadlines into half and reduced the order picker capacity significantly.

In Figure 6.5, we present for a varying picker capacity, the division in travel, tardiness, and split-up costs. Several observations are made. First, increasing picker capacity leads to less customer order violations. Tardiness costs are then reduced significantly while the travel costs and split-up costs remain similar. Second, the split up costs are a fairly constant factor for each instance, and each order picker's capacity. Also, large variations in cost division are observed among the different instances. For example, Instance 3 does almost completely consist of travel costs, whereas instances 1 and 6 are dominated by tardines costs for a capacity of 40. Hence, from these experiments we can conclude that allowing for a sufficient order picker capacity can have a great impact on reducing the tardiness costs. In particular, for practical systems one shouls imply design the warehouse so that order capicty is not a bottleneck of the total system performance.

6.5.10 Impact of the MIP operators and results verification

We test the impact of using the additional MIP operators as described in Section 4.4. We performed two rounds of additional experiments which we briefly summarize in the following.

First, it should be noted that the MIP operators are especially useful to find a good composition of already discovered batches. Hence, we found no additional benefit for using the proposed MIP operators on the datasets and associated G-JOBASRP instances as initially proposed. We, therefore, solved the G-JOBASRP instances associated with datasets 1-6 with a reduced picker capacity to imitate scenarios in which tardiness is observed in the solution.

We found that, especially in the early stages of the ALNS, a faster converging algorithm is found as the MIP operators found significant improvements by rescheduling the already observed batches. Hence, when less iterations are required, or if any guarentee on the optimality of the proposed schedule is needed (given the batches as discovered by the ALNS), we suggest to include the MIP operators in the ALNS. If, on the other hand, the runtime is not a limiting factor, the solutions proposed by the ALNS with and without MIP operators have a similar performance.

The second series of experiments are as follow. We solved the associated G-JOBASRP instances of datasets 1-6 with the ALNS without using the MIP operators. While doing so, we store all discovered batches to suffice as input for the MIP formulations, and accordingly the MIP operators.

We randomly selected $n = 10000$ batches from all the stored batches. We then compared the LP relaxation of the MIP formulations subject to the 10000 randomly

selected batches to the final solution. This is repeated 100 times for all the first instances of Datasets 1-6 with a order picker's capacity of 30 and 40. We did not find any improvement to the final solutions proposed by the ALNS. This shows that the ALNS has no problems scheduling batches correctly.

Concluding, although the above presented results are no conclusive prove that the ALNS provide near-optimal solutions, it shows that the ALNS provides high-quality solution so that the dynamics of the G-JOBASRP can be analyzed correctly. Lower bounds based on Langrangian relaxation can be constructed to provide an upper bound on the optimality gap, though this is outside the scope of this paper.

6.6 Conclusions

In this paper, we studied the generalized joint order batching, assignment, sequencing and routing problem in picker-to-parts warehouses. The generalizion stems from the inclusion of two characteristics inspired on the recent advances in e-commerce logistics. First, we included the restocking of returned products in the regular customer order processing operations, and second, we investigate the cost-savings potential of picking the distinct order lines that belong to a single customer order amongst multiple order picking batches.

To formally introduce the problem we studied in this paper, we provided an compact mixed integer programming formulation, as well as two extended formulations. We developed a parallel Adaptive Large Neighborhood Search (ALNS), in which the operators are inspired on classical ALNS moves, on classical iterative local search elements, and searches through neighborhoods defined by the extended MIP formulations. We created a testbed of instances of practical size; the largest solved instances consists of 8038 order lines.

We compared the performance of the ALNS against two intuitive and practically often observed heuristics and found on-average cost-savings of 17.4%. Using the ALNS, we show that incorporating product returns in the regular order picking processes can result up to 19% cost-savings compared to handling the product returns separately. In addition, we tested the maximal cost-savings potential of splitting up customer orders by comparing the situation in which splitting-up is free versus the situation in which splitting-up is not allowed. This is shown to be around 45%.

In particular, our model and solution algorithm help to organize the order picking process as a whole; to account for the interdependencies between batching and routing, multiple pickers, and deadlines; and to tackle increasing volumes of product returns. We showed that this integrated point of view is especially benificial when warehouses

are not working on their full capacity, which is typically observed in practice (except peak-periods around, for instance, Christmas). Hence, research should continue to explore integrated warehouse operation methods. For example, the storage location assignment exerts a substantial impact on order picking performance (Theys et al. 2010). Solution procedures with shorter computation times are also desirable as unexpected, short-notice rescheduling might be necessary sometimes. That is, exploiting dynamic strategies might be of interest to pursue. Other optimization models for the order picking problem focussing on determining optimal solutions, or lower bounds to real-life sized instances, are of particular relevance as well. Namely, only if such methods are available it is possible to infer whether more time should be spent on new (meta)heuristic methods.

Chapter 7

Two-stage robust network design with temporal characteristics

Abstract. *We analyze a two-stage stochastic network design problem inspired by operations between city distribution centers. Here, commodities have an uncertain delivery window in which they can be transported between their associated origin and destination location. By taking a robust optimization point of view, our goal is to analyze networks that on a day-to-day basis allow for transportation of the commodities while minimizing the worst-case realization of fixed and variable transportation costs. We introduce the concept of time-invariant vehicle paths: In the first-stage, we determine a sequence of visited locations, while in the second-stage, after observing uncertainty, we determine the departure and arrival times of the vehicle path and which commodities to transport. We model the use of time-invariant vehicle paths by combining a flat network with a time-expanded network. Next to providing a generic two-stage robust formulation, we provide a lower bound approach by means of a large-scale scenario-based mixed integer programming formulation, and an upper bound approach by means of robust counterparts to a static, single-stage formulation. We end this paper by sketching situations for which time-invariant vehicle paths are most promising. In addition, we provide a small numerical analysis comparing the scenario-based large-scale MIP with the static, single-stage counterparts. We show that the concept of time-invariant vehicle paths is a promising way to design robust networks between city distribution centers.*

This chapter is based on Schrotenboer, Ursavas, and Vis (2019c):
Schrotenboer AH, Ursavas E, Vis IFA, 2019c *Two-stage robust network design with temporal characteristics.* Working paper

7.1 Introduction

Network design problems generally consist in finding a cost-minimizing set of *links* in a network so as to *transport commodities* between their corresponding origin and destination locations along the opened links (Gendron, Crainic, and Frangioni 1999, Crainic 2000). Due to the increased interconnectivity of supply chains and distribution networks, the accurate modeling of the commodities' temporal characteristics such as the earliest possible pickup time and latest possible delivery time has become prevalent nowadays (Boland et al. 2017). Especially in city distribution networks, temporal characteristics have an high impact on the daily operations but are uncertain in practice. If not taken into account properly, small deviations might invalidate the operations planned, leading to high costs and dissatisfied customers up- or downstream the supply chain.

One particular way to deal with uncertainty is *two-stage robust optimization* (Ben-Tal et al. 2004, Hanasusanto, Kuhn, and Wiesemann 2015, Yanıkoğlu, Gorissen, and Den Hertog 2019). Here, recourse decisions are taken to best-react on first-stage decisions after uncertainty is revealed while minimizing overal worst-case performance. We adopt this approach to develop *robust* networks to deal with the uncertainty of the temporal characteristics. That is, we search for an a-priori network design that minimizes the worst-case outcome of daily operations. Compared to stochastic programming, where the average performance is minimized, two-stage robust optimization performs on-average only slightly worse while it excludes extremes of daily observed costs (as the worst case is minimized). To the best of the authors' knowledge, this is the first work that unites the fields of two-stage robust (integer) optimization and network design with temporal characteristics.

Network design, as approached in this paper, refers to the design of the network for a single consolidation carrier that is responsible for transporting all commodities (i.e., shipments of parcels or freight) between distinct origin and destination locations. Consolidation of the commodities leads to higher on-average truckloads and less required vehicles, and is the major opportunity to reduce costs in network design problems (Dejax and Crainic 1987, Chouman, Crainic, and Gendron 2016). At the same time, distribution networks typically depend upon upstream and downstream parts of the supply chain, which are often controlled by different stakeholders (Van der Heide et al. 2018). Hence, a commodity's earliest possible pickup time at the origin and its latest possible delivery time at the destination are beyond the consolidation carrier's control, and, therefore, possess a great amount of uncertainty and risk for daily operations. Namely, slight changes in temporal characteristics can disrupt the efficient

consolidation of commodities, thereby reducing average truck loads and increasing the number of required vehicles. We, therefore, aim to design robust networks so that changes in temporal characteristics can be handled efficiently in day-to-day operations (Ardestani-Jaafari and Delage 2017, Van Engeland et al. 2018).

We adopt a two-stage robust integer optimization paradigm (see, e.g. Bertsimas and Caramanis 2007, Bertsimas and Georghiou 2015, Hanasusanto, Kuhn, and Wiesemann 2015, Postek and Hertog 2016) to cope with the uncertain temporal parameters of the commodities. In two-stage robust (integer) optimization, also called adjustable robust (integer) optimization, the decision maker is allowed to take decisions both here-and-now (before uncertainty is revealed) and, in addition, to adapt or further specify their decisions after uncertainty is revealed. The objective is to minimize the sum of the first-stage decision costs and the worst-case realization of the second-stage decision costs. The resulting optimization problem is a trilevel optimization problem (min - max - min). This reflects situations, where the decision maker is able to alter decisions based on dynamically arising information, but for which it is crucial to provide feasible and robust solutions. Two-stage robust optimization thereby improves upon the static robust optimization paradigm in which parameter uncertainty is dealt with by taking here-and-now decisions *only*. Here, uncertainty is assumed to be modeled by user-specified uncertainty sets containing the possible outcomes of the uncertain parameters (see, e.g. Gorissen, Yanıkoğlu, and Den Hertog 2015, Yanıkoğlu, Gorissen, and Den Hertog 2019).

In this paper, we introduce the Two-stage Robust Network Design Problem with Temporal Characteristics, or short the Robust Network Design Problem (RNDP), in which we need to select a number of *time-invariant vehicle paths* in the first-stage, and after observing uncertainty, need to determine the departure and arrival times at the locations of the selected vehicle paths in the second-stage. The time-invariant vehicle paths include a fixed cost for using a vehicle and arc-dependent travel costs. We further consider commodity specific arc-dependent transportation costs. The goal of the RNDP is then to find a set of vehicle paths that simultaneously minimizes the first-stage vehicle paths costs and the worst-case realization of second stage commodity transportation costs. Here, realizations of temporal characteristics and commodity weights are assumed to be drawn from a user-specified uncertainty set.

We model the first-stage decisions of the RNDP on a flat network and the second-stage decisions on a time-expanded network. By discritizing the uncertainty set (i.e., taking scenarios), we present a large scale MIP formulation for solving the RNDP. We present a sampling approach that randomly selects such scenarios to provide lower bounds on the optimal solution of the RNDP. To determine the value of taking

decisions dynamically instead of statically, we present a robust counterpart for taking all decisions before uncertainty is revealed. The latter model is an upper bound for the optimal solution of the RNDP. We used these models to assess the value of dynamic decision making for different configurations of the network. By numerical examples and computational experiments we show the potential of using dynamic decision making and using time-invariant vehicle paths.

Throughout this paper, we interpret the notions of *links*, *commodities*, and *transport* in the context of transportation networks, i.e., an opened link corresponds to a vehicle transporting commodities (i.e., freight or parcels) between two locations in the network. However, we would like to stress that they can be interpreted as more general descriptors for numerous other applications including the design of flight networks for airline operations (Büdenbender, Grünert, and Sebastian 2000), post-disaster debris collection operations (Çelik, Ergun, and Keskinocak 2015), health-care scheduling (Zarrinpoor, Fallahnezhad, and Pishvaee 2018), or maintenance operations at offshore wind farms (Schrotenboer, Ursavas, and Vis 2019b).

In the remainder of the introduction, we first detail the notion of *time-invariant vehicle paths* and afterwards review the relevant concepts in robust optimization and indicate how our contributions are positioned within this field. Next, we provide an overview of network design problems that are already addressed from a robust optimization point of view, and how our work contributes there. We finish with a summary of this paper's contributions and an outlook of the remainder of this paper.

7.1.1 Time-invariant vehicle paths

Traditionally, two-stage optimization problems in network design with uncertain parameters ask to open a set of (capacited) links in the first stage, and, after observing uncertain parameters, to route the commodities from their origin to destination locations in the second-stage along the links opened in the first stage (see, e.g. Silva, Poss, and Maculan 2018a,b). This is, however, not particularly suitable for the temporal aspects that are involved in the RNDP, as we need to decide upon the departure and arrival time of locations associated with the opened links. Deciding in the first-stage might be to conservative and does not exploit information readily availably in the era of big data and the internet of things. On the other hand, if we decide upon arrival and departure times solely in the second-stage (and thereby which links to open) it requires practically yet unattainable high levels of coordination and cooperation between the operated vehicles on the links.

We introduce the notion of *time-invariant vehicle paths* in which the nodes to

be visited are determined a priori. After uncertainty is revealed, we determine the departure and arrival times associated with the a-priori selected time-invariant vehicle paths to route the commodities between origin and destiantion location. Using this concept, we decide upon the number of operated vehicles and their associated paths in a static way, but allow for the operational flexibility to determine departure and arrival times after uncertainty is revealed.

Besides its practical value, using time-invariant vehicle paths allows us to study the trade-off between the number of vehicles available in the system and its corresponding operational flexibility. Namely, we include a fixed cost for selecting a time-invariant vehicle path. If only single links are opened in the first stage (with associated times), it is difficult to determine the minimum number of required vehicles. This is a difficult optimization problem by itself and leads to a MIP formulation (of the complete problem) that is hard to solve.

7.1.2 Two-stage robust optmization with integer second stage.

We will model the RNDP on a so-called time-expanded network. In general terms, it entails copying the nodes (and links) for a set of discretized time steps, so that a node decodes a combination of time and location. The details on this can be found in Section 7.2, but it implies that the second-stage decisions are to select links associated with the time-invariant vehicle path in the time-expanded network.

Hence, in the RNDP, we both take integer first-stage decisions (i.e., which time-invariant vehicle path to choose) and second-stage integer decisions (i.e, choose associated links in the time-expanded network). We, therefore, briefly summarize the literature on two-stage integer robust optimization.

Two-stage robust optimization, or adaptive robust optimization Yanıkoğlu, Gorissen, and Den Hertog (2019), entails tri-level optimization problems where first-stage decisions are taken so that the worst-case second stage costs, where we are allowed to take decisions after uncertainty is revealed, is minimized. Two-stage robust optimization does not allow for directly determining the robust counterpart, which is the straightforward concept in static robust optimization (Gorissen, Yanıkoğlu, and Den Hertog 2015). Therefore, tailored methods have been developed for solving two-stage robust optimization problems, which we shortly review in the following.

Linear decision rules, originating from non-integer adaptive robust optimization, model the second stage decision as an affine function of the realization of uncertain parameters (Ben-Tal et al. 2004). This has recently been translated to also be valid for integer second-stage decisions by (Bertsimas and Georghiou 2015). Another way

to deal with two-stage robust integer models is finite adaptability (Bertsimas and Caramanis 2010). In such an approach, one restricts the number of second-stage decisions that are allowed to take. Geometric characteristics, under which conditions finite adaptability provides good approximations, are discussed in Bertsimas, Iancu, and Parrilo (2010). More recently, Hanasusanto, Kuhn, and Wiesemann (2015) discuss finite adaptability for two-stage robust binary optimization. We choose to not use K-adaptability for solving the RNDP, as it opposes the concept of time-invariant paths where we can still select arrivel and departure times completely dynamic in the second stage. Lastly, Bertsimas and Dunning (2016) and Postek and Hertog (2016) discuss the concept of iteratively splitting the uncertainty set. The general idea is to split the uncertainty in subsets with its own decisions.

We choose to follow an algorithmic approach recently proposed by Zeng and Zhao (2013). It is a column-and-row generation method that iteratively generates scenarios that are either infeasible for the current first-stage solution or define a a new worst-case scenario. As we aim to infer how much the value is of dynamically adjusting vehicle paths after observing uncertainty, i.e., using two-stage robust optimization instead of static robust optimzation, we deemed this approach suitable. Namely, it gives us lower bounds at each moment in time, allowing us to infer upon the value of dynamically adjusting paths at any point during the execution of the algorithm.

7.1.3 Network design problems

As network design problems are well established problems within the field of Operations Research, many applications have been studied, and also robust approaches have found there way (see, e.g., Van Engeland et al. 2018). First, we discuss deterministic works that are related to the RNDP, and afterward, we discuss to which extent uncertainty has been studied in general network design problems.

