



universität
wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

“The management of time in Inventory Routing Problems”

verfasst von / submitted by

Emilio Jose Alarcon Ortega

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

Wien, 2023 / Vienna 2023

Studienkennzahl lt. Studienblatt /

UA 794 370 403

degree programme code as it appears on the student
record sheet:

Dissertationsgebiet lt. Studienblatt /

Logistics and Operations Management

field of study as it appears on the student record sheet:

Betreut von / Supervisor:

Univ.-Prof. Mag. Dr. Karl Franz Dörner, Privatdoz.

Acknowledgements

In the first place, I wanted to show my gratitude to my supervisor and advisor Karl F. Dörner for offering me a position as part of the project COSIMA - Consistent Stochastic Inventory Routing Management funded by the Austrian Science Fund (FWF), for his scientific guidance, for allowing me to develop my research topics, for his patience, constructive feedback and for believing in me and my capabilities in the good and most important in the not so good moments.

Additionally, I want to thank the rest of the members of the project COSIMA: Michael Shilde, Sebastian Malicki, and Stefan Minner for their ideas, collaboration, feedback and evaluation of the work here presented and in progress.

My gratitude also goes to my parents, Francisco and Isabel, for their unconditional support and dedication, for believing in me, and for giving me the best education. I also want to thank my brother Paco for being a friend and an example to follow, and bringing to life together with Rocio to my beautiful niece and nephew.

I am really thankful to all my fellow PhD and PostDoc colleagues, Adria, Alina, Christina, Dani, David, Jasmin, Yannick, and many others, for making the last years an exciting and truly enjoyable adventure. Especially to Johanna, for giving me strength in the good and bad moments and encouraging me to go forward. The working and social conversations and plans during this time made me grow professionally and personally. I also want to show my gratitude to Carina Artner-Konecny and Christine Mauric, whose constant help in the department motivates and serves as an example. To my friends in Spain, for being there even in the distance, whenever I needed help they never hesitated to give me a hand, David, Ivan, Torres, Mike, Cris, etc. you are the best friends one could ask for.

I would like to gratefully acknowledge the anonymous reviewers for their constructive criticism and helpful comments.

Preface

This doctoral thesis consists of the research that I have conducted over the past years at the University of Vienna under the supervision of Univ.-Prof. Mag. Dr. Karl Franz Dörner. It comprises various topics of transportation logistics, and in particular, the inventory routing problem. Each chapter of the thesis corresponds with a separate research topic that has been either published or is a submitted manuscript with the exceptions of the introduction as Chapter 1, and the conclusion as Chapter 5.

In particular, Chapter 2 with the title ‘Matheuristic search techniques for the consistent inventory routing problem with time windows and split deliveries’ is published in Alarcon et al. [5] while Chapter 3 with the title ‘A sampling-based matheuristic for the continuous-time stochastic inventory routing problem with time-windows’ is published in Alarcon and Doerner [7]. Chapter 4 with the title ‘Stochastic inventory routing with dynamic demands and intra-day depletion’ has been submitted to *Computers & Operations Research* [1] and is currently under revision.

The results of these works were presented at several conferences: i.a. at the VeRoLog, the annual workshop of the EURO working group on Vehicle Routing and Logistics optimization, in 2017, and 2019; the MIC, the Metaheuristics International Conference, in 2017; the EURO, the Association of European Operational Research Societies, 2018, and in 2019; the TSL, the Odysseus International Workshop on Freight Transportation and Logistics, in 2018; the Ifors Conference, Conference of the International Federation of Operational Research Societies, in 2023 and 2018; and the GOR, Gesellschaft für Operations Research e.V, in 2017 and 2023.

Contents

List of Figures	IX
List of Tables	XI
List of Algorithms	XIII
List of Abbreviations	XV
1 Introduction	1
2 CIRPTWSD	5
2.1 Introduction and Literature Review	6
2.2 Problem Description	10
2.3 Solution Approach	18
2.4 Computational Results	28
2.5 Conclusions and further research	37
3 CTSIRPTW	41
3.1 Introduction	42
3.2 Literature review	44
3.3 Problem Description	46
3.4 Solution Approaches	53
3.5 Computational Experiments	67
3.6 Conclusions and further research	79
4 SIRPID	83
4.1 Introduction	84

4.2 Literature Review	85
4.3 Optimization Model	90
4.4 Solution Methods	92
4.5 Numerical Evaluation	104
4.6 Conclusion	112
5 Conclusion	115
Bibliography	117
Abstract	125
Zusammenfassung	127

List of Figures

2.1	Inventory flow and different characteristics of the problem for a given customer	10
2.2	Components of the method	18
2.3	Example of customer inserted in the priority list of Period 1 with the corresponding upper and lower bound amounts calculated.	20
2.4	Update inventory flow after destroy operator	25
2.5	Update inventory flow after repair operator	26
3.1	Example of the inventory flow of a customer. Taken from [6]	53
3.2	Percentage gap representation of group of instances “4p5a”, “4p5b” and “4p5c”	73
4.1	Daily sales data divided into five sub-periods	89
4.2	Stochastic dynamic decision process of the SIRPID, adapted from Meisel [65]	92
4.3	Flowchart of the adaptive lookahead procedure.	100

List of Tables

2.1	Notation	12
2.2	Characteristics of instances for the CIRPTWSD	30
2.3	Characteristics of real-world instances for the CIRPTWSD	31
2.4	Comparison to benchmark set of instances. Gaps with respect to the known optimal solutions when the algorithm finds solutions without stock- out situations.	33
2.5	Results of our method vs CPLEX. * CPLEX is not able to solve all instances optimally. ** CPLEX is not able to find a feasible solution for some instances.	35
2.6	Impact of time-windows	35
2.7	Real-world instances, seven periods one subperiod.	37
2.8	Real-world instances, seven periods two subperiods.	37
2.9	2-OPT chain length tests	39
2.10	ALNS acceptance parameter tests	40
2.11	Results of MLP vs. OU	40
3.1	Notation	48
3.2	Characteristics of instances for the CTSIRPTW	68
3.3	Average results four periods with low standard deviation.	70
3.4	Average results four periods with medium standard deviation.	71
3.5	Average results four periods with high standard deviation.	71
3.6	Average results eight periods planning horizon.	72
3.7	Average routing costs.	74
3.8	Average stock-out costs.	75
3.9	Average inventory costs.	76

3.10	Average consistency costs.	77
3.11	Impact additional delivery routes. Difference = Routing costs - Inventory costs - Stock-out costs	78
3.12	Impact deliveries in existing routes and additional routes	78
4.1	ALA runtimes depending on instance size [in seconds].	106
4.2	Algorithm performance depending on holding costs.	107
4.3	Algorithm performance depending on instance size.	109
4.4	Algorithm performance depending on coefficient of variation.	109
4.5	Total Cost Savings and runtime comparison of intra-day planning com- pared to full-day planning [in % and seconds].	110
4.6	Savings in holding costs, routing costs, and stock-out costs for intra-day planning depending on holding costs and coefficient of variation	111

List of Algorithms

1	Initial Solution	19
2	ALNS	23
1	ALNS	55
2	Outer procedure branch and regret algorithm	63
3	Inner procedure branch and regret algorithm	64
4	Multiple scenario approach	66
5	Sample Average Estimator	69
6	Outer loop branch and regret algorithm [50]	81
7	Inner loop branch and regret algorithm [50]	82
1	Initial routes generation	101
2	Adaptive policy evaluation mechanism	102

List of abbreviations

ADP	Approximate Dynamic Programing
ALNS	Adaptive Large Neighborhood Search
CC	Chance-Constrained Policy
CD-PFA	Customer Decomposition-Policy Function Approximation
CIRPTWSD	Consistent Inventory Routing Problem with Time Windows and Split Deliveries
COSIMA	Consistent Stochastic Inventory Routing Management
CTIRPTW	Continuous-Time Stochastic Inventory Routing Problem with Time Windows
IRP	Inventory Routing Problem
MDP	Markov Decision Process
MIP	Mix Integer Programing
MLP	Maximum Level Policy
MSA	Multiple Scenario Approach
OU	Order-up-to-level Policy
SAE	Sample Average Estimator
SDP	stochastic dynamic program
SIRP	Stochastic Inventory Routing Problem

SIRPID	Stochastic Inventory Routing Problem with Intra-Day Depletion under Dynamic Demands
SVRP	Stochastic Vehicle Routing Problem
VMI	Vendor Managed Inventory
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

Introduction

This doctoral thesis deals with variants of the inventory routing problem (IRP) which originates in the field of transportation logistics. The rapid development experienced by the transportation industry in the past decades created the need to study many logistic problems beyond the vehicle routing problems. The IRP resembles vendor-managed inventory system (VMI), in which a central supplier manages both, replenishment and distribution decisions to supply a set of customers or retailers. In real-world applications, many companies already have adopted a VMI strategy, where the main motivation lies in the capability of such systems to reduce overall logistic costs. The beverage and grocery industries are especially remarkable in their efforts to reduce their costs by means of central planning.

In this thesis, we perform a research study to find answers to logistic questions related to the IRP. The first question is related to the nature of the demand that industries experience. In general, most of the existing literature related to the IRPs considers demands as an instantaneous event, i.e., such approaches consider that demands occur instantaneously at the end or beginning of each period. However, many real-world businesses experience demands within the periods continuously. These demands can be linear or stationary but, as a result, the time of the period that deliveries take place impact tremendously in the total costs. Furthermore, inefficient delivery plans and late deliveries can translate into a loss of sales due to stock-out situations. The second question is how can the state-of-the-art mathematical formulations be adapted to consider the continuous nature of the demand, along with other characteristics such as the possibility of splitting the deliveries to the customers, the consideration of diverse opening times of such businesses, and the requirement of consistency in the arrival times.

In addition to these research questions, several methods were developed and compared to study their efficiency when solving the problems presented in this thesis. The design of these methods was tested and evaluated providing computational results and analysis of their different components. In particular, the methods used in this thesis are heuristics, metaheuristics, matheuristics, and approximate dynamic programs. I applied adaptive large neighborhood search, multiple scenario approach, branch and regret heuristic, policy function approximation, chance constraints algorithms and an exact mathematical program. For all heuristics, I used both standard operators from the literature and self-developed problem specific operators. Finally, to ensure the efficiency of the algorithms, I have performed a computational study and tested them on instances from the literature, adapted to comply with the respective problem characteristics, and real-world-based instances that use the information of grocery stores, restaurants, bars and other locations in the city of Vienna. To ensure the quality of the algorithms, I tested them on instances from the literature and compared them in terms of both solution quality and computation time.

The remainder of this thesis is organized as follows:

Chapter 2 presents a deterministic IRP that originates in the beverage industry, where business customers such as restaurants, bars, and supermarkets experience demand continuously during the periods. In addition, these customers present business-specific characteristics such as time windows and the need for consistency in delivery times. The scope of this chapter is to propose a mathematical formulation for the problem and a solution method that obtains efficient delivery plans that would reduce the routing, inventory, consistency and stock-out costs originating from late replenishments. The chapter was published in Alarcon et al. [5], with M. Schilde and Univ.-Prof. Mag. Dr. Karl Franz Dörner taking a supervising and guiding role.

Chapter 3 is a continuous-time stochastic IRP. In this case, the research focus of the chapter is the evaluation of robust solutions when the demands that the customers experience are, not only continuous within the time periods, but also stochastic. The

continuous-time stochastic IRP with time windows is formulated as a two-stage mathematical program. Furthermore, to solve this problem, we present different sampling-based matheuristic solution approaches and evaluate the impact of applying recourse actions to deal with expected lost sales during the planning horizon. This work was published in Alarcon and Doerner [7]. I developed the model, implemented the solution algorithm, conducted the computational experiments and wrote the manuscript with advice and guidance from my supervisor.

Chapter 4 deals with a stochastic dynamic inventory routing problem. In this chapter, the research focus is the stochastic and dynamic nature of demands which, in real-world applications differ on both a daily and intra-day basis. Therefore, to account for the non-linearity of the demands that customers experience during the periods we perform a division of the periods into subperiods with independent demands. We formulate the problem as a finite-horizon stochastic dynamic program and propose an iterative look-ahead solution approach to solve the problem. Then, the performance of the proposed algorithm is evaluated with respect to different benchmark replenishment policies and a policy function approximation algorithm. The results support the assumption that when the demand consumption that the customers experience is not continuously linear during the periods, utilizing intra-day demand information allows the central supplier to create more efficient replenishment schedules when compared to full-day planning with aggregated demand information. This work has been submitted to *Computers & Operations Research* [1] and is currently under revision. The initial motivation was provided by my co-authors with whom I specified the problem. I developed the model, implemented the solution algorithm, conducted the computational experiments and wrote the manuscript in collaboration with my co-author S. Malicki.

For all chapters that correspond to already published work, I have obtained the right to include the submitted version in my thesis. The only changes made, and the only differences to the published versions, concern formatting, structural changes (renumbering of sections), and updates to cited references. These articles were motivated

by the project COSIMA, Consistent Stochastic Inventory Routing Management, founded by the FWF which I thank for their support.

Matheuristic search techniques for the consistent inventory routing problem with time windows and split deliveries

Published in: *Operations Research Perspectives*

Matheuristic search techniques for the consistent inventory routing problem with time windows and split deliveries.

E.J. Alarcon Ortega, M. Schilde, & K.F. Doerner. © 2020 The Authors.

<https://www.sciencedirect.com/science/article/pii/S2214716020300427>

Abstract This article introduces a new variant of the inventory routing problem related to real-world businesses. Specifically, in the beverage industry, business customers such as restaurants and bars, demand consistent delivery times, have different opening times and delivery time-windows, and occasionally, due to special events, exhibit demands that exceed single-vehicle capacity leading to the need of splitting demands between several vehicles. We present two variants of a mathematical formulation that include all the characteristics of this inventory routing problem. In the first, we apply the maximum level policy, whereas in the second variant, we apply an order-up-to-level policy. As a solution technique, we propose a matheuristic based on an adaptive large neighborhood search algorithm for which we developed several destroy and repair operators specifically designed to address the special problem features. Extensive computational tests based on artificial and real-world instances affirm the efficiency of the solution approach. Furthermore, we analyze the solution

quality, the impact of the characteristics and policies applied, and the practicability for the real world.

2.1 Introduction and Literature Review

The consistent inventory routing problem with time-windows and split deliveries (CIRPTWSD) arises in the beverage industry, where transportation plans must meet multiple requests from customers and various characteristics create the need to develop innovative, efficient solution approaches for distribution and stocking decisions. Many companies already have adopted VMI systems, such that the supplier manages all the replenishment and distribution plans centrally, largely because the application of VMI can substantially reduce overall logistics costs [62]. Within the VMI framework, the IRP, first introduced in the seminal paper [18], aims to deal with this situation. Beverage industries, in particular beer industry, are especially remarkable in their central planning efforts to reduce overall logistics costs. Although beer consumption per capita and year is quite stable, consumption is very sensitive to external factors such as weekends, holidays, and special events (e.g., sports events, music festivals). Moreover, it encompasses different types of customers, such as bars, restaurants and retailers many of which have different opening hours and delivery time-windows. Hence, to deal with these differences between opening hours, to set a maximum driving time and due to different working shifts, customers divide each period into different subperiods (e.g. morning, afternoon, evening). Another special characteristic that customers present is the demand of consistency in the delivery times, to enable themselves to prepare for a delivery. In addition, temporary high demands and the need to deliver to every customer creates a distinct possibility of splitting deliveries across multiple trucks.

One critical characteristic, in the real world context where we describe our problem, is that the product is consumed continuously within the time periods. This characteristic is often present in articles about inventory management, however, it is a relatively new characteristic in the VRP and IRP literature, [60]. In this paper, the authors refer to this variant of the IRP as continuous-time IRP. In this work, authors study the critical components of a dynamic discretization discovery algorithm, where the algorithm aims to discover which times are needed to obtain an optimal solution by solving a small sequence of integer programs. Authors deal with the problem of replenishing a set of customers over a finite planning horizon but, unlike

the problem we describe in this work, stock-out situations are not considered, the replenishment plan is performed over a single period and consistency in the deliveries and time-windows are not considered. Due to this continuous consumption of product, high additional efforts must be driven to carefully plan delivery times and amounts. Customers that are visited too late in time can lost sales due to stock-out situations. On the other hand, early deliveries to customers can be unprofitable, as the total amount delivered to the customer can be too low and, therefore, additional deliveries can be necessary in the subsequent periods.

Our contributions in this paper are threefold: 1) We introduce and mathematically formulate a new variant as CIRPTWSD where the product is being consumed continuously at the customers and they demand consistency in the delivery times in order to anticipate and be ready to receive them. We present two different versions of the mathematical formulation using two different replenishment policies, the order-up-to-level policy (OU) and the maximum level policy (MLP). The first policy requires that every visit to a customer means the entire inventory capacity is filled. The second policy provides more flexibility to decide delivery amounts, such that for each customer visit, any amount can be delivered as long as inventory capacity is respected [12]. 2) We propose a matheuristic solution method based on an adaptive large neighborhood search algorithm (ALNS) to deal with the CIRPTWSD. In this algorithm several destroy and repair operators are applied to an initial solution to iteratively remove and rebuild parts of the solution. This algorithm also includes a mathematical subproblem where, given a fix plan of deliveries to the customers, the arrival times and delivery amounts are optimally calculated. 3) We evaluate the performance of the proposed algorithm and the impact of the characteristics considered in the problem. In order to evaluate the effectiveness of the algorithm, we solve a modified and more simple version of the algorithm to solve a benchmark set of instances for the IRP introduced in [12]. We then propose an adapted set of instances from another benchmark set and a real world based set of instances to conduce the experiments.

The CIRPTWSD belongs to the family of vehicle routing problems (VRPs), first present in [39], and, more specifically, the group of IRPs. The family of IRPs, are NP-hard problems and, it is in particular the CIRPTWSD, as it can be considered an extension of the VRP with time-windows (VRPTW) when considering a single period and subperiod (the VRPTW is an NP-hard problem [76]). Extensive reviews of IRP literature are available in [24] and [34], and in [9] we find an overview of different industrial applications of IRPs. Several articles include different aspects of the CIRPTWSD, and the two key features that represent the focus of our

research, consistency and split deliveries, are widely discussed in [33], [58], and [15]. With regard to consistency, Coelho et al. [33], present different interpretations of consistency (quantity consistency, driver consistency, and others), but not consistency in delivery times, that is the interpretation of consistency that the CIRPTWSD presents. Therefore, we propose defining deliveries to a customer as consistent if all deliveries arrive at the same time of day. Kovacs et al. [59] introduce consistency in delivery times in a mathematical model, by ensuring that the arrival time difference to a customer is lower than a certain parameter value. Further insights into VRPs in which consistency is an important characteristic are available in [58], as well as [57] and [30]. Turning to split deliveries, each customer can be serviced by more than one vehicle in the same period and subperiod. Recent studies ([70],[48]) introduce new applications for split pick-ups and split deliveries, first with discrete commodities and then in real-world situations related to the fuel industry. Furthermore, Christiansen [32] presents a problem with both inventory management and the possibility of split deliveries. Finally, Archetti and Speranza [15] consider the possibility of split deliveries in VRP problems with a single period, while also noting the main properties and different solution approaches for dealing with several variants of the basic VRP with split deliveries.

Another key characteristic, which we introduce herein, is time management combined with inventory routing. We consider an IRP with intra-day time-windows. In most existing literature, time-windows refer to a group of periods during which a delivery can take place, instead of a time interval within time periods. But because each customer has different opening times, we seek to operationalize delivery time-windows in a way more commonly adopted in research into VRP problems, such as in Azi et al. [17]. Although this approach is less common in relation to IRP, some works consider these types of time-windows, [63]. As another contribution, related to time and commodity management, we address linear commodity consumption by customers. This characteristic is also present in [60], while the possibility of lost sales is not considered. Combining linear consumption and the existence of intra-day time-windows enables us to account for the possibility of lost sales due to stockout situations, not just at the end of each period, but also within time periods. In our work, lost sales may occur if customers do not have enough goods to satisfy corresponding demand before the next delivery. In our formulation, we penalize the amount of lost sales in the objective function.

Despite the recent introduction of some exact solution methods for IRPs ([4], [40], [35], [14]), we seek to solve large, real-world instances, so we chose a heuristic method. Heuristic methods

are often used to deal with IRP variants, such as the combination of an integer programming approach and variable neighborhood search approach to deal with blood inventory and supply problems [47] or a template-based adaptive large neighborhood search applied to deal with the consistent vehicle routing problem [59], with template routes that include frequently serviced customers, then an introduction of sporadic customers in the template routes. An ALNS [33] might solve different interpretations of consistent IRP problems, which provides a heuristic solution for multi-vehicle inventory routing problems. Other solution approaches for IRP include a matheuristic solution approach [13], a variable neighborhood search [69], and a decomposition approach [29]. Notably, we find increasing research interest in developing solution approaches to deal with IRPs with stochastic demands, such as with the introduction of two simheuristic approaches for the stochastic IRP with stockouts for single and multiple period ([53], [45]).

In Section 2.2, we present a new formulation for the CIRPTWSD that integrates inventory management, vehicle routing, and delivery scheduling decisions with the previously detailed characteristics. In Section 2.3, we present a matheuristic solution approach in which we solve a linear subproblem based on the problem formulation of the CIRPTWSD, using the idea of an ALNS. We apply several destroy and repair operators to an initial solution obtained using a cheapest insertion heuristic, followed by two local search operators, then explore different solution neighborhood operators while, iteratively assigning more importance to the most successful neighborhoods by increasing the probability of using them. After we describe both, the mathematical formulation and the proposed solution approach, in Section 2.4, we present an extensive computational study. To evaluate the effectiveness of the solution approach, we use a set of instances of different sizes, adapted from a benchmark set. We also compare the results obtained by an exact solver with the results obtained using our proposed method. Furthermore, we study the impact of different time window sizes with respect to the different cost factors. Our algorithm provides high quality solutions for small instances after short computation times, compared with the exact solver, but for instances of medium and large sizes, the matheuristic provides better solutions than the exact solver; the solver cannot find optimal solutions within a given computational time limit of ten hours. In Section 2.5 we summarize the findings and propose additional research questions related to the CIRPTWSD.

The CIRPTWSD is represented on a complete directed graph $G = (V, A)$, where V is the set of all nodes and A is the set of all arcs. We denote the depot as 0 and V' as the set of customers. We consider a finite planning horizon with $p \in P$ periods; each of these periods is divided into a given number of subperiods $r \in R$ with length T_r . We apply this division to deal with two issues. First, it helps us state a maximum route duration. Second, it represents a real-world situation caused by the different working shifts that a company has. We consider a finite and homogeneous fleet of vehicles, $k \in K$, with capacity Q . A cost c_{ij} and a time t_{ij} , both non-negative values, are considered for each arc $(i, j) \in A$. Furthermore, for each customer $i \in V'$ and subperiod $r \in R$, a_i^r and b_i^r represent the start and end times when the customer can be served. Initial inventory levels $I_i^{0|R|}$ are assigned to each customer $i \in V'$ and so are the inventory capacities C_i . To assign these initial inventory levels, we create a dummy period 0, and we assign the initial inventory levels to the last subperiod $|R|$ of this dummy period. This way, we can use the same notation for all the inventory information about the problem. The service time for delivering to a customer is represented as s_i . Finally, d_i^{pr} represents the commodity consumption of customer i in period p and subperiod r . We consider this consumption continuous within subperiods.

In our model, we also include different decision variables. The binary variables x_{ij}^{kpr} and y_i^{kpr} indicate if an edge (i, j) is being used by vehicle k in period p and subperiod r and if customer i is visited by vehicle k in period p and subperiod r , respectively. Moreover, we include continuous variables related to time and commodity management. The variables t_i^{kpr} , \bar{t}_i^r , and \underline{t}_i^r relate the time and consistency management of the problem. The first type represents the arrival time of a vehicle k to a customer i in period p and subperiod r over all periods. The latter two types of variables represent the earliest and latest arrival time to each customer i for each subperiod r . Furthermore, σ^{kpr} represents the loading time of each vehicle at the depot in each period and subperiod. Finally, we have other groups of decision variables associated with the commodity management. First, variables q_i^{kpr} represent the amount of commodity delivered to each customer i by each vehicle k in period p and subperiod r . Second, I_i^{pr} shows the inventory levels of each customer i at the end of each period p and subperiod r . Third, Δ_i represents the difference between the initial and the ending inventory level of each customer i , if the initial inventory is higher than the ending inventory. We use Δ_i to create a pseudo-rolling horizon and to avoid empty ending inventories. Fourth, o_i^{pr} represents the amount of lost sales at customer i in period p and subperiod r due to stockout situations. Fifth, because of the split delivery characteristic of the problem, we introduce $A_i^{kk'pr}$ to represent the amount q_i^{kpr} delivered by vehicle k to customer

i if this vehicle arrives before vehicle k' in period p and subperiod r , and 0 otherwise. The notation used in the mathematical formulation is summarized in Table 3.1.

Data Sets	V	set of nodes
	V'	set of customers
	K	set of vehicles
	P	set of periods
	R	set of subperiods
Data and Parameters	c_{ij}	travel cost for arc (i, j)
	t_{ij}	travel time for arc (i, j)
	a_i^r, b_i^r	time-windows of customer i in subperiod r
	C_i	inventory capacity of customer i
	$I_i^{0 R }$	initial inventory level of customer i in dummy period 0 and subperiod $ R $
	s_i	service time of customer i
	d_i^{pr}	commodity consumption of customer i in period p and subperiod r
	Q	vehicle capacity
	T_r	duration of subperiod r
	x_{ij}^{kpr}	use of arc (i, j) by vehicle k in period p and subperiod r
Decision Variables	y_i^{kpr}	visit to customer i in period p and subperiod r
	I_i^{pr}	inventory level of customer i at the end of period p and subperiod r
	t_i^{kpr}	arrival time of vehicle k to customer i in period p and subperiod r
	$\underline{t}_i^r, \overline{t}_i^r$	earliest and latest arrival times to customer i in subperiod r
	σ^{kpr}	load time of vehicle k in period p and subperiod r
	Δ_i	final inventory decrease of customer i
	q_i^{kpr}	quantity delivered to customer i by vehicle k in period p and subperiod r
	$A_i^{kk'pr}$	previous deliveries to customer i in period p and subperiod r
	o_i^{pr}	quantity lost due to stockouts of customer i in period p and subperiod r

Table 2.1: Notation

Using this notation, we can formulate the CIRPTWSD.

2.2.1 Objective function:

$$\begin{aligned}
 \text{Minimize } & \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} \sum_{p \in P} \sum_{r \in R} c_{ij} x_{ij}^{kpr} + \alpha \sum_{i \in V'} \sum_{r \in R} (\bar{t}_i^r - \underline{t}_i^r) + \\
 & + \sum_{i \in V'} \sum_{p \in P} \sum_{r \in R} Lo_i^{pr} + \sum_{i \in V'} L \Delta_i
 \end{aligned} \tag{2.1}$$

The objective of the problem is to minimize total costs (3.1). We consider four different cost factors to minimize. The total routing cost incurred by servicing the customers, the penalty imposed according to the difference between the earliest and latest delivery to a customer, costs caused by stockout situations, such that the amount of commodity that each customer is not able to serve is weighted by a parameter L (initially set to three times the commodity cost), and the difference between initial and ending inventory levels at each customer, when this difference is positive, measured as sales lost, which enables us to create a pseudo-rolling horizon.

2.2.2 Time and routing flow constraints:

$$\sum_{j \in V} x_{ji}^{kpr} = y_i^{kpr} \quad \forall i \in V', \forall k \in K, \forall p \in P, \forall r \in R \tag{2.2}$$

$$\sum_{r \in R} y_i^{kpr} \leq |R| \quad \forall i \in V', \forall k \in K, \forall p \in P \tag{2.3}$$

$$\sum_{k \in K} y_i^{kpr} \leq 2 \quad \forall i \in V', \forall p \in P, \forall r \in R \tag{2.4}$$

$$\sum_{i \in V'} x_{0i}^{kpr} \leq 1 \quad \forall k \in K, \forall p \in P, \forall r \in R \tag{2.5}$$

$$\sum_{i \in V'} x_{0i}^{kpr} - \sum_{i \in V'} x_{i(n+1)}^{kpr} = 0 \quad \forall k \in K, \forall p \in P, \forall r \in R \tag{2.6}$$

$$\sum_{i \in V'} x_{ih}^{kpr} - \sum_{j \in V'} x_{hj}^{kpr} = 0 \quad \forall h \in V', \forall k \in K, \forall p \in P, \forall r \in R \tag{2.7}$$

$$t_i^{kpr} + s_i + t_{ij} - M_1(1 - x_{ij}^{kpr}) \leq t_j^{kpr} \quad \forall i, j \in V, \forall k \in K, \forall p \in P, \forall r \in R \tag{2.8}$$

$$a_i^r y_i^{kpr} \leq t_i^{kpr} \leq b_i^r y_i^{kpr} \quad \forall i \in V, \forall k \in K, \forall p \in P, \forall r \in R \quad (2.9)$$

$$t_0^{kp1} \geq \sigma^{kp1} \quad \forall k \in K, \forall p \in P \quad (2.10)$$

$$t_0^{kpr} \geq t_{n+1}^{kp(r-1)} + \sigma^{kpr} \quad \forall k \in K, \forall p \in P, r \in R \setminus \{1\} \quad (2.11)$$

$$t_0^{kpr} \geq \sum_{l=1}^{r-1} T_l \quad \forall k \in K, \forall p \in P, \forall r \in R \quad (2.12)$$

$$t_i^{kpr} \leq \sum_{l=1}^r T_l \quad \forall i \in V, \forall k \in K, \forall p \in P, \forall r \in R \quad (2.13)$$

$$\sigma^{kpr} = \beta \sum_{i \in V} s_i y_i^{kpr} \quad \forall k \in K, \forall p \in P, \forall r \in R \quad (2.14)$$

$$\bar{t}_i^r \geq t_i^{kpr} - M_2(1 - y_i^{kpr}) \quad \forall i \in V', \forall k \in K, \forall p \in P, \forall r \in R \quad (2.15)$$

$$\underline{t}_i^r \leq t_i^{kpr} + M_2(1 - y_i^{kpr}) \quad \forall i \in V', \forall k \in K, \forall p \in P, \forall r \in R \quad (2.16)$$

We include two Big-M parameters (M_1 and M_2) related to the time management of the problem set, equal to the period length. Constraints (3.3)-(3.4) guarantee that every customer can be visited by at most two vehicles in each subperiod and each vehicle can visit each customer in every period. Constraints (3.5)-(3.7) are flow conservation constraints that describe each individual route. Constraints (3.8)-(2.14) ensure the feasibility of the time schedule. Constraints (3.9) force t_i^{kpr} to be 0 if customer i is not served by vehicle k in period p and subperiod r . Constraints (3.11) ensure that every vehicle must return to the depot before or at the moment the current subperiod ends. Constraints (2.14) define the loading time for each route as the sum of the service times over all customers in the routes multiplied by a constant factor β . Constraints (3.12) and (3.13) define the earliest and latest arrival times of each vehicle k to each customer i in subperiod r .

2.2.3 Inventory flow constraints:

$$\sum_{i \in V'} q_i^{kpr} \leq Q \quad \forall k \in K, \forall p \in P, \forall r \in R \quad (2.17)$$

$$q_i^{kpr} \leq y_i^{kpr} M_3 \quad \forall i \in V', \forall k \in K, \forall p \in P, \forall r \in R \quad (2.18)$$

$$I_i^{p,1} - I_i^{(p-1)|R|} = \sum_{k \in K} q_i^{kp,1} - d_i^{p,1} + o_i^{p,1} \quad \forall i \in V', \forall p \in P \quad (2.19)$$

$$I_i^{pr} - I_i^{p(r-1)} = \sum_{k \in K} q_i^{kpr} - d_i^{pr} + o_i^{pr} \quad \forall i \in V', \forall p \in P, r \in R \setminus \{1\} \quad (2.20)$$

$$I_i^{pr} \geq 0 \quad \forall i \in V, \forall p \in P, \forall r \in R \quad (2.21)$$

$$I_i^{pr} \leq C_i \quad \forall i \in V', \forall p \in P, \forall r \in R \quad (2.22)$$

$$I_i^{0|R|} - I_i^{|P||R|} \geq \Delta_i \quad \forall i \in V' \quad (2.23)$$

$$\Delta_i \geq 0 \quad \forall i \in V' \quad (2.24)$$

In this second group of constraints, we include commodity and inventory management constraints. Constraints (3.15) establish a maximum vehicle capacity. Constraints (3.16) indicate that a customer cannot receive any amount by a vehicle if it is not visited by that vehicle in a period and subperiod. The big-M parameter in these constraints is equal to the vehicle capacity. Constraints (3.17) and (2.20) define the inventory levels of every customer at the end of each subperiod, calculated as the sum of the previous inventory level plus the amount of commodity received in the current subperiod and the amount that the customer is not able to serve due to stockouts. This value is further decreased by the commodity consumption of the customer in the current subperiod. Constraints (3.18) and (3.19) forbid inventory levels to be negative or exceed the inventory capacity of the customer. With Constraints (3.20) and (3.21), we calculate the difference between the initial and the ending inventory of each customer when the difference is positive.

