Pre-Releasing Strategies for Same-Day Home Delivery

Comparing Heuristics for Partially Dynamic Vehicle Routing with Stochastic Customers

MSc. Thesis

Jan Bosma

Pre-Releasing Strategies for Same-Day Home Delivery

Comparing Heuristics for Partially Dynamic Vehicle Routing with Stochastic Customers

by

Jan Bosma

A thesis submitted to the Delft Institute of Applied Mathematics for the completion of the degree Master of Science in Applied Mathematics

at the Delft University of Technology,

to be defended publicly on June 9th 2023, 13:30.

Student number:	4667093	
Project duration:	October, 2022 – June, 2023	
Daily supervisor:	MSc. C. (Cynthia) Slotboom	ORTEC
Thesis committee:	Dr. F.M. (Fernando) de Oliveira Filho,	TU Delft
	Prof.dr. D.C. (Dion) Gijswijt,	TU Delft
	Prof.dr.ir. G. (Geurt) Jongbloed,	TU Delft
	MSc. C. (Cynthia) Slotboom,	ORTEC

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

This thesis investigates the optimization of Same-Day Delivery (SDD) in the context of a Dynamic and Stochastic Vehicle Routing Problem with Time-Windows (DSVRPTW). A central focus is the concept of *pre-releasing*; the process of assigning an order to a specific route and preparing it for delivery, effectively fixing the order to the route. This happens before all orders are known, restricting the possible routes and thus increasing the costs.

By conducting computational experiments using real-life data from a European e-grocery, the study evaluates various pre-releasing heuristics. The most effective heuristic, which delays pre-releasing until the last possible moment before the route departs, results in a reduction of optimization costs of 20% compared to current operations. When allowing more than 2 time-windows per truck, the reduction increases to 32%.

The research also addresses the critical practical constraint of pre-releasing capacity, which represents the maximum number of orders that can be pre-released within an hour. Simulation analysis reveals that using the last-minute heuristic only slightly increases the pre-releasing rate compared to the current operations of companies. To address this constraint efficiently, the most cost-effective solution is to invest in additional warehouse workforce. Alternatively, pre-releasing orders one hour early incurs a 1% increase in costs, while pre-releasing two hours early results in a 4% cost increase. Furthermore, initiating pre-releasing activities in the morning rather than the previous night can lead to savings of up to 2%.

The e-grocery expresses a preference for a constant pre-releasing rate equal to the pre-releasing capacity. This thesis proposes eight additional heuristics that determine which orders should be pre-released in addition to those identified by the last-minute heuristic. While the results from limited data were inconclusive, on average, the best heuristics incur an 8% higher cost compared to only pre-releasing at the last moment by pre-releasing orders based on proximity to or distance from the warehouse. Notably, pre-releasing orders with closer time-windows appeared to be preferable. Implementing this strategy would allow the pre-release capacity to be reduced from 200 to 150, resulting in a savings of five full-time pickers in the warehouse.

Future research opportunities include testing the methods on different cases and datasets and estimating the probability of exceeding the pre-releasing capacity, which could be used for deciding whether or not it is necessary to pre-release extra orders.

Contents

Preface

This thesis has been written to fulfill the requirements of the Master of Science program in Applied Mathematics at Delft University of Technology. I am grateful for the supervision and guidance provided by my academic supervisors, Fernando Oliveira Filho and Krszysztof Postek.

This research has been conducted in close collaboration with ORTEC, which offered invaluable support throughout the nine-month period, particularly from my industry supervisors, Cynthia Slotboom and Nicoleta Cenusa. I would like to extend my heartfelt gratitude to all ORTEC employees that have supported me during the course of this research, especially Arjen, Corne, Cynthia, Ernst, Frank, Joep, Maarten, Nienke, Quinten, Thomas and Tom. I felt really part of the team, and the fun activities like playing padel, the team weekends and carpooling together really made my time at ORTEC very special.

ORTEC has generously provided an intern allowance during the time of this project. Despite the financial incentive, I believe that the research presented in this thesis remains objective and has not been influenced by this financial support.

My personal goal for this project was to show to myself what I can accomplish if I try my very best. That goal has definitely been accomplished, and I'm very proud to share my hard work with you now. I hope you enjoy reading this thesis as much as I did creating it!

Delft, May 2023

CHAPTER 1

Introduction

"If you want to deliver fast, go alone.
If you want to deliver efficiently, go together.
If you want to deliver fast and efficiently, use mathematics"
∼ variation on African Proverb

1. Problem Motivation

1.1. Home Delivery. In the grocery shopping industry, the need for convenience has led to a growing trend of home delivery options. This shift is demonstrated by the success of Albert Heijn, the biggest online supermarket in the Netherlands, which witnessed a 35% surge in e-commerce sales in 2021, currently managing over 100,000 orders weekly with a fleet of more than 800 vehicles. This impressive feat is made possible by efficiently solving the complex Vehicle Routing Problem (VRP), which seeks to find the optimal routes in order to minimize factors such as total distance and costs.

The challenges posed by the VRP are compounded by pressure from competitors to reduce the costs, distance, time, and CO_2 emissions of their deliveries. This pressure is further driven by consumer preferences, with 55% of shoppers in the Netherlands indicating that they would switch to another retailer if they offer faster delivery. In addition, customers expect to be able to select a delivery time that suits their schedule; 27% of shoppers have abandoned a purchase due to a lack of fast delivery options [15].

1.2. Same-Day Delivery. Same-day delivery (SDD) is a service offered by online retailers that allows customers to receive their orders within a few hours of placing them. This is in contrast to next-day delivery, which is the current industry standard. The rapid growth of the online grocery market, especially during the recent pandemic, has led to increased demand for ultra-convenient delivery options like SDD. In fact, 99% of retailers are expected to offer same-day options within the next three years, with 35% already offering the service [15]. While same-day delivery may provide greater convenience for customers, it also poses challenges in terms of efficiency, cost, and environmental impact. For example, the last mile of the supply chain, which is the final leg of the journey from a fulfillment center to the customer's doorstep, is already a major challenge for next-day delivery and accounts for roughly half of total delivery costs. The added complications of same-day delivery may lead to increased vehicle miles traveled, delivery costs, emissions, and congestion.

1.3. Challenges in Optimizing SDD. The optimization of SDD poses unique challenges due to the continuous influx of new orders throughout the day, introducing stochastic and dynamic elements to the traditional VRP problem. In the context of this problem, "stochastic" refers to the inherent uncertainty regarding the volume and location of orders to be delivered, while "dynamic" describes the need for iterative or continuous solution methods to accommodate changes in the problem as new orders are received (or not received).

In addition to the stochastic and dynamic aspects, the SDD problem requires the consideration of practical constraints or "business rules". Examples of such rules include driving time restrictions, cut-off times for placing the orders, and the use of different types of vehicles. These rules can significantly further complicate the problem. The business rule that is central in this research is the *pre-releasing* capacity in the warehouse.

1.4. Pre-Releasing. A crucial aspect of the SDD problem is the limited pre-releasing capacity at a warehouse, which restricts the number of orders that can be picked simultaneously. *Pre-releasing* is the act of assigning an order to a specific route, which involves fixing the order to the route. In the warehouse, a pre-released order is placed on the loading dock assigned to a particular truck. Importantly, an order cannot be removed from or changed to another loading dock. This means that pre-releasing might create *regret* later, when new orders have come in. However, all orders need to be pre-released in time, ideally with efficient routes. In Figure 1.1, a visual representation of the pre-releasing process is shown; the map of orders on the left, pre-releasing into loading docks in the middle, and the incoming trucks on the right.

In general, the ideal scenario is to pre-release orders as late as possible, when the maximum amount of information about incoming orders is available. In practice however, the pre-releasing capacity is limited. To ensure that orders can be picked and prepared on time, they are often pre-released a few hours in advance as a precautionary measure, minimizing the risk of not being able to process all orders in time. This scenario presents a critical trade-off between waiting for more information to make better-informed decisions and minimizing the risk of pre-releasing orders too late, which could detrimentally impact the efficiency of SDD. This trade-off is essential in addressing the challenges of optimizing SDD and will be the topic of this research.

2. Problem Description

The pre-releasing problem examined in this research is complex and specific, with only a few articles in the literature tackling similar topics. Therefore, it is essential to provide a detailed definition and understanding of the problem. This thesis focuses on a real-life customer case from a European e-grocery company, hereafter referred to as "BosMart" to preserve anonymity. BosMart is interested in improving their Same Day Delivery (SDD) service, as their current operations lack efficiency and are reliant on manual planning. This section provides a conceptual overview of the pre-releasing problem at BosMart, followed by a detailed formulation of the input, objective function and decision variables. Additionally, a specific example is presented to enhance understanding and visualize the intricacies of the problem.

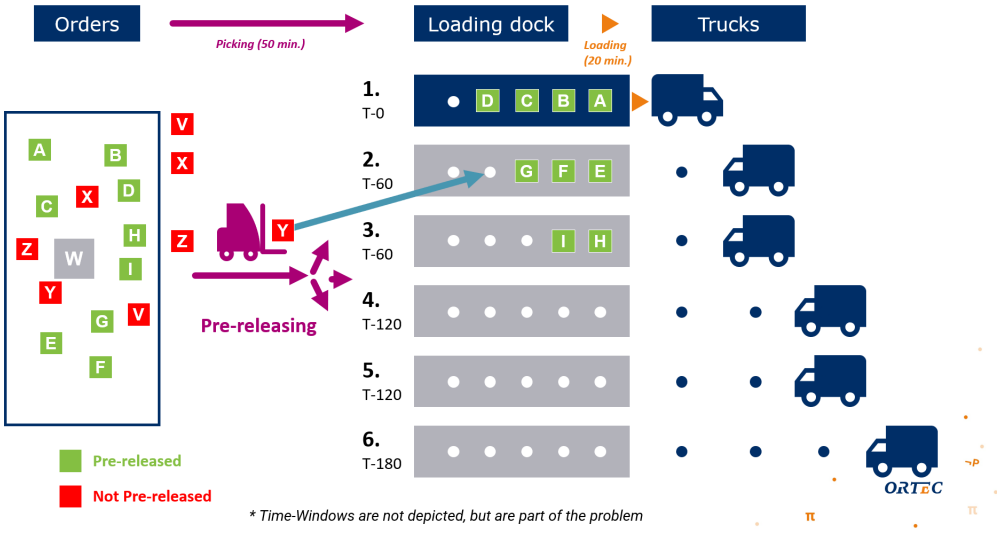

FIGURE 1.1. A visual representation of the pre-releasing problem, with the map of orders on the left, the orders being pre-released to a loading dock in the middle, and the incoming trucks on the right (blue dots represent the time left until departure). Order Y is being pre-released into loading dock 2.

2.1. Conceptual Explanation. BosMart operates as an online e-grocery retailer, offering customers the flexibility to choose a convenient 2-hour time-window for their orders to be delivered to their homes. Ensuring timely delivery while optimizing operational costs poses a significant challenge for BosMart.

In the warehouse, a team of pickers will collect the various products of the order and prepare them for delivery by placing them in a crate. This crate will then be moved to a loading dock, from which it will be loaded into a truck in the future. This process of assigning an order to a future truck is called pre-releasing. Crucially, once an order is pre-released to a particular truck, it cannot be reassigned to another truck in the future.

What complicates this process is that roughly half of the orders are placed less than 4 hours before their time-window starts. These orders cannot all be pre-released at the very last moment, as the pre-releasing capacity in the warehouse is limited. This means that parts of the routes need to be fixed (by means of pre-releasing) before all orders are known. This is the pre-releasing problem: determining which orders should be pre-released to minimize the expected total costs for the entire day.

2.2. The Input. Decisions are made based on three types of input data:

- **Order data**. For all orders that are placed before the optimization starts, the location (latitude and longitude), amounts (number of chilled, frozen and ambient boxes) and time-window (2-hour time-window) are known.

- **Route data**. A route is the path that a truck follows in a day. For all routes, the capacity, start and finish times, and costs (per kilometer, per hour and per route) are given.
- **VRP planning**. A commercial VRP solver (OHD) planned all orders and routes that were known in the previous optimization round earlier in the day. This planning also contains information on which orders are pre-released already.

2.3. The Objective Function. Pre-releasing decisions must be made throughout the day. The primary objective is to plan as many orders as possible, often aiming for full order fulfillment. The secondary objective is to minimize the costs, which is a weighted sum of the number of trucks used, the working time and the driving distance. To be precise, not the monetary costs need to be optimized, but the optimization costs, which can include various other factors, such as CO_2 emissions, driver happiness, and so on. The precise formula for optimization costs cannot be disclosed, but also is not relevant since the VRP is considered as a black box.

It is important to note that there is no clearly defined objective function for a single pre-releasing decision. Instead, the objective is to maximize order fulfillment and minimize costs over the entire day, in which multiple pre-releasing decisions are made. An attempt at an objective function for the individual pre-releasing problem could be the expected costs of future planning. However, due to the stochastic nature of future orders, this cannot be precisely defined. Hence, linear programming (LP) formulations of the pre-releasing problem are not useful, and therefore heuristic methods are required.

2.4. The Decision Variables. Two types of decisions need to be made at each decision moment:

- Determining which orders to **pre-release** to which routes. Pre-releasing an order to a route implies that the order will remain on that route in all future planning scenarios. The process of pre-releasing an order in the warehouse only takes a few minutes, but the number of orders that can be pre-released within an hour is constrained by the pre-releasing capacity.
- Deciding which routes to **finish**. Finalizing a route means that the route cannot be altered in the future, and thus will be part of the day's planning. A route can only be finalized when all orders on that route are pre-released. A route needs to be finish 20 minutes before it can depart from the warehouse.

2.5. The Timeline of the Day. At any point throughout the day, orders can be placed by a customer. This customer can select a 2-hour time-window, starting on the hour, from 06:00 to 22:00 (e.g. 06:00-08:00, 07:00-09:00 up to 22:00-00:00). In reality, time-windows start and end 10 minutes before the hour to ensure timely delivery and some time-windows are 1 hour, but these nuances are neglected in this research. Customers can only select a time-window starting in at least 2 hours in the future. For instance, at 05:59, the earliest available delivery window would be 08:00-10:00, while at 06:01, the earliest would be 09:00-11:00.

Pre-releasing decisions are made at predetermined moments throughout the day, such as every hour from 06:00 to 20:00. Trucks can depart ten minutes past every hour, from 07:10 until 21:10. Before departure, trucks need to be loaded,

which takes 20 minutes. All orders must be pre-released before they are loaded. Given that pre-releasing takes 50 minutes, 06:00 is the latest moment orders can be pre-released for routes departing at 07:10.

A timeline for a possible route is shown below in Figure 1.2. This route contains orders from 2 different time-windows: 12:00-14:00 and 13:00-15:00. This route is scheduled to depart at 11:10, so it needs to be loaded at 10:50. Therefore, the latest moment for pre-releasing its orders is at 10:00. At this point, no new orders for the time-window 12:00-14:00 can come in, but there is one more hour in which orders for time-window 13:00-15:00 can come in. After 15:00 the route returns to the warehouse. In principle, this route can be loaded again at 15:50, although it would be considered a new route in the optimization process. (In practice, planning routes with multiple "trips" would be desirable, but it is outside the scope of this thesis.)

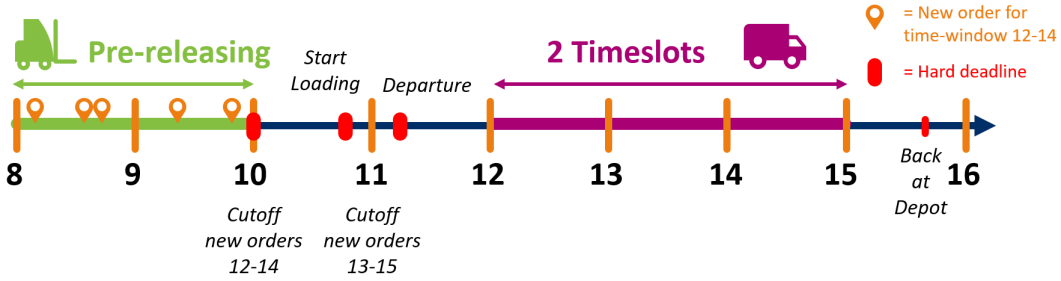

FIGURE 1.2. An example of a route containing orders with time-windows 12:00-14:00 and 13:00-15:00. All orders from this route need to be pre-released at (or before) 10:00.

3. Example of a Pre-Releasing Problem Instance

To improve the reader's understanding of the (complicated) pre-releasing problem, an example of a pre-releasing problem instance is shown in Figure 1.3. The question is which of the 11 known orders should be pre-released. Given that the pre-release capacity is 6, this can be done in $\sum_{i=0}^{6} \binom{11}{i} = 1486$ ways. For the real-life instances from BosMart considered in this thesis, up to 200 orders out of 2000 known orders can be pre-released, which can be done in many ways: $\sum_{i=0}^{200} \binom{2000}{i} = 7.7 \times 10^{280}$.

Looking at the example, it can be observed that the green route is planned to depart at 11:10, which means that loading should start at 10:50, which means that the latest moment to pre-release the orders for this route is 10:00. Therefore, a "Last-Minute" heuristic might suggest to at least pre-release these 3 orders, and finish the green route.

Then the question is if more orders should be pre-released or not. In general, pre-releasing more than necessary is bad for the VRP plannings, as the optimal routes might change when new orders come in. Therefore, the ideal scenario would be to only pre-release according to the "Last-Minute" heuristic.

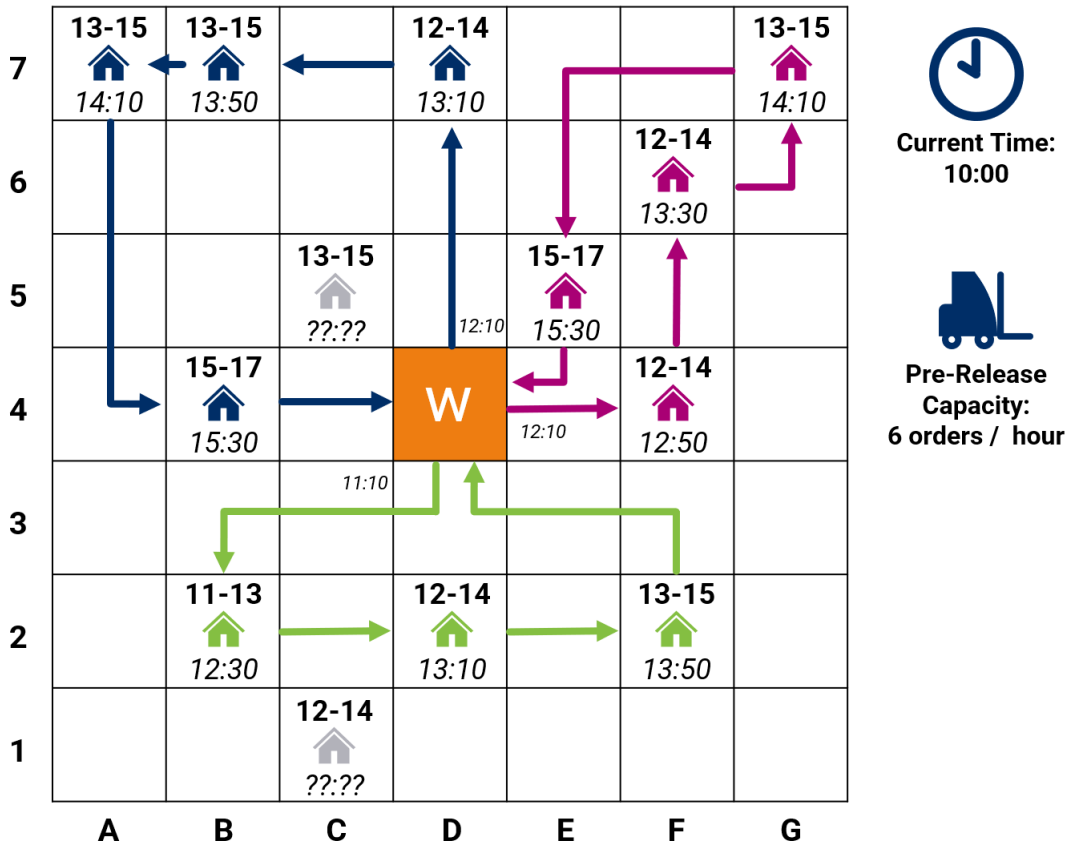

FIGURE 1.3. An example of a pre-releasing problem instance, with the warehouse in the middle and the orders are represented as houses, with their time-window above. OHD has planned the orders on 3 routes, indicated by the colors blue, purple and green. The grey orders (on C1 and C5) are not planned, because they have been placed after the previous optimization. Because they are not planned on a route, they can also not be pre-released. The scheduled arrival times are indicated below the houses, and the scheduled departure time is given at the start of the route. Driving time is assumed to be 20 minutes per square. None of the orders are pre-released already. The question is which 6 orders to pre-release at 10:00.