At the basis of RNDP lies the multi-commodity fixed-charge network design problem (see, e.g., Chouman, Crainic, and Gendron 2016). Here, the goal is to open a set of arcs and route the commodities from their origin to destination location along the opened arcs. In these problems, a fixed charge is payed when an arc is opened and variable commodity transportation costs are incurred if the arc is used. This also forms the core of the decisions present in the RNDP, however, two distinctions are observed. First, the multi-commodity fixed-charge network design problem does not consider temporal characteristics as we do, and second, there is no limit (and associated costs) to the number of vehicles that are required to perform the operations on the opened links.

A few papers consider the management of additional resources in network design problems. In Andersen et al. (2011) and Crainic et al. (2014), the authors consider the management of (among others) a fleet of vehicles, and ensure that the resulting network design is periodic. In their context, a period is typically a day, and a vehicle can only travel along a single arc in a single period. In the RNDP, we model operations inspired on high-pace city-logistics, where the complete time horizon comprises several hours instead of several days. Hence, we consider vehicle paths (i.e., a sequenced of opened arcs associated with a single vehicle) within a single period.

The RNDP has been studied in a determenistic setting by Boland et al. (2017), where the authors propose a so-called dynamic discretization discovery algorithm. This algortihm iteratively enhances the discretization instead of considering a complete time-expanded network. As we focus on the impact of dynamically adjusting decisions after uncertainty is revealed, we take the discretization as given which does not require us to explore opportunities of applying dynamic discovery algorithms in the robust setting.

From a robust optimization point of view, surprisingly few studies focused on 'flow' problems on graphs. In Atamtürk and Zhang (2007), two-stage robust network flows such as the maximum flow problem are studied, and found to be NP hard in general. They present specific network structures that allow for solving flow problems in polynomial time. After that, several works on robust 'flow' problems have emerged. Namely, robust optimization approaches to the single-commodity robust network design problem have been considered by Cacchiani et al. (2016), two-stage robust approaches for minimum cost flows for general graphs have been discussed by Simchi-Levi, Wang, and Wei (2018), and two-stage robust maximum flows problems for networks with failing arcs are discussed in Bertsimas, Nasrabadi, and Stiller (2013). Although these works provide fundamental insights in (two-)stage robust optimization for network design problems, they do not focus on practical and rich network design problems such as the RNDP. Finally, Silva, Poss, and Maculan (2018a) and Silva, Poss, and Maculan (2018b) study K-adaptability variants of the fixed-charge capacitated network design problem, for a set of given vehicle paths. However, the authors do not consider temporal characteristics as we do.

There are a few miscellaneous studies that relate to the RNDP, but do not belong to any particular category. The multi-commodity fixed-charge capacitated network design has been considered from a stochastic programming point of view recently (Rahmaniani et al. 2017). The authors propose enhanced benders-decomposition algorithms to solve this problem. Also heuristics have been considered within this context, see Sarayloo, Crainic, and Rei (2018). In Ardestani-Jaafari and Delage (2017),

the authors study multi-period location-transportation problems. In particular, they quantify the value of flexibility by comparing the static location-transportaion problem with a variant where demand can be fulfilled after uncertainty has been realized. Although this analysis has a similar aim as our study, i.e., it compares static with dynamic decision making, we introduce the new concept of time-invariant vehicle paths and investigate its impact on flexibility on network design problems with temporal characteristics.

7.1.4 Contributions and outlook

We aim to contribute by analyzing two-stage robust networks and possible ways to obtain lower and upper bounds to their associated optimal solutions. Our contributions can be summarized as follows:

- We study two-stage robust solutions of a practically relevant problem arising in transportation and logistics, namely, a service network design problem arising in city-logistics operations.

- We introduce the concept of *time-invariant vehicle paths* in which a sequence of location visits is made a-priori, and the associated departure and arrival times are determined after uncertainty is revealed.

- We provide a path-based formulation and reformulate it as a large-scale, scenario based, mixed-integer programming model. In addition, we present the robust counterpart to the static variant of the RNDP.

- We analyze the value of using time-invariant vehicle paths by comparing the efficiency gain of allowing two-stage decision making over single-stage decision making.

The remainder of this paper is as follows. In Section 7.2, we introduce the RNDP formally and provide a generic two-stage robust formulation. A lower bound approach, by means of a large-scale scenario-based MIP, is presented in Section 7.3. In Section 7.4, we analyze the single-stage variant of the RNDP and provide its tractable robust counterpart. We provide experimental insights in Section 7.5, and we conclude the paper in Section 7.6.

7.2 Problem formulation

In the following, we present a general formulation for the two-stage Robust Continuous Time Network Design Problem (RNDP). We start with a general description of the RNDP on a flat network, i.e., where time is not explicitly encoded within the network. Thereafter, we provide the necessary notation to model the RNDP on a time-expanded network (see, e.g. Boland et al. 2017). We end this section with a general formulation that exploits elements from both the flat network and the time-expanded network.

We model the uncertain parameters by a polyhedral uncertainty set, i.e., $\Xi := \{\xi \mid E\xi \leq f\}$, where E and f are a matrix and vector of compatible size, respectively. We denote dependency of uncertain parameters on this set by $\cdot(\xi)$ with $\xi \in \Xi$.

7.2.1 Problem statement

The RNDP is situated on directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \{1, \ldots, N\}$ denotes the set of nodes and $\mathcal{A} = \{1, \ldots, A\}$ denotes the set of arcs. We consider a set of commodities $\mathcal{K} = \{1, \ldots, K\}$, each of which have an associated quantity $q^k(\xi) \in \mathbb{N}_{\geq 0}$, an origin $o^k \in \mathcal{N}$, a destination $d^k \in \mathcal{N}$, an earliest possible pickup time $e^k(\xi) \in \mathbb{N}_{\geq 0}$ at the origin node, and a latest possible delivery time $\ell^k(\xi) \in \mathbb{N}_{\geq 0}$ at the destination node. We call $\delta^k(\xi) := (\ell^k(\xi) - e^k(\xi)) \geq 0$ the delivery time of commodity k. All events take place on a continuous time horizon $[0, T]$.

Let $\mathcal{V} = \{1, \ldots, V\}$ be the set of vehicles that are available for transporting commodities along the graph G. Using a vehicle comes at a fixed vehicle cost $f \in \mathbb{N}_{>0}$ and at transportation cost $c_{ij} \in \mathbb{N}_{>0}$ for traversing arc $(i, j) \in \mathcal{A}$. Moreover, variable commodity costs $\tilde{c}^k_{ij} \in \mathbb{N}_{>0}$ are incurred if commodity k is transported by some vehicle long arc $(i, j) \in \mathcal{A}$.

Let $\mathcal{P}$ be the exponentially large set of time-invariant vehicle paths (in short, vehicle paths) through graph G. Each vehicle path $p \in \mathcal{P}$ is defined by a sequence of visited arcs $(a^p_1, a^p_2, \ldots, a^p_{n^p}), a_j \in \mathcal{A}, j \in \{0, \ldots, n^p\}$. Let c^p denote the costs associated to path $p \in \mathcal{P}$, i.e., the fixed costs f and the transportation costs c_{ij} for all $(i, j) \in \mathcal{A}$ that are contained in path p.

We take decisions in two-stages. In the first stage, we select a set of vehicle paths along arcs $(i, j) \in \mathcal{A}$. Let x^p be binary decision variables that equal 1 if $p \in \mathcal{P}$ is selected in the first-stage, and 0 otherwise. We do not specify any temporal aspects of the selected paths, as we do not specify which commodities are transported along the set of paths. In the second stage, after uncertainty is revealed, we need to specify 1) the actual departure (or arrival) times at the nodes for each vehicle path selected in the first stage, and 2) we need to transport the commodities from their associated

origin to their associated destination locations along the 'timed' copies of the arcs.

Whereas the first-stage decisions are situated upon the flat network G, the second-stage decisions take place in a time-expanded network where time is explicitly modeled in the nodes of the network. Before we detail the construction of the time-expanded network and the corresponding second-stage decisions, we make several remarks explicit for future reference.

Remark 7.1. Although we do not specify any temporal aspect of the paths in the first stage, some necessary conditions can be imposed. For instance, the chosen paths should allow for transportation of all the comodities if we set the delivery times (and the time horizon length) arbitrarily large. Valid inequalities could be developed on the basis of such observations to enhance the computational efficiency of any method.

Remark 7.2. For degenerate Ξ and $f = 0$, the RNDP reduces to a single-stage deterministic continuous time network design problem as introduced by Boland et al. (2017). Namely, we can interpret a vehicle route as opening a single arc (i, j) since $f = 0$, and use the arc-based formulation of Boland et al. (2017) and their proposed solution approach. However, in order to give a meaningful and practical interpretation of robustness in continuous time service network design problems, we will consider vehicle routes with $f > 0$.

Remark 7.3. For degenerate Ξ, one can design a compact MIP formulation of the RNDP based on the flat network described above by tracking the order in which vehicles (i.e., open arcs) arrive at the nodes in the network and by introducing sufficient variables tracking the time of commodities throughout the graph. Nevertheless, this formulation includes many linearized constraints resulting in weak LP relaxations, which reduces its practical usability.

7.2.2 Time-expanded network and second stage decisions

In line with Boland et al. (2017), we consider a partial time-expanded network formulation in which for each node $i \in \mathcal{N}$ a set of times $\mathcal{T}(i)$ is defined so that for each $\tau^i \in \mathcal{T}(i)$ it holds that $0 \leq \tau^i \leq T$. We define $\mathcal{T} = \cup_{i \in \mathcal{N}} \mathcal{T}(i)$ as the collection of all time-node combinations.

Let $\mathcal{G}^{\mathcal{T}} = (N^{\mathcal{T}}, \mathcal{A}^T)$ be time-expanded network that results from evaluating graph G subject to $\mathcal{T}$. Each node $(i, \tau^i) \in \mathcal{G}^{\mathcal{T}}$ describes the original node $i \in \mathcal{N}$ and an explicit moment in time $\tau^i \in \mathcal{T}(i)$. Two different arc subsets constitute the arc set $\mathcal{A}^{\mathcal{T}}$. First, traveling arcs $((i, \tau^i), (j, \tau^j)) \in \mathcal{A}^{\mathcal{T}}$, with $i \neq j$, denote traveling along arc $(i, j) \in \mathcal{A}$ departing from i at τ^i and arriving at j before or at τ^j. Second, there are holdover arcs $((i, \tau^i), (i, (\tau + 1)^i)$ that model waiting at node $i \in \mathcal{N}$.

To determine the temporal characteristics of a vehicle path in the second-stage decision, we consider binary variables $y^p_{(i,\tau^i),(j,\tau^j)}$ equalling 1 if path $p \in \mathcal{P}$ visits in the time-expanded network arc $((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}$, and 0 otherwise. We only consider $y^p_{(i,\tau^i),(j,\tau^j)}$ variables that include timed arcs corresponding to path p. Moreover, for all $p \in \mathcal{P}$ it holds that

$$y^p_{(i,\tau^i),(j,\tau^j)} = 0 \text{ for all } ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}} \iff \bar{x}_p = 0. \tag{7.1}$$

This can straightforwardly be modeled by introducing parameters $\zeta^p_{(i,\tau^i),(j,\tau^j)}$ equalling 0 if arc $(i,\tau^i),(j,\tau^j) \in \mathcal{A}^{\mathcal{T}}$ cannot be traversed with path p. This allows defining $\mathbf{y}$ for all arcs in $\mathcal{G}^{\mathcal{T}}$ with $\mathbf{y} \leq \boldsymbol{\zeta}$. However, for readibility reasons, we ensure that by construction we only generate $\mathbf{y}$ variables that can take a positive value, and no constraints of the form $\mathbf{y} \leq \boldsymbol{\zeta}$ will be included in the mathematical formulations that will folow.

We visualize the relation between first and second stage decisions by the following example.

Example 7.1. Consider a graph with 4 nodes n_1, n_2, n_3, and n_4. Consider the first-stage vehicle path $p = (n_1, n_2, n_3, n_4)$ that visits all nodes. Let the travel times be equal to $t_{n_0,n_1} = 10, t_{n_1,n_2} = 5$, and $t_{n_2,n_3} = 10$. Suppose we have a sets of discretized time stemps $\mathcal{T}(n_i) = \{0, 5, 10, 15, 20, 25, 30, 35\}$ for all $i \in \{1, 2, 3, 4\}$. The corresponding time expanded network is given in Figure 7.1. In the time expanded network, we depicted all the arcs that are allowed to be choosen if path p is selected in the first-stage. Namely, that are all the red arcs corresponding to departing as early as possible from each node, all the blue arcs that correspond to the latest departure possible at each node, and all dashed arcs which fall between departing as early and as late as possible. It is clear that longer paths have less flexibility than shorter paths, as there is not much flexibility to allow for waiting at nodes. ◁

From the example, it follows that determining the second stage solution is equivalent to finding "cheapest" (s,t)-paths, beginning at node in the time-expanded network corresponding to the first visited location of the path at the earliest possible time stamp, ending at the node at the latest visited location of the path at the latest possible time stamp, only using the induced arcs of the vehicle path on the time-expanded network. Here "cheapest" means determining (s,t)-paths so that the costs of the worst-case second-stage solution are minimized.

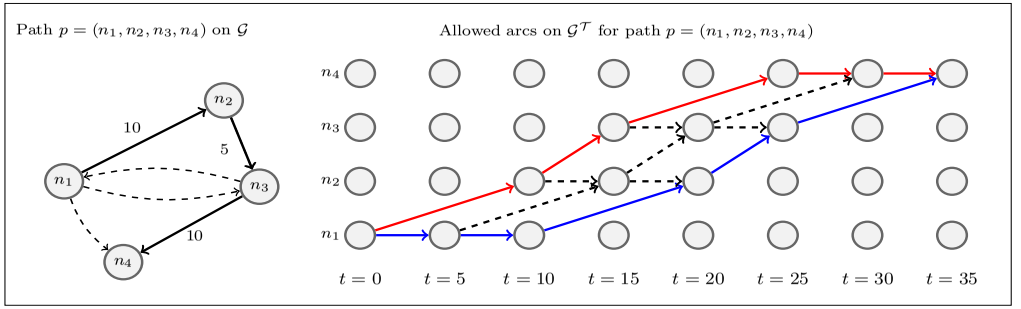


Figure 7.1: Example of allowed second-stage arcs for time-invariant vehicle path $p = (n_0, n_1, n_2, n_3)$.

7.2.2.1 Second-stage optimization model.

Next to the the first-stage variables $\mathbf{x}$ and the associated second-stage variables $\mathbf{y}$, we introduce the binary second-stage variables $z^k_{(i,\tau^i),(j,\tau^j)}(\xi)$ that equal 1 if commodity $k \in \mathcal{K}$ travels along arc $((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}$, and equal 0 otherwise. The relation between the variables $\mathbf{z}$ and the quantities $\mathbf{q}(\xi)$ is rather straightforward but nonlinear. The relation of the uncertain temporal characteristics (release times $\mathbf{e}(\xi)$ and deadlines $\boldsymbol{\ell}(\xi)$) with the $\mathbf{z}$ (and the $\mathbf{y}$) variables is less straighforward and certainly nonlinear. We can, however, project the uncertain release times and deadlines on $\mathcal{A}^{\mathcal{T}}$ by means of the binary parameter $\theta^k_{(i,\tau^i),(j,\tau^j)}(\xi)$ that equals 1 if transporting commodity k along arc $((i,\tau^i),(j,\tau^j))$ is feasible for realization ξ, and is 0 otherwise. Note that $\boldsymbol{\theta}$ typically exhibits a nonlinear relation to a convex uncertainty set (e.g., deadlines are from a convex set) or a linear relation to a nonconvex uncertainty set (e.g., deadlines are from a discrete nonconvex set). We make this relation explicit in Section 7.4.

We formulate the second-stage optimization model as a network design problem in which we need to conserve both the flow of the commodities as modeled by the $\mathbf{z}$ variables and the flow of the selected vehicle paths in the first stage by means of the $\mathbf{y}$ variables. To do so, we need to model the appropriate nodes in the time-expanded network that act as source and sink for the flows corresponding to the commodities and the vehicle paths. Let $\phi_{(i,\tau^i)})^k(\xi)$ be a parameter equalling -1 if $o^k = i$ and τ^i is the first feasible time stamp of node (i,τ^i) in $\mathcal{G}^{\mathcal{T}}$ (depending on ξ), equalling 1 if $d^k = i$ and τ^i is the latest feasible time stamp, and 0 otherwise. Let $\chi^p_{(i,\tau^i)}$ be a parameter equalling -1 if i is the first node of path p and τ^i is the first time stamp of node i in $\mathcal{G}^{\mathcal{T}}$, equalling 1 if i is last node of path p and τ^i is the latest time stamp of node i, and 0 otherwise.