2.2.4 Previous deliveries constraints:

$$t_i^{kpr} < t_i^{k'pr} \Rightarrow A_i^{kk'pr} = q_i^{kpr} \quad \forall i \in V', \forall k, k' \in K, \forall p \in P, \forall r \in R \quad (2.25)$$

$$t_i^{kpr} \geq t_i^{k'pr} \Rightarrow A_i^{kk'pr} = 0 \quad \forall i \in V', \forall k, k' \in K, \forall p \in P, \forall r \in R \quad (2.26)$$

We introduce two groups of constraints to deal with the split deliveries characteristic. By using Constraints (2.25) and (2.26), we define the matrix of variables $A_i^{kk'prz}$. If a customer i is visited more than once in a subperiod, the amounts delivered to the customer by the previous vehicles are calculated.

2.2.5 Stockout and overstock constraints:

$$I_i^{(p-1)|R|} - d_i^{p,1} t_i^{kp,1} / T_1 + o_i^{p,1} \geq - \sum_{k' \in K : k' \neq k} A_i^{k'kp,1} \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (2.27)$$

$$I_i^{p(r-1)} - d_i^{pr} (t_i^{kpr} - y_i^{kpr} \sum_{l \in R : l < r} T_l) / T_r + o_i^{pr} \geq - \sum_{k' \in K : k' \neq k} A_i^{k'kpr} \quad \forall i \in V', \forall k \in K, \forall p \in P, r \in R \setminus \{1\} \quad (2.28)$$

$$q_i^{kp,1} + I_i^{(p-1)|R|} - d_i^{p,1} t_i^{kp,1} / T_1 + o_i^{p,1} \leq C_i - \sum_{k' \in K : k' \neq k} A_i^{k'kp,1} \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (2.29)$$

$$q_i^{kpr} + I_i^{p(r-1)} - d_i^{pr} (t_i^{kpr} - y_i^{kpr} \sum_{l \in R : l < r} T_l) / T_r + o_i^{pr} \leq C_i - \sum_{k' \in K : k' \neq k} A_i^{k'kpr} \quad \forall i \in V', \forall k \in K, \forall p \in P, r \in R \setminus \{1\} \quad (2.30)$$

We create some constraints to avoid excessive stock at the customers at any time, considering a continuous commodity consumption. Furthermore, as a result of continuous commodity consumption, we have to calculate possible stockouts that occur at any time within the time periods. Constraints (3.22) and (3.23) ensure that at the arrival time of any vehicle to a customer, the inventory level is non-negative. They also account for whether any other delivery occurs before the arrival time of each vehicle. Whenever a stockout occurs, the amount out of stock is calculated. Constraints (2.29) and (2.30) act in the opposite way, to avoid excessive inventory levels while also considering possible previous deliveries.

2.2.6 Additional OU constraints:

The model we propose satisfies one of the most commonly used policies related to the inventory management and delivery amounts in IRP literature, the MLP. It is outpacing the OU in popularity, because the MLP allows companies to deliver any amount of commodity to each customer, as long as that amount, plus the current inventory level, does not exceed the inventory capacity. Alternatively, the OU policy ensures that every time a vehicle arrives to a customer, it delivers the exact amount of commodity necessary to fill inventory completely. Adapted to our problem, the OU policy ensures that, when the last vehicle arrives to a customer in a subperiod, the inventory must be full. If a customer is visited by multiple vehicles in a subperiod, the first vehicle does not necessarily fill the inventory entirely.

$$(1 - y_i^{kpr})M_4 + \sum_{l \in K} q_i^{lp,1} - (C_i - I_i^{(p-1)|R|}) + o_i^{p,1} \geq d_i^{p,1} t_i^{kp,1} / T_1$$

$$\forall i \in V', \forall k \in K, \forall p \in P \quad (2.31)$$

$$(1 - y_i^{kpr})M_4 + \sum_{l \in K} q_i^{lpr} - (C_i - I_i^{p(r-1)}) + o_i^{pr} \geq d_i^{pr} (t_i^{kpr} - y_i^{kpr} \sum_{l \in R, l < r} T_l) / T_r$$

$$\forall i \in V', \forall k \in K, \forall p \in P, r \in R \setminus \{1\} \quad (2.32)$$

We propose Constraints (2.31) and (2.32) to deal with the OU policy in our problem. By adding these constraints to the model, we ensure that, when the last vehicle has arrived, the total amount of commodity delivered satisfies the OU policy.

2.2.7 Variable domain:

$$x_{ij}^{kpr} \in \{0, 1\} \quad \forall i, j \in V, \forall k \in K, \forall p \in P, \forall r \in R \quad (2.33)$$

$$y_i^{kpr} \in \{0, 1\} \quad \forall i \in V', \forall k \in K, \forall p \in P, \forall r \in R \quad (2.34)$$

$$t_i^{kpr} \geq 0 \quad \forall i \in V, \forall k \in K, \forall p \in P, \forall r \in R \quad (2.35)$$

$$q_i^{kpr} \geq 0 \quad \forall i \in V', \forall k \in K, \forall p \in P, \forall r \in R \quad (2.36)$$

$$A_i^{kk'pr} \geq 0 \quad \forall i \in V', \forall k, k' \in K, \forall p \in P, \forall r \in R \quad (2.37)$$

In Constraints (3.24) to (2.37), we define the domain of each variable of the problem.

2.3 Solution Approach

To solve the CIRPTWSD, we propose a matheuristic solution approach. Our method is based on the idea of the ALNS [68], which provides an efficient algorithm for IRPs, particularly for IRPs for which consistency is an important characteristic([33], [59]). Given an initial solution, the ALNS applies several destroy and repair operators to iteratively remove and rebuild parts of the solution. We present different operators related to one or more characteristics of the problem. Thus, we explore different solution neighborhoods to increase the quality of the solutions. In contrast with a generic ALNS algorithm though, we propose a hybrid version with a mathematical subproblem, based on the mathematical formulation of the CIRPTWSD. Figure 2.2 shows an overview of the proposed method, including the two parts of the algorithm, the constructive heuristic and the ALNS, their main components, and where the exact subproblem is integrated in both.

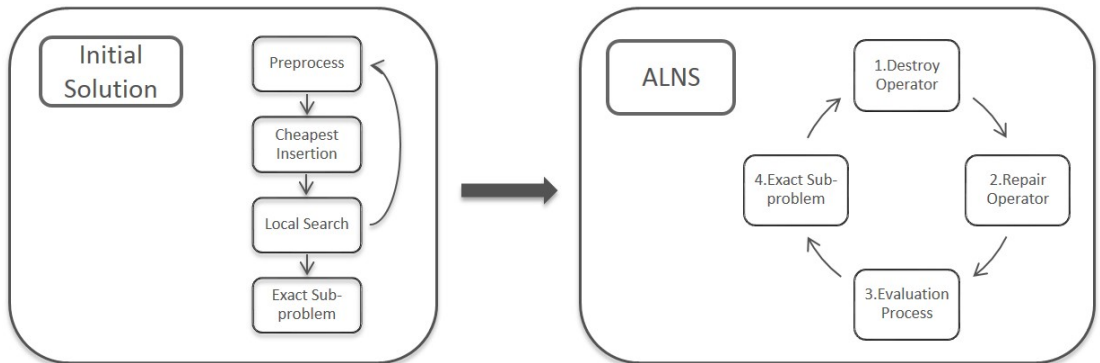


Figure 2.2: Components of the method

2.3.1 Initial solution

The initial solution is generated using an adaptation of the cheapest insertion heuristic [79] for a variant of the VRP. We combine this constructive heuristic with a method to decide which customers must be visited in each period and subperiod. We then apply two local search operators to improve the quality of the initial solution and solve a mathematical optimality subproblem to determine the optimal timing and delivery quantities and to introduce waiting times when necessary. A pseudocode of the complete heuristic is presented in Algorithm 1.

Algorithm 1 Initial Solution

```

1: Require Input Data
2: for  $p = 1 : |P|$  do
3:   for  $r = 1 : |R|$  do
4:     for  $i = 1 : |N|$  do
5:       if (Inventory of customer  $i \leq$  demand until next time window) then
6:         Add customer  $i$  to the Priority List
7:         Calculate upper and lower bounds for the delivery
8:       Cheapest insertion with maximum quantity for all customers in Priority List
9:       if (Priority list not empty) then
10:        Cheapest insertion with minimum quantity for the rest of customers in
        Priority List
11:      if (Priority list not empty) then
12:        Balance quantities and split deliveries
13:      Local search 1: delete single customer routes
14:      Local search 2: 2-opt
15: Solve exact subproblem return Initial Solution

```

Preprocessing

Before we construct the routes for each period and subperiod, we decide which customers must receive a delivery in that subperiod. We create a priority list containing all customers whose inventory level at the beginning of the subperiod is not sufficient to survive without a stockout before the next possible delivery time window. We then calculate an upper and lower bound for the delivery amount for each customer. The upper bound is the difference between the current inventory level and the inventory capacity. The lower bound is the amount necessary to avoid a stockout before the end of the next delivery time window. In Figure 2.3, we present a small example. The customer has enough inventory, so it will not suffer a stockout during Period 1.

However, it runs out of stock before the next time window ends (in this case, in Period 2). Thus, the customer must receive a delivery in the current period. Then, after deciding if the customer must be visited or not, we calculate the two bounds. We repeat this process before applying the insertion heuristic for every period and subperiod.

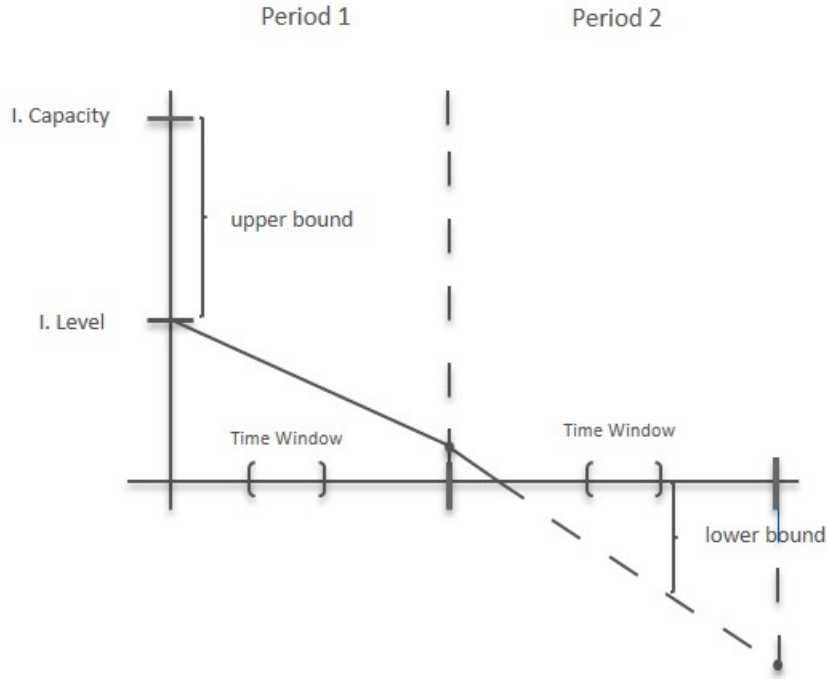


Figure 2.3: Example of customer inserted in the priority list of Period 1 with the corresponding upper and lower bound amounts calculated.

Cheapest insertion

When we obtain the list of customers to be served in the current period, we insert them as follows: First, we initialize a route by inserting the customer farthest away from the depot that is included in the list of customers that we just created. Second, we insert customers according to a cheapest insertion algorithm, using the largest possible quantity, which is the upper bound, increased by the quantity consumed until the arrival of the delivery. Third, if we cannot insert any more customers in the route, due to capacity constraints of trucks or time-windows of the remaining customers, we initialize a new route in the same way. We iterate until no more insertions are possible.

If all customers from the list have been inserted, the method stops. If there are still customers that need a delivery but cannot be inserted with the maximum quantity, we try to insert them with the minimum amount. We apply the cheapest insertion algorithm, as described, using the lower bound quantity increased by the amount that will be consumed until the delivery arrives.

If this second insertion step is not enough to insert all customers, a third procedure applies to balance the amount delivered to all customers and, if necessary, split deliveries to remaining customers into two different routes, assuming there is enough vehicle capacity available. If not, this customer receives a delivery in a later period. We balance the amounts delivered to customers by reducing the delivery amounts of customers inserted with the upper bound amount to reduce the total truck load and, therefore, allow for the insertion of more deliveries into the existing routes. If a customer's demand is greater than the vehicle capacity, or we have not been able to insert it, we try to split the amounts to be delivered in two vehicles.

Local search

To improve the quality of the initial solution, we apply two different local search operators that have been widely studied. The first operator aims to destroy single-customer routes and inserts these customers into other routes. The single customer route operator identifies the route and the customer to be re-allocated, then calculates the total amount of capacity left in all the remaining routes of the same period. Then, by decreasing the amounts delivered to the other customers of the route, it tries to insert the customer into an existing route. We only insert the customer in another route if the distance of the detour caused by this new insertion is lower than the total distance of the single customer route we are trying to avoid. The second operator is the well known 2-opt algorithm. In Appendix A, we list the results of the proposed algorithm using different chain lengths of the 2-opt algorithm. In the rest of the experiments, we apply the 2-opt algorithm with a maximum length of four customers. To ensure feasibility, we do not allow any changes that could violate the time-windows of the customers. Thus we search for better sequences of customers, that satisfy the other characteristics of the problem.

Exact subproblem

We repeat the described construction heuristic for every period and subperiod to obtain a feasible initial solution. However, this first solution may not be optimal for this route sequence. To obtain the optimal delivery quantities for customers, introduce waiting times to improve

possible inconsistent deliveries, and increase the final inventory levels that will create a pseudo-rolling horizon, we solve a reduced problem based on the problem formulation. The reduced problem contains all constraints of the original problem formulation, using MLP or OU when necessary, but the customer visits y_i^{kpr} and route sequences x_{ij}^{kpr} are no longer a variables and instead function as parameters.

2.3.2 Adaptive large neighborhood search

We use ALNS to improve the initial solution provided by the construction heuristic. The method integrates several operators to iteratively destroy and repair the current solution. Moreover, for some good solutions that we find during the ALNS, we solve the exact subproblem to obtain the optimal delivery schedule using the provided route sequences, as explained in the previous section. A pseudocode of the method is in Algorithm 1. To avoid local optima, we introduce the option to accept deteriorating solutions after long periods without finding an improving solution.

In each iteration of the ALNS, we generate a new solution by applying one destroy and one repair operator. The selection of these operators relies on a roulette-wheel selection operator based on the past performance of the operators. The selection and prioritization of destroy and repair operators are pairwise, rather than separate. Weights ρ_{0dr} related to each pair of destroy and repair operators are initially set to the same value of 20; when we find a new better solution, we update the weights of the operators, increasing them by $\gamma = 5$, then we resume the procedure of updating the operator weights as follows:

$$\rho_{idr} = \begin{cases} \rho_{(i-1)dr} + \gamma & \text{if new better solution found.} \\ \rho_{(i-1)dr} & \text{if no better solution found.} \end{cases} \quad (2.38)$$

In this case, i is the current iteration of the ALNS algorithm. We propose eight destroy operators and seven repair operators, as described next.

- Remove worst insertions: Given a solution s , we define the cost of the detour caused by the insertion of customer i as $cost = c_{ki} + c_{ij} - c_{kj}$, where k and j are the preceding and succeeding customers in the route respectively. Then, the operator calculates the detour

Algorithm 2 ALNS

Require: Initial solution and cost. $s_{initial}$ & $c_{initial}$

```

1:  $s_{best}, s_{incumbent} = s_{initial}$ 
2:  $c_{best}, c_{incumbent} = c_{initial}$ 
3:  $t, t_{last} = 0$  Start time and time of last improvement
4: while time  $t < \text{MAXTIME}$  do
5:    $s_{current} = s_{incumbent}$ 
6:   Select a pair of destroy and repair operators  $d$  and  $r$ 
7:   Apply operator  $d$  to  $s_{current}$ 
8:   Actualize inventory flows and stockouts
9:   Apply operator  $r$  to  $s_{current}$ 
10:  2-opt local search
11:  Evaluate solution and get cost  $c_{current}$ 
12:  if ( $c_{current} < 1.5 \cdot c_{best}$ ) then
13:    Solve exact subproblem and get cost  $c_{exact}$ 
14:    if ( $c_{exact} < c_{best}$ ) then
15:       $s_{best}, s_{incumbent} = s_{current}$ 
16:       $c_{best}, c_{incumbent} = c_{exact}$ 
17:      Update weight  $\rho_{idr}$  of operators  $d$  and  $r$ 
18:    else
19:      if ( $c_{exact} < c_{incumbent}$ ) then
20:         $s_{incumbent} = s_{current}$ 
21:         $c_{incumbent} = c_{exact}$ 
22:      else
23:        if ( $t - t_{last} > \text{timelimit}$ ) then
24:           $s_{incumbent} = s_{current}$ 
25:           $c_{incumbent} = c_{exact}$ 
return  $s_{best}$  &  $c_{best}$ 

```

cost caused by every insertion, selects the p worst insertions with respect to these costs, and removes them from the solution.

- Remove random insertions: This operator selects p different insertions from random periods, subperiods, and route, and removes them from the solution.
- Remove vehicle: This operator selects a vehicle at random and removes all deliveries performed by this vehicle in every period and subperiod.
- Remove subperiod: This operator selects a random period and subperiod and removes all routes of that subperiod.
- Remove customer: This operator selects p customers at random and removes all their deliveries.
- Remove customer and closest: This operator selects p customers at random and removes all deliveries to these customers and to their closest customer in terms of distance.
- Remove furthest customer and closest: This operator identifies the p furthest customers from the depot, in terms of distance, and removes all deliveries done to these customers and their closest customers in terms of distance.
- Remove similar inventory customers: This operator selects p customers at random and, for each, selects one other customer with a similar percentage of initial inventory level. Then, it removes all deliveries to all selected customers.

Every time we apply a destroy operator, the inventory levels of the removed customers change, and new stockout situations may occur. Because we consider intra-period stockouts, it is possible that a removed customer faces a stockout before the next delivery takes place but, at the end of the subperiod in which this next delivery happens, the inventory might partially fill again. As we describe in the mathematical formulation, stockout situations can also occur inside the periods and subperiods, which requires recalculating the new inventory flow. In Figure 2.4 we provide an example of how we update the inventories of removed customers. In this example, a customer with an inventory capacity of 30 units, a demand of 15 units per period, and a total period length of six hours has its first out of three deliveries removed. This delivery was performed two hours after Period 2 starts, and the quantity delivered was 30 units. The delivery arrived at the exact moment inventory ran empty. As a result of its removal, the customer does not have enough inventory to satisfy the whole period's demand and faces lost sales of $Demand - InventoryLevel = 15 - 5 = 10$

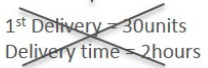
units. However, this delivery also affects the subsequent period, such that in Period 2. In Period 3, the customer does not have any delivery scheduled, and thus loses all demand, leading to the lost sales of Period 3 equal to $Demand - Inventory = 15 - 0 = 15$ units. Finally, in Period 4, the customer has a scheduled delivery two hours after the period starts and therefore loses all demand that occurs before the new delivery takes place. In this case, the new amount of lost sales is $Demand \cdot ArrivalTime / PeriodLength = 15 \cdot 2/6 = 5$ units.

Inventory Capacity = 30units


Constant demand = 15units

Period Length = 6hours

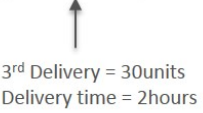
Day	0	1	2	3	4	5	6
Inventory levels before removal	20	5	20	5	20	5	20
Inventory levels after removal	20	5	0	0	20	5	20
New lost sales	0	0	10	15	5	0	0



~~1st Delivery = 30units~~
~~Delivery time = 2hours~~



2nd Delivery = 30units
Delivery time = 2hours



3rd Delivery = 30units
Delivery time = 2hours

Figure 2.4: Update inventory flow after destroy operator

After we update the inventory levels and the amount of lost sales for the removed customers, we create a list of customers that must be reinserted into the solution, because that need deliveries to avoid stockout situations caused by their removal. We then apply the corresponding repair operator to create new deliveries to the customers in the list.

- **Best insertion in destroy order:** This operator inserts customers in the same order in which they were removed from the previous solution. To reinsert these customers into the existing routes, we create a list of possible insertions for the period a stockout occurs and for all preceding periods. The possible insertions are sorted by the total distance of the detour caused by them. We then randomly select one of the three best possible insertion positions and perform the insertion. If, after evaluating the new inventory flow of the customer, there still are stockout situations in the later periods, we repeat the process between the new stockout period and the last insertion period. Figure 2.5 shows how this repair operator works with the same customer that we used in Figure 2.4. After updating the inventory

and the lost sales for the customer, we find a first stockout situation in Period 2. To avoid it, we evaluate all possible insertions in Period 2 and all earlier periods. As a result, we select a possible insertion of the customer with a delivery of 20 units four hours after Period 1 starts. We then calculate the resulting inventory flow and stockouts after this new delivery. Finally, we search for other possible stockouts, and we repeat the process, calculating possible insertions between the period of the new stockout(i. e., Period 3 in Figure 2.5) and the period adjacent to the previous delivery (i. e., Period 2).

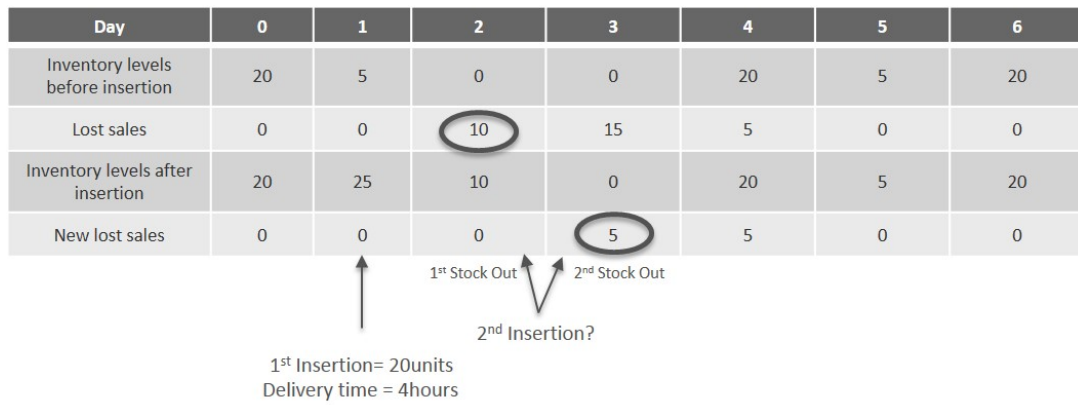


Figure 2.5: Update inventory flow after repair operator

- Best insertion in random order: This operator works in the same way as the previous operator, except that we randomly select the next customer to be reinserted into the solution.
- Random insertion in destroy order: This operator creates a list of possible insertions just like the *best insertion in destroy order* operator but then randomly selects one of all possible insertions from the entire list.
- Random insertion in random order: This operator creates a list of possible insertions just as the previous operator and then randomly selects one of them from the entire list. We repeat the process until a feasible solution is reached.
- Consistent insertion in destroy order: This operator inserts customers such that the difference between the earliest and the latest arrival times to this customer is minimized.

- Consistent insertion in random order: This operator inserts customers in the same way as the previous operator, but in this case, the customer to be inserted is selected randomly and not in the destroy order.
- Distance-related insertion: This operator can be considered as an extension of the *best insertion in destroy order* operator. It selects customers in the same order as they were removed from the solution, and we calculate the best possible insertions in terms of the detour they cause, as we explained for Figure 2.5. After inserting this customer, we search, in the list of customers to be reinserted, for its closest customer in terms of distance. We then try to insert this new customer next to the previous customer if possible. If by inserting this customer we can reduce its lost sales, we remove not only the first customer from the list but both customers.

When we obtain a new solution after applying both, the destroy and the repair operators, we evaluate the total costs of the solution. If the new solution is not more than 50% worse than the best known solution, we solve a reduced variant of the mathematical problem formulation, as explained in Section 2.3.1. We consider a maximum 50% decrease in the quality of the solution to be able to solve the reduced exact problem for a considerable amount of new solutions, without expending unnecessary time solving solutions with high costs. For a computational justification, in Appendix A we also provide alternative results obtained by solving instances when we consider different acceptance criteria parameters in the proposed ALNS. Finally, we have a new solution with optimal costs for these route sequences. If the new solution improves the incumbent solution, we update the incumbent solution and the weights for the destroy and repair operators used in this iteration. We do the same with the best known solution if the new solution improves on it. However, if the new solution does not improve the incumbent or the best known solution, we proceed in two different ways. If the total time since the last improvement is lower than a *timelimit*, we use the previous incumbent solution as the starting solution for the next ALNS iteration. If the total time since the last improvement exceeds that time limit, we accept the new solution as the incumbent solution, and therefore, as the input for the next iteration of the ALNS. We then reset the time of the last improvement found to zero. We accept this new solution to avoid strong local optima that prevent the method from continuing to improve the quality of the solutions. The stopping criterion for the ALNS is an overall limit on runtime.

2.4 Computational Results

To the best of our knowledge, this is the first work on the CIRPTWSD that takes into account time-windows and stockout situations within the time periods. To analyze the effectiveness of the proposed approach we first perform computational experiments using the benchmark set of instances first introduced in [12]. In a second step, to gain insights into the problem properties, we perform computational experiments on two sets of instances. The first set is adapted from a benchmark set, and the second is based on a real-life application of the problem. Other sets of instances have been proposed to solve other variants of IRPs, but the time window characteristic of our problem makes it rather difficult to adapt these instances to our problem setting. We evaluate the impact of different replenishment policies and the time-windows on our solutions and compare the results to others obtained using an exact solver on our set of instances.

The algorithm we propose was coded in C++, and all computational experiments were performed on a Linux system equipped with two Intel Xeon E5-2650(2.6 GHz) processors and 64 GB RAM. IBM CPLEX 12.6.3 was used as a MIP solver with a maximum computation solving time of ten hours.

2.4.1 Test Instances

Because previous efforts to demonstrate the computational effectiveness of the proposed methods do not include inventory information related to capacity and initial inventory at the customers, [63], we cannot compare this problem with any previous studies and instead create a new set of instances¹ adapted from a benchmark set[36]. This benchmark set of instances pertain to periodic vehicle routing problem with time-windows (PVRPTW), for which the algorithm chooses between different visit day combinations that are already given. In our case, the visit sequences must be chosen by the algorithm based on current inventory levels.

These instances consider a planning horizon of four days and the most efficient visit pattern must be chosen between different visit combinations. We create instances of different sizes (5, 10, 15, 20, and 48 customers) and a planning horizon of four periods with one or two subperiods. We use the customer information related to coordinates and time-windows from the instances of size 48 in the benchmark set. However, we increase the commodity consumptions of the benchmark set of instances to balance the inventory and stockout costs with the rest of the costs in the problem.

¹The set instances are available at <https://bda.univie.ac.at/research/data-and-instances/vehicle-routing-problems/> when published

In the proposed set of instances, we consider the commodity consumptions to be ten times the commodity consumptions of the benchmark set. For smaller conversion factors, some previous experiments show that the optimal decision would be not to perform any delivery, because the routing and consistency cost would be higher than stockout costs. Yet selecting a bigger factor would imply high stockout costs compared with the other costs considered in the problem. Finally, we created inventory information for our instances. We consider different scenarios to create inventory information and force every customer to be delivered at least once or twice, depending on its demand periodicity. Every customer with daily commodity consumption has an inventory capacity equal to two times the average demand. For customers that present consumption every second period, we set inventory capacity to 1.5 times the average consumption. For customers that present consumption in one period for instances of four periods and one subperiod, and two periods for instances with four periods and two subperiods, the inventory capacity is set to approximately 1.1 times the average consumption. Then, based on the inventory capacity of each customer, we generate initial inventory levels by randomly selecting an initial percentage of initial inventory relative to the inventory capacity (25%, 50%, 75%, or 100%). Vehicle capacity is set to 2,000 units, and we fixed the fleet size to be able to deliver twice the average commodity consumption per period on a single period. Furthermore, we set the penalty cost $L = 3$ for each unit of demand that customers are not able to satisfy or unit of difference between the initial and ending inventory level and the consistency cost weight $\alpha = 1$. We select $\alpha = 1$ because the benchmark instances [36] consider time-windows that depend on the routing distances, so both routing and consistency costs are measured using the same unit. We also set the penalty cost $L = 3$ to balance the possible stockout and inventory costs, relative to the rest of the costs considered in the problem.

The characteristics of the proposed instances are summarized in Table 3.2. For each instance size, we create four groups of ten instances. For fair comparisons, each group includes different characteristics of the problem to evaluate the impact of these aspects on the costs. Instances in groups of type a include regular commodity consumption, so most of the customers engage in commodity consumption in every period. Instances of type b include the same customers but without any time-windows. Instances of type c include customers with less regular consumption, such as customers with consumption once or twice per time horizon. The last group, instances of type d, includes information for four periods and two subperiods, unlike the other types for

which we include information of four periods and one subperiod. In Table 3.2, we also include the number of vehicles available at the depot at the beginning of the planning horizon.

Group	Size	Periods	SubPeriods	Vehicles	Consumption	Time-Windows
1a	5	4	1	1	High	TW
1b	5	4	1	1	High	No TW
1c	5	4	1	1	Low	TW
1d	5	4	2	1	High	TW
2a	10	4	1	2	High	TW
2b	10	4	1	2	High	No TW
2c	10	4	1	2	Low	TW
2d	10	4	2	2	High	TW
3a	15	4	1	2	High	TW
3b	15	4	1	2	High	No TW
3c	15	4	1	2	Low	TW
3d	15	4	2	2	High	TW
4a	20	4	1	3	High	TW
4b	20	4	1	2	High	No TW
4c	20	4	1	3	Low	TW
4d	20	4	2	2	High	TW
5a	48	4	1	6	High	TW
5b	48	4	1	3	High	No TW
5c	48	4	1	6	Low	TW
5d	48	4	2	3	High	TW

Table 2.2: Characteristics of instances for the CIRPTWSD

In addition to the described set of instances, we perform computational experiments using another two set of instances. In order to evaluate the performance of the algorithm and compare the effectiveness with respect of previous works present in the literature, we solve a simpler version of the problem using the benchmark set of instances introduced in [12]. This set of instances presents information about demand, inventory capacity and level and location of different sets of retailers whereas no time-related information is not included. Furthermore, different planning horizons are considered in the instances (three and six periods) and, unlike the problem we face in this work, two variants with high and low holding cost are considered. For further information about the benchmark set of instances we refer to the original paper.

In the last step, we create a set of instances based on a real-world application of the problem. We have information about average beer consumption per sit place, opening times and locations,

and real-world distances and times for 400 bars, restaurants, and beer stands in the city of Vienna. We generated five groups of real world-instances (a, b, c, d and e) that include information about 92 customers located in its inner city. For each group, we generate demand using a normal distribution that reflects daily average beer consumption. Each group contains five instances with different initial inventory levels generated, in accordance with the adapted benchmark set of instances. We provide further information about these instances in Table 2.3. They feature a planning horizon of a week (seven days) with one or two subperiods and a fleet size of five vehicles. Furthermore, customers prefer different time-windows and have varied opening days.

Group	Size	Periods	SubPeriods	Vehicles	Consumption	Time-Windows
92a	92	7	1	5	Normal	TW
92b	92	7	1	5	Normal	TW
92c	92	7	1	5	Normal	TW
92d	92	7	1	5	Normal	TW
92e	92	7	1	5	Normal	TW
92a2	92	7	2	5	Normal	TW
92b2	92	7	2	5	Normal	TW
92c2	92	7	2	5	Normal	TW
92d2	92	7	2	5	Normal	TW
92e2	92	7	2	5	Normal	TW

Table 2.3: Characteristics of real-world instances for the CIRPTWSD

2.4.2 Comparison to benchmark set of instances

In order to evaluate the effectiveness of the proposed method, and compare the obtained results with a benchmark set of instances, we have adapted the method to solve a simpler version of the problem. We use the set of instances introduced in [12]. In this work, authors present a different variant of the IRP where time-related information and the possibility of stock situations is not considered. Removing the calculations related to these characteristics from the proposed method, would drastically transform the algorithm. However, small adaptations can be performed to create a fair comparison. The proposed algorithm is adapted so that the continuous consumption of commodity is no longer considered. That is, customers experience demands at the end of the planning horizon and, therefore, it is no longer possible to incur in stock-out situations within the time periods, whereas the stock-out possibility remains at the end of the subperiods. Secondly, the mathematical subproblem solved within the algorithm must be also

adapted. In this case, we solve a mathematical formulation related to the IRP variant presented in [12]. Using this mathematical formulation, we can also calculate the different cost considered in this version of the problem. Inventory holding cost are calculated using this mathematical subproblem, and those solutions which present an stock-out situation can be discarded.