However, this strategy is not always possible, because the number of pre-releases is restricted by the pre-releasing capacity. In the example, the blue and purple routes are scheduled to depart at 12:10, which would mean that at 11:00 all of their 8 orders should be pre-released, which would exceed the pre-releasing capacity of 6. Therefore, it might be smart to use a heuristic to pre-release "Extra" orders, next

to the "Last-Minute" heuristic. If the pre-releasing capacity would be used fully in the example, 3 more orders should be pre-released.

An "Extra" pre-releasing heuristic should pre-release orders that are likely to stay on their current route, even when new orders come in and all routes are reoptimized. In the example, it seems logical to pre-release the orders at A7 and B7, as they are close to each other and have the same time-window. It is very unlikely that new orders come in which would result in a planning in which these orders are not on the same (blue) route. But which order should be the third to be pre-released?

The reader is encouraged to think about possible heuristics for themselves, and compare them to the heuristics suggested in Chapter 3.

4. Research Description

The goal of this research is to compare the effectiveness of various pre-releasing strategies in the case of BosMart specifically. The primary research question and three sub-questions are as follows:

- **Research Question:** *How should BosMart pre-release their orders and which procedural improvements are cost-effective?*
- **SQ1:** *What is the best pre-releasing heuristic?*
- **SQ2:** *What is the effect of the problem parameters on the optimization costs?*
- **SQ3:** *Which procedural improvements would be cost-effective?*

To answer these questions, a PRE-releasing Simulation TOol (PRESTO) was developed. PRESTO simulates the events of a day and at various points throughout the day applies a chosen heuristic to decide which orders will be pre-released and which routes will be finished. This approach enables the comparison of multiple heuristics under different problem parameter sets. The key problem parameters investigated are the pre-releasing capacity, the minimum time between pre-releasing and departure, the number of time windows per truck, and the optimization start times. In making these decisions, PRESTO utilizes the commercial Vehicle Routing Problem solver from ORTEC (OHD), which provides solutions for the VRP given the available orders and routes.

Several simplifying assumptions were made to model this pre-releasing problem. The following aspects are not within the scope of this thesis:

- **Solving the VRP.** The VRP solver OHD, which is a subroutine of PRESTO, is treated as a black box. While ORTEC has various sophisticated heuristics for solving a VRP, this research only employs the one used by BosMart in practice.
- **Stochastic simulation.** Due to limited data availability, this research was not able to make use of probability distributions to simulate potential future orders. This constraint precludes the use of forecasting models or data generation.
- **Determining the number of orders to pre-release.** As a consequence of not using stochastic simulation, it's not possible to estimate the probability of exceeding the pre-releasing capacity due to an unexpected surge in orders.
- **Planning with multiple trips per route.** All routes have a fixed earliest start time and a latest finish time. If a vehicle returns to the

warehouse well before the latest finish time, it must wait until the earliest start time of its next route. In practice, a route and its planning could encompass multiple "trips". A pre-releasing heuristic could, for example, plan a few short routes that return to the warehouse quickly, thereby increasing planning flexibility. All plannings in this thesis assume a "single trip" per route.

5. Contributions of this Research

This research makes several contributions to the existing literature on the Vehicle Routing Problem and online problems in general, and addresses novel challenges associated with pre-releasing in an online setting. The key contributions of this thesis are as follows:

(1) **Defining and Formulating Pre-Releasing in an Online Setting:** This study introduces and defines the concept of pre-releasing in the context of dynamic VRP with time-windows and stochastic customers, and considers the pre-releasing decision as a distinct problem seperate from the VRP.
(2) **Identification of Relevant Parameters:** The thesis identifies and examines the crucial parameters that impact the pre-releasing strategy's effectiveness and overall performance. This includes factors such as pre-releasing capacity, pre-releasing time, pre-releasing moments and the utilization of a time horizon.
(3) **Simulation Methodology and Benchmarks:** This research presents a comprehensive simulation method that enables the evaluation and comparison of various pre-releasing heuristics. Moreover, the thesis applies techniques from the literature on online problems to find benchmarks for comparing the performance of the heuristics, such as the OHD-In-Hindsight approach and random-based benchmarks. These benchmarks serve as crucial benchmarks for assessing the performance of the proposed heuristics.
(4) **Proposed Pre-Releasing Heuristics and Classification:** The thesis introduces 10 heuristics specifically designed for pre-releasing decisions. These heuristics are categorized into two main groups: "Last-Minute" heuristics, which determine routes to finish and pre-release associated orders, and "Extra" heuristics, which allow for additional pre-releasing beyond the last-minute decisions. The classification and analysis of these heuristics provide practical insights into their respective advantages and performance characteristics.
(5) **Performance Analysis and Parameter Sensitivity:** Through extensive experimentation and analysis, this research evaluates the performance of the proposed heuristics in different scenarios. Additionally, the study investigates the sensitivity of the pre-releasing strategy to parameter changes, offering insights into the robustness and adaptability of the proposed methods.
(6) **Recommendations for Practical Implementation:** Drawing on the research findings, this thesis provides practical recommendations for the application of pre-releasing heuristics in real-world logistics operations.

These recommendations guide practitioners in selecting appropriate heuristics, configuring relevant parameters, and integrating pre-releasing strategies into their existing delivery systems.

(7) **Applicability beyond VRP:** While the focus of this research is on pre-releasing in the VRP domain, the methods and insights presented in this study have broader applicability to other online problems. Problems that involve making decisions with partial information or fixed solutions before complete data is available, such as scheduling and resource allocation, may benefit from the methodologies and principles established in this research.

6. Thesis Outline and Reader's Guide

This thesis is organized into the following chapters:

(1) **Introduction**: This chapter briefly introduced the problem and its relevance and described the objectives and scope of the study.
(2) **Literature Review**: A review of relevant literature on VRP variations, Dynamic VRP, and Online Algorithms is presented, laying the foundation for the proposed methods and heuristics.
(3) **Proposed Heuristics**: A set of heuristics to be tested in this study are introduced, along with the rationale behind them.
(4) **Simulation Setup**: The simulation tool used to test the heuristics is described, along with a description of the data, performance metrics, and an overview of the different scenarios and parameter settings considered in the computational experiments.
(5) **Results and Analysis**: The results of the computational experiments are presented and analyzed, evaluating the performance of the heuristics and the effect of the problem parameters.
(6) **Conclusion and Recommendations**: The main findings of the study are summarized, practical implications are discussed, and directions for future research and potential improvements to the heuristics are suggested.

CHAPTER 2

Literature Review: Dynamic VRPs and Pre-Releasing

"It is impossible for a man to learn what he thinks he already knows."
∼ Epictetus

The rapid development of the logistics industry, combined with technological advancements, has spurred significant research interest in the Vehicle Routing Problem (VRP) and its numerous variations. Despite the wealth of literature, the sheer number of VRP variants and differing terminologies can make it challenging to obtain a clear overview of the field. Instead of describing the VRP landscape as a whole, this chapter aims to contextualize the Dynamic Stochastic Vehicle Routing Problem with Time-Windows (DSVRPTW) problem by providing a structured overview of the following topics:

- the standard VRP (Section 1)
- related VRP variations: the Capacitated VRP (Section 2) and the VRP with Time-Windows (Section 3))
- Dynamical VRPs (Section 4)
- Heuristics for Dynamic VRPs in Literature (Section 5)
- Online algorithms (Section 6)
- Literature on pre-releasing (Section 7)

Although pre-releasing itself is a little-studied topic, the existing literature discussed in this chapter has served as inspiration for developing and evaluating the pre-releasing heuristics that will be detailed in Chapter 3.

1. Standard Vehicle Routing Problem Formulation

The Vehicle Routing Problem (VRP), a combinatorial optimization problem, was first introduced by Dantzig and Ramser in 1959 [5]. The primary objective of the VRP is to determine the most efficient routes for a fleet of vehicles to deliver goods to a group of customers while minimizing the overall travel distance or cost. With a wide array of practical applications in transportation, logistics, and supply chain management, the VRP is a critical factor in enhancing the efficiency of these industries [20, 12].

The standard VRP can be represented by a complete, bidirected graph $G = (V, E)$, where $V = \{0, 1, \dots, n\}$ is the set of nodes, with node 0 representing the depot and nodes $1, \dots, n$ representing the customers. The set of arcs E corresponds to the connections between the nodes, and each arc $(i, j) \in E$ is associated with a non-negative cost c_{ij}, symbolizing the distance or travel time between nodes i and j. The VRP's objective is to identify a collection of vehicle routes that begin and

end at the depot, visit each customer exactly once, and minimize the total route cost.

The problem can be modeled as an ILP, where binary variable x_{ijk} is 1 if the arc from node i to node j is in the route driven by vehicle k, and 0 otherwise. The number of vehicles available for routing is K.

$$\text{minimize} \quad \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=1}^{n} c_{ij} x_{ijk} \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik}, \quad \forall j \in \{1, \dots, n\},\ k \in \{1, \dots, K\} \tag{2}$$

$$\sum_{k=1}^{K} \sum_{i=0}^{n} x_{ijk} = 1, \quad j \in \{1, \dots, n\} \tag{3}$$

$$\sum_{j=0}^{n} x_{0jk} = 1, \quad k \in \{1, \dots, K\} \tag{4}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1, \quad \forall S \subset \{1, \dots, n\}, S \neq \emptyset, k \in \{1, \dots, K\} \tag{5}$$

$$x_{ijk} \in \{0, 1\}, \quad i, j \in \{0, \dots, n\}, k \in \{1, \dots, K\} \tag{6}$$

This formulation can be explained as follows:

(1) The objective function (1): aims to minimize the total cost of all routes. In this formulation, the costs of a route equals the sum of the cost of the arcs that are used.
(2) Vehicle leaves node that it enters (Eq. 2): This constraint ensures that the number of times a vehicle enters a node is equal to the number of times it leaves that node. This guarantees that if a vehicle visits a customer node, it will also depart from it.
(3) Ensure that every node is entered once (Eq. 3): Combined with the first constraint, this constraint ensures that every node is entered only once, and it is left by the same vehicle. In other words, each customer is visited exactly once by a single vehicle.
(4) Every vehicle leaves the depot (Eq. 4): This constraint guarantees that each vehicle leaves the depot exactly once. Together with constraints 1 and 2, this ensures that every vehicle departs from and returns to the depot, forming a complete route. Note that if a route k is not used, $x_{00k} = 1$.
(5) Subtour elimination constraints (Eq. 5): In order to eliminate subtours in the VRP solution, the Dantzig-Fulkerson-Johnson (DFJ) subtour elimination constraint can be included in the problem formulation. This constraint guarantees that the total number of arcs entering and leaving nodes in any subset S of customer nodes by any vehicle is less than or equal to the number of nodes in S minus one. As a result, the solution cannot form a complete loop within the subset S, preventing the formation of subtours and ensuring feasible solutions.
(6) Binary decision variables (Eq. 6): This constraint defines the binary nature of the decision variables x_{ijk}, meaning that the variable takes the

value 1 if the arc from node i to node j is included in the route for vehicle k, and 0 otherwise.

2. Capacitated Vehicle Routing Problem (CVRP)

In a variant of the VRP, known as the Capacitated Vehicle Routing Problem (CVRP), vehicles have a limited capacity, and the demands of the customers they serve should not exceed their capacity. To include capacity constraints in the formulation, the following additional notation is introduced:

- q_j is the demand of each customer.
- Q_k is the capacity of vehicle k.

The capacity constraint can be added to the problem formulation as follows:

$$\sum_{j=1}^{n} q_j x_{0jk} \leq Q_k, \quad k = 1, \ldots, K \tag{7}$$

This constraint ensures that the total demand of all customers served by vehicle k does not exceed its capacity, Q_k. This makes the problem more realistic, as it considers the limited capacity of vehicles and the need to distribute the demand among multiple vehicles when necessary. Some problems also incorporate the loading, for example when considering the dimensions of the box, but this is not the case in this thesis.

3. Vehicle Routing Problem with Time Windows (VRPTW)

The Vehicle Routing Problem with Time Windows (VRPTW) is an extension of the Capacitated Vehicle Routing Problem (CVRP) where each customer has a specific time window within which the goods must be delivered. In this problem, the constraints automatically eliminate subtours, so there is no need to formulate any subtour elimination constraints. In this section, we will discuss the additional constraints required for the VRPTW formulation and explain them conceptually.

3.1. Time Window Constraints. To formulate a VRPTW, some extra parameters and variables are needed:

- $[a_i, b_i]$ is the time window of customer i. A vehicle must arrive at customer i at least at a_i and at most at b_i.
- t_{ij} denotes the time it takes to get from customer i to customer j. Any service time at customer i is included.
- variable s_i denotes the time that a vehicle starts serving customer i (which must be between a_i and b_i).

These constraints can be formulated as follows:

$$s_i + t_{ij} - M \cdot (1 - x_{ijk}) \leq s_j, \quad \forall i \in V,\ j \in V \quad 1,\ k \in 1, \ldots, p \tag{8}$$

$$a_i \leq s_i \leq b_i, \quad \forall i \in V \tag{9}$$

where $M = \max b_i + t_{ij} - a_i$ for all $i, j \in 1, \ldots, n$.

Constraint (8) ensures that the vehicle's service start time at customer j is at least t_{ij} later than the start of the service time at customer i. If the arc (i, j) is in the route, the constraint can be rewritten to $s_i + t_{ij} \leq s_j$. If the arc (i, j) is not in the route, the constraint is still valid, and it can be rewritten to $s_i + M \geq s_j$.

The value of M is the maximum value of $b_i + t_{ij} - a_i$, representing the maximum possible time between a_i and b_i.

Constraint (9) ensures that a vehicle can start serving a customer within the customer's time window.

The VRPTW formulation ensures that it is not possible to return to a previously served customer. This is because the time a customer is being served is always later than the previously served customers. As a result, the time window constraints automatically eliminate subtours, making it unnecessary to formulate separate subtour elimination constraints, such as the DFJ constraints.

3.2. Stochastic VRP. In real-world applications, many factors related to the Capacitated Vehicle Routing Problem (CVRP) are uncertain. To capture this variability, certain parameters in the CVRP can be modeled as stochastic variables, leading to the Stochastic (Capacitated) Vehicle Routing Problem (SVRP or SCVRP).

In SVRP, elements such as demand, customer presence, travel times, and service times can be uncertain. The most common version of SVRP is the Capacitated Vehicle Routing Problem with Stochastic Demand (CVRPSD), which is highly relevant to the real-world scenarios and aligns with the focus of this research on pre-releasing, as it also deals with uncertainty in customer demand.

Unlike deterministic VRP formulations, in SVRP, the decision-maker (DM) must make decisions before knowing the exact values of all parameters. As a result, once the actual parameter values are known, certain constraints may be violated, potentially leading to a solution "failing."

Stochastic problems are often modeled as either a Chance Constrained Program (CCP) or a Stochastic Program with Recourse (SPR). In CCP, the problem is solved to ensure the probability of route failure remains below a specified level. In SPR, the DM is allowed to make corrective decisions after observing the realization of the stochastic parameters, aiming to minimize the expected cost of decisions, including the initial decisions and any subsequent actions taken upon observing the actual parameter values.

4. Dynamical Vehicle Routing Problems

In practical situations, delivery problems are predominantly dynamic vehicle routing problems (DVRPs) as opposed to static ones. The key difference between these two types lies in the availability of order information. In static VRPs, all orders are known beforehand, whereas in DVRPs, only a portion of the orders are initially available for generating a preliminary schedule. As the vehicles proceed on their routes, new orders are dynamically introduced, necessitating adjustments to the existing route to accommodate these additions. This characteristic makes DVRPs more applicable and realistic for various practical situations, including delivery services, emergency response, and transportation networks.

To address a dynamic problem, it is essential to simulate dynamicity. A widely used method, as described by Kilby [13] and Montemanni [14], involves dividing the working day into time slices and incrementally solving the problems. This method introduces the concept of a working day lasting T_{wd} seconds, which the algorithm simulates. Initially, not all nodes are available; a subset is assigned an availability time when they can be accessed. The degree of dynamicity in the problem is determined by this percentage. At the beginning of the day, a preliminary tour is

established using the available nodes. The working day is then partitioned into n_{ts} time slices of duration t_{ts}, with t_{ts} being equal to T_{wd}/n_{ts}. The solution is updated at each time slice, effectively converting the dynamic problem into n_{ts} consecutive static problems to be solved. In DVRPTW, the objective is akin to that of static VRPs; however, some customers and their time windows are initially unknown, and portions of the solutions may already be committed.

In the remainder of this section, we will provide a comprehensive overview of different DVRP types, discuss recent advancements in the field, and concentrate specifically on the Dynamic VRP with Stochastic Customers and Time Windows (DSVRPTW).

4.1. Overview of Dynamic VRPs. DVRPs can be broadly classified according to the dynamic elements they involve, such as customer demands, travel times, and service times, among others. Recent research on DVRPs has centered on devising efficient algorithms and heuristic methods for solving various types of DVRPs, as well as evaluating the effects of dynamic elements on routing strategy performance. A taxonomy of Dynamical VRPs was recently given by Psaraftis [16].

Some dynamic variations of the VRP that are relevant to this work are:

(1) Dynamic VRP with Stochastic Customers and Time Windows (DSVRPTW): In this variant, customer requests (orders) are disclosed in real-time or dynamically during the operation. Each customer has a time window during which service must be provided. The DVRPTW is relevant to delivery or pick-up services that need to accommodate last-minute requests [2]. This is the type of dynamicity that is relevant for this research, and it will be further explained in the next section.
(2) Dynamic VRP with Stochastic Demands (DVRPSD): This problem deals with customer demands that are unknown in advance and are disclosed dynamically as vehicles visit customers. This variant is applicable to situations where customer orders are uncertain until the time of delivery [3].
(3) Dynamic VRP with Stochastic Travel Times (DVRPSTT): In this problem, travel times between locations are uncertain and are disclosed dynamically during the operation. The DVRPSTT accounts for factors such as traffic congestion and road conditions, which can impact the efficiency of routing decisions [11].
(4) Dynamic VRP with Stochastic Service Times (DVRPSST): This problem involves uncertain service times for customers that are disclosed dynamically during the operation. The DVRPSST is applicable to situations where the time required to service a customer is not known in advance, such as vehicle breakdowns, repair services, or uncertain processing times [7].

4.2. Dynamic VRP with Stochastic Customers and Time Windows (DSVRPTW). The Dynamic VRP with Stochastic Customers and Time Windows (DSVRPTW) is an extension of the VRPTW, where the presence of customers is uncertain. In this problem, each customer has a specific time window within which the goods must be delivered, as well as a probability of requiring service. The goal is to minimize the total travel cost while considering the constraints imposed by the time windows and the stochastic nature of the customers.