Let $\delta^+((i,\tau^i))$ denote all the nodes connected to (i,τ^i) via traveling arcs leaving (i,τ^i). Let $\delta^-((i,\tau^i))$ denote all the nodes connected to (i,τ^i) via traveling arcs entering (i,τ^i). Then, for a given first-stage solution $\hat{\mathbf{x}}$, the second-stage optimization problem can be written as follows:

$$\Psi(\hat{\mathbf{x}},\mathbf{y}(\xi),\mathbf{z}(\xi) \mid \mathcal{T}) := \max_{\xi\in\Xi} \min \sum_{((i,\tau^i),(j,\tau^j))\in\mathcal{A}^{\mathcal{T}}} \sum_{k\in\mathcal{K}} z^k_{(i,\tau^i),(j,\tau^j)} \tilde{c}^k_{ij} \tag{7.2}$$

$$\text{s.t.} \sum_{(j,\tau^j)\in\delta^-((i,\tau))} z^k_{(j,\tau^j),(i,\tau^i)} - \sum_{(j,\tau^j)\in\delta^+((i,\tau))} z^k_{(i,\tau^i),(j,\tau^j)} = \phi^k_{(i,\tau^i)}(\xi)$$

$$\forall (i,\tau^i) \in \mathcal{N}^{\mathcal{T}}, k \in \mathcal{K} \tag{7.3}$$

$$\sum_{(j,\tau^j)\in\delta^-((i,\tau))} y^p_{(j,\tau^j),(i,\tau^i)} - \sum_{(j,\tau^j)\in\delta^+((i,\tau))} y^p_{(i,\tau^i),(j,\tau^j)} = \chi^p_{(i,\tau^i)} \hat{x}^p$$

$$\forall (i,\tau^i) \in \mathcal{N}^{\mathcal{T}}, p \in \mathcal{P} \quad (7.4)$$

$$\sum_{k\in\mathcal{K}} z^k_{(i,\tau^i),(j,\tau^j)} q^k(\xi) \leq Q \sum_{p\in\mathcal{P}} y^p_{(i,\tau^i),(j,\tau^j)} \qquad \forall ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}} \quad (7.5)$$

$$z^k_{((i,\tau^i),(j,\tau^j))} \leq \theta^k_{(i,\tau^i),(j,\tau^j)}(\xi) \qquad \forall ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}, k \in \mathcal{K} \quad (7.6)$$

$$z^k_{(i,\tau^i),(j,\tau^j)} \in \{0,1\} \qquad \forall ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}, k \in \mathcal{K} \quad (7.7)$$

$$y^p_{(i,\tau^i),(j,\tau^j)} \in \{0,1\} \qquad \forall ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}, p \in \mathcal{P} \quad (7.8)$$

Here, Objective 7.2 minimizes the transportation costs of the commodities. Constraints (7.3) and (7.4) model flow conservation of the commodity variables $\mathbf{z}$ and the arc selection variables $\mathbf{y}$. Via Constraints (7.5) we ensure that the capacity of the selected vehicles is respected. Constraints (7.6) strengthen the bounds on $\mathbf{z}$ with respect to uncertain parameters. Finally, Constraints (7.7) and (7.8) impose the integrality conditions.

Remark 7.4. Note that we can impose so-called strong-inequalities (see, e.g. Trick 2005) on the above formulation. These are of the form $z^k_{(i,\tau^i),(j,\tau^j)} \leq y^p_{(i,\tau^i),(j,\tau^j)}$ for all $k \in \mathcal{K}$ and $p \in \mathcal{P}$. However, directly including such inequalities would result in a too large number of constraints. As modern MIP solvers can generate these inequalities in a dynamic fashion automatically, we will not implicitly mention such strong-inequalities in the remainder of this paper.

7.2.3 Two-stage Robust formulation

The complete two-stage robust integer programming formulation of the RNDP is given by

$$\min \quad \sum_{p\in\mathcal{P}} c^p x^p + \Psi(\mathbf{x},\mathbf{y},\mathbf{z} \mid \xi) \quad (7.9)$$

$$\text{s.t} \quad \sum_{p\in\mathcal{P}} x^p \leq R \quad (7.10)$$

$$x^p \in \{0,1\} \qquad \forall p \in \mathcal{P} \quad (7.11)$$

Here, Objective (7.9) minimizes the costs of the selected vehicle paths in the first-stage plus the costs corresponding to the worst-case realization $\Psi(\mathbf{x},\mathbf{y},\mathbf{z} \mid \xi)$ in the second-stage. Constraint (7.10) ensures that at most R paths are selected. Note that nonanticipativity constraints are already included via flow conservation of the $\mathbf{y}$ variables in the second-stage problem. Note that these nonanticipativity constraints

can be strengthen by strong inequalities as well, see Remark 7.4.

The set P is exponentially large, and hence, we will consider restricted path sets $\overline{\mathcal{P}} \subseteq \mathcal{P}$. Column generation is then the natural way to iteratively generate paths of reduced costs until optimality of the linear relaxation to the MIP is reached. However, the resulting pricing problem is far from trivial due to its dependency on the uncertain parameters and its associated second-stage decisions.

We choose to not pursue column generation based solution approaches for two main reasons. First, it will distract from our aim to introduce dynamic network design problems and the associated gain in flexibility over static network design problems. Second, from a practical point of view, city-distribution comprises typically of time horizons of at most a few hours. Hence, the path lengths will be relatively restricted and might be enumerated smartly. For the remainder of this paper, we therefore assume the set of vehicle paths to be given. We detail the construction in the experimental results (see Section 7.5).

7.3 A lower bound approach

In order to solve the two-stage robust integer formulation, we consider a discretized uncertainty set Ξ. That is, every $\xi \in \Xi$ can be considered a scenario (i.e. a particular uncertain realization). We will work with a subset of uncertain realizations $\bar{\Xi} \subseteq \Xi$, and we explicitly consider second stage solutions $(\mathbf{x}^{\bar{\xi}}, \mathbf{y}^{\bar{\xi}})$ for all $\bar{\xi} \in \bar{\Xi}$. Then, we can rewrite the RND by using an epigraph formulation, resulting in the following large-scale MIP model:

$$\min \quad \sum_{p \in \mathcal{P}} c^p x^p + \eta \tag{7.12}$$

$$\text{s.t.} \quad \eta \geq \sum_{((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}} \sum_{k \in \mathcal{K}} z^{k\bar{\xi}}_{(i,\tau^i),(j,\tau^j)} \tilde{c}^k_{ij} \qquad \forall \bar{\xi} \in \bar{\Xi} \tag{7.13}$$

$$\sum_{(j,\tau^j) \in \delta^-((i,\tau))} z^{k\bar{\xi}}_{(j,\tau^j),(i,\tau^i)} - \sum_{(j,\tau^j) \in \delta^+((i,\tau))} z^{k\bar{\xi}}_{(i,\tau^i),(j,\tau^j)} = \phi^{k\bar{\xi}}_{(i,\tau^i)}$$
$$\forall \bar{\xi} \in \bar{\Xi}, (i,\tau^i) \in \mathcal{N}^{\mathcal{T}}, k \in \mathcal{K} \tag{7.14}$$

$$\sum_{(j,\tau^j) \in \delta^-((i,\tau))} y^{p\bar{\xi}}_{(j,\tau^j),(i,\tau^i)} - \sum_{(j,\tau^j) \in \delta^+((i,\tau))} y^{p\bar{\xi}}_{(i,\tau^i),(j,\tau^j)} = \chi^p_{(i,\tau^i)} x^p$$
$$\forall \bar{\xi} \in \bar{\Xi}, (i,\tau^i) \in \mathcal{N}^{\mathcal{T}}, p \in \mathcal{P} \tag{7.15}$$

$$\sum_{k\in\mathcal{K}} z^{k\bar{\xi}}_{(i,\tau^i),(j,\tau^j)} q^{k\bar{\xi}} \leq Q \sum_{p\in\mathcal{P}} y^{p}_{(i,\tau^i),(j,\tau^j)} \qquad \forall \bar{\xi} \in \bar{\Xi}, ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}} \tag{7.16}$$

$$z^{k\bar{\xi}}_{((i,\tau^i),(j,\tau^j))} \leq \theta^{k\bar{\xi}}_{(i,\tau^i),(j,\tau^j)} \qquad \forall \bar{\xi} \in \bar{\Xi}, ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}, k \in \mathcal{K} \tag{7.17}$$

$$z^{k\bar{\xi}}_{(i,\tau^i),(j,\tau^j)} \in \{0,1\} \qquad \forall \bar{\xi} \in \bar{\Xi}, ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}, k \in \mathcal{K} \tag{7.18}$$

$$y^{p\bar{\xi}}_{(i,\tau^i),(j,\tau^j)} \in \{0,1\} \qquad \forall \bar{\xi} \in \bar{\Xi}, ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}, p \in \mathcal{P} \tag{7.19}$$

$$x^p \in \{0,1\} \qquad \forall p \in \mathcal{P} \tag{7.20}$$

$$\eta \geq 0 \tag{7.21}$$

In the above formulation, η is a continuous variable that is greater or equal to any second stage objective, as depicted by Constraints (7.13), and therefore equals the worst-case second stage solution. Hence Objective 7.12 minimizes the costs of the chosen vehicle paths and the worst-case costs of the second-stage decisions. Constraints (7.14) - (7.19) are the second-stage constraints for each scenario $\bar{\xi}$, and impose the same conditions as Constraints (7.3) - (7.8).

For fixed route sets $\mathcal{P}$ the above formulation can be solved by means of column and row generation, as is introduced by Zeng and Zhao (2013). It iteratively generates new scenarios which either invalidate the current worst-case objective or that result into infeasibility of the current solutions. In case of the former, we should include this scenario and resolve the complete model in order to check if worst-case costs can be reduced by taking different (first-stage) decisions. In case of infeasibility, it is clear that we need to take different decisions as a robust solution should be feasible for all scenarios.

Let $\bar{\Xi} := (\xi^1, \xi^2, \ldots, \xi^n)$ be the current set of scenarios. Let $z_{\text{DISC}}(\bar{\Xi})$ be the objective of solving the discretized formulation subject to the scenarios in $\bar{\Xi}$. Let $z_{\text{2-STAGE}}(\boldsymbol{x})$ be the solution to the second stage problem for a fixed first-sage decision $\boldsymbol{x}$. The column-and-row generation algorithm by Zeng and Zhao (2013) then consists of the following steps:

1. Solve the discretized formulation to obtain $z_{\text{DISC}}(\bar{\Xi})$. This is a lower bound on optimal solution. Let $\boldsymbol{x}^*$ be the corresponding first-stage solution and let $z_{\boldsymbol{x}^*}$ be the costs associated with the first-stage only.

2. Solve the second-stage subproblem, defined by equations (7.2) - (7.7) for the given first stage solution $\boldsymbol{x}^*$. Then $z_{\text{2-STAGE}}(\boldsymbol{x}^*) + z_{\boldsymbol{x}^*}$ is an upper bound for the optimal solution.

3. If $z_{2\text{-STAGE}}(\boldsymbol{x}^*) + z_{\boldsymbol{x}^*} > z_{\text{DISC}}(\bar{\Xi})$, include the scenario associated with $z_{2\text{-STAGE}}(\boldsymbol{x}^*)$ and go back to step 1. Otherwise, the upper bound equals the lower bound and the problem has been solved to optimality.

Major difficulty in this algorithm is how to solve the second-stage optimization problem, i.e., how to generate a new scenario. To the best of the authors' knowledge, no general methods exists yet how to generate such scenarios efficiently. Moreover, for the purpose of this paper, we deemed it feasible to sample "extreme" scenarios and to solve the discretized formulation directly. However, we urge future researcher to develop efficient scenario generation methods.

7.4 The single-stage RNDP as upper bound

In this section, we present an upper bound for the RNDP. It results from interchanging the maximization over the uncertain parameters and the minimization over the second-stage variables. In words, we take both first- and second-stage decisions before uncertainty is known. Indeed, this reflects a single-stage robust optimization approach, also called static robust optimzation.

In the following, we first give an explicit description of a polyhedral uncertainty set that will be used in the remainder of this paper. Then, we provide the tractable robust counterpart (which is a MIP) that solves the static variant of the RNDP.

7.4.1 The uncertainty set

Although the derivation in the following is similar for any affine $\Xi(\xi)$, we will illustrate it using the uncertainty set detailed below. It equals the following polytope.

$$\ell^k - e^k = \Delta_1^k \qquad \forall k \in \mathcal{K} \tag{7.22}$$

$$e^k \leq e^k_{\max} \qquad \forall k \in \mathcal{K} \tag{7.23}$$

$$e^k \geq e^k_{\min} \qquad \forall k \in \mathcal{K} \tag{7.24}$$

$$\sum_{k\in\mathcal{K}} e^k \leq \Delta_3 + \sum_{k\in\mathcal{K}} e^k_{\min} \tag{7.25}$$

$$q^k \geq q^{\min} \qquad \forall k \in \mathcal{K} \tag{7.26}$$

$$q^k \leq q^{\max} \qquad \forall k \in \mathcal{K} \tag{7.27}$$

$$\sum_{k\in\mathcal{K}} q^k \leq \Delta_2 + \sum_{k\in\mathcal{K}} q^k_{\min} \tag{7.28}$$

In Equation (7.22), we ensure that there is a fixed time window in which a commodity can be transported. The motivation stems from analyzing the network from a total throughput time point of view. Indeed, increasing Δ_1^k provides more time to deliver the commodities. Equations (7.23) and (7.24) impose bounds on the commodities' deadlines and the release times, respectively. With Equation (7.25), we control the degree of uncertainty that is present in the system. Namely, we restrict the sum of deviations from a (given) minimum release time. With Equations (7.26) and (7.27) we impose minimum and maximum quantities on the commodities' weights, respectively. Finally, with Equation (7.28), we restricted the sum of deviations from the (given) minimum quantities.

Modeling the uncertainty set in this way, we are able to relate the uncertain parameters $(\mathbf{e}, \boldsymbol{\ell}, \mathbf{q})$ to the second stage decision $(\mathbf{y}, \mathbf{z})$ by the following set of constraints:

$$\sum_{(j,\tau^j)\in\delta^+(o^k,\tau^{o^k})} z^k_{(o^k,\tau^{o^k}),(j,\tau^j)} e^k \leq \tau_{o^k} \qquad \forall (o^k, \tau^{o^k}) \in \mathcal{N}^{\mathcal{T}}, k \in \mathcal{K}, \quad (7.29)$$

$$(e^k + \Delta_1^k) \geq \sum_{(j,\tau^j)\in\delta^-(d^k,\tau^{d^k})} z^k_{(j,\tau^j),(d^k,\tau^{d^k})} \tau_{d^k} \quad \forall (d^k, \tau^{d^k}) \in \mathcal{N}^{\mathcal{T}}, k \in \mathcal{K}, \quad (7.30)$$

$$\sum_{k\in\mathcal{K}} z^k_{i,\tau^i} q^k \leq Q \sum_{p\in\mathcal{P}} y^p_{i,\tau^i} \qquad \forall (i, \tau^i) \in \mathcal{A}^{\mathcal{T}}. \quad (7.31)$$

With Inequalities (7.29) we ensure that we can only use arcs leaving the origin of a commodity $k \in \mathcal{K}$ if its corresponding time τ_{o^k} is larger than e^k. Similarly, Inequalities (7.30) models that only arcs can enter the destination of commodity k if τ_{d^k} is smaller than $\ell^k := e^k + \Delta_1^k$. Lastly, we denote the relation between the commodities weight and the arc usage via Inequalities (7.31). Note that these inequalities also model the relation between the $\boldsymbol{z}$ and $\boldsymbol{y}$ variables.

7.4.2 A MIP-based upper bound

We construct a compact MIP (i.e., with polynomial number of variables and constraints) that gives an upper bound on the optimal solution to (7.9) -(7.11). It is obtained by interchanging the inner minimization $\boldsymbol{y} \in \mathcal{Y}$ with the maximization $\boldsymbol{\xi} \in \Xi$. That is, we consider a static, single-stage robust optimization variant of the RNDP, where we select vehicle paths with associated loadings and departure times here-and-now so that the worst-case costs are minimized.