In Table 3.3, we show the gap obtained when solving the set of instances with respect to the known optimal. We report results for those instance sizes where the algorithm was able to find feasible solution within a total computational time of 30 minutes. Solutions that present stock-out situations are treated as non feasible solution, as the problem presented in [12] does not consider this situation. For most of the small instances (5, 10 customers) with high and low holding cost and three and six subperiods, the proposed algorithm is able to find optimal or nearly optimal solutions for the problem, even if the original proposed method does not aim to reduce holding cost or can consider stock-outs. For medium size instances (15, 20 customers), the solutions obtained by the method present relatively small gaps with respect to the optimal. Finally, we can see how, for large instances, the algorithm has difficulties to obtain solutions without stock-outs, as this is not the main feature considered in the method.

2.4.3 Comparison to the exact solver

In a second step, we solve the adapted set of instances and we compare the results obtained by applying our solution approach to instances of types a, c and d which include all the characteristics of the CIRPTWSD. We solve each instance of the different groups ten times with a maximum computation time of 30 minutes, where the *timelimit* without finding a new better solution is set to five minutes. In Table 2.5, we compare the results with those obtained by solving the instances using CPLEX with a time limit of ten hours.

In Table 2.5, each row reports the average objective cost of the ten instances included in every group with both CPLEX and our solution approach. Groups of instances with an asterisk (*) indicate that CPLEX was not able to prove that the best solution found was optimal within the time limit. Furthermore, groups of instances with two asterisk (**) meet that condition but also indicate that CPLEX was not able to find any feasible solution during the processing time for some instances. The column “CPLEX Gap” shows the average gap reported by CPLEX for those instances where the solver does not find the optimal solution, and the column “No Sol.” reports the number of instances of each group where CPLEX is not able to find any feasible solutions. For groups 3a, 3c, 4a, and 4c, CPLEX finds optimal solutions for some instances, whereas for

Instance	3 Periods				6 periods			
	High Holding cost		Low Holding cost		High holding cost		Low Holding cost	
	Optimal	Gap	Optimal	Gap	Optimal	Gap	Optimal	Gap
abs1n5.dat	2149.80	0%	1281.68	0%	3335.24	1%	5942.82	1%
abs2n5.dat	1959.05	0%	1176.63	0%	2722.33	0%	5045.91	0%
abs3n5.dat	3265.44	0%	2020.65	0%	4776	0%	6956.28	0%
abs4n5.dat	2034.44	0%	1449.43	0%	3246.66	2%	5163.42	1%
abs5n5.dat	2362.16	0%	1165.40	0%	2419.67	0%	4581.66	0%
abs1n10.dat	4970.62	0%	2167.37	0%	4499.25	1%	8870.15	1%
abs2n10.dat	4803.17	0%	2510.13	0%	5236.98	1%	8569.73	0%
abs3n10.dat	4289.84	0%	2099.68	0%	4652.53	3%	8509.81	0%
abs4n10.dat	4347.06	0%	2188.01	0%	5104.91	4%	8792.29	1%
abs5n10.dat	5041.62	0%	2178.15	0%	4670.76	0%	9620.07	0%
abs1n15.dat	5713.84	2%	2236.53	4%	5462.68	1%	12118.83	2%
abs2n15.dat	5821.04	0%	2506.21	0%	5494.74	4%	11932.10	3%
abs3n15.dat	6711.25	4%	2841.06	9%	6060.38	11%	13554.15	5%
abs4n15.dat	5227.56	0%	2430.07	0%	5504.65	5%	10618.55	3%
abs5n15.dat	5210.85	0%	2453.50	1%	5309.48	4%	10385.548	3%
abs1n20.dat	7353.83	5%	2793.29	12%	6490.18	-	14702.95	-
abs2n20.dat	7385.03	2%	2799.90	4%	6082.54	3%	14646.96	1%
abs3n20.dat	7903.97	-	3101.60	-	6950.20	-	14532.91	-
abs4n20.dat	7050.91	-	3239.31	-	7432.78	-	14539.72	-
abs5n20.dat	8405.83	4%	3330.99	9%	7210.73	-	15896.71	-
abs1n25.dat	8657.70	11%	3309.64	-	7095.86	-	15581.47	-
abs2n25.dat	9266.87	-	3495.97	-	7484.84	-	16823.16	-
abs3n25.dat	9843.60	-	3481.45	-	7728.76	-	18098.02	-
abs4n25.dat	8677.86	-	3272.74	-	7509.02	-	16303.69	-
abs5n25.dat	10857.68	0%	3695.94	1%	7452.28	-	19047.70	-

Table 2.4: Comparison to benchmark set of instances. Gaps with respect to the known optimal solutions when the algorithm finds solutions without stock-out situations.

groups 5a and 5c, CPLEX is not able to find any optimal solution. The column labeled “Opt” reports how many of the ten instances in each group our solution approach was able to solve to optimality. The last column (“Imp”) reports for how many instances, our solution approach was able to provide better results than the ones found by CPLEX. Our algorithm thus obtains good results relative to CPLEX.

For small instances with five and ten customers, we find optimal solutions for most cases. For example, for group 1a, the algorithm finds optimal solutions for all instances, with an average gap of 0.31%. For instances with 15 and 20 customers, our algorithm still finds good solutions, but we find bigger gaps compared with the solutions obtained by CPLEX when solving instances that contain 20 customers. The largest gap corresponds to the group of instances 3d, where CPLEX finds solutions that are 13.99% better on average. Furthermore, the algorithm can get stuck in local optima, such as in instances 3d and 4d, where, the maximum gaps compared with the best solution found by CPLEX reach 32.18%, though the average gap for these groups of instances is smaller than 14%. Finally, CPLEX is not able to obtain efficient solutions for bigger instances, and the proposed algorithm outperforms CPLEX when solving considerably big instances. For example, in groups 5a and 5d, the proposed algorithm finds better solutions for every instance, improving 60.88% and 72.08% on average, and for the group of instances 5c, it finds a better solution for nine out of ten instances.

Our algorithm performs better especially when customers exhibit more frequent commodity consumption, that is, for instances of group a. On the contrary, it becomes more difficult to integrate more customers with infrequent commodity consumption. However, for the biggest size of every group, the algorithm obtains better solutions than CPLEX with significantly shorter running times. Instances of type d, which present the results for a planning horizon of four periods and two subperiods, reveal similarities with instances with one subperiod, in that CPLEX is only able to provide optimal results for five customers, and some instances of 10, 15, or 20 customers, but it cannot find good solutions for bigger instances. Overall, the proposed method outperforms the alternative option for these large instances.

2.4.4 Time-windows

A key characteristic that can drastically affect the total cost is the existence of time-windows. Therefore, we examine the impact of time-windows on the different cost factors. This impact

Instance set	CPLEX	CPLEX Gap	Matheuristic	Max Gap	Avg Gap	Min Gap	No Sol.	Opt	Imp
1a	1045.05	-%	1048.51	3.06%	0.31%	0.00%	0	10	0
2a	2108.96	-%	2154.38	11.93%	2.14%	0.00%	0	7	0
3a*	2820.83	4.87%	2914.63	14.28%	3.39%	-0.51%	0	3	1
4a*	3348.52	37.54%	3553.61	22.09%	6.42%	-0.56%	0	1	1
5a*	24693.08	89.82%	7546.64	-49.36%	-60.88%	-82.78%	0	0	10
1c	1023.23	-%	1028.96	3.58%	0.55%	0.00%	0	8	0
2c	1371.44	-%	1381.52	6.27%	0.71%	0.00%	0	8	0
3c*	1774.83	1.19%	1813.37	12.11%	2.20%	-0.80%	0	2	1
4c**	2356.80	9.41%	2624.99	23.54%	11.39%	3.19%	2	0	0
5c**	5299.79	59.77%	5153.55	11.71%	-2.59%	-20.26%	1	0	9
1d	1709.77	-%	1747.43	15.86%	2.26%	0.00%	0	5	0
2d*	3162.38	3.67%	3349.76	20.16%	5.89%	-2.52%	0	0	3
3d	3941.18	8.15%	4483.99	32.18%	13.99%	-4.39%	0	0	1
4d*	5247.19	36.35%	5692.06	31.50%	8.74%	-10.18%	0	0	4
5d*	71007.81	94.24%	13730.58	-69.51%	-72.08%	-87.58%	0	0	10

Table 2.5: Results of our method vs CPLEX. * CPLEX is not able to solve all instances optimally. ** CPLEX is not able to find a feasible solution for some instances.

has been widely studied for different variants of the VRP, yet literature related to IRP problems with time-windows within the time periods is scarce.

To evaluate how time-windows affect the costs, we compare the results obtained by solving data sets a and b. These customers have the same coordinates, commodity consumption, and inventory information, but type b excludes time-windows. In Table 2.6, we list the average total cost for each instance size; the “Obj. Cost %” column reports the percentage of the decrease from the total cost, calculated using the formula $(TotalCost_{TW} - TotalCost_{NoTW})/TotalCost_{TW}$. In the last four columns, we present the difference for every cost factor using both policies.

Groups	TW	No TW	Obj.Cost(%)	Rou.Cost	Inv.Cost	S.O.Cost	Cons.Cost
1a vs 1b	1048.51	770.38	26.53%	15%	71%	-46%	0%
2a vs 2b	2154.38	1412.90	34.42%	10%	80%	37%	-894%
3a vs 3b	2914.63	1774.39	39.12%	13%	76%	58%	-80%
4a vs 4b	3553.61	2444.93	31.20%	9%	71%	-67%	-239%
5a vs 5b	7546.64	7307.72	3.32%	16%	15%	-247%	-245%

Table 2.6: Impact of time-windows

Time-windows have a strong impact on the overall costs; the total cost decreases for every instance without time-windows, even if that decrease is related mainly to the routing and inventory costs. The lack of time-windows makes it easier for the algorithm to obtain better routes in

terms of distances and also allows the customer to receive late deliveries to increase the ending inventory level and, therefore, reduce inventory costs. In the third row, for instances of size 15, 3a vs. 3b, the presence of time-windows implies an increase of 39.12% in the overall cost, whereas a smaller improvement takes place for instances of 48 customers, 5a vs. 5b, with an average of 3.32%. This difference on percentages is mainly due to a significant increase on the sales lost for bigger instances when not considering time-windows. In general, stockout costs decrease for small instances but increase for larger instances. The last column shows the increase in consistency costs, which arise because the time-windows force the algorithm to deliver to customers within smaller time ranges. Some of the reported percentages are quite large as in instances 2a vs. 2b, where the percentage is equal to -894% . These big values emerge when the consistency cost for one group of instances is very big (in this case, 2b), but for the other group (2a) is relatively low, or even close to zero.

2.4.5 Real-world based instances

In addition to the computational experiments, we evaluate the effectiveness of the proposed algorithm using instances based on a real-world application. These instances contain information about beer consumption in different establishments in the city of Vienna. However, no previous results exist, and we cannot solve these instances using CPLEX.

There exist commercial solvers that create delivery plans for vehicle routing problem applications. Nevertheless, these solvers do not consider inventory information or multi-period problems. Therefore, companies must create delivery plans using simple heuristics. In this section we evaluate the impact of applying the ALNS and report the costs decrease compared to the solution obtained with the construction heuristic (C.Heuristic).

In Tables 2.7 and 2.8, we report costs for instances that contain information related to 92 customers located in the inner city of Vienna with a planning horizon of seven periods and one and two subperiods, respectively. We provide the percentage of difference in total costs, as well as the total difference for every individual cost factor.

In the results, we can see how the use of the ALNS algorithm with the mathematical subproblem significantly improves the quality of the solutions. Using the proposed matheuristic, we find solutions that reduce the overall costs, generally achieved by inserting additional deliveries to customers, so that it avoids customer lost sales and reduces inventory costs as well. In Table 2.7 for example, regarding 92a, the matheuristic reduces the costs by 46.43% compared with the

Groups	C.Heuristic	Math.	Obj. Cost	Rou. Cost	Inv. Cost	S.O. Cost	Cons. Cost
92a	5609.20	3005.10	46.43%	-79%	59%	49%	18%
92b	6093.48	5031.46	17.43%	-53%	12%	39%	12%
92c	6321.41	4507.27	28.70%	-63%	38%	44%	12%
92d	7082.99	5034.76	28.92%	-60%	33%	47%	10%
92e	6967.24	5417.25	22.25%	-67%	20%	48%	8%

Table 2.7: Real-world instances, seven periods one subperiod.

Groups	C.Heuristic	Math.	Obj. Cost	Rou. Cost	Inv. Cost	S.O. Cost	Cons. Cost
92a2	4435.51	3420.17	22.89%	-98%	33%	32%	-1%
92b2	4302.27	3366.31	21.76%	-80%	37%	-9%	-22%
92c2	4816.63	3393.24	29.55%	-72%	33%	-75%	33%
92d2	2947.45	1934.91	34.35%	-124%	60%	27%	-11%
92e2	6635.6	3558.50	46.37%	-120%	55%	44%	-47%

Table 2.8: Real-world instances, seven periods two subperiods.

initial solution. It may increase the routing cost by 79%, but it reduces inventory, stockout, and consistency costs by 59%, 49%, and 18%, respectively. For instances with a planning horizon of seven days and two subperiods, the matheuristic also drastically improves the results obtained with the insertion heuristic, in that it performs more deliveries and thus diminishes sales lost due to stockouts while also avoiding low inventory levels at the end of the planning horizon.

Further computational experiments were driven in order to evaluate the impact of both, MLP and OU policies. This experiments show that the difference between the MLP and OU results are caused mainly by the randomness of the algorithm. The problem we present does not include inventory holding costs, which are the ones derived from creating the pseudo-rolling horizon, so an OU policy is often considered the best replenishment policy when using the MLP approach. We present these computational experiments in Appendix B.

2.5 Conclusions and further research

We introduce the CIRPTWSD, which is part of the family of IRPs. The characteristics of the problem we present arise from a real-world application namely, route planning and inventory management for beer and other beverages companies. Here, customers have different opening

times and time-windows, so to satisfy overall demand, it may be necessary to split the deliveries across more than one vehicle. Furthermore, consistency in delivery times improves service quality and customer satisfaction, which is a key factor in a competitive market. We propose a mathematical formulation for the CIRPTWSD. We formulate the problem as a MIP with two variants. The first variant includes all characteristics of the problem and uses the maximum level replenishment policy, which gives companies the flexibility to decide on the delivery amounts and times. The second variant includes all constraints of the first variant and some additional constraints, in line with an OU policy that requires the inventory capacity of each customer to be completely filled at the moment each customer is served.

To solve the proposed mathematical formulation, in Section 2.3, we propose a matheuristic solution approach to solve the CIRPTWSD. The solution approach we propose is based on an ALNS that includes several destroy and repair operators applied to improve an initial solution obtained the cheapest insertion algorithm. Furthermore, during the process of obtaining good solutions, we solve an exact subproblem up to optimality using good solutions. We solve this mathematical subproblem to obtain optimal delivery amounts and times using the given routes. Successful neighborhood operators become more important as the algorithm obtains new, better solutions, until a maximum processing time is reached and the algorithm ends.

In some previous literature, authors have presented an IRP with time-windows, but they rely on benchmark instances that do not include inventory information, so we cannot report any comparison with previous results. To test the efficiency of the proposed method, we present a range of computational tests with a benchmark set of instances and with new instance set adapted from a benchmark set. To perform the first computational experiments, and compare our results with previous results in the literature, we have adapted our algorithm to not account for some of the characteristics that the CIRPTWSD presents. This characteristics are related to the continuous consumption of product at the customers, and the possibility of lose sales. The algorithm finds good solutions in short processing times, whereas the exact solvers are not able to solve large instances even with long processing times. We report the results obtained from solving instances of different sizes. In these results, for small instances, the algorithm obtains optimal solutions in most occasions. However, for instances that include 20 customers, the algorithm uncovers bigger gaps compared with the results obtained using an exact solver. Finally, for the bigger instances solved, the algorithm outperforms the results obtained with CPLEX. Other experimental comparisons show the differences between the two described replenishment policies,

as well as the impact of the time-windows. Finally, we perform computational experiments using instances that include real-world information about beer consumption in the city of Vienna.

In this field, research could take different directions. Model-specific research might study other aspects of the problem, such as the possibility of a multi-echelon distribution systems in which a central retailer supplies different intermediate storage centers, and customers are replenished from these storage centers. It also may be interesting to include stochasticity in the problem. Several studies include stochastic travel times or commodity consumption by customers. We aim to develop a two-stage stochastic formulation for the CIRPTWSD. In a first stage, we will obtain an initial delivery plan that includes the set of routes and information about the delivery times and amounts to customers over the planning horizon. In the second stage, different recourse actions can be considered to reduce the overall costs when the stochastic data are being revealed. Another research possibility would be to develop more efficient solution methods. The proposed method provides better solutions for customers with high consumption periodicity, but as the number of non-frequent customers increases, it becomes more difficult to integrate them into the solution efficiently. Novel solution approaches could seek better solutions for instances with more punctual demands, larger instance sizes, and instances with longer planning horizons.

Appendix A

In Tables 2.9 and 2.10 we provide results obtained when solving instances of the group a, described in Table 3.2 using different values for the maximum chain length considered in the 2-Opt algorithm and different values of the acceptance criteria applied in the proposed ALNS. In the tables, we list the average costs obtained when solving the instances with the different parameters and the average gap compared with the results of the selected parameters.

Group	2-OPT 4	2-OPT 3		2-OPT 5	
	Avg	Avg	Gap	Avg	Gap
1a	1048.51	1048.63	0.01%	1048.11	-0.04%
2a	2154.38	2212.55	2.70%	2210.99	2.63%
3a	2914.63	3325.05	14.08%	3334.46	14.40%
4a	3553.61	3850.44	8.35%	3881.62	9.23%
5a	7558.39	8853.96	17.14%	8890.04	17.62%

Table 2.9: 2-OPT chain length tests

Group	ALNS 1.5	ALNS 1.25		ALNS 1.75		ALNS 2	
	Avg	Avg	Gap	Avg	Gap	Avg	Gap
1a	1048.51	1051.84	0.32%	1048.50	0.00%	1048.90	0.04%
2a	2154.38	2251.46	4.51%	2218.89	2.99%	2219.21	3.01%
3a	2914.63	3191.90	9.51%	3373.62	15.75%	3316.87	13.80%
4a	3553.61	3702.64	4.19%	3874.88	9.04%	3910.33	10.04%
5a	7558.39	8594.27	13.71%	8846.71	17.04%	8911.45	17.90%

Table 2.10: ALNS acceptance parameter tests

Appendix B

Table 2.11 presents a similar structure to Table 2.6. We present the percentage decrease in total cost between the results obtained solving the problem with both, MLP and OU policies, along with the total difference of every cost factor.

Instances	MLP	OU	Obj.Cost(%)	Rou.Cost	Inv.Cost	S.O.Cost	Cons.Cost
1a	1048.51	1051.38	-0.09%	0%	0%	-7%	0%
2a	2154.38	2153.19	0.06%	0%	-1%	6%	-511%
3a	2914.63	2925.03	-0.36%	-1%	-3%	22%	-22%
4a	3553.61	3548.27	0.15%	1%	0%	-6%	15%
5a	7546.64	7552.95	-0.08%	1%	0%	-9%	1%

Table 2.11: Results of MLP vs. OU

The difference between the MLP and OU results are caused by the randomness of the algorithm. The problem we present does not include inventory holding costs, which are the ones derived from creating the pseudo-rolling horizon, so an OU policy is often considered the best replenishment policy when using the MLP approach. For example, for instances 1a, 3a, and 5a, MLP policy obtains better solutions on average than the OU policy with a 0.36% average difference in group 3a. Nevertheless, the algorithm finds better solutions for groups ‘2a and 4a when applying OU policy.

A sampling-based matheuristic for the continuous-time stochastic inventory routing problem with time-windows

Published in: *Computers & Operations Research*

A sampling-based matheuristic for the continuous-time stochastic inventory routing problem with time-windows.

E.J. Alarcon Ortega & K.F. Doerner. © 2022, *Elsevier*

<https://www.sciencedirect.com/science/article/pii/S0305054822003598>

Abstract This article considers a continuous-time variant of the stochastic inventory routing problem. In most of the articles present in the literature related to inventory routing, customers reveal their demands at the end of each period, which is when the inventory level is calculated. In the variant of the problem at hand, the demand that each customer experiences on each period, results in a continuous decrease of the inventory for the customer during the period. This characteristic strongly affects the quantities that can be delivered to each customer and, if the deliveries are not sufficient or arrive too late, they can cause some stock-out situations within the periods. In the inventory routing problem, the vendor manages all the replenishment decisions of the vendees and, therefore, creates delivery plans for a planning horizon, aiming to reduce the routing, consistency, inventory and lost-sales costs. We formulate the problem as a two-stage mathematical program. Customers experience a continuous stochastic demand within each period and present information about

inventory level and capacity at each period, as well as the time-windows in which deliveries can take place. To solve this problem, we develop a matheuristic solution approach based on an adaptive large neighborhood search algorithm. In addition, we evaluate the impact of applying recourse actions to deal with expected lost sales during the planning horizon. We compare the performance of the algorithm with adapted variants of the multiple scenario approach and the branch and regret algorithms from the literature. Samples based on the stochastic demands are considered in the algorithms, to find robust solutions that can minimize the objective function. We evaluate the solutions by means of a sample average estimator procedure, and we compare the efficiency of the algorithms as well as the impact of different levels of stochasticity.

3.1 Introduction

In practice, multiple logistic optimization problems cannot be solved efficiently. The logistics related to the inventory and routing are specially challenging. Consider a supply chain wherein the objective is to minimize the long-term cost of a distribution network that consists of a set of stores served from a central facility. The coordination of both inventory and routing decisions is crucial for the correct functioning and allocation of resources for these companies. To deal with this situation, many companies adopt a vendor managed inventory system to centrally create and manage all the replenishment and distribution plans. This system should be largely used because the application of a vendor managed inventory can substantially reduce overall logistics costs [62].

In this paper, we consider such a system, and model it as the IRP for a single product, which was first introduced in the seminal paper [18]. In most real-life situations, demand experienced at stores is uncertain; therefore, we consider a stochastic variant of the problem but with one product type. We present the continuous-time stochastic IRP with time-windows (CTSIRPTW), wherein, a new characteristic of the problem is that the system evolves continuously over time. This means that, along each period, the product is consumed continuously from the inventory of the customers. The continuous-time product consumption is a characteristic recently introduced in the IRP literature (see [60]), however, it is not a new characteristic in articles about inventory

management. The continuous-time inventory depletion resembles the more prevalent real world situations, and, thanks to the growing application of sensors to measure the real-time inventory levels, the problem is becoming of increasing relevance. As a consequence of considering a continuous-time consumption rate, delivery times must be scheduled carefully, and the quantities delivered strongly affect the overall costs of the supply chain. This is an important difference with respect to the majority of the IRP literature, in which, even if the planning horizon is divided into a set of periods, the product consumption at the retailers takes place at the end of the period and it is an instantaneous action. In the paper at hand, we aim to efficiently plan the delivery times and quantities, to avoid the possible stock-out situations that the continuous consumption of the product can originate.

The CTSIRPTW is motivated by the IRP faced by companies in the beverage industry. These companies must create efficient delivery plans for the planning horizon, in order to satisfy the multiple requests from customers. The target customers of such companies experience demands that are sensitive to external factors (weather, holidays, weekends,...), and central planning efforts must be conducted to reduce the overall logistics costs. At the beginning of the planning horizon, companies that belong to this industry create delivery plans that take into account the different nature of the customers (bars, restaurants and retailers), their opening hours, and delivery time-windows. This delivery plan facilitates the communication between the distributor and the retailers, allowing them to know when the deliveries will take place and be prepared for such deliveries. Lastly, customers require consistency in the delivery times, which are requested to be performed at the same time of the day, and allow them to be prepared to receive the goods. At the beginning of the planning horizon, the central company creates delivery plans for the subsequent periods to minimize the routing costs, as well as the inventory and stock-out expected costs at the customers. At the end of the planning horizon, the real demands faced by each customer are disclosed, and the total calculation of the inventory and lost sales costs is performed. In addition to the above mentioned delivery plan, in this article we propose and evaluate the impact of recourse actions that, in case a customer is expected to experience a stock shortage at a certain period, additional visits to the customers are introduced.

The remainder of this paper is organized as follows. In Section 3.2, we perform a review of the related works existing in the literature. In section 3.3, we present the new formulation for the CTSIRPTW, that includes all previously mentioned characteristics. In Section 3.4, we introduce the different solution approaches, based on the idea of the ALNS, and we adapt the branch and

regret heuristic and multiple scenario approach algorithm (MSA) to solve the CTSIRPTW. A computational study is presented and analyzed in Section 3.5, where we compare the results obtained by means of the sample average estimator (SAE). Lastly, Section 3.6 summarizes the content and findings of this article, and we propose additional research directions characteristics related to the CTIRPTW.

The contribution of this paper is threefold: 1) We present a mathematical formulation for the new problem named CTSIRPTW. In this problem, a central distributor located at a central warehouse creates a delivery plan to minimize the overall costs. Each customer faces stochastic demands during a set of periods that form the planning horizon, the product is being consumed continuously at the customers' locations, consistency in the delivery times is required, and opening times differ between customers. 2) We present three versions of an algorithm to solve the CTSIRPTW. These methods use the idea of an ALNS, introduced in [5] for the deterministic consistent IRP with time-windows and split deliveries scenario, and include different stochasticity levels of demand uncertainty during the process. Furthermore, we include the solution of a reduced version of the mathematical formulation of the CTSIRPTW, when we find promising solutions within the search process. 3) We evaluate the performance of the presented algorithms. In addition to the proposed algorithms, we adapt two algorithms present in the literature which were developed for a single period stochastic vehicle routing problem (SVRP), the branch and regret heuristic first introduced in [50], and the MSA proposed in [81]. To create a fair comparison between the presented and the introduced methods, we use the SAE technique presented in [80]. We solve a set of instances presented in [5], where authors present a continuous-time IRP for a multi-period planning horizon, and consider deterministic demands. We adapt this set by considering different levels of stochasticity on the customers' demands. The set of instances will be made available when published¹.

3.2 Literature review

The CTSIRPTW belongs to the group of the IRPs whose origins goes back to [18], where the objective was to automatise the delivery process of liquid gases for the company Air Product. After the seminal paper was published, the interest for the IRP significantly increased and new variants related to its different parameters have been presented. Extensive reviews of the IRP literature are available in [24] and [34], and in [9]. As the purpose of this article is not to provide

¹<https://bda.univie.ac.at/research/data-and-instances/vehicle-routing-problems>

an exhaustive review of the IRP literature, we provide a review of the IRP works that share some of the most challenging and similar characteristics to the article at hand.

The newest characteristic of the problem is the continuous-time commodity consumption. This characteristic was first introduced in [60]. In this article, the authors consider a continuous-time variant of the IRP where the maximum quantities that can be delivered to each customer depends on the customer's inventory level at the moment that the deliveries take place. To solve the continuous-time variant of the IRP, the authors use a dynamic discovery algorithm to discretize the planning horizon and solve a small sequence of integer programs. However, unlike the problem we present in this article, the demands at the customers are considered deterministic, the planning horizon is one single period, and the possibilities of stock-out situations are not considered. A continuous-time deterministic IRP is extended to a multi-period planning horizon, and the stock-out situations and time-windows are considered in the problem that was first presented at the OR conference in 2017 ([6]), and recently published in [5]. Furthermore, the series of approaches proposed to solve the ROADEF/EURO Challenge 2016 are especially remarkable. This collection of articles ([10, 54, 2, 82, 46]) describe the model of the IRP of Air Liquide related to the distribution of bulk gases and propose different heuristic algorithms to solve the problem. They aim at solving the deterministic IRP, where customers present time-windows inside of the periods; therefore, time considerations are at the base of the proposed algorithm when constructing the delivery routes. However, the problem of the ROADEF/EURO Challenge 2016 is that the customers face demands at the end of the periods.

Several approaches have been presented to solve the deterministic variant of the problem. Exact solution methods for IRPs are considered in [4], [40], [35], [14]. On the other hand, many heuristic methods have been proposed to solve larger instances of the different IRP variants. A decomposition approach for the IRP was proposed in [29], where authors first create a delivery schedule for the customers and, in a second step, the set of delivery routes is created. In [47], the authors present an IRP that deals with the blood inventory and supply problems, and propose a variable neighborhood search to solve it. A general ALNS algorithm is presented in [33], and aims at solving different interpretations of consistent multi-vehicle IRPs. In [5], we present a matheuristic solution approach that combines an ALNS with the optimal solution of a mathematical subproblem, where the possibility of losing sales is also considered. In recent years, additional solution methods for IRPs have been introduced in the literature, an assign-and-route

matheuristic is presented in [83], in [13] the authors include a matheuristic solution approach, and a variable neighborhood search [69].

With respect to the stochastic variant of the problem, and, more specifically, in the literature related to the IRP with stochastic demands, in recent years we have found an increasing research interest. In the work that we present, customers experience a stochastic demand every period, and the vendor must incorporate this information to create efficient delivery plans that reduce the overall costs. In [53], [45], the authors introduced two simheuristic approaches for the stochastic IRP, with stock-outs, for single and multiple periods respectively. These simheuristic solution approaches combine a heuristic method with simulation on the stochastic demands to solve the considered problems. In [63], a stochastic variant of the IRP with time-windows is considered, and a two-stage heuristic is applied to solve the problem. In [66], authors formulated the stochastic IRP (SIRP) as an infinite horizon Markov decision process, and proposed a heuristic to reduce its complexity. In [55], authors allowed only direct deliveries in their Markov decision process formulation for the SIRP. Then, they extended this formulation to allow visiting a maximum of three retailers per route in [56], while in [3], the author limited the route duration instead. In recent years, new efficient algorithms have been presented, as in [38], where authors have proposed several approaches to solve the stochastic IRP for perishable products and compared them empirically, in terms of different aspects of the problem as average profit, product freshness, and service level. In [25], a dynamic lookahead policy is proposed to deal with the IRP related to bike-sharing systems. However, the continuous consumption of products, combined with other aspects of the problem we present as consistency and time-windows has not been considered previously in the literature related to the SIRP. For a more exhaustive review of the literature of SIRP approaches we refer to [74].

3.3 Problem Description

Consider a finite planning horizon, with $p \in P$ periods of length T , and a complete, directed graph $G = (V, A)$, with $V = 0, 1, \dots, n$, where node 0 denotes the depot, $V' = 1, \dots, n$ the set of customers, and A the set of all arcs. Each arc $(i, j) \in A$ has an associated cost c_{ij} and time t_{ij} , both non-negative values. Each customer location $i \in V'$, is characterized by a set of time-windows a_i^p, b_i^p , that represents the start and end times when the customer can be served on each period p , an initial inventory level I_i^0 and inventory capacity C_i , a service time s_i and a set

of random demands $\xi(i, p)$ which follows known stochastic distributions for each period p . We consider the demands at the customers to be continuously consumed during the periods. Finally, a finite and homogeneous fleet of vehicles is considered, $k \in K$, with capacity Q .

The CTSIRPTW is an NP-hard problem, as it is an stochastic extension of the VRP with time-windows, when we consider a single period (the VRPTW is an NP-hard problem [76]). In Equations (3.1)-(3.27), we present a mathematical formulation for the problem. The mathematical formulation uses different groups of decision variables. Each binary variable x_{ij}^{kp} is set to 1 if and only if edge (i, j) is being traversed by vehicle k in period p , and each binary variable y_i^{kp} indicates if customer i is visited by vehicle k in period p . Continuous variables t_i^{kp} , \bar{t}_i , and \underline{t}_i , are related to the time management of the problem. The first set of variables represent the arrival time of each vehicle k to a customer i in period p . The second and third sets, relate the earliest and latest arrival time to each customer i respectively. Finally, we have four additional groups of continuous decision variables that model the commodity flow of the problem. The first group of variables q_i^{kp} gives the amount of commodity delivered by each vehicle k in period p to each customer i . Second, I_i^p shows the inventory levels of each customer i at the end of each period p . Third, Δ_i represents the difference between the initial and the ending inventory level of each customer i , if the initial inventory is higher than the ending inventory. We use Δ_i to create a pseudo-rolling horizon and avoid empty ending inventories. Whenever the difference between the ending and the initial inventory levels is negative, we penalize this difference as lost sales in the objective function. This objective helps the central supplier to create a pseudo-rolling horizon and avoids customers to have empty inventories at the end of the planning horizon. Fourth, o_i^p represents the amount of lost sales at customer i in period p , due to stock-out situations. The notation used in the mathematical formulation is summarized in Table 3.1.