The DSVRPTW combines the challenges of both the VRPTW and the SVRP, requiring consideration of time windows and the stochastic presence of customers. Due to the increased complexity of this problem, finding optimal solutions can be computationally challenging, particularly for large instances. However, as with the SVRP, a range of solution methodologies, such as metaheuristics or exact methods, can be employed to tackle this problem.

Recent work on the DSVRPTW has focused on developing efficient algorithms and heuristics to handle the combined uncertainties in customer requests, demands, and time windows. Christiansen et al. [19] proposed a two-stage stochastic programming model for the DSVRPTW and developed a sample average approximation algorithm to solve the problem. Their approach showed promising results in terms of computational efficiency and solution quality.

5. Heuristics for Dynamic VRPs in Literature

Vehicle Routing Problems (VRP) and Vehicle Routing Problems with Time Windows (VRPTW) are both classified as NP-hard, as they generalize the NP-hard traveling salesman problem. Consequently, heuristic algorithms are extensively employed to address these problems. Classical examples encompass the nearest neighbor heuristic proposed by Flood [9] and the savings algorithm developed by Clarke and Wright [4], which is grounded on the savings concept and persistently combines two customers on an identical route.

In recent years, meta-heuristics have gained prominence in the field. Semet and Taillard [18] introduced a tabu search for discovering effective solutions to the VRP, whereas Baker and Ayechew [1] combined genetic algorithms and neighborhood search methods to yield satisfactory results. Dorigo and Gambardella [6] presented ant colony optimization, which leverages artificial ant colonies to construct the shortest route.

Despite the multitude of static VRP solvers available, only a limited number of algorithms are capable of tackling dynamic VRPs. Fundamentally, most algorithms previously described can be adapted to solve dynamic VRPs. Nonetheless, to effectively manage the dynamic aspects of the problem, the algorithm should incorporate mechanisms that facilitate the reuse of learned features of the problem from previous solutions. As indicated in Eyckelhof and Snoek [8], certain bio-mimetic ant-colony optimization algorithms seem to support dynamic adaptations of delivery routes effectively. For instance, in ant colony optimization, virtual pheromone trails are created to indicate advantageous directions if solutions only need partial modifications.

Yang et al.'s work on "Dynamic vehicle routing with time windows in theory and practice" [21] presents a heuristic approach for handling the dynamic aspects of vehicle routing with time windows, employing ant colony optimization (ACO) as their primary solution technique. Their heuristic is designed around the concept of committing to specific routes and customer assignments at the latest possible moment, with the goal of maximizing the flexibility of the solution in response to the dynamic nature of the problem.

The key idea behind their approach is to maintain as many available options as possible for as long as possible. By doing so, the algorithm is better equipped to accommodate new orders or changes that may arise during the execution of

the solution. In their ACO-based heuristic, artificial ants construct solutions incrementally by traversing a graph representation of the problem. The ants utilize pheromone trails to indicate favorable routes and customer assignments, which are updated dynamically to reflect the evolving problem landscape.

Yang et al.'s commitment strategy involves deferring the decision to commit to a particular customer assignment or route until it is absolutely necessary. This allows the ants to explore a wider range of potential solutions, increasing the likelihood of finding high-quality routes that can adapt to the dynamic nature of the problem. By focusing on maintaining flexibility, their heuristic can effectively navigate the challenges presented by dynamic vehicle routing problems with time windows. This concept inspired the idea of "Last-Minute" heuristics for the pre-releasing problem, as described in Chapter 3.

5.1. Two-stage planning. Two-stage stochastic programming is a technique employed to address optimization problems with uncertainty. The objective is to iteratively select the best value for a decision variable, x, while accounting for both the current cost of choosing x (first stage, denoted by $f(x)$) and the expected future costs of this choice (second stage, denoted by $E[Q(x, \xi)]$).

In this approach, it is assumed that the uncertain data, ξ, in the second stage has a probability distribution(which is not the case in this thesis). The expected future costs can be formulated as the summation of costs for all possible scenarios in ξ, with each scenario having an associated probability p_k:

$$E[Q(x, \xi)] = \sum_{k=1}^{K} p_k \cdot Q(x, \xi_k) \tag{10}$$

Considering the application of the two-stage stochastic programming approach to the Stochastic Vehicle Routing Problem (SVRP), several variables need to be taken into account: $X_{\text{pre-released}}$ represents the ordered set of previous choices of x, $X_{\text{available}}$ denotes the set of available nodes to select as the new choice of x, and X_{future} accounts for the uncertainty of nodes that might become available in the future.

To determine the score for both stages, one first needs a simple score for the first stage: the total costs if node x is added to the current tour. For the second stage, a set of scenarios ξ_k with associated probabilities p_k is required. Each scenario consists of a set of nodes, X_{future}. Assuming a simulator that generates the set X_{future} and assigns appropriate probabilities, a method is needed to create a tour starting at node x and incorporating all nodes in X_{future}. This can be achieved using any algorithm capable of solving the static VRP, such as ORTEC's commercial solver OHD. The algorithm should run for a limited time and return the best solution found. The score of this solution, $Q(x, \xi_k)$, corresponds to the k-th scenario.

In order to compute the expected score of choosing x, a sufficiently large set of k scenarios must be generated. A larger k value improves the accuracy of the expected score but increases computational costs. Thus, it is crucial to choose an appropriate k value and employ a VRP-solving algorithm with low time requirements.

The fact that no probability distribution on the future orders is given prevents the use of two-stage planning in this thesis. However, the multi-simulation heuristic suggested in Chapter 3 is inspired by two-stage planning.

6. Online Algorithms

Online algorithms are a class of algorithms designed to make decisions and produce outputs based on partial input data, without complete knowledge of the entire input sequence. These algorithms are especially relevant for dynamic and real-time problems, including vehicle routing problems, where input data (such as customer orders, travel times, or demands) might change or become available incrementally over time. This section provides an overview of online algorithms, discusses the relationship between dynamic vehicle routing problems and online algorithms, and reviews some recent work on this topic. Performance analysis of online algorithms will also be investigated, as this will inspire the development of the OHD-In-Hindsight benchmark for pre-releasing.

6.1. Introduction to Online Algorithms. Online algorithms process input data elements sequentially, making decisions at each step based on the available information at that point in time. This is in contrast to offline algorithms, which have access to the complete input data set before processing and can make decisions based on full knowledge of the problem instance. Online algorithms are well-suited to address dynamic and real-time problems, as they can adapt to changes in input data or problem parameters on-the-fly.

Formally, an online algorithm can be described as follows. Let $I = I_1, I_2, \ldots, I_n$ be the input sequence. An online algorithm $\mathcal{A}$ processes each input element I_i and produces an output O_i based on the current knowledge. The algorithm cannot change any output O_j with $j < i$ once it has been produced. The performance of online algorithms is typically evaluated by comparing their output to the output of an optimal offline algorithm.

In the context of the pre-releasing problem, an online algorithm $\mathcal{A}$ must decide, at each time step, whether to pre-release an order based on the available information (e.g., current orders, loading docks, and time windows). The objective is to deliver all orders within their specified time windows while minimizing the overall cost, which may include travel time, waiting time, and other relevant operational costs. As the algorithm processes orders sequentially, it must adapt its decisions to account for changes in customer demand, time windows, and other dynamic elements.

6.2. Competitive Analysis of Online Algorithms. Competitive analysis is a widely used method for evaluating the performance of online algorithms. The idea of competitiveness is to compare the output generated by an online algorithm to the output produced by an optimal offline algorithm, which is an omniscient algorithm that knows the entire input data in advance and can compute an optimal output. The better an online algorithm approximates the optimal solution, the more competitive it is.

The performance of an online algorithm $\mathcal{A}$ on an input sequence I can be quantified by its *competitive ratio*, which is defined as the ratio of the cost of the solution produced by $\mathcal{A}$ to the cost of the optimal offline solution:

$$\rho(\mathcal{A}, I) = \frac{\text{cost}(\mathcal{A}(I))}{\text{cost}(\text{OPT}(I))} \tag{11}$$

where $\text{cost}(\mathcal{A}(I))$ is the cost of the solution produced by the online algorithm $\mathcal{A}$ on input sequence I, and $\text{cost}(\text{OPT}(I))$ is the cost of the optimal offline solution.

The competitive ratio measures the worst-case performance of the online algorithm over all possible input sequences:

$$\rho(\mathcal{A}) = \sup_{I} \rho(\mathcal{A}, I) \tag{12}$$

An online algorithm is said to be *c-competitive* if its competitive ratio is at most c. The goal in designing online algorithms is to achieve the smallest possible competitive ratio, indicating that the online algorithm performs well compared to the optimal offline algorithm, even in the worst-case scenario.

6.3. Recent Work on Online Algorithms. The study of online algorithms has been an active area of research in recent years, with various approaches proposed to tackle different classes of problems. In the context of vehicle routing problems, researchers have focused on designing online algorithms that adapt to dynamic changes in customer orders, time windows, and other problem parameters.

For example, Gendreau [10] provided a comprehensive review of online algorithms for dynamic vehicle routing problems, including algorithms for the Dynamic Traveling Salesman Problem (DTSP), the Dynamic Vehicle Routing Problem (DVRP), and the Dynamic Pickup and Delivery Problem (DPDP). They discussed various strategies, such as insertion heuristics, local search, metaheuristics, and exact algorithms, for solving these problems in an online setting.

Bertsimas [3] proposed an online algorithm for the Dynamic Vehicle Routing Problem with Time Windows (DVRPTW) based on an adaptive large neighborhood search (ALNS) heuristic. The algorithm dynamically updates the search neighborhood based on the current solution and explores different routes in an online manner to adapt to changes in customer orders and time windows. They showed that their algorithm performs well in practice, achieving near-optimal solutions for various benchmark instances.

6.4. Dynamical VRP with Stochastic Customers and Time Windows. In an online setting, the DVRPSTW requires the algorithm to make decisions about the routes without full knowledge of future customer orders, demands, and time windows. As new information becomes available, the online algorithm must adapt its routes to ensure that all customer demands are met within their respective time windows while minimizing the expected total cost. This problem combines the challenges of both dynamic and stochastic aspects, making it a complex and interesting problem to study within the context of online algorithms.

6.5. Pre-Releasing and Online Algorithms. The concept of pre-releasing, as described earlier, involves placing an order in a loading dock, where it will be picked up by a truck. Once an order is pre-released, a part of the solution is fixed, and it cannot be changed or removed from the loading dock. This problem can be viewed as an online algorithm, as decisions about pre-releasing orders must be made without full knowledge of future orders that may arrive within the planning horizon.

7. Literature on Pre-Releasing

Although there is extensive research on the dynamic VRP, little attention has been given to practical warehouse problems like pre-releasing. In fact, the first paper addressing both dynamicity and time-windows simultaneously was only recently

published [21]. This paper serves as a major source of inspiration for the current research and is summarized below.

Yang et al.'s work is concerned with service delivery rather than goods transportation. They propose a heuristic method that commits to decisions at the latest feasible moment, maximizing adaptability to the dynamic aspects of the problem. By maintaining numerous options, the algorithm can efficiently address emerging orders or modifications. The authors employ ant colony optimization to tackle the dynamic aspects of the problem and design a robust solution strategy suitable for complex service delivery scenarios.

The concept of "committing" an order to a route is similar to the concept of "pre-releasing" discussed in this research. However, there are two major differences:

- "Committing" can be performed on-the-fly, whereas pre-releasing must be executed (far) before the start of the time-window. This difference arises because routes can only be finished and loaded at the depot (a few hours before delivery) whereas service-delivery vehicles need not return to the depot.
- Yang et al.'s heuristic primarily focuses on the routing aspect and employs a simple last-minute "strategy" for pre-releasing. In contrast, this thesis uses a black-box algorithm for the routing part and emphasizes the development of pre-releasing heuristics. In particular, orders will be pre-released before the last-minute to manage the pre-releasing capacity.

CHAPTER 3

Proposed Heuristics

"Think of many things - do only one." ∼ *Portuguese Proverb*

After having gained an understanding of the existing literature on VRPs and online algorithms, this chapter presents the heuristics for the pre-releasing problem suggested in this thesis, which will also be evaluated in the simulation tool PRESTO.

A pre-releasing heuristic should output which orders to pre-release and which routes to finish, based on the available information, including all known orders and routes at that moment, as well as an optimized VRP planning.

The pre-releasing heuristics are categorized into two types: "Last-Minute" heuristics and "Extra-Pre-Release" heuristics. The "Last-Minute" heuristics focus on determining the routes to finish and subsequently pre-releasing all orders within those routes. Failure to apply the "Last-Minute" heuristic may result in untimely pre-releases, leading to potential delivery delays. This chapter presents two "Last-Minute" heuristics: "upcoming orders," which identifies routes containing orders with time-windows starting within 2 hours, and "upcoming routes," which identifies routes scheduled to depart before the next optimization time.

On top of the pre-releases determined by the "Last-Minute" heuristic, it may be necessary to consider pre-releasing additional orders, up to the pre-releasing capacity, using an "Extra-Pre-Release" heuristic. This allows for the creation of a buffer to account for contingencies, such as warehouse issues or unexpected surges in orders that may exceed the finite pre-releasing capacity. In Section 2, eight different "Extra-Pre-Release" heuristics are suggested:

(1) **Random**: Pre-release randomly chosen orders. This is expected to be suboptimal, but provides a useful point of reference for the other heuristics.
(2) **Closest**: Pre-release the orders that are the closest to the warehouse, i.e. have the smallest distance to the warehouse.
(3) **Furthest**: Pre-release the orders that are furthest from the warehouse, i.e. have the largest distance to the warehouse.
(4) **Minimum Driving Time**: Pre-release the two orders in the VRP planning with the shortest driving time between them, if they are not already pre-released.
(5) **Maximum Distance**: Pre-release the order that has the highest distance to any other pre-released order (or the warehouse).
(6) **Furthest Seeding**: Pre-release the furthest s orders in each route, and then apply the Minimum Driving Time heuristic.
(7) **Closest Seeding**: Pre-release the closest s orders in each route, and then apply the Minimum Driving Time heuristic.

(8) **Next Time-Window**: Pre-release orders in a lexicographical order, prioritizing those with the closest time-window start, and then applying the Minimum Driving Time heuristic to determine which order to pre-release within each time-window.

It is important to note that when an order is chosen to be pre-released, it is assigned to the route it is planned on in the most recent optimized VRP planning.

To prevent pre-releasing orders from routes departing later in the day, these heuristics are applied within a predetermined pre-release horizon h, denoted as PR_Horizon. This limits the pre-releasing options to only the orders that are part of the next $h + 1$ time-windows.

At the end of this chapter, several additional ideas for more advanced pre-releasing strategies are proposed, providing potential avenues for further exploration.

1. Last-Minute Heuristics

The Last-Minute heuristics decide which routes needs to be finished, and consequently pre-releases all orders from that route. The idea behind these heuristics is that pre-releasing at the last moment maximizes the information available, so that the best decisions can be made, similar to the "committing" strategy suggested by Yang [21] discussed in the previous chapter. When the pre-releasing capacity is of no concern, the Last-Minute heuristic is sufficient, and no "Extra" Pre-Release heuristic is needed.

One way to finish routes at the last-minute is to finish a route if it contains an order with a time-window starting in 2 hours. For example at 10:00, all routes with an order with time-window 12:00-14:00 would be pre-released. The idea is that the truck needs to be loaded at 10:50 and pre-releasing takes 50 minutes, so 10:00 is the last moment to pre-release this order. This is called the "upcoming orders" heuristic. This idea stems from BosMart's current operations, where all orders have a label that suggest when the order needs to be loaded in order to be delivered at the start of the time-window.

However, a smarter heuristic exists; the "upcoming routes" heuristic. The flaw in the reasoning above is that an order with time-window 12:00-14:00 might only be planned to be delivered after 13:00, in which case it could have been pre-released an hour later. Therefore, the upcoming routes heuristic considers which routes to finish before the next optimization start time. To be precise, the upcoming route heuristic finishes a route if the scheduled loading time plus the time it takes to pre-release is larger than next optimization start time.

Consider the route depicted in Figure 3.1 below. The upcoming orders heuristic would finish this route at 10:00, but the upcoming routes heuristic would finish it at 11:00. Finishing the route at 11:00 would mean loading could start at 11:50 and the route departs at 12:10. This example shows why the upcoming route heuristic is preferred over the upcoming orders heuristic, as the simulations will confirm in Chapter 5. In the rest of this thesis, when referring to the "Last-Minute" heuristic, it is understood that this is the "upcoming routes" heuristic.

2. Extra-Pre-Release Heuristics

As discussed previously, it might be desirable to pre-release extra orders after applying the Last-Minute heuristic, in order to mitigate the risks of exceeding the

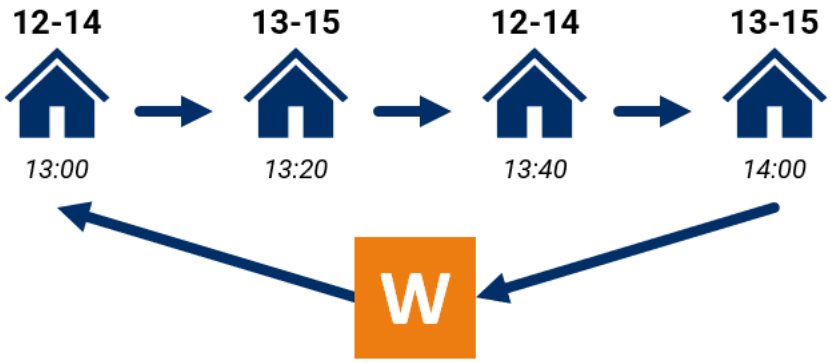

FIGURE 3.1. A route from the optimized VRP planning, containing 4 orders and the Warehouse (W). The time-windows are indicated above, and the scheduled arriving time is indicated below the houses. The upcoming orders heuristic would finish this route at 10:00, but the upcoming routes heuristic would finish it at 11:00.

pre-releasing capacity in the future. When deciding which orders to pre-release, one should pick orders that are likely to end up on the route they are currently planned on, even if new orders come in in the future. Eight of such heuristics are suggested below, alongside a brief motivation. Later, an example is given to demonstrate the heuristics.

The question of how many extra orders to pre-release is not in the scope of this thesis. Instead, it is assumed that the pre-releasing capacity is always fully used. This makes sense from a business perspective, as the pickers in the warehouse are already there and expect to be working. However, this is not an ideal strategy, as it would mean that after a few hours almost all orders will be pre-released, especially if the pre-releasing capacity is high. Since all heuristics have to pre-release the same amount of orders, their performance can be compared with each other. As the performance of the heuristics might depend on the (fixed) number of pre-releases, the heuristics will be compared for various pre-releasing capacities.

2.1. Random. This heuristic randomly pre-releases an order from the planned, but un-pre-released orders, until the pre-releasing capacity is reached. This heuristic does not use any additional information, and is therefore expected to not work very well. However, it does act as a useful reference point for later heuristics and the cost of pre-releasing in general.

2.2. Closest. This heuristic pre-releases the orders with the lowest distance to the warehouse. Ideally the driving time to the order would be used for this, but instead this thesis estimates the distance based on the Manhattan distance between the coordinates of the order and the warehouse.

The idea behind this heuristic is that pre-releasing the closest orders has the lowest risk; even if an order that is just North of the warehouse ends up on a route going to the South, at most two times the (small) distance between the order and the warehouse is wasted.

2.3. Furthest. This heuristic pre-releases the orders with the highest distance to the warehouse. The idea behind this heuristic is that orders that are far away are less likely to be changed to another route, whereas closer orders are more flexible to change to other routes, as all routes go through the area around the warehouse.