After interchanging the minimization and maximization, we can deploy standard robust optimization techniques to lose the inner maximization over the uncertainty set.

First, regarding the uncertain parameter $e^k, k \in \mathcal{K}$, we can replace $\phi^k_{(i,\tau^i)}$ by its robust counterpart $\bar{\phi}^k_{(i,\tau^i)}$ that is defined as

$$\bar{\phi}^k_{(i,\tau^i)} = \begin{cases} -1 & \tau^i > e^k_{\max} \text{ and } \tau^i - e^k_{\max} < \delta, \\ 1 & \tau^i > e^k_{\min} + \Delta^k_3 \text{ and } \tau^i - (e^k_{\min} + \Delta^k_3) < \delta, \\ 0 & \text{otherwise.} \end{cases} \tag{7.32}$$

Here, δ is the difference in time between the two consecutive time epochs in the time-expanded network, assuming all time epochs are of the same length. In other words, we set the delivery window for each commodity k as $[e^k_{\max}, e^k_{\min} + \Delta^k_3]$, assuming this is well defined.

Remark 7.5. When $\Delta^k_3 < e^k_{\max} - e^k_{\min}$, the above transformation of ϕ is remains valid. However, it would entail to plan multiple transports for the commodity k. After uncertainty is revealed, one can than choose which of the planned transports will be performed.

In order to formulate the tractable robust counterpart to the static variant of the RNDP, we need to introduce some dual variables. Let $\boldsymbol{\pi^j}, j \in \{1,2,3\}$ be dual variables corresponding to inequalities (7.26) - (7.28). Then, the robust counterpart is given by

$$\min \quad \sum_{p\in\mathcal{P}} c^p x^p + \sum_{((i,\tau^i),(j,\tau^j))\in\mathcal{A}^\mathcal{T}} \sum_{k\in\mathcal{K}} z^k_{(i,\tau^i),(j,\tau^j)} \tilde{c}^k_{ij} \tag{7.33}$$

$$\text{s.t.} \quad \sum_{(j,\tau)\in\delta^-((i,\tau))} z^k_{(j,\tau^j),(i,\tau^i)} - \sum_{(j,\tau)\in\delta^+((i,\tau))} z^k_{(i,\tau^i),(j,\tau^j)} = \bar{\phi}^k_{(i,\tau^i)}$$
$$\forall (i,\tau^i) \in \mathcal{N}^\mathcal{T}, k \in \mathcal{K} \tag{7.34}$$

$$\sum_{(j,\tau)\in\delta^-((i,\tau))} y^p_{(j,\tau^j),(i,\tau^i)} - \sum_{(j,\tau)\in\delta^+((i,\tau))} y^p_{(i,\tau^i),(j,\tau^j)} = \chi^p_{(i,\tau^i)} x^p$$
$$\forall (i,\tau^i) \in \mathcal{N}^\mathcal{T}, p \in \mathcal{P} \tag{7.35}$$

$$\sum_{k\in\mathcal{K}} \left[-\pi^1_{k,((i,\tau^i),(j,\tau^j))} q^k_{\min} + \pi^2_{k,((i,\tau^i),(j,\tau^j))} q^{\min}_k \right]$$
$$+ (\Delta_2 + \sum_{k\in\mathcal{K}} q^k_{\min}) \pi^3_{((i,\tau^i),(j,\tau^j))} \leq Q \sum_{p\in\mathcal{P}} y^p_{i,\tau^i} \quad \forall ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^\mathcal{T} \tag{7.36}$$

$$-\pi^1_{k,((i,\tau^i),(j,\tau^j))} + \pi^2_{k,((i,\tau^i),(j,\tau^j))} + \pi^3_{((i,\tau^i),(j,\tau^j))} \geq \sum_{k\in\mathcal{K}} z^k_{(i,\tau^i),(j,\tau^j)}$$
$$\forall ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^\mathcal{T} \tag{7.37}$$

$$z^k_{(i,\tau^i),(j,\tau^j)} \in \{0,1\} \qquad \forall((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}, k \in \mathcal{K} \tag{7.38}$$

$$y^p_{(i,\tau^i),(j,\tau^j)} \in \{0,1\} \qquad \forall((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}}, \hat{p} \in \hat{\mathcal{P}} \tag{7.39}$$

$$x^p \in \{0,1\} \qquad \forall p \in \mathcal{P} \tag{7.40}$$

$$\pi^1_{k,((i,\tau^i),(j,\tau^j))}, \pi^2_{k,((i,\tau^i),(j,\tau^j))}, \pi^3_{((i,\tau^i),(j,\tau^j))} \geq 0 \qquad \forall k \in \mathcal{K}, ((i,\tau^i),(j,\tau^j)) \in \mathcal{A}^{\mathcal{T}} \tag{7.41}$$

Objective (7.33) minimizes the costs of the selected vehicle paths and the variable commodity transportation costs. With Constraints 7.34 and (7.35) we model flow conservation of the $\boldsymbol{z}$ and $\boldsymbol{y}$ variables. Constraints (7.36) and (7.37) are the robust counterpart of the capacity constraints. As we considered the uncertainty constraint-wise, we required a 'for all arcs' identifier in these constraints. The remaining constraints indicate variable domains.

7.5 Experimental insights

The goal of this section is twofold. First, by means of an explorative example of the RNDP, we show the potential of using time-invariant vehicle paths. Second, by a numerical analysis, we compare the different models and quantify the added value of making decisions dynamically.

7.5.1 Potential of time-invariant vehicle paths

Consider the graph in Figure 7.1. Assume that $K = 3$, and that each commodity has a delivery window of 40 minutes on an overal time horizon of 1.5 hour. We discretize the time-horizon in steps of 10 minutes. We consider four commodities. Commodities 1-3 originate from node n_1 and destignate to n_2, n_2, and n_3 respectively. Commodity 4 originates from n_2 and needs to be delivered to n_4. Above the graph, the interval between $e^k_{\max}$ and $e^k_{\min} + \Delta^k_1$ is denoted for each commodity k. Furthermore, $e^k_{\min} = 0$ for all the commodities, the vehicle capacity is equal to the commodities weight, which is determenistic in this example. Finally, $\Delta_3 = 20$, indicating that total deviation of release times from the minimum values $e^3_{\min}$ is at most 20 minutes.

In Figure 7.1, we denote two time-expanded networks. The upper-one visualizes the static robust solution, and the lower-one visualizes the two-stage robust solution using time-invariant vehicle paths. It can be seen that four links are opened in both the

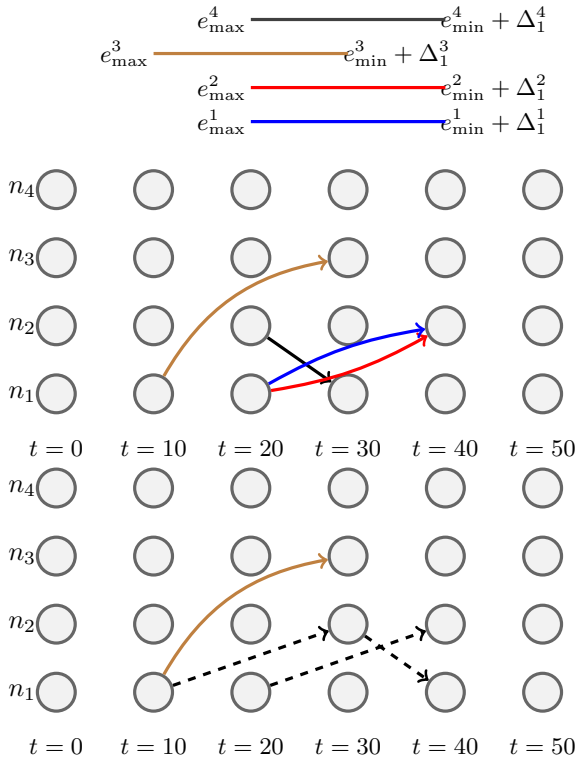


Figure 7.1: Graph associated with explorative example to show the potential of time-invariant paths

static and robust solution. However, as $\Delta_3 = 20$, there does not exist an uncertainty realization where both commodities 1 and 2 are released at $t = 20$. If one of them releases after $t = 10$, the other one will be available for pickup at $t = 10$. But, for a static robust solution, where we need to determine the commodities traveling along the chosen vehicle paths, it is impossible to anticipate upon this behavior. Hence, two vehicle paths starting at time $t = 20$ are used in the robust solution. This leads to using four vehicles.

In the two-stage robust solution, we can anticipate upon the limited deviation from the minimum release times. That is, we select a time-invariant vehicle path that will start at $t = 10$ and takes either commodity 1 or 2 (i.e., the one that will be available) and delivers it to node n_2, where it can then take commodity 4 and delivers that to node n_1. In total, only three vehicles are used, which is a 25% reduction compared to the static robust solution that uses 4 vehicles.

The above depicted benefit indicates that not-assigning commodities to the vehicle paths in the first-stage is promising. Next, we sketch when assigning departure and arrival times in the second-stage is beneficial.

Consider a situation where $e^k \sim U(0, 10)$ with probability 0.5 and $e^k \sim U(30, 40)$ with probability 0.5. Not using time-invariant vehicle paths then implies to treat each such commodity k as two distinct commodities, and to ensure that the distinct commodities are scheduled at both times. However, using time-invariant vehicle paths there is no need to consider two distinct commodities. Indeed, one can imagine to select a time-invariant vehicle path that can transport this commodity under each realization of e^k.

Finally, consider the following example, as depicted in Figure 7.2. Here, a small graphs is given with four locations $n_1, \ldots, n_4$, and links $(n_1, n_2), (n_2, n_3)$, and (n_3, n_4). The travel time along the links is indicicated in Figure 7.2. There are three commodities to transport, each with destination n_4 but with distinct origins n_1, n_2, and n_3. We consider a time horizon $[0, 80]$ and we assume there are no capacity restrictions. Let the earliest possible pickup time e^k at the origin locations of commodity $k \in \{1, 2, 3\}$ be realizations from a budget uncertainty set $B = \{\mathbf{e} \mid \sum_{k=1}^{3}(e^k - e^k_{\min}) \leq 20, e^k \leq e^k_{\max}\}$, with $e^k_{\min} = 0, 20, 40$ and $e^k_{\max} = 20, 40$, and 60 for $k = 1, 2$, and 3, respectively. Let delivery time be equal to 15 for each commodity. Clearly, the static robost solution implies to assign a vehicle dedicated for each transportation. However, a two-stage robust solution using time-invariant vehicle paths will only use two vehicles, as for any realization of uncertain parameters one vehicle path can always transport two of the commodities.

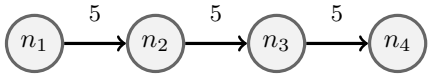


Figure 7.2: Graph to illustrate the potential of time-invariant vehicle paths

7.5.2 Numerical examples

To show the potential of the introduced models, and to explore the potential of two-stage robust decision making compared to static decision making, we perform a short numerical analysis.

We create three small-scale instances that vary in the number of nodes, arcs, and commodities required to transport, see Table 7.1 for the details. The nodes are randomly drawn in a 1000 by 1000 block, and arcs are inserted randomly between the nodes ensuring that there are no duplicates. The travel speed of the vehicle equals

100 so that traveling from corner to corner in the 1000 by 1000 block is at most $1000\sqrt{2}/100 \approx 29$ minutes. The vehicle capacity equals $Q = 50$ for all the instances. We include a fixed vehicle cost of 100 for each selected (time-invariant) vehicle path, and traveling along an arc equals the distance between the nodes divided by 200.

The uncertainty set is randomly generated for each instance as follows. The minimum weight $q^k_{\min}$ is randomly drawn between 5 and 10 for each commodity k, and its associated maximum weight $q^{\max}$ equals its minimum weight plus a randomly drawn number between 1 and 10. The release times e^k are drawn from a uniform distribution between 0 and 20, and the associated maximum release times $e^k_{\max}$ equals $e^k_{\min}$ plus a random uniformly drawn number between 30 and 80. This gives a good mix of express deliveries (with relatively short delivery windwos) and normal deliveries.

The commodities origin and destination locations are selected randomly, and duplicates are possible. Note that due to different temporal characteristics this still requires to treat them independently. Variable commodity transportation costs are included and are randomly distributed between 3 and 8.

All models are implemented in C++ using the constraint programming framework SCIP 6.0 in combination with CPLEX 12.8 to solve LP relaxations. All computations are obtained single-threaded and are performed on an i7-4500U CPU @ 1.80GHz.

In Table 7, we compare the resulting objective value of the single-stage, static robust solution ("Static obj") with a proxy for the 2-stage, dynamic robust solution ("Dynamic sol."). The latter is obtained by considering a randomly drawn number of scenarios (replicated 5 times and taking the worst outcome) that are generated so that the budget constraints for time and weight deviations hold with equality. We considered this 2-stage robust solution for 2, 3 and 4 scenarios included in the large-cale MIP formulation. Finally, all the reported numbers are averages for randomly generated route sets, where at least each commodity could be transported directly.

A few observations stand out. First, the reported objective values of the 2-stage robust solution with 4 scenarios are on average 12.52% lower than the single stage robust solution. Hence, dynamic decion making seems to have much potential as well in these numerical experiments. Second, the objective value of the dynamic solution does not increase much by considering more scenarios. This is in line with the observations of Zeng and Zhao (2013) where indeed only a small number of worst-case scenarios where generated. We finally like to mention that the computation times are all within a few minutes.

Inst.	N	A	K	Static sol.	# Scen.	Dynamic sol.	% Improvement
1	5	15	10	534.6	2	478.2	10.55
					3	480.6	10.10
					4	482.2	9.80
2	5	20	15	761.2	2	626	17.76
					3	646	15.13
					4	648.4	14.82
3	6	24	12	757.4	2	635.4	16.11
					3	656	13.39
					4	659.4	12.94

Table 7.1: Computational results for 3 small-scale instances

7.6 Conclusion

We studied the value of dynamic (i..e, two-stage) decision making in network design problems by introducing the concept of *time-invariant vehicle paths*. In the context of two-stage decision making under uncertainty, a time-invariant vehicle path consists of an a-priori selection and sequence of geographical locations. After observing uncertainty, as a second stage decision, one needs to assign departure times to the visited locations of the time-invariant vehicle path. This has a practical motivation: It provides guidance for the involved practictioners as visited locations are equal for the day-to-day operations, only the departure and arrival times may vary slightly.

We applied this concept in a two-stage robust network design problem with temporal characterics. Here, we take the viewpoint of a single consolidation carrier that is responsible for transporting multiple commodities between distinct origin and destination locations. We investigate the impact of uncertain temporal characteristics and commodity weights on robust decision making. With temporal characteristics we refer to an earliest available pickup time and a latest possible delivery time, and with robust decision making we indicate that we minimize the sum of first-stage costs and worst-case but minimized second-stage costs.

We modeled this by a combination of flat and time-expanded networks. We provide a generic two-stage robust integer programming formulation, as well as its scenario-based large-scale mixed integer programming programming formulation. This latter serves, for a given set of scenarios, as a lower bound for the optimal two-stage robust solution. To estimate the value of dynamic decision making, we also provided the static, single-stage robust optimization model of the dynamic network design problem.

In a short numerical section, we showed by means of two examples the potential of

time-invariant vehicle paths. Thereafter, we provided a small numerical experiment where we showed the potential of dynamic decision making. Namely, dynamic decision making leads to a decrease in objective value of 12.52% compared to its static counterpart.

There are ample of opportunities for further research. Let us first discuss the two clear extensions from an algorithmic point of view.

First, we considered a given set of time-invariant paths and evaluated these with the tools developed in this paper. However, one could utilize the concepts presented in this paper and develop a branch-and-price(-and-cut) framework that dynamically generates these time-invariant vehicle paths. This will give less dependencies on initalizing the models with vehicle paths and provides opportunities for solving large-scale instances.

Second, we take a given time discretiziation as given. For the deterministic version of the RNDP, good results are obtained by dynamically generating time epochs in the considered discretization. We urge future researchers to also explore algorithmic oppertunities in this area, as we observed that for fine discretizations the RNDP becomes difficult to solve.

Another avenue for future research is to increase the level of dynamism, thus to consider real-time decision making. This relates to dynamically solving pickup-and-delivery problems, which has received a fair amount of attention in the last few decades. However, important questions, such as determining the required capacity of the network (i.e., number of vehicles) to robustly perform day-to-day operations, are still left unanswered.

