Delivery Cost Approximations for Dynamic Time Slot Pricing

Author
F.R. Akkerman

Examination Committee

University of Twente
Dr.ir. M.R.K. Mes
Dr.ir. E.A. Lalla-Ruiz

ORTEC
Dr. T.R. Visser

UNIVERSITY OF TWENTE.

ORTEC

Management Summary

This research is conducted at ORTEC, a Dutch software and consultancy company. ORTEC is the leading supplier of mathematical optimization software and advanced analytics for, amongst others, workforce planning and vehicle routing. In the past two decades, package home delivery has been a thriving business area. We focus on attended home delivery (AHD), for which it is necessary that the customer is at home during delivery. Many companies offer their customers time slots for delivery, to provide high customer service and prevent costly delivery failures. AHD is needed for security reasons, in case of perishable goods, large packages, or when services are performed. The costs for offering time slots are often double compared to standard home delivery services (Yrjölä, 2001).

We focus on dynamic pricing of time slots. To influence customers to choose a time slot that is cheaper for the company, i.e., time slots that represent less operational delivery time and travel distances, companies often give incentives (discounts) or penalties (delivery charges) dependent on the costs of a time slot. The main focus of this thesis is on determining the costs of a time slot and adjusting time slot pricing accordingly. Typically, the costs of time slots are determined using (static) methods that are not fully able to predict the costs of a time slot correctly. The prediction of costs is not straightforward, since we do not know in advance the exact costs of adding a customer to a time slot. This difficulty is caused by several factors: (i) the dynamic nature of time slotting, i.e., the decision to insert a customer in a specific time slot will decrease the capacity and therefore influence the decision for the next customer, (ii) the uncertain nature of customer arrivals, i.e., it is unknown where and how much customers will arrive and (iii) the routing of vehicles to serve customers cannot be determined before all customer orders have been booked. We use a cost approximation method that uses machine learning regression models (random forests and neural networks) and a limited set of features to predict the costs of adding a customer to a certain area-time slot combination (ATC). Also, we present a rudimentary time slot incentive policy that can be used to test our cost approximation model.

As common for time slotting research, we use discrete-event simulation to validate our results. We simulate the booking horizon for a single delivery day, i.e., customers arrive and book a time slot during a horizon of several days, but customers always choose the time slots on the same delivery day. We use several different case studies for testing our approach: instances with differing customer location patterns, constraints and sizes. Also, we test our approach on a case with actual customer data from a large European e-grocery retailer.

Our results show that time slot incentives have added value for practice. In a hypothetical situation where customers are infinitely sensitive to incentives, we can plan 6% more customers and can decrease the per-customer travel costs by 11%. Furthermore, we show that our machine learning model (random forests) works especially well when customer locations are heavily clustered or when the area of operation is sparsely populated.

For a case of a European e-grocery retailer with less flexible customers, we show that we can save approximately 6% in per-customer travel costs, and plan approximately 1% more customers when using our time slot incentive policy. Figure 1 shows an overview of the effect of using dynamic time slot prices against using no time slot pricing. A larger area indicates better

performance. We see that with the use of time slot pricing, there is more room for customers and a similar number of time slots can be offered on average. The costs, expressed in travel time, waiting time and distance per customer, are significantly lower, although it should be noted that the waiting times are relatively low compared to the travel times.

We recommend to implement dynamic time slot incentive methods to save costs. However, further research needs to be conducted in customer behavior patterns and time slot incentive algorithms. We suggest to look at (deep) reinforcement learning policies that can learn a policy using Monte Carlo simulation. Difficulties that have to be overcome can be mainly found in the training procedure of models. Also, generic reinforcement learning models will need to be able perform in a variety of different cases, which might be difficult for specific instances. The eventual cost approximation influences the incentive policy, and in turn the incentive decisions influence the cost approximation. A reinforcement learning model could be valuable for learning this explicit relationship.

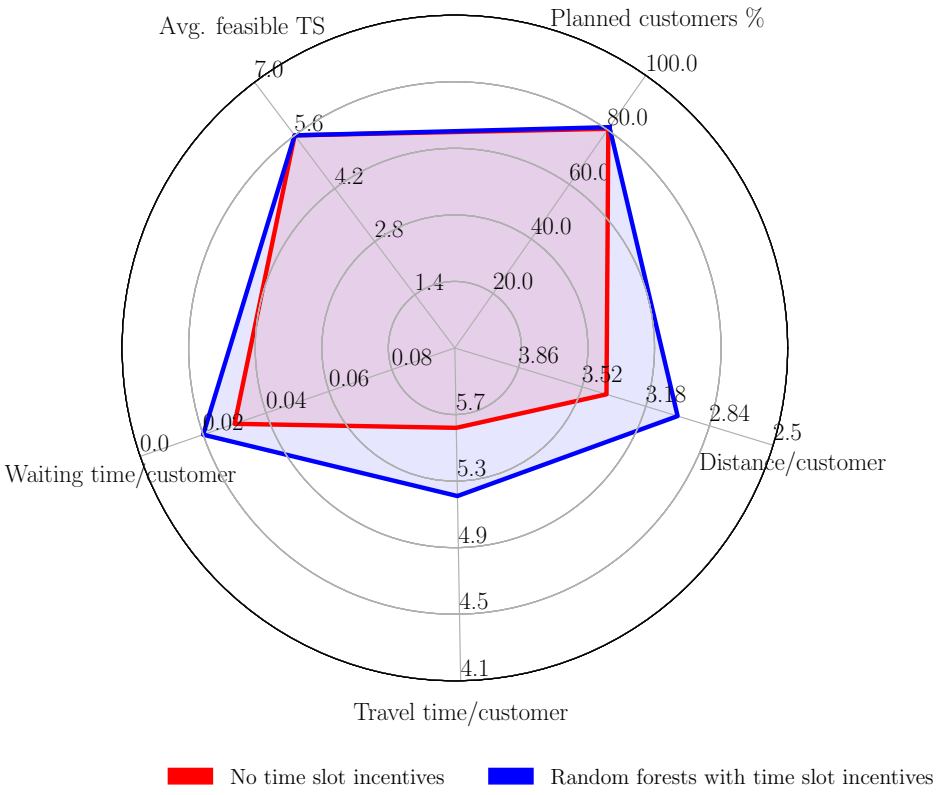

Figure 1: Radar chart for the performance of the simple incentive policy based on random forests and the situation without incentives, a larger area indicates better performance, results from a realistic case using 2 replications.

Preface

I owe a debt of gratitude to the people that made this thesis possible. I enjoyed every minute of my time at ORTEC and therefore want to thank everyone at the Math Innovation Team for making me feel part of the company. In special, I want to thank Thomas. His expertise, passion and enthusiasm for time slotting encouraged me to strive for a great thesis. I enjoyed our many talks and appreciate all the time he spend on this project. I would also like to thank Nathan for his help with the simulation tool, some of the plots in this thesis, and the insights he provided into the time slotting experiments he and Thomas did. Finally, I would like to thank Leendert for giving me the opportunity to do my thesis at ORTEC.

During the thesis I was advised by my supervisors from the University of Twente, Martijn and Eduardo. Their feedback and good ideas improved the quality and lead to more interesting insights. Also, I would like to thank them both for all the opportunities and the trust they put in me the last years when I was TA at their courses or other projects. I know for sure that I would not have learned as much without them.

Fabian Akkerman, April 2021.

Contents

Abbreviations

ADP	Approximate Dynamic Programming	*A solution technique for solving complex, discrete-time, multistage stochastic control processes.*
AHD	Attended Home Delivery	*A delivery type for which the customer needs to be home to receive the goods or service.*
ATC	Area-Time slot combination	*An aggregation structure that groups customers by their location and chosen time slot.*
MNL	Multinomial logit choice model	*A parametric choice model based on utility theory.*
OT	ORTEC Technology	*The software development department of ORTEC.*
ORS	ORTEC Route Optimization Service	*An ORTEC cloud product used for vehicle routing.*
OTSS	ORTEC Time Slotting Service	*An ORTEC cloud product used for time slotting.*
VRP	Vehicle Routing Problem	*A combinatorial optimization problem for finding the best routes for a fleet of vehicles to deliver to a set of customers.*
VRPTW	Vehicle Routing Problem with Time Windows	*A variant of the VRP that includes time windows for deliveries.*

Chapter 1

Introduction

First, the company at which the research is conducted is introduced in Section 1.1. Second, the research topic is introduced in Section 1.2, next the problem statement and motivation from a business perspective are explained in Section 1.3, and the research objective and research questions are stated in Section 1.4. We discuss the ethical considerations for this research in Section 1.5. Finally, the research approach and structure of this thesis are outlined in Section 1.6

1.1 About ORTEC

This research is conducted at ORTEC, which is a Dutch software and consultancy company operating around the world. ORTEC is the leading supplier of mathematical optimization software and advanced analytics. The company has over a 1000 employees and is based in 13 countries. The main fields in which ORTEC is active are marketing, sales, planning & forecasting, warehousing, asset management, workforce scheduling, strategy and routing & loading (ORTEC, 2020a).

The graduation project takes place at the Math Innovation Team of ORTEC, which is part of the ORTEC Technology (OT) business unit. OT is the software development department of ORTEC. The software solutions developed at OT can be divided into two groups: the goods supply chain and the service supply chain. The goods supply chain contains software solutions for, amongst others, vehicle routing, supply chain design and service planning. The service supply chain contains products for workforce planning and scheduling. Aside from these two product groups, OT also develops tailor made solutions for specific customers (ORTEC, 2020b).

The OT department is assembled by several units or "squads", which often have a single task or goal. The Math Innovation Team has as goal to reduce the time-to-market, i.e., the time it takes from developing a product until it becomes available for commercial use, of mathematical innovations with measurable customer impact. To reach their goal, the Math Innovation Team explores new innovations, validates innovations for small customer cases, proves applicability by doing larger projects at multiple customers, and builds products together with other OT-teams (ORTEC Math Innovation Team, 2020).

Most current projects of the Math Innovation Team are focused on vehicle routing software. ORTEC has vehicle routing solutions for many kinds of tactical and operational planning problems and for different types of businesses. This research will revolve around the ORTEC Time Slotting Service (OTSS) and the ORTEC Route Optimization Service (ORS), which will be further explained in the next sections.

1.2 Attended Home Delivery and Time Slotting

During the last two decades, many e-commerce initiatives have driven the demand for package delivery services. Variations of business-to-consumer (B2C) business models have sparked op-

portunities for marketing and customer service. One of the ultimate value-adding services is last-mile delivery, the delivery of packages to the customer's front door (Campbell and Savelsbergh, 2005). Home delivery services present big challenges for retailers, service providers, and logistics companies. Logistics needs to be organized in a way that is efficient, profitable, and satisfies the customers' wishes, while dealing with stochastic customer arrivals.

For this research, we focus on attended home delivery (AHD), for which it is necessary that the customer is at home at the delivery moment. AHD might be needed for security reasons (e.g., high value goods), perishable goods (e.g., groceries), physically large goods (e.g., home appliances), or because services are performed (e.g., product installation) (Agatz et al., 2008). Many companies that use AHD services provide their customers with time slots for choosing the delivery moment. Delivery time slots are offered to provide a high customer service and prevent costly delivery failure. When delivery has failed, the goods have to be offered for delivery on a different moment, which will result in additional storage, transportation and planning costs. In case of perishable goods, the costs of a delivery failure are even higher, since the goods may be spoiled before the next delivery opportunity. Early studies showed that the costs of AHD are often double the cost of unattended delivery (Yrjölä, 2001).

We focus on the process of offering time slots to customers. The ordering process, from a customer viewpoint, is shown in Figure 1.1.

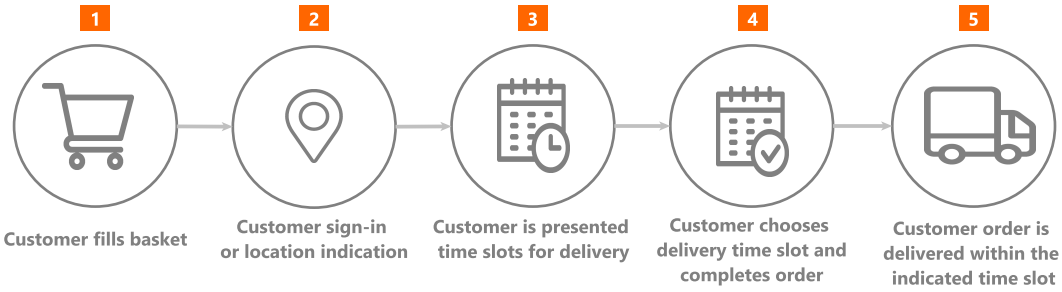

Figure 1.1: Customer ordering process (customer perspective).

First, the customer selects products or services in the online environment of the retailer. Some retailers may have a minimum ordering quantity for home delivery. When the customer is finished selecting products or services, he or she needs to indicate the delivery location. This may be done by first logging in on the personal customer account, or alternatively by only indicating the customer address or postal code. Some retailers require the customer to first sign in, before the customer can start selecting products. In that case step 1 and step 2 in Figure 1.1 are swapped. In step 3, the grocery retailer offers delivery time slots to the customer, possibly depending on the customer location. The possibilities for the design of the time slot offering will be discussed in Chapter 3. After the customer has selected a delivery time slot and has completed the order (step 4), the order is confirmed and delivered in step 5. In the past, some retailers offered to first select a time slot before starting the ordering process, so time slotting decisions had to be made without customer information (e.g., location and order quantity). Since this limits the possibilities for solution methods, the selection of time slots before ordering is neglected for this study.

Figure 1.2 shows how a time slots could be presented to the customer. In this example, a customer can choose a delivery moment on four days. The time slot on Wednesday is wide and has an "eco" symbol, to indicate it is an environmentally friendly option, since wider time slots allow more efficient routing. Notice that the time slots on Thursday are narrower and overlap. Friday's time slots are full, so no new customer booking can be done.

The time slots have different prices, as part of the company's pricing policy. Time slot pricing policies are intended to steer customer behavior towards time slots that are better for the company, i.e., these time slots represent lower transportation costs. By using incentives or

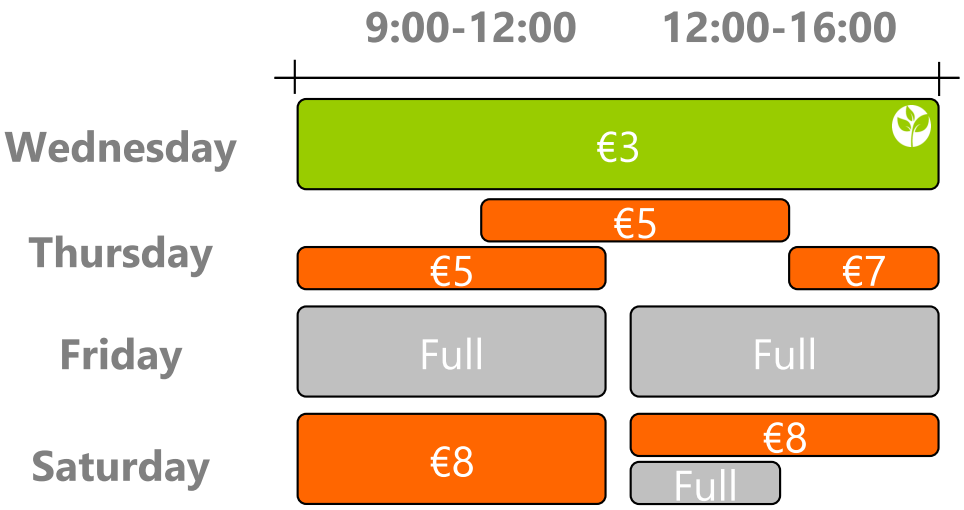

Figure 1.2: Example of a time slot customer interface.

penalties, a company can influence customer behavior in choosing a time slot. By influencing customer behavior, it is possible to reduce operational costs. The reduction of costs can be done by, e.g., smoothing the demand patterns or the geographical spread of customers over time to reduce demand peaks (Agatz et al., 2008), reducing vehicle routing distance or time, and reducing the required fleet size.

Although much research has been conducted on time slot allocation, i.e., the offering of only a subset of the feasible time slots, this study will consider the situation in which always all feasible time slots are offered, since this is common practice for all ORTEC's customers.

1.3 Problem Statement and Motivation

It is challenging to make effective time slot offering decisions. First, the software solution needs to use the currently available information, like the current schedule and remaining capacity, and the new incoming information, like the new customer's order size and location, to check what time slots are still available for offering. Second, the software needs to calculate the costs of inserting a new customer into a route, to obtain the cheapest options. Third, the software solution needs to record, store and update all current information after the customer made a choice for a time slot. Fourth, the time slot bookings need to be converted to vehicle routing schedules. The problem is further complicated by uncertain customer arrivals and customer behaviour. Especially since the vehicle routing problem (VRP) is an NP-hard combinatorial optimization problem, meaning that realistic instances can only be solved using heuristics (Cordeau et al., 2007). Since the VRP is NP-hard, and customers demand fast working online services, the time slots need to be offered quickly and cannot be determined using exact approaches. Recent research suggests that each 100-millisecond delay in the load time of websites can decrease sales conversion by 7% (Akamai, 2017).

Although there is no time to do many calculations before offering a time slot, it is crucial that the time slot offering methods takes the cost factors into account, consisting of, e.g., fuel, salary, vehicle rent and emissions. In addition, the opportunity costs can be considered, which are the cost of offering a time slot now compared to saving time slots for later, for potentially more profitable customers (Yang et al., 2016). A method that does not consider cost factors may yield longer vehicle routes, peak demand at popular times, and many idle vehicles at unpopular times (Yang et al., 2016).

ORTEC wants to research possibilities for improving their current time slotting service,

which is further explained in Chapter 2. Home delivery is an industry with small profit margins, but with a huge growth potential. Therefore, ORTEC wants to develop their home delivery services by, amongst others, providing dynamic time slotting solutions for retailers (ORTEC, 2020a). The current time slotting service of ORTEC uses an insertion cost calculation for determining the cost of adding a new customer to a certain time slot. These insertion cost are based on the additional distance, time, or setup costs paid for adding a new visiting location to existing or new routes. These insertion costs can be inaccurate and vary between under -or overestimating the actual costs of inserting a customer. It should be noted that there is a significant difference between time slot bookings and vehicle routes. When a customer is added to a certain time slot, it means that an additional location is added to the Vehicle Routing Problem with Time Windows (VRPTW), further explained in Chapter 3. It is possible that one vehicle route in the VRPTW-solution serves customers from different time slots. Also it is possible that customers in the same time slot are served by different vehicles (Agatz et al., 2008). The difference between time slot bookings and the VRPTW solution is caused by constraints on vehicle capacity and time windows. This depends on several factors, e.g., order size, customer location, vehicle capacity, time slot design, and the current routing schedule. Hence, the costs of inserting a customer into a time slot also depends on these factors.

Although ORTEC currently does not provide a service for determining the best pricing for time slots, most retailers that use ORTEC services do have some price differentiation between time slots. This price differentiation can be based on static rules as implemented by the retailers themselves, or it may be dynamically based on the insertion costs per time slot. However, if the cost approximation is inaccurate, wrong pricing decisions might be made, resulting in higher operational costs. In practice, time slot prices are used to stimulate customer behavior, but are never fully cost-effective, i.e., the costs of transportation are never covered by the fee that customers pay. The time slot pricing methodology is a relevant topic for ORTEC since it can be an opportunity to deliver better service to their retailing customers and help them to save costs.

Summarizing, the core problem, the difference between norm and reality, can be stated as follows. Currently, the estimated costs of visiting a customer in a certain time slot are based on relatively simple methods returning insertion costs that are not reflecting the actual situation. Because of this, the operational decisions are made based on unreliable information, which results in higher transportation costs. Also, time slot pricing policies are not considered by ORTEC, which means that possible opportunities for lowering transportation costs are neglected.

1.4 Research Objective and Research Questions

The research objective can be stated as follows:

> *"Develop and validate an automatic method that approximates the costs of inserting a customer into a time slot. Furthermore, to explore the potential of methods that direct customers using a pricing policy, utilize approximations to identify time slots that yield lower transportation costs for the company. The method should consider routing costs, practical requirements and constraints. The method should be able to dynamically adapt to the situation and should be fast enough."*

Although data from current ORTEC customers will be used to validate the proposed method, it will be required for the model to be applicable to a broad range of cases. The solution method needs to be flexible to cope with different time slotting designs and pricing policies. The method needs to be fast in an online setting and should estimate the costs of offering a certain time slot as accurate as possible. To achieve this objective, several research questions and sub-questions need to be answered, as stated below.

First, we need to find out how the current time slotting implementation of ORTEC works. This is needed to understand the practical demands for a solution method and to use the current situation as a benchmark.

1. How does the current ORTEC time slotting solution work?
2. What are the practical considerations and constraints for a new time slotting method and pricing policy?

Next, we study literature to find out the current standing of scientific research in vehicle routing, attended home delivery, time slotting and dynamic pricing. Additionally, we need to find out how delivery costs and revenues can be estimated, how customer behavior is modelled, and what methods can be used to solve the dynamic pricing problem.

3. How is time slotting and dynamic pricing jointly treated in literature?
    (a) What is the problem structure of vehicle routing problems with time windows?
    (b) What solution methods are applied to time slot allocation and pricing for attended home delivery in literature?
    (c) How is customer behavior on time slotting decisions modelled in literature?
    (d) How can transportation costs be estimated?

Next, we need to consider what method can best be applied to ORTEC's situation.

4. What cost approximation and dynamic pricing method can be applied to our case study?
    (a) What steps need to be taken in the method?
    (b) How can the method fulfill the practical requirements?

We need to validate our results. We need to find out how we can validate our model, and finally we need to show the performance of our model.

5. How can our cost approximation and dynamic pricing models be tested?
    (a) How can our cost approximation models be validated?
    (b) How can the customer behaviour be modelled?
    (c) How can available data be used in a simulation model?
    (d) How can our dynamic pricing model be compared to the current situation?
6. How well does our model perform compared to the current situation in different experimental settings?

Finally, we need to do recommendations to ORTEC and give the theoretical and practical insights and implications of our research.

7. What are the practical and theoretical implications of our research?
    (a) What is the contribution of our research to theory?
    (b) What are the practical implications of our research?
    (c) What can be recommended to ORTEC on the basis of our research?

1.5 Ethical Considerations

Influencing individuals' choices to reduce business costs might be perceived as unethical. The use of automated methods to influence behaviour is a sensitive topic in society. When automated methods make their decision based on customer information, e.g., their place of residence and order value, discriminating business decisions might be made. First, we want to stress that we do not intend to conduct unethical research or to support unethical behaviour. We are aware that it might be perceived unfair to base customer offerings and time slot prices on customers' place of residence or order value. Nevertheless, the place of residence has major influence on transportation costs and therefore it could be considered fair to base prices on delivery addresses. Regardless ethical concerns, we believe that our research is beneficial for business as well as consumers, since savings in transportation costs can result in lower prices for customers, and faster service. Second, we do not study customer behaviour directly, since we focus on the business decisions that can help to decrease costs, improve customer service and decrease environmental impact. We study how time slot offering and pricing can potentially decrease costs, but we do not suggest in any way that all methods we test are directly applicable to practice. Finally, we want to indicate that some methods we test might be rejected by business because of ethical concerns, but this does not mean that our research is pointless: the analysis and methods might give more insight and can help to benchmark and develop alternative methods.

1.6 Research Outline

To solve the research problem and reach the research objective, the research questions as stated in the Section 1.4 need to be answered. We outline our research methodology and the structure of the remainder of this thesis.

In Chapter 2, we answer the questions related to the current context and practical requirements for our solution method. In this chapter the currently in place software solutions of ORTEC are explained, as well as the underlying business processes. Also, the general setting of our case study about e-grocery retailing is introduced. Information about the research context is obtained using informal interviews. The research questions related to current literature are answered in Chapter 3. We discuss the vehicle routing problem with time windows (VRPTW), and discuss the state-of-the-art time slotting literature. Next, methods for approximating transportation costs are discussed. Also, our contributions to scientific literature are outlined at the end of this chapter. In Chapter 4, we give a formal problem formulation and outline our method for modelling customer behavior. In Chapter 5, the research questions related to our solution approach are answered. We present an approach to approximate delivery costs per time slot using several regression methods. Also, we propose a simple incentive policy that can be used to test our time slot cost approximation. The set of research questions dealing with solution validation using our simulation model are answered in Chapter 6. Also, the experimental design is outlined in this chapter. The final research questions, dealing with the performance of our proposed method, are answered in Chapter 7. In this chapter the experimental results are stated and explained. Our research methodology entails experiments with realistic instances that are based on several different case study scenarios, based on e-grocery retailer data. We obtain and validate our experimental results using discrete-event simulation. Finally, in Chapter 8, we discuss the results, draw conclusions from our research and make final recommendations. Also, we state the limitations and contributions of our research and discuss future research directions.

Chapter 2

Research Context

In this chapter, the research context is given. In Section 2.1 the business processes of ORTEC's customers using time slotting services are outlined. Next, in Section 2.2, the software solutions currently in use by ORTEC are described and in Section 2.3, the case studies used for this research are explained. Finally, in Section 2.4, the research questions related to this chapter are answered in a summarizing conclusion.

2.1 Time Slotting Business Processes

The business processes underlying the customer order process may differ per retailer. Figure 2.1 depicts the customer order process as common for the retailers using the ORTEC Time Slotting Service (OTSS). The figure shows the timeline for both the customer perspective and business process perspective. Not all possible underlying business processes are depicted, the focus is on ORTEC software that supports the various decisions during the customer order and delivery process. See Section 1.2 and Figure 1.1 for a full explanation of the ordering process from a customer perspective. Summarizing, the customer order process begins with the customer filling the online basket. After finishing shopping, the customer logs in to his/her account or fills in the postal code of the delivery address. The customer is offered several time slots for delivery. Next, the customer chooses a time slot and finishes the ordering process. Finally the order is delivered at the home address of the customer within the selected time slot.

The two depicted software solutions that aid the decision making process are the ORTEC Time Slotting Service (OTSS) and the ORTEC Route Optimization Service (ORS). As soon as the customer indicated the delivery location (step 2), the OTSS will receive a request for a time slot offer. Using the customer information (location and basket content) and the current state of the schedule (currently planned customers and remaining vehicle capacities) the OTSS will offer the available time slots to the customer. Time slot availability is based on feasibility of the solution concerning vehicle capacity and time window constraints. Aside from the feasibility calculation, the OTSS makes a quick calculation of the costs of inserting the new customer in all temporal routes. These temporal routes are called a "state" in the figure. OTSS constructs temporal routes to approximate the transportation costs and check feasibility. These routes are constructed with a simple and fast algorithm, and are therefore far from optimal. The routes that are constructed by OTSS contain only the customers that booked a time slot up to the moment of the optimization requests, i.e., do not contain all customers that need to be delivered on the delivery day. After the cutoff time, which is the time after which no more new customer arrivals can come in for a delivery day, the final routes are constructed with a more advanced routing algorithm that is closer to the optimum.

After the customer choice (step 4), the OTSS will update the current state with the new customer. OTSS uses a cheapest insertion algorithm (described in Section 2.2), which ensures fast runtimes. To get more reliable results from the cheapest insertion algorithm, the current

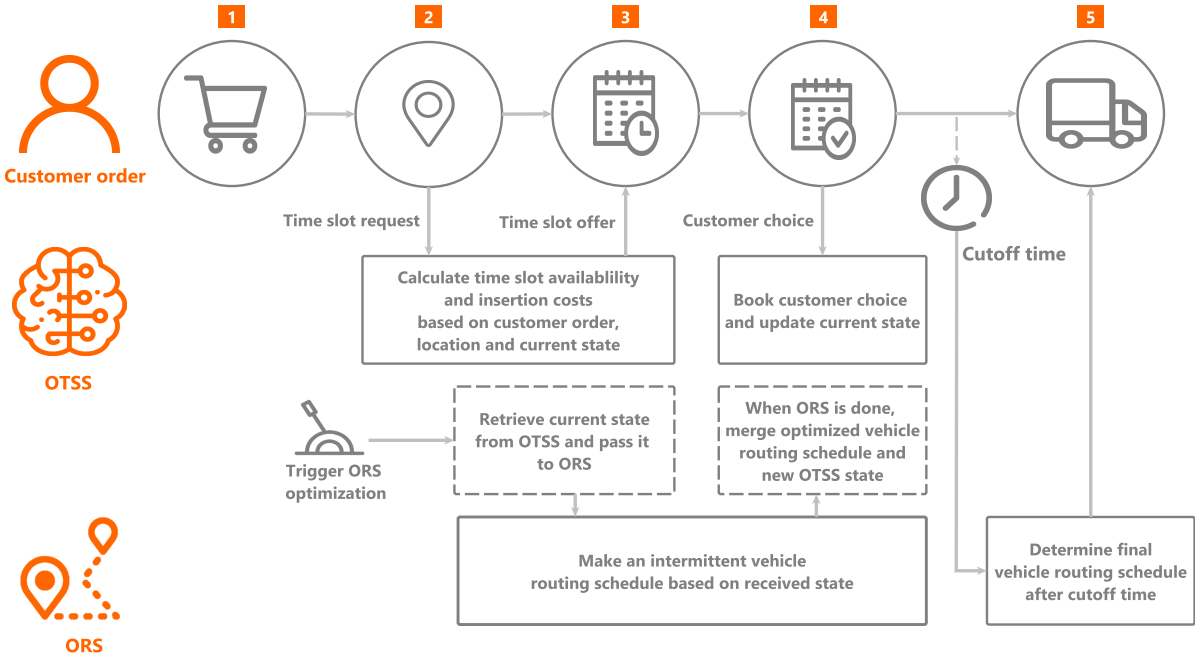

Figure 2.1: Customer ordering process (business process perspective).

state needs to be optimized and updated regularly to better reflect the actual routing costs. Therefore, during the day, the current (not final) vehicle routing schedule is optimized and updated using a request to ORS for obtaining an intermittent vehicle routing schedule. These requests can be triggered by time (e.g., optimize the schedule every 15 minutes), customer arrivals (e.g., optimize the schedule every 20 incoming customers) or based on some other indicator (e.g., vehicle capacity related). This optimization request to ORS is done to improve the schedule and to prevent the time slots to reach their capacity limit too early, resulting in customer rejection (see Section 2.2 for a full explanation). After the cutoff time ORS is ran one last time to make the final vehicle routing schedule.

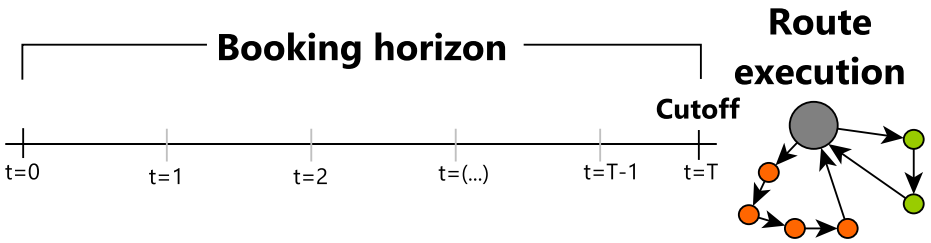

Figure 2.2: Time slot booking and execution horizon.

