MULTI-COMPARTMENT INVENTORY ROUTING PROBLEM WITH ADJUSTABLE COMPARTMENT SIZES

A Thesis

by

Ömer Berk Ölmez

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Industrial Engineering

Özyeğin University
June 2021

MULTI-COMPARTMENT INVENTORY ROUTING PROBLEM WITH ADJUSTABLE COMPARTMENT SIZES

Approved by:

Associate Professor Ali Ekici, Advisor
Department of Industrial Engineering
Özyeğin University

Assistant Professor İhsan Yanıkoğlu
Department of Industrial Engineering
Özyeğin University

Associate Professor Okan Örsan Özener, Co Advisor
Department of Industrial Engineering
Özyeğin University

Associate Professor Ertan Yakıcı
Department of Industrial Engineering
National Defence University

Date Approved: 10 June 2021

Assistant Professor Mehmet Önal
Department of Industrial Engineering
Özyeğin University

To my beloved family and friends...

ABSTRACT

In this study, we focus on a problem where the supplier manages the customers' inventories and coordinates the distribution of multiple products to customers. Hereby the supplier can minimize the transportation costs by determining distribution routes, frequency of visits and distribution amounts simultaneously in a way that customer demands are satisfied on time. Customers place orders for each day and each product without following any pattern and have separate storage capacities for each product. Multi-compartment vehicles are used to enable different products to be distributed on a single route. We assume to have a fleet of vehicles with a certain number of unit compartments and the capacity dedicated to each product on a route can be adjusted discretely with the use of separators. This compartment structure provides the supplier with flexibility in making distribution plans, while on the other hand makes the problem more challenging since the capacity dedicated to each product on each route emerges as an additional decision to be made. To solve the proposed multi-compartment inventory routing problem we develop a novel Adaptive Large Neighborhood Search based matheuristic. We compare our performance with a benchmark algorithm we adapt from the literature by using an extensive set of instances. We observe that our solution approach outperforms the benchmark algorithm by 24.57% on average.

ÖZETÇE

Bu çalışmada, tek bir tedarikçinin müşterilerin envanterini yönettiği ve birden fazla ürünün dağıtımını koordine ettiği bir problem üzerine yoğunlaşmaktayız. Böylece tedarikçi müşterilerin talepleri zamanında karşılanacak şekilde dağıtım rotalarını, ziyaretlerin sıklığını ve dağıtım miktarlarını eş zamanlı bir şekilde belirleyerek ulaşım maliyetini en azlayabilmektedir. Müşteri talepleri her bir gün ve ürün için rastgele oluşturulmuştur ve müşteriler her bir ürün için belirli bir stoklama kapasitesine sahiptir. Farklı ürünlerin tek rotada dağıtımına imkan sağlayan çok bölmeli araçlar kullanılmıştır. Her araç belirli sayıda birim bölmeden oluşur ve araçlarda bulunan ayraçlar kullanılarak her bir rotada her bir ürüne ayrılacak olan kapasite kesintili bir şekilde ayarlanabilmektedir. Bu bölme yapısı tedarikçiye, dağıtım planı yaparken esneklik tanısa da öte yandan her bir rotada her bir ürüne ayrılacak kapasite, verilmesi gereken ekstra bir karar olarak karşımıza çıktığı için problemi zorlaştırmaktadır. Bu çok bölmeli envanter rotalama problemini çözmek için Uygulanabilir Geniş Komşuluk Araması tabanlı bir mat-sezgisel yöntem sunmaktayız. Geniş çaplı bir dizi veri kümesi kullanarak literatürden uyarladığımız bir kıyaslama algoritması ile algoritmamızı karşılaştırdık. Çözüm yöntemimizin bulduğu sonuçların kıyaslama algoritmasının bulduğu sonuçlardan ortalamada %24.57 daha iyi olduğu gözlenmiştir.

ACKNOWLEDGEMENTS

I would like to express my great appreciation to my advisors, Dr. Ali Ekici and Dr. O. Örsan Özener, for their consistent support and guidance. It is a great honor and privilege to study under their guidance. I would also like to extend my sincere gratitude to Dr. Burcu Balçık and Dr. İhsan Yanıkoğlu. Their guidance helped me in all the time of research and study.