2.4. Minimum Driving Time. Orders that are close together will likely end up on the same route. For example, if two neighbours both make an order (for the same time-window), it is unlikely that the ideal planning does not combine them on one route. Therefore, a logical heuristic would be to pre-release the orders that are closest together, or rather have the smallest driving time between them. This information is directly contained in the optimized VRP planning. Note that this might result in either pre-releasing 2 orders (when both were not pre-released before), or in only pre-releasing 1 extra order (when 1 has been pre-released previously). If only 1 order needs to be pre-released but the minimum driving time is between 2 un-pre-released orders, a tie-break rule is applied that pre-releases the order furthest from the warehouse (see the example below).

This heuristic seems to be the most intuitive heuristic, as it "greedily" pre-releases the orders that will most likely end up together.

2.5. Maximum Distance. Instead of pre-releasing orders that will likely end up together (on the same route), another strategy is to pre-release orders that will likely not end up together (on different routes). By pre-releasing the order which is furthest from any other pre-released order (or the warehouse), it is unlikely that these orders would end up on the same route in the future, when new orders come in.

For this heuristic, one considers all orders that are planned in the latest planning, except those that are finished already. For each of the non-pre-released orders, the distance to the closest pre-released order (or the warehouse) is computed. The order that is furthest away from any pre-released order (or the warehouse) will be pre-released next. This process is repeated until the pre-releasing capacity is reached.

2.6. Furthest and Closest Seeding. The Furthest and Closest heuristics may appear intuitive when the number of pre-released orders is small, but their intuitiveness may diminish when a significant number of orders have already been pre-released. Also, simulations have suggested that pre-releasing orders to a route that already has many pre-releases is undesirable.

Therefore, the Furthest (or Closest) Seeding heuristic starts by pre-releasing the s orders from every route that are furthest from (or closest to) the warehouse, and then pre-releases additional orders via the Minimum Driving Time heuristic. This method is similar to some VRP construction algorithms, such as the Farthest Insertion Heuristic [17].

These Seedings heuristics are generalizations of the Furthest, Closest and Minimum Driving Time heuristics. When $s = 0$, the heuristics are the same as the Minimum Driving Time heuristic, whereas for large s (larger than the maximum number of orders in a route) the Seedings heuristic is equal to the Furthest (or Closest) heuristic.

2.7. Next Time-Window. Orders that need to be delivered far into the future are more likely to change routes than orders that need to be delivered sooner.

Therefore, the Next Time-Window heuristic pre-releases the orders based on a lexicographical ordering of their time-window start and driving time. First, it considers the orders with the closest time-window start. This means that orders with earlier time-window starts will be pre-released before orders with later time-window starts. Within each time-window, the Minimum Driving Time heuristic is applied to determine the order of pre-releasing based on driving time.

2.8. Example of a Pre-Releasing Problem Instance. To exemplify the 8 "Extra" pre-releasing heuristics suggested in this chapter, the example stated in the Introduction will be revisited in Figure 3.2 to showcase the workings of the heuristics. As explained in the Introduction, the green route will be finished according to the "Last-Minute" heuristic, so three more orders need to be pre-released. Below, the outcome of the heuristics are given alongside a brief explanation.

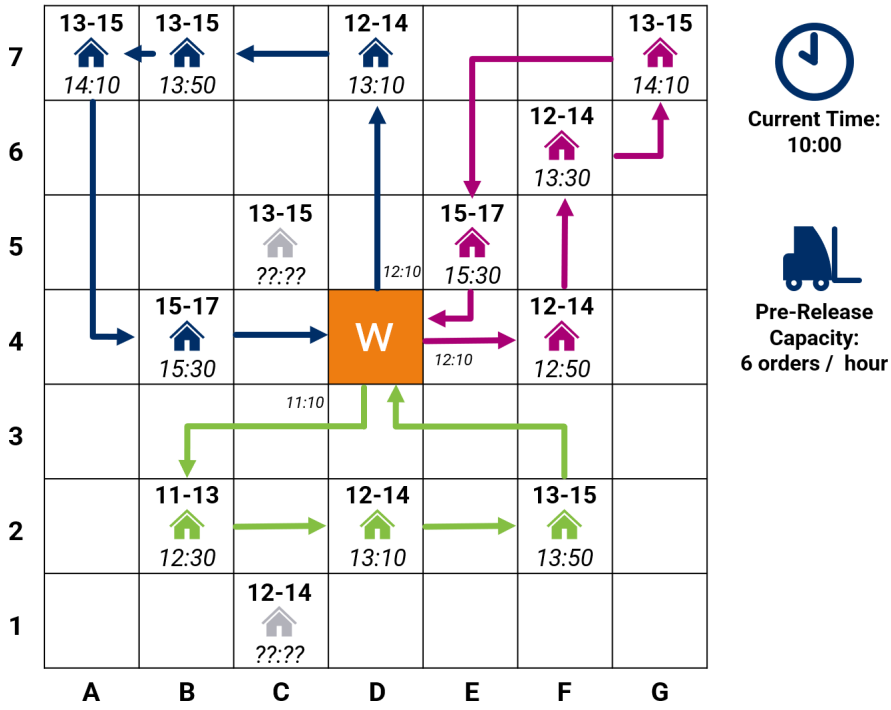

FIGURE 3.2. Example of a pre-releasing problem, with 3 optimized routes and order time-window and estimated arrival shown (see Section 3 for more details).

- **Random: B4, D7, F6**.
- **Closest: B4, E5, F4**. These orders have a Manhattan distance of 3 to the warehouse.
- **Furthest: A7, G7, B7**. Orders A7 and B7 have a Manhattan distance of 6, and B7 a distance of 5.
- **Minimum Driving Time: A7, B7, G7**. The shortest driving time is between A7 and B7. There are several driving times of 40 minutes, so the

tie-break rule decides that the furthest of these orders will be pre-released, so G7.

- **Maximum Distance: A7, G7, D7**. The order furthest from the warehouse is A7 (or G7). The order furthest from either the warehouse or A7 is G7. The order furthest from the warehouse, A7 and G7 is D7.
- **Furthest Seeding (s=1): A7, G7, B7**. The furthest order from each route is pre-released, so A7 and G7. Next, the Minimum Driving Time heuristic is applied. The minimum driving time is between A7 and B7 and A7 is already pre-released, so B7 will be pre-released.
- **Closest Seeding ($s = 1$): B4, F4, A7**. The closest order from each route is pre-released, so B4 and F4 (or E5). Next, the Minimum Driving Time heuristic is applied, so A7 is pre-released.
- **Next Time-Window: D7, F4, F6**. The first time-window in the remaining orders is 12:00 - 14:00, so the three orders with that time-window will be pre-released.

3. Suggestions for more Complex Heuristics

When a planner manually decides which orders to pre-release, they might intuitively use various other criteria in their decision-making process. After discussing the problem with several experts, the following criteria have been identified. These could potentially be incorporated into a heuristic through a penalty function, which would discourage pre-releasing certain orders. In general, pre-releasing should be discouraged for orders with...

- **Later time-windows**. The later the time-window for an order, the more new orders are likely to come in that could change the optimal route for the order. Therefore, pre-releasing such orders might not be advantageous.
- **Routes with few orders**. Routes that contain only a few orders are more susceptible to be disrupted or divided among other routes in the future. Hence, pre-releasing orders from these routes might not be the most efficient strategy.
- **Routes with many pre-releases already**. To maintain flexibility and accommodate future orders, it is desirable to leave some space on all routes, which means not pre-releasing all orders from a route. This ensures that there is room to adjust the route based on future demands.

These additional criteria provide more depth and complexity to the decision-making process, enabling a planner to make more informed and strategic decisions when pre-releasing orders. These recommendations have inspired the "Seeding" and "Next Time-Window" heuristics. Future research might try to implement these criteria in other ways, for example by means of a penalty function for orders with a late time-window.

3.1. Multi-Simulation heuristic. Inspired by the two-stage planning heuristic discussed in the previous chapter, this heuristic randomly generates r fictional sets of future orders. Each set is then optimized separately, resulting in r different VRP plannings. If an order consistently ends up on the same route across all r sets, it can be safely pre-released to that route. The Multi-Simulation heuristic pre-releases the orders that are most frequently assigned to their respective routes

across the r sets. As r increases, this heuristic intuitively approaches the "optimal" pre-releasing decision.

However, since no probability distribution of future orders is provided and randomly generating orders falls outside the scope of this thesis, this method will not be implemented in the PRESTO simulations.

CHAPTER 4

Simulation Setup

In this chapter, the simulation setup used to evaluate the performance of the proposed pre-releasing heuristics will be detailed. The simulation emulates the course of a day, during which new orders continuously arrive and the heuristic decides, at multiple optimization moments, which orders to pre-release and which routes to finish. This process involves significant administration and presents various practical challenges. The goal of this chapter is to clarify the workings of the simulation tool and discuss solutions to the encountered practical problems.

After a discussion of the dataset used for the simulation, the chapter will outline the steps taken in each optimization round (as illustrated in Figure 4.1): a Request with up-to-date problem data is sent to the black-box VRP solver OHD, which responds with a near-optimal VRP solution. Subsequently, the PRE-release Simulation TOol (PRESTO) converts this Response into a new Request by selecting which orders to pre-release and which routes to finish (based on one of the heuristics). Additionally, PRESTO handles necessary administration, such as tracking pre-released orders, updating routes in the new Request, and logging all finished routes for later analysis.

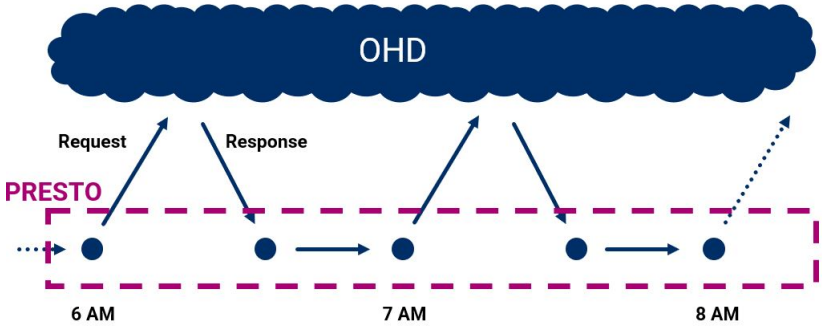

FIGURE 4.1. A visual representation of the simulation tool PRESTO: every optimization round, a Request with up-to-date problem data is sent to black-box VRP solver OHD, which returns a near-optimal VRP solution which is communicated in a Response. Next, PRESTO will convert this Response into a new Request by selecting which orders to pre-release and which routes to finish, as well as updating the set of available orders and routes.

Developing this simulation tool from scratch was by far the most complex step in this research. Although this chapter only outlines the high-level workings of

PRESTO, the intricacies and complexities of implementing it in practice should not be underestimated.

1. Data

The data used in this research originates from a real-life e-grocery chain in Europe that offers Same-Day Delivery, starting from 2 hours after order placement. The dataset encompasses seven consecutive days and includes information on shifts, orders, and planning. This data is stored in an Excel file that can be uploaded to OHD and subsequently converted into a Request.

1.1. Shift-data. The most relevant columns in the shift-data for each shift are summarized below. Note that this data is relevant for the VRP solver, but not necessarily for the pre-releasing problem.

- `start_from`: the earliest moment the vehicle can depart from the depot
- `finish_till`: the latest moment the vehicle must be back at the depot
- `start_depot_id`: the id of the depot from which the shift departs; crucially, in the problem under investigation, every shift departs from the same depot
- `vehicle_type`: the type of delivery vehicle used, e.g. a Peugeot Boxer
- `capacities_unit` and capacities_value: the capacity of the shift for the number of total boxes, ambient boxes, chilled boxes, and frozen boxes
- `deliver_duration_factor` and travel_duration_factor: factors adjusting for delivery speed or travel times, considered during optimization
- `optimization_costs_per shift, per_kilometer, and per_hour`: different shifts have varying costs
- `tags`: labels related to various aspects of the shift, primarily used for indicating the active zone of the truck or for logging purposes

1.2. Order-data. The most relevant columns in the order-data for each order are as follows:

- `amounts_unit` and `amounts_value`: the number of ambient, frozen, and total boxes required for this order
- `location_latitude` and `location_longitude`: the precise coordinates of the order, which are associated with an address by the map service in OHD
- `timeWindow_from` and `timeWindow_till`: the delivery time-window for the order; most time-windows are 2 hours long and start at every whole hour, with a 10-minute buffer before each hour to ensure timely delivery (the customer sees time-windows in whole hours)
- `required_shift_tags`: orders can only be scheduled on routes with all required shift tags. PRESTO also uses order tags to store the time when an order is received

To facilitate a comprehensive understanding of the simulation results, it is essential to grasp the nature of the data itself. While an in-depth data analysis can be found in the Appendix A, some key characteristics are highlighted here.

Each day encompasses approximately 1500 to 2500 orders, with nearly half of them known at the beginning of the day and the rest arriving throughout the day. As illustrated in Figure 4.2, almost half of orders must be delivered within 2 to 4 hours after placement, emphasizing the highly dynamic nature of the problem.

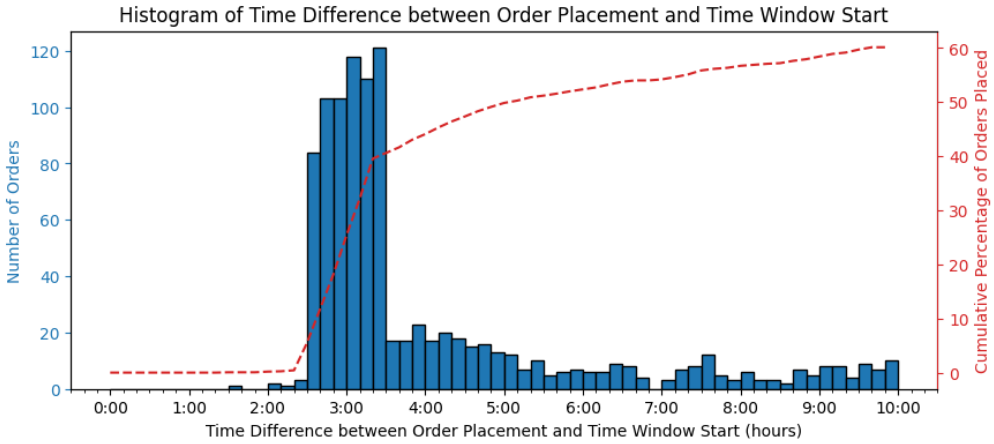

FIGURE 4.2. The figure shows the distribution of time differences (in hours) between the placement of orders and the start of their respective time windows. The histogram summarizes the number of orders within each time difference bin, with bins of 10-minute intervals. The cumulative percentage line indicates the cumulative percentage of orders placed within or before each time difference bin. Note that the 40% of orders that are placed over 10 hours before the start of the time-window are not displayed.

Figure 4.3 portrays the distribution of time-windows, revealing that the start of the day is the busiest period. Consequently, this research will investigate the comparison between commencing pre-releasing earlier in the morning and initiating it during the night before.

2. Request

Every optimization round starts with a Request being sent to OHD. This Request encompasses up-to-date information on routes and orders, as well as miscellaneous data required for optimization (e.g., the optimization script OHD should employ, whether OHD should utilize congestion data or not, and the depot's location). This information is conveyed in a JSON format, which shares the same structure as a Python dictionary. The types of information in the routes and orders resemble those described in the Data section.

The routes within the Request are copied from the previous Response, serving as a starting point for optimization. Notably, during optimization, the non-pre-released orders can be rearranged or even relocated to another route. The Request also encompasses pre-release information, denoting which orders cannot be removed from a route (although the order of orders within a route may be altered).

A proper Request should include orders that are known, but not finished. That is, the time_received (which is a tag of every order) should be before the current time, and the order should not be in any of the routes that is finished already.

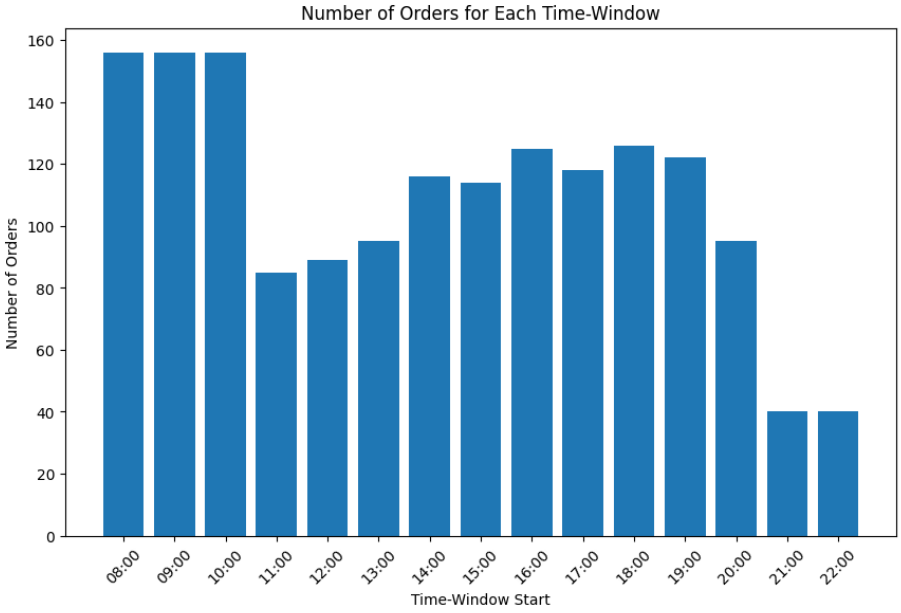

FIGURE 4.3. The number of orders per time-window start-time.

A request from the morning might include orders that only have to be delivered at the end of the day. This means that optimization in the morning takes longer because it also needs to plan these later orders. One attempt to improve simulation speed is by applying a Time-Horizon on the Requests, which only considers orders that need to be delivered in the next h hours, resulting in "shorter" requests with less orders. As will be demonstrated in the Results chapter, using a Time-Horizon larger than 3 had no effect on the optimization costs, but did make the simulation a few minutes faster.

3. OHD

OHD (ORTEC Home-Delivery) is a VRP planner that takes a Request as input, and returns a Response containing a near-optimal planning. To be precise, OHD is an online dashboard (depicted in Figure 4.4) visualizing the data and providing user-friendly ways to manipulate and inspect the data. For example, users can easily manually pre-release orders, finish routes or optimize a set of orders. The optimization itself is performed by CVRS (COMTEC Vehicle Routing Service). CVRS is considered as a black box in this thesis. This thesis uses the same version and settings of CVRS as BosMart used in their baseline.

OHD uses a third-party map service to find the driving time between two addresses, which includes an estimate for the congestion time. This lookup costs time and money, which is why most heuristics suggested in this thesis use a distance metric instead of a driving time metric. Furthermore, the planning in OHD

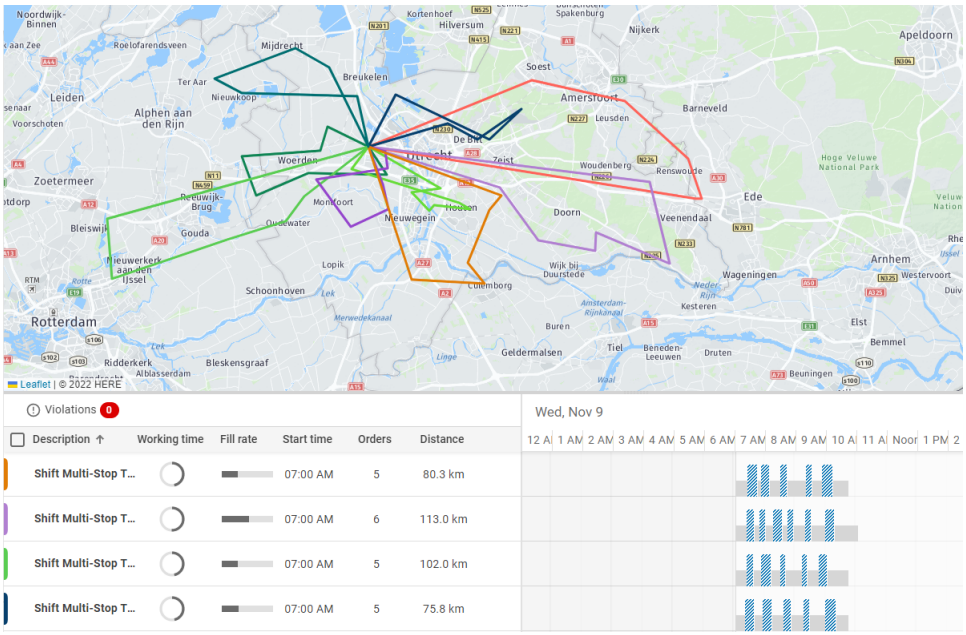

FIGURE 4.4. A screenshot of the OHD user-interface, displaying shifts with planned orders in them.

translates the location (longitude, latitude) into addresses. This makes generating random orders difficult, as a random location might not be associated with a correct address.