In practice, customers can choose between several delivery days and between different time slots on these days. In most time slotting literature, however, only a single delivery day is considered. That means, it is is assumed that the customer decision of choosing a time slot is independent from time slots on alternative delivery days (Yang et al., 2016). Often this assumption is needed to calibrate solution methods. In Figure 2.2, the horizon of a time slot booking process is depicted. At $t = 0$ the first customers can book a time slot for a single delivery day after the booking horizon. Customers can book a time slot until the cutoff time at time T. After the cutoff time, a final routing schedule is constructed and executed. To limit the scope of this research and decrease computational efforts, we also only consider a single delivery day.

2.2 ORTEC's Current Software Solutions

In this section, we discuss the workings of the two main software solutions used for time slotting: OTSS and ORS. We start with explaining the main service, namely the time slotter (OTSS). Next, we explain the general working of ORS.

ORTEC Time Slotting Service

OTSS is called when time slots needs to be offered or when a choice for a time slot has been made and needs to be booked. When a time slot offer is requested, OTSS first calculates which current vehicle routes are feasible for insertion. An insertion into an existing route is feasible when (i) the vehicle capacity is sufficient to add the respective customer order, and (ii) the customer, and all already booked customers, can be served within their indicated time windows. For the capacity calculation, both order volume and order weight can be considered.

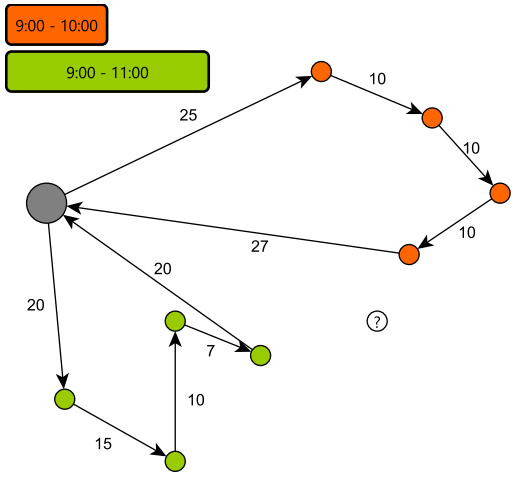

Figure 2.3a: Example of cheapest insertion with time slots: initial situation.

Figure 2.3a, Figure 2.3b, and Figure 2.3c show a simplified example of a time slotting instance and two possible new insertions. Figure 2.3a shows two current vehicle routes. The orange route serves customers that have chosen a time window from 9:00 to 10:00, the green route serves customers that have chosen a wider time slot, from 9:00 to 11:00. The numbers indicated at the arrows are the travelling times between locations in minutes. The grey node is the central depot, the starting and ending point of the vehicle routes. We assume that both vehicles start their route at 9:00 and incur 1 minute of delivery time at each customer. In Figure 2.3a we see that a new customer has arrived. In this case, OTSS first needs to consider which time slots are feasible to offer to this customer. For this example we assume that both vehicles still have enough capacity to add the customer order.

First, as shown in Figure 2.3b, the insertion of the customer into the orange route is considered. We see that the current final customer in the orange route is served at 9:59, (55 minutes travel time, 4 minutes delivery time). The addition of the new customer would mean that this customer would be served at 10:04 (4 minutes additional travel time, 1 minute service time). Therefore the insertion into the orange route can only be feasible if the route would be altered such that the new customer is served before 10:00, which might result in a delivery outside the time slot for an already booked customer.

The second option, as shown in Figure 2.3c, is inserting the new customer in the green route. The current final customer of the green route is served at 9:56 (52 minutes travel time, 4 minutes delivery time). The new customer insertion would mean that the customer can be served at 10:07 (10 minutes additional travel time, 1 minute additional delivery time), meaning that the green time slot is feasible. Cheapest insertion returns all feasible time slots, in this

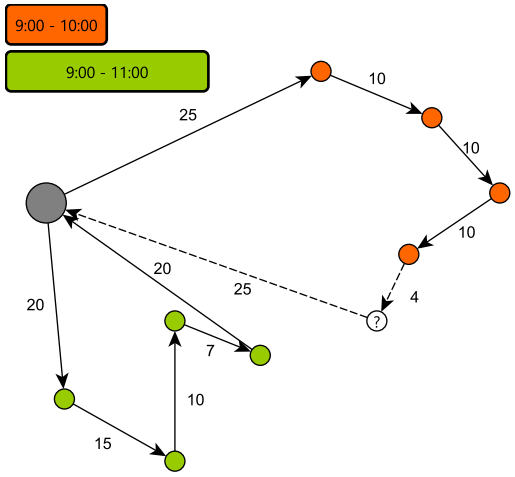

Figure 2.3b: Example of cheapest insertion with time slots: first insertion option.

case only the green time slot, and the insertion costs, i.e., the increase in distance, time and fixed costs. For the green route this means that the total increase in time by adding the new customer is 15 minutes (Δ15 minutes). When a new route is started for the customer, fixed costs are also returned by OTSS, these are the "startup costs" for a new route.

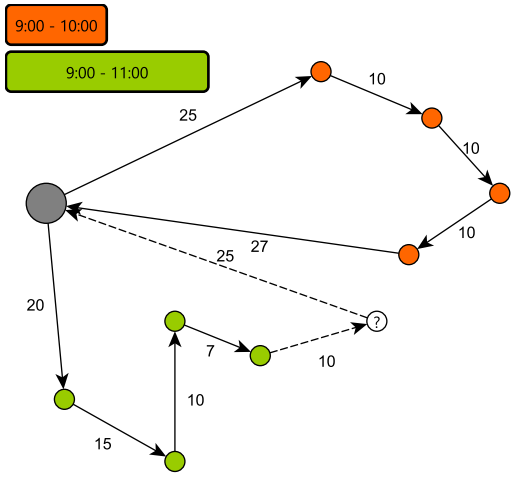

Figure 2.3c: Example of cheapest insertion with time slots: second insertion option.

For illustrative purposes, we only consider adding the new customer at the end of the vehicle routes. In reality, cheapest insertion can insert customers in any place in the route, also between two existing customers. Cheapest insertion has to consider many routes and has to decide where the new customer can be inserted into a respective route based on the cheapest insertion place, i.e., the feasibility check is a complicated process. It should be noted that the information needed for the insertion calculation, e.g., the distance and time addition, is also done online by calling the maps service used by ORTEC.

After the feasibility and insertion costs calculation, the customer can be offered a set of available time slots. As soon as the customer confirms a time slot for delivery, this choice is booked in OTSS and the vehicle routing schedule is updated with the new insertion.

Cheapest insertion is a simple, fast and intuitive algorithm that always makes greedy decisions. This means that the algorithm makes the best choice at that point in time, i.e., inserting a customer in an existing route in such a way that the distance and time is increased the least as possible. A disadvantage of greedy algorithms is that they often do not produce a close-to-optimal solution. A simplified example of a greedy VRP-solution is shown in Figure 2.4a. The

figure shows two vehicle routes, starting at two different depots. One vehicle can at most serve four customers. Therefore, when a new customer comes in (node 5), it can only be served by the green vehicle, since the orange vehicle has already reached its capacity.

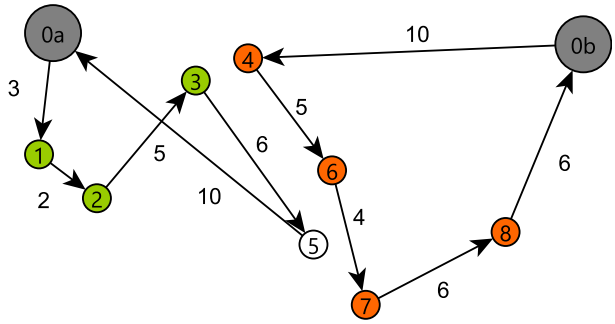

Figure 2.4a: Example of a greedy cheapest insertion VRP-solution.

It is easy to see that a better solution exists than currently displayed, as shown in Figure 2.4b. When node 4 is removed from the orange route and inserted in the green route, capacity becomes available to insert node 5 into the orange route, yielding a more efficient routing schedule. However, the greedy cheapest insertion only concerns the addition of the new customer (node 5) into the current schedule, not the re-optimization as can be obtained from ORS. For this example, cheapest insertion, as shown in Figure 2.4a, costs 18 distance units more than the alternative routing as shown in Figure 2.4b.

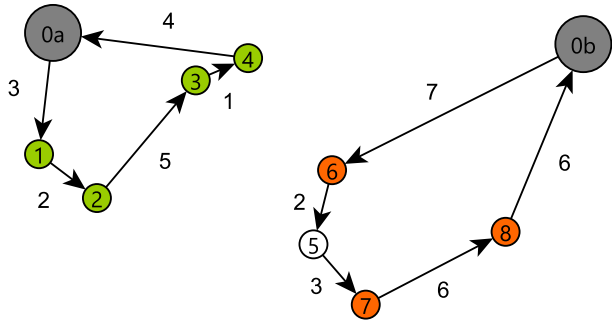

Figure 2.4b: Example of an alternative VRP-solution.

ORTEC Route Optimization Service

When only cheapest insertion is used for making intermittent routes, the vehicle routes would quickly become too inefficient and time slots would become full too early in the booking horizon. That is why intermittent optimization calls are done to ORS, the vehicle routing solution of ORTEC. Figure 2.4b shows the result of such an intermittent re-optimization. ORS uses a large collection of tunable algorithms, mainly constructive heuristics and metaheuristics (see Section 3.1), to optimize vehicle routes. The call to ORS can be triggered by several parameters, e.g., time (call ORS every 15 minutes), incoming customers (call ORS every 20 customers) or other parameters. Such an ORS optimization can take a long time. During the calculation time, new customers can arrive. So, as soon as ORS is done, the current OTSS vehicle routing schedule, based on the previous schedule including cheapest insertions of new customers, and the ORS routing schedule, a new routing schedule that lacks the new insertions, need to be merged. This merged schedule is the new OTSS routing schedule on which the new insertions will be based. For the merging of these two schedules, several different strategies exist, e.g., calling

ORS again when there is a difference between the two schedules or doing a cheapest insertion of the conflicting customers into the new ORS routing schedule. More merge strategies can be found in Cederhout (2020). ORS is also called one final time when no new customers can arrive and the final vehicle routing schedule needs to be constructed.

2.3 E-grocery Retailer Case

One of the main contemporary application areas of time slotting is e-grocery retailing, i.e., offering the possibility to order groceries online and delivering them at home. The reason that grocery retailers use time slots for delivery is that the goods are often perishable, so a failed delivery can be costly. E-grocery-specific elements are the delivery duration, customer dispersion, the fleet structure and capacity, depot locations and structure, and the relevance of time slot offering. The delivery duration, the time between arrival at and departure from a delivery address, is typically somewhere between 5 and 15 minutes. Customers are often dispersed over a larger region containing cities and rural areas. In cities, the customers are heavily clustered, in rural areas the customer locations are more dispersed. Grocery retailers typically have a heterogeneous fleet, with smaller vehicles used for cities and larger vehicles for rural areas. Often there is a single large depot, sometimes supported by several satellite hub locations. Finally, there is a difference between e-grocery retailers' time slot offering policy: price-competing retailers offer a limited number of time slots to their customer to save costs, while other retailers prefer to offer high customer service by offering as much time slots as possible. We study the second case, where supermarkets offer multiple time slots but try to steer customer behavior with dynamic pricing policies.

Although there are some case-specific elements, our research is aimed at obtaining general solution methods and insights, and will not be limited to e-grocery retailers, but more general applicable to vehicle routing problems with time windows.

2.4 Conclusions

In this chapter, we introduced the general business process of a customer ordering process with time slots. We showed the horizon used for time slot bookings and introduced a limitation to our research by only considering a single delivery day. We explained how the two ORTEC products (OTSS and ORS) are integrated in the time slotting process and we elaborated on the inner working of both software solutions. We explained the general idea of cheapest insertion for VRPTWs and discussed the advantages and disadvantages of this approach. Finally, we introduced the e-grocery retail case we study for this research and explained the case-specific elements. Also, we stressed that our research is not limited to e-grocery retail, i.e., our methods and results can be used more generally for vehicle routing problems with time windows.

Chapter 3

Literature

In this chapter, the research questions regarding the current scientific literature are answered. First, the vehicle routing problem with time windows is introduced in Section 3.1. Next, the attended home delivery and time slotting literature is reviewed in Section 3.2. Then, the time slot pricing literature is treated in Section 3.3 and the modelling of customer behavior regarding the choice of time slots is reviewed in Section 3.4. Section 3.5 treats the different cost approximation approaches found in scientific literature. In Section 3.6 we briefly introduce our chosen validation technique. Finally, the research questions related to this chapter are answered and the contributions to scientific literature are stated at the end of this chapter in Section 3.7.

3.1 Vehicle Routing with Time Windows

The vehicle routing problem (VRP) is concerned with the scheduling of vehicle routes, while adhering to vehicle capacity constraints and timing demands. The common objective for most VRPs is the minimization of transportation costs and maintaining a required service level. For a review and taxonomy of the existing scientific literature about VRPs, we refer to Bektas (2006), Braekers et al. (2016) and Konstantakopoulos et al. (2020). Since attended home delivery with time slots requires delivery to take place in a specified time interval, we consider a variant of the VRP, the Vehicle Routing Problem with Time Windows (VRPTW). The VRPTW is a popular generalization of the VRP with as objective to plan vehicle routes at minimal transportation costs, starting and ending at a central depot. There is a given fleet of vehicles and customer demand is known upfront. Each customer is assigned once to a vehicle, and vehicle capacities must be taken into account. Braekers et al. (2016) classify the VRPTW into customer restricted VRPTW, depot restricted VRPTW and vehicle restricted VRPTW. We only consider the customer restricted variant, which means that delivery to customers must be within the boundaries of the earliest and the latest time of a time window. Time windows can be hard, meaning that delivery must always occur within the indicated time window, or soft constraints can be used to allow for some slack in the delivery time, possibly against some penalty. Practical applications, other than attended home delivery, are amongst others, deliveries to banks, postal services and restaurants, school bus routing and security patrol services (El-Sherbeny, 2010). Again, there are extensions on the VRPTW, e.g., the periodic VRPTW (PVRPTW) which considers a cyclical planning horizon, or the multi-depot VRPTW (MDVRPTW), which includes multiple depots from which vehicles can start and end their routes. Most exact VRP formulations can be classified as flow-formulations, for which the costs of going from vertice i to vertice j are minimized, given a set of constraints, and set-partitioning formulations, for which subsets of feasible routes are selected that satisfy all constraints and minimize transportation costs (Munari, 2016).

We first state the most basic formulation of the capacitated VRP and then extend it to include time windows. We only state the flow formulation since it is most accessible. The

formulation is based on Munari (2016). The binary decision variable x_{ij} takes on the value 1 if the route goes from customer i to j directly, for $i, j \in \mathcal{N}$, where $\mathcal{N} = \mathcal{C} \cup \{0, n+1\}$. $\mathcal{N}$ are all vertices associated to the customers in $\mathcal{C}$ and the depot vertices 0 and $n+1$. We use two vertices to represent the same single depot at which all vehicle routes start and end. Also, y_j is a continuous decision variable corresponding to the cummulated demand on a route that visits vertice $j \in \mathcal{N}$ up to the visit of j.

$$\text{minimize} \quad \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} c_{ij} x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{\substack{j=1 \\ j \neq i}}^{n+1} x_{ij} = 1, \qquad \forall i = 1, \ldots, n, \tag{3.2}$$

$$\sum_{\substack{i=0 \\ i \neq h}}^{n} x_{ih} - \sum_{\substack{j=1 \\ j \neq h}}^{n+1} x_{hj} = 0, \qquad \forall h = 1, \ldots, n, \tag{3.3}$$

$$\sum_{j=1}^{n} x_{0j} \leq K, \tag{3.4}$$

$$y_j \geq y_i + q_j x_{ij} - Q(1 - x_{ij}), \qquad \forall i, j = 0, \ldots, n+1, \tag{3.5}$$

$$d_i \leq y_i \leq Q, \qquad \forall i = 0, \ldots, n+1, \tag{3.6}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j = 0, \ldots, n+1. \tag{3.7}$$

The objective function 3.1 minimizes the transportation costs, calculated by multiplying the cost of traversing an edge (c_{ij}) with the binary decision variable x_{ij}. Constraints 3.2 make sure that all customers are visited exactly once. Constraints 3.3 are flow preservation constraints, meaning that an incoming vehicle to vertice $h \in \mathcal{N}$ must also depart from this vertice. Constraints 3.4 limit the maximum number of routes to the number of vehicles K. Both Constraints 3.5 and Constraints 3.6 guarantee that the vehicle capacity is not exceeded. Constraints 3.5 are also subtour elimination constraints. Subtours are cyclical routes that do not pass through the depot, i.e., yield infeasible solutions. Many formulations exist for the subtour elimination constraint, Constraints 3.5 and Constraints 3.6 have as advantage that the model has a relatively low number of constraints in terms of the number of customers, but as disadvantage that the lower bound of the relaxation is relatively weak in comparison with other formulations (Munari, 2016).

To extent this capacitated-VRP formulation to also include time windows, we need to add more constraints. Let a time window be expressed by the time interval $[w_i^\alpha, w_i^b]$. The delivery at vertice $i \in \mathcal{N}$ cannot occur earlier than w_i^α nor later than w_i^b. To each edge (i, j) we assign a travel time t_{ij}. Also, we consider the fact that delivery cannot be done instant, i.e., it takes some time s_i to deliver the package after arrival at vertice i. We introduce a new continuous decision variable w_i, being the time of arrival at vertice $i \in \mathcal{N}$. We add the following constraints to 3.1 to 3.7:

$$w_j \geq w_i + (s_i + t_{ij}) x_{ij} - M_{ij}(1 - x_{ij}), \qquad \forall i = 0, \ldots, n; \forall j = 1, \ldots, n+1, \tag{3.8}$$

$$w_i^a \leq w_i \leq w_i^b, \qquad \forall i = 0, \ldots, n+1. \tag{3.9}$$

Constraints 3.8 ensure that travel time and delivery time are correctly calculated, with M_{ij} being a large value ($M_{ij} = max\{w_i^b - w_j^\alpha, 0\}$). Constraints 3.9 than ensure that delivery is done within the indicated time window.

This exact formulation can be suitable for understanding the problem structure of the VRPTW. Solving the exact formulation is not possible, since the VRPTW is NP-hard (Lenstra and Kan, 1981). This means that the exact formulation as stated above, cannot be solved to optimality for realistic instances and we must revert to approximate methods that obtain a sub-optimal solution. Konstantakopoulos et al. (2020) classify the VRP-heuristics in constructive heuristics, two-phase heuristics and local improvement heuristics. Constructive heuristics start with an empty solution, and step-by-step construct a complete solution using some set of decision rules at each iteration, a well-known example is the "Clarcke and Wright savings algorithm" (Clarke and Wright, 1964). Two-phase heuristics for VRPs first construct a solution for a simplified problem, then the first solution is adapted in the second phase to adhere to the problem characteristics. An example of this is the "Route-First-Cluster-Second" heuristic, which first constructs a giant tour for a single vehicle, and than splits this tour into feasible routes for multiple vehicles (Prins et al., 2014). Finally, a local improvement heuristic makes small adaptations to an initial solution to reduce costs. An example of a local improvement procedure is 2-opt, which is the swapping of two edges that cross each other (Croes, 1958).

Aside from the heuristics category, Konstantakopoulos et al. (2020) distinguishes metaheuristics, which are advanced methods used to find, generate or select a subset of solutions to try to find the best. Metaheuristics are then classified into several categories. First, local search is a method that explores the solution space by moving to promising solutions in a neighborhood. Examples are simulated annealing (Osman, 1993), tabu search (Montané and Galvão, 2006) and variable neighborhood search (Kuo and Wang, 2012). Second, population search is a method that aims to generate a new solution from a set of solutions by combing and pairing existing solutions, or by making solutions cooperate through a learning process, examples are the genetic algorithm (Baker and Ayechew, 2003), particle swarm optimization (Goksal et al., 2013) and ant colony optimization (Bell and McMullen, 2004).

3.2 Attended Home Delivery and Time Slotting

In Section 1.2, we already introduced attended home delivery (AHD) and time slotting. In this section, we give an overview of the state-of-the-art scientific literature in the field and we make a classification of the literature based on distinctive characteristics of the problems and solution methods.

In the seminal work on AHD by Campbell and Savelsbergh (2005) the, at that time, current practice in time slotting is explained. This practice uses a fixed number of bookings per time slot, based on fleet size and historic delivery times. When the time slot is booked up to its capacity, it is no longer presented as an option. Customers can choose all available time slots, regardless their order or home address. Campbell and Savelsbergh (2005) note that future models can improve the home delivery profitability by (i) dynamically determining whether a delivery can still be accommodated based on already existing routes and the new customer' location, and (ii) by considering the opportunity costs associated with accepting a customer, in view of possible later customer arrivals that are more profitable. After 2005, AHD received more attention. In a recent scientific literature review by Snoeck et al. (2020), the current standing of the field is summarized. They categorize the scientific literature based on four criteria: (i) control policy, (ii) decision time frame, (iii) routing model and (iv) customer choice model. The control policy (i) is the way a retailer controls the demand flow. This can be done using a pricing policy, i.e., differentiating prices per time slot to steer demand to certain time slots, or by using a quantity policy, which means that time slots can be closed for booking (Snoeck et al., 2020), the control policies will be discussed in Section 3.3. The time frame (ii) can be static or dynamic. Static decisions, in contrast to dynamic decisions, do not depend on real-time data. The criteria for the routing model (iii) concerns the algorithms that are used for the final routing of vehicles. Because of the problem characteristics, most scientific literature uses VRP

heuristics, as briefly discussed in Section 3.1. Finally, the customer choice model (iv) is the way the modelled customer responds to the decision making. Customer choice models will be further discussed in Section 3.4.

The strategic and tactical decisions regarding attended home delivery and time slotting can be, amongst others, about the fleet size, area of operation or time slot design. Most literature deals with a predefined number of time slots, with certain width and overlap. For instance, in Hernandez et al. (2017), a time slot schedule for a geographical zone must be selected using a tactical routing plan based on approximations. The tactical routing is only an approximation of the operational routing decision, but it can be a good way to select the best time slotting design. Building on earlier work by Agatz et al. (2011), Klein et al. (2019) present a mixed-integer linear programming formulation to anticipate delivery costs. They also include differentiated pricing for each time slot and model customer behavior to make the best tactical time slotting decision.

We do not consider the strategic and tactical decision making any further, and focus on operational decision making, since operational settings demand the most from dynamic models. First, we examine the problem structures used in case studies and numerical experiments. Most scientific research applies a conceptual model or framework, where the reality is often drastically simplified. Our literature review will help to get to know more about the differences between our case study and the case studies in scientific literature. In Section 3.3 the selected literature is examined for time slot allocation and incentives, and in Section 3.4 the used customer choice models are described.

We use the following classification elements for the problem structure: (i) delivery horizon length, (ii) customer arrival process, (iii) demand generation, and (iv) time slot design. The delivery horizon length (i) indicates how many days a customer can choose for delivery. The customer arrival process (ii) can be modelled using different probability distributions or can be based on historical data. The order generation (iii) is the way the orders (e.g., quantity, location or time slot) are generated. The time slot design (iv) indicates what width and possible overlap is considered by the authors. See Table 3.1 for the classification.

In Asdemir et al. (2009), a model is presented, but no numerical experiments are conducted. The model allows for a flexible horizon, but does not consider days of the week, nor seasonality. The customer arrival process is modelled using a non-homogeneous Poisson process via uniformization, as inspired by scientific work in revenue management in the airline industry (see Lee and Hersh (1993)). The order size is uniform among all customer classes. Campbell and Savelsbergh (2006) test their models on fictitious cases for which customers are uniformly scattered on a 60×60 grid. The time slots are for a single day while experiments are done with various widths of non-overlapping time slots. Customer arrivals are uniformly distributed between 0 and the cutoff time T. Vehicle capacity is not considered a restraining factor, since this is often the case in practice, where the time windows are the biggest constraint (Campbell and Savelsbergh, 2006). In Cleophas and Ehmke (2014), a computational study is conducted based on the metropolitan area of Stuttgart, which is divided in nine areas with varying population sizes that can choose between eight time slots. Demand is drawn from the normal distribution and is dependent on the area and the average income in those areas. There is a fixed number of vehicles (four); capacity is estimated with vehicle routing experiments. In a study that also considers metropolitan areas (Ehmke and Campbell, 2014), different travel time patterns are considered to model congestion in the morning peak-hours. There are three available vehicles, without capacity restrictions. Demand for the eight non-overlapping time slots is uniform, and for another experiment there is a demand peak for some time slots. Note that both Cleophas and Ehmke (2014) and Ehmke and Campbell (2014) model demand for time slots using a customer choice model, as further explained in Section 3.4. In Klein et al. (2018), 12 areas are served by a single central depot, at which the set of 1000 customers can arrive randomly, one at the time, and the size of demand is defined using the number of order totes. The fleet size is varied in the experiments and the capacity is the same for every vehicle.

Table 3.1: Classification of operational AHD and time slotting literature: problem structure.

Authors	Delivery horizon	Customer arrival process	Order generation	Time slot design
Asdemir et al. (2009)	Flexible	N/A	Uniform	N/A
Campbell and Savelsbergh (2006)	Single-day	Uniform	-	Non-overlapping, 1 or 2-hour width
Cleophas and Ehmke (2014)	-	N/A	Area-based, normal dist.	8 time slots
Ehmke and Campbell (2014)	Single-day	N/A	Uniform, Demand peaks	8 Non-overlapping, 1-hour width time slots
Klein et al. (2018)	Single-day	Random	General dist. of nr. of totes $i \in \{1,\ldots,10\}$	4 Non-overlapping, 2-hour width time slots
Yang et al. (2016)	Single-day	Time-dependent Poisson arrivals	Normal dist.	27 Partly-overlapping, 1-hour width time slots
Yang and Strauss (2017)	Single-day	Homogeneous Poisson arrivals	General dist. of nr. of totes $i \in \{1,\ldots,10\}$	17 Non-overlapping, 1-hour width time slots

Although the delivery time slots are all in a single day, the booking horizon covers a longer period. Yang et al. (2016) test their model on a realistic case, for which bookings on a single day arrive as early as 22 days in advance, with most bookings coming in the last three days before the cutoff time. Cancellation and re-scheduling is neglected. The number of vehicles at the single depot is varied, but vehicle capacity is homogeneous. The order quantity (number of totes) is drawn from the normal distribution, with mean and standard deviation obtained from historic data. They model the arrivals as a time-dependent Poisson process, and note that it may be an interesting future research direction to consider the dependence of delivery days. In a follow-up study, Yang and Strauss (2017) use the same data set, but the arrival process and vehicle capacity is modelled differently. The order quantity is again expressed in number of totes but not drawn from the normal distribution but from geographical data.

Summarizing, we see that many different problem aspects are considered in scientific literature. Over the years, problem instances have become more realistic and less limitations to generalizability are present. However, still many aspects of reality are neglected, e.g., longer planning horizons, multiple depot problems, different time slot designs, and complexities in geographical data. There is still a large difference between the problem complexity and realism in comparison with other research fields, e.g., the rich vehicle routing problem (Caceres-Cruz et al., 2014). We see that the problem formulations mostly consider the time slot allocation and incentives decisions, based on approximation of transportation costs. The time slot design and the effect of it on the results is less studied. Nevertheless, time slot design has a large influence on both operational revenues as customer satisfaction, since offering many options and short time windows is associated with high customer satisfaction (Agatz et al., 2008).

In the next section, we examine the used time slot allocation and incentive policies used in scientific literature.

3.3 Time Slot Allocation and Incentives

Time slot allocation and incentive literature has similarities with the literature concerning revenue management. Revenue management can be defined as the activities related to managing demand for products or services (van Ryzin and Talluri, 2005). Originated in the airline industry, revenue management is used to offer different categories of tickets and dynamically controlling the offered quantity of each ticket-segment to maximize profits (Snoeck et al., 2020). This process is closely related to time slotting, since the retailers face a heterogeneous market with limited flexibility in capacity. Also, the prices and product availability (time slots) can be changed to increase profits. However, Snoeck et al. (2020) notice three differences between airlines and retailers. First, the retailers sell both a physical product (e.g., groceries) and a service (delivery within a time slot). The product order characteristics have an influence on time slot capacity and profits. Second, the customer location and the other customers' location influence the operational costs. And third, customer choice models (see Section 3.4) can be better fitted to data since customers that request a time slot often also commit to buying, while customers buying airline tickets often check multiple prices at different providers before committing to a buy (Snoeck et al., 2020).

The field of AHD is typically divided into the following categories: static time slot allocation, dynamic time slot allocation, differentiated pricing, and dynamic pricing (Agatz et al., 2013; Yang et al., 2016; Klein et al., 2019). These classes are shown in Table 3.2. Time slot allocation can be summarized with the question: "which time slots should we offer to a customer?" and time slot pricing can be stated as: "which time slots should we incentivize and which should be penalized?".

Table 3.2: Classification of revenue management in AHD (Agatz et al., 2013; Yang et al., 2016; Klein et al., 2019).

	Time slot allocation	**Time slot pricing**
Static (Offline)	Differentiated slotting	Differentiated pricing
Dynamic (Online)	Dynamic slotting	Dynamic pricing

Static methods use forecast data or static rules and can be used to make strategic and tactical decisions, e.g., decide on the number of slots and the width of the slots. For differentiated allocation, the goal is to find which time slots to offer to which delivery area, e.g., certain low-populated areas might be offered less time slots, which is a tactical decision. Differentiated pricing tries to find the best static price policy to influence customer behavior. The decisions for a static pricing policy are not taken in real-time, meaning that pricing rules are either completely static and based on historic data, or they do include some rules based on, for instance, customer order value and location. Static methods can also be called "offline", which means that the computational efforts are not incurred during the decision making process (Powell and Ryzhov, 2013). When time slot allocation and pricing happens online, during the decision making, it is called dynamic. The decisions that are made in real-time have as advantage that they can be adapted to the current situation and do not fully rely on static rules. Dynamic decisions can consider real-time information about the customer and the current schedule to make better decisions. It also opens the possibility to consider opportunity costs, which are, to recall, the cost of offering a time slot now compared to saving time slots for future, potentially more profitable customers (Yang et al., 2016). However, the use of dynamic methods is limited by the computational time, since customers demand a fast working website, the time slot allocation and pricing cannot take too long. It should be noted that there is a difference between time slot allocation and feasibility checks. Time slot allocation and feasibility checks have a similar question to answer ("which time slots should be offered to a customer?"), but while for time slot allocation this may mean that certain time slots that still have capacity are closed for certain

customers, the feasibility check only concerns the question if a time slot can handle the new customer, regarding vehicle capacity and time windows. Before inserting a customer into an existing route, a feasibility check is always necessary.