We, finally, mention possible extensions of this work by considering more practical factors such as time-dependent travel times, restricted loading and unloading space for city-distribution networks in dense inner cities, or a non-homogeneous fleet of vehicles including electric vans and bike couriers. Including such practical factors would enhance applicability to real-life scenarios, and research is required to find out what the potential of time-invariant vehicle paths would be in such situations.

Chapter 8

Concluding remarks

We studied optimization problems inspired by recent developments in distributed logistics. Namely, the increasing awareness of sustainability in society and the ongoing struggle of e-commerce companies to become profitable in a landscape of fierce competition. In this context, we focussed on two emerging fields. The first emerging field, which we study in Part I of this thesis, is that of offshore wind maintenance service logistics. The offshore wind industry needs significant cost-savings in the operation and maintenance phase to become competitive with traditional energy suppliers such as coal-based power plants. We develop new methods to determine the short-term maintenance planning, and focused on maintenance planning on a tactical level of decision making as well. The second emerging field, which we study in Part II of this thesis, is that of transportation and logistics related to e-commerce activities. Here, we focused on the design of efficient warehouse order processing methods that incorporate nowaday's key characteristics such as product returns and the many but relatively small customer orders. Besides, we studied the implications of the increasing service requirements (e.g., same-day deliveries) on the design of city distribution delivery networks. Each of the problems that we considered is addressed from an Operations Research perspective. We developed new models and methods that enrich the extant literature and, consequently, are used to analyse the dynamics underlying the optimization problems to be able to provide decision support for practitioners.

In summary, a wide variety of optimization problems inspired by different practical scenarios were considered in this thesis. This wide variety clearly shows the abundance of exciting research that can (and should) be done in this area. Much more research is required to fully understand the dynamics, challenges, and opportunities underlying many recent innovations observed in the applications of distributed logistics. The

extent to which current and future businesses can take advantage of these developments will determine if they stay profitable and stay ahead of competitors, and thereby survive in this field of fierce competition. We, therefore, conclude with restating the claim in this thesis' introduction. Indeed, it is an exciting era to work on optimization methods in distributed logistics. It is a field where the continuation of adopting new technology and the development of novel business models is more than ever prevalent.

In the following, we conclude and discuss the main results of this thesis' both parts.

8.1 Offshore wind logistics

In Chapters 2-4, we introduced new optimization models that relate to planning maintenance activities at offshore wind farms. We address these problems from an Operations Research perspective focusing on the underlying mathematical problems that relate to vehicle routing and network design. This is in high contrast with most of the literature on offshore wind maintenance service logistics, which predominantly describes new decision support tools and reliability engineering and maintenance-oriented studies. We studied the essence of the underlying mathematical models driving operational and tactical operations. Consequently, we increased the understanding on a fundamental level, which can be incorporated in tomorrow's decision making for real-life offshore wind farms. In the following, we first summarize our main findings and afterward provide suggestions for future research while discussing our work.

8.1.1 Conclusions

In Chapter 2, we addressed a single wind-farm, single depot (i.e., port) scenario that is often encountered in practical situations of offshore wind. We formally introduce the Multiperiod Service Planning and Routing Problem (MSPRP), in which one determines an optimal short-term maintenance planning. The major complicating factor is that the maintenance tasks require distinct sets of differently skilled (and scarcely available) technicians as well as spare parts. Both technicians and spare parts need to be transported by a limited fleet of vessels, resulting in a complex optimization problem. With the development of resource-exceeding route inequalities, we were able to develop an efficient branch-and-price-and-cut algorithm for solving this problem to optimality. The inequalities developed can be applied to any routing problem that involves a scarce set of resources. Examples include forestry applications in which heavy equipment needs to be distributed daily to distinct worksites. For the MSPRP, the inequalities were found to be very effective by reducing root node optimality gaps

by on average 63.20%. Another troubling difficulty, in an algorithmic-technical sense, is the inclusion of fixed costs for technicians going offshore. The fixed costs cause a violation of some fundamental assumptions that are crucial for efficient column generation techniques. We overcame this problem by developing a tailored algorithm for pricing new columns. In the end, the branch-and-price-and-cut algorithm developed can solve instances of up to 92 nodes and 21 periods to optimality. Besides, we observe that the structure of the optimal solutions depends heavily on the trade-off between travel costs, maintenance costs, and technician costs. Reusing technicians for multiple maintenance tasks results in a higher mean time to maintenance. On the other hand, using technicians simultaneously leads to higher (fixed) technician costs but a lower mean time to maintenance. Depending on the actual parameters, differently structured routes and associated maintenance plannings will be developed.

In Chapter 3, we extended the setting studied in Chapter 2 to multiple wind farms and multiple depots (i.e., ports). Nowadays, it is observed that a single maintenance service provider becomes responsible for the maintenance of multiple wind farms. Here, operations are typically organized from distinct operating and maintenance bases (i.e., ports or depots). In particular, we were interested in the impact of coordinating activities as a whole instead of organizing maintenance operations in isolation at each operating and maintenance base. In other words, we studied the cost-savings potential of collectively using the technicians. We did so by introducing the Technician Allocation and Routing Problem (TARP), and two variants that allow for a certain extent of technician sharing. Whereas in the TARP technicians are not shared, the first variant allows determining the technician allocation once before the operations begin. In the second variant, we allowed for a daily reassignment of technicians among operating and maintenance bases. The TARP is thought to be intractable for exact solution methods. Therefore, we proposed an Adaptive Large Neighborhood Search that has been shown to find high-quality, and often optimal solutions on a set of existing benchmark instances in the literature. Using the ALNS, we show that the daily reassignment of the scarcely available technicians reduces, on average, costs by around 7%. Besides, the mean time to maintenance is reduced, while fewer vessel trips are needed. The ALNS is an attractive alternative to existing exact solution methods for multiple wind farms with multiple operating bases as the computation times are rather short. Such a fast solution method is especially helpful for practitioners that would like to use such optimization methods on a dynamic basis.

In Chapter 4, we take a tactical point-of-view to the planning of maintenance activities at offshore wind farms. In this setting, we formally introduced the Stochastic Maintenance Fleet Transportation Problem for Offshore wind farms (SMFTPO). In

this problem, we aim to find a minimizing assignment of maintenance tasks to vessels while controlling for uncertain maintenance tasks and weather conditions. We modeled the SMFTPO as a two-stage stochastic mixed integer programming problem, and solve it using a Sample Average Approximation algorithm. In the first stage, we assign vessels (that are already owned or chartered) to ports. In the second stage, after observing uncertain maintenance tasks and weather conditions, we assign maintenance tasks to the vessels. Different from the existing literature, we take the viewpoint of a single maintenance service provider that does not own the wind farm. Hence, the service provider does not bear the risk of downtime costs. The maintenance provider's only needs to adhere to minimum service requirements specifying contractually binding performance criteria. We consider three of such requirements, namely, to schedule all maintenance tasks, to leave a fraction unscheduled, and to incentivize performing maintenance rather quickly. These stylized settings suffice in providing realistic cost-estimations for performing maintenance under different contractual incentives.

Moreover, in Chapter 4 as well, we provide a review of common assumptions on an operational level for tactical decision making in offshore wind. These assumptions directly impact the second-stage decision of the SMFTPO, namely, how maintenance tasks are assigned to maintenance activities. We extensively compare the impact of different assumptions on the resulting estimation of the operational costs by solving reformulated second-stage problems. We find that for single wind farm scenarios, there is no need to preprocess maintenance tasks in so-called maintenance bundles (i.e., complete daily tasks). Furthermore, allowing for 2% of maintenance tasks unscheduled leads to cost reductions around 8%, and to give an incentive to perform maintenance rather quickly leads to cost increases around 10%. Consequently, we solve the two-stage stochastic programming models by means of sample average approximation. Incorporating the stochastic nature of maintenance tasks is shown to be crucial for these type of tactical decisions since the value of the stochastic solution is large. Besides, we estimate the expected value of perfect information around 10%.

8.1.2 Discussion and future research

In Chapters 2 and 3, we obtained a thorough mathematical and computational understanding of the impact of the scarce availability of technicians with different skills. In both the MSPRP (Chapter 2) and the TARP (Chapter 3), deterministic knowledge on future weather conditions is assumed to be available. This is true for many scenarios; for instance, during the summer months in which weather conditions are reasonably predictable. However, in the months preceding and after the winter,

this is less likely. Hence, it is valuable to extend the line of research for determining short-term maintenance plannings by considering stochastic (or at least predictions of) weather conditions. A two-stage stochastic optimization model on a rolling horizon basis is then a suitable approach.

Another limiting assumption in Chapters 2-4 is the assumed availability of spare parts. In practice, spare parts might not be available, and the corresponding maintenance tasks can therefore not always be scheduled in each period. Hence, further integration with advanced inventory control concepts is deemed required to increase the embracement of the models developed in this thesis by practitioners.

In Chapters 2 and 3, we observed that the availability and costs of technicians have a high impact on the structure of the optimal solutions. For practitioners, it is, therefore, essential to hire and assign the right technicians to the proper operating and maintenance basis. In line with existing research, we assumed that vessels work independently and, consequently, technicians cannot be transferred between vessels during the day. However, additional efficiency will be obtained if coordinated routing is employed. Then, technicians can be transferred between vessels either directly or via turbines, which is nowadays allowed due to advanced medical training of the technicians. Such advanced control concepts touch upon the notion of transshipments or transfers in pickup-and-delivery problems, which is a very complicating characteristic for routing problems. Extending our research in this direction then offers opportunities from a methodological point of view, as well as it enhances the practicality of the models studied.

We argued in Chapter 4 that single, large maintenance providers are becoming responsible for maintaining multiple wind farms, and do not bear the risk of the turbines' downtime. The cost-structures underlying the MSPRP and TARP in Chapter 2 and 3 are flexible concerning whether or not these downtime costs are incorporated. Nevertheless, the problems studied in this thesis are considered in isolation of the energy market. A further extension of our research will be the consideration of an offshore wind farm that is required to predict its future energy production. This incorporation puts new restrictions on the extent to which maintenance activities can take place. Also, sophisticated decision-making might result in a more reliable prediction of future energy production, possibly leading to an on-average higher output of the offshore wind farm.

In Chapter 4, we showed that assumptions on an operational level are required for solving tactical (and strategical) maintenance planning problems. There, we give an exact characterization of the operational activities for varying assumptions. However, it might be valuable to develop approximation algorithms for the operational problems

studied in Chapters 2 and 3. Then, such approximations can be used in tactical (or strategic) models as considered in Chapter 3. This will enhance the solvability of large and long-term strategic models at the expense of not modeling the operational activities exactly. Such approximation algorithms open the door for fewer restrictions on the operational problem, and further research should show whether such an approach will result in reliable cost-predictions.

Finally, as the offshore wind market will get more mature, there might develop a need for dynamic decision making. In Chapter 4, we showed there is still an estimated gap of 10% between the proposed two-stage stochastic programming solutions and the expected value of perfect information. Although the latter is not achievable in practice, it is promising to enhance the level of dynamism in our models (and solution approaches) to come closer to the perfect information situation. Hence, there is a need to develop optimization models based on stochastic (approximate) dynamic programming. Another dynamic aspect of offshore wind farms is the abundance of data available which describes the condition of (spare parts of) a turbine. This, typically real-time information, might be incorporated in such dynamic models as well.

Concluding, our work on offshore wind logistics is part of a research stream that is a natural starting point for many more exciting research topics. The main objectives should then be to integrate short-term maintenance planning problems with other domains of importance for offshore wind logistics. Examples include condition monitoring, spare parts inventory management, the dynamics of energy markets, and a computationally tractable integration with large-scale strategic planning problems.

8.2 E-commerce logistics

In Chapters 5-7, we introduced new optimization models inspired on emerging topics related to the rise of e-commerce companies. The primary motivating development for the design of these models is the increasing number of customer orders that are increasingly complicated to process (e.g., same-day delivery). On the one hand, new paradigms for customer order processing in (city) distribution centers are required to deal with a large number of typically small-sized customer orders and the significant stream of product returns. On the other hand, this necessitates the design of city distribution networks in which temporal (i.e., time) aspects of the distribution are managed precisely to enable efficient consolidation of parcels.

8.2.1 Conclusions

In Chapter 5, we studied how to include the restocking of returned products in regular order picker routes in picker-to-parts warehouses. In this Traveling Salesman Problem with Pickup and Deliveries (TSPPD), one designs a route in a warehouse so that all deliveries (that are initially on the depot) are brought back to their storage locations while a set of new products is picked. We develop a Hybrid Genetic Algorithm to solve the TSPPD. The genetic algorithm provides high-quality and often optimal solutions for small-scale instances, as is shown by a comparison against solving a MIP formulation for the TSPPD with a commercial solver. Experiments on a carefully constructed case did show that integration of product returns can lead to 23.48% less distance traveled, compared to processing the product returns and new orders separately. Besides, we also studied the effect of order picker blocking by considering multiple order pickers that travel simultaneously through the warehouse. For a given start-time, we count the number of times the order pickers would cross in the warehouse while they traverse their routes. To find high-quality solutions that incorporate this penalty, we proposed another Hybrid Genetic Algorithm. We found that small alterations to the order picker routes (only leading to marginal increases in travel distance) can almost eliminate picker blocking (or any interaction). In other words, multiple near-optimal solutions are significantly different from each other. Therefore, secondary objectives can be successfully incorporated relatively easily.

In Chapter 6, we extend the concept of integrating product returns in regular order picking processes (as studied in Chapter 5) by considering a setting that goes beyond order picker routing only. In the Generalized Joint Order Batching, Assignment, Sequencing, and Routing Problem (G-JOBASRP), we consider next to order picker routing, the design of order picker batches, the assignment of batches to order pickers, and the sequencing of those batches for each order picker. We aim to minimize the sum of routing costs (inside the warehouse), tardiness costs (i.e., violation of customer deadlines), and so-called split-up costs. These latter costs are stylized costs that resemble additional handling operations (outside our modeled operations) resulting from splitting up customer orders amongst multiple order-picking batches. We develop a parallel Adaptive Large Neighbourhood Search which significantly outperforms commonly-used, practical heuristics for instances up to 8038 to be processed order lines.

We show that integrating product returns in the G-JOBASRP is particularly useful for warehouses not working at their full capacity, which is throughout most of the year. Besides, we observe that the total costs can be reduced by up to 45% if customer orders can freely be split-up, compared to if customer orders cannot be split-up. For

small values of the stylized split-up cost, we find that significant cost-savings can still be obtained. It is observed that increasing split-up costs predominantly leads to an increase in travel costs instead of a major rise in incurred split-up costs. Apparently, it is cost-efficient to avoid customer split-ups at the costs of increasing travel costs. This last observation also strengthens the claim made in Chapter 6 that the TSPPD has relatively many near-optimal but structurally different solutions. Also in the G-JOBASRP, routes with slightly higher travel costs are chosen instead of simply incurring split-up costs. The structure of these routes is significantly different from the case without split-up costs.

In Chapter 7, we took a more strategic view of e-commerce operations. Instead of focussing on warehouse operations, we studied the design of robust city logistics networks. Such networks are designed so that day-to-day operations subject to temporal (i.e., time) uncertainties can be organized efficiently. We formalized this in the two-stage network design problem with temporal characteristics (RND), which represents the design of a network to enable commodity streams between (city) distribution centers. In the RND, we make decisions in two-stages. In the first stage, we select vehicle paths through the given network without determining actual departure or arrival times. Then, we observe the stochastic temporal characteristics, including the earliest possible pickup time and the latest possible delivery time of the commodities to be transported in the network. In the second stage, we decide upon the departure and arrival times of the vehicle paths chosen in the first stage. By taking a robust approach, we ensure that the sum of first-stage vehicle path costs and worst-case second stage costs are minimized. This is, by the best of our knowledge, the first study that takes such a robust optimization approach in the context of city distribution networks with temporal characteristics. This is relatively surprising, as network design with temporal characteristics is pre-eminent an application in which small disturbances have a high impact on the validity of the solutions proposed. We provided a general two-stage robust integer formulation, its associated large-scale scenario-based MIP formulation, and a tractable counterpart of the static variant of the problem considered. A detailed analysis showed the benefit of using the concept we propose.

8.2.2 Discussion and future research

From Chapter 5 and 6, we obtained a good understanding of the impact of incorporating product returns in regular order picking processes. Both the TSPPD and the G-JOBASRP are, however, based upon a perfect knowledge of customer demand. With the increase of same-day delivery or even 'within two hours delivery', more dynamic

optimization approaches become relevant. Real-time routing of order pickers, or the real-time routing of robots in mechanized warehouse concepts, might, therefore, be of particular interest for future researchers. Approximate dynamic programming approaches will become usefull to solve such problems.