I am grateful to my better half and also my colleague Ceren. Her intelligence and endless energy are my biggest supporters in life. I would like to thank my officemates, Ahmet, Birce, Eren, Kaan, and Merve for their support, encouragement, and friendship.

Finally, I owe thanks to my parents Ömer and Leyla, my sisters Özge and Ece for their support and encouragement at every stage of my education, and my niece Belis and nephew Ali for the joy they bring in my life.

This research is supported in part by TÜBİTAK grant 119M245.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER I

INTRODUCTION

Logistics management is crucial in terms of ensuring customer satisfaction by delivering the right amount of quantity on time with the minimum possible cost. Vendor-managed inventory is a valuable logistics management strategy, where the supplier determines when and how much of each product should be resupplied to each customer location. A vendor-managed inventory routing setting brings along coordination in distribution and is advantageous for all parties, since while the supplier can reduce the transportation and inventory costs, the customers are less likely to encounter situations where their demands are not met on time, plus they do not need to allocate resources for the inventory management.

We examine a variant of the vendor-managed inventory routing problem (IRP), where the supplier is in charge of optimizing the distribution of multiple products to customers for a multi-period planning horizon. The customers place orders for different products during the planning horizon with no certain pattern. Therefore, distribution needs may emerge at different times for different products. Customers have a limit on the amount of each product they can store, so distribution plans should be made accordingly. Using multi-compartment vehicles enables the distribution of different products on a single tour. When the vehicle capacity can be shared among different products, it becomes cheaper to meet the customers' unstable demands for different products. In this study, we examine the Multi-Compartment Inventory Routing Problem (MCIRP), where we use a fleet of multi-compartment vehicles to distribute various products to customers by ensuring that the daily demand of each customer places for each product is met with the minimum total transportation cost.

While in some applications of the MCIRP compartment capacities dedicated to products are assumed to be fixed, we assume that the capacity dedicated for each product can be adjusted discretely. In other words, a vehicle consists of a certain number of unit compartments and the number of unit compartments allocated to each product is a decision to be made. The configurable compartment structure provides the supplier with flexibility in meeting the inconsistent demands of the customers for different products. A compartment can be loaded with one type of product at a time, and the number of compartments dedicated to each product can be different for any route. This flexible compartment structure may be benefited in many real-world operations, such as distributing food to groceries [1, 2, 3] and collecting different colored glass waste to be recycled [4].

Since the vehicle routing problem is NP-hard [5], MCIRP falls into the class of NP-hard problems as well. Dealing with inventory management and routing simultaneously within a multi-period time horizon is already challenging even in the presence of a single product. As for MCIRP, since there are multiple products and the customers consume products at different rates, the supplier may need to visit the same customer at different times for different products, which further complicates the problem. On top of that, when we consider that the supplier also needs to determine the capacities dedicated to each product on each route, the solution space of our problem becomes even much larger.

To obtain a good solution for this challenging problem in a reasonable time, we use mathematical models within an Adaptive Large Neighborhood Search (ALNS) framework. Matheuristics arise as the integration of the mathematical models into metaheuristics and are popular in solving many routing problems [6, 7, 8], yet there exists no matheuristic proposed to solve the MCIRP. ALNS enables us to search the solution space in a systematic fashion while integrating mathematical models helps us make optimal decisions within the algorithm. Based on different assumptions in

the literature in terms of whether to allow split delivery to customers (i.e., allowing a customer to be visited by different vehicles on a day/period) we design our algorithm in three different ways while (i) a customer can only be visited by one vehicle on a day/period, (ii) a customer can be visited by different vehicles for different products on a day/period, but one product type can be distributed to a customer by a single vehicle, (iii) any product can be distributed to a customer by different vehicles on a day/period. The use of mathematical models also helps us modify the algorithm easily for these three assumptions since we only need to make slight adjustments to them.