Data Sets	V	set of nodes
	V'	set of customers
	K	set of vehicles
	P	set of periods
Data and Parameters	c_{ij}	travel cost for arc (i, j)
	t_{ij}	travel time for arc (i, j)
	a_i^p, b_i^p	time-windows of customer i in period p
	C_i	inventory capacity of customer i
	I_i^0	initial inventory level of customer i in dummy period 0
	s_i	service time of customer i
	Q	vehicle capacity
	T	duration the period
	L	penalty per unit of sales lost and inventory decrease
Stochastic Data	$\xi(i, p)$	stochastic distribution of commodity consumption of customer i in period p
	$d(i, p)$	sample demand realization of customer i in period p given by the stochastic distribution $\xi(i, p)$
Decision Variables	x_{ij}^{kp}	use of arc (i, j) by vehicle k in period p
	x	set of all decision variables x_{ij}^{kp}
	y_i^{kp}	visit to customer i in period p
	I_i^p	inventory level of customer i at the end of period p
	t_i^{kp}	arrival time of vehicle k to customer i in period p
	$\underline{t}_i, \bar{t}_i$	earliest and latest arrival times to customer i
	\underline{t}, \bar{t}	sets of all decision variables $\underline{t}_i, \bar{t}_i$ respectively
	Δ_i	final inventory decrease of customer i
	Δ	set of all decision variables Δ_i
	q_i^{kp}	quantity delivered to customer i by vehicle k in period p
	o_i^p	quantity lost due to stock-outs of customer i in period p
	o	set of all decision variables o_i^p

Table 3.1: Notation

Using the above notation, we can formulate the CTSIRPTW. The problem is formulated as a two stage optimization problem where the objective is to minimize the fixed and expected costs that are incurred when considering stochastic demands (3.1).

$$\min f(x, \bar{t}, \underline{t}) + E|\tilde{G}(y_s, t_s, \tilde{\xi})| \quad (3.1)$$

The function $f(x, \bar{t}, \underline{t})$ evaluates the costs for the first stage of the decisions. The total routing cost incurred by servicing the customers, and the penalty imposed according to the difference between the earliest and latest delivery to a customer are calculated in this first step, creating a fixed delivery plan for the planning horizon. We define $f(x, \bar{t}, \underline{t})$ as follows:

$$f(x, \bar{t}, \underline{t}) = \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} \sum_{p \in P} c_{ij} x_{ij}^{kp} + \alpha \sum_{i \in V'} (\bar{t}_i - \underline{t}_i) \quad (3.2)$$

Then, $\tilde{G}(y_s, t_s, \tilde{\xi}_s)$ is the optimal value of the second stage problem when the real demands at the customers are revealed.

The first stage problem is subject to several sets of constraints that are ensuring a correct time, routing, and commodity flow. Some constraints are common to all demand sample realizations and established a correct time and routing flow. The constraints (3.3) to (3.13), serve this purpose, ensuring a fix routing and delivery schedule for all demand realizations. Constraints (3.3) to (3.4) guarantee that every customer can be visited by at most one vehicle in each period, and each vehicle can visit each customer in every period. Constraints (3.5) to (3.7) are flow conservation constraints that describe each individual route. Constraints (3.8) to (3.13) ensure the feasibility of the time schedule. Constraints (3.9) force t_i^{kp} to be 0, if customer i is not served by vehicle k in period p . We include two Big-M parameters (M_1 and M_2), related to the time management of the problem set, equal to the period length. Constraints (3.11) ensure that every vehicle must return to the depot before or at the moment the current period ends. Constraints (3.12) and (3.13) define the earliest and latest arrival times of each vehicle k at each customer i , along all periods p .

$$\sum_{j \in V} x_{ji}^{kp} = y_i^{kpr} \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (3.3)$$

$$\sum_{k \in K} y_i^{kp} \leq 1 \quad \forall i \in V', \forall p \in P \quad (3.4)$$

$$\sum_{i \in V'} x_{0i}^{kp} \leq 1 \quad \forall k \in K, \forall p \in P \quad (3.5)$$

$$\sum_{i \in V'} x_{0i}^{kp} - \sum_{i \in V'} x_{i(n+1)}^{kp} = 0 \quad \forall k \in K, \forall p \in P \quad (3.6)$$

$$\sum_{i \in V'} x_{ih}^{kp} - \sum_{j \in V'} x_{hj}^{kp} = 0 \quad \forall h \in V', \forall k \in K, \forall p \in P \quad (3.7)$$

$$t_i^{kp} + s_i + t_{ij} - M_1(1 - x_{ij}^{kp}) \leq t_j^{kp} \quad \forall i, j \in V, \forall k \in K, \forall p \in P \quad (3.8)$$

$$a_i^p y_i^{kp} \leq t_i^{kp} \leq b_i^p y_i^{kp} \quad \forall i \in V, \forall k \in K, \forall p \in P \quad (3.9)$$

$$t_0^{kp} \geq 0 \quad \forall k \in K, \forall p \in P \quad (3.10)$$

$$t_i^{kp} \leq T \quad \forall i \in V, \forall k \in K, \forall p \in P \quad (3.11)$$

$$\bar{t}_i \geq t_i^{kp} - M_2(1 - y_i^{kp}) \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (3.12)$$

$$\underline{t}_i \leq t_i^{kp} + M_2(1 - y_i^{kp}) \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (3.13)$$

Given the first stage decisions \tilde{y}_s, \tilde{t}_s that establish which customer must be visited at each period and the time of these visits, the demand realizations $\tilde{\xi}$ result in an inventory problem with stock-outs. We calculate the other cost factors, such as ones caused by stock-out situations, and the difference between the initial and ending inventory levels at each customer; when this difference is positive. The objective of the second stage optimization problem is then defined as:

$$\tilde{G}(\tilde{y}_s, \tilde{t}_s, \tilde{\xi}_s) = \min \sum_{i \in V'} \sum_{p \in P} L o_i^p + \sum_{i \in V'} L \Delta_i \quad (3.14)$$

The first group of constraints of the second stage problem deals with commodity and inventory management for the demand realizations. The realization of the demands is a stochastic event that occurs in the second stage of the two-stage mathematical formulation. The demands are disclosed at the end of the planning horizon. Then, when we obtain these demands, the complete calcula-

tion of inventory and lost sales is calculated, considering the consumption of demands to be linear within the time periods. Constraints (3.15) ensure that the maximum vehicle capacity is respected, while constraints (3.16) indicate that a customer cannot receive any amount by a vehicle if it is not visited by that vehicle in a period. The Big-M parameter (M_3) in these constraints is equal to the vehicle capacity. Constraints (3.17) define the inventory levels of every customer at the end of each period for a sample demand realization, calculated as the sum of the previous inventory level plus the amount of commodity received in the current period and the amount that the customer is not able to serve due to stock-outs $I_i^p - I_i^{(p-1)} = \sum_{k \in K} q_i^{kp} - \xi(i, p) + o_i^p \forall i \in V', \forall p \in P$. This value is further decreased by the commodity consumption of the customer, given by the realization of the customer's demand in the current period. Constraints (3.18) and (3.19) forbid inventory levels to be negative or exceed the inventory capacity of the customer. With constraints (3.20) and (3.21), we calculate the difference between the initial and the ending inventory of each customer when the difference is positive.

$$\sum_{i \in V'} q_i^{kp} \leq Q \quad \forall k \in K, \forall p \in P \quad (3.15)$$

$$q_i^{kp} \leq \tilde{y}_i^{kp} M_3 \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (3.16)$$

$$I_i^p - I_i^{(p-1)} = \sum_{k \in K} q_i^{kp} - d(i, p) + o_i^p \quad \forall i \in V', \forall p \in P \quad (3.17)$$

$$I_i^p \geq 0 \quad \forall i \in V, \forall p \in P \quad (3.18)$$

$$I_i^p \leq C_i \quad \forall i \in V', \forall p \in P \quad (3.19)$$

$$I_i^0 - I_i^{|P|} \geq \Delta_i \quad \forall i \in V' \quad (3.20)$$

$$\Delta_i \geq 0 \quad \forall i \in V' \quad (3.21)$$

The second group of constraints of the second stage problem, (3.22) and (3.23), deal with the continuous-time commodity consumption at the customers end. For each demand realization, we create some constraints to avoid excessive stock at the customers at any time. In constraints (3.22) and (3.23), we also account for the possible stock-outs that occur at any time within the

time periods. Constraints (3.22) ensure that at the arrival time of any vehicle to a customer, the inventory level is non-negative. Whenever a stock-out occurs, the out-of-stock amount is calculated. Constraints (3.23) act in the opposite way, to avoid excessive inventory levels.

$$I_i^{(p-1)} - d(i, p)\tilde{t}_i^{kp}/T + o_i^p \geq 0 \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (3.22)$$

$$q_i^{kp} + I_i^{(p-1)} - d(i, p)\tilde{t}_i^{kp}/T + o_i^p \leq C_i \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (3.23)$$

Finally, in constraints (3.24) to (3.27), we define the domain of each variable of the problem.

$$x_{ij}^{kp} \in \{0, 1\} \quad \forall i, j \in V, \forall k \in K, \forall p \in P \quad (3.24)$$

$$y_i^{kp} \in \{0, 1\} \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (3.25)$$

$$t_i^{kp} \geq 0 \quad \forall i \in V, \forall k \in K, \forall p \in P \quad (3.26)$$

$$q_i^{kp} \geq 0 \quad \forall i \in V', \forall k \in K, \forall p \in P \quad (3.27)$$

For further understanding of the inventory management of the problem, in Figure 3.1, we represent the inventory flow of an example customer in a random realization of the demands. In this figure we can see how, for a specific customer and a planning horizon of three periods, the customer has a given initial inventory level. Then, at each period, the demand that this customer experiences is consumed continuously during the periods (1). Customers must be delivered within a certain time window (2) and, if a delivery takes place when the inventory is empty, the lost sales are calculated and penalized in the objective function (3). Furthermore, at the end of the planning horizon we compare the difference between the initial and the final inventory level (4).

If the initial inventory level is higher than the final one, the difference between inventory levels is calculated and penalized in the objective function as sales lost.

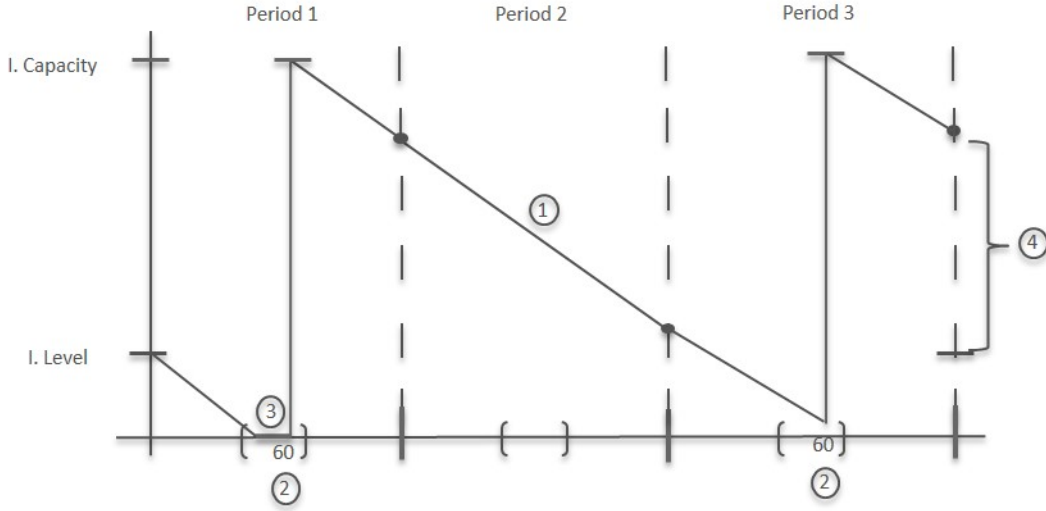


Figure 3.1: Example of the inventory flow of a customer. Taken from [6]

In the second stage of the problem, random demands for each period and customer is generated. This demand is assumed to be linear and continuous along the periods. Therefore, whenever a customer does not have the sufficient amount of stock to satisfy the amount of demand in a period, we calculate this amount and penalize it in the objective function. This penalization aims at representing the real-life context and helps in obtaining a robust solution that can minimize the total expected costs. In addition to the proposed mathematical formulation, we evaluate the impact of considering recourse actions to reduce costs related to sales lost and final inventory levels.

3.4 Solution Approaches

In this section, we present different methods to solve the CTSIRPTW. We propose a new population based matheuristic solution approach based on the idea of the ALNS algorithm [68], which is proven to be an efficient algorithm to solve IRPs, and we adapt different solution approaches existing in the literature. We present three variants of the a matheuristic algorithm that combines an ALNS with a mathematical subproblem solution. We propose a population based ALNS algorithm and two simpler versions where different stochastic demand samples are

generated within the algorithms. After we present the proposed solution approaches, we describe two additional recourse actions that the central supplier could apply to reduce the overall logistic costs. We also introduce the mathematical subproblem that is optimally solved in the core of all the presented methods. Furthermore, the methods we adapt as a comparison are based on the branch and regret heuristic presented in [50], and in the idea of a MSA presented in [81], and first introduced in [19], respectively.

3.4.1 Adaptive large neighborhood search population based algorithm

The first group of solution approaches that we present in this article are different variations of a matheuristic solution approach that integrates the mathematical subproblem described in Section 3.3.2. and the ALNS procedure. The first two variants resemble the solution approach presented in [5], whereas the third variant, the multi-solution stochastic ALNS, is based on a population based matheuristic algorithm. In general, the three algorithms we present consider different levels of sample generation and evaluation within the algorithms. The common structure of the approaches is represented in Algorithm 1.

To generate an initial solution for the ALNS algorithm, we apply a constructive heuristic. The initial solution is generated using the cheapest insertion heuristic, presented in [79]. For each period of the planning horizon, we decide which customers must be visited. We create a priority list containing all customers whose inventory levels at the beginning of the period is not sufficient to survive without a stock-out, before the next possible delivery time window. When we obtain the list of customers to be served in the current period, we insert them using the cheapest insertion algorithm. We first initialize a route by inserting the customer farthest away from the depot from the list of customers we just created. Then, we insert the customers requiring the largest possible quantity. When we cannot insert any more customers in the route, due to capacity constraints of trucks or the time-windows of the remaining customers, we initialize a new route in the same way. We iterate until no more insertions are possible. We then apply a local search operator to improve the quality of the initial solution and solve a mathematical subproblem to determine the optimal timing and delivery quantities. Note that this can also lead to waiting times when necessary. The local search that we apply is the well-known 2-opt algorithm with a maximum length of four customers. To ensure feasibility, we do not allow any changes that may violate the time-windows of the customers. Thus, we search for better sequences of customers, that satisfy the other constraints of the problem.

Algorithm 1 ALNS

Require: Input Data

```

1: for  $p = 1 : |P|$  do
2:   Obtain the set of customers to be replenished in the current period
3:   Create a delivery plan for the current period for the customers in the list
4:   Local search 1: delete single customer routes
5:   Local search 2: 2-opt
6: Solve exact subproblem
7: Start ALNS algorithm using the initial solution  $s_{initial}$  with the corresponding costs
    $c_{initial}$  from constructive heuristic
8:  $s_{best}, s_{incumbent} \leftarrow s_{initial}$ 
9:  $c_{best}, c_{incumbent} \leftarrow c_{initial}$ 
10:  $t, t_{last} \leftarrow 0$  Start time and time of last improvement
11: while time  $t < \text{MAXTIME}$  do
12:   Generate new sample  $q$ 
13:    $s_{current} = s_{incumbent}$ 
14:   Select a pair of destroy and repair operators  $d$  and  $r$ 
15:   Apply destroy and repair operators to  $s_{current}$  using demands in sample  $q$ 
16:   2-opt local search
17:   Evaluate cost  $c_{current}$  of the new solution obtained  $s_{current}$ 
18:   if ( $c_{current} < 1.5 \cdot c_{best}$ ) then
19:     Generate a pool of samples  $R$ 
20:     Evaluate solution and update cost  $c_{current}$  using samples in set  $R$ 
21:     if ( $c_{current} < c_{best}$ ) then
22:        $s_{best}, s_{incumbent} \leftarrow s_{current}$ 
23:        $c_{best}, c_{incumbent} \leftarrow c_{current}$ 
24:       Update weight  $\rho_{dr}$  of operators  $d$  and  $r$ 
25:     else
26:       if ( $c_{current} < c_{incumbent}$ ) then
27:          $s_{incumbent} \leftarrow s_{current}$ 
28:          $c_{incumbent} \leftarrow c_{current}$ 
29:       else
30:         if ( $t - t_{last} > \text{timelimit}$ ) then
31:            $s_{incumbent} \leftarrow s_{current}$ 
32:            $c_{incumbent} \leftarrow c_{current}$ 
return  $s_{best}$  &  $c_{best}$ 

```

When we obtain an initial solution $s_{initial}$ with cost $c_{initial}$ via the construction heuristic, we apply the ALNS to improve it. During this procedure, we save the information of the best solution found s_{best} and its cost c_{best} , and the information of the incumbent solution that we use at each iteration $s_{incumbent}$ and $c_{incumbent}$ to compare with the new solution. We propose three different variants of the algorithm. The methods integrate several operators to iteratively destroy and repair the current solution, and use different demand samples to simulate possible scenarios. In Algorithm 1 we show how, at each iteration of the ALNS, we generate a new solution by applying one destroy and one repair operator to a demand sample q . In the first variant we present, this demand sample q considers the expected demands for each customer and period, whereas for the second and third variations presented, at each iteration of the ALNS procedure, a random sample of demands is generated from the stochastic demand distributions of the customers. The selection of these operators relies on a roulette-wheel selection operator based on past performance. The selection and prioritization of destroy and repair operators are pairwise, rather than separate.

Deterministic ALNS

The first version of the matheuristic solution approach that we propose, considers the deterministic variant of the problem. At each iteration, we apply the repair and destroy operator, considering expected demands at the customers.

Weights ρ_{dr} related to each pair of destroy and repair operators are initially set to the same initial value; when we find a new best solution or better incumbent solution, we update the weights of the operators, increasing them by γ and σ , respectively, then we resume the procedure of updating the operator weights as follows:

$$\rho_{dr} = \begin{cases} \rho_{dr} + \gamma & \text{if new best solution found.} \\ \rho_{dr} + \sigma & \text{if new better incumbent solution found.} \\ \rho_{dr} & \text{if no better solution found.} \end{cases} \quad (3.28)$$

After applying the destroy and repair operators, if the new solution $s_{current}$ has a cost $c_{current}$ not more than 50% worse than the best known solution, we generate r samples and solve the reduced variant of the mathematical problem formulation. We consider a maximum 50% decrease in the quality of the solution, as it is applied in [5]. Finally, we have a new solution with optimal

costs for these route sequences. If the new solution improves the average costs of the incumbent solution $s_{incumbent}$, we update the incumbent solution and the weights for the destroy and repair operators used in this iteration. We do the same with the best known solution s_{best} if the new solution improves it. However, if the new solution does not improve the incumbent or best known solution, we proceed in two different ways. If the total time since the last improvement is lower than a *timelimit*, we use the previous incumbent solution as the starting solution for the next ALNS iteration. If the total time since the last improvement exceeds that time limit, we accept the new solution as the incumbent solution, and therefore, as the input for the next iteration of the ALNS. Then, we reset the time of the last improvement found to zero. We accept this new solution to avoid strong local optima that prevent the method from continuing to improve the quality of the solutions. The stopping criterion for the ALNS is an overall limit on runtime.

Stochastic ALNS

The second version of the matheuristic algorithm resembles the deterministic variant. However, in this approach, at each iteration of the algorithm, we generate a random sample using the Monte-Carlo sampling technique, and we apply the correspondent operators to a stochastic scenario. This version causes both positive and negative situations. On the one hand, it is possible that certain samples show low commodity consumption at the retailers and, as a result, the related inventory and stock-out costs are relatively low. However, on the other hand, considering stochastic scenarios at each iteration allows the learning procedure of the algorithm to account for all possible situations that may occur during the planning horizon. In this case, to be able to evaluate a higher range of solutions found during the solution approach and, to include in the later evaluation solutions with high commodity consumption, we increase the range of solutions that are evaluated. If the expected costs of the new solution are, at maximum, twice the costs of the best solution found, we generate a pool of samples, as in the first version of the algorithm, and solve the mathematical subproblem using all samples in the generated set. Then, we apply the adaptive procedure and update the operator weights, as in the previous version.

Multi-solution stochastic ALNS

Finally, the third version of the algorithm is the population based matheuristic algorithm, which includes one of the most important aspects of the MSA algorithm. In this case, the iterative procedure is similar to the second version of the algorithm. However, in this approach, we do

not consider the best solution found during the algorithm, but a set of good solutions S . At each iteration, by means of Monte-Carlo sampling technique, we randomly generate a demand scenario for all customers. Then, we apply the destroy and repair operators to the generated sample. When we obtain a new solution, we then compare the expected costs of the solution to the rest of the solutions in the set S . If the costs of the new solution are not worse than 75% of the worst solution in the set, we generate a new pool of samples and apply an evaluation procedure to obtain the average costs of the solution. In this case, if the new solution improves one of the existing solutions in the set S , we insert this new solution in the set and discard the worst solution stored. In this algorithm, we also vary the adaptive procedure of the algorithm in the following manner:

$$\rho_{idr} = \begin{cases} \rho_{(i-1)dr} + \gamma & \text{if new best solution found.} \\ \rho_{(i-1)dr} + \theta & \text{if new accepted solution in set } S. \\ \rho_{(i-1)dr} + \sigma & \text{if new better incumbent solution found.} \\ \rho_{(i-1)dr} & \text{if no better solution found.} \end{cases} \quad (3.29)$$

As a common framework for the three variants of the matheuristic solution approach, we propose six destroy operators and seven destroy operators. The repair operators are described as follows:

- Remove worst insertions: Given a solution s , we define the cost of the detour caused by the insertion of customer i as $cost = c_{ki} + c_{ij} - c_{kj}$, where k and j are the preceding and succeeding customers in the route, respectively. Then, the operator calculates the detour cost caused by every insertion, selects the p worst insertions with respect to these costs, and removes them from the solution.
- Remove random insertions: This operator selects p different insertions from random periods, and routes, and removes them from the solution.
- Remove vehicle: This operator selects a vehicle at random and removes all deliveries performed by this vehicle, in every period.
- Remove customer: This operator selects p customers at random and removes all their deliveries.

- Remove customer and closest: This operator selects p customers at random and removes all deliveries to these customers and their closest customers in terms of distance.
- Remove similar inventory customers: This operator selects p customers at random and, for each, selects one other customer with a similar percentage of initial inventory level. Then, it removes all deliveries to all selected customers.

Then, after applying a destroy operator, we select one of the following repair operators:

- Best insertion in destroy order: This operator inserts customers in the same order in which they were removed from the previous solution. To reinsert these customers into the existing routes, we create a list of possible insertions for the period a stock-out occurs as well as all the preceding periods. The possible insertions are sorted by the total distance of the detour caused by them. We then randomly select one of the three best possible insertion positions and perform the insertion. If, after evaluating the new inventory flow of the customer, there are still stock-out situations in the later periods, we repeat the process between the new stock-out period and the last insertion period.
- Best insertion in random order: This operator works in the same way as the previous operator, except that we randomly select the next customer to be reinserted into the solution.
- Random insertion in destroy order: This operator creates a list of possible insertions, just like the *best insertion in destroy order* operator, but then randomly selects one of all possible insertions from the entire list.
- Random insertion in random order: This operator creates a list of possible insertions just as the previous operator, and then randomly selects one of them from the entire list. We repeat the process until a feasible solution is reached.
- Consistent insertion in destroy order: This operator inserts customers such that the difference between the earliest and the latest arrival times to this customer is minimized.
- Consistent insertion in random order: This operator inserts customers in the same way as the previous operator but, in this case, the customer to be inserted is selected randomly and not in the destroy order.
- Distance-related insertion: This operator can be considered as an extension of the *best insertion in destroy order* operator. It selects customers in the same order as they were

removed from the solution, and we calculate the best possible insertions in terms of the detour they cause. When we insert the selected customer, we search for and identify the closest customer, in terms of distance, from the list of customers to be reinserted. We then insert this new customer next to the previous customer if it's possible. If we can reduce the lost sales by inserting this customer, we remove both customers from the list(not just the first).

Furthermore, the three variants of the ALNS presented include an additional adjustment mechanism for the operator weights. The scores are updated after a certain time segment j as follows:

$$\rho_{i,j+1} = \alpha \frac{\bar{\rho}_{i,j}}{a_i} + (1 - \alpha)\rho_{i,j} \quad (3.30)$$

This procedure calculates *smoothened* scores after each time segment j that, in the proposed algorithms, correspond to every 30 seconds of the total ALNS time (10 minutes of running time in our computational experiments). The factor $\bar{\rho}_{i,j}$ represents the observed score of each operator pair i in time segment j . At the end of each time segment, a_i accounts for the number of times each operator has been used in the time segments, and the reaction factor α is set to 0.5. This weight helps the algorithm take into account equally the scores of the most recent segment and the scores of the previous time segments.

3.4.2 Mathematical subproblem

In our considered CTSIRPTW, it is possible to determine the delivery quantities, the delivery times, and evaluate the quality of the solutions that are iteratively generated within the solution approaches. At each step of the algorithms, we obtain feasible solutions. However, these solutions may not be optimal for this route sequence, the delivery quantities and delivery times can be reoptimized to reduce the consistency costs, and the expected inventory and stock-out costs. This mathematical subproblem is embodied within the algorithms. Whenever we have a set of routes that define the delivery plan for a range of periods, we can obtain the cost of the current solution for an instance sample as well as the different delivery quantities and delivery times necessary. To obtain the optimal delivery quantities and times as well as calculate the correct inventory levels for each customer, we solve a reduced problem based on the problem formulation.

The reduced problem contains all constraints of the original problem formulation by using the maximum level policy. This policy allows the central supplier to deliver any amount, up to the inventory capacity of the customer. However, the customer visits y_i^{kp} , and route sequences x_{ij}^{kp} are no longer variables and, instead, function as parameters. As a result of the fixation of the integer and binary variables, the mathematical problems that we solve are linear and the solving times are lower than seconds for the bigger instances that we present.

3.4.3 Recourse Strategies

The problem presented in this article is a stochastic variant of the IRP. However, in practice, companies could apply recourse strategies to reduce the expected costs during the planning horizon. In this subsection, we present two additional recourse strategies in addition to the stock-out penalty applied whenever a customer experiences lost sales.

The first recourse strategy we consider creates additional routes for each period to restock customers that may experience a shortage in their inventory at a given period. In this setting, at the beginning of each period p , the initial inventory of each customer n is compared to the expected demand during that period. For those customers which may incur into a stock-out situation, and are not planned to be visited in the current period, we create additional routes to replenish their inventory. In this recourse strategy, we consider the earliest and latest arrival times calculated in the first stage of the algorithm and we deliver to these customers within these time-slots to not increase the consistency costs. The amount delivered to each customer considered is equal to the expected lost sales from the period p until the next delivery takes place. Furthermore, in the last period of the planning horizon, we calculate the expected final inventory level of each customer, and compare its initial inventory level. If the expected final inventory level is lower than the initial, and the customer is not replenished in the final period, we create an additional delivery to this customer using the same strategy previously mentioned.

The second recourse strategy evaluates the customers for each period p in the same manner as the first recourse strategy. However, in this strategy we try to redesign the delivery plans by creating new visits to the customers at hand, and inserting the respective deliveries into already existing routes. This recourse strategy tries to insert the deliveries to the additional customers in the existing routes, nevertheless, postponing deliveries to customers that already have a scheduled visit could create additional stock-out situations. For this reason, the additional deliveries must

not create delays in the scheduled arrivals to the already introduced customers, and the vehicle capacity of the vehicles must be respected.

3.4.4 Adapted algorithms

To obtain an initial solution for the CTSIRPTW and compare the results obtained with the new proposed algorithm, we adapt two algorithms present in the literature to solve the problem. These methods solve different variants of the dynamic and stochastic VRP which, through the artificial division of the planning horizon into different periods, are able to readjust the initial solution. In this sense, this artificial division of the planning horizon resembles the IRP structure. The algorithms we adapt are iterative procedures that evaluate the efficiency of the solutions by considering different scenarios. These methods were originally proposed to solve the stochastic dynamic vehicle routing. We extend these methods to a multiperiod scenario and use the stochastic information to create different demand samples to optimize the problem along the planning horizon.

Branch and regret algorithm

The first algorithm we adapt is based on the branch and regret heuristic, first introduced in [50]. In this article, the authors dealt with a stochastic VRP motivated by a real-world problem faced by a major transporter in Norway. In the original method, the proposed heuristic artificially divides the planning horizon into different subperiods related to the different events that may occur during this horizon. In this way, whenever a customer is visited, or new customers that need to be visited appear, the algorithm recalculates the given routes to consider the new information. For a better understanding of the algorithm, we show the pseudo-code presented in the original work in Appendix A.

We adapt this algorithm to solve the CTSIRPTW by taking the beginning of each period as the recalculating point. In this way, the division of the planning horizon occurs naturally and corresponds to the different periods presented in the problem. This solution approach applies two main loops and, in addition, also integrates the creation of different sample scenarios, using the probability distributions of each customer demand.

Given that each period is considered separately in the planning horizon I_1, I_2, \dots, I_p , the algorithm can be described as having two parts; an outer and an inner iterative procedure. In Algorithm 2, we show the outer procedure. At each period, the routes previously calculated

are fixed and new demand samples using Monte-Carlo sampling are generated. Then, using the previous information and the generated samples, we apply the inner procedure to obtain the plan for the current period.

Algorithm 2 Outer procedure branch and regret algorithm

```

for Each period  $I_1, I_2, \dots, I_p$  do
    Take the solution from the previous interval and fix the plan up to the current
    period.
    Generate  $r$  demand samples using the probability distributions
    Find plans for the current period using the inner procedure (Algorithm 3).
    Evaluate the solution.
Return obtained solution

```

The idea of the inner procedure of the branch and regret heuristic follows the same concept as in the original paper. This iterative procedure aims at evaluating the impact of visiting customers in the current period, or in subsequent ones. In Algorithm 3, we report the pseudo code of the procedure. The inputs for this algorithm are the solution constructed until the current period and the set of samples generated in the outer procedure. In this step, to initially obtain the inventory levels and the optimal delivery amounts for each sample, we solve the mathematical subproblem described in Section 3.3.1. Then, when we have the current inventory state of each customer, after which we initialize two sets, S_{in} and S_{out} , initially empty, and we apply an iterative procedure to evaluate different delivery plans. At each iteration, we create a delivery plan for each sample, forcing all customers in S_{in} to be served in the current period and forbidding S_{out} to be visited in the same, while in the first iteration, no customer is forced or forbidden into the solution. We create delivery routes by means of the cheapest insertion algorithm described in [79] and a local search based on the 2-opt algorithm. After we obtain the delivery routes for each sample, we identify the customer c most commonly replenished in the solutions corresponding to each generated sample. Using this information, at each iteration, we solve again the problem and create two delivery plans for each sample generated, with customer c being included in set S_{in} and set S_{out} . When we create the delivery plans for each sample we evaluate the impact of delivering customer c in the current period, or in subsequent ones, and select the option that provides the better results. We then add customer c to the set that provides the best average cost and repeat the iterative procedure until all customers present in the solutions of the samples are evaluated. When this iterative procedure ends, we terminate the inner procedure with a delivery

plan for the current period.

Algorithm 3 Inner procedure branch and regret algorithm

```

for Each sample generated do
  Create  $S_{in}$  and  $S_{out}$  initially empty
  Solve a mathematical subproblem to obtain the optimal delivery plan for the
  previous periods
  Decide which customers must be replenished for the current sample and period
  Generate the routes using the list of customers that need to be replenished
   $Stop \rightarrow FALSE$ 
while  $Stop = FALSE$  do
  Find the customer  $c \in N \setminus (S_{in} \cup S_{out})$  most frequently visited
  if No further customer  $c \in N \setminus (S_{in} \cup S_{out})$  is visited in the scenarios then
     $Stop \rightarrow TRUE$ 
  else
    Solve the samples for the current and subsequent periods adding customer  $c$  to
     $S_{in}$  and  $S_{out}$  respectively
    Add customer  $c$  to whichever set that is evaluated as the best choice
  Terminate with a plan for the current period
  
```

Multiple scenario approach

The second algorithm adapted from the literature is the MSA, which was first introduced in [19]. In this work, the authors solve a dynamic multi-vehicle routing problem with time-windows, wherein customer's appearance is considered to be stochastic. The MSA continuously generate routing plans for scenarios including both known and stochastic future requests. In recent years, the MSA has been adapted to solve a wide range of problems. We find in [81] an adaptation of the MSA for the two stage problem related to the stochastic team-orienteering problem with consistency constraints.