Moreover, both the TSPPD and the G-JOBASRP assumed a random storage location assignment of products to warehouses. In practice, one typically exploits storage assignment strategies based upon the frequency of products being ordered and the correlation between product orders. The integration of storage assignment into the G-JOBASRP is, therefore, the next step into solving integrated warehouse order processing problems. Besides this integration inside the warehouse, other integrations outside the warehouse are prevalent as well. Consider, for instance, the scheduling of trucks to pick up and deliver shipments to and from the warehouse. However, such integrated problems will become intractable to solve, even with metaheuristic techniques. It is, therefore, that approximations of operational performance inside the warehouse might be useful for such integrated settings. To the best of the author's knowledge, no such research has yet been done despite its practical relevancy.

Besides, whereas we considered the impact of order picker interaction (e.g., blocking in narrow aisles) in the TSPPD (Chapter 5), we did not consider this in the G-JOBASRP. Including this aspect might reduce the practicality of splitting customer orders amongst multiple batches, as order picker routes are likely to consists of only a few visited aisles, leading to more interaction and delays due to order pickers crossing. Furthermore, precise modeling of operations outside the warehouse might shed more light on the impact of splitting up customer orders. This will quantify the effect of splitting up customer orders, and making this visible is crucial for the embracement by practitioners.

Furthermore, there is the need for more rigid exact solution methods based upon column generation or Lagrangian relaxation, especially for the large-scale instances considered in the G-JOBASRP and encountered in practice nowadays. If such approaches were developed, it would enable the assessment of the quality of the proposed solutions for such large-scale instances. Only then it will be known if more research should be spent on the development of metaheuristic (or other heuristic) approaches for solving (variations of) the G-JOBASRP.

Many opportunities for future research arise in the context of network design and robust optimization. First, and foremost, the presented algorithm in Chapter 5 is not yet able to provide optimal solutions to large-scale RNDP instances. A column-generation based method is required on top of efficiently solving the second-stage subproblem. The latter involves benders decomposition methods, and consequently, Benders decom-

position is included as part of the column-generation procedure. However, this choice is somewhat arbitrary as one could also provide a Benders decomposition method on top of column-generation subprocedures. Future research will determine which structure is more computationally efficient and if there are possibilities to enhance one of the layers with information obtained from the other layer by, for instance, developing valid inequalities. Another avenue for future research is the development of more dynamic methods. Opportunities range from multi-stage robust optimization problems to complete dynamic settings solved with approximate dynamic programming techniques.

Bibliography

Ahmadi-Javid A, Seddighi AH, 2012 *A location-routing-inventory model for designing multi-source distribution networks. Engineering Optimization* 44(6):637–656.

Alinaghian M, Shokouhi N, 2018 *Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. Omega* 76:85–99.

Andersen J, Christiansen M, Crainic TG, Grønhaug R, 2011 *Branch and price for service network design with asset management constraints. Transportation Science* 45(1):33–49.

Ardestani-Jaafari A, Delage E, 2017 *The value of flexibility in robust location–transportation problems. Transportation Science* 52(1):189–209.

Atamtürk A, Zhang M, 2007 *Two-stage robust network flow and design under demand uncertainty. Operations Research* 55(4):662–673.

Baldacci R, Bartolini E, Mingozzi A, 2011 *An exact algorithm for the pickup and delivery problem with time windows. Operations Research* 59(2):414–426.

Barnhart C, Hane CA, Vance PH, 2000 *Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Operations Research* 48(2):318–326.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MW, Vance PH, 1998 *Branch-and-price: Column generation for solving huge integer programs. Operations Research* 46(3):316–329.

Battarra M, Cordeau JF, Iori M, 2014 *Chapter 6: pickup-and-delivery problems for goods transportation.* Toth P, Vigo D, eds., *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, 161–191 (SIAM).

Ben-Tal A, Goryashko A, Guslitzer E, Nemirovski A, 2004 *Adjustable robust solutions of uncertain linear programs. Mathematical Programming* 99(2):351–376.

Ben-Tal A, Nemirovski A, 2002 *Robust optimization–methodology and applications. Mathematical Programming* 92(3):453–480.

Berbeglia G, Cordeau JF, Gribkovskaia I, Laporte G, 2007 *Rejoinder on: Static pickup and delivery problems: A classification scheme and survey. TOP* 15(1):45–47.

Berman O, Krass D, Mahdi Tajbakhsh M, 2011 *On the benefits of risk pooling in inventory management. Production and Operations Management* 20(1):57–71.

Bertsimas D, Caramanis C, 2007 *Adaptability via sampling. 2007 46th IEEE Conference on Decision and Control*, 4717–4722 (IEEE).

Bertsimas D, Caramanis C, 2010 *Finite adaptability in multistage linear optimization. IEEE Transactions on Automatic Control* 55(12):2751–2766.

Bertsimas D, Dunning I, 2016 *Multistage robust mixed-integer optimization with adaptive partitions. Operations Research* 64(4):980–998.

Bertsimas D, Georghiou A, 2015 *Design of near optimal decision rules in multistage adaptive mixed-integer optimization. Operations Research* 63(3):610–627.

Bertsimas D, Iancu DA, Parrilo PA, 2010 *Optimality of affine policies in multistage robust optimization. Mathematics of Operations Research* 35(2):363–394.

Bertsimas D, Nasrabadi E, Stiller S, 2013 *Robust and adaptive network flows. Operations Research* 61(5):1218–1242.

Bhandari A, Scheller-Wolf A, Harchol-Balter M, 2008 *An exact and efficient algorithm for the constrained dynamic operator staffing problem for call centers. Management Science* 54(2):339–353.

Boland N, Hewitt M, Marshall L, Savelsbergh M, 2017 *The continuous-time service network design problem. Operations Research* 65(5):1303–1321.

Boysen N, De Koster R, Weidinger F, 2019 *Warehousing in the e-commerce era: A survey. European Journal of Operational Research* 277(2):396–411.

Boysen N, Stephan K, Weidinger F, 2019 *Manual order consolidation with put walls: the batched order bin sequencing problem. EURO Journal on Transportation and Logistics* (2):1–25.

Braekers K, Ramaekers K, Van Nieuwenhuyse I, 2016 *The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering* 99:300–313.

Büdenbender K, Grünert T, Sebastian HJ, 2000 *A hybrid tabu search/branch-and-bound algorithm for the direct flight network design problem. Transportation Science* 34(4):364–380.

Bulhões T, Subramanian A, Erdoğan G, Laporte G, 2018 *The static bike relocation problem with multiple vehicles and visits. European Journal of Operational Research* 264(2):508–523.

Cacchiani V, Jünger M, Liers F, Lodi A, Schmidt DR, 2016 *Single-commodity robust network design with finite and hose demand sets. Mathematical Programming* 157(1):297–342.

Campbell AM, Savelsbergh M, 2006 *Incentive schemes for attended home delivery services. Transportation Science* 40(3):327–341.

Cardona M, Duch-Brown N, Francois J, Martens B, Yang F, et al., 2015 *The macro-economic impact of e-commerce in the eu digital single market. Institute for Prospective Technological Studies. Digital Economy Working Paper* 9.

Çelik M, Ergun Ö, Keskinocak P, 2015 *The post-disaster debris clearance problem under incomplete information. Operations Research* 63(1):65–85.

Chan FT, Chan H, 2011 *Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage. Expert Systems with Applications* 38(3):2686–2700.

Chen F, Wang H, Qi C, Xie Y, 2013 *An ant colony optimization routing algorithm for two order pickers with congestion consideration. Computers & Industrial Engineering* 66(1):77–85.

Chen F, Wang H, Xie Y, Qi C, 2016 *An aco-based online routing method for multiple order pickers with congestion consideration in warehouse. Journal of Intelligent Manufacturing* 27(2):389–408.

Chen F, Wei Y, Wang H, 2018 *A heuristic based batching and assigning method for online customer orders. Flexible Services and Manufacturing Journal* 30(4):640–685.

Chen MC, Huang CL, Chen KY, Wu HP, 2005 *Aggregation of orders in distribution centers using data mining. Expert Systems with Applications* 28(3):453–460.

Chen TL, Cheng CY, Chen YY, Chan LK, 2015 *An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. International Journal of Production Economics* 159:158–167.

Chen X, Thomas BW, Hewitt M, 2016 *The technician routing problem with experience-based service times. Omega* 61:49–61.

Chen X, Thomas BW, Hewitt M, 2017 *Multi-period technician scheduling with experience-based service times and stochastic customers. Computers & Operations Research* 82:1–14.

Chouman M, Crainic TG, Gendron B, 2016 *Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. Transportation Science* 51(2):650–667.

Ciancio C, Laganà D, Musmanno R, Santoro F, 2018 *An integrated algorithm for shift scheduling problems for local public transport companies. Omega* 75:139–153.

Crainic TG, 2000 *Service network design in freight transportation. European Journal of Operational Research* 122(2):272–288.

Crainic TG, Hewitt M, Toulouse M, Vu DM, 2014 *Service network design with resource constraints. Transportation Science* 50(4):1380–1393.

Dai L, Stålhane M, Utne IB, 2015 *Routing and scheduling of maintenance fleet for offshore wind farms. Wind Engineering* 39(1):15–30.

Davarzani H, Norrman A, 2015 *Toward a relevant agenda for warehousing research: literature review and practitioners input. Logistics Research* 8(1):1.

De Koster MBM, Van Der Poort ES, Wolters M, 1999 *Efficient orderbatching methods in warehouses. International Journal of Production Research* 37(7):1479–1504.

De Koster R, De Brito MP, de Vendel MA, 2002 *Return handling: an exploratory study with nine retailer warehouses. International Journal of Retail & Distribution Management* 30(8):407–421.

De Koster R, Le-Duc T, Roodbergen KJ, 2007 *Design and control of warehouse order picking: A literature review. European Journal of Operational Research* 182(2):481–501.

De Koster RB, De Brito MP, Van De Vendel MA, 2002 *Return handling: an exploratory study with nine retailer warehouses. International Journal of Retail & Distribution Management* 30(8):407–421.

Dejax PJ, Crainic TG, 1987 *Survey paper: A review of empty flows and fleet management models in freight transportation. Transportation Science* 21(4):227–248.

Dell M, Iori M, Novellani S, Stützle T, et al., 2016 *A destroy and repair algorithm for the bike sharing rebalancing problem. Computers & Operations Research* 71:149–162.

Desrochers M, 1987 *La fabrication d'horaires de travail pour les conducteurs d'autobus par une methode de generation de colonnes.* Ph.D. thesis.

Desrochers M, Desrosiers J, Solomon M, 1992 *A new optimization algorithm for the vehicle routing problem with time windows. Operations Research* 40(2):342–354.

Detti P, Papalini F, de Lara GZM, 2017 *A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare. Omega* 70:1–14.

Dumas Y, Desrosiers J, Soumis F, 1991 *The pickup and delivery problem with time windows. European Journal of Operational Research* 54(1):7–22.

Enthoven DLJU, Jargalsaikhan B, Roodbergen KJ, Uit het Broek MAJ, Schrotenboer AH, 2019 *The two-echelon vehicle routing problem with covering options.* Revised and Resubmitted.

EY, 2015 *Offshore wind in Europe: Walking the tightrope to success* .

Faugère L, Montreuil B, 2018 *Smart locker bank design optimization for urban omnichannel logistics: Assessing monolithic vs. modular configurations. Computers & Industrial Engineering* In press.

Feillet D, Dejax P, Gendreau M, Gueguen C, 2004 *An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks* 44(3):216–229.

Feillet D, Gendreau M, Medaglia AL, Walteros JL, 2010 *A note on branch-and-cut-and-price. Operations Research Letters* 38(5):346–353.

Ferreira RS, Feinstein CD, Barroso LA, 2014 *Operation and maintenance contracts for wind turbines.* Sanz-Bobi MA, ed., *Use, Operation and Maintenance of Renewable Energy Systems*, 145–181 (Springer).

Fischetti M, Fraccaro M, 2019 *Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. Computers & Operations Research* 106:289–297.

Froger A, Gendreau M, Mendoza JE, Pinson E, Rousseau LM, 2016 *Maintenance scheduling in the electricity industry: A literature review. European Journal of Operational Research* 251(3):695–706.

Gademann A, Van Den Berg J, Van Der Hoff HH, 2001 *An order batching algorithm for wave picking in a parallel-aisle warehouse. IIE Transactions* 33(5):385–398.

Gademann N, Van De Velde S, 2005 *Order batching to minimize total travel time in a parallel-aisle warehouse. IIE Transactions* 37(1):63–75.

Gamrath G, Fischer T, Gally T, Gleixner AM, Hendel G, Koch T, Maher SJ, Miltenberger M, Müller B, Pfetsch ME, Puchert C, Rehfeldt D, Schenker S, Schwarz R, Serrano F, Shinano Y, Vigerske S, Weninger D, Winkler M, Witt JT, Witzig J, 2016 *The scip optimization suite 3.2.* Technical Report 15-60, ZIB, Takustr.7, 14195 Berlin.

Gendron B, Crainic TG, Frangioni A, 1999 *Multicommodity capacitated network design.* Sansò Brunilde SP, ed., *Telecommunications Network Planning*, 1–19 (Springer).

Ghilas V, Demir E, Woensel TV, 2016 *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. Computers & Operations Research* 72:12–30.

GL Garrad Hassan, 2013 *A guide to UK offshore wind operations and maintenance.*

Gong Y, De Koster M, 2009 *Approximate optimal order batch sizes in a parallel aisle warehouse. Lecture notes in economics and mathematical systems*, volume 619, 175–194 (L. Bertazzi and M. Grazia Speranza and J.A.E.E. Nunen (editors)).

Gong Y, De Koster RBM, 2011 *A review on stochastic models and analysis of warehouse operations. Logistics Research* 3(4):191–205.

Gorissen BL, Yanıkoğlu İ, Den Hertog D, 2015 *A practical guide to robust optimization. Omega* 53:124–137.

Gschwind T, Irnich S, Rothenbächer AK, Tilk C, 2018 *Bidirectional labeling in column-generation algorithms for pickup-and-delivery problems. European Journal of Operational Research* 266(2):521–530.

Gu J, Goetschalckx M, McGinnis LF, 2007 *Research on warehouse operation: A comprehensive review. European Journal of Operational Research* 177(1):1–21.

Gu Z, Nemhauser GL, Savelsbergh MW, 1999 *Lifted cover inequalities for 0-1 integer programs: Complexity. INFORMS Journal on Computing* 11(1):117–123.

Gundegjerde C, Halvorsen IB, Halvorsen-Weare EE, Hvattum LM, Nonås LM, 2015 *A stochastic fleet size and mix model for maintenance operations at offshore wind farms. Transportation Research Part C: Emerging Technologies* 52:74–92.

Gutierrez-Alcoba A, Ortega G, Hendrix EM, Halvorsen-Weare EE, Haugland D, 2017 *A model for optimal fleet composition of vessels for offshore wind farm maintenance. Procedia Computer Science* 108:1512–1521.

Halvorsen-Weare EE, Gundegjerde C, Halvorsen IB, Hvattum LM, Nonås LM, 2013 *Vessel fleet analysis for maintenance operations at offshore wind farms. Energy Procedia* 35:167–176.

Halvorsen-Weare EE, Norstad I, Stålhane M, Nonås LM, 2017 *A metaheuristic solution method for optimizing vessel fleet size and mix for maintenance operations at offshore wind farms under uncertainty. Energy Procedia* 137:531–538.

Hanasusanto GA, Kuhn D, Wiesemann W, 2015 *K-adaptability in two-stage robust binary programming. Operations Research* 63(4):877–891.

Henn S, 2015 *Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. Flexible Services and Manufacturing Journal* 27(1):86–114.

Henn S, Schmid V, 2013 *Metaheuristics for order batching and sequencing in manual order picking systems. Computers & Industrial Engineering* 66(2):338–351.

Hernández-Pérez H, Rodríguez-Martín I, Salazar-González JJ, 2009 *A hybrid grasp/vnd heuristic for the one-commodity pickup-and-delivery traveling salesman problem. Computers & Operations Research* 36(5):1639–1645.

Hernández-Pérez H, Rodríguez-Martín I, Salazar-González JJ, 2016 *A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem. European Journal of Operational Research* 251(1):44–52.