There exist a branch of studies in the maritime transportation dealing with the MCIRP, which is called the Maritime Inventory Routing Problem (MIRP). However, maritime operations differ from the land operations in many aspects, therefore so do the routing and the scheduling problems studied in these areas [9, 10]. The most significant differences between MCIRP and the MIRP can be summarized as; in maritime transportation, (i) as in customer nodes, there are multiple supply nodes with certain storage capacities, (ii) vessels travel for days and visit only a couple of nodes where they unload a great portion of the vehicle, whereas inland transportation, vehicles unload a smaller portion of products at customer nodes, thus many customers can be visited on a route, (iii) on contrary to the homogeneous vehicle fleet used in the land transportation, vessels usually differ in size, compartment structure and cost. Due to these differences, throughout the paper, we only refer to the multi-compartment studies in the land transportation.

To the best of our knowledge, there exists no multi-period routing problem adopting the flexible compartment structure in this study. The first contribution of this study is that we are the first to elaborate the MCIRP assuming that the capacities dedicated to each product can be adjusted discretely in terms of the unit compartment capacity. And secondly, we are the first to propose a matheuristic to solve the

MCIRP.

We test the performance of our algorithm on a set of instances we generate by adapting from [11]. We deem it appropriate to use the algorithm proposed by [1] as our benchmark. Similar to us, they assume that compartment capacities can be adjusted discretely on each route depending on the distribution needs, however, they approach the problem for a single period as the Multi-Compartment Vehicle Routing Problem (MCVRP). To make a fair comparison, we solve their algorithm for every day of the planning horizon separately. We obtain 24.57% better results on average compared to their algorithm.

CHAPTER II

LITERATURE REVIEW

In this section, we review the multi-compartment routing studies covering both single-period and multi-period planning horizons. In [12], authors present the most recent literature review on the MCVRP studies with a focus on the different compartment structures, delivery assumptions, and solution approaches. We observe that multi-compartment routing studies can be examined in three groups in terms of how the compartments are structured in a vehicle as follows;(i) compartments have fixed capacities and are dedicated to products, (ii) compartments have fixed capacities and they are not dedicated to products, (iii) compartments have flexible capacities and are dedicated to products.

The first group of multi-compartment routing studies is only examined under the single period assumption, meaning that no inventory decisions are addressed. Here, compartment capacities are assumed to be fixed and compartments are assigned some certain products meaning that they can only be loaded with those products. This group of studies has real-life applications in the feed distribution to livestock farms [13], distribution of foods with different temperature needs to groceries [14], and the collection of different recyclable wastes [15, 16]. In [13], authors propose two algorithms to solve MCVRP; a memetic algorithm combined with a path relinking method and a tabu search method. In [15] and [16], authors use an ant colony system to solve the MCVRP and report high-quality solutions with two-compartment vehicles. In [17], authors study an MCVRP where they respect unloading times and propose an iterated tabu search algorithm to solve the problem. In [18], authors study a multi-depot MCVRP and develop a hybrid solution method using ALNS and

Variable Neighborhood Search (VNS). In [19], authors present a new local search procedure which is a combination of local search moves such as 2-opt, relocate, exchange and cross to solve the MCVRP.

The second group of studies contributes to the literature by providing examples of the MCIRP. Here, compartments are assumed to have fixed capacities and they are not assigned to any products, meaning that any compartment can be loaded with any product. Compared to the first group of studies, compartments being not dedicated to any product provides flexibility to the supplier in making distribution plans when compartments have different capacities. For example, in the first group of studies, if one compartment has a relatively lower capacity, the product dedicated to that compartment must always be delivered in relatively small amounts. However, when compartments can carry any product, this is no longer a must. Yet, the assignment of products to compartments on each route arises as a new decision to be made. The multi-compartment routing studies with this compartment structure have real-life applications in the collection of different quality olive oils from the manufacturer [20], and the distribution of different fuel types to gas stations [11, 21, 22, 23, 24, 25].

In line with the second group of studies, when it comes to the distribution of liquid products, some studies assume that vehicles are equipped with debit meters [20], which allow them to split the content in a compartment among several customers. Whereas, it is more common to ignore the presence of debit meters to deal with a simpler problem [11, 21, 24, 26, 27, 28, 29]. These studies assume that when a vehicle visits a customer to deliver a product, the content in the respective compartment must be emptied at that customer node. It means that the content in a compartment cannot be split among different customers and partial deliveries are not allowed. This assumption simplifies the problem since the route size is limited by the number of compartments in a vehicle.