We adapt the algorithm presented in [81] to solve the problem we introduce in this paper. In this algorithm we create a pool of different solutions that are proposed for different demand scenarios. In the first step of the algorithm, we generate different demand samples providing a deterministic demand realization for each customer and period. Then for each demand sample, we solve an independent inventory routing problem with deterministic demands. In the second step of the proposed algorithm, we propose a consensus function to use the information obtained from each individual solution and we propose a solution for the CTSIRPTW.

In order to obtain a solution for the deterministic inventory routing problems related to each demand sample, we apply the deterministic version of the algorithm described in section 3.4.1. We first generate an initial solution by means of the cheapest insertion heuristic. In this heuristic, we decide which customers must be replenished on each period using the demand samples generated, instead of the expected demand as in the algorithm previously described. We then apply the well-known 2-opt algorithm with a maximum length of four customers to improve the quality of initial solutions generated. To ensure feasibility, we do not allow any changes that may violate the time-windows of the customers. Thus, we search for better sequences of customers, that satisfy the other constraints of the problem.

After we obtain an initial solution for a demand realization, we apply the deterministic version of the ALNS, described in section 3.4.1.1. However, at each iteration of the ALNS algorithm, we do not generate additional demand samples but, on the other hand, we calculate costs related to the solutions using the information of the demand sample at hand. In Algorithm 4, we present the pseudocode of the proposed algorithm.

When the iterative procedure ends, we have generated a solution for each demand sample $n \in N$ generated at the beginning of the algorithm. Then, in a second step of the algorithm, we apply a consensus function to select a solution from the set of generated solutions N . The consensus function we apply follows the idea of the proposed consensus function in [81]. We evaluate the quality of each solution by defining a function that measures the difference between the visit variables y_n and $y'_{n'}$, corresponding to two different demand samples n and n' . This consensus function is based on the Hamming distance, which measures the difference between matrices by comparing individual entries. Therefore, we can evaluate the difference between two solutions related to two demand samples by evaluating if each customer is being replenished in the same periods in both solutions. The Hamming distance between two solution is calculated as follows:

$$H(y_n, y'_{n'}) := \sum_{i \in V'} \sum_{p \in P} |(y_n)_i^p - (y'_{n'})_i^p| \quad (3.31)$$

Algorithm 4 Multiple scenario approach

```

1: Generate  $n$  demand samples using the probability distributions
2: for each sample  $n$  do
3:   for  $p = 1 : |P|$  do
4:     Obtain the set of customers to be replenished in the current period and sample
        $n$ 
5:     Create a delivery plan for the current period for the customers in the list
6:     Local search 1: delete single customer routes
7:     Local search 2: 2-opt
8:     Solve exact subproblem
9:     Start ALNS algorithm using the initial solution  $s_{initial}$  with the corresponding
       costs  $c_{initial}$  from constructive heuristic
10:     $s_{best}, s_{incumbent} \leftarrow s_{initial}$ 
11:     $c_{best}, c_{incumbent} \leftarrow c_{initial}$ 
12:     $t, t_{last} \leftarrow 0$  Start time and time of last improvement
13:    while time  $t < \text{MAXTIME}$  do
14:       $s_{current} = s_{incumbent}$ 
15:      Select a pair of destroy and repair operators  $d$  and  $r$ 
16:      Apply destroy and repair operators to  $s_{current}$  using demands in sample  $n$ 
17:      2-opt local search
18:      Evaluate cost  $c_{current}$  of the new solution obtained  $s_{current}$ 
19:      if ( $c_{current} < 1.5 \cdot c_{best}$ ) then
20:        if ( $c_{current} < c_{best}$ ) then
21:           $s_{best}, s_{incumbent} \leftarrow s_{current}$ 
22:           $c_{best}, c_{incumbent} \leftarrow c_{current}$ 
23:          Update weight  $\rho_{dr}$  of operators  $d$  and  $r$ 
24:        else
25:          if ( $c_{current} < c_{incumbent}$ ) then
26:             $s_{incumbent} \leftarrow s_{current}$ 
27:             $c_{incumbent} \leftarrow c_{current}$ 
28:          else
29:            if ( $t - t_{last} > \text{timelimit}$ ) then
30:               $s_{incumbent} \leftarrow s_{current}$ 
31:               $c_{incumbent} \leftarrow c_{current}$ 
    return  $s_{best}$ 

```

Then, the solution selected is the solution that is closest to the other solutions for the other demand samples in terms of the Hamming distance defined.

$$\bar{n} \in \arg \min_{n \in N} \sum_{n' \in N, n' \neq n} H(y_n, y_{n'}) \quad (3.32)$$

3.5 Computational Experiments

The continuous-time version of the IRP, and especially the stochastic version, is a relatively new problem in the literature. To gain insights into the problem properties, we performed computational experiments on sets of instances, adapted from a benchmark set. Other sets of instances have been proposed to solve other variants of IRPs, but the time window characteristic and the continuous commodity consumption considered in our problem makes it rather difficult to adapt these instances to our problem setting.

The algorithm we propose was coded in C++, and all computational experiments were performed on a Linux system, equipped with two Intel Xeon E5-2650(2.6 GHz) processors and 64 GB RAM. IBM CPLEX 12.7 was used as a MIP solver, with a maximum computation solving time of 10 hours.

3.5.1 Test instances

In [60], the authors introduce the continuous-time variant of the problem and present results for a set of instances of maximum 15 customers, considering a planning horizon of one period. However, authors do not consider the possibility of stock-out situations, and a customer can be visited multiple times in the same period. To evaluate the effectiveness of the presented methods, we use a set of instances introduced in [5]. This sets of instances was initially adapted from a benchmark set [36] that pertains to the periodic vehicle routing problem with time-windows(PVRPTW).

These instances include information of different number of customers (5, 10, 15, 20, and 48 customers), and a planning horizon of four and eight periods. The instances use the customer information related to coordinates and time-windows from the instances of size 48 in the benchmark set. Vehicle capacity is set to 2,000 units, and the fleet size is such that it can deliver, within a single period, twice the average commodity consumption per period. Furthermore, we

set the penalty cost $L = 3$ for each unit of demand that customers are not able to satisfy, or unit of difference between the initial and ending inventory level and the consistency cost weight $\alpha = 1$. We select $\alpha = 1$ because the benchmark instances [36] consider time-windows that depend on the routing distances, so both routing and consistency costs are measured using the same unit. We also set the penalty cost $L = 3$ to balance the possible stock-out and inventory costs, relative to the rest of the costs considered in the problem. For further information about the instances, we refer to [5].

For each instance size, we generate three groups of 10 instances. Demands that each customer experiences follow a normal distribution $\xi(i, p)$, and demands are considered independent across customers. The main difference between the three groups of instances is the stochastic factor. In the first group of each instance size, the standard deviation of each demand is randomly set between 10% and 20%, in the second group, this deviation is randomly set between 20% and 30%, and in the last group it is set between 40% and 50%. The characteristics of the proposed instances are summarized in Table 3.2.

Group	Size	Periods	Vehicles	Std. Dev	Group	Size	Periods	Vehicles	Std. Dev
4p1a	5	4	1	Low	8p1a	5	8	1	Low
4p1b	5	4	1	Medium	8p1b	5	8	1	Medium
4p1c	5	4	1	High	8p1c	5	8	1	High
4p2a	10	4	2	Low	8p2a	10	8	2	Low
4p2b	10	4	2	Medium	8p2b	10	8	2	Medium
4p2c	10	4	2	High	8p2c	10	8	2	High
4p3a	15	4	2	Low	8p3a	15	8	2	Low
4p3b	15	4	2	Medium	8p3b	15	8	2	Medium
4p3c	15	4	2	High	8p3c	15	8	2	High
4p4a	20	4	3	Low	8p4a	20	8	3	Low
4p4b	20	4	3	Medium	8p4b	20	8	3	Medium
4p4c	20	4	3	High	8p4c	20	8	3	High
4p5a	48	4	6	Low	8p5a	48	8	6	Low
4p5b	48	4	6	Medium	8p5b	48	8	6	Medium
4p5c	48	4	6	High	8p5c	48	8	6	High

Table 3.2: Characteristics of instances for the CTSIRPTW

3.5.2 Evaluation mechanism

To create a fair comparison between the presented methods, we apply the SAE used in [80]. For this evaluation procedure, we use as input the different solutions found using the proposed algorithms. The pseudocode of the evaluation procedure is shown in Algorithm 5.

Algorithm 5 Sample Average Estimator

```

Request as input a solution  $s$  found by one of the solution approaches
Generate a pool of samples  $R$ 
 $Cost_{total} \leftarrow 0$ 
for Each sample  $r \in R$  do
    Solve the mathematical subproblem using solution  $s$  and sample  $r$ . Obtain  $Cost_s^r$ 
     $Cost_{total} += Cost_s^r$ 
 $AverageCost \leftarrow Cost_{total}/|R|$  return  $AverageCost$ 

```

In this algorithm, we generate a large pool of demand samples using the Monte-Carlo sampling technique. Then, using the solution obtained by each of the algorithms, we obtain the average costs of the solution, using a similar mathematical subproblem to the one presented in Section 3.3.1. However, in this mathematical subproblem, in addition to the customer visits y_i^{kp} and route sequences x_{ij}^{kp} variables, the arrival t_i^{kp} are considered parameters. To create a fair comparison, we use the same samples to evaluate the solutions obtained by all algorithms. This is made by fixing a random seed for the algorithm; therefore, the samples generated are the same for all solutions.

3.5.3 Computational results

To evaluate the effectiveness of the proposed method and compare the obtained results, each of the presented methods is applied using the set of instances described before. In Table 3.3, we show average costs obtained when applying the algorithms to the instances with low standard deviation. We also report the gap obtained with respect to the first of the presented methods, the branch and regret heuristic. Furthermore, for the branch and regret heuristic we report the average running times in seconds, whereas the maximum running time for the ALNS algorithms is set to 10 minutes and for the adapted MSA we generate $N = 30$ samples and a running time each respective ALNS of 1 minute. In this table, we can see how the multi-solution stochastic ALNS provides the best overall results. In general, the branch and regret heuristic provides the worst results for all set of instances. This situation is due to the lack of flexibility that the algorithm considers when dealing with different demand scenarios. It is also remarkable that,

on average, the multi-solution stochastic ALNS solution approach and the deterministic variant of the ALNS and provide the best results (up to 42%, and 40% improvement respectively, for the bigger instance size). Furthermore, the MSA reports good results when solving the instances with bigger size (42%). In the MSA, and multi-solution stochastic ALNS, the methods do not consider one possible solution but a pool of good solutions. This aspect allows the algorithms to, in a last step, evaluate the solutions over a bigger amount of scenarios, and therefore select the best solution of the set of good solutions. On the other hand, the stochastic variant of the ALNS performs worse in general than the before mentioned strategies, as the approach only considers one possible solution after each iteration, and the solution quality can be biased by the samples generated during the algorithm.

Instance	B&R		MSA		Det. ALNS		Sto. ALNS		Multi-sol. ALNS	
	Average	Time(s)	Average	Gap	Average	Gap	Average	Gap	Average	Gap
4p1a	1255.05	0.29	1254.39	1.26%	1151.64	-7.37%	1213.09	-2.44%	1149.06	-7.60%
4p2a	3059.41	1.32	2810.62	-7.83%	2696.37	-11.27%	2694.54	-11.60%	2582.51	-15.36%
4p3a	4713.77	5.16	4264.37	-8.42%	3977.16	-14.76%	4119.15	-10.37%	3727.57	-19.94%
4p4a	6303.25	12.17	4820.35	-21.97%	4653.68	-25.00%	4598.44	-25.95%	4572.75	-26.40%
4p5a	18141.44	239.94	10883.87	-41.96%	11152.55	-40.35%	11294.43	-39.59%	10746.02	-42.89%
Average	6694.58	51.78	4806.74	-15.78%	4726.28	-19.75%	4783.93	-17.99%	4555.58	-22.44%

Table 3.3: Average results four periods with low standard deviation.

In Tables 3.4 and 3.5, we show the results obtained when solving the problem using the proposed algorithms for the given set of instances with medium and high standard deviations. These tables are structured in the same way as the previous one for low standard deviations. The most important outcome of the table is that, when considering higher standard deviations, the MSA lacks on average effectiveness when comparing to the three other proposed algorithms. However, for the bigger group of instances “4p5b” and “4p5c”, the MSA shows a good performance when compared with the stochastic variant of the ALNS, although considering higher standard deviations reduce its effectiveness in average. In general, the multi-solution stochastic ALNS algorithm provides the best overall results for all groups of instances, where the bigger gaps with respect to the other algorithms are found in instances with 15 customers (22% for medium and 23% for high standard deviations).

Instance	B&R		MSA		Det. ALNS		Sto. ALNS		Multi-sol. ALNS	
	Average	Time(s)	Average	Gap	Average	Gap	Average	Gap	Average	Gap
4p1b	1451.12	0.31	1377.93	-2.51%	1256.07	-12.26%	1273.16	-11.13%	1234.57	-13.50%
4p2b	3386.39	1.38	3012.63	-10.66%	2992.16	-10.40%	2909.49	-13.38%	2810.58	-16.70%
4p3b	5284.35	5.43	4561.55	-13.37%	4296.95	-18.37%	4295.75	-18.18%	4079.56	-22.35%
4p4b	6447.04	11.56	5306.21	-16.91%	5101.34	-20.50%	5112.49	-20.10%	5035.60	-21.45%
4p5b	19597.87	243.78	12102.37	-38.08%	11784.76	-39.71%	11959.74	-38.80%	11565.97	-40.83%
Average	7233.35	52.49	5272.14	-16.51%	5086.26	-20.25%	5110.12	-20.32%	4945.26	-22.97%

Table 3.4: Average results four periods with medium standard deviation.

Instance	B&R		MSA		Det. ALNS		Sto. ALNS		Multi-sol. ALNS	
	Average	Time(s)	Average	Gap	Average	Gap	Average	Gap	Average	Gap
4p1c	1677.77	0.28	1562.62	-5.70%	1485.32	-10.16%	1464.093	-11.56%	1427.17	-13.87%
4p2c	3897.37	1.29	3574.81	-7.31%	3397.56	-12.24%	3372.752	-13.18%	3189.23	-17.60%
4p3c	6189.63	5.76	5482.92	-9.94%	4850.30	-20.46%	4952.809	-18.82%	4695.01	-23.18%
4p4c	7383.37	11.83	6349.22	-13.33%	6164.15	-16.08%	6141.952	-16.34%	6002.07	-18.24%
4p5c	21721.26	245.16	14511.58	-32.55%	13031.02	-39.58%	13475.57	-37.71%	12894.12	-40.21%
Average	8173.88	52.86	6296.23	-13.76%	5785.67	-19.71%	5881.44	-19.52%	5641.52	-22.62%

Table 3.5: Average results four periods with high standard deviation.

After we test the proposed and adapted algorithms using the instances with planning horizon of four periods, in Table 3.6, we show the overall results obtained when considering a longer planning horizon. When considering a planning horizon of eight periods, the multi-solution stochastic ALNS reports the best results for all the group of instances except for the set of instances “8p1c” where the deterministic version of the ALNS obtains an improvement of 14.84% with respect to the branch and bound algorithm. However, for instances that consider 10 or more customers, the multi-solution stochastic ALNS outperforms the rest of the presented methods.

It is also noticeable that, when considering a planning horizon of eight periods, the computational times needed to obtain a solution by means of the branch and regret heuristic. For the larger set of instances the computational times needed exceed the 20 minutes, reaching the 28 minutes in average when applying the MSA to solve the group of instances “8p5c”.

Instance	B&R		MSA		Det. ALNS		Sto. ALNS		Multi-sol. ALNS	
	Average	Time(s)	Average	Gap	Average	Gap	Average	Gap	Average	Gap
8p1a	2795.82	1.28	2061.76	-23.44%	1932.66	-29.00%	1862.88	-27.76%	1937.07	-30.29%
8p2a	7538.40	7.33	4875.15	-35.16%	4792.10	-36.02%	4878.13	-34.74%	4808.23	-36.11%
8p3a	10415.01	16.85	6810.15	-33.98%	6479.71	-37.53%	6668.00	-35.63%	6344.31	-38.51%
8p4a	12785.05	42.57	8539.79	-33.16%	7627.60	-38.42%	7647.50	-40.17%	7860.93	-40.32%
8p5a	45385.45	1087.66	18788.95	-58.52%	19391.90	-57.18%	18921.60	-58.22%	17959.70	-60.34%
Average	15783.94	231.14	8215.16	-36.18%	8044.79	-39.63%	7995.62	-39.30%	7782.05	-41.11%
8p1b	2886.39	0.87	2318.01	-19.27%	2277.98	-21.10%	2114.80	-26.36%	2113.05	-26.43%
8p2b	7596.43	7.51	5592.39	-26.42%	5369.33	-29.29%	5389.27	-28.93%	5100.85	-33.10%
8p3b	11944.60	18.53	7798.26	-34.73%	7721.59	-35.17%	7114.50	-40.30%	6735.08	-43.46%
8p4b	14951.90	44.82	9135.76	-38.59%	8834.08	-40.26%	8906.98	-40.22%	8881.67	-40.75%
8p5b	45056.30	1197.42	20355.80	-54.79%	20837.15	-53.73%	20387.10	-54.74%	20051.05	-55.48%
Average	16487.12	253.83	9040.06	-34.76%	9008.02	-35.91%	8782.53	-38.11%	8576.34	-39.84%
8p1c	3002.23	1.33	2893.22	-9.36%	2541.93	-14.84%	2628.28	-12.06%	2579.99	-13.27%
8p2c	8261.15	10.85	7276.61	-11.69%	6790.53	-17.69%	6891.10	-16.48%	5947.69	-27.92%
8p3c	13634.70	0.25	10163.10	-25.18%	8929.28	-34.35%	8815.35	-35.24%	8662.85	-36.43%
8p4c	16114.80	58.29	11501.15	-27.51%	11364.00	-28.79%	11358.85	-28.74%	10946.20	-31.10%
8p5c	55979.15	1218.21	25965.20	-53.57%	23908.45	-57.22%	23387.25	-57.36%	23832.70	-58.17%
Average	19398.41	257.79	11559.86	-24.13%	10706.84	-30.58%	10616.17	-29.98%	10393.88	-33.38%

Table 3.6: Average results eight periods planning horizon.

To a further analysis of the results, in Figure 3.2, we show a boxplot of the percentage gaps of the different methods. In this figure, we show gaps of the sets of instances of bigger size (“4p5a”, “4p5b” and “4p5c”), considering low, medium and high standard deviations. Each of the boxplots represent the gaps obtained when solving all instances of the before mentioned groups using the MSA, and the three variants of the ALNS and compared to the results obtained by means of the branch and regret heuristic. In general, the multi-solution stochastic ALNS obtains the best improvement gaps for all instances in the sets. In the boxplot graphs we can also appreciate how the deviation from the mean remains consistently low when applying the multi-solution stochastic ALNS and, when the standard deviation considered is high, this small deviation from the mean is general for the four methods.

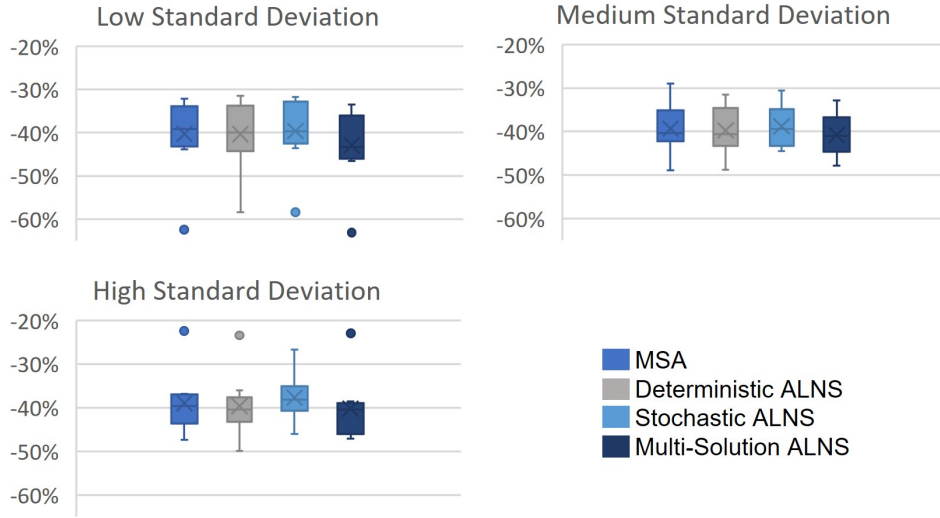


Figure 3.2: Percentage gap representation of group of instances “4p5a”, “4p5b” and “4p5c”

Finally, in Tables 3.7 to 3.10, we report the differences between the different routing, stock-out, inventory and consistency costs, respectively, that we obtained when solving all sets of instances with low, medium and high standard deviations. These tables show that the biggest difference between the obtained results are reported in the stock-out costs in Table 3.8. The multi-solution stochastic ALNS and the MSA provide the best average results and, even when incurring in additional deliveries and, as a result, in higher routing costs as for the sets of instances “4p5a”, “4p5b” and “4p5c” are able to significantly reduce the average expected stock-out costs with respect to the other methods. This is also noticeable in Table 3.9, where the inventory costs reported are reduced with respect to the other methods, except for the branch and regret heuristic. Finally, due to the matheuristic solution approach, and the impact of the time-windows in the problem, we can observe that the requirements of the problem with respect to consistency in the arrival times is satisfied for all solution approaches.

Instances	B&R	MSA		Det. ALNS		Sto. ALNS		Multi-sol. ALNS	
	Average	Average	Gap	Average	Gap	Average	Gap	Average	Gap
4p1a	830.12	933.44	-12%	806.60	3%	791.56	5%	815.06	2%
4p2a	1551.31	1806.09	-16%	1606.84	-4%	1589.93	-2%	1602.55	-3%
4p3a	2014.21	2370.49	-18%	2204.96	-9%	2152.35	-7%	2241.49	-11%
4p4a	2299.05	2924.07	-27%	2644.62	-15%	2642.03	-15%	2635.16	-15%
4p5a	4562.59	5985.21	-31%	5844.97	-28%	5793.93	-27%	5798.59	-27%
4p1b	854.52	1025.00	-20%	804.40	6%	807.90	5%	804.64	6%
4p2b	1533.98	1846.56	-20%	1565.17	-2%	1636.42	-7%	1680.37	-10%
4p3b	2064.11	2432.50	-18%	2151.71	-4%	2203.34	-7%	2280.21	-10%
4p4b	2369.89	3008.57	-27%	2635.16	-11%	2642.23	-11%	2710.26	-14%
4p5b	4416.28	6322.39	-43%	5732.43	-30%	5904.49	-34%	5838.74	-32%
4p1c	811.63	920.30	-13%	812.39	0%	819.32	-1%	852.69	-5%
4p2c	1507.52	2020.73	-34%	1612.87	-7%	1480.87	2%	1803.02	-20%
4p3c	2096.72	2581.22	-23%	2124.44	-1%	2247.05	-7%	2358.64	-12%
4p4c	2513.45	2977.09	-18%	2638.95	-5%	2649.48	-5%	2807.64	-12%
4p5c	4387.61	5805.74	-32%	5885.48	-34%	5598.94	-28%	5957.31	-36%

Table 3.7: Average routing costs.

	B&R	MSA		Det. ALNS		Sto. ALNS		Multi-sol. ALNS	
Instances	Average	Average	Gap	Average	Gap	Average	Gap	Average	Gap
4p1a	180.04	74.63	59%	83.14	54%	118.68	34%	81.27	55%
4p2a	646.67	165.43	74%	261.30	60%	266.00	59%	173.12	73%
4p3a	1339.91	437.36	67%	453.45	66%	532.37	60%	278.29	79%
4p4a	2393.66	292.94	88%	439.15	82%	423.01	82%	394.43	84%
4p5a	9150.36	571.55	94%	1045.04	89%	1169.12	87%	655.08	93%
4p1b	318.66	101.59	68%	195.00	39%	210.48	34%	168.53	47%
4p2b	1014.12	326.20	68%	514.54	49%	459.99	55%	328.98	68%
4p3b	1805.70	642.63	64%	730.67	60%	655.36	64%	551.21	69%
4p4b	2422.24	649.80	73%	870.02	64%	874.13	64%	725.32	70%
4p5b	10696.69	1359.12	87%	1666.17	84%	1633.13	85%	1371.14	87%
4p1c	591.13	312.14	47%	370.31	37%	360.06	39%	289.08	51%
4p2c	1435.46	664.75	54%	896.36	38%	942.14	34%	524.82	63%
4p3c	2519.91	1293.42	49%	1229.65	51%	1249.89	50%	976.88	61%
4p4c	3059.20	1566.34	49%	1791.93	41%	1743.15	43%	1426.12	53%
4p5c	12487.11	4155.64	67%	2681.68	79%	3375.57	73%	2536.47	80%

Table 3.8: Average stock-out costs.

Instances	B&R	MSA		Det. ALNS		Sto. ALNS		Multi-sol. ALNS	
	Average	Average	Gap	Average	Gap	Average	Gap	Average	Gap
4p1a	231.98	238.34	-3%	252.44	-9%	294.12	-27%	243.26	-5%
4p2a	825.35	793.47	4%	794.43	4%	801.30	3%	771.31	7%
4p3a	1298.77	1390.32	-7%	1268.86	2%	1396.22	-8%	1145.23	12%
4p4a	1525.72	1512.93	1%	1506.79	1%	1472.21	4%	1483.81	3%
5a	4166.47	3989.09	4%	3984.08	4%	4021.36	3%	3958.95	5%
4p1b	265.39	243.05	8%	247.86	7%	243.65	8%	247.12	7%
4p2b	796.36	796.10	0%	872.02	-10%	781.72	2%	756.85	5%
4p3b	1362.32	1413.18	-4%	1363.08	0%	1385.52	-2%	1192.85	12%
4p4b	1571.25	1536.86	2%	1529.06	3%	1528.89	3%	1502.10	4%
4p5b	4230.15	4071.21	4%	4088.38	3%	4125.57	2%	4013.92	5%
4p1c	261.15	322.47	-23%	289.48	-11%	274.66	-5%	271.83	-4%
4p2c	904.67	839.51	7%	852.26	6%	920.79	-2%	813.12	10%
4p3c	1512.51	1531.72	-1%	1450.89	4%	1399.44	7%	1293.19	15%
4p4c	1728.53	1695.58	2%	1663.39	4%	1683.35	3%	1657.44	4%
4p5c	4601.08	4208.46	9%	4161.42	10%	4188.98	9%	4118.07	10%

Table 3.9: Average inventory costs.

Instances	B&R	MSA		Det. ALNS		Sto. ALNS		Multi-sol. ALNS	
	Average	Average	Gap	Average	Gap	Average	Gap	Average	Gap
4p1a	12.91	7.98	38%	9.46	27%	8.73	32%	9.47	27%
4p2a	36.08	45.63	-26%	33.79	6%	37.32	-3%	35.53	2%
4p3a	60.88	66.17	-9%	49.89	18%	38.21	37%	62.56	-3%
4p4a	84.81	90.55	-7%	63.12	26%	61.19	28%	59.35	30%
4p5a	262.03	338.03	-29%	278.47	-6%	310.02	-18%	333.40	-27%
4p1b	12.55	8.30	34%	8.81	30%	11.13	11%	14.28	-14%
4p2b	41.93	43.77	-4%	40.43	4%	31.37	25%	44.39	-6%
4p3b	52.22	73.24	-40%	51.49	1%	51.53	1%	55.29	-6%
4p4b	83.65	110.98	-33%	67.10	20%	67.23	20%	97.91	-17%
4p5b	254.74	349.66	-37%	297.80	-17%	296.55	-16%	342.16	-34%
4p1c	13.86	7.70	44%	13.14	5%	10.05	27%	13.57	2%
4p2c	49.73	49.83	0%	36.07	27%	28.95	42%	48.26	3%
4p3c	60.49	76.55	-27%	45.32	25%	56.43	7%	66.29	-10%
4p4c	82.20	110.21	-34%	69.89	15%	65.97	20%	110.86	-35%
4p5c	245.44	341.73	-39%	302.45	-23%	312.08	-27%	282.27	-15%

Table 3.10: Average consistency costs.

3.5.4 Recourse strategies

After we evaluate the effectiveness of the proposed algorithm, we evaluate the impact of including two recourse strategies to reduce the sales lost and the inventory related costs, in addition to the penalization of lost sales in the objective function. In Table 3.11, we show the costs variation when we create additional delivery routes to replenish customers that are expected to run out of stock as described in section 3.4.3. In this table, columns named “Route”, show the average increase in routing costs when considering low, medium and high standard deviations. Furthermore, columns named “Inventory” and “Stock”, show the increase in inventory and stock-out costs respectively. Finally, columns named “Difference”, show the average increase in overall

costs when applying the recourse action to the obtained solution by means of the multi-solution stochastic ALNS.

In Table 3.12, we proceed in the same manner as in Table 3.11. However, in this case, we first try to insert customers that may experience a loss of sales in already existing routes and, if there are additional deliveries that cannot be inserted in existing delivery routes, we create new delivery routes.

Instance size	Low Standard Deviation				Medium Standard Deviation				High Standard Deviation			
	Route	Inventory	Stock	Difference	Route	Inventory	Stock	Difference	Route	Inventory	Stock	Difference
4p1	41.31	-3.94	-7.85	29.51	56.55	-5.53	-16.93	34.09	62.68	-12.93	-26.20	23.55
4p2	142.57	-45.59	-19.81	77.17	87.91	-21.29	-36.02	30.60	140.43	-55.08	-103.06	-17.70
4p3	263.25	-165.88	-34.51	62.86	309.90	-196.36	-124.47	-10.93	352.70	-229.84	-244.88	-122.01
4p4	174.61	-63.08	-45.43	66.10	216.46	-91.20	-88.96	36.29	319.96	-163.52	-316.66	-160.22
4p5	287.20	-95.36	-41.35	150.49	395.65	-136.75	-236.01	22.89	731.76	-359.92	-955.99	-584.15

Table 3.11: Impact additional delivery routes. Difference = Routing costs - Inventory costs - Stock-out costs

Instance size	Low Standard Deviation				Medium Standard Deviation				High Standard Deviation			
	Route	Inventory	Stock	Difference	Route	Inventory	Stock	Difference	Route	Inventory	Stock	Difference
4p1	41.58	-4.07	-7.81	29.69	55.94	-5.40	-16.25	34.29	63.05	-14.76	-27.06	21.23
4p2	142.55	-45.78	-18.78	77.98	87.65	-22.05	-36.12	29.48	139.32	-56.79	-97.97	-15.43
4p3	264.41	-167.14	-34.61	62.66	309.66	-198.03	-124.31	-12.67	354.59	-233.80	-253.74	-132.95
4p4	174.92	-62.20	-46.99	65.72	218.28	-90.99	-91.90	35.39	321.61	-165.68	-314.97	-159.03
4p5	286.14	-94.84	-40.24	151.05	398.92	-138.55	-232.92	27.45	727.08	-359.74	-948.53	-581.19

Table 3.12: Impact deliveries in existing routes and additional routes

In these tables we observe that, when the demands that the customers experienced are predictable (low and medium standard deviations), the recourse actions help reduce the stock-out and inventory related costs. However, this reduction in costs is smaller than the increase in routing costs originated by creating new delivery routes and/or inserting customers in already existing routes. On the other hand, when the standard deviation considered for the stochastic distribution of the demands is high, the overall costs are reduced as the instance size considered increases. This results are consistent with real world situations where, when the demands that customers experience are highly stochastic, additional actions within the period horizon would

help the central supplier to minimize the overall expected costs. However, when the demand stochasticity is relatively low, additional deliveries would increase the original delivery plan.

3.6 Conclusions and further research

In this article, we introduce the CTSIRPTW, which is part of the family of IRPs. The characteristics of the problem we present arise from a real-world application related to beverage industries that require a great effort, especially when dealing with the related route planning and inventory management. The problem includes one recent characteristic in the literature, that is, the continuous-time commodity consumption. In this group of problems, the product usage at each customer is continuous within the periods, and creating bad delivery plans may impact in additional costs related to stock-out situations. Furthermore, customers have different opening times and time-windows, and consistency in delivery times improves service quality and customer satisfaction, which is a key factor in a competitive market. We present a two-stage mathematical formulation for the CTSIRPTW. We formulate the problem as a MIP, wherein, the first step involves calculating delivery times, sequences and quantities, and the second step involves calculating the related inventory stock-out costs after the demands at the customers are disclosed. In practice, in addition to the proposed algorithm that provides a route and delivery plan for the planning horizon, companies that expect a shortage in their stock may need an additional delivery. We present and evaluate the impact of recourse actions to modify the proposed delivery plans.