We return to scientific literature to examine the time slot allocation and incentive methods that are used. Our classification will be based on (i) the slot allocation method, (ii) the slot incentive method, if applicable, and (iii) the objective variables of the algorithms. See Table 3.3 for the classification.

Table 3.3: Classification of operational AHD and time slotting literature: slot allocation, incentive method and objective.

Authors	Slot allocation method	Slot incentive method	Objective
Asdemir et al. (2009)	Feasibility check	Dynamic, Markov decision process model	Maximize net benefit
Campbell and Savelsbergh (2006)	Heuristic feasibility check	Dynamic LP-based model	Maximize total profits
Cleophas and Ehmke (2014)	Dynamic, ESMR	N/A	Maximize value of orders
Ehmke and Campbell (2014)	Static/Dynamic, I1 insertion heuristic	N/A	Maximize the number of accepted requests
Klein et al. (2018)	Feasibility check	Dynamic, MILP-model for opportunity costs	Maximize profits
Yang et al. (2016)	Heuristic feasibility check	Dynamic, opportunity costs, SDP	Maximize profits
Yang and Strauss (2017)	Feasibility check	Dynamic	Maximize profits

In Asdemir et al. (2009), a Markov decision process model is proposed that dynamically adjusts the delivery charges per customer. The optimal prices are calculated based on an "equal profit" policy, meaning that the retailer gains the same profit in the remaining booking horizon, regardless the customer choice. Delivery prices can change based on order size, depending on the time left in the booking horizon. The allocation method offers all time slots, only restricted by capacity constraints. The objective function maximizes net benefit, also concerning opportunity costs. Campbell and Savelsbergh (2006) develop a model that dynamically determines feasibility of a time slot insertion. This model uses a combination of insertion heuristics and randomization to determine a feasible schedule. Next, the allocation and size of incentives are determined using a linear programming model which maximizes the profits related to time slot offerings, meaning the total revenue minus incentive and delivery costs. The authors conclude the following from their research: (i) incentive schemes can substantially reduce costs, (ii) performance of incentive schemes can be improved using intelligent methods, (iii) incentives may reduce walkaways (lost sales), (iv) it is sufficient to provide incentives to only few slots (≤ 3), (v) an increase in time slots triggers the need for more sophisticated incentive schemes, (vi) it is easier to persuade customers to choose a wider time window than to let them choose a specific time slot, and finally (vii), the use of incentives can be critical already in the early stages of making a routing schedule (Campbell and Savelsbergh, 2006). In Cleophas and Ehmke (2014), the offering of time slots to customers is dynamically determined using the order value. The used method is called "Estimated Marginal Seat Revenue heuristic", also called EMSR, as described by Belobaba (1987). EMSR determines buckets for order values and allocates time slots accordingly, i.e.,

customers with a high order value, falling in a high-value bucket, will get more time slot offers than customers with low order value. The objective is to maximize the expected value of orders, given transport capacities. The authors do not consider the steering of customer behavior with time slot incentives. Ehmke and Campbell (2014) also do not consider incentives. They define both static as dynamic approaches to determine the time slot allocation to maximize the number of accepted time slot requests. The static methods uses capacity restrictions and a static rule that takes into account the time windows in which a delivery must be feasible. The dynamic method uses expected, dynamically determined, travel times. They expand this method to also have a buffer for lateness and consider stochastic travel times. Their insertion heuristic is a time-dependent adaptation to the well-known I1 insertion heuristic (Solomon, 1987). Klein et al. (2018) use a mixed-integer linear programming model which is integrated in a widely used dynamic programming model for AHD (Yang et al., 2016). Their MILP-model maximizes expected profits and is used as an approximation of opportunity costs. The time slots availability is checked, but time slots are always offered when capacity allows it.

The often cited dynamic programming model as described in Yang et al. (2016), is the proposed "de facto" framework for dynamic pricing. After doing a heuristic feasibility check, based on Campbell and Savelsbergh (2006), the insertion costs are calculated. Their pricing solution is dynamic, but for practical reasons it does not differentiate between customers that choose the same time slot and have the same location and order value. They develop two policies, one only considering the current insertion costs, the other also including the opportunity costs.

Their approximate dynamic programming (ADP) model aims to find an optimal price vector for time slots offered to a single customer that arrives in period t. The time periods t are assumed to be small enough such that only one customer can arrive during it. The delivery region is divided in non-overlapping sub-areas $a \in \mathcal{A}$. The vehicle fleet can be homogeneous or heterogeneous and is based at a single or multiple depots. The single delivery day is divided in potentially overlapping time slots $s \in \mathcal{S}$. The booking horizon consists of T periods (equal to the number of arriving customers) and ends after the cutoff time. Order sizes are assumed to be uniform. The profit r per customer is known before incentive decisions are made. These profits are often obtained from historic data (Yang et al., 2016). The occurrence probability that an order arrives from area a in time t is expressed with λ_{ta}. The state space $S_{t,a,s}$ contains the number of customer orders on time t, that need to be delivered in area a, in time window s. The decision to make is twofold: first the available (i.e., feasible) time slots to offer to a new customer need to be found, and next each available time slot needs to get a price. The feasibility check is ignored for now, since most research uses a separate feasibility check to limit the ADP-algorithm decision space. So, in our further notation we assume that all time slots in $\mathcal{S}$ are feasible. Therefore, the decision x_t is a vector of dynamically determined prices, often chosen from a discrete list. The single price element in the decision vector is indicated with x_{ts}. Negative prices represent discounts or incentives. The delivery costs are determined using differing methods, in Yang et al. (2016) a Daganzo-formula is used to obtain the delivery costs and in Klein et al. (2018) a seed-based approximation is applied. The delivery costs are expressed with $C(S_t, x_t)$. The exogenous information W_{t+1} is the chosen time slot by a customer, given the presented price vector. The probability that a customer selects a time slot s is expressed with $\mathbb{P}_s$ and is dependent on the state S_t and decision x_t. The Bellman-equation maximizes benefits:

$$V_t(S_t) = \max_{x_t \in \mathcal{X}} \left(\sum_{a \in \mathbb{A}} \lambda_{ta} \sum_{s \in \mathcal{S}} \mathbb{P}_s(x_t) \Big[r + x_{ts} - \big(V_{t+1}(S_t) - V_{t+1}(S_t + 1_{as}) \big) \Big] + V_{t+1}(S_t) \right). \quad (3.10)$$

The unit vector 1_{as} indicates the addition of a single customer to ATC-pair (a, s). The term $(V_{t+1}(S_t) - V_{t+1}(S_t + 1_{as})$ represents the opportunity costs of adding the new customer (Yang et al., 2016). The optimal pricing policy that solves the Bellman-equation is given by:

$$x_t^* = \underset{x_t}{\operatorname{argmax}} \sum_{s \in \mathbb{S}} \mathbb{P}_s(x_t)\big[r + x_{ts} - O_{tas}\big], \tag{3.11}$$

where O_{tas} is a separate expression for the opportunity costs, incurred when adding the customer arriving on time t in area a to time slot s, which need to be approximated too (Klein et al., 2019). For more information about approximate dynamic programming, we refer to Powell (2007), Mes and Perez Rivera (2017) and Sutton and Barto (2018).

Yang et al. (2016) showed that dynamic pricing methods that do not take future expected demands (i.e., opportunity costs) into account can produce worse results than static pricing methods (Yang et al., 2016). In a later work, Yang and Strauss (2017), expanded their model to use an area-specific cost estimation as input for an approximate dynamic programming approach. They show that the decomposition into smaller areas can successfully decrease computational efforts and estimate the costs.

Apart from monetary penalties or incentives, retailers can offer different type of incentives, like bonus points or, "green-labels", that indicate the environmentally best choice. The effect of these "green" options is studied in Agatz et al. (2020). They show, with a combination of experiments with volunteers and simulation techniques, that green labels are an effective tool to steer behavior, especially for environmental-conscious individuals. The green labels remain effective when used together with price incentives, and the effectiveness remains when time slots are wider.

Summarizing, we see that the selected literature chooses exclusively between time slot allocation and time slot incentives. In the literature that only concerns incentives, it is often stated that the closing of time slots for certain customers (i.e., time slot allocation) is a method that results in lost sales and customer dissatisfaction (Asdemir et al., 2009). In literature that concerns incentives, dynamic pricing is perceived as the better method, since it can balance the trade-off between lost sales and profits. Also, we see that the topic of cost approximation, being opportunity costs or transportation costs, is much studied. We recognise two different options for dynamic pricing solutions: (i) approximate the costs of a time slot and use this as basis for setting time slot prices (Campbell and Savelsbergh, 2006), or (ii) optimize the time slot prices, such that the behavior of customers is nudged optimally, like is done in the ADP-model of Yang et al. (2016). Most research focuses on maximizing the (expected) profits, although maximizing the number of accepted requests is also considered. We further consider transportation cost approximation literature in Section 3.5.

In the next section, we discuss customer choice models.

3.4 Customer Choice Modelling

The scientific field of customer choice models is widely spread. Discrete choice models are nowadays used in many fields for which some customer choice needs to be modelled, e.g., in transportation, housing, energy, and marketing (Train, 2009). A discrete choice model is a behavioral model that contains a choice set, the options to choose from, a certain probability for each choice option, and a certain customer utility, i.e., benefit (Train, 2009). Discrete choice models can be categorized as parametric, non-parametric, or multi-stage (Strauss et al., 2018). Parametric models are the most used category. These type of models use utility theory, which is based on utilities given to each choice alternative, and assumes that customers always maximize their utility (Train, 2009; Strauss et al., 2018). Although these models are widely used, they have as weakness that the customer choice is captured in a function, that might not correctly follow the actual choice behavior (Strauss et al., 2018). As an alternative to parametric models, non-parametric models mostly use choice-rankings. A ranking of all alternatives is made, and the customer chooses the available option with the highest ranking. Rankings are often based

on customer types or customer segments. Finally, multi-stage models use a "consider-then-choose" strategy, for which the modelled customer first make a consideration set, after which they choose from the available alternatives (Strauss et al., 2018).

In the remainder of this section, we examine scientific literature and describe the currently in use customer choice models in more detail. See Table 3.4 for an overview of the used customer choice models.

Table 3.4: Classification of operational AHD and time slotting literature: customer choice model.

Authors	Customer choice model
Asdemir et al. (2009)	Multinomial logit
Campbell and Savelsbergh (2006)	Exogenous probability
Cleophas and Ehmke (2014)	Exogenous probability
Ehmke and Campbell (2014)	Exogenous probability
Klein et al. (2018)	Multinomial logit
Yang et al. (2016)	Multinomial logit
Yang and Strauss (2017)	Multinomial logit

Asdemir et al. (2009) use a logit choice model, as has become the common method in later works. The multinomial logit (MNL) choice model assumes that the entire customer population can be described with a set of parameters β (Strauss et al., 2018). The utility of alternative j for a customer is expressed with:

$$U_j = u_j + \epsilon_j, \tag{3.12}$$

for which u_j is a deterministic component that can be influenced, and ϵ_j is a random, i.i.d. variable that follows a Gumbel distribution with a mean of zero and variance of $\mu^2\pi^2/6$ with $\mu > 0$ (Yang et al., 2016; Strauss et al., 2018). The deterministic part follows a linear function:

$$u_j = \beta_0 + \beta_s + \beta_d, \tag{3.13}$$

where β_0 is the base utility across all options, β_s is the utility associated with the time slot itself, and β_d is the utility sensitivity to the incentive (Yang et al., 2016). The values for β need to be fitted to actual customer segment data. A customer is assumed to always maximize its utility, i.e., choose the highest U_j. Yang et al. (2016),Yang and Strauss (2017) and Klein et al. (2018) use a similar customer choice model. As a recent alternative to the MNL choice model, Klein et al. (2019) use a non-parametric rank-based model to model customer choice, for which customers have a ranking for all alternatives and choose the best ranking available alternative. More simple models use probabilities that can be manually adjusted, but in general these methods do not use incentives (Cleophas and Ehmke, 2014; Ehmke and Campbell, 2014), or they use a more advanced methodology to adjust probabilities according to incentives (Campbell and Savelsbergh, 2006). In Campbell and Savelsbergh (2006), the probability p_i^t a customer i will select a time slot t is linked to the (size of) incentive B. To compensate for an increase of certain probabilities due to incentives, the probabilities of selecting non-incentive time slots

decrease. The authors mention, that from practical experience, even small incentives can change customers' selection of delivery windows.

Summarizing, we see that most recent literature uses relatively simple parametric models that can be based on actual data. Rank-based models are used also. The more simplistic probabilistic models, that are easy to set up and to tune, are not that common anymore, largely because of the necessary assumptions and complicated method to make these models adaptive to incentives.

3.5 Transportation Cost Approximation

As Snoeck et al. (2020) recognised, attended home delivery literature can also be categorised on the method for including routing costs. Most literature uses the costs resulting from explicit routing decisions, often obtained from a heuristic, since the VRPTW is NP-hard (Campbell and Savelsbergh, 2006; Cleophas and Ehmke, 2014; Ehmke and Campbell, 2014; Yang et al., 2016). Alternatively, an approximation of the routing costs, without making explicit routing decisions, can be used, e.g., with Daganzo-approximation (Robuste et al., 1990) or a seed-based approximation method (Klein et al., 2018, 2019). In this section, we describe several different cost approximation methods, stemming from both AHD-literature as literature from other fields of research. Many authors of AHD-literature try to obtain a computationally fast approximation of the (transportation) costs and revenues for inserting a newly arrived customer in a time slot. As described in Chapter 2, transportation costs can be expressed in driving distance, number of used vehicles, time to deliver all customers, and many more costs indicators.

In Section 3.5.1, we discuss some well-known heuristic functions for approximating routing costs without solving a TSP or VRP. In Section 3.5.2 a seed-based method and its application to an AHD problem is described. Finally, in Section 3.5.3, the theory behind regression models and their potential for AHD is explained.

3.5.1 Heuristic Functions for Approximating Transportation Distance

In TSP and VRP research, various simple heuristic functions have been developed to approximate, mostly, travel distances. One of the first studies that used a distance approximation (Beardwood et al., 1959), recognised that for an instance with n customers and a convex area A, the length of a tour can be approximated with:

$$\text{TSP distance} \approx c\sqrt{nA}, \qquad \text{when n} \to \infty, \tag{3.14}$$

with c being a constant. Other TSP distance approximations were developed by, amongst others, Christofides and Eilon (1969); Chien (1992); Hindle and Worthington (2004). These authors identified the importance of including the shape or size of the area of operation. They used different methods for including parameters for the area, e.g., fitting the smallest rectangle that covers all customer (Chien, 1992) or by looking at the densities of customers in sub-areas (Hindle and Worthington, 2004).

VRP distance-approximations are more complex than for the TSP, since these need to consider multiple vehicles and vehicle capacities. The distances between customers and the depot (Webb, 1968), standard deviations of coordinates and distances between customers and the center of the area A (Çavdar and Sokol, 2015), and capacities of vehicles (Eilon et al., 1974), are used as parameters for VRP distance approximation. A well-known approximation of routing costs is Daganzo-approximation. The formula gives a fairly accurate estimate of the distance of a vehicle routing problem (Robuste et al., 1990). The equation is shown below:

$$\text{VRP distance} \approx [0.9 + \frac{kN}{C^2}] \cdot \sqrt{AN}, \tag{3.15}$$

where k is an area shape constant, N the number of customers, C the maximum number of customers a vehicle can serve, and A is the area size. In a later work, time window constraints are considered by partitioning the time horizon in periods and aggregating customers in rectangles (Daganzo, 1987). One of the more recent works on VRP distance approximation by Figliozzi (2008) improved the approximation with the following formula:

$$\text{VRP distance} \approx b\frac{n-m}{n}\sqrt{An} + m2r, \tag{3.16}$$

with b being a constant estimated with linear regression, n the number of customers, m the number of vehicles, and r being the average distance between customers and the depot.

3.5.2 Seed-based Approximations for Transportation Costs

A much used method in attended home delivery, is to approximate routing costs on a seed-based scheme. Seed-based schemes partition routing costs in two separate costs: depot-to-seed costs and seed-to-customer costs. Seeds are conceptual sub-depots, from where customers are served. A graphical representation of the seed-structure is shown in Figure 3.1.

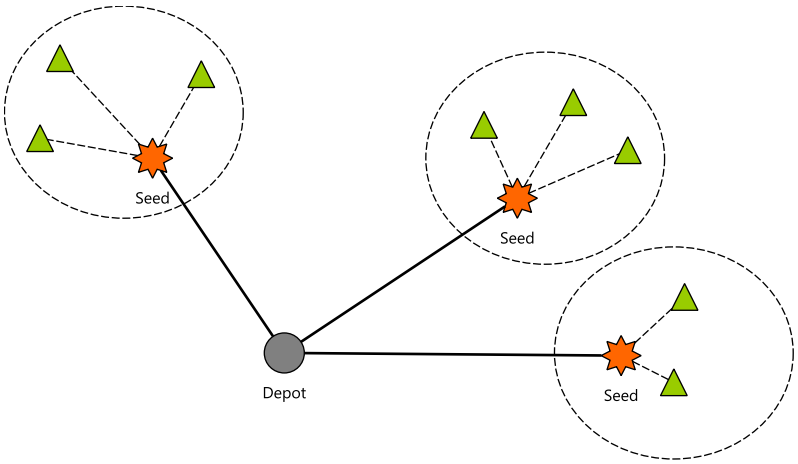

Figure 3.1: Example of a seed-based structure.

The advantage of this structure is that only the distance from the seed to the customer needs to be determined, depending on the seed-area a customer is assigned to. Agatz et al. (2011) present an integer program, based on a seed-based approach, that determines what time slots to offer in each zip-code, which is a tactical decision and therefore not applicable to our problem. In Klein et al. (2018), a seed-based scheme is used for the operational problem of assigning incentives to time slots. They model delivery costs with a seed-based approach that anticipates future demand management, by making forecasts about the residential areas of expected customers. Their model aggregates customers by "area-time slot combinations" (ATC), represented by the tuple (a, s), for each area $a \in \mathcal{A}$ and time slot $s \in \mathcal{S}$. Since it is unknown where future customers will be located, they use historical data to obtain the *expected seed-to-customer distance* $\check{d}^v_{as}$ for each vehicle $v \in \mathcal{V}$. The distance $\check{d}^v_{as}$ is the average distance between the seed in (a, s) and the historical customers in pre-defined area a. Secondly, the vehicle needs to drive once from the central depot to the first seed, which is called the depot-to-seed distance $\widehat{d}^v_{as}$ (Klein et al., 2018). The seed locations are made more accurate by dynamically adjusting the seed location according to already booked customers. A graphical reprsentation of their approach, including the time slot aspect, is shown in Figure 3.2, for a more elaborate explanation we refer to Klein et al. (2018).

Figure 3.2 shows an illustration for the ATC-structure, for illustrative purposes only showing for a single vehicle, four sub-areas, and two time slot options. Areas 1, 2 and 3 show the already

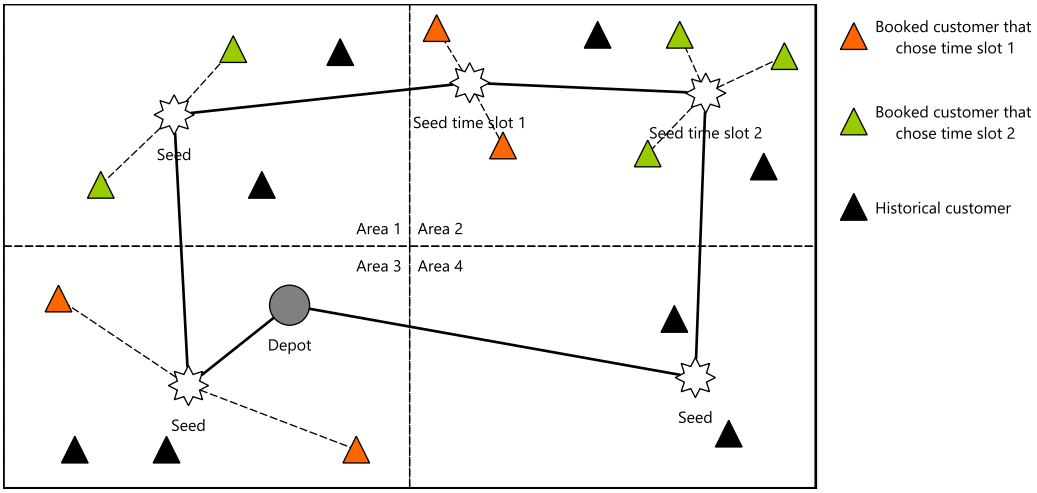

Figure 3.2: Example of a seed-based structure for area-time slot combinations as described in Klein et al. (2018).

accepted and booked customers, in Area 4 no customers have been booked yet. This shows that the seed is initially based on historic data (Area 4), but is adapted as soon as new customers have been booked (Areas 1, 2 and 3). Area 2 contains customers from both time slot 1 as time slot 2, so it has two separate seeds, for (a, s)-pairs: $\{(2, 1), (2, 2)\}$. The tour, as shown in Figure 3.2, follows the shortest path from depot-to-seed, next from seed-to-seed, and finishing by one seed-to-depot distance. To incorporate future demand, the number of expected customers from each area a that chooses time slot s is determined by assuming probabilities that a customer chooses a certain time slot s. These expected number of requests for delivery at (a, s) are multiplied by the seed-to-customer distances and the corresponding seed-to-seed, depot-to-seed, and seed-to-depot distances are added, which results in the total expected delivery distance.

3.5.3 Regression Models for Approximating Transportation Costs

Regression models, e.g., linear regression, random forests or neural network regressors, are statistical learning methods that can be used to predict quantitative responses. For linear regression, we try to predict the value of observation y^n of the target variable, based on a set of *features* $\{x_1^n, x_2^n, \ldots, x_I^n\}$. We try to estimate feature coefficients that solve:

$$\min_{\beta} \sum_{k=1}^{n} \left(y^k - (\beta_0 + \sum_{i=1}^{I} \beta_i x_i^k) \right)^2, \tag{3.17}$$

where the *feature coefficients* β, are constants that are estimated with the regression model. The parameter β_0 is called the *intercept*, and represents the bias of the model (James et al., 2013). Bias is an offset for all predictions we make. The features used to predict the target can be numerical or categorical. Categorical features have multiple levels, called dummy variables. The linear regression method is simple to implement, but can only approximate linear relationships between features and the target.

Decision trees is a method that splits data into separate leaves by means of sequential binary decisions (James et al., 2013). A regression decision tree is build in (roughly) two steps: (i) divide the feature space, i.e., all possible values for $\{x_1^n, x_2^n, \ldots, x_I^n\}$, in J non-overlapping regions $R_1, R_2, \ldots, R_J$, and (ii) for every observation that falls in region R_j we make the same prediction, which is the mean of the observations in region R_j. The goal for a decision tree algorithm is to find a set of regions R that predict with the lowest error. Although decision trees

can better handle non-linear relationships between features and the target, the performance of decision trees is lacking in comparison with other regression models (James et al., 2013).

Random forests is an algorithm that builds multiple decision trees. Every time a region split is considered, a random sample of m features is considered as split candidates from the full set of features X. The number of features m considered for a split is typically $\sim \sqrt{p}$, although this can be tuned (Breiman, 2001). The selection of features ensures that the strongest features are not always in the top splits of the decision trees, instead it gives more chance to weaker features. The process of selecting different subsets of features and ensuring that trees do not resemble each other too much, is called decorrelating (James et al., 2013). The advantage of random forests over linear regression and decision trees is that it can handle non-linear relationships, is harder to overfit on training data, can report feature importance, and is robust to outliers (Elith et al., 2008).

Artificial neural networks are a popular learning technique that simulate the biological learning mechanisms as found in the human brain, with as important property that they can be used in a supervised setting and an (semi-)unsupervised setting (Aggarwal, 2018). Further benefits opposed to other methods are the ability to handle nonlinear relationships and the adaptivity to new data (Haykin, 1998). In the language of the field, neural networks consist of three components: (i) an input layer receiving external data, (ii) hidden computation layers, and (iii) an output layer that produces the final output. Each node in these layers is called a *neuron* or *perceptron*. The output of a neuron is the input of another neuron in the network. Neurons can have multiple inputs and outputs (Aggarwal, 2018). A graphical representation of a three-layer neural network is shown in Figure 3.3.

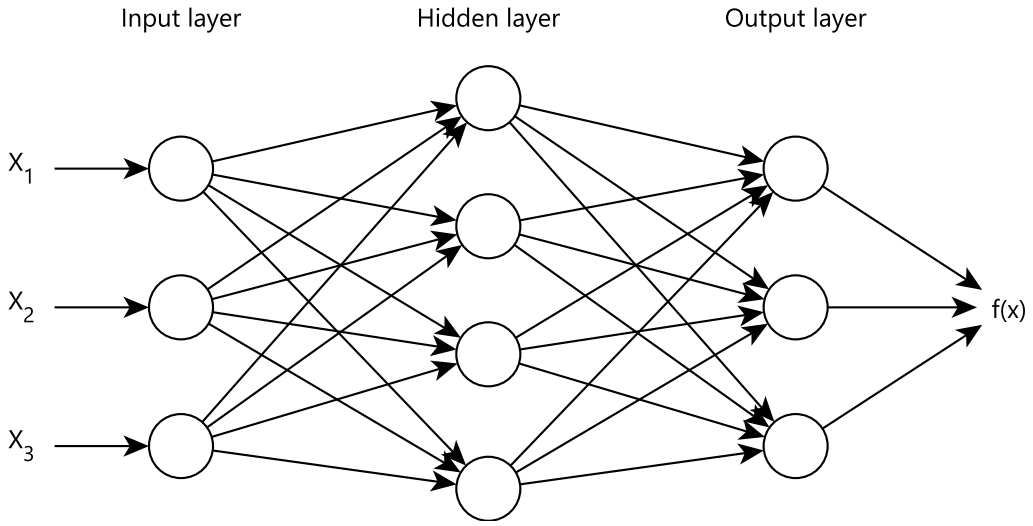

Figure 3.3: Schematic overview of a three-layer artificial neural network.

The input signal X_n can be a set of feature values that is communicated to the input layer. There are $\mathcal{I}^{(1)}$ features as input to the model. Inside a neuron, inputs are multiplied by weights and summed, as shown in the equation below.

$$Y_j^{(2)} = \sum_{i \in \mathcal{I}^{(1)}} w_{ij}^{(1)} X_i^{(1)}, \qquad j \in \mathcal{I}^{(2)}, \tag{3.18}$$

where the input layer produces $J = |\mathcal{I}^{(2)}|$ outputs, $\mathcal{I}^{(1)}$ is the set of inputs in the first layer (feature values) and $Y_j^{(2)}$ are the inputs to a nonlinear activation function that may decrease or magnify the input. The weights $w_{ij}^{(1)}$ for layer (1) are multiplied with the individual neuron input

values $X_j^{(1)}$ (Powell, 2007). An example of such an activation function is a logistics function:

$$\sigma(y) = \frac{1}{1 + e^{-\beta y}}, \tag{3.19}$$

where β is used for scaling. The logistics function $\sigma(y)$ is shown in Figure 3.4.

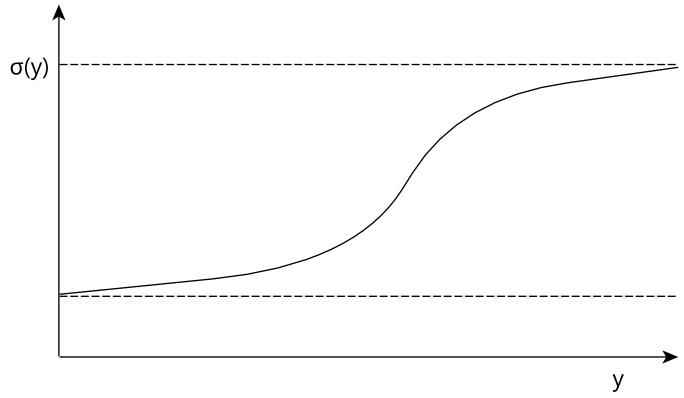

Figure 3.4: Logistics function for nonlinear behavior in artificial neural networks.

This function introduces nonlinearity in the signal to the next neuron and layer. So, the calculation with the logistics function is expressed as:

$$X_i^{(2)} = \sigma(Y_i^{(2)}), \qquad i \in \mathcal{I}^{(2)}. \tag{3.20}$$

$X_i^{(2)}$ is used as input to the second layer. In the second layer, the following computation is done:

$$Y_j^{(3)} = \sum_{i \in \mathcal{I}^{(2)}} w_{ij}^{(2)} X_i^{(2)}, \qquad j \in \mathcal{I}^{(3)}. \tag{3.21}$$

The resulting output is used in the output layer to compute a single output:

$$f = \sum_{i \in \mathcal{I}^{(3)}} w_{ij}^{(3)} X_i^{(3)}. \tag{3.22}$$

The logistics function is only an example of an activation function. Activation functions can take many forms, but are usually logistic, hyperbolic tangent or rectified linear (Haykin, 1998). When training an artificial neural network, the weights w_{ij} are initialized randomly. During the training phase, predictions are done after which errors between predictions and actual values are calculated and used to update the weights to minimize the error. Weights are usually updated until the error drops below a predefined threshold (Aggarwal, 2018). When tuning a neural network, many choices need to be made considering, amongst others, the number of layers, nodes per layer, error-penalties, activation-layer functions, and step-size. This large array of hyperparameters and settings potentially introduce problems with stability (Haykin, 1998; Aggarwal, 2018).

The use of regression models for predicting transportation costs, e.g., travel distance or time, based on customer statistics, i.e., customer location and customer demand, without solving a vehicle routing problem is shown in Nicola et al. (2019). They use, amongst others, features for the number of customers, average distance between customers, average distance from customers to the customer centre of mass (centroid), and the variance of coordinates to describe TSP and VRP instances. They test their linear regression models on Solomon instances (Solomon, 1987), and show that vehicle routing distance can be approximated with high accuracy. They

also present features related to time windows, e.g., sum and variance of time window length and overlap, and show the value of including time window related features in their regression model (Nicola et al., 2019).

Summarizing, we showed different methods to approximate the cost of transportation. First we showed simple heuristic formulas, like Daganzo-approximation (Robuste et al., 1990). Next we showed how seed-based schemes can assist in making better predictions for vehicle routing distance and service time. We showed how linear regression and neural networks work and how they can help to predict routing costs.