Next, we review how the MCIRP studies with the fixed-capacity and undedicated

compartment structure are handled in the literature. In [11, 21, 24], the authors narrow the feasible region by limiting the number of customers to be visited on a route. In [11] and [21], the authors address almost the same problem where they assume that the compartment capacities are equal and the periodic demand of each customer for each product is constant. The only difference in [21], is that they assume there are two different vehicle types with two and four compartments. In [11], the authors propose a VNS to solve the problem, while in [21], authors solve it by a variable neighborhood descent algorithm. In [24], the authors propose an exact solution method where they decompose the problem and handle the truck loading and routing decisions separately. Similar to [24], in [29] and [30], the authors decompose the problem to avoid the challenges of handling all the decisions simultaneously. In [29], the authors consider late deliveries and in their solution method, they first obtain the optimal order quantities and time windows by using an inventory model which aims to minimize the total expected costs emerge in the customer nodes, then they use this information in determining vehicle routes by using mixed-integer models. They use a VNS framework to enhance the distribution routes. In [30], in the first stage, the authors generate a set of multi-period routes by using a column generation approach and in the second stage, they solve the problem by providing these routes to a mixed-integer linear programming model. In [22], the authors handle the MCIRP both with and without debit meters, as well as considering both allowing and disallowing split deliveries to customers. They solve the problem with a branch-and-cut exact solution method. Finally, in [31], the authors study an application of the MCIRP where animals are collected from farms and transported to slaughterhouses under very restricted conditions with respect to health and hygiene rules. They use a column generation based exact solution method.

The third group of multi-compartment routing studies is also examined only under the single period assumption as MCVRPs. The compartment structure adopted in

this group of studies is the same as we consider in this study, where compartments consist of a certain number of unit compartments and with the use of separators, the total capacity (i.e., the total number of unit compartments) dedicated to each product can be adjusted discretely. In [1], the authors study the problem by considering the loading and unloading costs arising from the flexible compartments and develop a Large Neighborhood Search (LNS) algorithm by defining the problem-specific removal and insertion operators. In [2], the authors study a similar problem and develop an LNS algorithm to determine the vehicle types to be used among single compartment and multi-compartment vehicles since the loading and unloading costs depend on the compartment structures of the vehicles.

To the best of our knowledge, we are the first to address the challenges of the MCIRP with this challenging, yet very useful in practice, compartment structure. With our novel matheuristic approach, we believe that we bridge an existing gap in the literature by effectively solving the MCIRP by providing the supplier with flexibility in adjusting compartment capacities depending on the distribution needs.

CHAPTER III

PROBLEM DEFINITION AND FORMULATION

The MCIRP can be defined on a network with a single supplier (i.e., depot) and a set of customers, each with deterministic demands that can vary over product and/or period basis. On each period over the planning horizon, a set of identical multi-compartment vehicles perform routes between the depot and the customers, to distribute their needs in a way that no customer faces stock-outs in any product on any period. Customers can keep inventory up to a certain level specified for each product. Considering that the working hours of the drivers cannot be exceeded, the length of each route is restricted by an upper bound. Capacities dedicated to each product on a vehicle can be adjusted depending on the distribution needs. Each vehicle consists of a certain number of unit compartments. Herein, the supplier needs to determine how many of these compartments to allocate for each product on each route. Overall, the MCIRP examined in this study deals with determining the following for each period; (i) amount of products to be distributed to the customers, (ii) routes to be performed, (iii) inventory levels of the customers, and the (iv) allocation of the vehicle capacity among products. We examine this problem under three different distribution assumptions; (i) each customer can only be visited by one vehicle on a period, (ii) each customer is allowed to be visited by one vehicle for each product on a period, (iii) each customer can be visited by more than one vehicle on a period. We formulate the problem as a mixed-integer programming (MIP) model. We explain how to modify the model for each of these assumptions. The notation and variables used in these models are provided in Table 1.