To solve the proposed mathematical formulation, in Section 3.4, we propose five different solution approaches. The first method is adapted from the branch and regret heuristic, first introduced in [50], where at each period, an iterative procedure is applied to determine the impact of delivering the customers in the current or subsequent periods. The second solution approach is an adaptation of the MSA present in [81]. In this approach, a pool of possible solutions is taken into account at each period, and is extended using both past and future demand scenarios. Finally, we propose three versions of a matheuristic solution approach, based on an ALNS that includes several destroy and repair operators applied to improve an initial solution, obtained using the cheapest insertion algorithm. The three versions vary in terms of the usage of the stochastic information and using different amounts of demand samples and good solutions within the algorithm. Furthermore, during the process of obtaining good solutions, we solve an exact

subproblem up to optimality using good solutions. We include the solution of this mathematical subproblem in all the proposed methods to obtain optimal delivery amounts and times, using the given routes.

To test the efficiency of the proposed method, we present a range of computational tests with a benchmark set of instances, and with a new instance set adapted from a benchmark set and presented in [5]. We adapted the set of instances to include different levels of stochasticity. Results show that the multi-solution stochastic ALNS reports the best overall solutions for all instance sizes, where, for the largest instance size, it finds solutions at 42%, 40% and 40% better than the branch and regret heuristic, when considering low, medium and high standard deviations, respectively. We report the results obtained from solving instances and the costs related to routing, stock-out, inventory and consistency. In these results, the multi-solution stochastic ALNS drastically reduces the expected stock-out cost of the solution, even if, as in sets of instances “4p5a” and “4p5b”, the central warehouse incurred higher routing costs. These results show that additional deliveries are of great use when minimizing the overall costs of a supply chain. In addition, we report results obtained when considering a longer planning horizon of eight subperiods, where the results obtained remain consistent as the instance size increases.

The continuous-time IRP problem is a new variant in the literature. In this field, research could take different directions. Model-specific research might study other aspects of the problem, such as the possibility of a multi-echelon distribution systems in which a central retailer supplies different intermediate storage centers and customers are replenished from these storage centers. Another model-specific variant could include the multiple-product situation, wherein the central supplier manages more than one product and the coordination of deliveries to each customer is a key aspect of the problem. It may also be interesting to consider the dynamic stochastic variant of the problem; at the beginning of each period, the central warehouse can re-optimize the current solution and include new information disclosed in previous periods. On the other hand, several other studies include stochastic travel times, increasing the level of stochasticity and the complexity of the problem. Another research possibility would be to develop more efficient solution methods. Finally, other metaheuristic solution approaches, and problem specific methodologies, can be applied to find solutions that would reduce the total expected costs.

Appendix A

In Algorithms 6 and 7 we provide the pseudo codes for the inner and outer loop of the branch and regret procedure in the original work.

Algorithm 6 Outer loop branch and regret algorithm [50]

1. At the start of each interval I_1, \dots, I_v :
 - (a) Take the solution from the previous interval and lock the plan up to the moment at which the current time interval begins. If a vehicle has left its last customer, a lock is set to indicate that it must continue to the customer that followed in the solution from the previous interval.
 - (b) Add to the problem all customer requests that became known during the last interval.
 - (c) Find plans (for the current interval I_u) by using the BRH subprocedure.
 2. Evaluate the final solution.
-

• •

Algorithm 7 Inner loop branch and regret algorithm [50]

Let C be the set of known customers at the start of an interval, I_u , and let S_{in} and S_{out} , be subsets of C , both initially empty.

1. Create r sample scenarios with the same known customers as in C , including stochastic customers drawn from the given distributions.
 2. Repeat the following:
 - (a) Solve the sample scenarios, while forcing all customers in S_{in} to be served during I_u and all customers in S_{out} to be served later than I_u .
 - (b) Find the customer $c \in C \setminus (S_{in} \cup S_{out})$ that is most frequently visited in the time interval I_u in the r sample scenarios.
 - (c) Resolve the sample scenarios (as in step 2a), considering both $S_{in} := S_{in} \cup c$ and $S_{out} := S_{out} \cup c$. Add c permanently to whichever set that is evaluated as the best choice.
 3. Repeat until S_{in} is empty:
 - (a) Solve the sample scenarios (as in step 2a).
 - (b) Count the number of times each customer c in S_{in} is served first by each vehicle in each scenario solution, disregarding customers that have already been placed.
 - (c) Consider the customer-vehicle pair with the highest count, and potentially a second pair where either the customer or the vehicle is the same as for the pair with highest count. For both alternatives: lock the customer to be served before all remaining customers by the given vehicle, and resolve the sample scenarios. Enforce the alternative that gives the least average cost over all sample scenario solutions, and, for any remaining iterations, disregard the enforced customer in step 3b.
 4. Terminate with a plan for the given time interval I_u .
-

Stochastic inventory routing with dynamic demands and intra-day depletion

Submitted to: *Computers & Operations Research*

under revision, October 31st 2023.

E.J. Alarcon Ortega, S. Malicki, K.F. Doerner & S. Minner.

Abstract Businesses such as supermarkets face an inventory routing problem with distinct stochastic dynamic demand distributions for various weekdays and times of the day. Most research, however, considers aggregated per-period demand and inventory depletion, thereby oversimplifying actual events. We formulate the problem as a finite-horizon stochastic dynamic program that accounts for the dynamic demand throughout the planning horizon and within each planning period. The models include different levels of information aggregation to decide the replenishment quantities and times by dividing periods into sub-periods with distinct demand distributions. This division helps the supplier create efficient replenishment plans and account for intra-day inventory levels. The modeled characteristics include routing, holding and stock-out costs, delivery time windows, inventory and vehicle capacities, and delivery time windows. The problem is solved using an iterative lookahead algorithm with an adaptive large neighborhood search for the routing and policy learning for the replenishment decisions. Numerical evaluations show the benefit of intra-day planning, effects of instance sizes, different coefficients of variation, and holding costs compared to benchmark replenishment policies. Considering intra-day depletion and demand data for each sub-period helps achieve savings of more than

20% compared to planning for full periods. The savings are obtained by adjusting delivery quantities, incorporating intra-day consumption, creating more efficient delivery routes, and correctly timing replenishments.

4.1 Introduction

The IRP resembles the VMI approach, in which a supplier manages customer replenishment and distribution. Applying VMI can substantially reduce overall logistics costs ([62]). Companies must decide the quantities each store receives during each period and create efficient delivery routes. Lot-sizing and vehicle routing are solved in an integrated manner rather than sequentially. This integration enables the utilization of synergies by correctly synchronizing customer replenishments and obtaining savings by synchronizing the delivery schedules of proximate customers or those who can be combined in favorable routes regarding time windows.

We present a new IRP variant that focuses on the stochastic and dynamic nature of demands that businesses face in real-world applications. In the real world, demands differ on both a daily and intra-day basis. Late deliveries can result in lost sales not only at the end of periods but also within them. Moreover, customers decide the time of the day, or the time range, when they want delivery. Most research aims to find delivery schedules that minimize routing and customer holding costs over a given planning horizon. However, the problems only partially represent customers' continuous inventory depletion, neglecting intra-period demand dynamics. The Stochastic Inventory Routing Problem with Intra-Day Depletion under Dynamic Demands (SIRPID) arises in the context of beverage, food, and retailer industries, in which companies must develop efficient delivery plans for customers with different delivery time windows, inventory capacities, (intra-day) demand patterns, and the number of other close-by locations. For this problem, the supplier must create delivery plans for a finite planning horizon. However, unlike in other applications, the demands that each customer experiences do not only occur at the end of each period. Each period during the planning horizon is divided into different sub-periods with independent stochastic demands. This division of periods into sub-periods is derived from many real-life applications where the demands experienced during the periods of the planning horizon are not consumed linearly and are subject to stochasticity.

The supplier develops replenishment and routing schedules to mitigate these stochasticity effects. Customers provide the supplier with information about their inventory levels, and the supplier decides the delivery quantities and routes for the upcoming periods. These demand dynamics are accounted for and incorporated into the holding, stockout, and routing cost calculations. Another characteristic of IRPs is that customers have different opening hours. Customers present different time windows in which they can receive replenishment, which is particularly important for catering businesses such as bars, restaurants, and retailers.

The contribution is threefold: (1) We formulate the SIRPID as a finite-horizon stochastic dynamic program (SDP) that minimizes the sum of routing costs, customer inventory holding costs, and lost sales costs over multiple sub-periods. (2) We develop solution methods by applying different replenishment policies and considering a different granularity of demand data. We present sequential decision algorithms, an OU policy, chance-constrained (CC) policies with different run-out times, a policy function approximation algorithm for independent retailers, and an integrated approach, the adaptive lookahead algorithm. (3) We conduct a computational study comparing the performance of the solution methods. We derive managerial insights into the benefit of intra-day demand data and compare the effects of instance sizes, different coefficients of variation and holding costs, and the number of available vehicles on holding, routing, stockout, and total costs.

The remainder of this paper is structured as follows: Section 4.2 presents a literature review, emphasizing works that present demand stochasticity, and briefly summarizes different applications and solution approaches. In Section 4.3, we formulate the SIRPID as a finite SDP and describe the VRP that is solved after the decision regarding which customers to replenish. Due to the high complexity of the problem and the curses of dimensionality, solving the SIRPID optimally is impossible. Therefore, Section 4.4 proposes different approximations to solve instances of different sizes. The solution approaches vary in levels of sophistication and efficiency. In Section 4.5, we perform computational experiments to evaluate the performance of the different methods. In Section 4.6, we summarize the contributions and provide concluding remarks.

4.2 Literature Review

In the seminal paper on IRP, [18] study the distribution of industrial gases in a VMI supply environment. Dror and Ball [41] show that an integrated planning of lot-sizing and vehicle routing

produces better results than planning consecutively. Furthermore, for an extensive literature review related to the IRP, we refer to Coelho et al. [34], Cordeau et al. [37], Andersson et al. [9], and Bertazzi et al. [23].

4.2.1 Deterministic IRP

In this subsection, we discuss a collection of IRP problems in a deterministic context that shares some of the most important characteristics with the problem we introduce. Specifically, we first describe the works in the literature, which include continuous inventory depletion. This feature more accurately represents the real-world nature of inventory consumption by customers. Then, several proposed strategies to solve IRP with different characteristics are discussed.

Intra-day depletion is common in inventory management. However, it is a relatively new characteristic in VRP and IRP literature. Alarcon et al. [5] present a deterministic continuous-time IRP with a multi-period planning horizon, stockouts, and time windows in which a supplier must create efficient inventory and routing plans to reduce overall costs. They use a matheuristic in which an ALNS algorithm is combined with a mathematical subproblem to obtain optimal replenishment quantities and times using fixed routes. In contrast to that work, the demands in the paper in our research are stochastic, and distinct demand distributions for multiple sub-periods that divide the periods are used. In addition, the ROADEF/EURO motivated a collection of approaches applied to the Air Liquide IRP, which deals with the distribution of bulk gases and proposes different heuristics to solve the deterministic IRP with time windows (Andre et al. [10], Kheiri [54], Absi et al. [2], Su et al. [82], He et al. [46]). However, customers face demands at the end of each period and do not experience stockouts.

In the literature, we can find various strategies to solve IRP. Anily and Federgruen [11] and Campbell and Savelsbergh [28] propose clustering heuristics for a multiple customer problem and an IRP, respectively. In these approaches, the problem is first decomposed into different replenishment problems. Then, whenever a customer in a cluster needs replenishment, all the customers in the cluster are replenished. Another solution strategy is presented by Campbell et al. [27], who propose a two-phase heuristic, which first determines delivery quantities and then finds cost-minimal routes for the replenishments. Bertazzi et al. [22] introduce a local search algorithm with an OU level policy applied to all but the last customer on each delivery route. The quantity delivered to the last customer is the minimum between the OU quantity and the residual transportation capacity of the vehicle.

4.2.2 Stochastic IRP

A wide variety of articles on different Stochastic Inventory Routing Problems (SIRPs) can be found in the literature. Minkoff [66] formulates the SIRP as an infinite horizon Markov Decision Process (MDP) and proposes a heuristic to reduce its complexity. In their MDP formulation, Kleywegt et al. [55] allow only direct deliveries. Kleywegt et al. [56] extend this formulation by allowing a maximum of three customer replenishments per route. On the contrary, Adelman [3] limits the route duration instead.

The literature on approximate dynamic programming methods that deal with different variants of the Stochastic VRP and IRP has grown significantly. Pavone et al. [67] present a policy function approximation algorithm that uses pre-defined policies to decide about the routing for an SVRP. Various cost function approximations are given in [73] and Ulmer et al. [84]. Scenario-based approaches are introduced in Schilde et al. [77] and Ferruci et al. [44]. An extended heuristic for the VRP to account for holding and shortage costs is proposed by Federgruen and Zipkin [43]. For a further review of approximate dynamic programs related to the SVRP, we refer to the literature review presented in Soeffker et al. [78].

With respect to SIRP, several approaches have been proposed to solve it, in which demand realizations occur instantaneously at the end (or the beginning) of each period. Campbell et al. [27] propose a dynamic program and do not account for holding costs to reduce the complexity of the model. Hvattum et al. [51] use truncated scenario trees and determine a critical length of a lookahead horizon, whereas Liu and Lee [63] propose a combined variable neighborhood search and tabu search to solve the SIRP. Jaillet et al. [52] use a rolling-horizon approach and derive cyclic solutions. More recently, Raba et al. [72] present a reactive simheuristic for a real-life SIRP applied to the animal-food industry. Alvarez et al. [8] present a two-stage stochastic program and analyze several recourse mechanisms to deal with an IRP under stochastic supply and demand. Brinkmann and Mattfeld [26] and Brinkmann et al. [25] present two different approximate dynamic programs to deal with SIRP in the context of bike-sharing systems that consist of a coordinated lookahead and a dynamic lookahead policy, respectively.

In the context of IRPs that consider demands to occur not only at the end of the periods during the planning horizon but also within, Lagos et al. [60] and Alarcon Ortega and Doerner [7] present two mathematical formulations to deal with the characteristics of respective problems, and broad computational analysis that compare the effectiveness of different solution approaches. In the first article, a continuous-time IRP, in which a product is consumed continuously throughout the

periods, is introduced. This article deals with the problem of replenishing a set of customers over a finite planning horizon. They propose a dynamic discretization discovery algorithm to decide the part of the period to discretize and solve the problem in a small sequence of integer programs. However, unlike in our work, stock-out situations are not considered, and the replenishment plan considers a single period. This problem is extended by Lagos et al. [61], who minimize the cost of a delivery plan that allows out-and-back routes. The presented method consists of a dynamic discovery discretization algorithm using partially constructed time-expanded networks. The second article, a stochastic IRP in which continuous-time commodity consumption on the customers' premises is considered. This paper presents a two-stage stochastic formulation for a multi-period IRP in which stockouts can occur within the periods. They propose a sampling-based matheuristic based on ALNS to solve the problem and compare the performance of this approach with respect to other methods.

Bertazzi et al. [21] and Zhao et al. [86] propose and evaluate different replenishment policies for the stochastic IRP: an OU policy under which the customers' inventories are filled to their full inventory capacity each time their inventories are below a reorder point, and a combination of fixed partition policies and power-of-two policies to solve a three-echelon IRP.

The more recent literature further investigates other applications and characteristics of the stochastic IRP. Bertazzi et al. [20] propose a hybrid roll-out algorithm with stock-out penalties. Crama et al. [38] solve the stochastic IRP for perishable products and compare the solutions regarding average profit, product freshness, and inventory service levels. Gruler et al. [45] present a variable neighborhood search simheuristic. The problem is solved using a fixed delivery plan to minimize the expected overall costs. Malicki and Minner [64] propose a stochastic cyclic IRP with dynamic safety stocks for auto-regressive demands. Almost all the mentioned papers use approximations to overcome the curses of dimensionality.

In the inventory management literature, the IRP is formulated as a joint replenishment problem, where the major setup costs depend on the routing solution(s). A formulation of the IRP as a nested joint replenishment problem is presented in Viswanathan and Mathur [85]. Federgruen and Zheng [42] solve the joint replenishment problem with monotone and concave joint order costs, reflecting economies of scale when replenishing products (customers) jointly. Herer and Roundy [49] compute optimal power-of-two policies, which yield only slightly higher cost than the optimal policy does. Periodic policies for a multi-item stochastic joint replenishment problem with transportation are provided in Qu et al. [71]. Çetinkaya and Lee [31] solve an IRP in which

the supplier can hold back small orders for a specific dispatch time. Axsäter [16] improves their approximation and shows an exact solution procedure.

This paper addresses a gap in the literature by incorporating intra-day inventory depletion into an IRP by splitting planning periods into multiple sub-periods. Decisions are made every periods, while replenishments and routing under time window restrictions are planned for every period's sub-periods. By doing so, it is possible to incorporate information on the intra-day demand patterns as the sub-periods feature distinct demand distributions. Figure 4.1 shows daily sales data of 78 customers discretized to 5 sub-periods per day in a heat map. Dark colors indicate a high sales volume for a particular customer and a specific sub-period, whereas light colors indicate low sales. It can be seen how, for each retailer, demand peaks and lows vary. When planning replenishment quantities and routes for those customers, a division into sub-periods introduces the necessity to consider replenishment timings inside of a period. However, the decision on when to time the replenishments is interconnected with planning the replenishment quantities and routing sequences as they mutually affect each other.

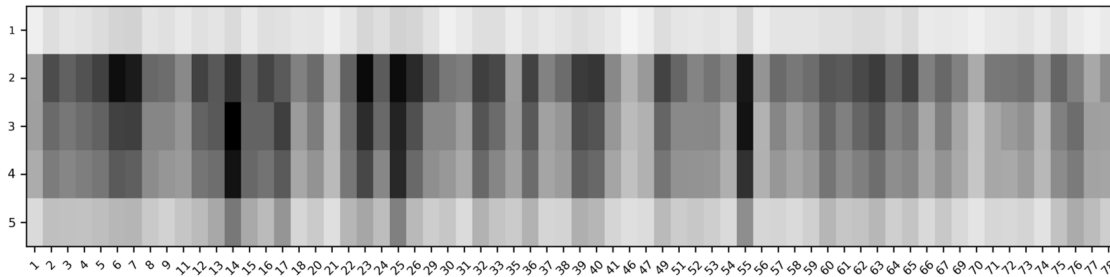


Figure 4.1: Daily sales data divided into five sub-periods

The division of periods into sub-periods makes it possible to account for different demand distributions across periods and sub-periods, as demands are stochastic and differ across customers, day of the week, time of the day, and many other factors. For this reason, we discretize available demand data into sub-periods, rather than using dynamic discretization as shown by Lagos et al. [61], who assume linear and deterministic consumption throughout the planning period. Leveraging this more detailed data enables significant savings by correctly determined replenishment times and amounts compared to a period-based planning, which considers aggregated per-period demand data and neglects intra-day consumption.

4.3 Optimization Model

We formulate the SIRPID as an SDP. We consider the distribution of a single product from one supplier (depot) to a set of customers $i \in N = \{1, \dots, |N|\}$ over a finite set of discrete periods $t \in T = \{1, \dots, |T|\}$, which are further divided into a set of sub-periods $u \in U = \{1, \dots, |U|\}$. I_{itu} denotes the inventory at each customer at time (t, u) , i.e., at the beginning of period t and sub-period u . If, for any parameter introduced with indices i , t , and u , one of the indices is dropped, we refer to the vector or matrix over the dropped index/indices. Each customer has an inventory capacity of \bar{I}_i , $i \in N$. Demands $D_{tu} = \{D_{1tu}, \dots, D_{|N|tu}\}$ are stochastic and follow the known independent discrete probability distributions $_{itu}$. Inventories are reviewed at the beginning of (i.e. before) every period t . Based on the inventory level at the beginning of each period and the information on the customers' demand distributions in the upcoming sub-periods, replenishment decisions are made at the beginning of each period t and cannot be altered during the period's sub-periods. Replenishments are delivered using a homogeneous fleet of $|V|$ vehicles with capacity Q each. As all unsatisfied demand is assumed to be lost, the state space is $I_{tu} \in \{0, \dots, \bar{I}_1\} \times \{0, \dots, \bar{I}_2\} \times \dots \times \{0, \dots, \bar{I}_{|N|}\}$.

The quantity delivered to a customer in a sub-period is denoted by q_{itu} , where $(tu)^-$ refers to the sub-period prior to sub-period u in period t . At the beginning of each period t , $\mathcal{A}(I_t)$ denotes the set of feasible actions. Action $q_{tu} = \{q_{1tu}, \dots, q_{|N|tu}\}$ defines the quantities delivered to each of the customers in period t , sub-period u . Replenishments are possible only during a customer's time window $\{e_{it}, l_{it}\}$, with $e_{it}, l_{it} \in U \wedge e_{it} \leq l_{it}$ and have to be considered in the routing sequence. Replenishments can therefore happen only if sub-period e_{it} has begun and as long as sub-period l_{it} has not yet ended. Furthermore, we assume that deliveries cannot be split among the vehicles. Therefore, for the sub-periods outside of the time window, the customers must not be replenished, i.e., $\sum_{u \in U \setminus \{e_{it}, \dots, l_{it}\}} \mathbb{1}(q_{itu} > 0) = 0 \forall i \in N, t \in T$, whereas for the sub-periods inside of their time window, they cannot be visited more than once, i.e.,

$\sum_{u \in \{e_{it}, \dots, l_{it}\}} \mathbb{1}(q_{itu} > 0) \leq 1 \forall i \in N, t \in T$, with the indicator function $\mathbb{1}()$ returning 1 if the argument is true, and 0 otherwise. Hence, the set of feasible actions $\mathcal{A}(I_t)$ depends on the customers' inventory capacities \bar{I}_i , the available vehicle capacity, and the customers' time windows.

The inventory capacity imposes a bound on the quantity q_{itu} delivered to each customer because $I_{i(tu)^-} + q_{itu} \leq \bar{I}_i$ has to hold for all customers and sub-periods. The set of possible actions is given by $q_t \in \mathcal{A}(I_t) = \{q_{1t1}, \dots, q_{1t|U|}\} \times \dots \times \{q_{|N|t1}, \dots, q_{|N|t|U|}\} \forall t \in T$, with $\sum_{i \in N} \sum_{u \in U} q_{itu} \leq |V|Q$,

$q_{tu} \in \{0, \dots, \bar{I}_1 - I_{1(tu)^-}\} \times \dots \times \{0, \dots, \bar{I}_{|N|} - I_{|N|(tu)^-}\}$, $\sum_{u \in U \setminus \{e_{it}, \dots, l_{it}\}} \mathbb{1}(q_{itu} > 0) = 0 \ \forall i \in N, t \in T$, and $\sum_{u \in \{e_{it}, \dots, l_{it}\}} \mathbb{1}(q_{itu} > 0) \leq 1 \ \forall i \in N, t \in T$. The inventory level at the end of every sub-period is given by $I_{itu} = \max\{I_{i(tu)^-} + q_{itu} - D_{itu}, 0\}$. Additionally, initial inventory levels I_{i0} for all customers are known. With $_{itu}(D_{itu})$ for the probability of the demand occurrence D_{itu} , h_i for customer i 's inventory holding costs per unit and period, and ϕ_i denoting the lost sales penalty customer i faces per unit short, the expected inventory costs for period t depending on the action q_t are given by:

$$C_I(I_t, q_t) = \sum_{i \in N} \sum_{u \in U} \left(\sum_{D_{itu}=0}^{I_{i(tu)^-} + q_{itu}} \frac{h_i}{|U|} (I_{i(tu)^-} + q_{itu} - D_{itu})_{itu}(D_{itu}) + \sum_{D_{itu}=I_{i(tu)^-} + q_{itu} + 1}^{\infty} \frac{\phi_i}{|U|} (D_{itu} - (I_{i(tu)^-} + q_{itu}))_{itu}(D_{itu}) \right), \quad (4.1)$$

Routing, replenishment quantities, and sub-periods of replenishment are planned at the beginning of each period t . Once planned, they cannot be altered throughout the period (day of operation). The routing costs for a given action in period t , $C_R(q_t)$, depend on the delivery quantities q_t and are calculated by solving a capacitated VRP with time windows with the given quantities.

Hence, using the Bellman equation, the SDP can be formulated as follows:

$$V^*(I_t) = \min_{q_t \in A(I_t)} \left\{ C_I(I_t, q_t) + C_R(q_t) + \mathbb{E} \left[\sum_{t'=t+1}^{|T|} (C_I(I_{t'}, q_{t'}) + C_R(q_{t'})) \right] \right\}. \quad (4.2)$$

We now describe the problem using the terminology and approach in Soeffker et al. [78]. Figure 4.2 shows a graphical representation of the information model and decision model of the underlying problem, assuming $|U| = 5$. The information model consists of the given demand distributions for each sub-period. The demand realizations of each sub-period inside of a period become known at the end of that period. The decision model consists of decision points occurring every full period, i.e., every $|U|$ sub-periods. Decisions are made considering the system's state, given by the customers' inventory levels I_t at the end of period t , and the information on the demand distributions of the upcoming sub-periods. The decisions are: which customers to replenish, how

much to replenish, in which sub-period to replenish, and how to sequence the replenishments in replenishment routes such that holding costs, routing costs, and stockout costs are minimized.

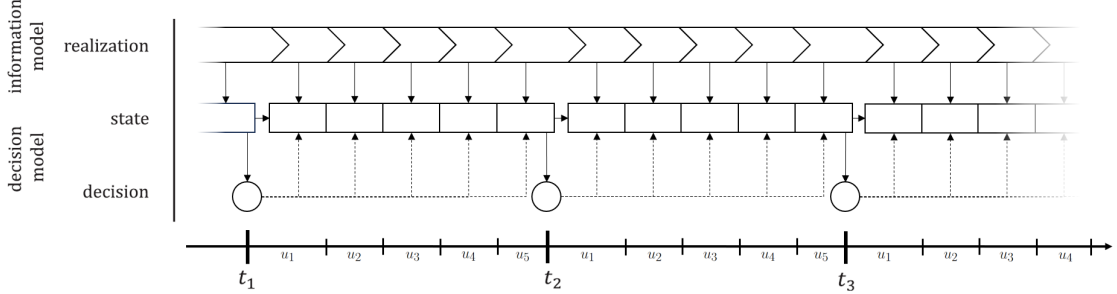


Figure 4.2: Stochastic dynamic decision process of the SIRPID, adapted from Meisel [65]

The pre-decision state of a period t is given by the ending inventory of the preceding period $t - 1$, i.e., sub-period $(t - 1, |U|)$. Based on the information about current inventories (state) and the future sub-periods' demand distributions, a decision on the replenishment quantities and routing sequence for the upcoming period t , i.e., sub-periods $(t, 1)$ up to $(t, |U|)$, has to be made. Once decided, replenishment quantities and routing sequence cannot be altered throughout that period and are assumed to arrive on time in the planned sub-period. The demand realizations are subtracted from the customers' inventories at the end of each sub-period, as introduced in the inventory balance equations above. Unmet demand is lost. The cost evaluation is performed at the sub-period level. While we assume that demands are not correlated over time, an adaptation of the replenishment quantities to account for inter-period demand correlation can be performed as shown in Malicki and Minner [64].

4.4 Solution Methods

Due to the curses of dimensionality, solving the full SDP is not tractable. Hence, in this section, we describe different heuristics based on approximations for the SIRPID.

4.4.1 Approximate Inventory Policies

The first two approximate policies we present in this article are OU and Chance Constrained for τ periods ($CC\tau$). These approximate policies use an intuitive way to reduce the problem's complexity. They use a sequential decision structure in which, at each period, the decision on the

customers to be replenished and the amount to be delivered is made, and after that, a routing algorithm is applied. The methods consider expected demands and apply retailer-managed inventories, i.e., deciding on the lot sizes independently for each customer and designing the routes for the replenishments afterward. In $(CC\tau)$, customers maintain safety stock levels by setting a target non-stockout-probability α .

Once all customers' replenishment quantities and sub-periods of replenishment are known, we design the delivery routes for the replenishments using an ALNS for each period. In this approach, the sub-period of delivery is determined when solving the VRP of the current period. Note that the amount actually restocked may be less than planned, depending on the extent of the demand. If, for instance, a planned replenishment exceeds the inventory capacity because the demand realizations in a period were smaller than expected, the replenishment quantity delivered to the customer is reduced to the amount necessary to completely fill the inventory. Obviously, once decisions on delivery quantities for a period are made, they cannot be increased throughout the period anymore.

Order-Up-To

Replenishment quantities are set to fill the customers' inventories to their full capacity if and each time they are replenished. Inventories are reviewed once at the end of every period t . This resembles an (R, s, S) policy with review period $R = 1$, reorder point $s = \sum_{u \in U} \mathbb{E}[D_{itu}]$ and order-up-to-level $S = \bar{I}_i$ for each customer. Hence, under OU, the replenishment quantities for each customer are obtained using

$$q_{it} = \begin{cases} \bar{I}_i - I_{i,t-1,|U|} & \text{if } I_{i,t-1,|U|} < s \\ 0 & \text{else} \end{cases} \quad (4.3)$$

Note that the index u is dropped as the replenishment quantities for OU do not depend on the sub-period of delivery. The replenishment timings are obtained by the ALNS used for the routing.

Chance Constrained $(CC\tau)$

The algorithm incorporates information on the uncertainty of demands. It sets a safety stock for each customer based on a target non-stockout probability α and a planned run-out time of τ periods. This resembles an (R, s, S) policy with review period $R = 1$, the replenishment level is given by $S = q_{itu}(\alpha, \tau)$, which is the amount necessary to guarantee a non-stockout probability α

at customer i if replenished in sub-period (t, u) covering the demands of the next τ periods, i.e.,

$$q_{itu}(\alpha, \tau) = \sum_{\theta=(t,u)}^{(t+\tau,u)} \mathbb{E}[D_{i\theta}] + F^{-1}(\alpha) \sqrt{\sum_{\theta=(t,u)}^{(t+\tau,u)} \sigma_{i\theta}^2}, \quad (4.4)$$

where $F^{-1}(\alpha)$ is the inverse cumulative unit normal distribution and σ_{itu} represents the standard deviation of customer i 's demand in period and sub-period (t, u) . The reorder point is $s = q_{itu}(\alpha, 1)$, such that if the inventory level at the beginning of the period is insufficient to maintain the α -service level until the end of that period, the replenishment quantity is set to fill customers' inventories to a level sufficient to obtain an α -service level for the next τ periods. If the replenishment level S should be greater than the inventory capacity, the replenishment quantity is set to completely fill the customer's inventory. Therefore, under $CC\tau$, the replenishment quantities for each customer are obtained using

$$q_{itu} = \begin{cases} \min\{\bar{I}_i - I_{i,t-1}, q_{itu}(\alpha, \tau) - I_{i,t-1}\} & \text{if } I_{i,t-1} \leq q_{itu}(\alpha, 1) \\ 0 & \text{else.} \end{cases} \quad (4.5)$$

4.4.2 Retailer Decomposition - Policy Function Approximation

The third heuristic is the Customer Decomposition - Policy Function Approximation (CD-PFA). This algorithm obtains a replenishment policy for each independent customer, considering an estimation of the expected routing costs that a replenishment would cause. In CD-PFA, we solve the SDP given in (4.2) for each customer separately to overcome the curses of dimensionality. The routing costs are substituted by $C_R(q_t)$ with an expected routing costs approximation $\tilde{C}_{Ri}(k)$, which is based on the average distance of customer i to its k nearest customers.

The approximate value function for each customer is given by:

$$\tilde{V}_i(I_{it}) = \min_{q_{it} \in A(I_{it})} \left\{ C_{Ii}(I_{it}, q_{it}) + \tilde{C}_{Ri}(q_{it}) + \mathbb{E} \left[\sum_{t'=t+1}^{|T|} (C_{Ii}(I_{it'}, q_{it'}) + \tilde{C}_{Ri}(q_{it'})) \right] \right\}. \quad (4.6)$$

All previously mentioned approaches are designed to achieve target service levels. To guarantee comparability to those approaches, CD-PFA includes a component that ensures target service levels at the expense of a purely cost-based optimization. Using the approximate cost function (4.6), we obtain approximate replenishment policies through:

$$\tilde{q}_{it}(I_{itu}, \alpha) = \begin{cases} \underset{q_{itu} \in A(I_{itu})}{\operatorname{argmin}} \tilde{V}_i(I_{itu}) & \text{if } P\left(I_{itu} - \mathbb{E}\left[\sum_{t'=(t,u)}^{(t+1, e_{i,t+1})} D_{it'}\right] \geq 0\right) \geq \alpha \\ q'(I_{itu}, \alpha) & \text{else,} \end{cases} \quad (4.7)$$

with $q'(I_{itu}, \alpha)$ being a replenishment quantity large enough to ensure a non-stockout probability α until the next replenishment possibility at customer i , i.e., the start of the customer's delivery time window in the next period, i.e., $e_{i,t+1}$. Hence, $q'(I_{itu}, \alpha) = \sum_{\theta=(t,u)}^{(t+1, e_{i,t+1})} \mathbb{E}[D_{i\theta}] + F^{-1}(\alpha) \sqrt{\sum_{\theta=(t,u)}^{(t+1, e_{i,t+1})} \sigma_{i\theta}^2}$.