3.6 Solution Validation

An often used validation method in operations research, including time slotting research, is simulation (Campbell and Savelsbergh, 2005; Cleophas and Ehmke, 2014; Ehmke and Campbell, 2014; Yang et al., 2016; Klein et al., 2018). Simulation is a technique used to validate (mathematical) models in complex environments that cannot be solved exactly (Law, 2015). A simulation imitates (part of) a system or real-world process over time. A simulation model is often composed of a set of assumptions and abstractions of reality. Simulation models can help to study sub-systems, effects of changes in systems, and the interaction between variables and inputs. Also, simulation models are a device to verify the performance of mathematical models (Banks et al., 2010). However, simulation techniques also have disadvantages. It can be hard to make conclusions about models when randomness causes differences in performance, and the real-world can be too complex to model, causing oversimplified simulation models and less useful results (Pegden et al., 1995).

Discrete-event simulation is a simulation type that only concerns the change of the system model on discrete time steps. These points in time are events that might change the state of the system. A discrete-event simulation model consist of, amongst others, a system state, a simulation clock, events, an initialization routine, and randomness generators (Law, 2015). Because parts of our problem are NP-hard, and our system is highly complex and affected by stochasticity (e.g., customer behavior), we choose to use simulation for validating our solution methods. See Chapter 6 for the description of our simulation model.

3.7 Conclusions and Research Contributions

In this chapter we studied the scientific literature about vehicle routing, attended home delivery and costs approximation. We introduced the VRPTW and showed both an exact formulation and some heuristic approximations. We scoped the AHD-literature to evaluate the different problem characteristics and models used.

AHD is still a developing field of research. In recent years, more and more realistic problem aspects have been considered. However, many studied problems are still abstract and have many assumptions and limitations. Amongst others, limitations are still present in horizon lengths, VRP-characteristics, time slot designs, and geographical data. When real-world data is used, still many abstractions from reality are used considering customer behavior and VRPTW-settings. We found that the time slotting literature is divided into two fields: a field that only looks at time slot allocation methods and a field that looks at time slot incentive methods. Most authors that study incentives focus on the approximation of transportation costs. Most solution methods have as objective to maximize profits, although maximizing the number of booked customers is also used. When using profits as objective, approximations need to be made for both the revenues and the costs. Customer choice models come in many forms, but in AHD mostly the multinomial logit choice model is used, that requires customer behavior data to get a good fit. Early time slotting literature uses probabilistic models and non-parametric rank-based models. We found different options for approximating transportation costs, ranging

from simple rules to regression models. We recognise the following two options for dynamic pricing research: (i) estimate the costs of a time slot and use this as basis for setting time slot prices, or (ii) optimize the time slot prices, such that the behavior of customers is nudged optimally. For our solution method, we focus on approximating the costs of transportation, and combine this with simple incentive models and a novel customer choice model.

The contributions of this thesis to existing scientific literature are (i) the application of regression models for approximating downstream transportation costs instead of the currently in use heuristic methods, (ii) a novel parametric rank-based method for modelling customer behavior that, compared to the currently in use multinomial logit choice model, does not require behavioral data and requires less computations, (iii) the application of our solution approach and customer choice model to a realistic time slotting case using costs approximations and a limited set of basis functions, and (iv) the application of a commercial vehicle routing solver and time slot allocation software to our case.

Chapter 4

Problem Formulation

In this chapter, we give a formal problem formulation of the time slot pricing problem we consider. First, we state the complete problem formulation and introduce all notation in Section 4.1. Next, we introduce a parametric rank-based customer choice model in Section 4.2. In Section 4.3, we describe the method we use for determining per-customer transportation costs. We end this chapter with a summarizing conclusion in Section 4.4.

4.1 Problem Characteristics

In this section, the notation of all variables, parameters and sets is introduced, based on the formulation in Visser et al. (2019). We adhere to the order process as perceived by a customer. This process consists of three steps: (i) customer arrival, (ii) time slot offering, and (iii) time slot selection and confirmation. During a certain period, customers place orders at a retailer, after which the customers are offered a time slot for delivery. As common in these type of problems, we specify this period as $[0, T]$, for which 0 is the first time a customer can place an order and T is the so-called "cutoff time", which is the last moment a customer can place an order. After time T, the final delivery schedule is made for a single day or daypart, by solving a VRPTW using a set of constructive heuristics and metaheuristics. The customer arrival times are unknown upfront. A customer i is part of the set of customers $\mathcal{C}$, i.e., $i \in \mathcal{C}$. Customer arrivals can happen at any time in the horizon $[0, T]$, we indicate the time of arrival of customer i as t_i. Customer orders have a certain size, for example indicated by weight or volume, q_i. The order quantity q_i is also unknown upfront. Each customer has a delivery service duration, i.e., the time it takes for the deliverer, after arrival at the address, to hand over the package. The service duration is indicated with l_i. The expected delivery duration can be estimated with a fixed time component and a variable time component that is dependent on the order quantity q_i.

After the customer arrival, the customer needs to be offered a set of time slots for delivery. We consider a single day of delivery time slots, these are all part of the set $\mathcal{T}$, with the earliest time slot beginning after T. A subset of $\mathcal{T}$ is offered to a customer, depending on feasibility and the offering policy. Time slots in $s \in \mathcal{T}$ can be of differing length and can be overlapping or non-overlapping. The individual time slot duration is denoted with $[a_s, b_s]$.

The calculation time, needed to calculate the feasibility and costs of offering certain time slots, is called the algorithmic time z_i. This time is dependent on the used algorithms and problem size. After the offer is made to the customer i, he/she needs to make a decision, which will take some time d_i. So, at time $t_i + z_i + d_i$, we know what choice the customer made. The set of offered time slots is denoted $\mathcal{S}_i$, so that $\mathcal{S}_i \subseteq \mathcal{T}$. s is a single element in this set, i.e., $s \in \mathcal{S}_i$. Each time slot that is offered, gets a certain incentive to steer customer behaviour. We consider incentives on a continuous scale, part of the incentive set $\mathcal{G}$. The incentives can be dynamically determined, and differ per time slot. The incentive given to a certain time slot

is $\mathcal{G}_s$. The incentives $\mathcal{G}$ are decimal numbers on the domain $[-1, 1]$, with a negative number indicating a penalty, and a positive number indicating an incentive.

During the calculation time z_i, we need to determine (i) which time slots are feasible to offer, concerning both vehicle capacities as well as time window constraints and (ii) we need to determine the costs of offering a certain time slot.

We call the set of customers that accepted a time slot and need to be planned $\mathcal{C}'$. The directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ models the system where vertices $\mathcal{V} = \mathcal{C}' \cup \mathcal{D}$ consist of the set of customers $\mathcal{C}'$ and the set of depots $\mathcal{D}$. Each customer $i \in \mathcal{C}'$ can be served from every depot in the set $\mathcal{D}$. The travel time on edge $(i, j) \in \mathcal{E}$ can be expressed with $\tau_{i,j}$. A single depot $d \in \mathcal{D}$ has a fixed number of vehicles L_d available for delivery. The fleet is homogeneous and a vehicle has a capacity of H. To make a delivery, a vehicle has to visit the vertice along a route. A vehicle route always ends at the same depot it started. For the planning of vehicle routes, we consider three constraining factors: (i) the vehicle capacity H cannot be exceeded, (ii) the vehicle routes must start and finish in the interval $[a_d, b_d]$, dependent of depot d, and (iii) the delivery of customers must be done within their selected time slot. A vehicle can only leave from a customer i after the full service duration l_i.

4.2 Customer Choice Model

For mimicking customer behavior, i.e., reacting on time slot incentives, we develop a rank-based choice model with a utility theory scoring component. We model customer preference as follows: a customer has a ranking for all time slots, i.e., the first preferred time slot is ranked highest and the least preferred time slot is ranked lowest, as is normal for rank-based models (see Section 3.4). The model combines two models found in literature; a rank-based model and a parametric utility theory model. The ranking of time slots is based on scores and therefore the ranking can be influenced by incentives, similar to models based on utility theory, e.g., the multinomial logit model (see Section 3.4).

Each customers gives "base scores" to all time slots, expressed with $K_i \subseteq \mathcal{T}$. For our experiments we use a preference list that entails all time slots, i.e., $|K_i| = |\mathcal{T}|$. We model different types of customers. Some customers can be seen as "rigid", and others are perceived more "sensitive" to incentives. The level of incentive sensitivity per customer is expressed with f_i, which is a continuous parameter on the scale $[0, 1]$, with 0 being rigid and 1 sensitive. The incentive effectiveness is directly related to the incentive sensitivity parameter f_i of a customer. We do not know the customer incentive sensitivity upfront.

We define the number β_s as the base score of a time slot s, with β_s on the domain $[\frac{1}{|K_i|}, 1]$, with $|K_i|$ being the cardinality, the number of time slots in the base preference list of customer i. The assignment of scores to time slots is done in a decreasing fashion, i.e., the first preference gets the highest score (1), and the last preference gets the lowest score ($\frac{1}{|K_i|}$). The lowest possible score is $\frac{1}{|K_i|}$ instead of 0 because this prevents problems when there are only few time slots and the difference in base score is too large for incentives to have any effect. The equation for determining base scores $\beta_{i,s}$ is given by:

$$\beta_{i,s} = \frac{|K_i| - k_{i,s} + 1}{|K_i|}, \tag{4.1}$$

where $\beta_{i,s}$ is the base score for time slot s of customer i, $|K_i|$ is the number of time slots in the preference list of customer i, and $k_{i,s}$ is the randomly drawn ranking of time slot s for customer i, where the ranking is an integer number $k_s \in \{1, 2, \ldots, |K_i|\}$. We can influence the ranking of the base preferences using incentives. The incentive decision needs to be made for all feasible time slots. The incentives we can give are indicated with $\mathcal{G}_s$, and are continuous numbers with $\mathcal{G}_s$ on the domain $[-1, 1]$. A negative incentive can be interpreted as a penalty. The incentives are multiplied with the customer incentive sensitivity f_i, and then added to the

base preference scores. Then, the list is re-ordered from high to low and the customer chooses the highest ranking time slot that is offered, as common for utility theory models. The total score of a time slot for a customer is expressed with U_s and is calculated using Equation 4.2.

$$U_{i,s} = \beta_{i,s} + f_i \cdot \mathcal{G}_s. \tag{4.2}$$

4.3 Determining the Downstream Costs

To obtain the routing costs per customer, we need to do some transformations with routing data. We use a method we call "half-edge partitioning" (HEP) that can be applied to most VRP and VRPTW solutions. HEP, as illustrated in Figure 4.1, is a simple method that allocates half of the time needed to travel an edge to the customer from which the edge departs, and the other half to the customer at which the edge arrives.

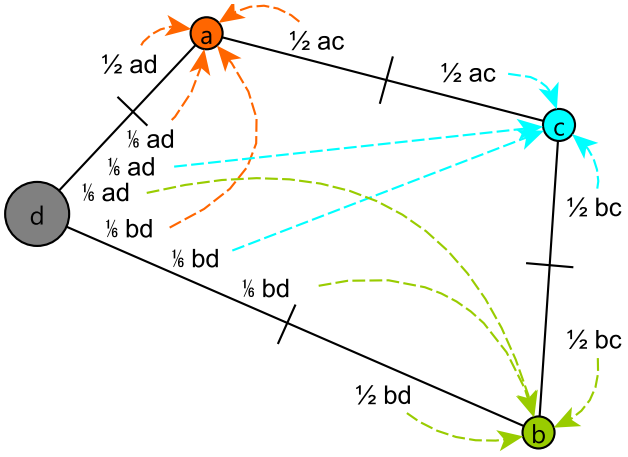

Figure 4.1: Example of cost allocation to customers with half-edge partitioning.

The edges that depart from and arrive to the depot are partially allocated to their arriving and departing customer, respectively. The other half of these depot edges are equally divided over all customers. The routing costs, expressed in travel time, of a single customer c served by a vehicle that serves a set of customers $\mathcal{C}'$, can be expressed with:

$$\text{Travel time of customer } c = \frac{1}{|\mathcal{C}'|}(0.5t_{d,f} + 0.5t_{l,d}) + 0.5t_{i,c} + 0.5t_{c,j}, \tag{4.3}$$

with $t_{i,j}$ being the travel time in the final routing schedule on edge (i, j). The depot is indicated with d, and customer f and customer l are the first and last customer of a vehicle route, respectively. For the example in Figure 4.1, the costs attributed to customer a are: $\frac{1}{3}(0.5t_{a,d} + 0.5t_{b,d}) + 0.5t_{a,d} + 0.5t_{a,c}$ which equals: $\frac{1}{6}t_{a,d} + \frac{1}{6}t_{b,d} + \frac{1}{2}t_{a,d} + \frac{1}{2}t_{a,c}$. The sum of all travel times over all customers and vehicles will again return the total travel time as from the original VRP-schedule. The advantage of HEP is that it uses the actual edge travelling costs, i.e., road network distance and time, and does not neglect the use of multiple vehicles. Also, the method can easily be applied to multi-depot instances.

4.4 Conclusions

In this chapter, the formal problem formulation with all notation was introduced. We explained all decisions that need to be made for a solution method. We proposed a parametric rank-based model that can be used to test time slot pricing policies but does not require customer behavior data. Finally, we showed how we determine the costs per customer using half-edge partitioning.

Chapter 5

Solution Approach

In this chapter, our solution approach is explained. First, we explain our benchmark method for approximating transportation costs using cheapest insertion in Section 5.1. Next, we explain our main method for approximating transportation costs using regression models in Section 5.2. In Section 5.3, we show a simple policy that can be used to translate predicted transportation costs into time slot incentives. We close this chapter with a summarizing conclusion in Section 5.4.

From our review of scientific literature in Chapter 3, we concluded that there are two possible options for a solution method: (i) approximate the costs of a time slot and use this as basis for setting time slot incentives, or (ii) optimize the time slot incentives, such that the behavior of customers is nudged optimally. For our solution approach we focus on the first option, namely the cost approximation. An overview of our solution approach fitted to the current business process as depicted before in Figure 2.1, can be found in Appendix A.

5.1 Insertion Costs as Transportation Costs Approximation

As explained in Chapter 2, cheapest insertion costs is a common method used for time slotting. We apply cheapest insertion as cost approximation method and benchmark for our method explained in Section 5.2. The idea of cheapest insertion is relatively simple; during the booking horizon, we keep track of a preliminary routing schedule that contains all booked customer orders up to the respective moment. This preliminary routing schedule is sequentially constructed using cheapest insertion, and periodically re-optimized (after every 20^{th} customer arrival) using the vehicle routing software in ORS. When a new customer arrives, the cheapest insertion algorithm calculates how much it would cost, in terms of travel time, to add the new customer to a vehicle route. The cheapest insertion algorithm returns the costs of insertion for every feasible time slot. These costs differ per time slot since vehicles that serve customers in the same time window may be close by, or alternatively have to make a detour. Cheapest insertion is simple, fast and dynamic, since it uses all current customer information for estimating costs. The disadvantage is that it is greedy, i.e., it makes the best decision at a point in time but cannot make a forecast about future customers. For a more elaborate explanation of cheapest insertion and its advantages and disadvantages, we refer to Section 2.2.

5.2 Regression Based Transportation Costs Approximation

In this section, we present our solution method. For this method, we determine the costs of a customer insertion, using regression models with a limited set of features. Opposed to insertion costs, we try to estimate the future, downstream costs of a decision. We explain our model structure and features in Section 5.2.1. In Section 5.2.2 and Section 5.2.3 we explain the application of random forests and neural networks, respectively. Finally, we explain how we obtain data for training our model in Section 5.2.4.

5.2.1 Features for Predicting Transportation Costs

In this section, we present several features or basis functions to predict the final routing costs. It might be difficult to predict the routing costs of individual customers because of high variance of customer locations, that is hard to explain using single-location-based features. Ultimately, we are interested in predicting the costs of inserting a customer in a certain *time slot* s. That is why we group customers together and predict the average costs of a group. Based on a collection of historical data, obtained from running multiple simulations with different customers, we can partition the area of operation in smaller spatial areas ($a \in \mathcal{A}$). The exact partitioning of an area of operation can be highly instance specific because of specific attributes, e.g., customer dispersion, natural borders like rivers or forests and abstract borders stemming from legislation or business rules. Therefore, we give a simplified example of an area partition. Figure 5.1 shows an example of historic customer order data. Inside an area, we can aggregate customers based on their time slot choice, $s \in \mathcal{S}$. Figure 5.1 shows that customers chose between two time slots. The area of operation has been divided in four spatial areas.

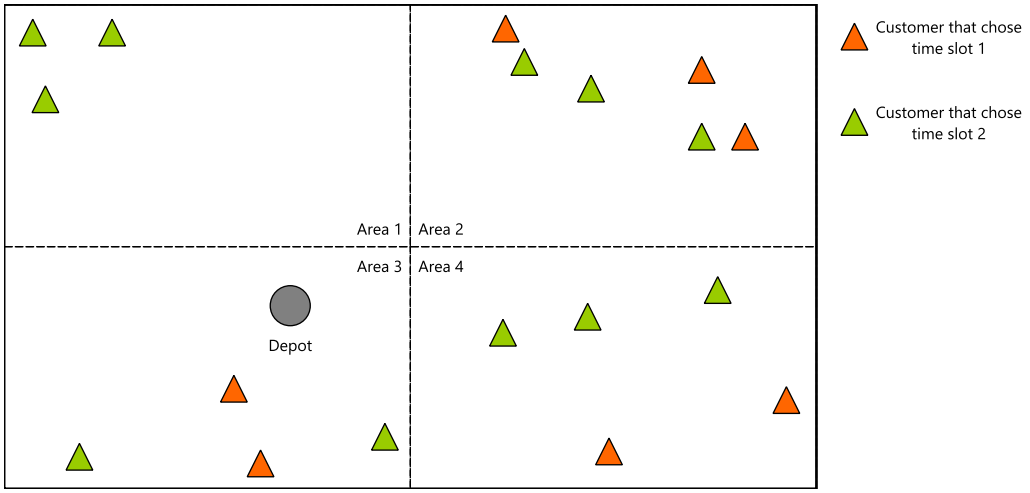

Figure 5.1: Aggregation areas of an historic instance.

For this example, seven area-time slot combinations (ATCs) can be recognised: $\{(1,2),(2,1),(2,2),(3,1),(3,2),(4,1),(4,2)\}$. For each of these pairs, we aggregate customer and routing information. The following information of an ATC is stored: customer locations expressed in latitude and longitude, customer order volume expressed in kilograms, and the routing costs per customer. Aggregation-based features give a synopsis of the characteristics of an *area* and *time slot* cluster (ATC) a customer is in. We use cheapest insertion to check what time slots are feasible to offer. The feasibility check using cheapest insertion is further explained in Section 2.2. For every feasible time slot option, we calculate the feature values *before* and *after* the potential insertion of the new customer, to obtain the expected increase in routing costs. All features are summarized in Table 5.1. First we explain the location features.

We calculate the mean and variance of bearings between customers and the depot. The bearing between the depot d and customer c, $\beta_{d,c}$, is the angle between the straight line connecting the two points and the north-south line of the earth. Figure 5.2 illustrates the feature. The bearing is calculated with the following equations:

$$x = \cos(lat_c) \cdot \sin(\Delta(lon_d, lon_c)), \tag{5.1}$$

$$y = \cos(lat_d) \cdot \sin(lat_c) - \sin(lat_d) \cdot \cos(lat_c) \cdot \cos(\Delta(lon_d, lon_c)), \tag{5.2}$$

$$\beta_{d,c} = \arctan(x, y). \tag{5.3}$$

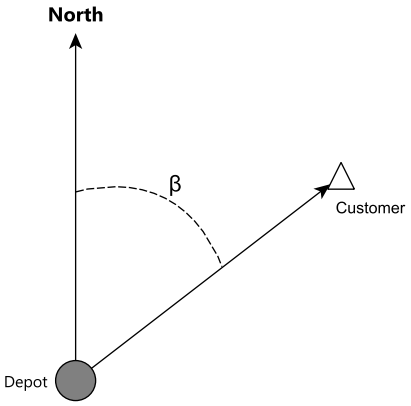

Figure 5.2: The bearing between the depot and a customer.

This feature can be instance specific since the dispersion of customers and the depot location relative to the customer locations vary among different problem instances.

The second location feature is the euclidean distance from the ATC-centroid location to the depot from which the customer is served. This distance is calculated with the *haversine* formula, as based on the law of haversines for determining distances between points on a sphere (Brummelen, 2013). The haversine distance between depot d and customer c is calculated with:

$$\text{Haversine} = 2r \arcsin \sqrt{\sin^2\left(\frac{lat_d - lat_c}{2}\right) + \cos(lat_d)\cos(lat_c) + \sin^2\left(\frac{lon_d - lon_c}{2}\right)}, \quad (5.4)$$

where r is the radius of the sphere, the earth radius is by approximation 6378 kilometers, and *lat* and *lon* are the latitude and longitude of an address, respectively.

Aside from location based features, we use the number of customers in an ATC as feature. The time slot specific features are calculated over the area, but not over specific time slots, i.e., the features summarize information of all time slots in area a. These features are: the number of time slots that have been booked in area a, the distance in hours between the first and last booked time slot in area a, and the variance of the time slot population in area a, e.g., if $TimeSlot1$ has 0 booked customers in area a, $TimeSlot2$ 20 and $TimeSlot3$ 34, the variance is: $\sigma^2 = \frac{(0-\mu)^2+(20-\mu)^2+(34-\mu)^2}{3} \approx 194.7$, with $\mu = 18$. Even though these three features are calculated over a group of time slots, the features are useful to capture differences between time slots, e.g., when a customer is booked in $TimeSlot1$, the new time slot population variance will become 182.9, while the addition of the potential customer to $TimeSlot2$ will result in a new variance of 196.2.

We also define two new categorical features, one indicating the time slot s and the other indicating the area a. Categorical features often need to be transformed before they can enter a regression model. The most common method for transforming is called *dummy encoding* which means that each level of the categorical feature is encoded into a separate binary feature. For example, if we have two time slots and two areas in an instance, we would have the following binary features: $TimeSlot1$, $Timeslot2$, $Area1$, and $Area2$. When a feature is set to "1", it means that the customer falls in the respective ATC. Summarizing, we show the eleven features in Table 5.1, including the partition of data we use for calculating the feature values.

Since we use a forecast of the transportation costs to make time slot incentive decisions, we expect a cyclical effect; the cost approximation affects the time slot incentives, and the given incentives in turn affect the (performance of) the cost approximation. Models like ours influence the decision making and therefore have to be applied in situations that are different compared to the initial training data set.

Table 5.1: Summary of features used for the regression models.

Feature	Feature description	Data partition
Days untill the cutoff time (F1)	The number of days left at the arrival of the customer until the cutoff time	N/A
Number of customers in ATC (F2)	The number of customers accepted in the ATC	ATC
Haversine distance from ATC centroid to depot (F3)	The distance from the centroid of all accepted customers in ATC to the depot	ATC
Average distance between customers in an ATC (F4)	The average distance between all accepted customers in ATC	ATC
Variance customer-depot bearing (F5)	The variance of the bearings between the customers in ATC and the depot	ATC
Average customer-depot bearing (F6)	The mean of the bearings between the customers in ATC and the depot	ATC
Area ID (F7)	Binary vector indicating the area	$\mathcal{A}$
Time slot ID (F8)	Binary vector indicating the time slot	$\mathcal{S}$
Variance of time slot population (F9)	The variance of the number of accepted customers per time slot in area $a \in \mathcal{A}$	$a \in \mathcal{A}$
Time slot distance (F10)	The distance measured in time slots between the first and last populated time slot in area $a \in \mathcal{A}$	$a \in \mathcal{A}$
Number of time slots (F11)	The number of booked time slots in $a \in \mathcal{A}$	$a \in \mathcal{A}$

5.2.2 Random Forests

Because of the transformation of our data using the ATC-structure, the relationship between our feature values and the target becomes non-linear. Therefore, we do not consider the most simple prediction model, linear regression, for this study. Random forests can handle non-linear relationships between features and the target. After a feature importance analysis and the subsequent feature selection using the Boruta algorithm (see Appendix D), we tune our model using exhaustive grid search, i.e., automatically testing many different settings for hyperparameters. Tuning models is intended to improve predictive performance and prevent overfitting to the training data. We tune the hyperparameters as summarized in Table 5.2.

Table 5.2: Summary of random forests hyperparameters.

Hyperparameter	Hyperparameter description
Number of trees	The number of trees in the forests
Split criterion	The function to measure split quality (MAE or MSE)
Max. tree depth	The maximum depth of the tree
Min. samples for splitting	The minimum number of samples required to split a node
Min. samples for terminal node	The minimum number of samples needed to be at the terminal (leaf) node
Max. features for splitting	The maximum number of features considered when making a split

5.2.3 Neural Networks

As a second prediction model, we utilize multi-layer perceptrons, i.e., neural networks. Neural networks might show different predictive performance, and therefore are also tested aside from random forests. Since it is difficult to obtain feature importance scores when using neural networks, we do not consider a feature selection method and always use all features. We can tune the hyperparameters, which are summarized in Table 5.3.

Table 5.3: Summary of neural network hyperparameters.

Hyperparameter	Hyperparameter description
Number of hidden layers	The number of hidden layers in the neural network
Number of nodes per hidden layer	The number of neurons per layer
Activation function	The function that activates the hidden layer(s)
Initial Learning rate	The initial weight updating rate
Learning rate	The method used for updating the learning rate
L_2 penalty	The L_2 regularization penalty

5.2.4 Obtaining Training Data

As further explained in Chapter 6, we use a simulation model to test different models and policies. To train our models, we need to obtain data. We do this by generating a separate set of instances and running full simulations on these. For these training instances, we do not use any nudging policy, i.e., customers choose the offered time slot that has the highest base score β_s.

We obtain the following data after a simulation run: (i) a final VRPTW-schedule, (ii) all customer locations, and (iii) the time slots chosen by customers. We use the final routing schedule to reflect the *downstream* transportation costs, opposed to the greedy insertion costs that only estimates the costs at the time of insertion. In most cases, we use a method where the feature values, calculated over an ATC-cluster, are stored after every new customer booking, i.e., the number of ATC data points available for training is equal to the number of booked customers. As further illustrated in Chapter 7, we can also obtain data using only the final routing schedule, i.e., we have a single data point for each ATC-cluster, that contain all booked customers after the cutoff time.

5.3 Simple Incentive Policy

In this section we present a simple incentive policy that can use time slot cost approximations and a tunable parameter as input, and give an incentive, on the domain $[-1, 1]$ as output. The main idea of this policy is that we give incentives based on the approximated costs associated with a time slot s, relative to the estimated costs of all other time slots $\mathcal{S}$. After obtaining a cost approximation for all feasible time slots, the set $\mathcal{S} \subseteq \mathcal{T}$, we first calculate the mean $\overline{C}_{\mathcal{S}}$ and standard deviation $\sigma_{C_{\mathcal{S}}}$ of the predicted costs over all feasible time slots. Next, we calculate the difference between the predicted costs for time slot s and the mean estimated costs over all time slots $\mathcal{S}$:

$$\widehat{c}_s = -1 \cdot \left(c_s - \overline{C}_{\mathcal{S}}\right). \tag{5.5}$$

We multiply with -1 to give higher incentives to low costing time slots and vice versa. Next, we use a tunable parameter W multiplied with the standard deviation $\sigma_{C_{\mathcal{S}}}$ to control how much standard deviations distance from the mean $\widehat{c}_s$ is considered large, and adjust the magnitude of incentives accordingly. In our experiments we use different levels for the parameter W. In case that $W\sigma_{C_{\mathcal{S}}} \ll \widehat{c}_s$, we cap the incentives to remain in the domain $[-1, 1]$. When the costs for all the time slots are the same, i.e., $\sigma_{C_{\mathcal{S}}} = 0$, no incentives are given:

$$\text{Incentive for time slot s} = \begin{cases} 0, & \text{if } \sigma_{C_{\mathcal{S}}} = 0, \\ -1, & \text{if } \frac{\widehat{c}_s}{W\sigma_{C_{\mathcal{S}}}} \leq -1, \\ \frac{\widehat{c}_s}{W\sigma_{C_{\mathcal{S}}}}, & \text{if } -1 < \frac{\widehat{c}_s}{W\sigma_{C_{\mathcal{S}}}} < 1, \\ 1, & \text{if } \frac{\widehat{c}_s}{W\sigma_{C_{\mathcal{S}}}} \geq 1. \end{cases} \tag{5.6}$$

5.4 Conclusions

In this chapter, we explained our proposed solution approach. First we explained the use of insertion costs as cost approximation. Next, we presented our regression model with a limited set of features based on ATC-clusters. We explained the two regression methods (random forests and neural networks) we use and showed the different hyperparameters that need to be tuned. We explained how we we obtain data to train our models on. Finally, we showed a simple incentive policy that can use time slot cost approximations to determine time slot incentives, in combination with our proposed customer choice model.

Chapter 6

Simulation Model and Experimental Design

In this chapter the validation method, a discrete-event simulator, is described and our experimental design is explained. The simulation model is used to validate our different solution methods and test their settings. First, in Section 6.1 the discrete-event simulation model is described. Next we explain the instance settings in Section 6.2 and experimental design in Section 6.3. Finally, we state a conclusion related to this chapter in Section 6.4.

6.1 Simulation Model Description

We use a simulation model that mimics customer behavior and integrates all ORTEC time slotting services, i.e., OTSS for time slot allocation and ORS for routing. The simulation model is built in C# and maintained by the ORTEC Math Innovation Team. All cost approximation models have been trained using the Python Scikit-learn library (Pedregosa et al., 2011) and are loaded in C# using the ONNX standard artificial intelligence format (Bai et al., 2019). For a detailed description of all events and data in the simulation model, we refer to Appendix B. The simulation model is based on the model as described in Visser et al. (2019).

Some abstractions are made to limit the scope of this research. In general, we decided to simulate a single delivery day, limit customer behavior options, and limit the booking horizon.

The goal of the simulation model is (i) to tune the regression models and find the best settings for hyperparameters, (ii) compare our solution methods, and (iii) give insight in incentives and customer behavior patterns, and their effect on the overall performance. The fleet and depot characteristics, customer order size, delivery location and arrival time are directly derived from retailer data. Customers' behavioral aspects are limited for our simulation study. In reality, retailers offer the possibility to change the basket content (add or remove products) after the time slot booking. This is not possible in our simulation. Also, our customer choice model (see Section 4.2) is limited to specific customer behavior, which deviates from reality.

We define three key moments in the simulation horizon: (i) the start of the booking horizon at $t = 0$, (ii) the end of the booking horizon (cutoff time) at $t = T$ and (iii) the route execution. See Figure 6.1 for a timeline of a simulation run. For our experiments we vary the length of the booking horizon, depending on the retailer data, and we always plan for a single execution day, with a varying time slot structure.

The general event structure of the simulator follows the following events: (i) a customer arrives and requests a time slot offering, (ii) a feasibility check for every time slot is done using cheapest insertion and the feasible time slots are offered to the customer, (iii) the customer chooses a time slot, (iv) the customer choice is recorded in the system. ORS is called after every 20^{th} customer arrival to update the intermediate routing schedule, and after the simulation to obtain the final routing schedule. See Appendix B for the full model description.