Table 1: Notation and variables used in MIP

Sets

V_0 Set of customer locations ($V = \{1, \ldots, n\}$), and the depot (0)

$\mathcal{T}$ The planning horizon ($\mathcal{T} = \{1, \ldots, T\}$)

P Set of products ($P = \{1, \ldots, m\}$)

K Set of vehicles

Parameters

b_{ij} Cost of traversing from node i to j $\forall i \in V_0, j \in V_0$

d_{ipt} The demand of customer i from product p on day t $\forall i \in V, p \in P, t \in \mathcal{T}$

L Maximum route length

$\mathcal{J}^0_{ip}$ Initial inventory level of customer i from product p $\forall i \in V, p \in P$

C_{ip} Storage capacity of customer i for product p $\forall i \in V, p \in P$

B Unit compartment capacity

H Total number of compartments in a vehicle

Decision Variables

q_{iptk} Amount of product p delivered to customer i on day t by vehicle k $\forall i \in V, p \in P, t \in \mathcal{T}, k \in K$

x_{ijtk} 1, if vehicle k travels from node i to node j on day t; 0, otherwise $\forall i, j \in V_0, t \in \mathcal{T}, k \in K$

I_{ipt} Amount of product p in the inventory at customer i on day t $\forall i \in V, p \in P, t \in \mathcal{T}$

y_{iptk} 1, if vehicle k delivers product p to customer i on day t; 0, otherwise $\forall i \in V, p \in P, t \in \mathcal{T}, k \in K$

z_{itk} 1, if vehicle k visits customer i on day t; 0, otherwise $\forall i \in V, t \in \mathcal{T}, k \in K$

o_{ptk} Number of unit compartments that product p is loaded with on vehicle k on day t $\forall p \in P, t \in \mathcal{T}, k \in K$

First, we present the model assuming that each customer can be visited by multiple vehicles on a period.

$$\text{MIP: min} \quad \sum_{i\in V_0}\sum_{j\in V_0}\sum_{t\in\mathcal{T}}\sum_{k\in K} b_{ij}x_{ijtk} \tag{1}$$

$$st. \quad \mathcal{J}^0_{ip} + \sum_{k\in K} q_{ip1k} = I_{ip1} + d_{ip1} \qquad \forall i \in V, p \in P \tag{2}$$

$$I_{ip,t-1} + \sum_{k\in K} q_{iptk} = I_{ipt} + d_{ipt} \qquad \forall i \in V, p \in P, t \in \mathcal{T}\backslash\{1\} \tag{3}$$

$$\sum_{i\in V} q_{iptk} \leq Bo_{ptk} \qquad \forall p \in P, t \in \mathcal{T}, k \in K \tag{4}$$

$$\sum_{i\in V_0}\sum_{j\in V_0} b_{ij}x_{ijtk} \leq L \qquad \forall t \in \mathcal{T}, k \in K \tag{5}$$

$$I_{ip,t-1} + \sum_{k\in K} q_{iptk} \leq C_{ip} \qquad \forall i \in V, p \in P, t \in \mathcal{T}\backslash\{1\} \tag{6}$$

$$\mathcal{J}^0_{ip} + \sum_{k\in K} q_{ip1k} \leq C_{ip} \qquad \forall i \in V, p \in P \tag{7}$$

$$\sum_{i,j\in S} x_{ijtk} \leq |S| - 1 \qquad \forall t \in \mathcal{T}, k \in K, S \subseteq V, |S| \geq 2 \tag{8}$$

$$q_{iptk} \leq C_{ip}y_{iptk} \qquad \forall i \in V, p \in P, t \in \mathcal{T}, k \in K \tag{9}$$

$$y_{iptk} \leq \sum_{j\in V_0} x_{jitk} \qquad \forall i \in V, p \in P, t \in \mathcal{T}, k \in K \tag{10}$$

$$\sum_{i\in V_0} x_{ijtk} = \sum_{i\in V_0} x_{jitk} \qquad \forall j \in V_0, t \in \mathcal{T}, k \in K \tag{11}$$