One nice feature of CVRS is that it is deterministic, in a sense that the same input will always result in the same output. The methods in CVRS itself are stochastic, but always use the same random seed. Because all pre-releasing heuristics suggested in this research are also deterministic, it follows that all experiments are deterministic, and can thus be reproduced (which has been done for several unexpected results). However, a small change in input, such as rearranging the order of the routes, will result in a different output.

4. Response

After receiving a Request and computing a near-optimal planning, OHD communicates the planning back in the form of a Response. This Response is a JSON file containing Key Performance Indicators (KPIs), routes, and orders that have not been planned. The KPIs provide a quantitative evaluation of the solution generated by OHD.

The KPIs included in the Response are:

- `additionalCosts`
- `breakDurationInSeconds`
- `capacityPenalty`
- `costs`
- `distanceInMeters`

- `drivingDurationInSeconds`
- `maxDistancePerTripPenalty`
- `numberOfPlannedOrders`
- `numberOfUsedRoutes`
- `numberOfUsedTrips`
- `overTimeInSeconds`
- `routeDurationInSeconds`
- `routeFinishTimePenalty`
- `timeWindowPenalty`
- `waitDurationInSeconds`
- `workTimePenalty`

Not all of these KPI's are of equal interest. The primary objective of the optimization process is to plan as many orders as possible, and only after that, minimize optimization costs. The optimization cost is a weighted sum of distance, numberOfUsedTrips, and routeDuration.

5. PRESTO

After the Request has been processed by OHD and a Response is received, the PRE-release Simulation TOol (PRESTO) will create a new Request for the following optimization round. Next to some important administration, such as adding and removing routes and orders, PRESTO will also apply one of the heuristics to determine which orders will be pre-released, and which routes will be finished. After a new Request has been created, PRESTO will send it to OHD, which will answer with a new Response, from which PRESTO builds a new Response, and so on. Throughout the simulation, PRESTO also takes care of the logging of the finished routes, as well as the pre-release history.

In the following subsections, these main functionalities of PRESTO will be explained further, as well as some of the complications that this brings about.

5.1. Input and Output of PRESTO. To use PRESTO, it suffices to create a folder with Request files, containing all information of the orders and routes of that day. In particular, the time that an order is received is stored in the "tag" key of every order.

Next, some settings are required, for example the working directory (which should point to the folder with the Requests) and some authentication keys for communicating with OHD (unfortunately, this part means that non-ORTEC users cannot use PRESTO).

Finally, the parameters that should be tested need to be given, by providing a list of parameter values per parameter. The parameters that can be incorporated in the simulation are:

- `use_capabilities` (boolean): True means that shifts can only contain 2 different time-windows, which is how BosMart currently operates and what is also used in the baseline. False means that a shift can contain any number of different time-windows.
- `Time_Horizon` (integer): A time_horizon h means that only the next $h+1$ time-windows are considered. As an example, if an optimization is started at 6 with a time_horizon of 3, the next relevant time-window is 08:00 - 10:00 (as pre-releasing takes 50 minutes and the 08:00 - 10:00 orders will

be loaded at 06:50), so the Request will contain time-windows up to 11:00 - 13:00.

- `PR_Horizon` (integer): Similar to the Time_Horizon, but a PR_Horizon h means that only the orders from the next $h + 1$ time-windows can be pre-released. As an order can only be pre-released if it is in the Request, we have PR_Horizon $\leq$ Time_Horizon.
- `Optimization_Start_Times` (list of floats): This indicates at what times optimization should start. Note that it might be interesting to change the optimization period (ever hour, every half hour, etc.), as well as when the first optimization starts (this research compares starting at 21 vs 4 vs 6).
- `PR_time_dt` (datetime object): The time it takes to pre-release an order. In practice, this takes 50 minutes, so orders on a route departing at 10:10 (which means loading starts at 09:50) can be pre-released up to 09:00. However, it might be interesting to increase the pre-releasing time, for example to smoothen the pre-release rate.
- `Upcoming_Routes_Bool` (Boolean): True means that the upcoming route heuristic will be used as the last-minute heuristic, False means that the upcoming orders heuristic will be used.
- `pre-releasing_Capacity` (integer): A hard constraint on how many orders can be pre-released per hour.
- `heuristic_name` (string): The name of one of the 10 heuristics PRESTO can simulate. For the seeding heuristics, the parameter s is required as well.

5.2. PRESTO task 1: Creating a new Request based on the previous Response. Creating a new Request is a complicated process, involving the following steps:

(1) Initialize the process by using the previous Request as a starting point. Import the routes from the previous Response into the Request.
(2) Include any new orders that have been placed since the previous optimization up until the current time. Add these new orders to the Request to incorporate them into the planning process.
(3) Apply a heuristic to determine which orders to pre-release and which routes to finish. At the very least, a last-minute heuristic should be implemented to finish all routes that are scheduled to depart between the current time and the next optimization period.
(4) Remove the finished routes and their corresponding orders from the Request. This ensures that only the remaining active routes are considered for further optimization.
(5) Adjust the earliest_start_times for the routes that are scheduled to depart between the current time and the next optimization time, accounting for the time required to pre-release orders. By incrementing the earliest_start_times by 1, routes can be planned with sufficient time for pre-releasing their orders before departure. Omitting this step not only caused problems in the simulations for this thesis, but also in BosMart's actual planning.
(6) Eliminate any empty routes from the Request where the updated earliest_start_time exceeds the latest_finish_time, as those cannot be used anymore.

(7) Add the pre-release information to the pre-released orders. Each order that is pre-released will have the "fixedInRoute" key set to True. Note that pre-release information is not retained in the Response, so all orders need to be pre-released again in each optimization round. To track the pre-release decisions, the information is logged in a separate PR_Dict file.

By following these steps, a new Request is created, incorporating any updates, pre-releases, and route removals necessary for the subsequent optimization round.

5.3. PRESTO task 2: Run the simulation throughout the day. PRESTO takes care of the simulation from start to finish. At the start, communication with OHD is established, which may require obtaining an authentication key from the ORTEC servers. After that, a file "Request_with_time_received.JSON" is created from the given Request, which cleans the Request by adjusting some formatting, and also takes out invalid routes and orders (for example, there are some fake orders which need to be delivered at the depot).

Every optimization round starts with a Request, which is first transformed to a "Shorter_Request", taking out all orders that fall outside of the given Time_Horizon. This shorter request is sent to OHD via an API call, and PRESTO periodically checks for a Response. This Response is then saved, after which PRESTO decides which orders to pre-release and which routes to finish. These decisions lead to 2 new files: the PR_Dict saves which orders have been pre-released to which routes, and the file SavedRoutes contains a list of the routes that have been finished thus far. After converting the Response to a new Request (as is explained in the previous subsection), PRESTO will save this new Request before sending it to OHD.

At the end of the day, PRESTO's final action is to compute the KPI's of the finished routes from that day.

Next to obtaining the KPI's PRESTO also performs several checks to validate the feasibility of the results. Some of the checks include:

- Check if all orders are pre-released before they are finished.
- Check if orders that are supposed to be pre-released are actually pre-released in the following Requests.
- Check if no orders are pre-released more than once.
- Check if the number of planned orders is equal to the number of pre-releases.
- Check if the pre-releasing capacity is never exceeded

CHAPTER 5

Results and Analysis

"The only real mistake is the one from which we learn nothing." ~ Henry Ford

In this chapter, the main results from the simulations with PRESTO are presented. Given the wide variety of problem parameters and heuristics, it is helpful to outline the structure of this chapter:

- Section 1 contains an analysis of the baseline performance of BosMart. Additionally, the method "OHD-IH" (OHD In-Hindsight) is introduced and analyzed, serving as a offline benchmark for comparing the performance of the heuristics.
- Section 2 presents the results of all problems in which the pre-releasing capacity does not play a role. This means that only applying the "Last-Minute" heuristic suffices. After comparing the performance of the last-minute heuristics, the effects of several parameters (2 or more time-windows, time_before_PR, time_horizon) are studied.
- Section 3 summarizes the simulations where there is a fixed pre-releasing capacity that cannot be exceeded. For this, the 8 suggested "Extra" pre-releasing heuristics will be compared for various pre-releasing capacities and PR_Horizons.

Throughout the chapter, some parameters will be fixed after analysis indicates that there is a clearly best-performing value. The most important parameters, as well as their default value, are given below in Table 5.1. Except stated otherwise, these parameters are used in the simulations in this chapter.

As was discussed in the previous chapter, all simulations in PRESTO are deterministic, in the sense that repeating an experiment with the same input will give the exact same results. Only the simulation time might vary due to variable cloud computing performance.

1. Baselines: BosMart versus OHD In-Hindsight

A useful first step in analyzing this (dynamical) problem is to compare the current performance with the "optimal" performance to get an idea of the potential savings. In most interesting practical optimization problems, the optimal solution or objective value is not known (otherwise the problem would be solved already), and therefore the optimality gap between a solution and the best solution is not known. However, dynamical problems have the nice feature that there exists an offline version of the problem, in which all problem data is known beforehand. For this pre-releasing problem, the offline version is a VRP with time-windows where all orders are known in advance. This can also be seen as solving the problem

TABLE 5.1. Optimal parameter values

Parameters	**Optimal Value**	**Explanation**
Time-Horizon	6	A time_horizon h means that only the next h time-windows are considered. There is no significant difference in optimization costs for Time_Horizons between 3 and 14.
Use-capabilities	FALSE	Using capabilities means that trucks can only have 2 different time-windows, which is what BosMart used in their baseline operations. Not using capabilities leads to better performance.
Optimization start times	[6,7,...,19,20]	This list contains the time at which a Request will be sent to OHD. Starting at 6 is optimal (if the pre-releasing capacity allows), else starting in the morning is preferred over starting in the evening.
PR_time	50 minutes	The time it takes to pre-release an order. In particular, the last-minute heuristic will trigger if the current time - PR time $<$ next optimization time.
Upcoming routes vs upcoming orders	Upcoming routes	Parameter for setting the used last-minute heuristic. Upcoming routes performs strictly better than upcoming orders.

"In-Hindsight", to find what the optimal planning and pre-releasing strategy would have been.

This planning serves as a offline benchmark for the costs, as it solves the dynamical problem with perfect information. In practice, this offline solution is found by sending a Request with all problem data to OHD, which computes a near-perfect planning. However, this OHD-IH solution is not always a feasible solution, as is illustrated by the following example. Suppose a planning contains a route with different time-windows (e.g. 12-14 up to 16-18), so the orders in this route must be pre-released at 10:00 the latest. However, it may be the case that some of the orders in that truck are not known at the moment the route is planned to depart from the depot; customers can place an order for 16-18 until 14:00 (in fact, customers frequently place such last-minute orders, as can be seen in Appendix A). Of course, orders that have not yet been placed cannot be pre-released, so the OHD-IH method might produce solutions that are not possible in practice. However, the effect of this infeasibility is small (as can be seen in Table 5.2), so the OHD-IH benchmark is close to the optimal solution of the dynamical problem.

In Table 5.2, the optimization costs per day are presented for BosMarts baseline performance, the best-performing last-minute heuristic (upcoming routes) and OHD-IH. For these simulations, the parameters are set to their default values, except "use_capabilities" which is shown for both True (2TW) and False (>2TW).

One important remark is that the simulations with 2 time-windows failed to plan up to 1% of orders compared to BosMart (who always plans everything). This effect is caused by the fact that PRESTO never plans with violations, whereas BosMart uses up to 100 violations per day.

TABLE 5.2. Optimization costs for BosMarts baseline, the best-performing last-minute heuristic in PRESTO and OHD-IH. Results are shown for use_capabilities=True (2TW) and False (>2TW). Percentages are with respect to BosMart's baseline. The bottom row gives the average optimization costs over the 7 days.
* Simulations with 2TW fail to plan up to 1% of orders. Interestingly, PRESTO sometimes manages to plan more orders than OHD-IH.

Day	**BosMart (2TW)**	**PRESTO (2TW*)**	**OHD-IH (2TW*)**	**PRESTO (>2TW)**	**OHD-IH (>2TW)**
1	74363 (100%)	59067 (-21%)	58016 (-22%)	50103 (-33%)	49114 (-34%)
2	79985 (100%)	65958 (-18%)	65700 (-18%)	58347 (-27%)	56432 (-29%)
3	76653 (100%)	62213 (-19%)	61236 (-20%)	54318 (-29%)	52633 (-31%)
4	63307 (100%)	51338 (-19%)	50796 (-20%)	42721 (-33%)	41864 (-34%)
5	65653 (100%)	54209 (-17%)	53948 (-18%)	47462 (-28%)	46129 (-30%)
6	69321 (100%)	52072 (-25%)	51081 (-26%)	43572 (-37%)	43922 (-37%)
7	76988 (100%)	59646 (-23%)	58516 (-24%)	51096 (-34%)	49621 (-36%)
Avg	72324 (100%)	57786 (-20%)	57042 (-21%)	49660 (-32%)	48531 (-33%)

The performance of BosMart's operations can be significantly improved by implementing a more effective pre-release strategy. Even when adhering to the 2 time-window policy, the use of a better pre-release strategy leads to an average cost reduction of 20%, which is an enormous margin in the e-grocery sector. Allowing more than two time-windows results in cost savings of 32%.

Furthermore, the best-performing pre-releasing heuristic performs very well and is often within 1% of the OHD offline benchmark. Interestingly, the heuristic even outperforms OHD in Day 6. Also, for the case of 2 time-windows per truck, the heuristic sometimes manages to plan more orders than OHD-In-Hindsight (this is not shown in the Table). These findings highlight that OHD-In-Hindsight does not consistently provide the optimal solution. One potential explanation for the superior performance of the heuristics could be attributed to the repeated optimizations, which allows for more refined and localized search.

1.1. Comparing BosMart versus PRESTO with other KPIs. The optimization cost is the KPI showed in most tables in this thesis, but from a business

point of view it's interesting to look at other KPIs as well. The average KPIs are displayed in Table 5.3 below, comparing BosMart's baseline operations with the best-performing heuristic in PRESTO with 2TW or >2TW.

TABLE 5.3. Average of KPIs over the 7 days for BosMart (2TW) and the best-performing heuristic in PRESTO (2TW and >2TW).

	BosMart (2TW)	**PRESTO (2TW)**	**PRESTO (>2TW)**
Cost per order (fuel + driver's payroll)	Secret	-17%	-30%
#Planned orders	1612 (100%)	1597 (-1%)	1612 (100%)
Optimization costs per order	44.9 (100%)	36.2 (-20%)	30.8 (-32%)
Distance per order (km)	6.43 (100%)	5.65 (-12%)	4.91 (-24%)
Route duration per order (min.)	27.4 (100%)	23.3 (-15%)	21.8 (-21%)
#Routes	260 (100%)	211 (-19%)	178 (-32%)
Average #orders per route	6.2 (100%)	7.6 (+23%)	9.1 (+47%)

It is clear that the suggested heuristic performs better in all areas, even when only using 2 time-windows like BosMart. The largest savings are achieved for the number of routes used. This could bring down the costs per order even more, as less cars are required for BosMart's fleet. The current cost per order estimation disregards fixed costs like car rent.

The fact that PRESTO plans routes with more orders is also visualized in Figure 5.1, where the number of orders per route for BosMart and PRESTO are compared.

The results shown so far do not tell the whole story; the pre-releasing rate is a crucial part of the equation. Maintaining a constant pre-releasing rate, or at least not exceeding the pre-releasing capacity, is the main reason BosMart is pre-releasing suboptimally. The fear is that pre-releasing at the last minute might exceed the pre-releasing capacity, which would cause some orders to be delivered too late. For an e-grocery, that would be unacceptable. However, as will be shown in the next section, pre-releasing at the last-minute does not seem to cause any problems with the pre-releasing capacity. Also, smarter strategies of equalizing the pre-releasing rate will be suggested, such as pre-releasing a few hours early, or starting pre-releasing in the morning instead of in the evening.

2. Infinite Pre-Releasing Capacity (using only "Last-Minute" heuristic)

2.1. Comparing Last-Minute Heuristics. If the pre-releasing capacity is of no concern, there is no reason to pre-release earlier than is necessary. In general, pre-releasing orders is undesirable, as it adds an extra constraint to the problem, namely that that order needs to go on that route. Therefore, using a last-minute heuristic is a logical strategy. The performance of the 2 last-minute heuristics

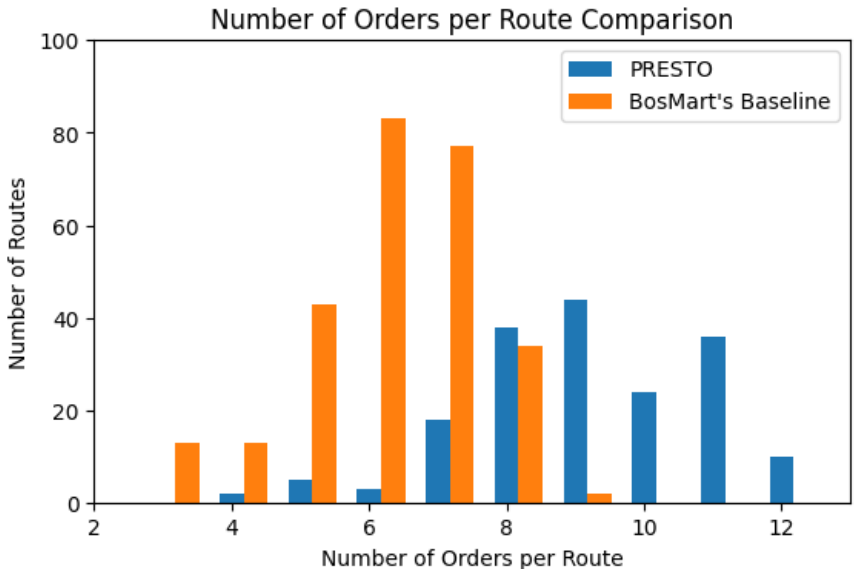

FIGURE 5.1. A bar chart showing how many routes have how many different time-windows. These routes are taken from the best-performing heuristic for day 1, with use_capabilities=False.

discussed in Section 1 are compared in Table 5.4 below. The default parameters are used, except "use_capabilities" which is shown for both True (2TW) and False (>2TW).

The results clearly show that the upcoming routes heuristic performs better in all circumstances. This was to be expected, as the upcoming orders heuristic might finish a route before the upcoming routes heuristic would, whereas the converse is never true. For example, a route containing time-windows 10:00-12:00 and 11:00-13:00 would always be finished at 8:00 according to the upcoming order heuristic, whereas the upcoming routes heuristic might only finish the route at 9:00, which occurs if all 10:00-12:00 orders will be delivered after 11:00.

During these simulations, solutions were found that deliver more orders than the OHD-IH solution did. This is a very curious result, as the OHD-IH solution is able to use all orders in its optimization, even those that have not yet been placed. This goes to show that the OHD-IH solution is indeed not perfect. One explanation for this phenomenon is that PRESTO repeatedly optimizes its routes, and starts its search with the routes from the previous Response. Therefore, PRESTO is able to do more local search than OHD-IH can.