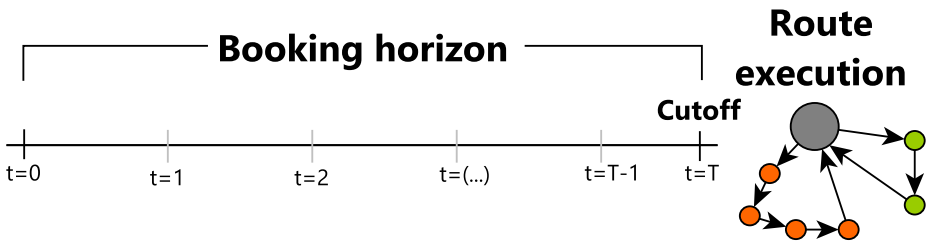

Figure 6.1: Timeline of the simulation booking and route execution period.

6.2 Instance Settings

In this section, we describe the two different instance sets used for our experiments. First, we describe the synthetic case, which is based on generated data, in Section 6.2.1. Next, we describe the case based on an European e-grocery retailer in Section 6.2.2.

6.2.1 Synthetic Instances

The main sets of experiments is conducted on generated data (synthetic case), based on the general structure of a home delivery retailer. For these instances, we use a single depot with a fleet of 20 vehicles. Customer locations are generated using two different methods: random scattering (R) and random clustering (RC). For the random instances, locations are randomly drawn in a radius of 50 kilometers from the depot. For the random clustered instances, customers are assigned to one of eight randomly drawn cluster locations within 50 kilometer of the depot and assigned to a cluster with an 80% chance, or are randomly scattered 50 kilometer of the depot with a 20% chance. The retailer offers six different, non-overlapping time slots of 2-hour width. The intensity of customer arrivals during the booking horizon of 21 days is generated using historic customer arrivals obtained from an European e-grocery retailer. See Figure 6.2 for an example of arrival intensities on a horizon of 21 days with 15 minute bin widths.

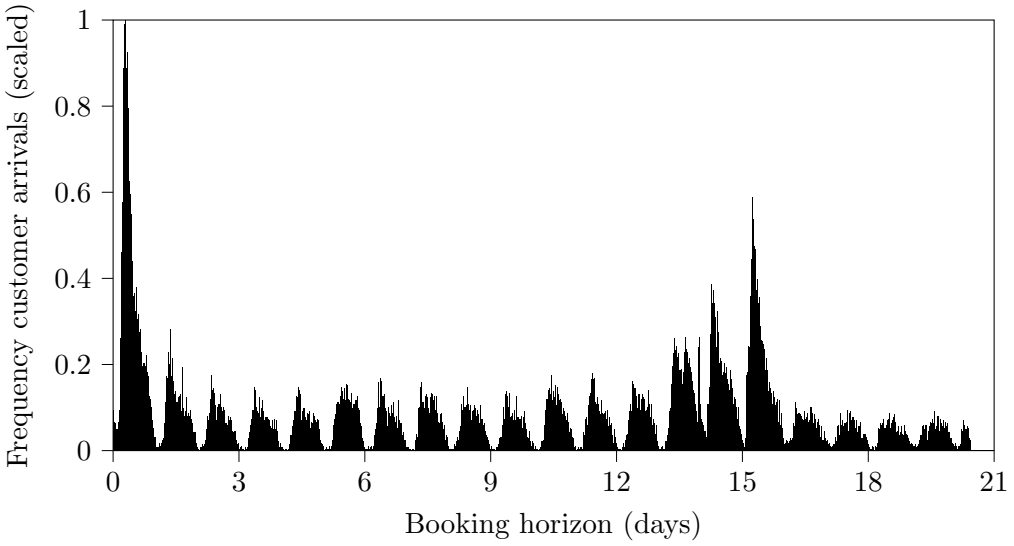

Figure 6.2: Example of a booking horizon with customer arrivals as used for generating instances.

Customers have a base preference list that entails all six time slots, i.e., customers can be nudged to every time slot that is feasible. We use two different settings for fleet capacities: one where there are no constraints on capacity, i.e., the main limiting factor for accepting a customer

are the time slots (T), and a setting where a vehicle has been limited to serve 25 customers (C). For all instances, 750 customers arrive during the 21 days. Table 6.1 summarizes the four different instance types.

Table 6.1: Summary of instance settings for the synthetic case study.

Instance aspect	**R-T**	**R-C**	**RC-T**	**RC-C**
Number of customers	750	750	750	750
Fleet size	20	20	20	20
Vehicle capacity (customers)	N/A	25	N/A	25
Location generation	Random	Random	Random clustered	Random clustered
Number of Clusters	N/A	N/A	8	8
Cluster radius	N/A	N/A	8 km	8 km
Clustering probability	N/A	N/A	0.8	0.8
Number of time slots in base lists K	6	6	6	6
Number of ATC areas a	8	8	8	8
Booking horizon length (days)	21	21	21	21

The area of operation has been divided in 8 area clusters, using a structure where the area is divided in a 2×4 grid. Figure 6.3 shows the operation areas with an example of a customer region structure. For all instances we determine the customer regions upfront and always use the same area of operation for training and testing the models.

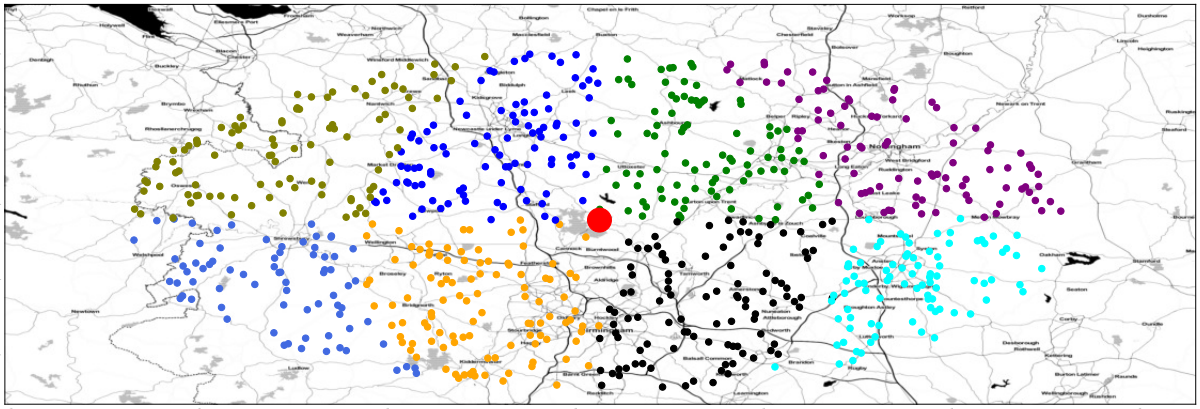

Figure 6.3: Example of the area of operation for the synthetic case with different colors for every customer region, source of background map: maps.stamen.com.

Figure 6.4 shows an example of a vehicle routing schedule after a simulation run on the synthetic case. Road network distances and times are used, but for illustrative reasons the direct arcs are plotted, excluding the depot arcs.

As variant on this larger case, we also conduct experiments with a single vehicle instance that serves 60 customers. This instance, as further explained in Chapter 7, has a similar structure to the synthetic instances, but the area of operation is smaller.

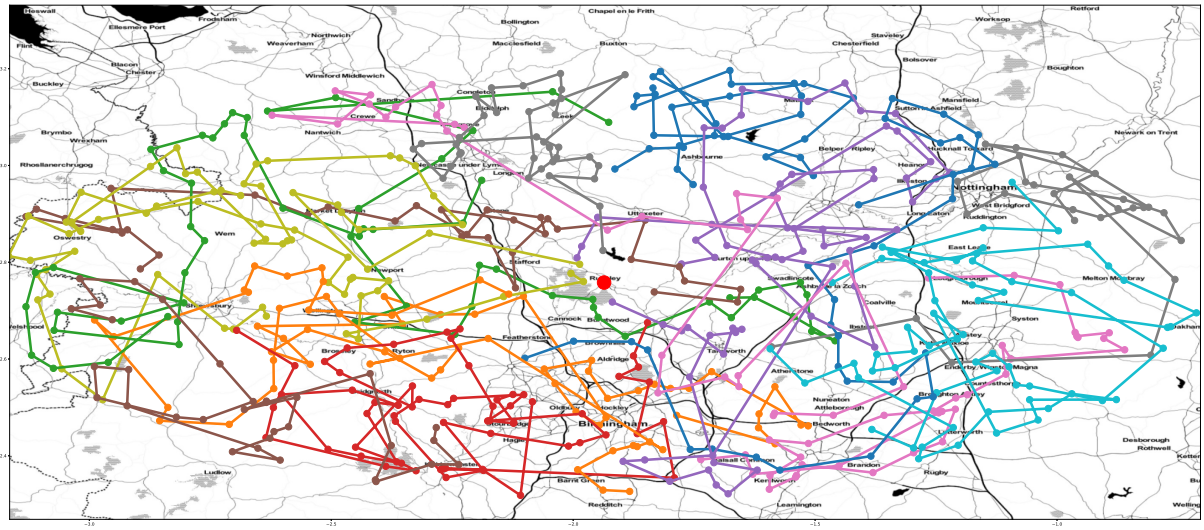

Figure 6.4: Example of a routing schedule for the synthetic time slotting case, source of background map: maps.stamen.com.

6.2.2 European E-grocery Retailer Instances

As an additional case study, we conduct experiments on an actual customer case with data from a large European e-grocery retailer. The main differences with the synthetic case, as summarized in Table 6.2, are that customer locations are no longer generated but are derived from actual customer data, the seven time slots overlap, and there are multiple depots (4) with differing fleet sizes. For the individual instances, we use order data obtained from the same day of the week, to prevent seasonality differences. The fleet is heterogeneous in terms of vehicle capacity and driving speed. The retailer offers five overlapping time slots of 2-hour width, and two time slots of 4 and 5 hour width, respectively. Customers arrive on a booking horizon of 9 days. Because of excessive computational efforts and therefore long simulations, only a limited set of replications will be used for the European e-grocery retailer case.

Table 6.2: Summary of instance settings for the European e-grocery retailer case study.

Instance aspect	European e-grocery retailer
Number of customers	$\sim$ 2000
Fleet size	77
Number of vehicles per depot	(50, 13, 7, 7)
Number of time slots in base lists K	7
Number of ATC areas a	6
Booking horizon length (days)	9

Since some of our features are calculated with the depot location, but we do not know upfront which depot serves a customer area, we always use the main depot for feature value calculations.

6.3 Experimental Design

We can divide our experiments in three parts: (i) test our cost approximation model and the potential of dynamic pricing in combination with our cost approximation model (Section 6.3.1), (ii), show the true effect of dynamic time slot pricing with our cost approximation model and simple incentive policy (Section 6.3.2), and (iii) test our method on real data from the European e-grocery retailer (Section 6.3.3).

6.3.1 Experiment 1

For experiment 1, we start with generating routing and time slotting realization data using our simulation model and the synthetic instances. All simulation runs used for training our models do not use any form of incentives, i.e., customers choose the time slot with the highest base preference. Next, we test different regression models, as introduced in Chapter 5.

We use a particular experimental setting for the second part of experiment 1: we model infinitely sensitive customers, i.e., customers that always choose the time slot we give the highest incentive. For this experiment, we use two different settings: in the first experiment, we offer the cheapest time slot to a customer, and in a second experiment, we offer the most expensive time slot to a customer. This way, we can evaluate the accuracy of the model in predicting the cheapest and most expensive time slots and can show the potential of giving incentives. We compare two methods for approximating the costs of time slots: cheapest insertion based on a preliminary constructed route and our regression model. To give a complete comparison, we use two benchmark methods: first, we show the performance on the same instances without time slots, i.e., a situation where we can serve customers during the whole day, without the time window restriction. Next, we show a benchmark where time slots do not get any incentive, as is also used for obtaining training data, i.e., customers select the offered time slot that is highest on their preference list. We report for four different instance types: (i) randomly scattered with time restriction (R-T), (ii) randomly scattered with capacity restriction (R-C), (iii) randomly clustered with time restriction (RC-T), and (iv) randomly clustered with capacity restriction (RC-C).

6.3.2 Experiment 2

For experiment 2, we test our simple incentive policy (see Section 5.3) on the synthetic case. We use different levels for the tunable parameter ($W \in \{0.5, 1, 1.5, 2\}$) to find the best value. A higher value for W means that the incentive policy is less sensitive for large deviations of the cost prediction. The incentive policy is tested only on the random clustered instances, in both the time constrained and capacity constrained variant (RC-T and RC-C). For the customer incentive sensitivity f, we use a fixed value of 1. By using an incentive sensitivity of 1, we model customers that are always sensitive to incentives, i.e., the incentives always have an effect. However, opposed to the "infinite" sensitivity in experiment 1, the time slot with the highest incentive is not necessarily always chosen, since the base scores, before incentives, have influence on the eventual time slot choice. The goal of this experiment is to test our proposed solution method on a more realistic case where the policy needs to give multiple incentives to time slots and has to identify the relationship between time slots, instead of only finding the cheapest or most expensive time slot as for experiment 1.

6.3.3 Experiment 3

Finally, for experiment 3, we conduct experiments on the European e-grocery retailer case. We first conduct an experiment where the cheapest and most expensive time slot, according to cheapest insertion and our proposed regression model respectively, are nudged with an infinitely

flexible customer, similar to experiment 1. Next, we test our regression model and the cheapest insertion costs using the simple incentive policy, as tested for experiment 2. Again, we use different levels for the tunable parameter ($W \in \{0.5, 1, 1.5, 2\}$) to find the best value. Finally, we propose a cost approximation method that combines our regression model and the insertion costs method and show the performance on the European e-grocery retailer case.

6.4 Conclusions

In this chapter, we introduced our solution validation method, namely a discrete-event simulation model. We explained the settings for the two different instance types, i.e., the synthetic case and the European e-grocery retail case, we use for our experiments. Finally, we discussed the three experiments we conduct, aimed at testing our regression model and obtaining insights about dynamic pricing for time slotting.

Chapter 7

Computational Experiments and Results

In this chapter, the results of our three experiments are presented. In Section 7.1, we show the design and performance of several variants of our regression model on validation data, i.e., without simulation validation. In Section 7.2, we show the performance of cheapest insertion and the regression models on several simulation runs, in the extreme situation with infinitely flexible customers (experiment 1). Also, we show a method for improving approximation models. Next, in Section 7.3, we show the performance of cheapest insertion and the regression model on a more realistic situation with our simple incentive policy (experiment 2). We show the performance of the approximation models on the European e-grocery retailer case in Section 7.4 (experiment 3). Based on our findings, we conduct an additional experiment. In Section 7.5, we use a cost approximation policy that combines our regression model with the insertion costs. We close this chapter with a summarizing conclusion in Section 7.6.

7.1 Routing Costs Approximations

In this section, we test our regression models on the validation data, i.e., data that is left out before training the models. This section is structured as follows: first, we test our model on the final situation in Section 7.1.1, i.e., we train and validate on the situation as observed after cutoff time T when all customers have arrived. Although this situation is unrealistic since in reality the model needs to make predictions with fewer customers, it gives indication of the predictive potential of the model. Next, in Section 7.1.2, we train and evaluate on the intermediate situation, during a simulation run, i.e., the situation as observed between time 0 and cutoff time T, when only a subset of customers have arrived. We report the adjusted R^2, which is a statistic that indicates the goodness-of-fit of a model and indicates overfitting. The R^2 is adjusted for the number of features in a model. Next, we report the relative mean absolute error (rMAE) and relative root mean squared error (rRMSE), as calculated by:

$$\text{rMAE} = \frac{\frac{1}{N}\sum_{i=1}^{N}|Predicted_i - Actual_i|}{\overline{Actual}} \cdot 100\%, \tag{7.1}$$

$$\text{rRMSE} = \frac{\sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}}{\overline{Actual}} \cdot 100\%. \tag{7.2}$$

7.1.1 Regression Models for Cutoff Observations

In this section, we subsequently evaluate the random forests regression (RFR) and neural networks (NN) models on the synthetic case in both the randomly scattered situation (R) as the

randomly clustered setting (RC), see Section 6.2. All the hyperparameters are tuned using 5-fold cross validation grid search and summarized in Appendix C. The training data after the cutoff contains less features, because the static situation causes some features to have lower value, e.g., the number of time slots that are booked in an area $a \in \mathcal{A}$ is always the same, since all time slots are always booked in all areas.

Table 7.1 shows the performance of our model on the training set (in sample) and validation set (out of sample) for randomly scattered instances (R) of the synthetic case. The R^2 is relatively high on both the training and validation data set. The relative MAE and RMSE on the validation set is small enough for our purposes, both for random forests as neural networks.

Table 7.1: Model performance on cutoff observations for the randomly scattered synthetic instances (R).

Statistic	**RFR (i)**	**NN**
Number of Features	8	8
Adjusted R^2 (in sample)	0.988	0.957
rMAE (in sample)	2.5%	4.8%
rRMSE (in sample)	3.2%	6.2%
Adjusted R^2 (out of sample)	0.888	0.868
rMAE (out of sample)	7.6%	8.1%
rRMSE (out of sample)	9.7%	10.6%

The performance of our models on the randomly scattered instances (RC) is shown in Table 7.2. The performance is similar or slightly higher compared to the performance on the (R) instances. Again, we do not observe overfitting and the rMAE and rRMSE are small enough for our purposes.

Table 7.2: Model performance on cutoff observations for the randomly clustered synthetic instances (RC).

Statistic	**RFR (i)**	**NN**
Number of Features	8	8
Adjusted R^2 (in sample)	0.987	0.943
rMAE (in sample)	2.5%	5.1%
rRMSE (in sample)	3.2%	6.6%
Adjusted R^2 (out of sample)	0.907	0.853
rMAE (out of sample)	6.6%	7.9%
rRMSE (out of sample)	8.3%	10.5%

Figure 7.1 shows the actual driving time for serving all customers in an ATC compared with the predicted driving time, predicted on the validation set. For every prediction, we show a confidence interval that indicates how certain we can be for a prediction to be correct by estimating the standard error of a prediction using bootstrap estimates. This estimate of the standard error is calculated with a variance estimate, called the unbiased infinitesimal jackknife:

$$\widehat{V}_{IJ}^{\infty} = \sum_{i=1}^{N} \mathrm{Cov}_*\big(N_i^*, t^*(x)\big)^2, \tag{7.3}$$

where we calculate the covariance between N_i^*, the number of times the i^{th} training example

appears in a bootstrap sample, and $t^*(x)$, the random forests prediction (Efron, 2014). So, the error term gives us an idea on how much a prediction would change if the model were trained on a different data set. More information about the calculation of the confidence intervals for random forests can be found in Wager et al. (2014) and Polimis et al. (2017). Both the model for the random (R) instances as the randomly scattered (RC) instances show good predictive performance, i.e., following the diagonal prediction-observation line well. The confidence intervals for the RC instances are somewhat larger than for the R instances, so the model is less confident about its prediction. The case where the error bars do not cross the diagonal prediction-observation line indicates there is residual noise in the data that cannot be explained by the random forests (Wager et al., 2014).

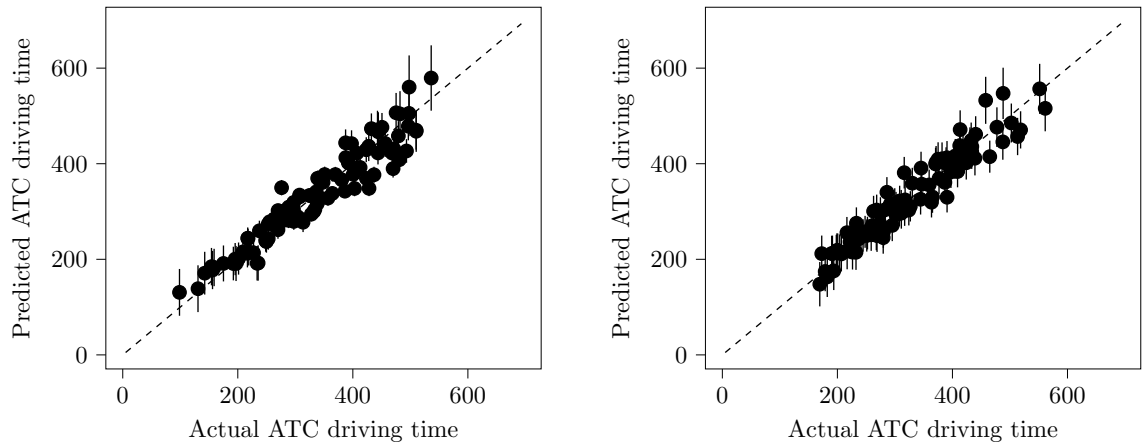

Figure 7.1: Unbiased standard errors for the R (left) instances and RC (right) instances on the cutoff time data for the ATC driving time in minutes, predicted with random forests.

7.1.2 Regression Models for Intermediate Observations

In this section, we again consider the R and RC instances of the synthetic case and show performance of both random forests (RFR) and neural networks (NN). However, now we consider a data set obtained during the booking horizon $[0, T]$. In this situation, sometimes only a subset of the complete set of customers has arrived. Still, we train to predict the final routing costs. This way of evaluating allows us to show the performance of our model as it would be during a simulation run, when indeed the ATC-clusters still contain only few (or no) customers. We store feature values of an ATC-cluster, used for training our model, after every new arrival of a customer, i.e., the number of data points we can train or validate on is equal to the number of customers that were booked. Therefore, the same downstream routing costs for an ATC-cluster are associated with different feature values, since more customers are added to an ATC during the booking horizon. This effect is illustrated in Figure 7.2.

We deal with this problem by splitting up the training data in time aggregations and training separate models for each time aggregation. The booking horizon for the synthetic instances (see Section 6.2) is 21 days. Because we need enough data to train on, we decide to split up the training data in three separate sets, i.e., each time aggregation lasting 7 days. Figure 7.3 summarizes the data sets we train our different models on. RFR (i) is the model trained on the final routing schedule, RFR (ii) is trained on observations from the complete booking horizon, where we store the feature values after every new customer booking, and RFR (iii) is trained on partitions of the booking horizon, again with data obtained by storing feature values after every new customer booking.

We compare the average performance of the model trained on all 3 weeks combined (RFR

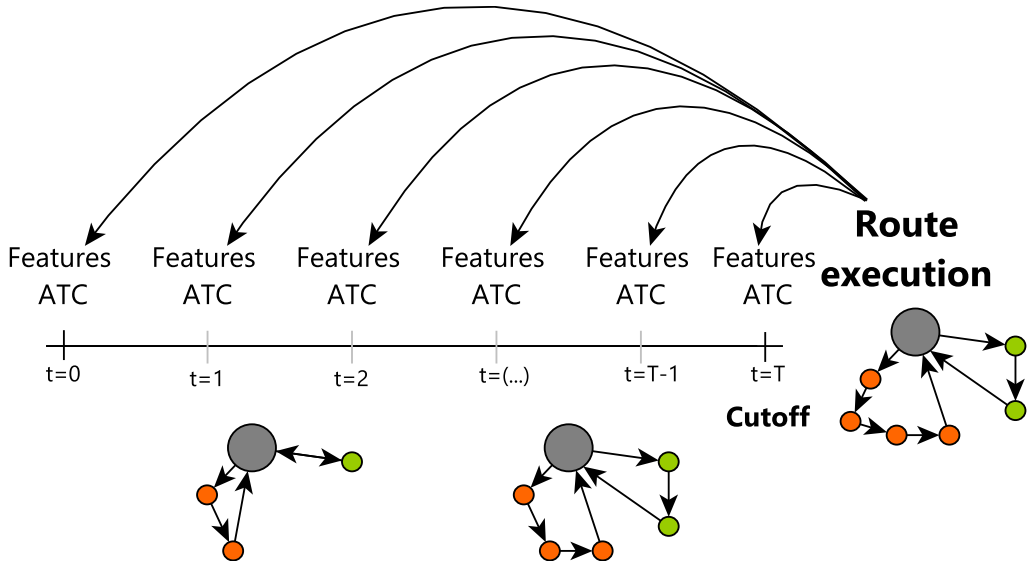

Figure 7.2: Timeline illustrating the problem of associating final routing costs with different feature values.

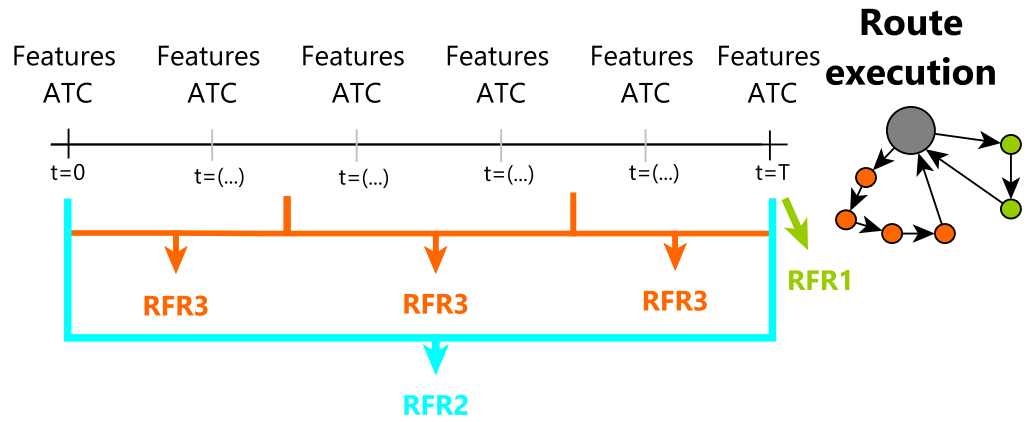

Figure 7.3: Structure for partitioning the booking horizon to obtain data from a simulation run for the three different random forests regression (RFR) models.

(ii)) with the separate 7-days model (RFR (iii)). Table 7.3 shows the performance on the R instances. We observe a large drop in performance for all models in comparison with Section 7.1.1. The in sample performance is still relatively good, with the random forests outperforming neural networks. This might be caused by the limited size of the neural network and the used L_2 penalty for preventing overfitting, see Appendix C. Nevertheless, the performance of neural networks on the validation data (out of sample) is similar to random forests. When comparing the models trained on the complete horizon with the models trained on partitions of the horizon, we observe small differences, slightly favoring the model trained on partitions of the horizon (RFR (iii)).

Table 7.4 shows the performance of the models on RC instances. The performance is similar to the models trained and validated on R instances. We observe that the RFR (ii) model performance on in sample data is similar to the out of sample data. RFR (iii) however, does show a large drop in performance when tested on out of sample data. This might be caused by the lower amount of data that is used for RFR (iii), since the data is split in three partitions, which causes an overfitted model.

Figure 7.4 shows the unbiased standard errors for the two proposed models on the R in-

Table 7.3: Model performance on intermediate observations for the randomly scattered synthetic instances (R).

Statistic	RFR (ii)	RFR (iii) (3 model avg.)	NN	NN (3 model avg.)
Number of Features	11	11	11	11
Adjusted R^2 (in sample)	0.723	0.901	0.71	0.691
rMAE (in sample)	13.9%	5.9%	18.1%	17.2%
rRMSE (in sample)	16.2%	7.6%	22.3%	21.2%
Adjusted R^2 (out of sample)	0.576	0.512	0.523	0.530
rMAE (out of sample)	16.1%	14.5%	18.6%	17.1%
rRMSE (out of sample)	21.3%	19.5%	24.1%	21.5%

Table 7.4: Model performance on intermediate observations for the randomly clustered synthetic instances (RC).

Statistic	RFR (ii)	RFR (iii) (3 model avg.)	NN	NN (3 model avg.)
Number of Features	11	11	11	11
Adjusted R^2 (in sample)	0.623	0.898	0.67	0.673
rMAE (in sample)	12.9%	6.0%	18.8%	17.6%
rRMSE (in sample)	16.6%	7.9%	24.5%	22.9%
Adjusted R^2 (out of sample)	0.638	0.593	0.592	0.530
rMAE (out of sample)	17.5%	14.8%	19.1%	17.5%
rRMSE (out of sample)	22.4%	19.7%	24.6%	22.4%

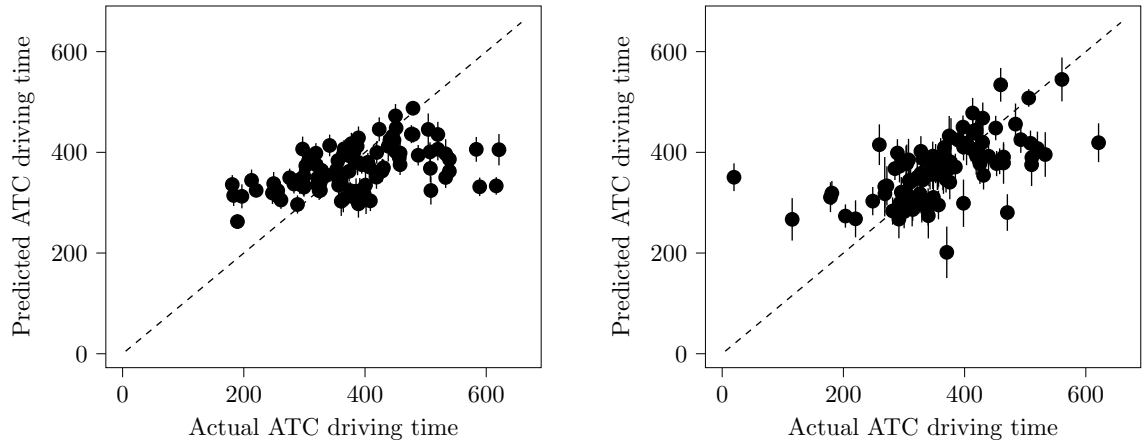

Figure 7.4: Unbiased standard errors for the R instances on the intermediate observations for the ATC driving time in minutes, predicted with random forests using a single model (left) and separate models for time intervals (right).

stances. On the left, the performance of the random forests trained on the complete data set is shown. We observe that the random forests has large prediction errors. Also, the error bars indicate that the model is reasonably confident about its prediction, disregarding some outliers. This is caused by the effect we discussed before and illustrated in Figure 7.2, the way we train

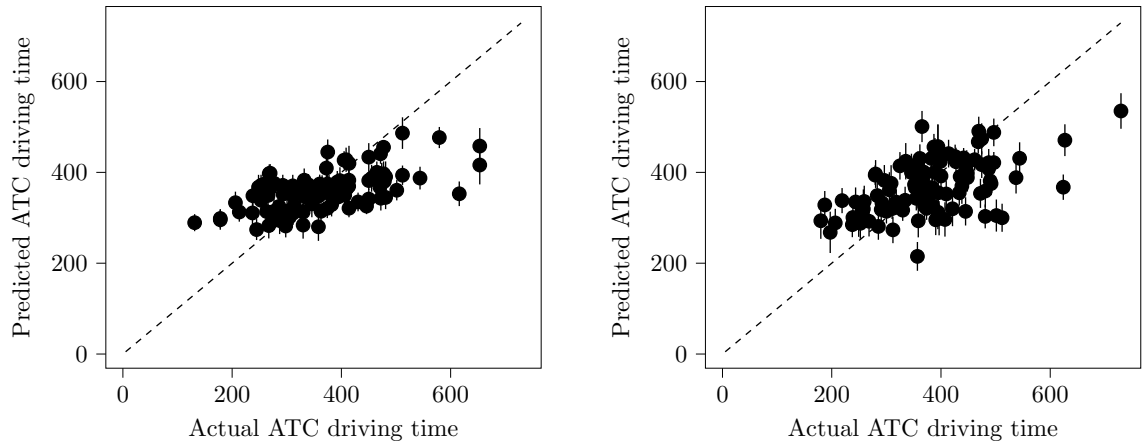

Figure 7.5: Unbiased standard errors for the RC instances on the intermediate observations for an ATC in minutes, predicted with random forests using a single model (left) and separate models for time intervals (right).

our data causes much residual noise that cannot be explained by the model. The model performance shown on the right, the model trained on partitions of the horizon, seems to follow the data somewhat better. For the RC instance type we see similar results, as depicted in Figure 7.5.