$$H \geq \sum_{p\in P} o_{ptk} \qquad \forall t \in \mathcal{T}, k \in K \tag{12}$$

$$q_{iptk} \geq 0 \qquad \forall i \in V, p \in P, t \in \mathcal{T}, k \in K \tag{13}$$

$$x_{ijtk} \in \{0, 1\} \qquad \forall i, j \in V_0, t \in \mathcal{T}, k \in K \tag{14}$$

$$I_{ipt} \geq 0 \qquad \forall i \in V, p \in P, t \in \mathcal{T} \tag{15}$$

$$y_{iptk} \in \{0, 1\} \qquad \forall i \in V, p \in P, t \in \mathcal{T}, k \in K \tag{16}$$

$$o_{ptk} \geq 0 \textit{ and integer} \qquad \forall p \in P, t \in \mathcal{T}, k \in K \tag{17}$$

The objective function (1) aims to minimize the total travel distance. Constraints

(2) and (3) are the inventory flow-balance constraints. Constraints (4) ensure that the total amount of a product distributed to the customers by a vehicle on any period, cannot exceed the total capacity allocated for that product. Constraints (5) set an upper limit for the total distance traveled by each route. Constraints (6) and (7) ensure that the storage capacities cannot be exceeded for any customer-product pair on any period. Constraints (8) are the sub tour elimination constraints. Constraints (9) and (10) determine whether a distribution is performed to a customer on a route. Constraints (11) force each vehicle visiting a customer to also leave that customer. Constraints (12) ensure that the total number of compartments used for all products on a vehicle cannot exceed the overall vehicle capacity. Constraints (13)-(17) are either the nonnegativity or the integrality constraints.

If we assume that each customer is allowed to be visited by one vehicle for each product on a period, we need to include Constraints (18) into the MIP.

$$\sum_{k \in K} y_{iptk} \leq 1 \qquad \forall i \in V, p \in P, t \in \mathcal{T} \tag{18}$$

If we assume that each customer can only be visited by one vehicle on a period, then we need to include Constraints (19)-(22) into the MIP.

$$q_{iptk} \leq C_{ip} z_{itk} \qquad \forall i \in V, p \in P, t \in \mathcal{T}, k \in K \tag{19}$$

$$\sum_{k \in K} z_{itk} \leq 1 \qquad \forall i \in V, t \in \mathcal{T} \tag{20}$$

$$\sum_{j \in V_0} (x_{ijtk} + x_{jitk}) \leq 2 z_{itk} \qquad \forall i \in V, t \in \mathcal{T}, k \in K \tag{21}$$

$$z_{itk} \in \{0, 1\} \qquad \forall i \in V, t \in \mathcal{T}, k \in K \tag{22}$$

CHAPTER IV

SOLUTION APPROACH

4.1 Initial Feasible Solution

We needed a computationally efficient and well-performing method to find an initial feasible solution to our matheuristic approach. In this regard, to save the model from building routes, we formulate the MIP-IFS, which takes a set of routes generated in advance as an input. MIP-IFS aims to determine which routes to be used on each period and how much of each product to be distributed to customers on each route while minimizing the total length of the routes performed. Here, the larger the size of the set of routes generated, the more likely the model will find a better objective function value. As a matter of fact, if the MIP-IFS can be solved with all the possible routes provided, it finds the optimal solution for the problem. However, the number of all possible routes grows exponentially as the number of customers increases, therefore it is not efficient to generate all possible routes (unless the instance size is small enough). Hence, in the route set we generate, we include all the feasible routes with sizes one and two and some routes (the feasible ones among 1000 randomly generated routes) with sizes three to seven. Here, the feasibility of the routes is ensured by keeping the route length within the upper limit (L). Each route in the route set should be TSP-optimal, meaning that the routes should navigate between the depot and the customers with the minimum cost. To obtain the TSP-optimal routes, we use the Lin-Kernighan algorithm [32], which is a prevalent approach in the literature.

Table 2 introduces the new sets and parameters required to formulate the MIP-IFS, as well as presents the decision variables used in the model.