Hernández-Pérez H, Salazar-González JJ, 2004 *A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. Discrete Applied Mathematics* 145(1):126–139.

Hernández-Pérez H, Salazar-González JJ, 2004 *Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. Transportation Science* 38(2):245–255.

Hernández-Pérez H, Salazar-González JJ, 2014 *The multi-commodity pickup-and-delivery traveling salesman problem. Networks* 63(1):46–59.

Ho YC, Su TS, Shi ZB, 2008 *Order-batching methods for an order-picking warehouse with two cross aisles. Computers and Industrial Engineering* 55(2):321–347.

Hofmann M, 2011 *A review of decision support models for offshore wind farms with an emphasis on operation and maintenance strategies. Wind Engineering* 35(1):1–15.

Hong S, Johnson AL, Peters Ba, 2012 *Large-scale order batching in parallel-aisle picking systems. IIE Transactions* 44(2):88–106.

Irawan CA, Ouelhadj D, Jones D, Stålhane M, Sperstad IB, 2017 *Optimisation of maintenance routing and scheduling for offshore wind farms. European Journal of Operational Research* 256(1):76–89.

Irnich S, Desaulniers G, 2005 *Shortest path problems with resource constraints.* Desaulniers G, Desrosiers J, Solomon MM, eds., *Column Generation*, 33–65 (Springer).

Jepsen M, Petersen B, Spoorendonk S, Pisinger D, 2008 *Subset-row inequalities applied to the vehicle-routing problem with time windows. Operations Research* 56(2):497–511.

Kerkhove LP, Vanhoucke M, 2017 *Optimised scheduling for weather sensitive offshore construction projects. Omega* 66:58–78.

Kirkpatrick S, Gelatt CD, Vecchi MP, 1983 *Optimization by simulated annealing. Science* 220(4598):671–680.

Kleywegt AJ, Shapiro A, Homem-de Mello T, 2002 *The sample average approximation method for stochastic discrete optimization. SIAM Journal on Optimization* 12(2):479–502.

Leuschner R, Rogers DS, Charvet FF, 2013 *A meta-analysis of supply chain integration and firm performance. Journal of Supply Chain Management* 49(2):34–57.

Lidén T, 2015 *Railway infrastructure maintenance-a survey of planning problems and conducted research. Transportation Research Procedia* 10:574–583.

López-Santana E, Akhavan-Tabatabaei R, Dieulle L, Labadie N, Medaglia AL, 2016 *On the combined maintenance and routing optimization problem. Reliability Engineering & System Safety* 145:199–214.

Lourenço HR, Martin OC, Stützle T, 2003 *Iterated local search. Handbook of Metaheuristics*, 320–353 (Springer).

Lozano L, Duque D, Medaglia AL, 2015 *An exact algorithm for the elementary shortest path problem with resource constraints. Transportation Science* 50(1):348–357.

Maher SJ, 2015 *Solving the integrated airline recovery problem using column-and-row generation. Transportation Science* 50(1):216–239.

Mancini S, 2016 *A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. Transportation Research Part C: Emerging Technologies* 70:100–112.

Marchet G, Melacini M, Perotti S, 2015 *Investigating order picking system adoption: a case-study-based approach. International Journal of Logistics Research and Applications* 18(1):82–98.

Menéndez B, Bustillo M, Pardo EG, Duarte A, 2017 *General variable neighborhood search for the order batching and sequencing problem. European Journal of Operational Research* 263(1):82–93.

Montreuil B, 2011 *Toward a physical internet: meeting the global logistics sustainability grand challenge. Logistics Research* 3(2-3):71–87.

Morganti E, Seidel S, Blanquart C, Dablanc L, Lenz B, 2014 *The impact of e-commerce on final deliveries: alternative parcel delivery services in france and germany. Transportation Research Procedia* 4:178–190.

Mosheiov G, 1994 *The travelling salesman problem with pick-up and delivery. European Journal of Operational Research* 79(2):299–310.

Mostard J, De Koster R, Teunter R, 2005 *The distribution-free newsboy problem with resalable returns. International Journal of Production Economics* 97(3):329–342.

Muter I, Birbil Şİ, Bülbül K, 2013 *Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. Mathematical Programming* 1–36.

Naddef D, Rinaldi G, 2001 *Branch-and-cut algorithms for the capacitated VRP*. Toth P, Vigo D, eds., *The Vehicle Routing Problem*, 53–84 (Society for Industrial and Applied Mathematics).

Palhazi Cuervo D, Vanovermeire C, Sörensen K, 2016 *Determining collaborative profits in coalitions formed by two partners with varying characteristics. Transportation Research Part C: Emerging Technologies* 70:171–184.

Pan JCH, Shih PH, 2008 *Evaluation of the throughput of a multiple-picker order picking system with congestion consideration. Computers & Industrial Engineering* 55(2):379–389.

Paraskevopoulos DC, Laporte G, Repoussis PP, Tarantilis CD, 2017 *Resource constrained routing and scheduling: Review and research prospects. European Journal of Operational Research* 263(3):737–754.

Parida A, Kumar U, Galar D, Stenström C, 2015 *Performance measurement and management for maintenance: a literature review. Journal of Quality in Maintenance Engineering* 21(1):2–33.

Parikh PJ, Meller RD, 2009 *Estimating picker blocking in wide-aisle order picking systems. IIE Transactions* 41(3):232–246.

Parragh SN, Doerner KF, Hartl RF, 2008 *A survey on pickup and delivery problems. Journal für Betriebswirtschaft* 58(1):21–51.

Pillac V, Guéret C, Medaglia A, 2018 *A fast reoptimization approach for the dynamic technician routing and scheduling problem.* Amodeo L, Talbi EG, Yalaoui F, eds., *Recent Developments in Metaheuristics*, 347–367 (Springer).

Pillac V, Gueret C, Medaglia AL, 2013 *A parallel matheuristic for the technician routing and scheduling problem. Optimization Letters* 7(7):1525–1535.

Polat O, Kalayci CB, Kulak O, Günther HO, 2015 *A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit. European Journal of Operational Research* 242(2):369–382.

Post RM, Buijs P, uit het Broek MAJ, Alvarez JAL, Szirbik NB, Vis IF, 2018 *A solution approach for deriving alternative fuel station infrastructure requirements. Flexible Services and Manufacturing Journal* 30(3):592–607.

Postek K, Hertog Dd, 2016 *Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. INFORMS Journal on Computing* 28(3):553–574.

Powell WB, 2007 *Approximate Dynamic Programming: Solving the curses of dimensionality* (John Wiley & Sons).

Rahmaniani R, Crainic T, Gendreau M, Rei W, 2017 *A Benders decomposition method for two-stage stochastic network design problems.*

Raknes NT, Ødeskaug K, Stålhane M, Hvattum LM, 2017 *Scheduling of maintenance tasks and routing of a joint vessel fleet for multiple offshore wind farms. Journal of Marine Science and Engineering* 5(1):11.

Ratliff HD, Rosenthal AS, 1983 *Order Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. Operations Research* 31(3):507–521.

Röckmann C, Lagerveld S, Stavenuiter J, 2017 *Operation and Maintenance Costs of Offshore Wind Farms and Potential Multi-use Platforms in the Dutch North Sea*, 97–113 (Cham: Springer International Publishing).

Roodbergen KJ, De Koster R, 2001 *Routing methods for warehouses with multiple cross aisles. International Journal of Production Research* 39(9):1865–1883.

Roodbergen KJ, De Koster R, 2001 *Routing order pickers in a warehouse with a middle aisle. European Journal of Operational Research* 133(1):32–43.

Roodbergen KJ, Vis IF, 2009 *A survey of literature on automated storage and retrieval systems. European Journal of Operational Research* 194(2):343–362.

Ropke S, Cordeau JF, 2009 *Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science* 43(3):267–286.

Ropke S, Cordeau JF, Laporte G, 2007 *Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks* 49(4):258–272.

Ropke S, Pisinger D, 2006 *An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science* 40(4):455–472.

Rouwenhorst B, Reuter B, Stockrahm V, Van Houtum G, Mantel RJ, Zijm WHM, 2000 *Warehouse design and control: Framework and literature review. European Journal of Operational Research* 122(3):515–533.

Sampaio A, Savelsbergh M, Veelenturf L, Van Woensel T, 2019 *Crowd-based city logistics.* Faulin J, Juan AA, Grasman SE, Hirsch P, eds., *Sustainable Transportation and Smart Logistics*, 381–400 (Elsevier).

Santoso T, Ahmed S, Goetschalckx M, Shapiro A, 2005 *A stochastic programming approach for supply chain network design under uncertainty. European Journal of Operational Research* 167(1):96–115.

Sarayloo F, Crainic T, Rei W, 2018 *A Learning-based Matheuristic for Stochastic Multicommodity Network Design.*

Savelsbergh M, 1997 *A branch-and-price algorithm for the generalized assignment problem. Operations Research* 45(6):831–841.

Savelsbergh M, Van Woensel T, 2016 *50th anniversary invited articlecity logistics: Challenges and opportunities. Transportation Science* 50(2):579–590.

Savelsbergh MWP, Sol M, 1995 *The general pickup and delivery problem. Transportation Science* 29(1):17–29.

Scholz A, Schubert D, Wäscher G, 2017 *Order picking with multiple pickers and due dates–simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. European Journal of Operational Research* 263(2):461–478.

Schrotenboer AH, 2019 *Optimaal plannen van onderhoudstaken voor windmolenparken op zee. STAtOR* 21.

Schrotenboer AH, Phoa TA, van der Heide G, Kilic OA, Buijs P, 2019a *A resource-efficient freight transportation network inspired by public transport.* Submitted.

Schrotenboer AH, Savelsbergh M, 2019 *Service network design for city logistics.* Working paper.

Schrotenboer AH, Uit het Broek MA, Jargalsaikhan B, Roodbergen KJ, 2018a *Coordinating technician allocation and maintenance routing for offshore wind farms. Computers & Operations Research* 98:185–197.

Schrotenboer AH, Ursavas E, Vis IFA, 2019a *A branch-and-price-and-cut algorithm for solving resource constrained pickup and delivery problems. Transportation Science* 53(4):1001–1022.

Schrotenboer AH, Ursavas E, Vis IFA, 2019b *Mixed integer programming models for maintenance planning at offshore wind farms under uncertainty. Transportation Research Part C: Emerging Technologies* In press.

Schrotenboer AH, Ursavas E, Vis IFA, 2019c *Two-stage robust network design with temporal characteristics.* Working paper.

Schrotenboer AH, Ursavas E, Zhu SX, Wenneker R, 2018b *Robust reserve crew recovery in air transportation: Reserve-crew scheduling to mitigate risks.* Submission in preparation.

Schrotenboer AH, Wruck S, Roodbergen KJ, Veenstra M, Dijkstra AS, 2017 *Order picker routing with product returns and interaction delays. International Journal of Production Research* 55(21):6394–6406.

Schrotenboer AH, Wruck S, Vis IFA, Roodbergen KJ, 2019b *Integrating product returns and decomposition of customer orders in e-commerce warehouses.* Submitted.

Shafiee M, 2015 *Maintenance logistics organization for offshore wind energy: Current progress and future perspectives. Renewable Energy* 77:182–193.

Shafiee M, Sørensen JD, 2017 *Maintenance optimization and inspection planning of wind energy assets: Models, methods and strategies. Reliability Engineering & System Safety* In press.

Silva M, Poss M, Maculan N, 2018a *k-adaptive routing for the robust network loading problem. Electronic Notes in Discrete Mathematics* 64:95–104.

Silva M, Poss M, Maculan N, 2018b *Solving the bifurcated and nonbifurcated robust network loading problem with k-adaptive routing. Networks* 72(1):151–170.

Simchi-Levi D, Wang H, Wei Y, 2018 *Constraint generation for two-stage robust network flow problems. INFORMS Journal on Optimization* 1(1):49–70.

Smilowitz K, Nowak M, Jiang T, 2013 *Workforce management in periodic delivery operations. Transportation Science* 47(2):214–230.

Sperstad IB, Stålhane M, Dinwoodie I, Endrerud OEV, Martin R, Warner E, 2017 *Testing the robustness of optimal access vessel fleet selection for operation and maintenance of offshore wind farms. Ocean Engineering* 145:334–343.

Stålhane M, Christiansen M, Kirkeby O, Mikkelsen AJ, 2017 *Optimizing jack-up vessel strategies for maintaining offshore wind farms. Energy Procedia* 137:291–298.

Stålhane M, Halvorsen-Weare E, Nonås L, 2016 *A decision support system for vessel fleet analysis for maintenance operations at offshore wind farms.* Technical report, Sintef.

Stålhane M, Hvattum LM, Skaar V, 2015 *Optimization of routing and scheduling of vessels to perform maintenance at offshore wind farms. Energy Procedia* 80:92–99.

Stålhane M, Vefsnmo H, Halvorsen-Weare EE, Hvattum LM, Nonås LM, 2016 *Vessel fleet optimization for maintenance operations at offshore wind farms under uncertainty. Energy Procedia* 94:357–366.

Staudt FH, Alpan G, Di Mascolo M, Rodriguez CMT, 2015 *Warehouse performance measurement: a literature review. International Journal of Production Research* 53(18):5524–5544.

Stock JR, Mulki JP, 2009 *Product returns processing: an examination of practices of manufacturers, wholesalers/distributors, and retailers. Journal of Business Logistics* 30(1):33–62.

Su X, 2009 *Consumer reutrn policies and supply chain performance. Manufacturing & Service Operations Management* 11(4):595–612.

Subramanian A, Cabral LdAF, 2008 *An ils based heuristic for the vehicle routing problem with simultaneous pickup and delivery and time limit. European Conference on Evolutionary Computation in Combinatorial Optimization*, 135–146 (Springer).

Theys C, Bräysy O, Dullaert W, Raa B, 2010 *Using a TSP heuristic for routing order pickers in warehouses. European Journal of Operational Research* 200(3):755–763.

Tompkins JA, White JA, Bozer YA, Tanchoco JMA, 2010 *Facilities planning* (John Wiley & Sons).

Topteam Energie, 2012 *Topteam Energie innovatiecontract wind op zee.*

Trick M, 2005 *Formulations and reformulations in integer programming. International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, 366–379 (Springer).

Tsai CY, Liou JJH, Huang TM, 2008 *Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. International Journal of Production Research* 46(22):6533–6555.

Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A, 2017 *New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research* 257(3):845–858.

Uit het Broek MA, Teunter RH, de Jonge B, Veldman J, Van Foreest ND, 2019 *Condition-based production planning: adjusting production rates to balance output and failure risk. Manufacturing & Service Operations Management* In press.

uit het Broek MA, Veldman J, Fazi S, Greijdanus R, 2019 *Evaluating resource sharing for offshore wind farm maintenance: The case of jack-up vessels. Renewable and Sustainable Energy Reviews* 109:619–632.

Uit het Broek MAJ, Schrotenboer AH, Jargalsaikhan B, Roodbergen KJ, Coelho LC, 2019 *Valid inequalities and a branch-and-cut algorithm for asymmetric multi-depot routing problems. CIRRELT, 2019-02* Revised and Resubmitted.

Ulmer MW, Thomas BW, 2018 *Same-day delivery with heterogeneous fleets of drones and vehicles. Networks* 72(4):475–505.

United Nations, 2019 *Global e-commerce sales surged to $29 trillion.* Conference on Trade and Development.

Ursavas E, 2017 *A benders decomposition approach for solving the offshore wind farm installation planning at the north sea. European Journal of Operational Research* 258(2):703–714.

Valle CA, Beasley JE, da Cunha AS, 2017 *Optimally solving the joint order batching and picker routing problem. European Journal of Operational Research* 262(3):817–834.

Van der Heide G, Buijs P, Roodbergen K, Vis I, 2018 *Dynamic shipments of inventories in shared warehouse and transportation networks. Transportation Research Part E: Logistics and Transportation Review* 118:240–257.

Van Engeland J, Beliën J, De Boeck L, De Jaeger S, 2018 *Literature review: Strategic network optimization models in waste reverse supply chains. Omega* In press.

Van Gils T, Ramaekers K, Caris A, De Koster RB, 2018 *Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. European Journal of Operational Research* 267(1):1–15.

Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W, 2012 *A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research* 60(3):611–624.

Vidal T, Crainic TG, Gendreau M, Prins C, 2013 *A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research* 40(1):475–489.

Vis IF, Ursavas E, 2016 *Assessment approaches to logistics for offshore wind energy installation. Sustainable Energy Technologies and Assessments* 14:80–91.