For RD-PFA, each customer's replenishment policy is approximated by (4.7). Afterwards, this information is used to obtain the replenishment quantities q_{itu} (and alter them according to the sub-period of delivery u inside a period t) for the routing planning by applying the ALNS. We resort to a retailer decomposition because a joint optimization for multiple customers already becomes intractable computationally for a small number of customers due to the curses of dimensionality.

4.4.3 Routing

Every period, the customers to be replenished are decided according to their reorder point s , or calculated replenishment policy. The quantity to be delivered is derived using one of the proposed approximate policies. To create the delivery routes, we propose a metaheuristic solution approach based on ALNS, introduced by Pisinger and Ropke [68], which is commonly used to efficiently compute solutions for IRPs, periodic VRPs, and VRPs. We generate an initial solution using the cheapest (feasible) insertion heuristic [cf. 79] extended with checks for time window feasibility. If a customer cannot be inserted into a route due to time window feasibility, a new route is started. If no more vehicles are available to do so, the customer is not inserted into the routes of the initial solution. This does not mean that the customer is left out ultimately, as we then iteratively apply several destroy and repair operators to remove and rebuild parts of the solution. To enhance the quality of the routes, we apply different operators and explore different solution neighborhoods.

Cheapest Insertion Heuristic

First, we initialize a route by inserting the customer farthest to the depot, included in the set of customers that must be served in that period. After that, we insert customers with the delivery quantities determined by their replenishment policy. Third, if we cannot insert any more customers in the route due to capacity constraints of trucks, we initialize a new route in the same way. We iterate until all customers to be served throughout that period have been inserted. Finally, to improve the quality of the initial solution, we apply the 2-opt algorithm that seeks better sequences of customers within the routes and respects the feasibility of the routes when considering vehicle capacity.

Adaptive Large Neighborhood Search

ALNS integrates several operators to destroy and repair the current solution iteratively. Furthermore, to avoid local optima, we introduce the option of accepting deteriorating solutions after long periods without finding an improving solution. In each iteration of the ALNS, we generate a new solution by applying one destroy and one repair operator. The selection of these operators relies on a roulette-wheel selection operator based on the past performance of the operators. The selection and prioritization of destroy and repair operators are pairwise rather than separate. Weights $\rho_{iter,d,r}$ related to each pair of destroy and repair operators are initially set to the same value, in which the first index is an iteration (*iter*) counter, whereas the second and third indices relate to the destroy (*d*) and repair (*r*) operator, respectively. When we find a new best solution or better incumbent solution, we update the weights of the operators, increasing them by γ and β , respectively. Then, we resume the procedure of updating the operator weights as follows:

$$\rho_{iter,d,r} = \begin{cases} \rho_{iter-1,d,r} + \gamma & \text{if new best solution found.} \\ \rho_{iter-1,d,r} + \beta & \text{if new better incumbent solution found.} \\ \rho_{iter-1,d,r} & \text{if no better solution found.} \end{cases} \quad (4.8)$$

When we obtain a new solution after applying both the destroy and the repair operators, we apply the 2-opt algorithm to the newly obtained solution to find possible improvements for each route sequence. If the new solution is better than the existing one, the existing solution and the

weights for the destroy and repair operators used in this iteration are updated. The best-known solution is also updated if the new solution improves. If the new solution does not improve the incumbent or the best-known solution, the further steps depend on the amount of time that has passed since the last improvement. If the amount of time is lower than a *time limit*, the previous incumbent solution is used as a starting solution for the next ALNS iteration. If that time limit is exceeded, the new solution is accepted as the incumbent solution and, therefore, as the input for the next iteration of the ALNS. After this, the time of the last improvement found is reset. The stopping criterion for the ALNS is an overall limit on runtime set to one second per ALNS run.

We propose five destroy operators and five repair operators, as described next.

- **Remove worst:** Given a solution *sol*, this operator calculates the detour cost of every insertion of each customer *i*, $cost = c_{ki} + c_{ij} - c_{kj}$, where *k* and *j* are the preceding and succeeding customers in the route, respectively. Then, select the *p* worst insertions concerning these costs and remove them from the solution.
- **Remove random:** remove *p* different insertions from random routes.
- **Remove vehicle:** remove all insertions from a random vehicle.
- **Remove close:** select *p* random insertions and their closest neighboring customer and remove them from the solution.
- **Remove furthest customer and closest:** remove the *p* insertions furthest from the depot and their closest insertion in terms of distance.

Every time we remove a customer from the solution, we insert this customer into a list with those who are not part of the current solution but must be delivered in the current period. For the resulting list of customers, we then apply one of the following repair operators corresponding to the above destroy operators. Time window and vehicle capacity feasibility checks are performed such that an insertion is only made if it does not violate any of the two restrictions.

- **Best insertion in destroy order:** Insert customers, in the order that they were removed, into the existing routes. We list all possible insertions of each customer that do not violate the time windows of the already inserted customers and the capacity of the vehicles, sorting them by the total distance of the detour caused by them. We then randomly select and apply one of the three best possible insertions.

- **Best insertion in random order:** This operator works in the same way as the previous one, except that we randomly select the next customer to be reinserted into the solution.
- **Random insertion in destroy order:** List the possible insertions just like the *best insertion in destroy order* operator but then randomly select one of all possible insertions from the entire list.
- **Random insertion in random order:** Select a random customer that was removed from the solution and perform a random insertion. Repeat the process until the removed customers have been reinserted.
- **Distance-related insertion:** This operator selects customers in the same order as they were removed from the solution and calculates the best possible insertions in terms of the detour they cause (as in the remove worst operator). After inserting a customer, we search for the closest customer in the list of customers to be reinserted and insert it next to the previously inserted one. If this is not possible, inserting the nearby customer is omitted. This process is repeated until all removed customers have been reinserted.

4.4.4 Adaptive lookahead Procedure

The methods presented in Subsection 4.4.4.1 apply a delivery policy to the customers that need to be replenished. In Subsection 4.4.4.2, we present a policy function algorithm that obtains the best inventory replenishment policy independently for each customer using a routing cost approximation. In this subsection, we present an iterative procedure based on the idea of an adaptive lookahead algorithm. This solution approach integrates the decisions related to delivery quantities and routing, such that delivery quantities and routing decisions are made by combining the four different replenishment policies that can be applied to each customer previously described (OU, and $CC\tau$).

The algorithm can be divided into two phases for each period of the planning horizon. The first phase creates a plan for the current period and the periods within the lookahead horizon using a pre-defined replenishment policy for each customer. We create an initial set of routes for these periods using the cheapest insertion algorithm and the ALNS described previously. For an instance that would consider three customers and three periods with one sub-period, the first phase would apply the routing algorithms for the current period using a pre-defined replenishment policy. After we obtain a set of routes for the current period and the lookahead

periods, the second phase uses these routes as input. It iteratively aims at improving the solution by evaluating other replenishment policies for each customer independently. This means that, at each iteration, we randomly change the replenishment policies of randomly selected customers and evaluate the impact of applying different replenishment policies. Furthermore, the second phase of the algorithm also modifies the list of customers that must be replenished by evaluating the different routing, holding, and stock-out costs.

Figure 4.3 presents a flowchart of the proposed method. The algorithm identifies the best replenishment policy for each customer independently, increasing the probability of selecting these policies through an iterative and adaptive procedure.

Initial routes generation

The first phase of the iterative algorithm obtains an initial delivery plan for each period, considering information on the lookahead algorithms. In algorithm 1, we present the pseudocode for each period p during the planning horizon.

At the beginning of the algorithm, we initialize the weights of each of the above-mentioned replenishment policies with value 20, and we generate a pool of sample scenarios using descriptive sampling, as given in Saliby [75]. For that purpose, a set of scenarios \mathcal{S} is drawn randomly from each customer's descriptive samples of their demand distributions for a specific period t and sub-period u , using ψ intervals with $F^{-1}[(\varsigma - 0.5/\psi)]$, and $\varsigma \in \{1, \dots, \psi\}$. Then, after the different sample scenarios are generated, we initialize the algorithm.

The aim of the initial route generation phase is to create an initial set of replenishment routes for the current period and the lookahead period that will be then used in the second phase of the algorithm. For each period of the planning horizon, the first phase of the adaptive lookahead procedure creates an initial set of routes to replenish the customers for the current and subsequent periods in the lookahead horizon. However, in the first period of the planning horizon, the four described policies are applied independently to the lookahead planning horizon, considering expected demands. This warm start creates four different initial partial solutions within the lookahead horizon of the first period. This procedure provides a better starting point for the algorithm, in which the information related to the holding and routing costs are utilized to obtain a good base solution. In this way, when we deal with instances that consider high holding costs at the customers, policies that typically deliver smaller quantities to the customers

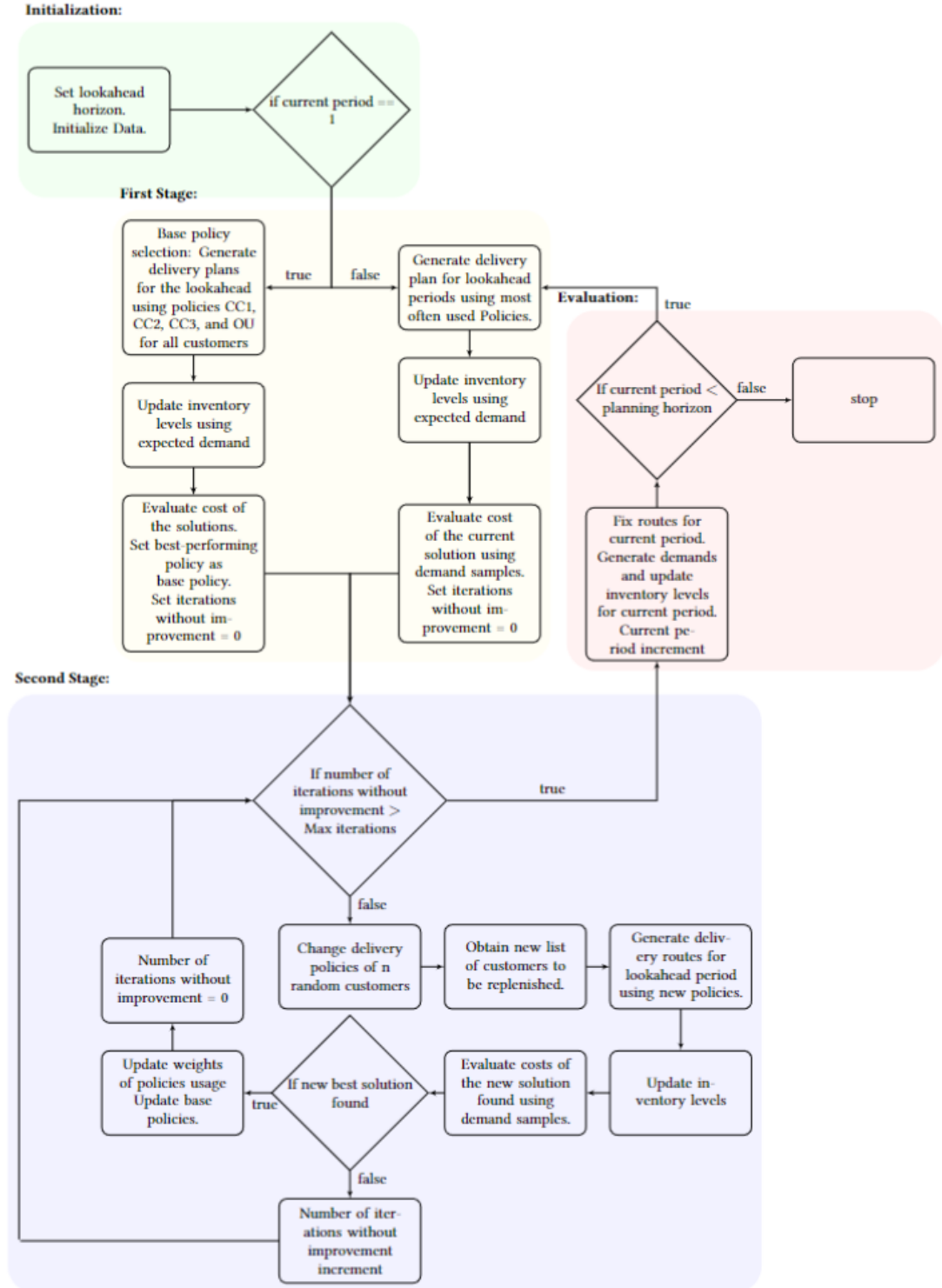


Figure 4.3: Flowchart of the adaptive lookahead procedure.

Algorithm 1 Initial routes generation

```

1: if (currentperiod=1) then
2:   Initialize policy weights
3:   Obtain delivery plan for period currentperiod and lookahead using CC2 replen-
   ishment policy
4:   Evaluate solution CC2 & get  $cost_{CC2}$ 
5:   Obtain delivery plan for period currentperiod and lookahead using CC3
6:   Evaluate solution CC3 & get  $cost_{CC3}$ 
7:   if  $cost_{CC2} < cost_{CC3}$  then
8:     Obtain delivery plan for period currentperiod and lookahead using CC1
9:     Evaluate solution CC1 & get  $cost_{CC1}$ 
10:    if  $cost_{CC1} < cost_{CC2}$  then
11:      Set CC1 as base replenishment policy
12:      Return Solution CC1 &  $cost_{CC1}$ 
13:    else
14:      Set CC2 as base replenishment policy
15:      Return Solution CC2 &  $cost_{CC2}$ 
16:  else
17:    Obtain delivery plan for period currentperiod and lookahead using OU
18:    Evaluate solution OU & get  $cost_{OU}$ 
19:    if  $cost_{OU} < cost_{CC3}$  then
20:      Set OU as base replenishment policy
21:      Return Solution OU &  $cost_{OU}$ 
22:    else
23:      Set CC3 as base replenishment policy
24:      Return Solution CC3 &  $cost_{CC3}$ 
25:  else
26:    Obtain delivery plan for period currentperiod and lookahead using best replen-
    ishment policy for each customer
27:    Evaluate solution & get  $cost_{best}$ 
28:    Return  $Solution_{best}$  &  $cost_{best}$ 

```

but perform more visits to them (CC1, CC2) are more likely to be selected as base policy and, on the opposite, CC3 or OU policies are more likely to become the base policy if holding costs are low. When applying these methods, we obtain four different delivery plans for the set of customers by means of the cheapest insertion heuristic and ALNS algorithm for each period in the lookahead horizon. Next, we evaluate the partial results for the lookahead horizon using the generated demand samples. The policy that reports the lowest total costs is selected as the initial base policy for the adaptive lookahead procedure. From the second period up to the end of the planning horizon, we create a delivery and replenishment schedule using the already selected base policies for each customer.

In each period, after we obtain an initial solution for the current and subsequent periods considered in the lookahead, we evaluate the routing, inventory, and stock-out costs incurred at each customer using random demand samples. We then apply the second phase to modify the current solution.

Adaptive policy evaluation mechanism

In the second phase of the algorithm, we use the information of the solution created in the first phase of the algorithm, and we apply an iterative procedure to improve it in the current and the lookahead periods by evaluating the impact of independently applying different replenishment policies to each customer. In this phase, we gain further insights by evaluating the expected costs related to each customer when delivering or not delivering to them, as well as the replenishment policy. In this situation, deliveries to customers who are far away from the rest of the customers and who are unlikely to run out of stock could be put off until later periods. On the other hand, it is possible to serve certain customers in the current period if the distance to the rest of the customers in the list is relatively low and, as a consequence, advance their delivery to create a better routing schedule. In Algorithm 2, we present a pseudocode of the second phase of the algorithm for each period of the planning horizon.

Algorithm 2 Adaptive policy evaluation mechanism

Require: Initial solution and cost. $sol_{initial}$ & $cost_{initial}$

Require: $Maxiter$

```

1:  $sol_{best} := sol_{initial}$  &  $cost_{best} := cost_{initial}$ 
2: Set Iteration without improvement  $It = 0$ 
3: while  $It < Maxiter$  do
4:   Change replenishment policy of  $n$  random customers from  $sol_{best}$  with random
   policy-change
5:   Obtain delivery plan for period currentperiod and lookahead using new replen-
   ishment policies
6:   Evaluate new solution  $sol_{new}$  get  $cost_{new}$ 
7:   if  $cost_{new} < cost_{best}$  then
8:      $sol_{best} := sol_{new}$  &  $cost_{best} := cost_{new}$ 
9:      $It = 0$ 
10:  else
11:     $It++$ 
12: Return  $sol_{best}$  &  $cost_{best}$ 

```

We propose an adaptive iterative procedure in which, in each iteration, we select a random number of customers and modify the policy used to serve them for the current and subsequent periods. The number of selected customers to whom the policy changes are applied is a random

number between 1 and $|N|$. We select a random policy-change operator by means of the roulette wheel selection and weights $\delta_{iter,p}$ for each policy change operator p . We describe the policy-change operators as follows and use the ascending order $CC1 < CC2 < CC3 < OU$ among the policies:

- **Increase:** For each random customer selected, we use the currently applied replenishment policy and change it for the next higher-indexed policy in the order described before. If no such policy exists, i.e., if the replenishment policy currently applied is OU, it remains with this policy.
- **Decrease:** For each random customer selected, we proceed in the opposite way as the previous operator. We assign the previous replenishment policy in the order described before. If no lower policy exists, assign CC1.
- **Random:** We assign a random replenishment policy to each customer selected, which may differ between customers.
- **Single:** We randomly select a replenishment policy and apply it to all customers selected.
- **Reset:** To be able to reach unexplored regions of the entire solution space, we propose a reset operator with a fixed probability in which we select a random policy not only for the selected customers but for all customers.

After applying the policy changes, we evaluate the costs related to each customer in the previous solution. Then, we modify the list of customers that must be replenished during the current period. This decision is taken according to two different situations. If, during the lookahead period, any customer not included in the list experiences an expected loss of sales smaller than a certain percentage of the holding costs of the considered periods, we introduce this customer into the list. On the other hand, customers may face excessive holding costs compared to the routing cost. We evaluate the customers whose delivery can be delayed or postponed to one of the following periods, even if they are in the list of customers to be replenished in that period. We then remove them from the current list of customers that need to be replenished. We advance the replenishment of the customers whose inventory is expected to fall below their reorder point (ROP) specified by the policy chosen before. We achieve this by temporarily shortening their time windows up to the sub-period in which we expect this to happen. When applying this random selection procedure, each of the customers present in the list to be served presents a

related replenishment quantity for this iteration. We then generate the routes for the current period using these quantities and the constructive heuristic and ALNS previously described, and we apply an OU policy to simulate the routing cost until the end of the lookahead horizon.

During the iterative procedure, we evaluate the different costs of the applied solution using the pool of samples generated using the descriptive sampling technique. Then, if the newly generated solution has the lowest total costs, we update the weights of the policies applied to each customer to increase the likelihood of these policies being selected more frequently for each customer. The following defines the update operator procedure:

$$\delta_{iter,d,r} = \begin{cases} \delta_{iter-1,d,r} + \gamma & \text{if new best solution found.} \\ \delta_{iter-1,d,r} & \text{if no better solution found.} \end{cases} \quad (4.9)$$

This learning procedure facilitates dealing with different problem parameters and adapts independently to each customer. We also update the information on the best replenishment decisions for the current period and continue with the next iteration. Finally, if, after a given max number of iterations, we cannot find any combination of replenishment policies that would reduce the expected costs, the iterative procedure stops, and the best solution strategy obtained is applied to the current period. When this procedure ends, the demands for the current period are disclosed, the new inventory states calculated, and the iterative procedure for the next period applied until the end of the planning horizon is reached.

4.5 Numerical Evaluation

We perform computational experiments on instances based on real-world information to gain insights into the effects different problem characteristics have. This section presents the results obtained when evaluating different parameters crucial in a stochastic dynamic problem, such as the stochasticity of demands, the instance size, and the costs of holding inventory and losing sales.

The algorithms we propose are coded in C++ and all computational experiments performed on virtual machines with eight cores and 40GB of RAM running on an Intel(R) Xeon(R) Gold 6240R CPU.

4.5.1 Data and Instances

To test the efficiency of the presented algorithms, we create a new real-world-based data set of instances. We consider distances and travel times of a city street network. Distances contribute to the cost term in the objective function, and travel times enable feasibility checks for the time windows in the routing. Demand information resembles sales of stores of a retail chain during five three-hour intervals per day for six days a week (Monday to Saturday). Note that the stores do not operate on Sundays. We fit each customer's demand into normal distributions. Hence, the underlying instances consist of $|T| = 6$ periods with $|U| = 5$ sub-periods each, allowing for a trade-off between more detailed demand data and increased problem complexity. The demand distributions $\mathcal{N}(\mu_{itu}, \sigma_{itu})$ for each $i \in N$, $t \in T$ and $u \in U$ are independent between customers and truncated, such that demands are positive. Furthermore, for a greater instance variability σ_{itu} is obtained assuming coefficients of variation $cv \in \{0.1, 0.25, 0.4\}$ and $\sigma_{itu} = \mu_{itu} \cdot cv$. The time windows resemble the replenishment time windows of the stores.

In addition, we create three sets of instances in which customers face low, medium, and high holding costs per unit and period. The sets of instances generated are denoted by LO , MI , and HI , where $h_i^{LO} \in \mathcal{U}[0.02, 0.1)$, $h_i^{MI} \in \mathcal{U}[0.1, 0.3)$, and with $h_i^{HI} \in \mathcal{U}[0.3, 0.5]$. Furthermore, we consider per unit stock-out costs $\phi_i = 2$. Inventory capacities are a multiple in the interval of 2 to 3.5 days of the customers' highest mean demand of all periods, i.e., $\bar{I}_i = \max_{t \in T} \{ \sum_{u \in U} \mu_{itu} \} \nu_i$, where $\nu_i \in \mathcal{U}[\frac{1}{3}, \frac{7}{12}]$. Initial inventories are given by $I_{i0} = \bar{I}_i \xi_i$, with $\xi_i \in \mathcal{U}(0.25, 1.0)$. The target non-stockout probability for all instances is $\alpha = 0.95$.

We consider four different instance sizes $|N| \in \{5, 10, 20, 40\}$. For each $|N|$, we solve the problem assuming two vehicle fleet sizes, i.e., two or three available vehicles for sizes $|N| \in \{5, 10, 20\}$, and four or five vehicles for $|N| = 40$. Furthermore, we set vehicle capacities for the homogeneous fleet so that the whole fleet can serve three times the maximum daily demand. Finally, for $CC\tau$, we use $\tau \in \{1, 2, 3\}$. We explore these run-out times as the inventory capacity of each customer is sufficient to store at most the expected demand for the next three to four periods.

4.5.2 Performance of Approximations, RD-PFA and Adaptive lookahead

In the first group of computational experiments, we evaluate the performance of the adaptive lookahead procedure compared to the rest of the benchmark policies and the RD-PFA introduced

in Sections 4.4.4.1 and 4.4.4.2. Furthermore, we investigate the influence of different lookahead period lengths 2, 3, and 4 when applying the proposed algorithm and refer to the different versions of this algorithm as ALA2, ALA3, and ALA4.

During the computational experiments, one maximum runtime for the ALNS algorithm used in all the methods is set. We use a total solving time for the ALNS equal to one second. Hence, the computational times for OU and $CC\tau$ are always approximately 6 seconds. Secondly, the performance of the CD-PFA algorithm was tested considering different numbers of closest customers for the routing cost approximation, where the best performance was obtained when considering only one customer. Therefore, the reported results show the best-performing option for which the runtimes vary between 6 seconds for the smallest instances and 11 seconds for the instances with 40 customers. On the other hand, the computation time needed when applying the proposed algorithms depends on the number of improvements found during the adaptive lookahead procedure. For this purpose, Table 4.1 presents the average computation times of the proposed algorithm for lookahead horizons of two, three, and four periods. For the adaptive lookahead procedure, the computation time increases with the length of the lookahead period and the number of customers. The stopping criterion allows for 15 iterations without improvement. Additionally, with more lookahead periods, the routes for a larger number of periods need to be optimized, which causes the runtimes to lengthen. For larger number of customers, the chance to find improvements is larger. Hence, the counter for iterations without improvement is reset more often causing longer runtimes.

algorithm/ $ N $	5	10	20	40	avg
ALA2	284	351	395	469	375
ALA3	437	516	550	651	539
ALA4	537	587	597	711	608

Table 4.1: ALA runtimes depending on instance size [in seconds].

In Tables 4.2, 4.3, and 4.4, we present the results obtained when solving all sets of instances. We calculate the average percentage deviation in total costs compared to the best-performing algorithm in each instance Δ_{TC} . Furthermore, Δ_H and Δ_R indicate the percentage difference in routing and holding costs compared to the best-performing algorithm, according to Δ_{TC} . Finally, (H—R—S) presents the total cost shares of holding costs (H), routing costs (R), and stock-out costs (S). Note that for some lines, their sum is not equal to 100 due to rounding.

In Table 4.2, we evaluate the impact of considering different holding costs. On average, ALA4

shows the best results compared to the other algorithms. This version of the algorithm deviates by 2.3% compared to the best result obtained by any algorithm used in each instance, whereas ALA3 shows a good performance, deviating by 2.6%. When considering LO holding costs, ALA3 outperforms the rest of the methods, whereas when holding costs are MI and HI, ALA4 performs better. Furthermore, all the versions of the iterative lookahead procedure widely outperform the benchmark approximations and the CD-PFA on average due to their ability to assess the future impact of replenishment and routing decisions and their interplay.

h		OU	CC1	CC2	CC3	RD-PFA	ALA2	ALA3	ALA4
avg	Δ_{TC}	17.0	32.1	19.6	20.5	20.4	4.9	2.6	2.3
	Δ_H	36.7	-11.4	23.4	44.3	-12.0	5.9	7.7	1.3
	Δ_R	-7.4	70.7	14.2	-3.3	46.5	2.5	-2.2	2.5
	(H/R/S)	56/41/03	34/64/02	50/49/01	57/42/01	36/60/04	49/49/02	50/48/02	48/50/03
LO	Δ_{TC}	10.7	53.2	20.2	12.7	35.4	4.8	1.3	4.0
	Δ_H	9.1	-28.8	-1.2	16.1	-15.8	-3.3	1.9	-3.5
	Δ_R	7.3	97.6	32.3	11.8	61.4	9.2	1.2	7.0
	(H/R/S)	35/61/04	16/82/02	29/70/02	36/63/01	22/75/03	32/66/02	35/63/01	32/65/02
MI	Δ_{TC}	14.2	26.3	16.1	17.6	17.5	3.9	2.4	1.5
	Δ_H	34.2	-13.1	21.2	41.0	-13.6	4.2	5.3	0.1
	Δ_R	-6.7	72.1	15.0	-2.4	48.1	4.0	0.4	3.4
	(H/R/S)	59/38/02	35/64/02	53/46/01	60/39/01	37/59/04	51/47/02	52/46/02	50/48/02
HI	Δ_{TC}	26.2	16.9	22.6	31.3	8.3	5.9	4.1	1.5
	Δ_H	66.8	7.7	50.1	75.9	-6.5	16.8	15.9	7.5
	Δ_R	-22.6	42.4	-4.6	-19.3	30.0	-5.6	-8.3	-3.0
	(H/R/S)	75/24/01	52/46/01	70/30/01	76/24/00	49/45/06	63/34/03	63/34/03	60/37/03

Table 4.2: Algorithm performance depending on holding costs.

There is a noticeable tendency among the benchmark approximations for low, medium, and high holding costs. The benchmark approximation OU is the best replenishment strategy on average. However, there is a clear increase in costs as the holding costs share increases. The OU approximation is the best option when the holding costs are low, as customers can store large commodity quantities at low costs. However, when considering medium and high holding costs, the strategies in which the amounts replenished are lower and the number of visits to each customer increases (CC1 and CC2) obtain better results. This situation aligns with the intuition to reduce replenishment frequencies if holding costs are low compared to routing costs and, on the other hand, to increase the replenishment frequencies if holding costs are high compared to routing costs. Another insight is that CD-PFA offers better replenishment plans than the rest of approximate algorithms when holding costs are HI as for this setting the static routing cost

approximation is not very relevant and replenishments are performed almost every periods for every customer.

The second insight obtained when analyzing the results derives from the three different lookahead horizons. When solving instances with LO holding costs, ALA3 outperforms the other methods (1.3%). However, when holding costs considered are MI or HI, ALA4 benefits through further exploration of the next periods to find a more efficient replenishment plan on average (1.5% in both cases).

Regarding Δ_H and Δ_R for the overall case, for ALA, the values are quite balanced. In contrast, the other approaches which lack the capability for adaptation have a clear tendency. In addition, we can observe that considering a longer lookahead horizon allows ALA4 to reduce costs for all holding cost settings compared to the rest of the algorithms. For OU and CC3, for which a lot of inventory is held, we see a high Δ_H and a low Δ_R as these approaches feature high replenishment quantities and low routing frequencies as opposed to CC1, for which we see a low Δ_H , and a high Δ_R because for CC1 replenishment quantities are low, which results in a high routing frequency. For low holding costs, we see CC3 performing very well on Δ_R because replenishment quantities are set to cover three upcoming periods, which comes with a low replenishment frequency. On the other hand, the service level requirement for the CD-PFA algorithm translates to higher routing costs than all methods except CC1.

In Table 4.3, we evaluate the performance of the algorithms depending on the instance size. ALA3 and ALA4 show the best cost performance for all instance sizes $|N|$, where ALA4 shows a better performance for instances with 5, 10, and 20 customers and ALA3 for instances with 40 customers for a difference of 0.1%. Furthermore, the benchmark approximations $CC\tau$ and the CD-PFA perform increasingly well with larger instance sizes as, for instances with fewer customers, fallback options are few in routing and potentially fewer nearby customers. The adaptive lookahead procedure finds the right replenishment timings and quantities and significantly outperforms the other approaches. Additionally, the effects of instance size in columns Δ_H , Δ_R , and the cost-composition (H—R—S) are similar to the ones presented in Table 4.2 and discussed above.

N		OU	CC1	CC2	CC3	RD-PFA	ALA2	ALA3	ALA4
5	Δ_{TC}	26.0	39.7	26.3	27.3	24.5	7.0	2.6	2.4
	Δ_H	45.2	2.8	33.9	53.9	6.4	0.5	5.0	-0.7
	Δ_R	0.3	65.7	16.6	1.8	38.4	10.1	-0.2	3.9
	(H/R/S)	52/45/03	35/64/01	48/52/01	54/46/01	39/60/01	42/56/02	45/53/01	43/55/02
10	Δ_{TC}	17.5	34.2	21.1	22.6	22.5	5.0	2.6	1.7
	Δ_H	35.3	-5.9	24.9	44.7	-7.5	6.7	5.8	0.0
	Δ_R	-5.5	71.9	14.3	-2.2	51.8	2.2	-0.9	2.3
	(H/R/S)	57/41/02	37/63/00	52/47/01	59/41/00	38/60/01	50/48/02	51/48/02	49/49/02
20	Δ_{TC}	10.9	30.8	17.2	17.1	19.1	3.9	2.5	2.4
	Δ_H	28.3	-21.8	15.7	36.8	-23.2	7.2	7.6	0.0
	Δ_R	-12.6	80.6	15.6	-4.9	55.6	0.5	-2.2	4.4
	(H/R/S)	59/39/02	33/65/02	51/48/01	59/40/01	34/61/05	52/46/02	53/45/02	50/48/02
40	Δ_{TC}	13.6	23.7	13.9	15.2	15.6	3.7	2.7	2.8
	Δ_H	37.9	-20.7	18.9	41.9	-23.6	9.2	12.4	6.0
	Δ_R	-11.7	64.7	10.4	-7.8	40.3	-2.6	-5.7	-0.7
	(H/R/S)	57/39/03	33/63/03	50/48/02	58/41/01	32/58/10	50/46/05	51/44/04	49/47/05

Table 4.3: Algorithm performance depending on instance size.