Table 2: Notation and variables used in MIP-IFS

Sets

R Set of routes generated

R^i Among the set of routes generated, the ones that visit customer i $\forall i \in V$

Parameters

α_{ir} 1, if route r visits customer i; 0, otherwise $\forall i \in V, r \in R$

L_r Length of route r $\forall r \in R$

Decision Variables

q_{iprt} Amount of product p distributed to customer i on route r on day t $\forall i \in V, p \in P, r \in R, t \in \mathcal{T}$

x_{rt} 1, if route r is performed on day t; 0, otherwise $\forall r \in R, t \in \mathcal{T}$

I_{ipt} Amount of product p in the inventory at customer i on day t $\forall i \in V, p \in P, t \in \mathcal{T}$

y_{iprt} 1, if route r is used to distribute product p to customer i on day t; 0, otherwise $\forall i \in V, p \in P, r \in R, t \in \mathcal{T}$

w_{prt} Number of unit compartments that product p is loaded with on route r on day t $\forall p \in P, r \in R, t \in \mathcal{T}$

First, we present the model assuming that each customer can only be visited by one vehicle on a period.

$$\text{MIP-IFS:} \quad \min \quad \sum_{t \in \mathcal{T}} \sum_{r \in R} L_r x_{rt} \tag{23}$$

$$st. \quad \sum_{r \in R^i} x_{rt} \leq 1 \qquad \forall i \in V, t \in \mathcal{T} \tag{24}$$

$$I_{ip,t-1} + \sum_{r \in R} q_{iprt} = I_{ipt} + d_{ipt} \qquad \forall i \in V, p \in P, t \in \mathcal{T} \backslash \{1\} \tag{25}$$

$$\mathcal{J}^0_{ip} + \sum_{r \in R} q_{ipr1} = I_{ip1} + d_{ip1} \qquad \forall i \in V, p \in P \tag{26}$$

$$I_{ip,t-1} + \sum_{r \in R} q_{iprt} \leq C_{ip} \qquad \forall i \in V, p \in P, t \in \mathcal{T} \backslash \{1\} \qquad (27)$$

$$\mathcal{J}^0_{ip} + \sum_{r \in R} q_{ipr1} \leq C_{ip} \qquad \forall i \in V, p \in P \qquad (28)$$

$$\sum_{i \in V} q_{iprt} \leq B w_{prt} \qquad \forall p \in P, r \in R, t \in \mathcal{T} \qquad (29)$$

$$\sum_{i \in V} \sum_{p \in P} \sum_{t \in \mathcal{T}} q_{iprt}(1 - \alpha_{ir}) = 0 \qquad \forall r \in R \qquad (30)$$

$$\sum_{p \in P} w_{prt} \leq H x_{rt} \qquad \forall r \in R, t \in \mathcal{T} \qquad (31)$$

$$q_{iprt} \geq 0 \qquad \forall i \in V, p \in P, r \in R, t \in \mathcal{T} \qquad (32)$$

$$x_{rt} \in \{0, 1\} \qquad \forall r \in R, t \in \mathcal{T} \qquad (33)$$

$$I_{ipt} \geq 0 \qquad \forall i \in V, p \in P, t \in \mathcal{T} \qquad (34)$$

$$w_{prt} \geq 0 \text{ } \textit{and integer} \qquad \forall p \in P, r \in R, t \in \mathcal{T} \qquad (35)$$

The objective function (23) aims to minimize the total length of the routes performed. Constraints (24) ensure that each customer can only be visited by one route on a period. Constraints (25) and (26) are the inventory flow-balance constraints. Constraints (27) and (28) ensure that the storage capacities cannot be exceeded for any customer-product pair on any period. Constraints (29) ensure that the total amount of a product distributed to the customers on a route on any period, cannot exceed the total capacity allocated for that product. Constraints (30) ensure that if a customer is not visited on a route, there cannot be any product distributed to that customer on that route. Constraints (31) ensure that if a route is performed on a period, the total number of compartments used for all products cannot exceed the overall vehicle capacity. Constraints (32)-(35) are either the nonnegativity or the integrality constraints.