One might propose that the upcoming routes heuristic is the optimal solution for the case with infinite pre-releasing capacity. This is not true however. There might be cases where you "regret" finishing an order, especially when the order is on a shift with 5 time-windows, for example with time-windows from 10:00-12:00 up to 14:00-16:00 (which does occur). This means that the orders with time-window 14:00-16:00 will be pre-released at 8:00 already, while it might very well be that

TABLE 5.4. Optimization costs for BosMarts baseline, the best-performing last-minute heuristic in PRESTO and OHD-IH. Results are shown for use_capabilities=True (2TW) and False (>2TW). Percentages indicate how much the upcoming route is cheaper than the upcoming orders heuristic. The average of these percentage is given in the final column.
* Simulations with 2TW fail to plan up to 1% of orders, upcoming routes plans 2-4 orders more than upcoming orders.

Day	**Upcoming orders (2TW*)**	**Upcoming routes (2TW*)**	**Upcoming orders (>2TW)**	**Upcoming routes (>2TW)**
1	60382 (100%)	59067 (-2%)	52488 (100%)	50103 (-5%)
2	67918 (100%)	65958 (-3%)	59763 (100%)	58347 (-2%)
3	63851 (100%)	62213 (-3%)	55801 (100%)	54318 (-3%)
4	52641 (100%)	51338 (-2%)	45475 (100%)	42721 (-6%)
5	55485 (100%)	54209 (-2%)	49085 (100%)	47462 (-3%)
6	52904 (100%)	52072 (-2%)	45884 (100%)	43572 (-5%)
7	60337 (100%)	59646 (-1%)	52270 (100%)	51096 (-2%)
Avg %	100%	-2%	100%	-4%

between 08:00 and 12:00 new orders come in that would have fitted nicely with this order.

From now on, all simulations will assume that more than 2 time-windows per route are allowed (i.e., use_capabilities=False). This way, more possible VRP solutions are available, and the difference between heuristics will become more pronounced. The downside of allowing more than 2 time-windows is that comparisons with BosMart become unfair, as they used 2 time-windows per truck. However, at this point it is evident that BosMart's performance is inferior to all proposed heuristics. Also, BosMart plans to allow more than 2 time-windows themselves in the near future.

2.2. Pre-releasing Rate in the Last-Minute Heuristic. The savings presented in the previous section are based on the assumption that the pre-releasing capacity is sufficiently high throughout the day. A pre-releasing rate higher than the pre-releasing capacity would mean that the planning cannot be executed (in time), which would result in undelivered orders. This constraint of pre-releasing capacity is why BosMart is currently pre-releasing so far in advance. Also, it is part of the reason why BosMart is limiting the number of different time-windows per truck to 2.

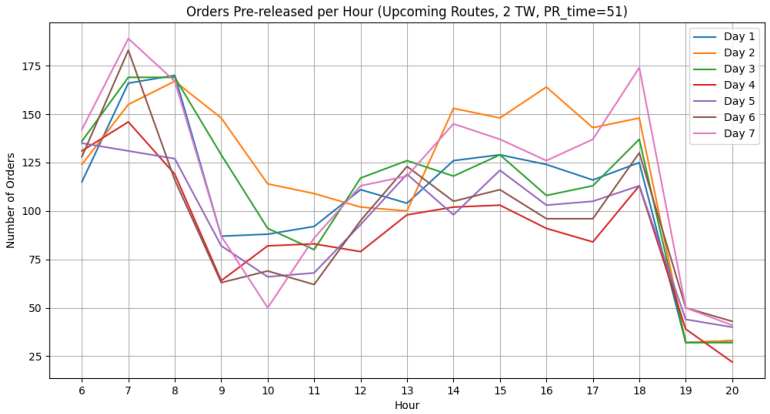

FIGURE 5.2. The pre-release rate per hour for the 7 days, with last-minute heuristic Upcoming Routes, Use_capabilities=False (so 2 TW) and PR_time = 50

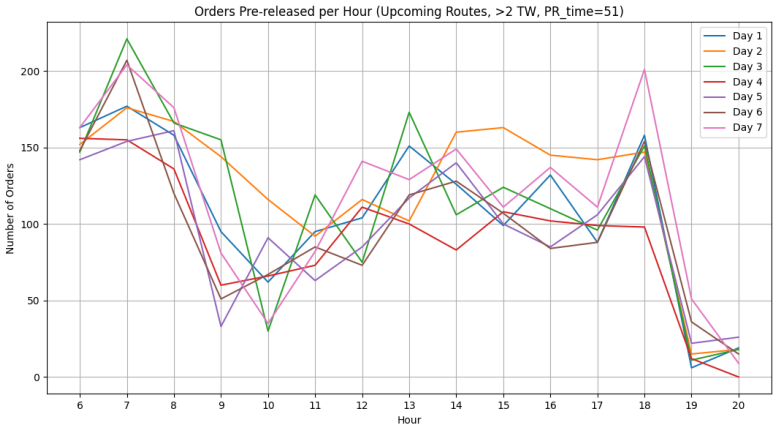

FIGURE 5.3. The pre-release rate per hour for the 7 days, with last-minute heuristic Upcoming Routes, Use_capabilities=False (so >2 TW) and PR_time = 50

The expectation is that pre-releasing all orders at the last-minute would result in "peaks" in the pre-releasing rate, and therefore it would be necessary to pre-release in advance. The pre-releasing rate when using the last-minute heuristic is shown, first with 2 time-windows per route in Figure 5.2, and also for >2 time-windows per route in Figure 5.3.

The maximum pre-releasing rate is much less than expected; only a few peaks are present that exceed 200 pre-released orders per hour. To put this in perspective;

BosMart currently sometimes pre-releases more than 200 orders per hour, and the average pre-releasing rate to deliver 2500 orders in 15 hours would be 167. If the number of time-windows per truck is limited to 2, the peaks in pre-releasing rate are even lower.

This unexpected lack of peaks in pre-releasing rate can be explained by the way time-windows are offered to customers. For this, a separate optimization problem is solved, which uses a cheapest insertion heuristic to find which time-windows would be the cheapest for the order. This method often results in an insertion into a less busy route (and time-window), as those routes have more flexibility. Therefore, this time-window heuristic places more orders on the less busy routes, resulting in a more evenly distributed pre-releasing rate.

An important conclusion can be drawn at this point: BosMart could have pre-released all orders at the last-minute (with the upcoming route heuristic) with their current picking capacity of 200, and would have saved over 21%. Also, the restriction of 2 time-windows per route was not necessary.

Although pre-releasing at the last-minute is desirable from an optimization point of view, it is a risky strategy in real-life operations. Practical problems at the warehouse or unexpectedly high demand might result in orders being pre-released and delivered too late. There are 3 ways to reduce the probability that the pre-releasing capacity is exceeded:

(1) **Increasing the pre-releasing capacity**. This way, the last-minute pre-releasing strategy can always be used without exceeding the pre-releasing capacity. This would require an investment, for example in hiring extra workforce in the warehouse, where one extra picker would increase the pre-release capacity by 10. A smarter way would be to only increase pre-releasing capacity in the morning and evening, as that is when peaks are most likely to occur. However, BosMart prefers to have a constant pre-releasing rate, to prevent workers from doing nothing. This is not an ideal strategy from an optimization point of view, as it will result in pre-releasing more than is necessary.
(2) **Applying the last-minute heuristic a few hours in advance**. Currently, the last-minute heuristic pre-releases orders and finishes routes if the start time of that route minus the PR_time is before the next optimization time. A buffer can be created here, if the PR_time is increased, resulting in finishing routes a few hours before one has to. This method is analyzed in Subsection (2.3).
(3) **Pre-releasing extra orders from further in the future**. These heuristics will be discussed in Section 3.

2.3. Finishing Routes more than 1 hour before departure. Pre-releasing not at the very last minute, but a fixed time before the last minute creates an extra buffer to pre-release orders from earlier "peaks". Suppose for example that there are 200 orders on routes that should be loaded at 07:50. According to the upcoming routes heuristic, these orders should be pre-released at 07:00. Instead, a safer strategy would be to pre-release these orders before 07:00, for example at 06:00. This way, any orders that are failed to be pre-released at 06:00 (due to limited pre-releasing capacity or problems at the warehouse), could still be pre-released at 07:00, as the act of pre-releasing one order still takes only a few minutes.

The effect of this strategy is presented in Table 5.5 for various PR_times (remember that the default is PR_time=50 minutes).

TABLE 5.5. Optimization costs for various values of PR_time. Results are shown for use_capabilities=False (>2TW),Time_Horizon=100 and using the upcoming routes heuristic. Percentages are taken with respect to the costs of the PR_time=50 simulation from that day.

Day	**PR_time =50**	**PR_time =110**	**PR_time =170**	**PR_time =230**	**PR_time =290**
1	50103 (100%)	50684 (101%)	51254 (102%)	54543 (109%)	64252 (128%)
2	58347 (100%)	58342 (99.99%)	59688 (102%)	62966 (108%)	71098 (122%)
3	54318 (100%)	54065 (99.5%)	55804 (103%)	61557 (113%)	70881 (130%)
4	42721 (100%)	44366 (104%)	47712 (112%)	51704 (121%)	59417 (139%)
5	47462 (100%)	47520 (100%)	49147 (104%)	53275 (112%)	61037 (129%)
6	43572 (100%)	44307 (102%)	46338 (106%)	49392 (113%)	58794 (135%)
7	51096 (100%)	51136 (100%)	51406 (101%)	54461 (107%)	64129 (126%)
Avg %	100 %	101%	104%	112%	130%

As expected, pre-releasing earlier generally results in higher costs. Surprisingly, this effect is relatively small when pre-releasing 1 or 2 hours before the last minute. Therefore, this strategy might be interesting for practical use, as it provides extra safety margins while keeping the optimization costs low. The dramatic increase in optimization costs when pre-releasing 3 or 4 hours in advance might be explained by the fact that approximately 40% of all orders have less than 4 hours between order placement and the start of the time-window (see Figure A.2 in the Data Analysis Appendix). If pre-releasing occurs 3 or 4 hours early, these orders are not included in the optimization, resulting in bad pre-releasing decisions.

Another unexpected result occurs in Day 2 and 3, when pre-releasing 110 minutes before loading results in lower costs than pre-releasing 50 minutes before loading. At first, this was thought to be a mistake in simulation; how can an optimization with less information result in a better solution? However, repeating the experiment gives the same results.

When pre-releasing orders (and finishing routes) early, it becomes possible that new orders with the same time-window come in thereafter. For example, orders with time-window 10:00-12:00 can be placed until 08:00. So if an order is planned on a route that should be loaded at 08:50, it should be pre-released at 08:00 the latest. At that point, no new orders for the next time-window (10:00-12:00) can come in, but later time-windows (which might also be in the route) could still come in. In fact, roughly half of the orders come in only 4 hours before the time-window

starts, as can be seen in Appendix A. These new orders cannot be planned with the previously pre-released orders, as the last-minute heuristic has already finished these routes. Therefore, these new orders must be planned together or with later time-windows. This restricts the solution space, which is generally undesirable but could in theory focus the local search of CVRS into a better solution. Also, it might be that the little information you have when pre-releasing early actually points the optimization towards the "optimal" planning, and the extra information available when pre-releasing later is misleading.

There is one major caveat to the results in Table 5.5: the pre-releasing rate at 06:00 increases tremendously. For Day 1 for example, the pre-releasing rate for PR_time= 50 is 163, for PR_time= 110 is 321, PR_time= 170 is 163, for PR_time= 230 is 598 and for PR_time= 290 is 607.

In general, pre-releasing 2 hours earlier will shift the entire pre-releasing rate 2 hours to the left, except for the orders that come in after pre-releasing. However, as 06:00 is the earliest optimization start time, all orders on routes that should be loaded at 06:50, 07:50 and 08:50 need to be pre-released. For infinite pre-releasing capacity this is no problem, but for practical applications it will become necessary to start pre-releasing earlier. In the next subsection, we compare starting pre-releasing earlier in the morning against starting the night before.

2.4. Starting Optimization in the Morning or in the Night before. Currently, BosMart uses the time from 21:00-23:00 to pre-release orders for the next morning, to prevent the workers from doing nothing. It should be clear by now that this strategy is far from optimal, as pre-releasing too far in advance results in bad performance. Even between 22:00 and 06:00, over 100 new orders are placed, most of which need to be delivered in the first time-windows of the next morning (see Appendix 1.1 for a detailed analysis).

Pre-releasing earlier in the morning is therefore expected to have better performance than pre-releasing the night before. Table 5.6 below compares starting optimizing at 21:00 and 22:00 versus 04:00 and 05:00, for the case where PR_time=170, so 2 hours before the last-minute.

Starting optimization in the morning performs better than starting in the evening before, except for days 2 and 3. As before, this is likely due to randomness of the VRP solver, or perhaps the 100 orders that come in overnight (and thus were not considered in the pre-releasing decision in the evening) were very favorable in day 2 and 3, for example if they would fit nicely together on a route, which might not have been found if they were considered with all other orders.

The maximum pre-releasing rate required does not seem to depend on whether the optimizations start in the morning or evening, but it is informative to compare the pre-releasing rates for the various starting times, as is shown in Figure 5.4.

In Figure 5.4, the peak at 06:00 for PR_time=50 is expected, but undesirable. Starting in the morning or the evening distributes the peak more evenly. The lines for starting in the evening or morning with PR_time=170 have the same maximum pre-releasing rate as those starting at 06:00 with PR_time=50, but are shifted to the left by 2 hours. As a result, the pre-releasing in those cases also ends 2 hours earlier.

2.5. Time-Horizon. Simulating a day in PRESTO can take up to 45 minutes, most of which is spent waiting for a Response from OHD. For practical use, this

TABLE 5.6. Optimization costs and maximum pre-releasing rate for simulations starting at 21:00 or 4:00 with PR_time = 170, compared to starting at 06:00 with PR_time = 50. Results are shown for use_capabilities=False (>2TW), Time_Horizon=100 and using the upcoming routes heuristic.

	PR at 6,7,8 …		PR at 4,5,6 …		PR at 21,22,6 …	
Day	**costs**	**max PR_rate**	**costs**	**max PR_rate**	**costs**	**max PR_rate**
1	50103 (100%)	177	51672 (103%)	170	52095 (104%)	176
2	58347 (100%)	176	60139 (103%)	205	59717 (102%)	180
3	54318 (100%)	221	57064 (105%)	198	56664 (104%)	184
4	42721 (100%)	156	47504 (111%)	161	49121 (115%)	172
5	47462 (100%)	161	49904 (105%)	162	50257 (106%)	163
6	43572 (100%)	207	46669 (107%)	182	47494 (109%)	205
7	51096 (100%)	204	52631 (103%)	228	54255 (106%)	207
Avg	100%	186	105%	187	107%	184

is no problem, as one optimization round only takes a few minutes. However, attempts have been made to decrease the simulation time, mainly by implementing a Time_Horizon.

The main reason that OHD takes so long to optimize, is because of the number of orders in every Request. In the first optimization of the day, all orders (that are known at that point) need to be planned, even those that should only be delivered at the end of the day. One attempt to decrease simulation time is by only considering the next time-windows (up to a Time_Horizon), and leaving all later orders out of the Request. In PRESTO, this can be done by using the parameter Time_Horizon, which includes the h time-windows after the upcoming time-window in the Request. For example, an optimization starting at 6 always includes the time-window 8-10, but a Time_Horizon of 3 would also include the time-windows 9-11, 10-12 and 11-13. A Time_Horizon of 0 only includes the next time-window, while a Time_Horizon of 100 includes also the 100 time-windows after that (so all orders in the day). The optimization costs and simulation time for various Time_Horizons are presented below in Table 5.7.

Generally, a larger Time_Horizon tends to yield lower costs, although it does increase the simulation time. However, this isn't always the case. For instance, on days 1, 2, 3, 5, and 7, a Time_Horizon of 100 doesn't necessarily result in the lowest optimization cost. This suggests that having more information doesn't always lead to an improved solution.

One possible explanation for this could be that the VRP solver (OHD) becomes "distracted" by later time-windows, which are less pertinent to the pre-releasing

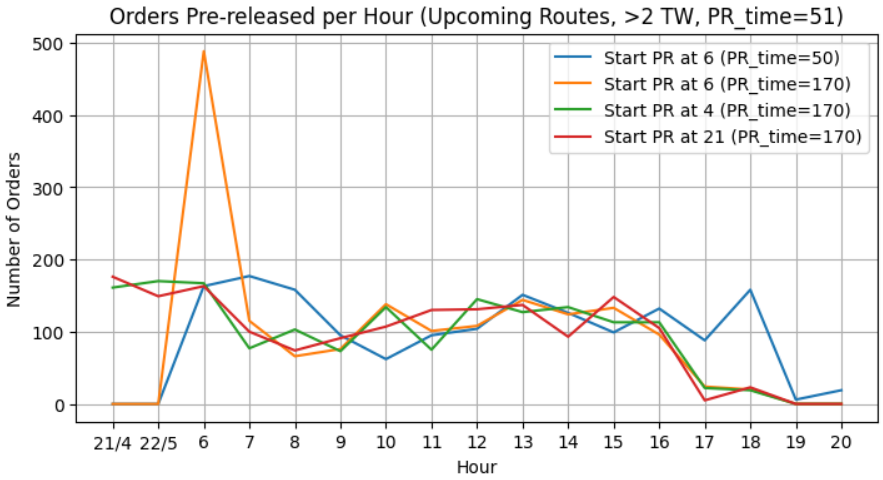

FIGURE 5.4. The pre-release rate per hour for Day 1, comparing starting pre-releasing at 04:00 or 06:00 in the morning, or at 21:00 the night before. The PR_time is 170, except for the blue line.

TABLE 5.7. Optimization costs for various values of PR_time. Results are shown for use_capabilities=False (>2TW) and using the upcoming routes heuristic.

Day	**TH=0**	**TH=1**	**TH=2**	**TH=3**	**TH=5**	**TH=8**	**TH=100**
1	61692 (370s)	52843 (508s)	50547 (996s)	50194 (1206s)	49700 (1601s)	50176 (1852s)	50103 (1841s)
2	68642 (443s)	60040 (602s)	58043 (1163s)	58695 (1507s)	58655 (1782s)	58148 (2157s)	58347 (2286s)
3	66082 (372s)	57565 (621s)	54250 (907s)	53881 (1373s)	53136 (1464s)	53009 (1689s)	54318 (1765s)
4	54242 (319s)	47046 (467s)	43435 (708s)	43104 (871s)	43909 (982s)	43737 (1098s)	42721 (1131s)
5	57829 (290s)	49787 (481s)	46888 (773s)	46555 (981s)	45854 (1153s)	46525 (1288s)	47462 (1370s)
6	55384 (379s)	47508 (488s)	43949 (810s)	43966 (1028s)	44050 (1175s)	43788 (1359s)	43572 (1417s)
7	62493 (433s)	54530 (630s)	51330 (996s)	51439 (1462s)	51409 (1783s))	50857 (2000s)	51096 (2202s)
Avg	60909 (372s)	52760 (542s)	49777 (908s)	49690 (1204s)	49530 (1420s)	49463 (1635s)	49660 (1716s)

problem being addressed. In the majority of simulations, the number of distinct time-windows per truck rarely exceeds five, which is also consistent with the In-Hindsight solutions. As a result, one might reasonably expect at least the next five windows to be relevant. While time-windows beyond that might indirectly

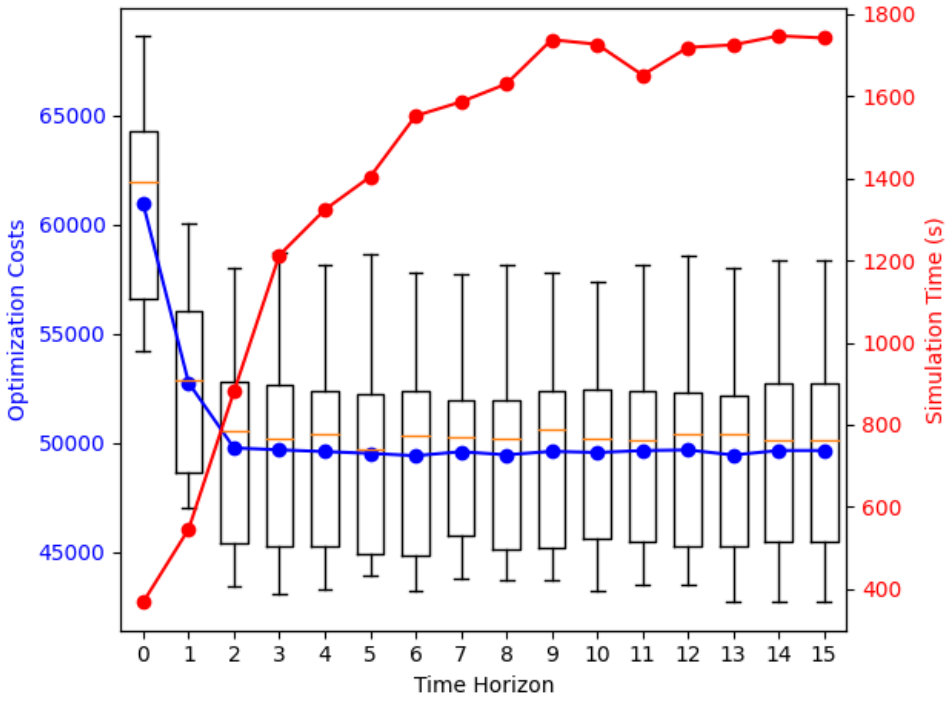

FIGURE 5.5. For each Time Horizon, a boxplot is made of the optimization costs of the 7 days. The lines represent averages for the optimization costs (blue, left axis) and simulation time (red, right axis).

offer useful information, the orders associated with these windows cannot be pre-released.