Welte TM, Sperstad IB, Halvorsen-Weare EE, Netland Ø, Nonås LM, Stålhane M, 2018 *Operation and maintenance modelling. Offshore Wind Energy Technology* 269.

Willis D, Niezrecki C, Kuchma D, Hines E, Arwade S, Barthelmie R, DiPaola M, Drane P, Hansen C, Inalpolat M, et al., 2018 *Wind energy research: State-of-the-art and future research directions. Renewable Energy* 125:133–154.

Wruck S, Vis IFA, Boter J, 2013 *Time-restricted batching models and solution approaches for integrated forward and return product flow handling in warehouses. Journal of the Operational Research Society* 64(10):1505–1516.

Xu PJ, Allgor R, Graves SC, 2009 *Benefits of Reevaluating Real-Time Order Fulfilment Decisions. Manufacturing & Service Operations Management* 11(2):340–355.

Yanıkoğlu İ, Gorissen BL, Den Hertog D, 2019 *A survey of adjustable robust optimization. European Journal of Operational Research* 277(3):799–813.

Zamorano E, Stolletz R, 2017 *Branch-and-price approaches for the multiperiod technician routing and scheduling problem. European Journal of Operational Research* 257(1):55–68.

Zarrinpoor N, Fallahnezhad MS, Pishvaee MS, 2018 *The design of a reliable and robust hierarchical health service network using an accelerated benders decomposition algorithm. European Journal of Operational Research* 265(3):1013–1032.

Zeng B, Zhao L, 2013 *Solving two-stage robust optimization problems using a column-and-constraint generation method. Operations Research Letters* 41(5):457–461.

Zhang Y, Wei-Hua L, Huang M, Hu X, 2019 *Multi-warehouse package consolidation for split orders in online retailing. European Journal of Operational Research* In Press.

Zhao F, Li S, Sun J, Mei D, 2009a *Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. Computers & Industrial Engineering* 56(4):1642–1648.

Zhao FG, Sun JS, Li SJ, Liu WM, 2009b *A hybrid genetic algorithm for the traveling salesman problem with pickup and delivery. International Journal of Automation and Computing* 6(1):97–102.

Zhong Y, Zheng Z, Chou MC, Teo CP, 2017 *Resource pooling and allocation policies to deliver differentiated service. Management Science* 64(4):1555–1573.

Summary

The research field of distributed logistics concerns the efficient and effective planning and control of operations to transport goods and people between origin and destination locations. The overarching contribution of this thesis is the development of new theoretical methodology, consisting of both exact and heuristic methods, to solve new optimization problems in distributed logistics. We applied our new methods in two specific subfields of distributed logistics: e-commerce logistics and offshore wind maintenance logistics. In particular, we solve practically inspired routing and network design problems for which no solutions were yet available, and we identify the key insights that should be considered for future decision making.

In the first part of this thesis, we start by studying the short-term maintenance planning problem for offshore wind farms. Here, maintenance tasks need to be planned on a particular day, technicians should be assigned to those tasks, and the right vessels and spare parts should be available. We present multiple approaches to solve this problem. First, we study the properties of the underlying mathematical formulation leading to an exact, branch-and-price-and-cut algorithm. The algorithm's computational performance is driven by new valid inequalities, a tailored method for generating vessel routes, and the interplay between both. Second, we extend our view to multiple wind farms and the impact of sharing technicians on the quality of short-term maintenance plannings. In this case, we develop an adaptive large neighborhood search heuristic. We show that smartly coordinating the technicians reduces both the number of vessel trips to the wind farms and the average time to complete all maintenance tasks. The heuristic provides high-quality solutions in short computation times, which is convenient for practitioners.

We then zoom out from short-term planning to tactical decision making, where we study the problem of optimally allocating a fleet to multiple wind farms while considering the uncertainty of weather conditions and daily maintenance activities. We model the problem as a two-stage stochastic program and solve it using Sample Average Approximation. We show that it is crucial to consider the stochastic dynamics of the

day-to-day operations, the service requirements specified by the wind farm owner, and the impact of operational modeling assumptions on computational tractability. Not properly doing so will either lead to suboptimal decisions or unnecessarily complicated optimization models. These insights increase the general understanding of the peculiarities of optimization in the offshore wind sector and form a starting point for the development of tactical decision support tools based on mathematical optimization.

In the second part of this thesis, we start with analyzing warehouse order fulfillment operations in the presence of product returns. Despite the enormous challenge to process product returns in practice, little is known in the literature on how to incorporate returns in warehouse operations. We, therefore, study this incorporation by first focusing solely on order-picker routing, the problem of finding the shortest route to pick and return a specified list of items. We develop a genetic algorithm that provides high-quality solutions in short computation times. We show that considerable cost-savings can be obtained by incorporating the restocking of returned products. Besides, we study the delay caused by order-pickers being to close in the warehouse. It is shown that this can easily be circumvented by selecting order-picker routes of only slightly larger distances.

We continue with investigating the efficiency of integrating product returns in a large-scale order-picker routing, batching, and scheduling problem. By developing a tailored parallel adaptive large neighborhood heuristic, we confirm our previous findings on the efficiency of product return integration. Besides, we show that it might be of particular interest to split up the products belonging to the same customer order, as it significantly enhances flexibility within the warehouse.

Zooming out from warehouse fulfillment operations, we investigate the added value of dynamic decision making in city-distribution networks by adopting two-stage robust integer programming methods. We introduce the concept of "time-invariant vehicle paths", in which routes are designed a-priori, but actual departure and arrival times are determined daily. We show that, particularly in high-paced city logistics operations, this flexibility is valuable and outperforms completely static decision making.

Summarizing, we made theoretical contributions to the literature by developing new methodologies to enable the solving of practically inspired optimization problems. In particular, six different optimization problems within two subfields of logistics are introduced, modeled, and appropriate solution approaches are developed. This allows us, next to our methodological contributions, to provide practical insights to help future decision making.

Samenvatting

Het onderzoeksgebied van gedistribueerde logistiek beschouwt het efficiënt en effectief plannen en regulieren van transportoperaties om goederen en mensen tussen oorsprong en bestemming te vervoeren. De overkoepelende contributie van dit proefschrift is de ontwikkeling van nieuwe theoretische methodologie, bestaande uit exacte en heuristische methodes, zodat het oplossen van nieuwe optimaliseringsproblemen in gedistribueerde logistiek mogelijk wordt. We passen onze nieuwe methodes toe in twee specifieke subvelden van gedistribueerde logistiek: E-commerce logistiek en onderhoudslogistiek van windmolenparken op zee. Wij focussen, in het bijzonder, op praktisch geïnspireerde routering en netwerk ontwerp problemen waarvoor geen oplossingen beschikbaar zijn, en identificeren de meest relevante inzichten voor toekomstige ondersteuning bij het nemen van beslissingen.

In het eerste deel van dit proefschrift bestuderen we het korte-termijn onderhoudsplanningsprobleem voor windmolenparken op zee. In dit probleem worden onderhoudstaken gepland, moeten monteurs aan de taken worden toegewezen, en moeten de juiste boten en onderdelen beschikbaar zijn. Eerst bekijken we de eigenschappen van de onderliggende wiskundige formulering, wat leidt tot een exact branch-and-price-and-cut algoritme. De computationele prestatie komt voornamelijk door nieuwe ongelijkheden voor de wiskundige formulering, een gespecialiseerde methode voor het genereren van nieuwe routes, en de interactie tussen beide. Vervolgens verbreden we onze blik naar meerdere windmolenparken, en bestuderen de impact van het delen van monteurs op de efficiency van de dagelijkse onderhoudsplanning. Om dit te doen, ontwikkelen we een adaptive large neighborhood search heuristiek, en we tonen aan dat door het slim coördineren van monteurs het aantal dagelijkse bezoeken van boten aan het windmolenpark daalt terwijl de gemiddelde tijd totdat een turbine wordt gerepareerd daalt. We tonen tevens aan dat onze heuristiek oplossingen van hoge kwaliteit vindt in korte rekentijden.

Vervolgens bekijken we een tactisch planningsprobleem. In dit tactische planningsprobleem bestuderen we hoe we optimaal een verzameling boten kunnen toewijzen aan

meerdere windmolenparken, terwijl we de onzekerheid van de dagelijkse onderhoudstaken en weersomstandigheden hierin meenemen. We modelleren dit probleem als een two-stadia stochastisch optimaliseringsprobleem en we lossen het op met behulp van Sample Average Approximation. We tonen aan dat het cruciaal is om de stochastiek van de dagelijkse operaties, de servicevereisten van de windmolenpark eigenaar, en de impact van aannames op het operationeel niveau op de computationele prestatie, mee te nemen. Als dat niet op de juiste manier wordt gedaan, kan dit leiden tot suboptimale beslissingen of overgecompliceerde optimaliseringsmodellen. Deze inzichten verhogen het algemene begrip van de bijzonderheden van optimalisatie in de wind op zee sector, and vormen een begin voor de ontwikkeling van tactische beslissings-ondersteunende tools gebaseerd op wiskundige optimalisatie.

In het tweede deel van het proefschrift, beginnen we met het analyseren van orderverwerking operaties in distributiecentra met de aanwezigheid van geretourneerde producten. Ondanks de enorme uitdaging om retouren te verwerken in de praktijk, is er nog weinig bekend in de literatuur over hoe men retouren kan inbedden in reguliere orderverwerkingsoperaties. Daarom bestuderen we deze incorporatie, te beginnen met de focus op louter orderpicking, het probleem waarbij de kortste route om producten te verzamelen en retouren terug te leggen wordt gezocht. Hiervoor ontwikkelen we een genetisch algoritme die oplossingen van hoge kwaliteit vindt in korte rekentijden. We tonen aan dat kosten van het orderpicken substantieel dalen bij de incorporatie van retouren in het reguliere orderpicking proces. Bovendien bestuderen we de vertraging die wordt veroorzaakt door orderpickers die dicht bij elkaar door het distributiecentrum bewegen. We tonen aan dat dat soort vertragingen makkelijk te voorkomen zijn door het selecteren van andere routes die slechts marginaal langer zijn.

Vervolgens bekijken we een grootschalig, geïntegreerd, orderpicker routering, batching, en roosteringsprobleem. We ontwikkelen een toegespitste parallel werkende adaptive large neighborhoud search heuristiek, en we tonen daarmee aan dat het inderdaad efficiënt is om retouren in het reguliere proces te incorporeren. Bovendien tonen we aan dat het interessant kan zijn om producten die behoren bij dezelfde klantorder op te splitsen in het orderverwerkingsproces. Dit vergroot de flexibiliteit van de dagelijkse operaties significant.

Tot slot zoomen we uit van orderverwerkingsoperaties binnen distributiecentra naar het bestuderen van de toegevoegde waarde van het nemen van dynamische beslissingen binnen een netwerk van stadsdistributiecentra, door het gebruik van technieken uit robuust integer programming. We introduceren het concept van tijd-invariante voertuigpaden, waarin paden a-priori worden gecreëerd maar waarin vertrek en aankomsttijden op een dagelijkse basis worden bepaald. We tonen aan dat, voornamelijk

in stadslogistiek operaties die plaatsvinden met hoog tempo, de flexibiliteit van tijd-invariante voertuigpaden waardevol is en beter presteert dan beslissingen compleet statisch te nemen.

Samenvattende beschrijft dit proefschrift theoretische contributies op de literatuur door het ontwikkelen van nieuwe methodologie die ervoor zorgt dat praktisch geïnspireerde optimaliseringsproblemen opgelost kunnen worden. Dit hebben we gedaan voor zes verschillende optimaliseringsproblemen, die we hebben geïntroduceerd, gemotiveerd, gemodelleerd, en voor elk probleem hebben we geschikte oplossingsmethodes ontwikkeld.

Acknowledgements

To all. This thesis is the result of my work as a PhD Candidate at the Department of Operations, Faculty of Economics and Business, University of Groningen. What started completely unexpected, resulted into a very pleasant and enjoyable journey. I will remember all the interesting discussions, talks, lunches, collaborations, and other social activities with all the talented researchers, colleagues, and friends at the department.

To my Thesis advisors Iris Vis and Evrim Ursavas. With your great experience about academia, and your knowledge on topics related to my research, you made the last four years an instructive and enjoyable time. I am sure that there will be many more joint research projects ahead of us in the coming years. Thank you.

To the lecturer of the Master's course "OR analysis of complex systems" Kees Jan Roodbergen. By participating in your Master's course I became curious to doing research and getting a 'PhD'. Thank you for providing such an opportunity and for co-authoring three of the chapters of this Thesis.

To Iris and Kees Jan, thank you for providing me with the opportunitiy to continue to develop myself while pursuing research together in (at least) the next two year.

To my thesis' committee members Rommert Dekker, Martin Savelsbergh, and Ruud Teunter. Thank you for taking the time to be part of it. Your comments and feedback provided me the opportunity to improve the work even more.

To everyone involved in the project "Sustainable Service Logistics for Offshore Wind Farms". I am grateful that I was hired to work on this project (financed by NWO/TKI DINALOG), and that it provided me the opportunity to obtain a PhD degree.

To all-round OR expert, office mate, scrupulously precise, enthusiastic, soon-to-be doctor, and "dear colleague and friend" Michiel uit het Broek. Pursuing a PhD in the same project, but on different topics, did not keep us from pursuing joint research projects. This is how collaborating should be, and it will continue for many more years. Thanks for the amazing time and the great friendship, and I look forward to

the next two years.

To R specialist, FC Groningen supporter, professional FIFA player, and "dear colleague and friend" Dennis Prak. I enjoyed the numerous coffee breaks and other social activities including conferences in Lunteren, Dublin and Phoenix. Thanks for the amazing time and your always reasonable opinion, and good luck in Munich.

To my other friends at the department. Be it on conferencences, the 12 o'clock lunches, or any other social gathering, I have enjoyed it very much. Thanks for the great time Bolor, Roel, Jose, Bart, Hendrik, Aline, Babette, Jan Eise, Robert Jan, Sabine, Lisanne, Anne, Mitchel, Michiel, Dennis, Gerlach, Ward, Jelmer, Bram, and Paul.

To my friends originally outside academia. Thanks for providing some of the reality of non-academic life, and thanks for being there for me in good and in bad times. Thanks to Fester, Vivienne, Roos, Antwan, Michiel, Dennis, Patrick, Martin, Remco, Nick, and Niels.

To all current and past collaborators on research projects that started during my PhD. I have (had) the pleasure to work with many of you on a wide variety of topics. This is one of the reasons I have enjoyed working at the department (and abroad) so much. Thank you (again) Iris, Evrim , Kees Jan, Arjan, Marjolein, Susanne, Bolor, Michiel, Rob, Stuart, Paul, Onur, Gerlach, Tim, David, Martin, and Leandro.

To all current and future collaborators. I look forward to continuing working together!

Finally, to my parents. Thank you for your unconditional support in the last 28 years. Let her keep a light on in our hearts.

Groningen, December 2019.

Curriculum Vitæ

Albert Harm Schrotenboer was born on 7 July 1991 in Hoogeveen, The Netherlands. After graduating from the RSG Wolfsbos in 2009, he started studying Econometrics and Operations Research at the University of Groningen. He obtained his Bachelor's degree (220 credits) in 2014 and his Master's degree (Cum Laude - 71 credits) in 2015. His Master's thesis on order picker routing in warehouses has been published in a peer-reviewed scientific journal. He has been teaching assistant for various mathematics and statistics courses both inside and outside the university.

He continued his academic career by starting as a PhD candidate at the Department of Operations, University of Groningen, in 2015. He joined the research project "Sustainable service logistics for offshore wind farms", with Iris Vis and Evrim Ursavas as his PhD advisors. He has been award an NWO travel grant to be a visiting researcher for three months at the H. Milton Stewart School of Industrial and Systems Engineering at the Georgia Institute of Technology. His research has been internationally recognized with scientific publications in peer-reviewed journals, including Transportation Science, Computers & Operations Research, Transportation Research Part C: Emerging Technologies, and International Journal of Production Research. Many more research papers are currently in various stages of the reviewing process. His research findings are presented at scientific conferences such as Odyseus, VeRoLog, TSL Conference, EURO, and IFORS.

Albert's research interests focus on the development of exact and heuristic methods to solve practically inspired problems in the area of distributed logistics. In particular, he is experienced with various optimization paradigms for Mixed Integer Programming Models such as column generation, branch-and-cut, and branch-and-price. Currently, he is exploiting the opportunities for robust and stochastic decision making in the field of transportation by using Markov decision processes, approximate dynamic programming, and (stochastic) mixed-integer programming.