We can show the contribution of individual features by calculating feature importance scores. We show this analysis using the Boruta algorithm in Appendix D.

Summarizing, we showed three different ways to train our model: on cutoff observations (RFR (i)), on data obtained during the complete booking horizon (RFR (ii)), and with different models for partitions of the booking horizon (RFR (iii)), see Table 7.5. We showed that random forests and neural networks have a similar performance. To ease the presentation, we decide to further only report the performance of the random forests model.

Table 7.5: Summary of random forests regression models.

Model	Description	Data for training
RFR (i)	Random forests model only trained on the data observed after the cutoff time	ATC-clusters obtained from the final routing schedule
RFR (ii)	Random forests model trained on the data observed during the simulation horizon $[0, T]$	ATC-clusters observed after every new customer booking
RFR (iii)	Separate random forests models trained on data as observed during partitions of the booking horizon $[0, T]$	ATC-clusters observed after every new customer booking, partitioned in three data sets based on time of arrival

7.2 The Potential of Time Slot Pricing

In this section we test our regression models during a simulation run. This section is related to the second part of experiment 1 (see Section 6.3). First, we test our models on the synthetic case in Section 7.2.1, next we show the performance of our model on a special variant of the

synthetic case with only a single vehicle in Section 7.2.2. Finally, in Section 7.2.3 we show a methodology for improving approximation models in an iterative fashion.

7.2.1 Synthetic Case Experiments

Because the results for the randomly scattered instances (R) are similar to the randomly clustered instances (RC), we report the results for the randomly scattered instances (R) in Appendix E, for the R-T setting in Table E.1 and for the R-C setting in Table E.2. We report seven different statistics: (i) the percentage of all customers that could be planned and served, (ii) the average number of time slots that were feasible to offer to a customer, (iii), the percentage of customers that were nudged to a different time slot than their first preference, (iv) the average absolute difference in ranking between the time slot that was chosen and the time slot that was first on the customer preference list, (v) the average travel time per customer in minutes, (vi) the average waiting time per customer in minutes, and (vii) the average travel distance per customer in kilometers. Both the travel time and travel distance are calculated with actual road network costs, as obtained from ORS. Traffic congestion has not been accounted for in calculating travel times. For all statistics, we show the average and the standard deviation (±) over 5 replications. Waiting time is reported because it is an essential element of VRPTWs: potentially, driving times can be low, but the driver has to wait because the arrival at the customer is too early.

Table 7.6 shows the results for the RC instance in a time constrained setting. First, we observe a large difference between the results for a setting with and without time slots. The addition of time slots, disregarding incentives, causes a decrease of the number of served customers of 19.9% and an increase in travel time, waiting time and distance per customer of 93.6%, 89.5% and 52.5%, respectively.

Table 7.6: Simulation run statistics on the randomly clustered instances with a time restriction (RC-T), using 5 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No TS	100% ±0%	N/A	N/A	N/A	9.02 ±0.03	0.38 ±0.03	6.94 ±0.06
No incentive	80.1% ±0.5%	4.3 ±0.04	N/A	N/A	17.46 ±0.2	0.72 ±0.06	14.62 ±0.2
IC Cheap	84.9% ±1.0%	4.8 ±0.06	80.1% ±1.5%	2.5 ±0.09	15.70 ±0.4	0.65 ±0.1	13.15 ±0.4
IC Expensive	76.6% ±1.0%	3.4 ±0.16	83.0% ±0.4%	2.6 ±0.03	18.80 ±0.4	0.69 ±0.02	15.90 ±0.4
RFR (i) Cheap	79.9% ±1.1%	3.7 ±0.03	73.8% ±1.6%	1.4 ±0.06	17.44 ±0.37	0.59 ±0.04	14.69 ±0.44
RFR (i) Expensive	78.6% ±1.1%	3.1 ±0.03	76.0% ±1.3%	2.0 ±0.06	17.79 ±0.31	0.56 ±0.03	15.23 ±0.36
RFR (ii) Cheap	79.9% ±1%	3.4 ±0.1	75.9% ±2.4%	1.6 ±0.05	17.47 ±0.31	0.60 ±0.06	14.77 ±0.31
RFR (ii) Expensive	79.4% ±0.9%	3.4 ±0.05	74.6% ±1.2%	1.7 ±0.05	17.65 ±0.29	0.57 ±0.05	15.05 ±0.21
RFR (iii) Cheap	84.2% ±1.0%	4.8 ±0.03	78.4% ±1.2%	2.4 ±0.07	15.68 ±0.3	0.64 ±0.04	14.25 ±0.4
RFR (iii) Expensive	66.2% ±0.9%	4.7 ±0.03	80.3% ±1.2%	2.5 ±0.07	18.01 ±0.3	0.71 ±0.08	15.99 ±0.4

The performance of the insertion costs (IC) show us that the addition of an incentive strategy has added value. Compared with the strategy without incentives, we see that 6% more

customers could be served and the travel time and distance per customer decreased with 11.2%. Comparing the different incentive strategies, we observe that RFR (i) and RFR (ii) show similar performance, which is structurally lower compared to the incentive strategy based on the cheapest insertion costs (IC). The number of planned customer and the average number of feasible time slots is somewhat lower compared to the cheapest insertion costs and do not show any improvement over a strategy without incentives. The same goes for the travel time and distance. Remarkably, the percentage of customers that were nudged to a different time slot than their first preference is significantly lower for the RFR models compared with the cheapest insertion costs model. It seems that although the RFR models yield a more reliable estimation of routing costs, the cheapest insertion costs are more extreme and result in a more aggressive incentive strategy. Also, the difference between the nudging of a cheap time slot compared with nudging an expensive time slot is low, indicating that the random forests model prediction is off. The RFR (iii) model does show a large difference between the cheap and expensive model, with a similar percentage of booked customers and operating costs compared to the IC-based model.

Figure 7.6 shows the prediction errors of the three models over the booking horizon. A negative error indicates a too low prediction, a positive error a too high prediction. The errors are robustly scaled to better compare outcomes. The shaded area shows the actual errors, the line is the moving average (window = 5). We see that all models move around the zero error line. At the begin of the horizon, RFR (i) shows high errors, probably caused by the structure of the training data that does not contain empty ATC-clusters. RFR (ii) and RFR (iii) seem to have comparable performance, but in the final week before the cutoff time, RFR (iii) performance drops and shows large outlying errors, after which it improves again towards the cutoff time. This drop in performance is possibly caused by the increase of deviation and noise in the third week training data, caused by the effect of an (almost) full schedule, inducing more customer rejections.

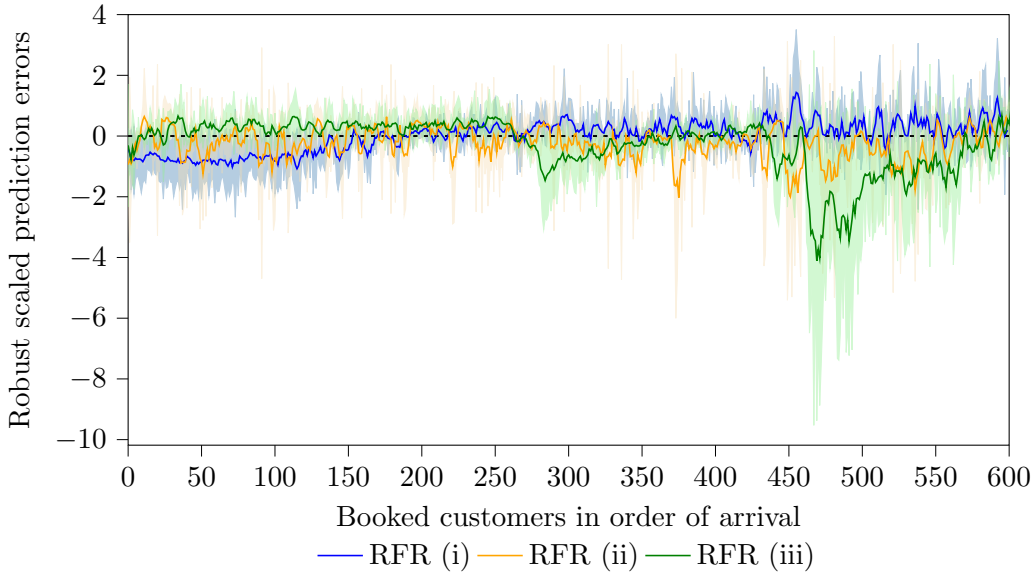

Figure 7.6: Robust scaled prediction errors for a single replication of the three RFR models on the RC-T case.

Table 7.7 shows the performance of the models on the capacity constrained instances. Since the fleet capacity for this instance type is intentionally set lower than there is capacity with solely time window restrictions, we see that all models have a similar number of planned customers, i.e., the main difference with the time constrained instance is that the number of customers that can be served is lower. We observe that the random forests models now show higher performance

Table 7.7: Simulation run statistics on the randomly clustered instances with a capacity restriction (RC-C), using 5 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No TS	66.7% ±0%	N/A	N/A	N/A	12.0 ±0.09	0.56 ±0.07	9.66 ±0.14
No incentive	66.6% ±0.1%	3.8 ±0.01	N/A	N/A	21.74 ±0.14	3.31 ±0.26	16.28 ±0.18
IC Cheap	65.9% ±0.4%	3.9 ±0.01	81.3% ±1.8%	2.5 ±0.1	21.81 ±0.28	1.81 ±0.41	15.36 ±0.46
IC Expensive	66.3% ±0.3%	3.2 ±0.1	82.5% ±0.8%	2.5 ±0.1	21.48 ±0.25	4.37 ±0.21	17.52 ±0.39
RFR (i) Cheap	66.6% ±0.1%	3.4 ±0.03	72.3% ±2.5%	1.2 ±0.06	20.72 ±0.18	1.06 ±0.33	16.76 ±0.37
RFR (i) Expensive	66.6% ±0.1%	2.9 ±0.04	73.3% ±0.8%	2.0 ±0.08	19.70 ±0.30%	1.86 ±0.22	16.65 ±0.48
RFR (ii) Cheap	66.6% ±0.1%	3.1 ±0.06	76.9% ±2.2%	1.6 ±0.05	20.85 ±0.31%	1.15 ±0.07	16.88 ±0.14
RFR (ii) Expensive	66.6% ±0.1%	3.1 ±0.04	74.4% ±0.9%	1.5 ±0.02	19.98 ±0.15%	2.01 ±0.41	16.71 ±0.20
RFR (iii) Cheap	66.6% ±0.1%	3.2 ±0.03	79.4% ±1.2%	2.5 ±0.07	24.78 ±0.3	1.01 ±0.31	15.37 ±0.4
RFR (iii) Expensive	66.6% ±0.1%	3.1 ±0.03	81.3% ±1.2%	2.4 ±0.07	24.83 ±0.3	1.11 ±0.37	15.93 ±0.4

compared to the cheapest insertion costs method. We observe that RFR (i) and RFR (ii) now perform significantly better compared to IC and RFR (iii). Remarkably, we observe that the travel times for all models that nudge the cheapest time slot are higher than for nudging the most expensive time slot, which again is compensated by lower waiting times. The average distance travelled per customer is lowest for the IC and RFR (iii) model.

7.2.2 Single Vehicle Instance Experiments

From the previous experiments, we notice that our proposed solution method performs considerably better on the capacity constrained case.

Table 7.8: Simulation run statistics on the single vehicle, randomly scattered instances with a time restriction (R-T), using 10 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No incentive	37.3% ±3.9%	1.6 ±0.3	N/A	N/A	26.47 ±3.4	1.96 ±1.2	22.25 ±3.6
IC Cheap	38% ±1.3%	1.3 ±0.07	73.3% ±0.05%	2.6 ±0.2	25.30 ±0.7	1.68 ±0.3	21.63 ±0.9
IC Expensive	31.7% ±2.4%	1.4 ±0.2	72.2% ±0.1%	2.5 ±0.2	32.70 ±3.1	2.30 ±0.6	29.19 ±3.35
RFR (ii) Cheap	39.7% ±1.9%	1.7 ±0.2	77.2% ±0.09%	2.3 ±0.2	23.58 ±0.79	1.30 ±0.7	20.27 ±0.84
RFR (ii) Expensive	37% ±2.4%	1.7 ±0.1	72.2% ±0.09%	2.3 ±0.7	26.43 ±3.85	1.53 ±0.9	21.72 ±3.84

This might has to do with the following: when fewer customers can be served but the area of operation is the same, distances and travel times between customers will be higher. The impact of a customer choosing a time slot might be larger when at that time of the operation no vehicle is close by, and a large detour will be needed. To review this observation somewhat further, we generate new synthetic instances where the fleet size is reduced to a single vehicle, the area of operation is reduced by 50% and the number of arriving customers is reduced to 60. We only run a single instance type, namely the random scattered instance with a time restriction (R-T). Table 7.8 shows the results over 10 replications. We only show results for the RFR (ii) model. We observe that the RFR model now significantly improves in terms of travel times, compared with the situation without incentives (12.3%) and the IC model (7.3%). The same results are reflected by the waiting time and traveled distance.

7.2.3 Iterative Predictive Model Improvement

Since our models are all trained on a training data set where no time slot nudging is done, the models are fitted to a different situation than the situation we simulate and test on. So, iterative retraining the model on the new obtained data, with nudging, might be an opportunity to improve the (predictive) performance of our models. This process of improving a model using new data is illustrated in Figure 7.7.

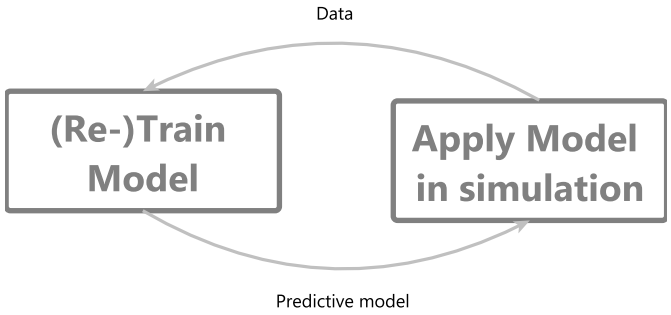

Figure 7.7: The process of obtaining data, training a predictive model, applying it in a simulation, and retraining it using the simulation data.

We retrain the RFR (ii) model on the data obtained from the simulation, and test it again on new generated instances of the random clustered, time constrained (RC-T) case, since on this case the RFR model performance was the lowest. We observe an improvement in predictive performance after the first iteration of the procedure as depicted in Figure 7.7.

This is reflected in an increased adjusted R^2 and a prediction that follows the actual costs somewhat better, as illustrated in Figure 7.8. Figure 7.9 shows the percentage of customers that were planned (left) and the summed travel time and waiting time (right) among two iterations of the feedback loop, for both the situation where the cheapest and most expensive time slot is nudged with ultimately flexible customers. We observe that after retraining the model, the predictive performance increases, since the model that nudges the cheapest time slot can plan more customers and needs less operating time, while the model that nudges the most expensive time slot shows an opposite effect. After a single iteration of the feedback loop shown in Figure 7.7, it seems that the model has converged. Nevertheless, the performance of this improved model is still lower compared to the model based on cheapest insertion costs.

We conducted an additional experiment where the random forests models are trained on the cheapest possible situation, without time slots (No TS). For more information about the training procedure and the results of this experiment, we refer to Appendix E.

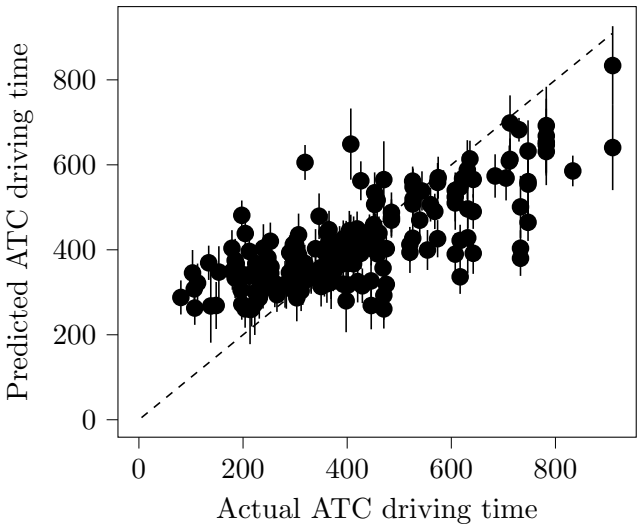

Figure 7.8: Unbiased standard error for the RC instances on the intermediate observations for an ATC in minutes, predicted with random forests trained after the first iteration of the feedback loop.

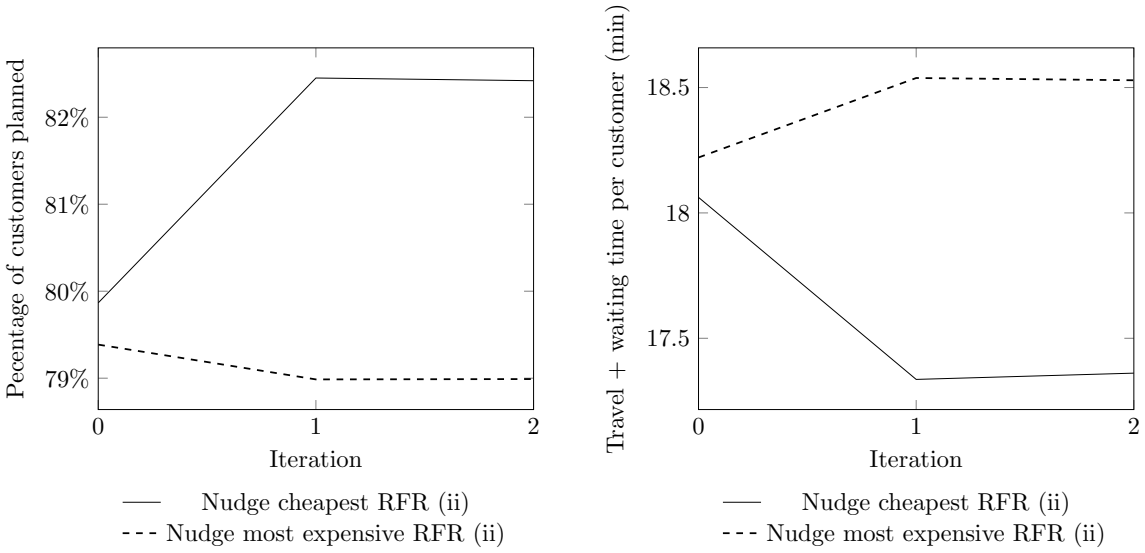

Figure 7.9: Performance of random forests (RFR (ii)) over 2 iterations of the feedback loop using 5 replications on the RC-T instances.

7.3 Time Slot Pricing Policy Experiments

In this section, we test our cost approximation model for a more realistic case where multiple time slots can be nudged. We test our RFR model using the simple incentive policy as explained in Section 5.3 and compare it with the cheapest insertion costs as cost approximation.

We tune our incentive policy with different levels of the parameter W. A higher value for W means that the incentive policy is less sensitive for large deviations of the cost prediction, i.e., the incentive policy is less aggressive. We only show results of the RFR (ii) policy on the RC time constrained and capacity constrained cases, since this model seemed to perform best and is easiest to train. We trained the model on data obtained from the situation without time slot incentives.

Table 7.9 shows the results for the time constrained case. We observe that the performance for both IC and RFR is roughly in-between the performance of nudging the cheapest and most

expensive time slot, respectively. Again, IC shows better performance than RFR, looking at the percentage of planned customers, operating time and travel distance. For both IC and RFR the best value for W seems to be 1. The most remarkable difference between the results of experiment 1 is the percentage of nudged customers.

Table 7.9: Simulation run statistics for the incentive policy for the randomly clustered instances with a time restriction (RC-T) for different levels of tuning parameter W, using 5 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No incentive	80.1% ±0.5%	4.3 ±0.04	N/A	N/A	17.46 ±0.2	0.72 ±0.06	14.62 ±0.2
IC ($W = 0.5$)	83.6% ±0.8%	4.6 ±0.06	52.9% ±2.1%	1.3 ±0.03	16.17 ±0.27	0.86 ±0.11	13.22 ±0.30
IC ($W = 1$)	85.5% ±0.9%	4.7 ±0.06	58.2% ±0.9%	1.4 ±0.02	15.54 ±0.29	0.91 ±0.12	12.49 ±0.19
IC ($W = 1.5$)	84.5% ±0.7%	4.7 ±0.02	59.5% ±1.1%	1.5 ±0.04	15.93 ±0.26	0.93 ±0.06	13.01 ±0.28
IC ($W = 2$)	84.6% ±0.5%	4.7 ±0.04	58.1% ±3%	1.4 ±0.04	15.82 ±0.21	0.99 ±0.1	12.82 ±0.34
RFR ($W = 0.5$)	80.2% ±0.5%	4.0 ±0.08	60.6% ±1.8%	1.4 ±0.04	17.46 ±0.24	0.84 ±0.04	14.56 ±0.29
RFR ($W = 1$)	81.9% ±0.5%	4.0 ±0.05	66.5% ±2.5%	1.6 ±0.05	16.89 ±0.18	0.86 ±0.07	14.03 ±0.28
RFR ($W = 1.5$)	81.7% ±0.8%	4.0 ±0.07	66.2% ±1.9%	1.7 ±0.04	16.98 ±0.26	0.89 ±0.07	14.0 ±0.3
RFR ($W = 2$)	80.9% ±0.8%	4.0 ±0.1	60.6% ±3.7%	1.4 ±0.1	17.21 ±0.27	0.81 ±0.07	14.35 ±0.34

For experiment 1, the nudging based on insertion costs is more "active" than RFR, i.e., a higher percentage of customers is nudged to a different time slot. For experiment 2, this is effect is inverted, although the differences are relatively small. Table 7.10 shows the results for the capacity constrained case. Again, $W = 1$ seems to be the best value for both IC and RFR. Both IC and RFR perform significantly worse compared to the nudging based on the most expensive time slot (experiment 1). Although the travel distance is somewhat lower, the travel time and especially the waiting time is much longer. IC performs slightly better compared to RFR.

Because RFR shows somewhat lower performance compared to IC, it might be effective to retrain the RFR model on realization data (see Section 7.2.3). We use the data obtained from the $W = 1$ experiment to retrain our model, and show performance on a different data set for both the time and capacity constrained case in Table 7.11.

Our RFR model improved significantly and now shows better performance than the cheapest insertion costs method. Compared with the situation without incentives we save 6.9% and 1.5% in travel time, for the time constrained and capacity constrained case, respectively.

In Figure 7.10 we depict the average absolute incentive that was given to all time slots, the line is a moving average (window = 15). This figure indicates how much difference between time slots, in terms of costs, is recognised by the cost approximation policy. We only show the capacity constrained case, since the time constrained case shows similar results. It becomes visible that our RFR model returns less erratic costs approximations between time slots in comparison with the insertion costs (IC) method, since the IC method covers a broader range for the average absolute incentive. Additionally, we observe that insertion costs sometimes does not recognise any difference between time slots, resulting in no incentives. At the end of the

Table 7.10: Simulation run statistics for the incentive policy for the randomly clustered instances with a capacity restriction (RC-C) for different levels of tuning parameter W, using 5 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No incentive	66.6% ±0.1%	3.8 ±0.01	N/A	N/A	21.74 ±0.14	3.31 ±0.26	16.28 ±0.18
IC ($W = 0.5$)	66.4% ±0.4%	3.9 ±0.02	54% ±2.4%	1.2 ±0.06	22.0 ±0.13	4.53 ±0.25	15.3 ±0.31
IC ($W = 1$)	66.6% ±0.07%	3.9 ±0.02	58.6% ±2%	1.3 ±0.06	21.93 ±0.13	4.77 ±0.24	14.98 ±0.28
IC ($W = 1.5$)	66.5% ±0.3%	3.9 ±0.01	57.8% ±3.9%	1.3 ±0.06	22.0 ±0.22	4.69 ±0.34	15.16 ±0.23
IC ($W = 2$)	66.6% ±0.1%	3.9 ±0.01	58.2% ±3.3%	1.3 ±0.06	22.0 ±0.16	4.71 ±0.31	15.14 ±0.24
RFR ($W = 0.5$)	66.6% ±0.1%	3.7 ±0.03	60.8% ±1.4%	1.3 ±0.03	22.82 ±0.21	4.60 ±0.21	16.16 ±0.23
RFR ($W = 1$)	66.7% ±0%	3.7 ±0.03	66.3% ±1.8%	1.5 ±0.04	22.78 ±0.13	4.81 ±0.34	15.96 ±0.36
RFR ($W = 1.5$)	66.6% ±0.1%	3.6 ±0.06	66.1% ±2.3%	1.6 ±0.08	22.9 ±0.14	5.18 ±0.15	15.64 ±0.29
RFR ($W = 2$)	66.6% ±0.1%	3.6 ±0.02	61.3% ±2.9%	1.4 ±0.1	22.9 ±0.16	4.97 ±0.32	15.76 ±0.48

Table 7.11: Simulation run statistics for the incentive policy for the randomly clustered instances with a time (RC-T) or capacity restriction (RC-C) for $W = 1$ and a retrained regression model, using 5 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No incentive (RC-T)	80.1% ±0.5%	4.3 ±0.04	N/A	N/A	17.46 ±0.2	0.72 ±0.06	14.62 ±0.2
RFR (RC-T retrained) ($W = 1$)	81.6% ±1.6%	4.0 ±0.06	65.8% ±2.5%	1.6 ±0.1	16.25 ±0.53	0.60 ±0.1	13.52 ±0.57
No incentive (RC-C)	66.6% ±0.1%	3.8 ±0.01	N/A	N/A	21.74 ±0.14	3.31 ±0.26	16.28 ±0.18
RFR (RC-C retrained) ($W = 1$)	66.6% ±0.07%	3.7 ±0.04	65.3% ±2.3%	1.5 ±0.08	21.42 ±0.35	3.60 ±0.53	15.86 ±0.58

booking horizon, both RFR and IC converge to a situation where incentives need to be capped to stay in the domain $[-1, 1]$, or there is no significant cost difference between time slots, so the incentive is zero. When incentives are capped, the incentives are as high as possible and have the most influence on the time slot choice. This effect is caused by the decreased capacity, i.e., less time slots are feasible to offer and therefore the standard deviation of time slot costs decreases too, causing high incentives that need to be capped to stay in the domain $[-1, 1]$. Later, the incentives become zero because only a single time slot can be offered.

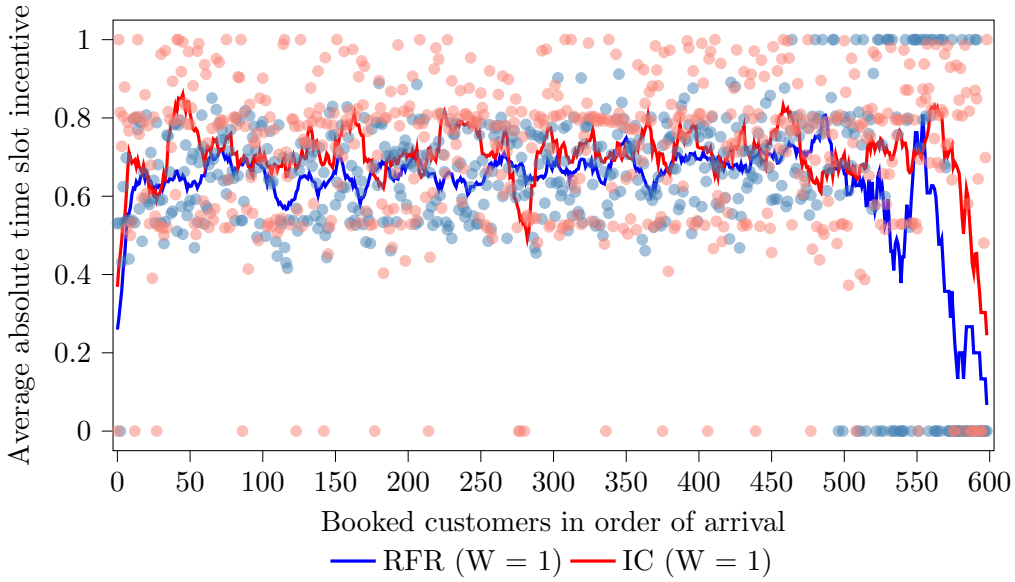

Figure 7.10: Average absolute incentive over all feasible time slots during the booking horizon, given by random forests (RFR) and the insertion costs method (IC), for a single replication of the capacity constrained case.

7.4 European E-grocery Retailer Case Experiments

In this section, we show the results of experiment 3, where we test our regression model on a real case derived from data of a large European e-grocery retailer. We first train our RFR model on simulation runs without any nudging. We observe that the model performance on the validation set is better in comparison with the model trained on the synthetic case. Probably this has to do with the heavy clustering of customers for the European e-grocery retailer case. This higher performance is illustrated in Figure 7.11. It becomes visible that a subset of the customers is "expensive", e.g., far away from the serving depot. The largest set of customers is close to the depots and therefore less expensive.

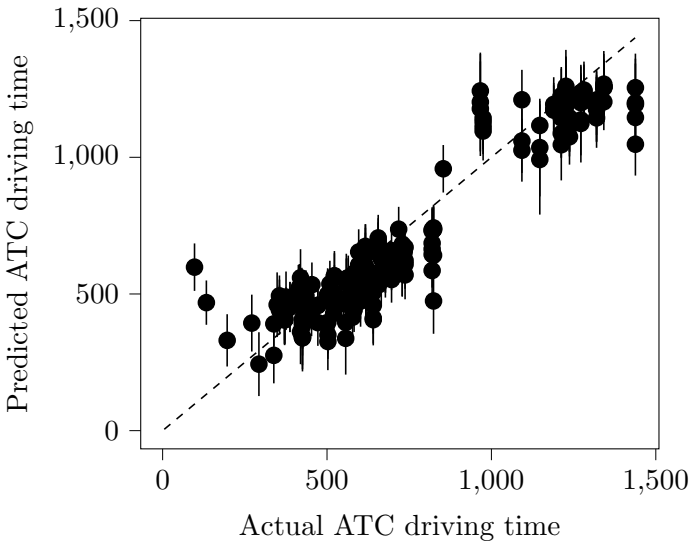

Figure 7.11: Unbiased standard error for the European e-grocery retailer instances for an ATC in minutes, predicted with random forests.