Considering the significant impact of the Time_Horizon parameter, simulations were conducted for all 7 days using all 16 possible Time_Horizons (ranging from 0 to 15). Figure 5.5 below displays boxplots of the optimization costs from these 112 simulations, along with the average simulation time.

From the figure, it becomes clear that there is no significant difference in optimization costs for Time_Horizons between 3 and 15; their averages are all between 49450 and 49700. The lowest average (49421) occurs at Time_Horizon=6, but this should be taken with a grain of salt given the small sample size.

The extra simulation time diminishes, which is logical as a larger time-window only adds more orders in the evening; at 15:00, there is no difference between Time_Horizon=5 and Time_Horizon = 100. In fact, simulations for Time_Horizon = 14 and Time_Horizon=15 gave the exact same results, which was to be expected as the Time_Horizon = 14 already includes all orders from the 15 different time-windows (08:00 - 10:00 up to 22:00 - 00:00) in every Request.

3. Finite Pre-releasing Capacity (using "Extra" heuristics)

In this section, pre-releasing strategies are compared that always pre-release orders up to the pre-releasing capacity. This means that next to the "Last-Minute" heuristic, one of the "Extra" heuristics described in Chapter 3 is required.

First, Table 5.8 shows the optimization costs of 5 heuristics for 4 different pre-releasing capacities and 2 different PR_Horizons, averaged over the 7 days. To put these numbers in perspective; only applying the Last-Minute heuristic resulted in an average cost of 49660, which required a pre-releasing capacity of just over 200.

The optimizations for capacities 200 and 220 start at 06:00, whereas the optimizations for capacities 150 and 170 start at 05:00, in order to be able to pre-release all (180) orders according to the upcoming route heuristic at 06:00. Simulations with capacity 140 starting at 05:00 and with capacity 180 starting at 06:00 resulted in violations of the pre-releasing capacities, due to the high number of pre-releases from the upcoming route heuristic in the morning.

TABLE 5.8. Optimization costs for various heuristics under different PR_capacities, averaged over 7 days. The costs for the Random heuristic are averaged over 10 simulations.
PR_Horizon = Time_Window = 5 and 2.

PR_Horizon = Time_Window = 5

PR capacity	**Random**	**Closest**	**Furthest**	**Minimum Driving Time**	**Maximum Distance**
220	55682	53995	53723	54700	55265
200	55835	53951	53742	53967	55752
170	57316	54188	54962	56235	57926
150	59312	54609	54354	55142	58351

PR_Horizon = Time_Window = 2

220	53241	53955	53712	53640	54447
200	53483	53697	53874	54119	53922
170	54022	54309	54472	53763	55821
150	54127	53566	54088	53763	54751

It should be noted that these results are all very close to each other, and given the restricted dataset no definite conclusions can be drawn. However, some *hypotheses* can be formulated:

- **Pre-releasing from earlier time-windows might be more desirable.** The Random heuristic performs not much worse as the other heuristics, except for when PR_Horizon = 5. This might indicate that pre-releasing an order from a close time-window is (on average) better than pre-releasing an order from a later time-window. This result motivated the "Next-Timewindow" heuristic that will be tested later.
- **No one heuristic clearly outperforms the others, but Closest and Furthest seem to be the best.** The Closest and Furthest heuristic always have a cost below 55000 and have a lower cost on average. This

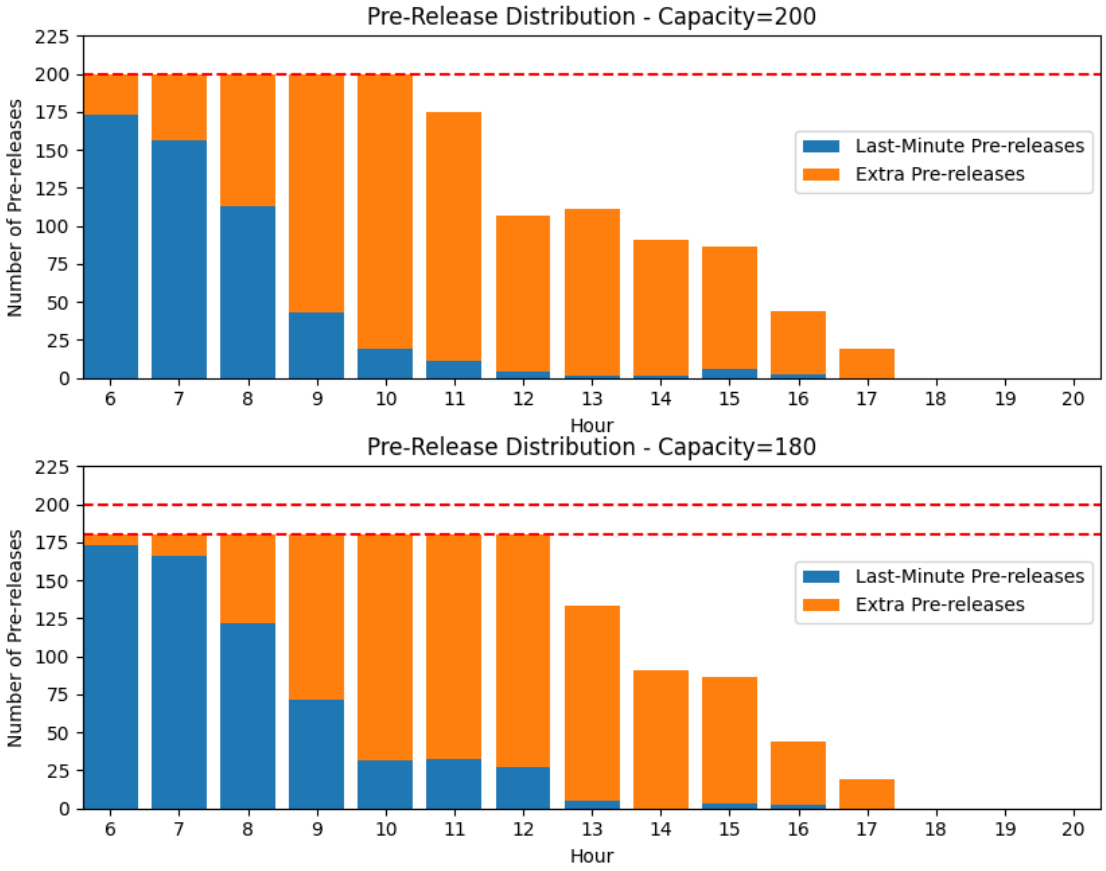

FIGURE 5.6. Two stacked bar charts that illustrate the distribution of pre-releases, showcasing the upcoming route pre-releases at the bottom and the extra pre-releases stacked on top. The dashed red line represents the pre-release capacity. This figure is from the simulation of day 1, using the Minimum Driving Time heuristic

motivated the "Furthest Seeding" and "Closest Seeding" heuristics that will be tested later.

- **Higher PR_capacities seem to lead to lower optimization costs.** Especially for the PR_Horizon = 5 simulations, the lower capacities (which start at 05:00) have a much higher cost than the higher PR_capacities.

To better understand the effects of always pre-releasing to the pre-releasing capacity, it's interesting to observe the actual pre-release rate throughout the day, as shown in Figure 5.6. In particular, the number of "Last-Minute" pre-releases versus the number of "Extra" pre-releases is of interest.

At the start of the day, most of the pre-releasing capacity is used to pre-release according to the "Last-Minute" heuristic. However, as the day progresses, more and more orders are pre-released in advance due to the "Extra" Minimum Driving Time

heuristic. At 11:00 all known orders are pre-released, so the pre-releasing capacity cannot be used fully. After that point, only the orders that just came in can be pre-released with the upcoming route heuristic if they are planned to depart in the next hour. Note that without the PR_Horizon, more "Extra" pre-releases would have taken place at 11:00, but these orders were not allowed to be pre-released yet.

This figure indicates the problem with always pre-releasing to the capacity; at some point you pre-release orders while you could have afforded to wait a bit longer, as no future peaks are expected.

All "Extra" heuristics face this effect, though not necessarily to the same extent, as some heuristics might prefer to pre-release "later" orders, resulting in more "Last-Minute" pre-releases. However, this effect is assumed to be small, as there is no indication in the data for this; the number of "Extra" pre-releases is similar for all heuristics.

3.1. Next Time-Window heuristic. The hypothesis that pre-releasing from earlier time-windows might be more desirable led to the development of the Next Time-Window heuristic, which first pre-releases all orders of the next time-window. The results of these simulations is shown below in Table 5.9 for PR_Horizon = Time_Window = 2 and 5.

TABLE 5.9. Optimization costs for the Next Time-Window heuristic under different PR_capacities, averaged over 7 days.

PR capacity	**PR_Horizon = 5**	**PR_Horizon = 2**
220	54498	53535
200	54327	53579
170	54815	54723
150	54145	53907

Again the results are inconclusive, but the Next Time-Window heuristic seems to have similar performance to the Minimum Driving Time heuristic. Also, the Next-Time Window heuristic seems to perform better for shorter PR_Horizons.

3.2. Seed heuristics. The hypothesis that the heuristics perform worse when the number of pre-releases per route is large inspired the development of the Furthest and Closest Seeding heuristics, which first pre-releases s orders from each route according to the Furthest or Closest heuristic, and applies the Minimum Driving Time heuristics for the remaining pre-releases. The performance of these heuristics is shown in Figure 5.7, where the average optimization cost over 7 days is shown for $s = 0$ up to $s = 10$, for a pre-releasing capacity of 170 and a PR_Horizon of 5.

As expected, both heuristics have the same performance as the Minimum Driving Time heuristic for $s = 0$. For higher values of s, the optimization cost seems to go down, approaching the optimization costs of the Furthest and Closest heuristic. This suggests that the Seeding heuristic performs worse than the heuristics without seeding, for all values of s.

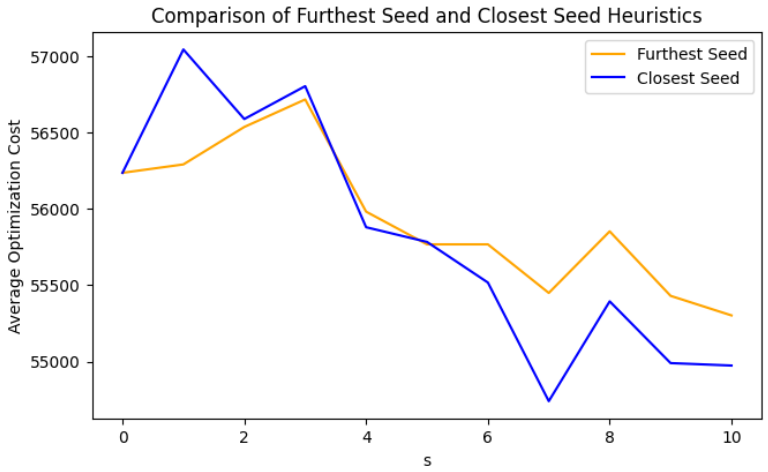

FIGURE 5.7. The average optimization costs over 7 days for various values of s, for both the Furthest Seed and the Closest Seed heuristic. Picking capacity is 170 and Time_Horizon = PR_Horizon = 5.

CHAPTER 6

Conclusion and Recommendations

"A bird does not sing because it has an answer.
It sings because it has a song."
~ Chinese Proverb

This final chapter will summarize the main conclusions from this research, aiming to answer the research questions. First, the sub-questions are answered one by one. Together, these answers resolve the main research question: How should BosMart pre-release their orders and which procedural improvements are cost-effective?

Next to the practical advise for e-grocery businesses, more details on the effect of various heuristics and parameters on the problem are described. Finally, the limitations of this research are discussed, and suggestions for future research on pre-releasing are provided.

1. Answers to Research Questions

1.1. Sub-Question 1: What is the best pre-releasing heuristic? The best-performing heuristic (the one with the lowest average optimization cost) for the pre-releasing problem is the last-minute heuristic "upcoming routes". This heuristic pre-releases all orders and finishes all routes scheduled to depart before the next optimization period, plus the PR_Time. This heuristic significantly outperforms the baseline, obtaining on average 21% cost reductions when 2 time-windows per truck are allowed, and 32% savings when more than 2 time-windows per truck are allowed. This is within 1% of the offline benchmark of OHD-IH.

Importantly, the pre-releasing rate in these simulations seldom exceeded 200 orders per hour, which aligns with BosMart's current capacity. This implies that the suggested plannings would be feasible to execute. However, this last-minute strategy does pose an increased risk of late deliveries, especially in the face of warehouse issues or unexpected order surges. Therefore, a pre-releasing buffer might be desirable from a business perspective.

The optimal way to establish this buffer is by applying the last-minute heuristic 1 or 2 hours earlier. For instance, a route scheduled to depart at 11:10, needing to be pre-released by 10:00 at the latest, could be pre-released at 09:00 or 08:00 instead. Simulations have shown that this strategy increases the optimization costs by a few percent; on average only 1% when pre-releasing 1 hour early, and 4% when pre-releasing 2 hours early. This doesn't affect the pre-releasing capacity required, but it does necessitate that pre-releasing starts before 06:00 to avoid an unmanageable pre-releasing rate at 06:00. Simulations have shown that starting

pre-releasing 2 hours early from 04:00 results in 5% higher costs compared to last-minute pre-releasing beginning at 06:00. Starting the night before at 21:00 results in 7% higher costs.

In the case where the pre-releasing capacity is always used fully, "Extra" heuristics are required to pre-release orders on top of the pre-releases from the "Last-minute" heuristic. Eight of such heuristics have been tested for various pre-releasing capacities and PR_Horizons. On average the best heuristic costs 8% more compared to only pre-releasing with only the "Last-Minute" heuristic. Also, pre-releasing orders with a close time-window appeared to be preferable. This strategy would allow the pre-release capacity to be lowered from 200 to 150, which would save 5 full-time pickers in the warehouse.

1.2. Sub-Question 2: What is the effect of the problem parameters on the optimization costs? Based on applying the heuristics under different sets of parameters, the following results are found:

- `Use_Capabilities`: Allowing more than 2 time-windows per truck improves the potential cost savings from 20% to 32%
- `PR_Time`: Pre-releasing not at the last minute but 1 hour earlier costs 1% more, 2 hours earlier 4%, 3 hours earlier 12% and 4 hours earlier 40%, although the pre-releasing rate at 06:00 would be very high.
- `Optimization_Start_Times`: Pre-releasing 2 hours earlier starting at 04:00 instead of 06:00 costs 5% more, whereas starting at 21:00 costs 7% more.
- `Time_Horizon`: There is no significant difference in optimization costs for Time_Horizons between 3 and 14, but the simulation time is lower for lower Time_Horizons.

1.3. Sub-Question 3: Which procedural improvements would be cost-effective? One obvious improvement would be to allow more than 2 time-windows per truck. When using the upcoming route heuristic, this increases the cost savings from 21% to 33%. The extra peaks in the pre-releasing rate are not severe; increasing the capacity by 30 would be sufficient. This could be done by hiring 3 extra pickers, but relocating the pre-releasing capacity from the afternoon to the morning (06:00 - 08:00) and evening (17:00 - 19:00) is also possible.

Next, pre-releasing should be done at most 2 hours before the last-minute. Pre-releasing 2 hours early is still within 4% of the best-performing heuristic, whereas pre-releasing 4 hours early (as BosMart currently does) incurs over 30% extra costs.

Pre-releasing orders in the morning (starting at 04:00) instead of the previous evening (starting at 21:00) saves 1-2% in optimization costs. However, starting that early might not be preferable for warehouse staff.

Various "Extra" heuristics have been tested for when the pre-releasing capacity is limited and the pre-releasing rate is fixed, but with inconclusive results. The simulations suggest that the Closest and Furthest heuristic perform best, with an average optimization cost of approximately 54000, which is 8% higher than pre-releasing with infinite pre-releasing capacities. However, using these Extra heuristics allows to decrease the pre-releasing capacity to 150, compared to the 200 that would be necessary for applying only the "Last-Minute" heuristic. This would save BosMart 5 full-time pickers in the warehouse, which might be worth the 8% extra costs.

1.4. Main Question: How should BosMart pre-release their SDD-orders and which procedural improvements are cost-effective? Based on the simulation results and analysis, the most effective strategy for BosMart to pre-release their orders is by employing the last-minute "upcoming routes" heuristic. This approach pre-releases all orders and finalizes all routes that are planned to depart before the next optimization starts, plus the PR_Time. This strategy has consistently shown to be the most cost-effective compared to others.

However, a purely last-minute strategy comes with an inherent risk of delivering orders too late in cases of unexpected warehouse issues or sudden peaks of orders. To mitigate this, a pre-releasing buffer could be established by applying the last-minute heuristic 1 or 2 hours earlier. Simulation results have shown that this only increases the optimization costs by 1-4%, thus making it a feasible and cost-effective adjustment.

Beyond these, several procedural improvements could be considered to further enhance cost-effectiveness:

- Allowing more than 2 time-windows per truck could increase cost savings from 21% to 33%.
- Pre-releasing orders 2 hours early can maintain optimization costs within 4% of the best-performing heuristic.
- Starting pre-releasing in the early morning rather than the previous evening could result in 1-2% savings in optimization costs.

BosMart's wish to have a constant pre-releasing rate would cost 8% more than only using the "Last-Minute" heuristic, but would allow the pre-releasing capacity to be decreased from 200 to 150, which would require 5 fewer full-time pickers.

2. Surprising Results

Having answered the research questions, it might be instructive to investigate and learn from the most unexpected results from the simulations. As Isaac Asimov once said: "The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny...'".

First, the fact that using a Time_Horizon larger than 3 performed as good as optimizing over all orders was unexpected (which is why many simulations have used the suboptimal Time_Horizon of 100). It was assumed that more information would yield a better result. One possible explanation might be that the extra information "distracted" OHD from the immediate pre-releasing problem. As only the orders from the first 5 time-windows can be pre-released, all orders after that only impact the solution indirectly. Their inclusion might, for instance, decrease the time spent on local search for upcoming routes. These reasons might also explain the fact that pre-releasing 1 hour before the last-minute sometimes outperformed pre-releasing at the last-minute.