Table 4.4 shows the performance of the algorithms depending on cv . The observed effects are similar to those in the previous table and prove the adaptability of the presented adaptive lookahead algorithm for different levels of demand variability. The versions of the proposed algorithm ALA3 and ALA4 perform best for all cv , outperforming the benchmark approximations and the CD-PFA by at least 12% lower total costs as it evaluates many replenishment schedules and their intra-period timings considering the effect they have on routing synergies, resulting holding costs and stockouts. The results in Tables 4.3 and 4.4 show a stable performance and robustness of ALA concerning the number of customers in an instance and the coefficient of variation.

cv		OU	CC1	CC2	CC3	RD-PFA	ALA2	ALA3	ALA4
0.1	Δ_{TC}	21.1	33.4	19.5	20.2	21.4	5.6	2.3	1.9
	Δ_H	43.6	-12.9	22.0	44.3	-11.2	5.2	7.1	0.1
	Δ_R	-5.0	74.2	16.1	-4.6	45.2	3.4	-2.6	3.0
	(H/R/S)	56/41/02	33/65/02	49/50/01	57/42/01	35/59/05	48/49/03	49/48/03	47/51/03
0.25	Δ_{TC}	15.0	31.7	19.8	20.8	20.1	4.4	2.8	2.4
	Δ_H	33.8	-10.1	24.8	45.2	-11.9	6.6	8.1	2.2
	Δ_R	-8.7	68.8	13.0	-2.8	47.0	2.2	-2.1	2.1
	(H/R/S)	56/41/03	35/63/02	51/48/01	58/42/00	36/60/04	49/49/02	50/48/02	48/50/02
0.4	Δ_{TC}	14.9	31.2	19.6	20.6	19.7	4.7	2.6	2.7
	Δ_H	32.7	-11.2	23.3	43.6	-12.8	5.9	7.9	1.7
	Δ_R	-8.4	69.1	13.6	-2.5	47.4	2.1	-2.1	2.4
	(H/R/S)	56/41/03	35/63/02	51/48/01	58/42/00	36/60/04	49/49/02	51/47/02	48/49/02

Table 4.4: Algorithm performance depending on coefficient of variation.

4.5.3 Benefit of Intra-Day Planning

We compare the results obtained when planning using aggregated daily data in full-day planning (FP) instead of more granular data over five sub-periods per day with intra-day planning (IP). For this comparison, we use the method that reports consistent average results and running time, i.e., ALA3.

To generate the aggregated demand distributions for FP, we calculate the cumulative demands of the five sub-periods and use this aggregation as average demand. Then we calculate the aggregated standard deviation related to each period as $\sigma_{it} = \sqrt{\sum_{u=1}^{|U|} \sigma_{itu}^2} \forall i \in N, t \in T$. We solve the instances that do not consider a sub-period division using the aggregated demand distributions. Next, we use the solution obtained for FP and evaluate the cost performance considering the multi-sub-period scenario.

Table 4.5 shows the deviation in total cost for each combination of holding cost, instance size, and coefficient of variation when comparing the solutions obtained for both planning strategies FP and IP. On average, applying IP reduces the overall costs by 24.5%. These results show that planning using aggregated information generates inefficient delivery plans. The aggregated demand information does not allow the supplier to estimate the demand that customers experience until the planned replenishment. Due to this incorrect information, a correct adaptation of delivery times and quantities is impossible.

cv $ N /h$	0.1				0.25				0.4				avg	Δ_{time}
	LO	MI	HI	avg	LO	MI	HI	avg	LO	MI	HI	avg		
5	32.5	33.8	31.4	32.6	33.7	31.1	26.6	30.5	33.2	29.4	28.5	30.3	31.1	89
10	17.3	18.5	19.6	18.5	14.8	16.5	16.3	15.9	16.5	16.9	14.1	15.8	16.7	88
20	16.7	19.0	19.5	18.4	16.5	19.3	16.5	17.4	13.8	17.2	15.3	15.4	17.1	100
40	43.6	33.4	26.1	34.4	39.7	30.5	23.9	31.3	43.5	32.2	24.0	33.2	33.0	172
avg	27.6	26.2	24.1	25.9	26.2	24.3	20.8	23.8	26.7	23.9	20.5	23.7	24.5	113

Table 4.5: Total Cost Savings and runtime comparison of intra-day planning compared to full-day planning [in % and seconds].

In Table 4.5, it is also noticeable that IP outperforms FP the most for 5 and 40 customers. However, the improvements are smaller for instances that consider 10 and 20 customers. The reason behind this behavior is the number of vehicles combined with the number of customers. When solving instances with five customers, the FP algorithm often uses a single vehicle to replenish the customers in a period. This is different for IP, in which creating more routes

avoids stockouts due to late deliveries. This shows that the intra-day demand data helps predict inventory depletion more precisely. Even so, IP still lowers routing costs because, in FP, customers need more deliveries within the planning horizon as they run out of stock more often. So, when there are 10 or 20 customers, the route is less flexible because the number of vehicles stays the same as with only five customers. This reduction diminishes the advantage of IP over FP described above. Finally, with 40 customers, having four or five vehicles instead of two or three makes the route more flexible. Hence, for instances with 40 customers, the benefit of IP over FP is comparable to the instances with five customers. The difference in runtimes between IP and FP in seconds is given by Δ_{time} . With an increasing number of customers, Δ_{time} is increasing. Instances with fewer customers are solved in 88 and 89 seconds longer runtimes compared to FP. For instances with 20 and 40 customers, runtimes are extended by 100 and 172 seconds. On average, solving the instances using IP takes 113 seconds longer than under FP.

Table 4.6 shows the average deviation in the different costs comparing the impact of the sub-period consideration. In columns Δ_H and Δ_R , we present the savings in holding and routing costs when creating a delivery plan using IP instead of FP. $S_{IP|S_{FP}}$ shows the average stock-out costs share of the total costs when considering sub-periods (IP) and the case considering aggregated daily data (FP). We can see how, for IP, the average holding costs decrease by 13.1% and routing costs decrease by 9.9% on average. Moreover, the share of stock-out costs is reduced to an average of 3% of total costs by IP, whereas for FP, it is 13% on average. This observation can be explained by the feature to advance replenishments used in IP. For high holding costs, the replenishment frequency is high and inventory levels relatively low, which makes stock-outs more likely. Hence, with rising holding costs, FP performs increasingly poorly in stock-out cost share, whereas it remains almost stable for IP.

cv	0.1				0.25				0.4				avg
	LO	MI	HI	avg	LO	MI	HI	avg	LO	MI	HI	avg	
Δ_H	6.0	16.5	19.0	13.8	6.1	15.0	17.0	12.7	5.8	15.8	16.5	12.7	13.1
Δ_R	18.4	14.4	6.7	13.2	16.7	10.0	- 3.4	7.8	16.7	11.1	- 1.9	8.6	9.9
$IP FP$	03/17	03/13	03/10	03/14	03/16	02/13	02/11	02/13	02/17	02/12	03/11	02/13	03/13

Table 4.6: Savings in holding costs, routing costs, and stock-out costs for intra-day planning depending on holding costs and coefficient of variation

4.6 Conclusion

We present a new stochastic dynamic inventory routing problem that represents intra-day commodity depletion (SIRPID) at customers such as supermarkets and grocery stores. The problem at hand represents a real-world-based case in which a supplier manages the inventories of customers who experience stochastic demands during a planning horizon. Each period is divided into sub-periods with independent demands to represent the continuous-time demand consumption at customers during the planning horizon periods. Furthermore, we model additional real-world characteristics, such as delivery time windows, holding and routing costs, the possibility of lost sales, and limited inventory and vehicle fleet capacities. We formulate the problem as a finite-horizon SDP and use approximate inventory policies and an adaptive lookahead algorithm to solve the problem efficiently.

The results of our numerical evaluation show that if demand data is available with higher granularity, it helps reducing overall supply chain costs by obtaining a better replenishment plan that avoids lost sales. The proposed adaptive lookahead policy widely outperforms other approximate policies in terms of total cost on average by adapting each customer's replenishment policy and advancing urgent deliveries. Utilizing intra-day demand information allows for accurate predictions of inventory depletion throughout the planning period. By using the adaptive lookahead algorithm, the right delivery amounts and times can be found. This generates more efficient routes, lower holding costs, and fewer stockouts because the real nature of the demand is taken into account. Total costs decrease by 24.5% compared to full-day planning with aggregated demand information. We show that achieving these savings requires flexibility in planning the delivery routes to provide just-in-time deliveries. The opportunity to do so is more seldom for limited vehicle fleets or vehicle capacity, or even impossible to determine when assuming that inventory depletion happens only at the end of a period.

An attractive future research opportunity is to consider continuous time rather than dividing periods into sub-periods. This would allow us to determine replenishment quantities and timings more precisely but would result in significantly increased computational complexity. The length of the lookahead period in the lookahead algorithm is another area that could be studied. Choosing the lookahead period length dynamically could further improve the ability of the algorithm to adapt to different problem characteristics. Optimizing the ROPs and replenishment quantities without limiting them to certain replenishment policies using a data-driven approach should also produce more accurate replenishment quantities and times. However, highly-granular data is

hard to obtain. Still, the results of this research should encourage decision-makers to improve the availability and quality of data in their businesses.

Conclusion

This thesis studies multiple variants of the inventory routing problem (IRP), a subtopic of logistics and operations research. The topics presented in this manuscript advance the state-of-the-art research thematically and methodologically by considering recent characteristics related to IRPs and through extensive computational analyses. The most important characteristic considered is related to the demands that customers experience. In most of the existing literature related to the IRPs, the demands that customers experience are considered instantaneous, taking place at the beginning or end of each period. However, in many real life applications, these demands are not instantaneous events, but are consumed continuously during the period. The impact of considering this new feature has been studied considering different levels of complexity, in terms of stochasticity and non-linearity of the demands. Furthermore, additional industry-related characteristics are evaluated in this thesis, such as time windows, consistency, or the possibility of losing sales due to inefficient replenishment plans. Methodologically, this thesis adapted many well-known heuristic solution methods to solve the proposed algorithms, and developed new problem specific methods and operators.

In Chapter 2 a new IRP with continuous-time demand is introduced: the consistent inventory routing problem with time windows and split deliveries (CIRPTWSD). A new mathematical formulation is presented, where routing and replenishment decisions are managed from a central organization. A matheuristic solution approach was developed, in which an adaptive large neighborhood search algorithm is applied to iteratively destroy and repair the solutions, and the optimal replenishment times and quantities are calculated exactly by solving a mathematical subproblem. The efficiency of the matheuristic is evaluated using adapted instances from the literature, and real-world-based instances. The algorithm finds good solutions in short processing times, whereas the exact solvers are not able to solve large instances even with extended processing times.

In Chapter 3 a stochastic IRP with time-windows is examined. This problem is formulated as a two-stage mathematical problem, in which in the first stage of the problem, delivery routes, times and quantities are calculated. Then, in the second stage of the problem, the stochastic

demands that each customer faces are disclosed, and the related inventory and stock out costs are calculated. To solve this problem, three variants of a matheuristic algorithm are presented. Each of the variants of the solution approach considers different levels of simulation, considering different demand realizations during the algorithm. In addition, we evaluate the impact of considering recourse actions to replenish customers that experience a loss of sales. To prove the quality of the algorithms presented, we adapt other methods existing in the literature, the multiple scenario approach and branch and regret heuristic, and compare the obtained results. Results show that the matheuristic algorithm multi-solution stochastic ALNS reports the best overall solutions for all instance sizes. These results show that additional deliveries are of great use when minimizing the overall costs of a supply chain.

Chapter 4 investigates a new stochastic dynamic inventory routing problem in which demands that customers experience are not linear during the period. The stochastic inventory routing with dynamic demands and intra-day depletion (SIRPID) is formulated as a stochastic dynamic program in which each period of the planning horizon is divided into different subperiods with independent stochastic demands. To solve the problem, an adaptive lookahead procedure is presented. Furthermore, different approximate replenishment policies and a policy function approximation are presented. The proposed adaptive lookahead policy widely outperforms other approximate policies in terms of total cost on average by adapting each customer's replenishment policy and advancing urgent deliveries. Furthermore, we evaluate the impact of considering intra-period demand information. When solving the instances by means of the adaptive lookahead algorithm, the right delivery amounts and times can be found. This generates more efficient routes, lower holding costs, and fewer stockouts allowing a total cost decrease of 24.5% compared to full-day planning with aggregated demand information.

Bibliography

- [1] (2023). Stochastic inventory routing with dynamic demands and intra-day depletion. *Computers & Operations Research*, Resubmitted.
- [2] Absi, N., Cattaruzza, D., Feillet, D., Ogier, M., and Semet, F. (2020). A heuristic branch-cut-and-price algorithm for the roaDEF/euro challenge on inventory routing. *Transportation Science*, 54(2):313–329.
- [3] Adelman, D. (2004). A price-directed approach to stochastic inventory/routing. *Operations Research*, 52(4):499–514.
- [4] Adulyasak, Y., Cordeau, J.-F., and Jans, R. (2013). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, 26(1):103–120.
- [5] Alarcon, E. J., Schilde, M., and Doerner, K. F. (2020). Matheuristic search techniques for the consistent inventory routing problem with time windows and split deliveries. *Operation Research Perspectives*, 7.
- [6] Alarcon, E. J., Schilde, M., Malicki, S., and Doerner, K. F. (2018). Consistent inventory routing with split deliveries. *Operation Research Proceedings 2017*, pages 395–401.
- [7] Alarcon Ortega, E. J. and Doerner, K. F. (2023). A sampling-based matheuristic for the continuous-time stochastic inventory routing problem with time-windows. *Computers & Operations Research*, In Press.
- [8] Alvarez, A., Cordeau, J.-F., and Jans, R. (2021). Inventory routing under stochastic supply and demand. *Omega*, 102:102304.
- [9] Andersson, H., Hoff, A., Christiansen, M., Hasle, G., and Lokketangen, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers and Operations Research*, 37(9):1515–1536.

-
- [10] Andre, J., Bourreau, E., and Calvo, R. W. (2020). Introduction to the special section: Roadef/euro challenge 2016 — inventory routing problem. *Transportation Science*, 54(2):299–301.
- [11] Anily, S. and Federgruen, A. (1990). One warehouse multiple retailer systems with vehicle routing costs. *Management Science*, 36(1):92–114.
- [12] Archetti, C., Bertazzi, L., Laport, G., and Speranza, M. G. (2007). A branch-and-cut algorithm for a vendor-managed inventory routing problem. *Transportation Science*, 41(3):382–391.
- [13] Archetti, C., Boland, N., and Speranza, M. G. (2017a). A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing*, 29(3):377–387.
- [14] Archetti, C., Desaulniers, G., and Speranza, M. G. (2017b). Minimizing the logistic ratio in the inventory routing problem. *EURO Journal on Transportation and Logistics*, 6(4):289–306.
- [15] Archetti, C. and Speranza, M. G. (2012). Vehicle routing problems with split deliveries. *International Transactions in Operational Research*, 19:3–22.
- [16] Axsäter, S. (2001). A note on stock replenishment and shipment scheduling for vendor-managed inventory systems. *Management Science*, 47(9):1306–1310.
- [17] Azi, N., Gendreau, M., and Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763.
- [18] Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., and Prutzman, P. J. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces.*, 13(6):4–23.
- [19] Bent, R. W. and Hentenryck, P. V. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research.*, 52(6):977–987.
- [20] Bertazzi, L., Bosco, A., Guerreiro, F., and Lagana, D. (2013). A stochastic inventory routing problem with stock-out. *Transportation Research Part C: Emerging Technologies*, 27:89–107.
- [21] Bertazzi, L., Paletta, G., and Speranza, M. G. (2002). Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, 36(1):119–132.

-
- [22] Bertazzi, L., Paletta, G., and Speranza, M. G. (2005). Minimizing the total cost in an integrated vendor—managed inventory system. *Journal of Heuristics*, 11(5):393–419.
- [23] Bertazzi, L., Savelsbergh, M., and Speranza, M. G. (2008). Inventory routing. In Golden, B. L., Subramanian, R., and Wasil, E. A., editors, *The vehicle routing problem: latest advances and new challenges*, pages 49–72. Springer.
- [24] Bertazzi, L. and Speranza, M. G. (2008). Inventory routing. In: *Golden B., Raghavan R., Wasil E. (eds) The vehicle routing problem latest advances and new challenges, operations research/computer science interfaces series, Springer, Berlin*, 43:49–72.
- [25] Brinkmann, J., Ulmer, M. W., and Mattfeld, D. C. (2018). Dynamic lookahead policies for stochastic-dynamic inventory routing in bike sharing systems. *Computers and Operations Research*, 106:260–279.
- [26] Brinkmann, Jan and Ulmer, M. and Mattfeld, D. (2020). The multi-vehicle stochastic-dynamic inventory routing problem for bike sharing systems. *Business Research*, 13:69–92.
- [27] Campbell, A., Clarke, L., Kleywegt, A., and Savelsbergh, M. W. (1998). The inventory routing problem. In Crainic, T. G. and Laporte, G., editors, *Fleet management and logistics. Centre for Research on Transportation*, pages 95–113. Springer.
- [28] Campbell, A. M. and Savelsbergh, M. W. (2004a). A decomposition approach for the inventory-routing problem. *Transportation Science*, 38(4):488–502.
- [29] Campbell, A. M. and Savelsbergh, M. W. (2004b). A decomposition approach for the inventory routing problem. *Transportation Science*, 38(4):488–502.
- [30] Campelo, P., Neves-Moreira, F., Amorim, P., and Almada-Lobo, B. (2019). Consistent vehicle routing problem with service level agreements: A case study in the pharmaceutical distribution sector. *European Journal of Operational Research*, 273(1):131–145.
- [31] Çetinkaya, S. and Lee, C.-Y. (2000). Stock replenishment and shipment scheduling for vendor-managed inventory systems. *Management Science*, 46(2):217–232.
- [32] Christiansen, M. (1999). Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, 33(1):3–16.

- [33] Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C*, 24:270–287.
- [34] Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.
- [35] Coelho, L. C. and Laporte, G. (2014). Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, 155:391–397.
- [36] Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for the vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936.
- [37] Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. (2007). Vehicle routing. *Handbooks in operations research and management science*, 14:367–428.
- [38] Crama, Y., Rezaei, M., Savelsbergh, M. W. P., and Woensel, T. V. (2018). Stochastic inventory routing for perishable products. *Transportation Science*, 52(3):526–546.
- [39] Dantzig, G. B. and Ramser, J. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.
- [40] Desaulniers, G., Rakke, J. G., and Coelho, L. C. (2015). Branch-and-price-and-cut algorithm for the inventory-routing problem. *Transportation Science*, 50(3):1060–1076.
- [41] Dror, M. and Ball, M. (1987). Inventory/routing: Reduction from an annual to a short-period problem. *Naval Research Logistics (NRL)*, 34(6):891–905.
- [42] Federgruen, A. and Zheng, Y.-S. (1992). The joint replenishment problem with general joint cost structures. *Operations Research*, 40(2):384–403.
- [43] Federgruen, A. and Zipkin, P. (1984). A combined vehicle routing and inventory allocation problem. *Operations Research*, 32(5):1019–1037.
- [44] Ferruci, F., Bock, S., and Gendreau, M. (2013). A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods. *European Journal of Operational Research*, 225(1):130–141.

- [45] Gruler, A., Panadero, J., de Armas, J., Perez, J. A. M., and Juan, A. A. (2018). A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. *International Transactions in Operational Research* ., 00:1–22.
- [46] He, Y., Artigues, C., Briand, C., Jozefowicz, N., and Ngeueu, S. U. (2020). A matheuristic with fixed-sequence reoptimization for a real-life inventory routing problem. *Transportation Science*, 54(2):355–374.
- [47] Hemmelmayr, V., Doerner, K. F., Hartl, R. F., and Savelsbergh, M. W. (2009). Delivery strategies for blood products supplies. *OR Spectrum*, 31(4):707–725.
- [48] Hennig, F., Nygreen, B., Furman, K., and Song, J. (2015). Alternative approaches to the crude oil tanker routing and scheduling problem with split pick-up and split delivery. *European Journal of Operational Research*., 243(1):41–51.
- [49] Herer, Y. and Roundy, R. (1997). Heuristics for a one-warehouse multiretailer distribution problem with performance bounds. *Operations Research*, 45(1):102–115.
- [50] Hvattum, L. M., Lokketangen, A., and Laporte, G. (2007). A branch-and-regret heuristic for the stochastic and dynamic vehicle routing problem. *Networks*, 49(4):330–340.
- [51] Hvattum, L. M., Løkketangen, A., and Laporte, G. (2009). Scenario tree-based heuristics for stochastic inventory-routing problems. *INFORMS Journal on Computing*, 21(2):268–285.
- [52] Jaillet, P., Bard, J. F., Huang, L., and Dror, M. (2002). Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Science*, 36(3):292–300.
- [53] Juan, A. A., Grasman, S. E., Caceres-Cruz, J., and Bektas, T. (2014). A simheuristic algorithm for the single-period stochastic inventory-routing problem with stock-outs. *Simulation Modelling Practice and Theory* ., 46:40–52.
- [54] Kheiri, A. (2020). Heuristic sequence selection for inventory routing problem. *Transportation Science*, 54(2):302–312.
- [55] Kleywegt, A. J., Nori, V. S., and Savelsbergh, M. W. P. (2002). The stochastic inventory routing problem with direct deliveries. *Transportation Science*, 36(1):94–118.

-
- [56] Kleywegt, A. J., Nori, V. S., and Savelsbergh, M. W. P. (2004). Dynamic programming approximations for a stochastic inventory routing problem. *Transportation Science*, 38(1):42–70.
- [57] Kovacs, A. A. and Braekers, K. (2016). A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological*, 94:355–377.
- [58] Kovacs, A. A., Golden, B. L., Hart, R. F., and Parragh, S. N. (2014a). Vehicle routing problems in which consistency considerations are important: a survey. *Networks*, 64(3):192–213.
- [59] Kovacs, A. A., Parragh, S. N., and Hartl, R. F. (2014b). A template-based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks*, 63(1):60–81.
- [60] Lagos, F., Boland, N., and Savelsbergh, M. (2020). The continuous-time inventory-routing problem. *Transportation Science.*, Articles in Advance:1–25.
- [61] Lagos, F., Boland, N., and Savelsbergh, M. W. (2022). Dynamic discretization discovery for solving the continuous time inventory routing with out-and-back routes. *Computers & Operations Research*, 141.
- [62] Lee, H. L. and Whang, S. (2008). The whose, where and how of inventory control design. *Supply Chain Management, Review* 12(8):22–29.
- [63] Liu, S.-C. and Lee, W.-T. (2011). A heuristic method for the inventory routing problem with time windows. *Expert Systems with Applications.*, 38(10):13223–13231.
- [64] Malicki, S. and Minner, S. (2021). Cyclic inventory routing with dynamic safety stocks under recurring non-stationary interdependent demands. *Computers & Operations Research*, 131:105247.
- [65] Meisel, S. (2011). *Anticipatory optimization for dynamic decision making*, volume 51. Springer Science & Business Media.
- [66] Minkoff, A. S. (1993). A markov decision model and decomposition heuristic for dynamic vehicle dispatching. *Operations Research*, 41(1):77–90.
- [67] Pavone, M., Bisnik, N., and Isler, V. (2008). A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *Mobile Networks and Applications*, 14:350–364.

-
- [68] Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34:2403–2435.
- [69] Popovic, D., Vidovic, M., and Radivojevic, G. (2012). Variable neighborhood search heuristic for the inventory routing problem in fuel delivery. *Experts Systems with Applications*, 39(18):13390–13398.
- [70] Qiu, M., Fu, Z., Eglese, R., and Tang, Q. (2018). A tabu search algorithm for the vehicle routing problem with discrete split deliveries and pick-ups. *Computers and Operations Research*, 100:102–116.
- [71] Qu, W. W., Bookbinder, J. H., and Iyogun, P. (1999). An integrated inventory–transportation system with modified periodic policy for multiple products. *European Journal of Operational Research*, 115(2):254–269.
- [72] Raba, D., Estrada-Moreno, A., and Juan, A. A. (2020). A reactive simheuristic using online data for a real-life inventory routing problem with stochastic demands. *International Transactions in Operational Research*, 27(6):2785–2816.
- [73] Riley, C., Van Hentenryck, P., and Yuan, E. (2020). Real-time dispatching of large-scale ride-sharing systems: Integrating optimization, machine learning, and model predictive control. *arXiv preprint arXiv*, 2003.10942.
- [74] Roldan, R. F., Basagoiti, R., and Coelho, L. C. (2017). A survey on the inventory-routing problem with stochastic lead times and demands. *Journal of Applied Logic*, 24(A):15–24.
- [75] Saliby, E. (1990). Descriptive sampling: a better approach to monte carlo simulation. *Journal of the Operational Research Society*, 41(12):1133–1142.
- [76] Savelsbergh, M. (1985). Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305.
- [77] Schilde, M., Doerner, K. F., and Hartl, R. F. (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, 238(1):18–30.

- [78] Soeffker, N., Ulmer, M. W., and Mattfeld, D. C. (2022). Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research*, 298(3):801–820.
- [79] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.
- [80] Solomon, M. M. (2004). An ant-based approach to combinatorial optimization under uncertainty. In *4th International Workshop on Ant Colony Optimization and Swarm Intelligence. Springer Berlin / Heidelberg.*, pages 238–249.
- [81] Song, Y., Ulmer, M. W., Thomas, B. W., and Wallace, S. W. (2020). Building trust in home services- stochastic team-orienting with consistency constraints. *Transportation Science*, 54(3):823–838.
- [82] Su, Z., Wang, Z. L. Z., Qi, Y., and Benlic, U. (2020). A matheuristic algorithm for the inventory routing problem. *Transportation Science*, 54(2):330–354.
- [83] Touzout, F. A., Ladier, A.-L., and Hadj-Hamou, K. (2021). An assign-and-route matheuristic for the time-dependent inventory routing problem. *European Journal of Operational Research*, Available online.
- [84] Ulmer, M., Nowak, M., and Mattfeld, D. (2020). Binary driver-customer familiarity in service routing. *European Journal Operational Research*, 286(2):477–493.
- [85] Viswanathan, S. and Mathur, K. (1997). Integrating routing and inventory decisions in one-warehouse multiretailer multiproduct distribution systems. *Management Science*, 43(3):294–312.
- [86] Zhao, Q.-H., Chen, S., and Zang, C.-X. (2008). Model and algorithm for inventory/routing decision in a three-echelon logistics system. *European Journal of Operational Research*, 191(3):623–635.

Abstract

The hereby presented PhD thesis deals with variations of the inventory routing problem (IRP), which originates in the field of transportation logistics. The IRP resembles vendor-managed inventory system, in which a central supplier ages both replenishment and distribution decisions to replenish the inventory of a set of customers or retailers. This research thesis on IRPs investigates topics and mathematical formulations that incorporate existing and new characteristics for these problems. In particular, the conductive thread of this thesis is the importance of obtaining efficient delivery plans when the customers experience demands continuously within the periods.

Most of the existing literature related to the IRPs considers that demands occur instantaneously at the end or beginning of each period. However, many real-world businesses experience demands within the periods and, as a result, inefficient delivery plans and late deliveries can translate into a loss of sales due to stock-out situations.

In this context, the first topic originates in the beverage industry, where business customers such as restaurants, bars, and supermarkets experience demand continuously during the periods. In addition, these customers present business-specific characteristics such as time windows and the need for consistency in the delivery times. We study the consistent inventory routing problem with time windows and split deliveries (CTIRPTWSD) for which we provide a mathematical formulation and a matheuristic solution approach. Furthermore, an extensive computational analysis is performed to test the efficiency of the proposed algorithm.

The second topic of the thesis deals with a stochastic variant of the continuous-time IRP. In this case, the research focus of the article is the evaluation of robust solutions when the demands that the customers experience are, not only continuous within the time periods, but also stochastic. Many real-world businesses create delivery plans at the beginning of the planning horizon, considering different levels of stochasticity within their planning. The continuous-time stochastic IRP with time windows (CTSIRPTW) is formulated as a two-stage mathematical program. To solve this problem, we present different sampling-based matheuristic solution approaches and evaluate the impact of applying recourse actions to deal with expected lost sales during the planning horizon.

Finally, the third topic deals with the stochastic inventory routing problem with intra-day depletion under dynamic demands (SIRPID). This stochastic dynamic inventory routing problem focuses on the stochastic and dynamic nature of demands which, in real-world applications, differ on both a daily and intra-day basis. Therefore, to account for the non-linearity of the demands that customers experience during the periods, we perform an a-priori discretization of each period into different subperiods. We formulate the problem as a finite-horizon stochastic dynamic program and propose an iterative look-ahead solution approach to solve the problem. Then, the performance of the proposed algorithm is evaluated with respect to different benchmark replenishment policies and a policy function approximation algorithm. The results support the assumption that, when the demand consumption that the customers experience is not continuously linear during the periods, utilizing intra-day demand information allows the central supplier to create more efficient replenishment schedules when compared to a full-day planning with aggregated demand information.

Zusammenfassung

Die vorliegende Dissertation befasst sich mit Variationen des Inventory Routing Problems (IRP), welches aus dem Bereich der Transportlogistik stammt. Das IRP ähnelt einem lieferantengesteuerten Inventarsystem, bei dem ein zentraler Lieferant sowohl Nachschub- als auch Distributionsentscheidungen trifft, um den Bestand einer Reihe von Kunden oder Einzelhändlern aufzufüllen. In dieser Dissertation über IRPs werden Themen und mathematische Formulierungen untersucht, die bestehende und neue Eigenschaften dieser Probleme berücksichtigen. Der zentrale Fokus dieser Arbeit ist insbesondere die Wichtigkeit effizienter Lieferpläne, wenn die Kunden innerhalb der Perioden kontinuierliche Nachfragen erhalten.

Der Großteil der vorhandenen Literatur zu IRPs geht davon aus, dass die Nachfrage sofort am Ende oder am Anfang jeder Periode auftritt. In vielen realen Unternehmen tritt die Nachfrage jedoch innerhalb der Perioden auf, so dass ineffiziente Lieferpläne und verspätete Lieferungen zu Umsatzeinbußen aufgrund von Stock-Out-Situationen führen können.

In diesem Zusammenhang hat das erste Thema seinen Ursprung in der Getränkeindustrie, wo Geschäftskunden wie Restaurants, Bars und Supermärkte während der Perioden eine kontinuierliche Nachfrage verzeichnen. Darüber hinaus weisen diese Kunden geschäftsspezifische Merkmale wie Zeitfenster und die Notwendigkeit konsistenter Lieferzeiten auf. Wir untersuchen das consistent inventory routing problem with time windows and split deliveries (CTIRPTWSD), für das wir eine mathematische Formel aufstellen.

Das zweite Thema der Dissertation beschäftigt sich mit einer stochastischen Variante des continuous-time IRP. In diesem Fall liegt der Forschungsschwerpunkt des Themas auf der Bewertung robuster Lösungen, wenn die Anforderungen der Kunden nicht nur innerhalb der Zeiträume kontinuierlich, sondern auch stochastisch sind. Viele reale Unternehmen erstellen Lieferpläne zu Beginn des Planungshorizonts und berücksichtigen dabei verschiedene Stufen der Stochastizität. Das continuous-time stochastic IRP with time windows (CTSIRPTW) wird als zweistufiges mathematisches Programm formuliert. Zur Lösung dieses Problems stellen wir verschiedene stichprobenbasierte mathematische Lösungsansätze vor und bewerten die Auswirkungen der

Anwendung von Regressmaßnahmen zur Bewältigung erwarteter Umsatzeinbußen während des Planungshorizonts.

Das dritte und letzte Thema befasst sich mit dem stochastic inventory routing problem with intra-day depletion under dynamic demands (SIRPID). Dieses stochastische dynamische Bestandsführungsproblem konzentriert sich auf die stochastische und dynamische Natur der Nachfrage, die in realen Anwendungen sowohl täglich als auch innerhalb eines Tages unterschiedlich ist. Um der Nichtlinearität der Nachfrage der Kunden während der Perioden Rechnung zu tragen, führen wir daher eine a-priori-Diskretisierung jeder Periode in verschiedene Unterperioden durch. Wir formulieren das Problem als stochastisches dynamisches Programm mit endlichem Horizont und schlagen einen iterativen Lösungsansatz mit Vorausschau vor, um das Problem zu lösen. Anschließend wird die Leistung des vorgeschlagenen Algorithmus im Hinblick auf verschiedene Benchmark-Auffüllstrategien und einen Algorithmus zur Approximation der Strategiefunktion bewertet. Die Ergebnisse stützen die Annahme, dass, wenn der Nachfragekonsum der Kunden während der Perioden nicht kontinuierlich linear ist, die Nutzung von Intra-Day-Nachfrageinformationen dem zentralen Lieferanten erlaubt, effizientere Wiederbeschaffungspläne zu erstellen, verglichen mit einer ganztägigen Planung mit aggregierten Nachfrageinformationen.