Although the error bars indicate that the model is not completely certain about its prediction, it seems that the model follows the prediction-observation line well. Most error bars cross the diagonal prediction-observation line, indicating that the model does not suffer a lot from residual noise in the data.

Table 7.12 shows the results of the first part of experiment 3, where we nudge infinitely flexible customers to the cheapest and most expensive time slot, respectively. Note that for these instances there are seven time slots available, opposed to the six time slots used for the previous experiments. We report the same statistics as before using 2 replications. We observe from the No TS experiment that we cannot plan more than 81.1% of the customers due to vehicle capacity restrictions. The addition of time slots causes an increase of travel time and travel distance of 34.5% and 33.9%, respectively. We observe that IC and RFR both can plan more customers when nudging the cheapest time slot, compared to the situation without incentives. IC saves 15.7% in travel time and 15.0% in distance per customer, compared with the situation without incentives. RFR improves slightly less compared with the situation without incentives; it saves 7.3% in travel time and 11.2% in distance per customer.

Table 7.12: Simulation run statistics for the European e-grocery retailer case, using 2 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No TS	81.1% ±0.2%	N/A	N/A	N/A	4.18 ±0.02	0 ±0	2.53 ±0.02
No incentive	80.4% ±0.04%	5.6 ±0	N/A	N/A	5.62 ±0.02	0.03 ±0.01	3.39 ±0.05
IC Cheap	81.1% ±0.2%	5.7 ±0.01	85.6% ±0.7%	3.0 ±0.03	4.74 ±0.10	0 ±0	2.88 ±0.07
IC Expensive	80.2% ±0.3%	5.2 ±0.02	85.2% ±0.06%	3.0 ±0.02	6.23 ±0	0.02 ±0.01	3.91 ±0.04
RFR Cheap	81.0% ±0.2%	5.6 ±0.02	86.8% ±0.6%	3.0 ±0.02	5.21 ±0.06	0.02 ±0.01	3.01 ±0.01
RFR Expensive	80.4% ±0.2%	5.5 ±0	86.8% ±0.3%	3.0 ±0.01	6.07 ±0.05	0.07 ±0.03	3.91 ±0.11

Table 7.13 shows the results for the simulation runs of the European e-grocery retailer case using our simple incentive policy. The results for this experiment are similar to the results of experiment 2. The incentive policy improves in comparison with the situation without incentives. In comparison with the best performing incentive policy setting, we see 0.7% more planned customers, 6.2% less travel time and 5.3% less traveled distance per customer. Waiting times are low and the differences between waiting times are insignificant. Again, IC shows somewhat better performance on most statistics, but RFR seems to be the more "active" policy with more nudging and more absolute changes of time slots. The difference between RFR and IC in terms of the percentage of customers that were nudged, seems to be larger than observed for experiment 2. The differences between values for W are small, $W = 1$ seems to be the best value overall. Although the predictive performance (see Figure 7.11) increased significantly, we do not see a similar performance improvement, indicating that only predicting downstream costs might not be enough for an effective incentive policy. The performance might be better when we make adaptations to our generic model to account for the increased complexity; the instances are multi-depot with an heterogeneous fleet and overlapping time slots.

Table 7.13: Simulation run statistics for the incentive policy for the European e-grocery retailer case for different levels of tuning parameter W, using 2 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No incentive	80.4% ±0.04%	5.6 ±0	N/A	N/A	5.62 ±0.02	0.03 ±0.01	3.39 ±0.05
IC ($W = 0.5$)	80.6% ±0.4%	5.6 ±0.03	42.2% ±0.2%	0.7 ±0.02	5.24 ±0.01	0.002 ±0	3.18 ±0.01
IC ($W = 1$)	80.5% ±0.3%	5.6 ±0.02	45.4% ±0.9%	0.8 ±0.03	5.25 ±0.02	0.02 ±0.01	3.15 ±0
IC ($W = 1.5$)	81.0% ±0.1%	5.6 ±0.01	45.0% ±0.3%	0.8 ±0	5.27 ±0.07	0.01 ±0	3.22 ±0.04
IC ($W = 2$)	81.0% ±0.3%	5.7 ±0.03	43.3% ±1.3%	0.8 ±0.01	5.27 ±0.07	0.01 ±0	3.21 ±0.08
RFR ($W = 0.5$)	80.5% ±0.08%	5.4 ±0.04	70.2% ±0.2%	1.8 ±0.2	5.92 ±0.21	0.03 ±0.03	4.04 ±0.29
RFR ($W = 1$)	81.0% ±0.6%	5.5 ±0.09	75.1% ±1.0%	2.2 ±0.05	5.84 ±0.17	0.08 ±0.05	3.96 ±0.24
RFR ($W = 1.5$)	80.8% ±0.6%	5.5 ±0.1	76.2% ±0.2%	2.2 ±0.04	5.99 ±0.09	0.05 ±0.03	4.20 ±0.04
RFR ($W = 2$)	80.7% ±0.6%	5.5 ±0.1	74.2% ±1.0%	2.1 ±0.01	5.93 ±0.28	0.06 ±0.03	4.14 ±0.16

7.5 Joint Cost Approximation with Random Forests and Insertion Costs

As we observed in the previous experiments, our RFR models often fail to outperform the insertion costs method (IC). We noted that the insertion costs sometimes results in a more aggressive strategy, caused by the more erratic nature of cheapest insertion. However, cheapest insertion is, as explained in Chapter 2, a method with high prediction errors. In this section, we propose a model that combines the insertion costs and our RFR model for estimating costs. This model is applied as follows: the first half of the booking horizon, we use the insertion costs as cost approximation method, in the second half of the the horizon, we use our RFR model as cost approximation, containing the cheapest insertion costs as new feature. The reasoning behind this as follows: in the first half of the booking horizon, most time slots have enough capacity to insert all customers, so nudging might be less important. Additionally, we observed that our regression model shows higher prediction error at the begin of the booking horizon, and insertion costs is only reliable as feature later in the booking horizon. Therefore, we only use the random forests model in the second half of the booking horizon. In order to use the insertion costs as feature, which are calculated per customer-time slot, we need to calculate the other features over single customer-time slot combinations as well, instead of the ATC structure as used before, i.e., we train our regression model on individual customers. We do this by defining ATC areas that are small enough so that they can only contain a single customer, i.e., $|\mathcal{A}| = |\mathcal{C}|$. Further, we use the same features as used for RFR (ii), with some adaptations for the application to a single-customer area $a \in \mathcal{A}$. We train on data obtained in the second half of the booking horizon. The performance for three different settings is shown: nudging an infinitely sensitive customer to the cheapest or the most expensive time slot, and nudging a customer with incentive sensitivity $f = 1$ using our simple incentive policy with $W = 1$. The results of the experiments for these three settings are shown in Table 7.14. It seems that the combined strategy is not better at recognising the cheapest and most expensive

time slot, since we did not improve compared to the previously used setting (see Table 7.12), in terms of travel costs or the percentage of planned customers. However, when we use the joint cost approximation model for giving incentives to multiple time slots with our simple incentive policy, we can improve, with slightly less travel and waiting time and a higher percentage of planned customers, compared to the best performing strategy shown in Table 7.13. Compared to the situation without incentives we save 7.1% in travel time per customer and 5.6% in travel distance per customer. Additionally, we can plan 0.7% more customers.

Table 7.14: Simulation run statistics for the European e-grocery retailer case with the combined cost approximation, using 2 replications.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No incentive	80.4% ±0.04%	5.6 ±0	N/A	N/A	5.62 ±0.02	0.03 ±0.01	3.39 ±0.05
IC & RFR Cheap	80.7% ±0.1%	5.6 ±0	85.2% ±0.8%	3.0 ±0.02	5.32 ±0.06	0.03 ±0.02	3.29 ±0.08
IC & RFR Expensive	80.4% ±0.2%	5.2 ±0.05	86.2% ±1.3%	3.0 ±0.02	5.99 ±0.10	0.02 ±0	3.78 ±0.12
IC & RFR simple policy ($W = 1$)	81.0% ±0.3%	5.6 ±0.02	58.7% ±0.1%	1.4±0.02	5.22 ±0.03	0.02 ±0	3.20 ±0

Figure 7.12 depicts the average absolute incentive that was given to all time slots, the line is a moving average (window = 15). This figure indicates how much difference between time slots, in terms of costs, is recognised by the joint cost approximation policy. The vertical dotted line shows the moment on the booking horizon where the switch from the insertion costs policy to the joint model is made.

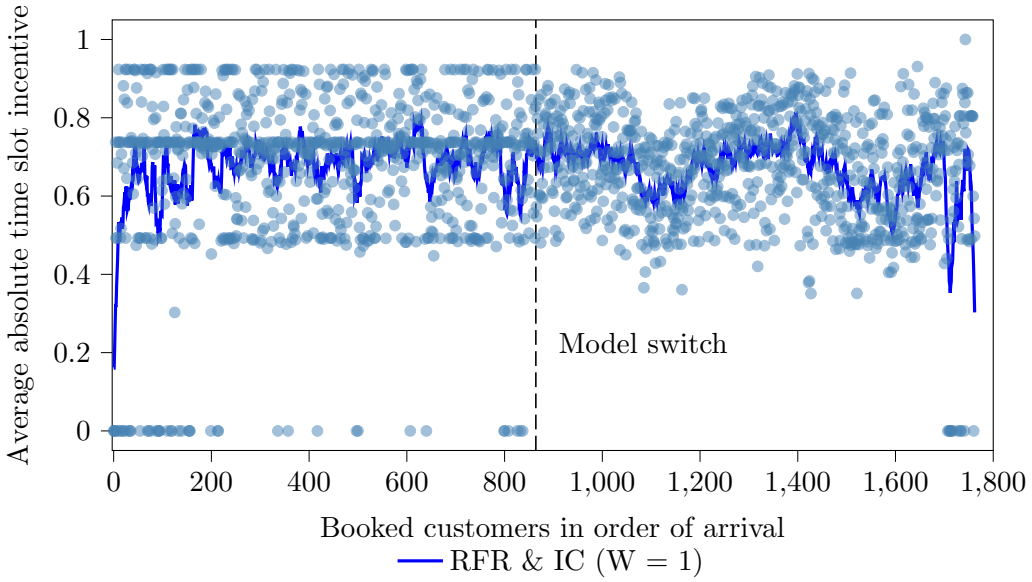

Figure 7.12: Average absolute incentive over all feasible time slots during the booking horizon, given by a joint cost approximation strategy of first only insertion costs, and later a random forests model, including insertion costs as feature, for a single replication of the European e-grocery retailer case.

We observe that the incentives given with insertion costs (first part of horizon) are mostly between two bounds, with many observations in the middle, i.e., often incentives with similar magnitude are given. Also, we sometimes see that no difference between time slots is recognised, and no incentive is given. After the model switch, we still observe that incentives are roughly between two bounds, but the average incentives are less centred around one mean, i.e., more spread and slightly more erratic.

7.6 Conclusions

In this chapter, we showed the computational results of our three experiments. First, we showed the performance of our regression model on validation data. Next, we applied our regression model to a case where infinitely flexible customers are nudged to the cheapest or most expensive time slot. We showed the relevance of time slot research and the added value of dynamic time slot pricing. Also, we showed in what situations our regression model performs the best. Next, we applied the regression model and showed the performance on real case data from an European e-grocery retailer. Finally, we showed the performance of a joint cost approximation model that combines insertion costs with random forests regression.

Chapter 8

Conclusions and Recommendations

In this chapter, we summarize our results, draw conclusions and make recommendations. In Section 8.1 we state our main findings, answer our research questions and discuss the results and limitations. Next, in Section 8.2 we do recommendations to ORTEC based on our findings and discuss opportunities for further research.

8.1 Conclusions

For this research, we explored the possibilities for improving time slot solutions in terms of transportation costs. Our main focus was finding methods that can approximate the costs of adding a customer to a time slot, and subsequently we studied the effects of dynamic pricing based on these cost approximations.

To model customer behavior without the need of a behavioral study, we developed a parametric rank-based customer choice model for which we can influence the ranking of time slots by giving incentives or penalties. We developed a simple incentive policy to test our customer choice model and time slot cost approximation.

Our solution approach for approximating costs of time slots is centred around the prediction of downstream transportation costs using regression models. To improve prediction and reduce noise, we aggregated customers in area-time slot combination (ATC) clusters, after which we trained regression models to predict the travel times to serve an ATC-cluster. To train our model, we divided the downstream travel time over all customers using a method we call "half-edge partitioning". We used a limited set of features as input for our regression models. Examples of these features are: the number of customers in an ATC, the number of days between customer arrival and the end of the horizon, the distance between the depot and the ATC-centroid, the variance of the angles between customers in an ATC and the depot and the average distance between customers in an ATC. We showed the results for both random forests and neural network regression models for this research.

For validation, we used a discrete-event simulation model that simulates the complete booking horizon for time slot booking on a single delivery day. The model is based on the model as presented in Visser et al. (2019).

We tested our proposed solution on two different cases. First, we ran several experiments with a synthetic case, i.e., a case with generated data in different spatial variants and using both a time constrained variant and a capacity constrained variant. For the second case, we used data from an European e-grocery retailer.

We answered our research questions by means of three experiments. First, we studied the performance of both random forests and neural networks on a validation set of the synthetic case. We found that our model performs well on static data, i.e., a model that is trained and tested on data after the end of the booking horizon. However, during the booking horizon the feature values take different values, e.g., when there are still few customers in an ATC-cluster.

When testing on the complete horizon, our regression model performance dropped. We proposed a method that splits the booking horizon in partitions, on which different regression models can be trained. We showed that this method potentially performs better, but requires more data and can be more sensitive to noise. Since random forests and neural networks performed similar, we only displayed the performance of random forests for further experiments.

In the second part of experiment 1, we showed the performance of our proposed models on a hypothetical case where customers are infinitely flexible and always go to the time slot we give an incentive to, either the cheapest or most expensive. We compared our three proposed random forests models (RFR) with the cheapest insertion costs (IC) method as currently in use by the ORTEC Time Slotting Service. First, we demonstrated the relevance of time slotting research. When we compared the situation without time slots, i.e., customers can be planned the whole day, with the situation with time slots, we saw a decrease of the percentage of customers that can be served ($\sim$ 20%), and an increase in travel time, waiting time and distance per customer of 93.6%, 89.5% and 52.5%, respectively. Second, we showed the importance of time slot pricing. When giving incentives, we can plan 6% more customers and decrease travel time, waiting time and distance per customer by 11% compared to the situation without incentives. Our random forests model often performed similar or worse compared to the insertion costs method. We observed that especially for the capacity constrained variant, where less customers can be planned, our model trumps the insertion costs method. When there are many customers, and all ATC-clusters are populated, the effect of a time slot incentive policy is relatively small. However, when the area of operation is more sparsely populated, the effect of adding a customer to a time slot may be larger, since a vehicle might has to do a large detour. To further illustrate this, we conducted an additional experiment with a fleet consisting of a single vehicle, serving a smaller area. With this additional experiment we confirmed our general idea and showed the value of our model: our model can serve a similar number of customers compared to the insertion costs method, but reduces the travel time per customer by 7.3% and the distance travelled per customer by 6.3%.

Next, we showed an iterative method for improving prediction models. We trained all our models on data without nudging. During the simulation, we did nudge customers. Therefore, there is a difference between the training environment and the application environment. We propose a model that obtains data and retrains predictive models in an iterative fashion. In our experiment, we showed the potential of this method. We showed a significant increase in prediction quality (R^2) and an increase in performance in terms of percentage of customers that can be planned (+3.1%) and travel and waiting time (-4.2%).

For experiment 2, we conducted tests for a more realistic configuration where multiple time slots are offered and need to get an incentive according to the cost approximation method. We used our simple incentive policy to test how good our regression model could predict the cost of a time slot *relative* to other time slots. We observed that the performance of both the insertion costs method and our random forests model is roughly in-between the performance of nudging the cheapest and most expensive time slot, respectively. However, after applying our iterative model improvement framework, we observed an improvement and we could match or improve compared to the cheapest insertion costs method. We also showed that our model returns less erratic costs predictions in comparison with IC, as the incentives cover a smaller range of values during the complete booking horizon.

For experiment 3, we tested our regression model and simple incentive policy on the European e-grocery retailer case. This case is larger in terms of customers and fleet, and has a distinct customer dispersion. Also, the instances for this case have multiple depots and use a heterogeneous fleet. In the situation with infinitely flexible customers, IC could save in travel times (-15.7%) and distance (-15.0%) per customer, while planning slightly more customers compared to the case without incentives. Our regression model planned a similar number of customers, and saved 7.3% in travel time and 11.2% in distance per customer, respectively.

For the second part of experiment 3, we showed performance using our simple incentive policy. We observed an increase in the number of planned customers (+0.7%) and a significant saving in travel time per customer (−6.2%) and distance travelled per customer (−5.3%), compared with the situation without incentives. Also, our RFR model is more "active" in nudging, i.e., more customers are nudged to a different time slot and are nudged further away from their first preference, compared to the insertion costs policy. Although our RFR model showed better predictive performance on the case in comparison with the synthetic case, we did not see a similar improvement in terms of operational costs. This indicates that only predicting operational downstream costs might not be enough as input for an effective incentive policy.

Finally, we proposed a new cost approximation model that combines cheapest insertion and our random forests model. In the first part of the booking horizon this joint model only uses cheapest insertion costs, in the second part of the booking horizon it uses random forests with cheapest insertion as new feature. This joint random forests model does not calculate feature values over ATC-clusters as before, but calculates over single customer-time slot combinations. Especially in the configuration for which we use our simple incentive policy, we saw an improvement compared to solely using the IC or RFR model. We could plan 0.7% more customers and save 7.1% in travel times, compared to the situation without incentives.

We only studied the effect of our solution approach on a single delivery day. Therefore, we made the assumption that customer time slot choice is independent from offerings on different delivery days. In practice, this is probably not the case. Also, we modelled customer behavior in a practical but simplistic way. Customers received an incentive sensitivity score of 1 for most experiments, which means that we assumed that customers are always fully sensitive to time slot incentives. In reality, some customers might be indifferent to incentives. Customer behavior was limited further, since in reality customer behavior might be more complex and cannot be captured by a parametric rank-based model. Because of these limitations, we cannot be certain of the actual effects of time slot incentives. We only showed the *potential* effect for a best-case scenario where customers are fully cooperative. With our European e-grocery retailer case, we showed a realistic case, in terms of customer data and fleet and depot structure. However, the way we chose to divide downstream costs over customers (half-edge partitioning) further limits our results, since a different method might render different performance.

We can conclude that dynamic time slot pricing has added value for time slotting. Significant potential savings can be made when an effective incentive strategy is used. Our regression model performs well, especially for instances with clustered customers or sparsely populated areas. We observed that in most cases, our regression model struggles to generalize far away from the training data. As already mentioned in related work (Yang et al., 2016; Klein et al., 2018), only considering the transportation costs as input for an incentive strategy restrains the potential of dynamic pricing. Although downstream transportation costs are an important factor, delivery revenues are important too. When a customer is planned for a time slot, opportunity costs are incurred, since the capacity for future customers is decreased. This is especially relevant when the fleet capacity is reaching its limits at the end of the booking horizon. Although we study the correlation between customer time slots and downstream costs, there is a lacking *causality* between giving incentives and downstream costs. Hence, the dynamic nature and complexity of time slotting causes a disconnect of our time slot cost approximation, time slot incentive policy and the final downstream costs. Our rudimentary incentive policy fails to capture the full potential of a time slot pricing policy, when comparing with the performance of nudging the cheapest time slot with an infinitely flexible customer. In Section 8.2 we further discuss this issue and possible further research.

Summarizing, the contributions of this thesis to existing scientific literature are: (i) our cost approximation regression model that can be used for estimating the costs of time slots and can be applied to general VRPTW problems where customer selection is required, (ii) the application of our regression model to the attended home delivery case, (iii) the rank-based customer choice

model that combines a ranking-based choice model with a utility theory approach, and (iv) the application of a commercial vehicle routing solver and time slot allocation software to our problem. Our contributions to practice are: (i) the results that show the potential of dynamic pricing for time slotting, and (ii) the practical implications of using regression models for time slotting, e.g., the way we use data for training, the features we use, and how we evaluate our models.

8.2 Recommendations and Further Research

Based on the conclusions of this research, we formulate several recommendations to ORTEC. Furthermore, we point to several subjects for further research.

With our experiments, we showed that giving incentives can potentially decrease costs, and therefore we advice to implement a time slot incentive policy. We showed that the currently in use insertion costs method has its limitations, but can be used as a basic cost approximation policy. On the short-term, if a dynamic pricing policy needs to be implemented, we recommend to use the insertion costs method as main input for an incentive policy. Most retailers have their own business logic and time slot pricing policy that can be linked to the insertion costs calculation. However, the insertion costs returns highly erratic costs approximations, therefore it might not be the best method to directly link it to an incentive policy.

On the longer-term, we believe it is beneficial to use a different method than insertion costs as input for an incentive policy. We showed a possible way to combine insertion costs with a machine learning model, which could be an interesting first step towards a more advanced cost approximation model. However, more research needs to be conducted to find an appropriate method. Our regression model can be a starting point for more advanced methods. It will be necessary to obtain more information from retailers about customer behavior and preferences in terms of the balance between transportation costs and service levels. The importance of incentives during the booking horizon may vary, e.g., incentives may only be important at the last few days before the cutoff time. This needs more research. Also, the cost approximation method needs to be able to deal with complex settings, e.g., overlapping time slots. Regardless the method, we advice to review ethical concerns related to influencing customer behavior, before implementing an automated dynamic time slot incentive policy.

It could be interesting for ORTEC to look at methods that integrate cost approximations and time slot incentives in a single solution algorithm. Although advanced algorithms for each separate part can help to decrease costs and increase service levels, as shown in this thesis, it seems that more savings can be made, in terms of both operational as computational costs, when algorithms are integrated. As example, for our research we separated the two components, i.e., the cost approximation method is used as decision support for a separate incentive policy. However, it might be beneficial to change this two-phase approach to a single-phase approach, by developing an algorithm that can use a new customer arrival as input, and give time slot incentives as output.

Further research can be done on several topics. First, we neglected to elaborate on the aggregation structure used for aggregating customers in spatial areas. We used a method that produced acceptable aggregation areas, but we did not study different aggregation structures, nor did we develop an automatic method for aggregating areas. Nevertheless, the way customers are assigned to areas has major influence on the performance of cost predictions. Therefore, further research could be done into area aggregation techniques, e.g., using adaptive grids that automatically identify customer clusters. The definition of downstream costs is another interesting aspect that requires more research, since the half-edge partitioning method we used might be improved to consider more than only travel time or distance.

As mentioned, we observed that for our approach there is a disconnect between costs approximation, the incentive policy and the (downstream) costs. A method that integrates these

three aspects could be beneficial. The effects of pricing decisions cannot be easily observed and are hard to capture in a standard Bellman-equation with direct and future costs. The use of Monte Carlo simulation in combination with (deep) reinforcement learning methods could be an interesting path for further research. A policy based method could be learned to give incentives by doing many Monte Carlo simulations. Difficulties that have to be overcome can be mainly found in the training procedure, since the computational costs are high and customer behavior modelling is complex. Also, reinforcement learning models will need to be able to generalize far away from their training data, which might be difficult for specific instances. The eventual cost estimation influences the incentive policy, and in turn the incentive decisions influence the cost estimates during the booking horizon. A reinforcement learning model could be valuable for learning this explicit relationship.

References

Niels Agatz, Ann Melissa Campbell, Moritz Fleischmann, and Martin Savelsbergh. Challenges and opportunities in attended home delivery. In Bruce Golden, S. Raghavan, and Edward Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 379–396. Springer US, Boston, MA, 2008. ISBN 978-0-387-77778-8. doi: 10.1007/978-0-387-77778-8-17.

Niels Agatz, Ann Campbell, Moritz Fleischmann, and Martin Savelsbergh. Time slot management in attended home delivery. *Transportation Science*, 45(3):435–449, 2011. ISSN 00411655, 15265447.

Niels Agatz, Ann Campbell, Moritz Fleischmann, Jo Nunen, and Martin Savelsbergh. Revenue management opportunities for internet retailers. *Journal of Revenue & Pricing Management*, 12:128–138, 03 2013. doi: 10.1057/rpm.2012.51.

Niels Agatz, Yingjie Fan, and Daan Stam. Going green: the effect of green labels on delivery time slot choices. *ERIM report series research in management Erasmus Research Institute of Management*, (ERS-2020-009-LIS), July 2020. URL `http://hdl.handle.net/1765/128912`.

Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, 2018. ISBN 978-3-319-94462-3. doi: 10.1007/978-3-319-94463-0.

Akamai. Akamai online retail performance report: Milliseconds are critical. https://www.akamai.com/uk/en/about/news/press/2017-press/ akamai-releases-spring-2017-state-of-online-retail-performance-report.jsp, 2017.

Kursad Asdemir, Varghese S. Jacob, and Ramayya Krishnan. Dynamic pricing of multiple home delivery options. *European Journal of Operational Research*, 196(1):246 – 257, 2009. ISSN 0377-2217. doi: 10.1016/j.ejor.2008.03.005.

Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. `https://github.com/onnx/onnx`, 2019.

B.M. Baker and M.A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers and Operations Research*, 30(5):787–800, 2003. doi: 10.1016/S0305-0548(02)00051-5.

J. Banks, J. S. Carson, B. L. Nelson, and D. Nicol. *Discrete-Event System Simulation*. Prentice Hall, 5 edition, 2010. ISBN 0136062121.

Jillian Beardwood, J. H. Halton, and J. M. Hammersley. The shortest path through many points. *Mathematical Proceedings of the Cambridge Philosophical Society*, 55(4):299–327, 1959. doi: 10.1017/S0305004100034095.

T. Bektas. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006. doi: 10.1016/j.omega.2004.10.004.

J.E. Bell and P.R. McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1):41–48, 2004. doi: 10.1016/j.aei.2004.07.001.

Peter Belobaba. *Air travel demand and airline seat inventory management.* PhD thesis, Massachusetts Institute of Technology, Cambridge, 1987.

Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300 – 313, 2016. ISSN 0360-8352. doi: 10.1016/j.cie.2015.12.007. URL `http://www.sciencedirect.com/science/article/pii/S0360835215004775`.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324.

Glen Van Brummelen. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry.* Princeton University Press, 2013. ISBN 9780691148922.

J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera, and A.A. Juan. Rich vehicle routing problem: Survey. *ACM Computing Surveys*, 47(2), 2014. doi: 10.1145/2666003.

Ann Campbell and Martin Savelsbergh. Incentive schemes for attended home delivery services. *Transportation Science*, 40:327–341, 08 2006. doi: 10.1287/trsc.1050.0136.

Ann Melissa Campbell and Martin W. P. Savelsbergh. Decision support for consumer direct grocery initiatives. *Transportation Science*, 39(3):313–327, 2005. doi: 10.1287/trsc.1040.0105.

Quinten Cederhout. The impact on the final delivery schedule of different procedures that handle arriving customers during route optimization in a real-time dynamic time slot management system. Master's thesis, Delft University of Technology, 2020. URL `http://resolver.tudelft.nl/uuid:08b61211-bfd7-4630-81a0-2b5595a69f0c`.

T. Chien. Operational estimators for the length of a traveling salesman tour. *Comput. Oper. Res.*, 19:469–478, 1992.

Nicos Christofides and S. Eilon. Expected distances in distribution problems. *Journal of the Operational Research Society*, 20:437–443, 1969.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964. doi: 10.1287/opre.12.4.568.

Catherine Cleophas and J. Ehmke. When are deliveries profitable? *Business & Information Systems Engineering*, 6:153–163, 2014.

Jean-François Cordeau, Gilbert Laporte, Martin W.P. Savelsbergh, and Daniele Vigo. Chapter 6 vehicle routing. In Cynthia Barnhart and Gilbert Laporte, editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367–428. Elsevier, 2007. doi: https://doi.org/10.1016/S0927-0507(06)14006-2.

G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6): 791–812, 1958. ISSN 0030364X, 15265463.

Carlos F. Daganzo. Modeling distribution problems with time windows: Part i. *Transportation Science*, 21(3):171–179, 1987. doi: 10.1287/trsc.21.3.171. URL `https://doi.org/10.1287/trsc.21.3.171`.

Bradley Efron. Estimation and accuracy after model selection. *Journal of the American Statistical Association*, 109(507):991–1007, 2014. doi: 10.1080/01621459.2013.823775.

Jan Fabian Ehmke and Ann Melissa Campbell. Customer acceptance mechanisms for home deliveries in metropolitan areas". *European Journal of Operational Research*, 233(1):193 – 207, 2014. ISSN 0377-2217. doi: 10.1016/j.ejor.2013.08.028.

S. Eilon, C. D. T. Watson-Gandy, N. Christofides, and R. de Neufville. Distribution management-mathematical modelling and practical analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-4(6):589–589, 1974. doi: 10.1109/TSMC.1974.4309370.

Nasser A. El-Sherbeny. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University - Science*, 22(3):123 – 131, 2010. ISSN 1018-3647. doi: 10.1016/j.jksus.2010.03.002.

J. Elith, J. R. Leathwick, and T. Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77(4):802–813, 2008. doi: https://doi.org/10.1111/j.1365-2656.2008.01390.x.

Miguel Andres Figliozzi. Planning approximations to the average length of vehicle routing problems with varying customer demands and routing constraints. *Transportation Research Record*, 2089(1):1–8, 2008. doi: 10.3141/2089-01. URL `https://doi.org/10.3141/2089-01`.

F.P. Goksal, I. Karaoglan, and F. Altiparmak. A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery. *Computers and Industrial Engineering*, 65(1), 2013.

Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, USA, 2nd edition, 1998. ISBN 0132733501.

Florent Hernandez, Michel Gendreau, and Jean-Yves Potvin. Heuristics for tactical time slot management: a periodic vehicle routing problem view. *International Transactions in Operational Research*, 24(6):1233–1252, 2017. doi: 10.1111/itor.12403.

A Hindle and D Worthington. Models to estimate average route lengths in different geographical environments. *Journal of the Operational Research Society*, 55(6):662–666, 2004. doi: 10.1057/palgrave.jors.2601751. URL `https://doi.org/10.1057/palgrave.jors.2601751`.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL `https://faculty.marshall.usc.edu/gareth-james/ISL/`.