The second surprise was the lack of peaks in the pre-releasing rate, even when using the last-minute heuristic. It was expected that most orders would need to be delivered in the morning or evening, and this would result in high peaks in the pre-releasing capacity, which is also the reason BosMart was pre-releasing suboptimally in the first place. One explanation for this relates to the way time-windows are offered to customers. Using a cheapest insertion method, the less busy time-windows are recommended more frequently (or at a lower price) than the busier time-windows, resulting in a more evenly distributed pre-releasing.

The third surprising result emerged when comparing the performance of the (infeasible) offline benchmark solution obtained by OHD-In-Hindsight, which could optimize the VRP with complete information, against the heuristics that had to make pre-releasing decisions without having knowledge of all the orders. In some cases, the heuristics outperformed OHD-In-Hindsight, even though they had to operate with limited information. For instance, when only two time-windows per truck were allowed, the heuristics were able to plan more orders compared to OHD-In-Hindsight. Additionally, in scenarios where more than two time-windows were permitted, the heuristics achieved lower optimization costs than OHD-In-Hindsight (as observed on day 6 in Table 5.2). These results demonstrate that OHD-In-Hindsight did not always yield the optimal solution.

One potential explanation for the superior performance of the heuristics could be attributed to the hourly optimization process, allowing for more refined and localized search. However, the fact that the use of a Time_Horizon had minimal effect weakens this argument.

3. Limitations of this Research

While this research provides valuable insights into the optimization of pre-releasing SDD-orders, it is important to acknowledge its limitations. One of the primary limitations stems from the narrow scope of the data used. The fact that only seven days' worth of data from one company were used limits the generalizability of these findings to other instances.

Another limitation is the research's assumption that routes can only be planned with a single trip, as opposed to multi-trip plannings. This restricts the flexibility in the plannings because routes always have to wait until the start time of the next trip instead of being available for loading immediately after they have returned to the depot. This restricts not only the "black box" VRP solver, but also the pre-releasing options. For instance, a multi-trip pre-releasing strategy might use a combination of long and short routes, where the long routes improve efficiency and the short routes create flexibility to handle last-minute orders.

The question of how many extra orders to pre-release is not in the scope of this thesis. Instead, it is assumed that the pre-releasing capacity is always fully used. This is not an ideal strategy, as it would mean that after a few hours almost all orders will be pre-released, especially if the pre-releasing capacity is high. Even though this does offer the same conditions to each of the heuristics, the performance of the heuristics might be different when pre-releasing only a few extra orders.

4. Future Research

The pre-releasing problem, although not widely studied until now, is set to become a prevalent issue in many businesses in the coming years. This research has underscored the importance of optimizing pre-releasing, as potential cost savings of up to 33% are substantial.

Future research could address the limitations of this study by testing on a wider variety of datasets and using multi-trip planning. However, there are also other interesting areas for exploration:

- **Estimating the risk of peaks.** This study used a fixed set of orders arriving at fixed times. However, it would be useful to estimate the probability of exceeding pre-releasing capacity, for instance, a 10% probability

when the capacity is 200, 1% if the capacity is 250, etc. This would require simulating orders stochastically, which was beyond the scope of this thesis.

- **Heuristics for the quantity of orders to pre-release.** This research primarily focused on which orders to pre-release. But to strike the right balance between preparing for pre-releasing peaks and keeping optimization costs low, it might be interesting to experiment with the number of orders that are pre-released in addition to those determined by the last-minute heuristic. For example, if the number of orders for the evening is higher than normal around noon, it might be necessary to start pre-releasing extra.
- **Studying the effects of the order placement cut-off time.** This study allowed customers to place orders starting more than 2 hours in advance. Investigating the effects of increasing this time between placement and delivery could be beneficial, as this would likely enable strategies with lower optimization costs. The impact of decreasing this time could also be researched, although this would first require logistical improvements in the warehouse, such as reducing the time it takes to pre-release or load orders.
- **Testing the more complicated heuristics with penalties and multi-simulation.** In Chapter 3 various more complicated heuristics are suggested, which have not been tested in this thesis. Especially the multi-simulation heuristic is expected to perform well, especially when the number of simulations is large.
- **Residue** Allow upcoming routes to exceed PR capacity sometimes, provided that these orders can be delivered on time in later routes.

References

[1] Barrie M Baker and MA1951066 Ayechew. "A genetic algorithm for the vehicle routing problem". In: *Computers & Operations Research* 30.5 (2003), pp. 787–800.

[2] Russell W Bent and Pascal Van Hentenryck. "Scenario-based planning for partially dynamic vehicle routing with stochastic customers". In: *Operations Research* 52.6 (2004), pp. 977–987.

[3] Dimitris J Bertsimas. "A vehicle routing problem with stochastic demand". In: *Operations Research* 40.3 (1992), pp. 574–585.

[4] George Clarke and Jules W Wright. "Scheduling of vehicles from a central depot to a number of delivery points". In: *Operations research* 12.4 (1964), pp. 568–581.

[5] George B Dantzig and John H Ramser. "Truck dispatching problem". In: *Management science* 6.1 (1959), pp. 80–91.

[6] Marco Dorigo and Luca Maria Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem". In: *IEEE Transactions on evolutionary computation* 1.1 (1997), pp. 53–66.

[7] Fausto Errico et al. "A priori optimization with recourse for the vehicle routing problem with hard time windows and stochastic service times". In: *European Journal of Operational Research* 249.1 (2016), pp. 55–66.

[8] Casper Joost Eyckelhof and Marko Snoek. "Ant systems for a dynamic TSP: Ants caught in a traffic jam". In: *Ant Algorithms: Third International Workshop, ANTS 2002 Brussels, Belgium, September 12–14, 2002 Proceedings*. Springer. 2002, pp. 88–99.

[9] Merrill M Flood. "The traveling-salesman problem". In: *Operations research* 4.1 (1956), pp. 61–75.

[10] Michel Gendreau, Gilbert Laporte, and Jean-Yves Potvin. "Metaheuristics for the Capacitated VRP". In: *Vehicle Routing: Problems, Methods, and Applications*. Ed. by Paolo Toth and Daniele Vigo. SIAM, 2014, pp. 31–70.

[11] Gianpaolo Ghiani, Emanuele Manni, and Barrett W Thomas. "A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem". In: *Transportation Science* 46.3 (2012), pp. 374–387.

[12] Bruce Golden, S Raghavan, and Edward Wasil. *The vehicle routing problem: latest advances and new challenges*. Vol. 43. Springer, 2008.

[13] Philip Kilby, Patrick Prosser, and Paul Shaw. "Dynamic VRPs: A study of scenarios". In: *University of Strathclyde Technical Report* 1.11 (1998).

[14] Roberto Montemanni et al. "Ant colony system for a dynamic vehicle routing problem". In: *Journal of combinatorial optimization* 10 (2005), pp. 327–343.

[15] *Navigating the market headwinds - the state of Grocery Retail 2022*. Aug. 2022. URL: https://wp.eurocommerce.eu/navigating-the-market-headwinds-the-state-of-grocery-retail-2022/.

[16] Harilaos N Psaraftis, Min Wen, and Christos A Kontovas. "Dynamic vehicle routing problems: Three decades and counting". In: *Networks* 67.1 (2016), pp. 3–31.

[17] Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis. "An analysis of several heuristics for the traveling salesman problem". In: *SIAM journal on computing* 6.3 (1977), pp. 563–581.

[18] Frederic Semet and Eric Taillard. "Solving real-life vehicle routing problems efficiently using tabu search". In: *Annals of Operations research* 41 (1993), pp. 469–488.

[19] Jørgen Skålnes et al. "The multistage stochastic vehicle routing problem with dynamic occasional drivers". In: *Computational Logistics: 11th International Conference, ICCL 2020, Enschede, The Netherlands, September 28–30, 2020, Proceedings 11*. Springer. 2020, pp. 261–276.

[20] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. Vol. 9. Siam, 2002.

[21] Jian Yang, Patrick Jaillet, and Hani Mahmassani. "Real-time multivehicle truckload pickup and delivery problems". In: *Transportation Science* 34.2 (2000), pp. 123–138.

APPENDIX A

Data Analysis

This data analysis aims to provide a more detailed description of the data used for the simulations in this thesis, so that the results can be interpreted with more nuance. This is done by inspecting the orders of Day 1 in Section 1, and the routes of some of the simulations in Section 2. The distribution of the data is similar for most other days, although differences do exist. For example, the number of orders for Sunday morning might be different compared to Monday morning. Table A.1 below shows the total amount of orders for each day.

TABLE A.1. Total order amounts for the 7 days.

Day	1	2	3	4	5	6	7
Orders	1633	1855	1702	1359	1469	1484	1780

To secure anonymity of the "real" client, some details of the data that are not crucial for understanding the experiments are omitted. For example, a scatter plot showing the locations of the orders and the warehouse was not allowed to be shared. However, as the warehouse and all delivery addresses are in a densely populated European city, one might be able to imagine what it would roughly look like.

1. Order Analysis

1.1. Time Received. Orders can come in at any moment throughout the day. It is relevant to know how many orders come in at which moment, as this says something about the degree of dynamicity of the problem. Figure A.1 shows when the orders are received.

At the start of the day, roughly half of the orders are known. The other half of the orders come in throughout the day. Also, some orders come in between 22:00 and 06:00. These orders can be taken into account when pre-releasing starts in the morning instead of in the evening, as is discussed in Section 2.4.

As several pre-releasing strategies have been studied that pre-release a few hours early, it is also crucial to know how much time is between the order placement and the start of the time-window. This information is summarized in Figure A.2 below.

The graph clearly indicates that many orders are placed in a time-window starting in less than 4 hours. This explains why pre-releasing 3 or 4 hours early had such a detrimental effect (see Table 5.5). This behavior is caused by the fact that by default the next available time-window is suggested to the customer.

1.2. Demand for the time-windows. Some time-windows are more popular than others. This might result in the earlier described peaks, in which the required pre-releasing rate exceeds the pre-releasing capacity. Figure A.3 below shows the distribution of orders over the time-windows.

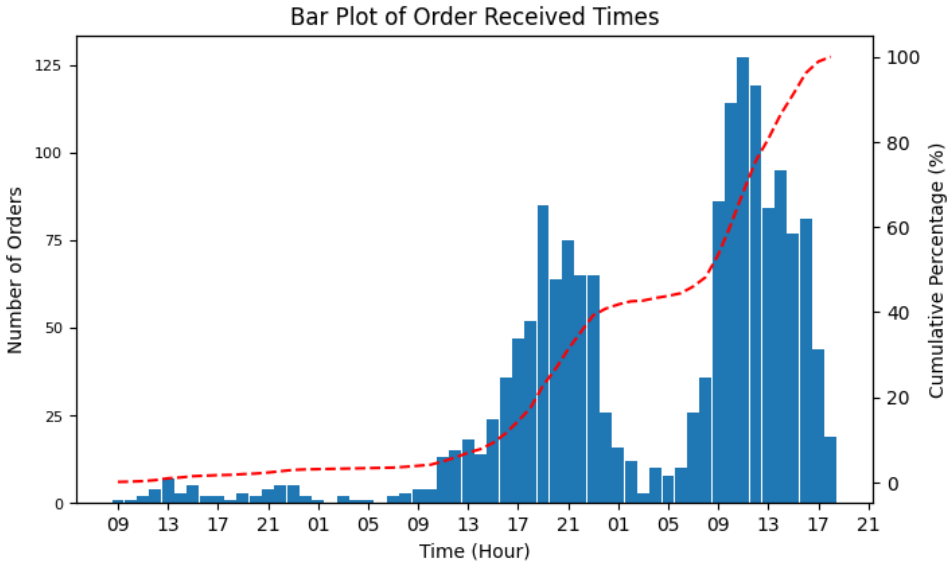

FIGURE A.1. The figure illustrates the distribution of order placement times over a three-day period. Each bar represents the number of orders received within a specific hour. The x-axis displays the hours of the day, while the y-axis represents the number of orders. The cumulative percentage line shows the cumulative percentage of total orders received up to each hour.

The most popular time-windows are early in the morning, and later around 18:00. The fact that the first three time-windows are capped is the result of the time-window heuristic, which limits the number of orders per time-window.

2. Route Analysis

This thesis suggests to allow more than 2 different Time-Windows per truck. The optimal planning indeed often uses more than 2 time-windows per truck, as can be seen in Figure A.4.

The number of orders per route is shown in Figure A.5 for the best-performing heuristic and BosMart's baseline on day 1.

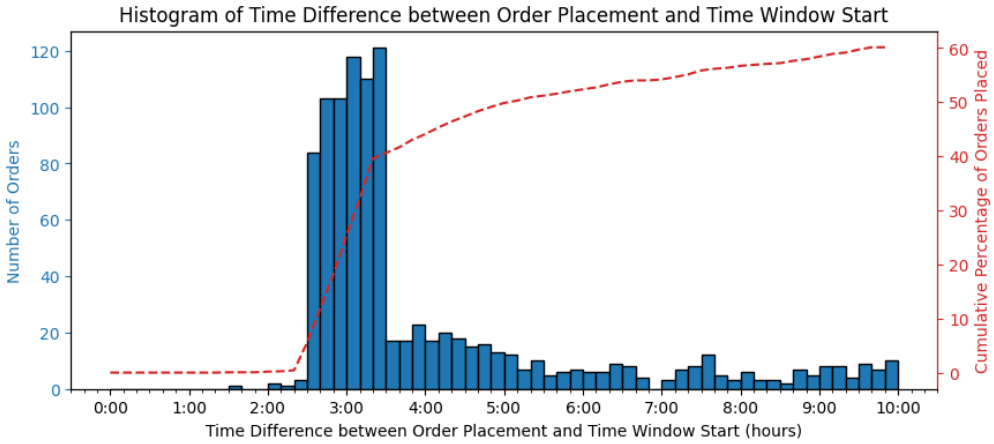

FIGURE A.2. The figure shows the distribution of time differences (in minutes) between the placement of orders and the start of their respective time windows. The histogram summarizes the number of orders within each time difference bin, with bins of 10-minute intervals. The cumulative percentage line indicates the cumulative percentage of orders placed within or before each time difference bin. Note that the 40% of orders that are placed over 10 hours before the start of the time-window are not displayed.

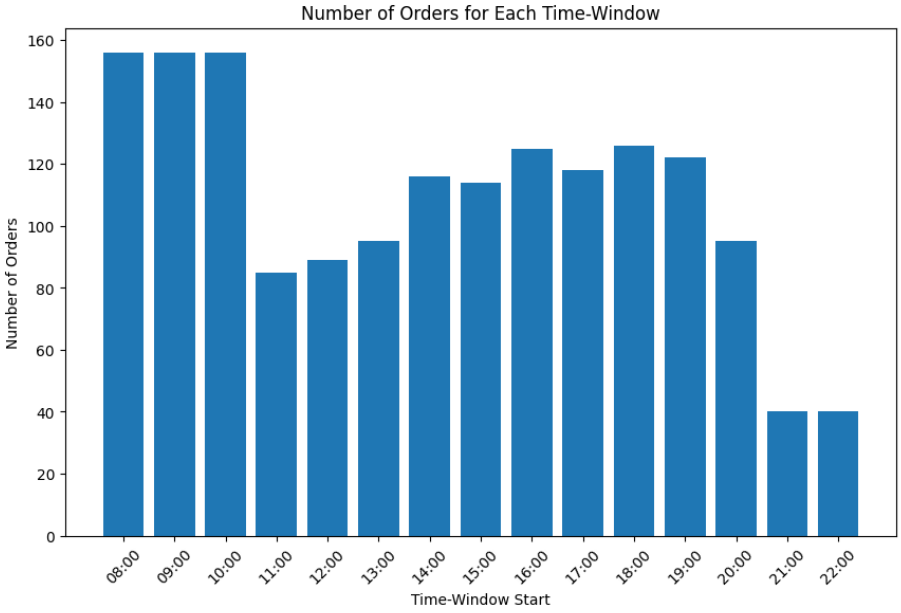

FIGURE A.3. The number of orders per time-window start-time.

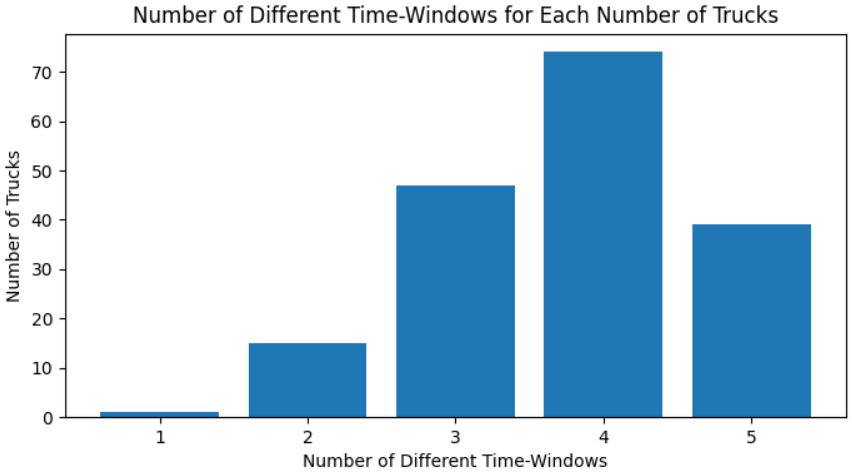

FIGURE A.4. A bar chart showing how many routes have how many different time-windows. These routes are taken from the best-performing heuristic for day 1 (upcoming routes, Time_Horizon=6), with use_capabilities=False.

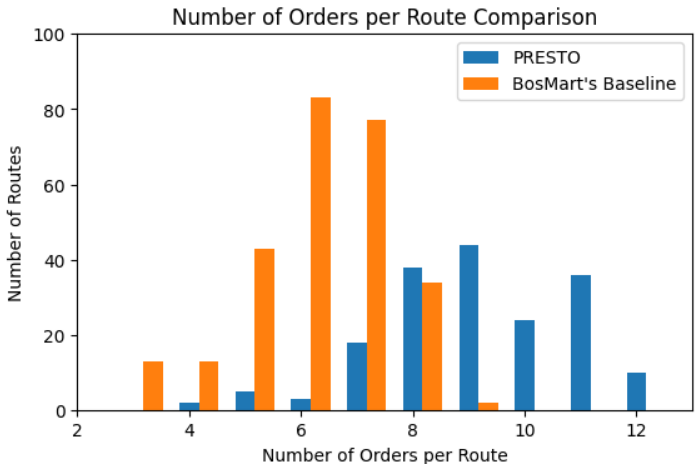

FIGURE A.5. A bar chart showing how many routes have how many different time-windows. These routes are taken from the best-performing heuristic for day 1, with use_capabilities=False.

APPENDIX B

ORTEC

This research has been performed under the supervision of ORTEC, a leading provider of planning and optimization software with over 40 years of experience and over 1000 employees. ORTEC has helped many companies improve their operations, such as Coca Cola, Albert Heijn, and PostNL. The company specializes in developing extensive software solutions to solve various variants of the Vehicle Routing Problem (VRP) and optimize complex logistics and supply chain operations. ORTEC describes their mission statement as follows:

- **Vision:** "We consider it our responsibility to make applied mathematics available in a transparent, safe and sustainable way, so organizations can improve their impact on the world."
- **Purpose:** "We want to improve the world using our passion for mathematics."

As part of their comprehensive suite of solutions, ORTEC offers Home Delivery Optimization, a cloud product that helps planners to improve their plannings on a strategic, tactical, and operational level. By optimizing home delivery operations, retailers can achieve several benefits. ORTEC's Home Delivery Optimization solution can improve customer service levels, enhances route efficiency, improve driver happiness and reduce transportation costs by up to 10%. It incorporates dynamic pricing, time slotting, and routing to balance workload and minimize "no shows". This results in improved customer satisfaction, smoother demand management, and decreased CO_2 emissions.