Eoghan Keany. Boruta-shap: Borutashap. Nov 2020. doi: 10.5281/zenodo.4247618.

Robert Klein, Jochen Mackert, Michael Neugebauer, and Claudius Steinhardt. A model-based approximation of opportunity cost for dynamic pricing in attended home delivery. *OR Spectrum*, 40:969 – 996, 12 2018. doi: 10.1007/s00291-017-0501-3.

Robert Klein, Michael Neugebauer, Dimitri Ratkovitch, and Claudius Steinhardt. Differentiated time slot pricing under routing considerations in attended home delivery. *Transportation Science*, 53(1):236–255, 2019. doi: 10.1287/trsc.2017.0738.

G.D. Konstantakopoulos, S.P. Gayialis, and E.P. Kechagias. Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. *Operational Research*, 2020. doi: 10.1007/s12351-020-00600-7.

Y. Kuo and C.-C. Wang. A variable neighborhood search for the multi-depot vehicle routing problem with loading cost. *Expert Systems with Applications*, 39(8):6949–6954, 2012. doi: 10.1016/j.eswa.2012.01.024.

Miron B. Kursa and Witold R. Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010. ISSN 1548-7660. doi: 10.18637/jss.v036.i11.

Averill M. Law. *Simulation Modeling & Analysis*. McGraw-Hill, New York, NY, USA, 5 edition, 2015.

Tak C. Lee and Marvin Hersh. A model for dynamic airline seat inventory control with multiple seat bookings. *Transportation Science*, 27(3):252–265, 1993. ISSN 00411655, 15265447.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. doi: 10.1002/net.3230110211.

Martijn R.K. Mes and Arturo Eduardo Perez Rivera. Approximate dynamic programming by practical examples. In Richard Boucherie and Nico M. van Dijk, editors, *Markov Decision Processes in Practice*, number 248 in International Series in Operations Research & Management Science, pages 63–101. Springer, mar 2017. ISBN 978-3-319-47766-4. doi: 10.1007/978-3-319-47766-4_3.

F.A.T. Montané and R.D. Galvão. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Research*, 33(3): 595–619, 2006. doi: 10.1016/j.cor.2004.07.009.

Pedro Augusto Munari. A generalized formulation for vehicle routing problems. *ArXiv*, abs/1606.01935, 2016.

D. Nicola, R. Vetschera, and A. Dragomir. Total distance approximations for routing solutions. *Computers & Operations Research*, 102:67 – 74, 2019.

ORTEC. About ORTEC, 2020a. URL `https://ortec.com/en-us/about-us`. [Online; accessed 29-July-2020].

ORTEC. ORTEC Technology general onboarding information, 2020b.

ORTEC Math Innovation Team. Math innovation team, why, what and how, 2020.

I.H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41(4):421–451, 1993. doi: 10.1007/BF02023004.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

C. Dennis Pegden, Randall P. Sadowski, and Robert E. Shannon. *Introduction to Simulation Using SIMAN*. McGraw-Hill, Inc., USA, 2nd edition, 1995. ISBN 0070493200.

Kivan Polimis, Ariel Rokem, and Bryna Hazelton. Confidence intervals for random forests in python. *Journal of Open Source Software*, 2(1), 2017.

Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience, USA, 2007. ISBN 0470171553.

Warren B. Powell and Ilya O. Ryzhov. *Optimal Learning and Approximate Dynamic Programming*, chapter 18, pages 410–431. John Wiley & Sons, Ltd, 2013. ISBN 9781118453988.

C. Prins, P. Lacomme, and C. Prodhon. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies*, 40:179–200, 2014. doi: 10.1016/j.trc.2014.01.011.

Francesc Robuste, Carlos F. Daganzo, and Reginald R. Souleyrette. Implementing vehicle routing models. *Transportation Research Part B: Methodological*, 24(4):263 – 286, 1990.

Witold R. Rudnicki, Marcin Kierczak, Jacek Koronacki, and Jan Komorowski. A statistical method for determining importance of variables in an information system. In Salvatore Greco, Yutaka Hata, Shoji Hirano, Masahiro Inuiguchi, Sadaaki Miyamoto, Hung Son Nguyen, and Roman Słowiński, editors, *Rough Sets and Current Trends in Computing*, pages 557–566, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-49842-1.

André Snoeck, Daniel Merchán, and Matthias Winkenbach. Revenue management in last-mile delivery: state-of-the-art and future research directions. *Transportation Research Procedia*, 46:109 – 116, 2020. ISSN 2352-1465. doi: 10.1016/j.trpro.2020.03.170. The 11th International Conference on City Logistics, Dubrovnik, Croatia, 12th - 14th June 2019.

Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987. doi: 10.1287/opre.35.2.254.

Arne K. Strauss, Robert Klein, and Claudius Steinhardt. A review of choice-based revenue management: Theory and methods. *European Journal of Operational Research*, 271(2):375 – 387, 2018. ISSN 0377-2217. doi: 10.1016/j.ejor.2018.01.011.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Kenneth E. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2 edition, 2009. doi: 10.1017/CBO9780511805271.

Garrett J. van Ryzin and Kalyan T. Talluri. *An Introduction to Revenue Management*, chapter Chapter 6, pages 142–194. INFORMS tutorials in Operations Research, 2005. doi: 10.1287/educ.1053.0019.

Thomas Visser, Niels Agatz, and Remy Spliet. Simultaneous customer interaction in online booking systems for attended home delivery. Technical Report ERS-2019-011-LIS, ERIM report series research in management Erasmus Research Institute of Management, October 2019. URL `http://hdl.handle.net/1765/120585`.

Stefan Wager, Trevor Hastie, and Bradley Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *Journal of machine learning research : JMLR*, 15: 1625–1651, 05 2014.

M. H. J. Webb. Cost functions in the location of depots for multiple-delivery journeys. *Journal of the Operational Research Society*, 19(3):311–320, 1968. doi: 10.1057/jors.1968.74. URL `https://doi.org/10.1057/jors.1968.74`.

Xinan Yang and Arne Strauss. An approximate dynamic programming approach to attended home delivery management. *European Journal of Operational Research*, 263:935–945, 06 2017. doi: 10.1016/j.ejor.2017.06.034.

Xinan Yang, Arne K. Strauss, Christine S. M. Currie, and Richard Eglese. Choice-based demand management and vehicle routing in e-fulfillment. *Transportation Science*, 50(2):473–488, 2016. doi: 10.1287/trsc.2014.0549.

Hannu Yrjölä. Physical distribution considerations for electronic grocery shopping. *International Journal of Physical Distribution and Logistics Management*, 31(10):746–761, 2001. ISSN 0960-0035.

Bahar Çavdar and Joel Sokol. A distribution-free tsp tour length estimation model for random graphs. *European Journal of Operational Research*, 243(2):588 – 598, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2014.12.020. URL `http://www.sciencedirect.com/science/article/pii/S0377221714010212`.

Appendices

Appendix A

Business Process with the Solution Approach

Figure A.1 shows the complete business process with ORTEC software, as depicted before in Figure 2.1. In the orange box, we indicate how our solution approach, cost approximation and time slot incentives, is fitted inside the current process. After a feasibility check by the ORTEC Time Slotting Service (OTSS) a cost approximation is done for all feasible time slots, and all feasible time slots are given an incentive. Next, the time slots are returned to the customer, who can make a choice for the delivery moment.

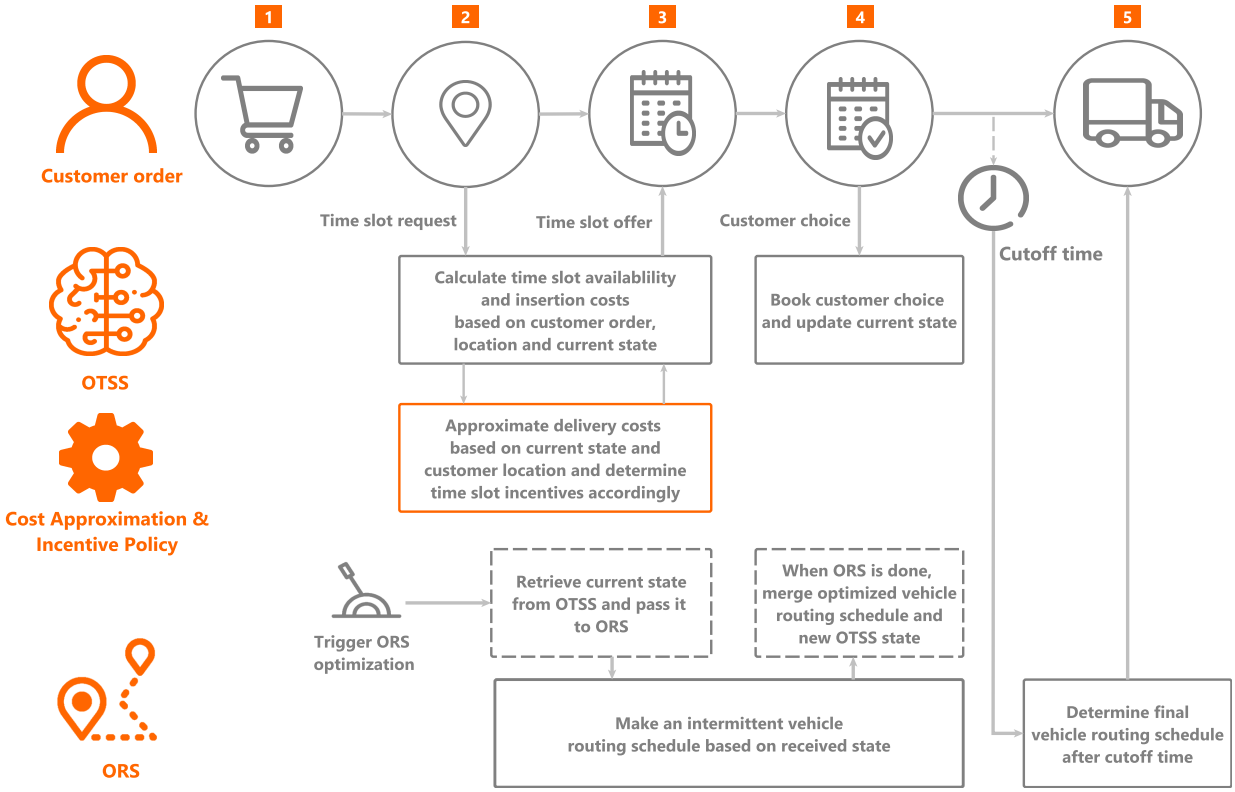

Figure A.1: Customer ordering process (business process perspective) including our solution approach.

Appendix B

Discrete-event Simulation Model

In this chapter we show the practical elements of the discrete-event simulation model in more detail. In Section B.1 we elaborate on the vital elements of the simulation model and in Section B.2 we describe the simulation input data.

B.1 Simulation Model Elements

The discrete-event simulation consist of several key elements. In this section, we discuss the vital elements: the system state, event structure, simulation clock, and event routine.

An *event* can be defined as an instantaneous occurrence that might change the state of a system (Law, 2015). We summarize all possible events and their potential effect on the system state in Table B.1.

Table B.1: Summary of all events in the time slot simulation tool.

Event	Event description
TSOfferRequestEvent	Customer arrives, call OTSS for insertion feasibility and costs
TSOfferResponseEvent	Customer chooses a delivery time slot or leaves system
TSBookingRequestEvent	Request a new customer booking in OTSS
TSBookingResponseEvent	Book new customer or return a blocking of the customer
TSBookingApplyEvent	Update OTSS state with new customer and check if a new ORS call should be started
RouteOptimizationRequestEvent	Request a new intermittent ORS optimization
RouteOptimizationResponseEvent	Obtain new routing schedule from ORS after a call
RouteOptimizationApplyEvent	Apply the ORS schedule to OTSS, deal with conflicts if applicable
FinalRouteOptimizationEvent	Call a final ORS optimization after the cutoff time

The system state is defined as all variables that are used to describe the system at a certain point in time (Law, 2015). Our system state consists of (i) all currently arrived and booked customers including all their customer characteristics as booked in the ORTEC Time Slotting

Service (OTSS), (ii) all (intermittent) routing schedules as constructed by OTSS or the ORTEC Route Optimization Service (ORS), and (iii) all other performance indicators tracked during the simulation. As becomes apparent from Table B.1, the simulation model interacts with the two ORTEC products (OTSS for time slotting and ORS for routing). We describe the different events following the simulation flowchart, as shown in Figure B.1. Also see Figure 2.1 for

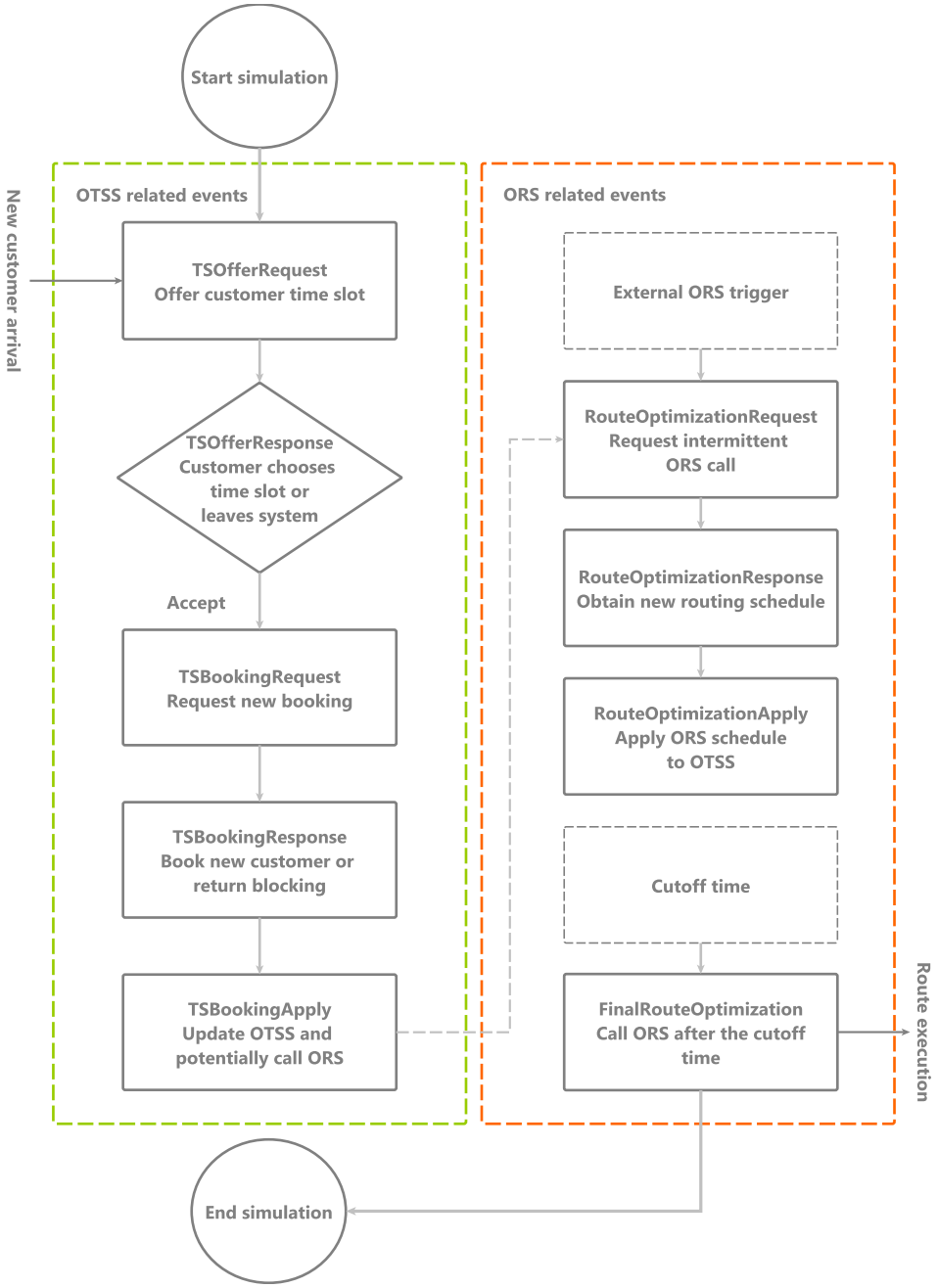

Figure B.1: Flowchart of the discrete-event simulator.

a business process view of the customer order process. The first event, a TSOfferRequest, is triggered when a new arrived customer requests a time slot. A call is made to OTSS to obtain all feasible time slots and the corresponding insertion costs. No system state changes are made yet. The TSOfferResponse event is a customer choice related event, since it records the choice of the customer: choose one of the offered time slots, or leave the system without choosing a time slot. When the customer leaves the system without a choice, the next event (triggered by the simulation clock) is a new customer arrival or the end of the booking horizon. When

the customer accepted a time slot, a new booking in the OTSS is requested (TSBookingRequest), which in turn triggers a TSBookingApply, in this event the OTSS state is updated with a new customer booking. If trigger criteria are met (e.g., trigger every 20^{th} customer), the TSBookingApply event in turn triggers a new RoutoptimizationRequest event. The current OTSS state (all booked customers) are send to ORS and a new optimized schedule is constructed and obtained after a RouteOptimizationResponse event. An ORS call can, certainly when the VRPTW instance is large, take a long time. In practice, an ORS request can take up to 45 minutes. During this time, new customers might have arrived and have been booked in the OTSS state. When ORS is finished, the OTSS state needs to be merged with the ORS schedule. For more information about this merge procedure, we refer to Cederhout (2020). Depending on the model settings, a RouteOptimizationRequest can be triggered by the simulation clock, i.e., when the settings for a trigger are time related and not related to new customer arrivals.

Only when no new customer can arrive, after the cutoff time, the final route optimization event is triggered. After this event, the definitive routing plan is obtained and the simulation ends. Although the events in Figure B.1 have been catagorized to OTSS-related or ORS-related, this does not mean that events are solely executed by ORS or OTSS. For instance, a RouteOptimizationApply event is triggered by a newly obtained ORS schedule, but is executed by OTSS. Note that OTSS keeps track of its own routing schedule, constructed using cheapest insertion, see Chapter 2 for the full explanation.

B.2 Simulation Model Input Data

The input data for the simulation tool consist of two different structures: a configuration file and an instance file. The configuration file sets the settings for ORS and OTSS. This file contains a distinct "recipe" of heuristics and/or metaheuristics that are used to obtain an optimized routing schedule. Also, the external trigger, if any, and merge strategy are defined in this structure (see Figure B.1).

The instance file structure contains all information needed for a simulation run. This includes a list of the following four elements: (i) the start and end time of the simulation, (ii) all data related to a region of operation (we only consider a single region per simulation). This information consists of the regional depot coordinates and the available time slots in the respective region. The third element (iii) consist of all routing information. For the homogeneous fleet, the following data is stated: earliest start and finish time of the drivers, capacities (for different type of products), costs per distance unit, costs per hour of operation, startup costs of a vehicle, the start and finish location (depot), speed factor, and allowed slack before or after the earliest start and finish time in minutes. Finally, the fourth element (iv) are all customers and their characteristics. For this simulation, most random customer data is generated before the simulation. A customer entity consist of the customer location, order quantities (of all different product types), service time duration in seconds, offer request time (event time), time needed to select a time slot, and region identification code. Included in our customer choice model, a customer also has a list of time slot preferences, listed in descending order, from highest base score to lowest base score. The order of this list can be internally altered by time slot incentives, depending on the customer incentive sensitivity (see Section 4.2).

Appendix C

Hyperparameters for Random Forests and Neural Networks

Table C.1: Random forests hyperparameters found by exhaustive grid search with 5-fold cross validation (N is the total number of features).

Model ID	Number of trees	Split criterion	Max. tree depth	Min. samples for splitting	Min. samples for terminal node	Max. features for splitting
RFR (i) R	1000	MSE	20	2	1	N
RFR (i) RC	1000	MSE	20	3	1	N
RFR (ii) R	1000	MSE	15	5	3	$\sqrt{N}$
RFR (ii) RC	1000	MSE	17	5	3	$\sqrt{N}$
RFR (iii) R	1000	MSE	11	5	2	$\log(N)$
RFR (iii) RC	1000	MSE	13	2	3	$\log(N)$
RFR (ii) 1 vehicle	1000	MSE	19	2	1	N
RFR European retailer	1000	MSE	10	3	2	$\sqrt{N}$
RFR & IC European retailer	1000	MSE	12	2	1	N

Table C.2: Neural networks hyperparameters found by exhaustive grid search with 5-fold cross validation.

Model ID	Number of hidden layers	Number of nodes per hidden layer	Activation function	Weight optimization solver	Initial Learning rate	Weight updating	L_2 penalty
R	2	(10,10)	ReLU	Stochastic gradient descent	0.001	Constant	0.0001
RC	2	(10,10)	Hyperbolic tangent	Adam	0.001	N/A	0.0001

Appendix D

Feature Importance Analysis

Reporting feature scores directly is unreliable since the randomness in random forests can cause large variances in feature importance scores. In random forests, the importance score for all features is calculated using a Z score. To find feature importance, we cannot use this Z score directly, as it is not directly related to the statistical significance of feature importance since its distribution is not $N(0, 1)$ (Rudnicki et al., 2006). Therefore, we use the Boruta method for feature selection, which uses an iterative procedure of copying features and randomizing them to remove the correlation with the target. These new features are called "shadow features". The Boruta algorithm compares the performance of features with shadow features and calculates statistically significant Z scores. For a complete explanation of the Boruta algorithm, we refer to Kursa and Rudnicki (2010).

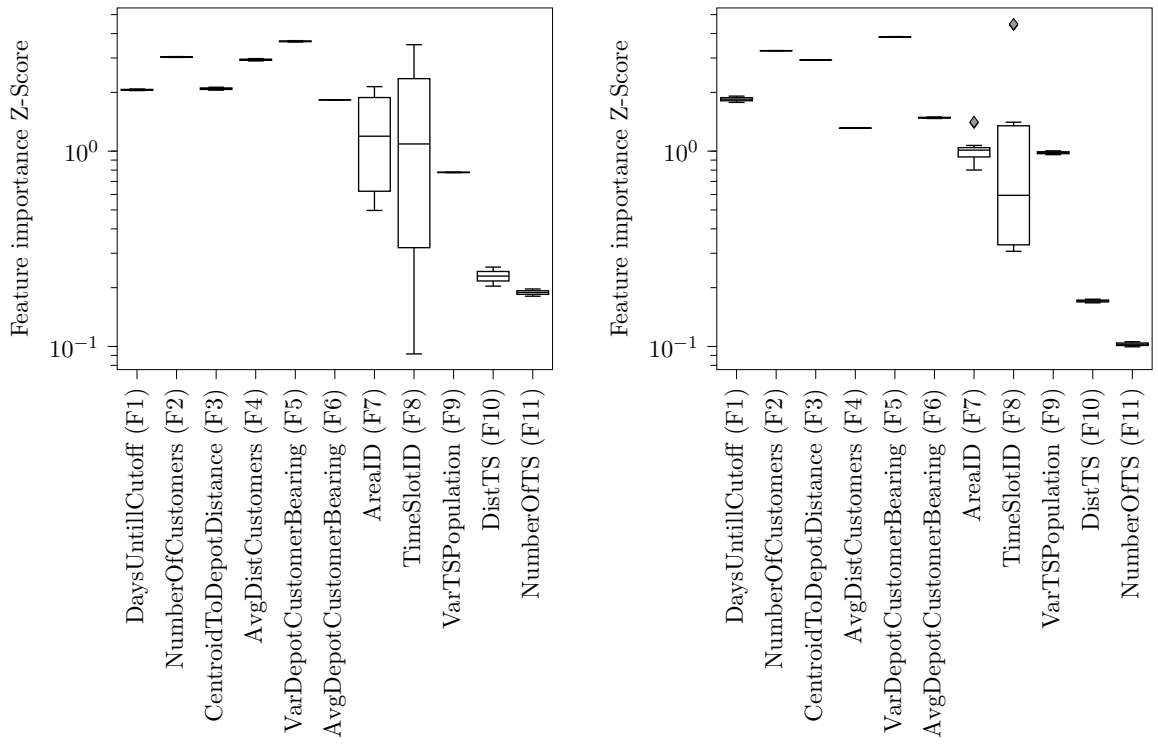

Figure D.1: Feature importance scores (Z-scores) from the Boruta algorithm with Shapley values as internal importance measure using 25 trials, plotted on a logarithmic scale. For both the R instances (left) and the RC instances (right), trained on the full booking horizon

We use a variant of the Boruta algorithm that uses Shapley values as internal importance measure, as using this statistic aids the process of finding global feature importance (Keany, 2020). Since we do not observe significant differences between the model trained on the complete booking horizon and the models trained on separate parts of the booking horizon, we only report the feature importance scores for the model trained on the complete booking horizon, for both the R and RC instances. Figure D.1 shows the feature importance scores. We observe that almost all features have high importance. Feature 10 (distance between booked time slots in an area) and Feature 11 (number of time slots booked in an area) show lower scores. The feature importance for both the area and time slot dummy features (F7 and F8) deviate, this is caused by the vector structure of these features, i.e., certain areas or time slots might be more important than others. For the R instances, the feature importance for Feature 7 and Feature 8 shows a larger deviation than for the RC instances, probably caused by the scattered nature of the R instance type. We conclude that we can leave all features in the model, since F10 and F11 still show high enough importance to have added value for the prediction.

Appendix E

Additional Experimental Results

Table E.1 shows the results of experiment 1 for the randomly scattered customers case with time constraints. The results are similar to the RC case: cheapest insertion (IC) performs slightly better than all RFR models. RFR (iii) seems the obtain the best results, with the lowest travel time and the largest difference between the cheapest and most expensive time slot nudging procedure.

Table E.1: Simulation run statistics on the randomly scattered instances with a time restriction (R-T).

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No TS	100% ±0%	N/A	N/A	N/A	9.06 ±0.11	0.38 ±0.05	6.97 ±0.12
No incentive	79.9% ±0.7%	4.3 ±0.02	N/A	N/A	17.61 ±0.29	0.74 ±0.05	14.83 ±0.33
IC Cheap	85.1% ±0.9%	4.8 ±0.1	79.9% ±0.8%	2.5 ±0.02	15.69 ±0.29	0.65 ±0.09	13.13 ±0.29
IC Expensive	76.3% ±0.3%	3.3 ±0.1	81.6% ±1.7%	2.5 ±0.01	18.91 ±0.13	0.59 ±0.07	16.11 ±0.11
RFR (i) Cheap	79.9% ±1.5%	3.7 ±0.1	75.9% ±2.3%	1.3 ±0.06	17.40 ±0.56	0.54 ±0.07	14.70 ±0.47
RFR (i) Expensive	79.0% ±1.2%	3.1 ±0.1	73.2% ±2.8%	2.1 ±0.1	17.74 ±0.50	0.53 ±0.07	15.12 ±0.61
RFR (ii) Cheap	79.9% ±1.6%	3.4 ±0.1	73.9% ±2.1%	1.6 ±0.1	17.53 ±0.54	0.53 ±0.04	14.96 ±0.49
RFR (ii) Expensive	79.9% ±1.2%	3.4 ±0.1	75.5% ±2.1%	1.7 ±0.1	17.76 ±0.45	0.54 ±0.06	15.19 ±0.44
RFR (iii) Cheap	76.6% ±0.1%	3.1 ±0.03	79.4% ±1.2%	2.2 ±0.07	16.34±0.3	0.64 ±0.07	15.37 ±0.4
RFR (iii) Expensive	76.6% ±0.1%	3.0 ±0.03	79.3% ±1.2%	2.3 ±0.07	18.63 ±0.3	0.71 ±0.09	16.73 ±0.4

Table E.2 shows the results for the randomly scattered instance with a capacity restriction. Similar to the RC instances, RFR shows structural better performance than the cheapest insertion costs. RFR (i) and RFR (ii) seem to obtain the best cost approximation.

Table E.2: Simulation run statistics on the randomly scattered instances with a capacity restriction (R-C).

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
No TS	66.7% ±0%	N/A	N/A	N/A	11.81 ±0.15	0.44 ±0.05	9.58 ±0.13
No incentive	69.6% ±0.5%	4.1 ±0.03	N/A	N/A	25.74 ±0.22	8.0 ±0.5	15.59 ±0.40
IC Cheap	69.6% ±0.4%	4.1 ±0.01	82.9% ±1.3%	2.5 ±0.04	24.79 ±0.22	4.9 ±0.3	15.20 ±0.15
IC Expensive	69.7% ±0%	3.8 ±0.03	83.1% ±1.1%	2.5 ±0.1	23.52 ±0.19	7.5 ±0.22	16.51 ±0.11
RFR (i) Cheap	70% ±0%	3.9 ±0	74.2% ±0.8%	1.1 ±0	23.99 ±0.29	2.22 ±0.66	16.15 ±0.22
RFR (i) Expensive	70% ±0%	3.5 ±0	72.1% ±1.8%	1.9 ±0.01	19.72 ±0.44	5.58 ±0.48	15.61 ±0.47
RFR (ii) Cheap	70% ±0%	3.7 ±0	71.7% ±2.2%	1.3 ±0.08	21.85 ±0.53	4.26 ±0.51	15.62 ±0.38
RFR (ii) Expensive	70% ±0%	3.7 ±0	77.3% ±1%	1.6 ±0.03	21.98 ±0.88	4.40 ±1.07	15.65 ±0.46
RFR (iii) Cheap	70% ±0.1%	3.7 ±0.03	79.4% ±1.2%	2.6 ±0.07	22.74 ±0.3	4.28 ±0.45	14.34 ±0.4
RFR (iii) Expensive	70% ±0.1%	3.5 ±0.03	81.3% ±1.2%	2.7 ±0.04	23.94 ±0.3	4.74 ±0.84	16.93 ±0.4

Table E.3 shows the results of the model when retrained on the case without time slots. To obtain feature values per time slot, we projected time slots on the routing schedule, e.g., customers that are served between 10:00 and 12:00 are assigned to that specific time slot, after which we can train our model on calculated feature values. The results of this approach show a slight improvement, but retraining the model again on actual realization data (see Section 7.2.3) seems to be of more value.

Table E.3: Simulation run statistics for the RFR (ii) model trained on data obtained from the lowest cost situation without time slots.

Offer Strategy	Planned customers (%)	Avg. number of feasible TS	Nudged customers (%)	Avg. TS change (abs)	Travel time/ customer (min.)	Waiting time/ customer (min.)	Distance/ customer (km)
RFR (ii) Cheap	80% ±0.7%	4.2 ±0	72.2% ±1%	2.5 ±0.03	17.33 ±0.16	0.46 ±0.07	14.76 ±0.25
RFR (ii) Expensive	79.9% ±0.6%	2.9 ±0	73.3% ±1%	2.5 ±0.03	17.46 ±0.23	0.68 ±0.09	14.6 ±0.25

UNIVERSITY
OF TWENTE.

ORTEC