If we assume that each customer is allowed to be visited by one vehicle for each product on a period, we need to exclude Constraints (24) from the MIP-IFS, and

include Constraints (36) and (38) into it.

$$q_{iprt} \leq BH y_{iprt} \qquad \forall i \in V, p \in P, r \in R, t \in \mathcal{T} \tag{36}$$

$$\sum_{r \in R} y_{iprt} \leq 1 \qquad \forall i \in V, p \in P, t \in \mathcal{T} \tag{37}$$

$$y_{iprt} \in \{0, 1\} \qquad \forall i \in V, p \in P, r \in R, t \in \mathcal{T} \tag{38}$$

If we assume that each customer can be visited by multiple vehicles on a period, then it is sufficient to exclude Constraints (24) from the MIP-IFS.

4.2 Adaptive Large Neighborhood Search

Adaptive Large Neighborhood Search (ALNS), introduced by [33], is an extension of the Large Neighborhood Search (LNS) metaheuristic which is first proposed by [34]. ALNS allows the use of multiple destroy and repair methods within the same search process rather than using one large neighborhood as in LNS.

In the ALNS proposed in [33], at each iteration, the algorithm destroys the current solution, s, using a destroy operator and repairs it differently with a repair operator to find a unique solution s'. Afterward, it applies an improvement operator to s' to generate a better solution s'', and it accepts or rejects the obtained solution according to some acceptance criteria. Every destroy, repair, and improvement operator has a weight that the algorithm adjusts at the end of every segment.

We adapt the ALNS structure proposed in [7] and [35] to solve the MCIRP. In this ALNS variation, unlike the ALNS introduced previously, the algorithm only selects destroy operators by using a roulette-wheel mechanism, and it applies repair and improvement operators in a given order for every iteration. In other words, only the destroy operators have weights that the algorithm adjusts at the end of every segment. This ALNS adaptation, as in many other ALNS adaptations, uses the simulated annealing (SA) method as the acceptance criteria to consider including worse solutions found in each iteration into the algorithm. However, in this study, we

observe that not accepting any worse solutions gives better results when solving the MCIRP. We check the feasibility of every solution that is obtained after applying a destroy operator because a feasible solution does not need to enter a repair operator. We use a linear programming model (LP-S Model) to check whether the solution is feasible or not. The LP-S Model takes routing and compartment size information as parameters and minimizes stock-out amounts of every customer over the time horizon. If the objective function of this model is zero, it means that the current solution is feasible for our problem. If it is more than zero, it means it is not feasible with the current route and compartment sizes.

For the following LP-S Model, we introduce a new decision variable S_{ipt} that denotes the amount of stock-out customer i has in product p at the end of the day t. We also introduce β_{irt}, which is a binary parameter indicating whether customer i is visited by route r on the day t. R_t denotes the set of routes that are performed on the day t. Finally, w_{prt} is a parameter in this model, indicating the number of unit compartments dedicated to the product p on route r, on the day t in the current solution.

$$\text{LP-S:} \quad \min \quad \sum_{i\in V}\sum_{p\in P}\sum_{t\in\mathcal{T}} S_{ipt} \tag{39}$$

$$I_{ip,t-1} + \sum_{r\in R_t} q_{iprt} + S_{ipt} = I_{ipt} + d_{ipt} \qquad \forall i \in V, p \in P, t \in \mathcal{T}\backslash\{1\} \tag{40}$$

$$\mathcal{J}^0_{ip} + \sum_{r\in R_1} q_{ipr1} + S_{ipt} = I_{ip1} + d_{ip1} \qquad \forall i \in V, p \in P \tag{41}$$

$$I_{ip,t-1} + \sum_{r\in R_t} q_{iprt} \leq C_{ip} \qquad \forall i \in V, p \in P, t \in \mathcal{T}\backslash\{1\} \tag{42}$$

$$\mathcal{J}_{ip}^{0} + \sum_{r \in R_1} q_{ipr1} \leq C_{ip} \qquad \forall i \in V, p \in P \tag{43}$$

$$\sum_{i \in V} q_{iprt} \leq B w_{prt} \qquad \forall p \in P, r \in R_t, t \in \mathcal{T} \tag{44}$$

$$\sum_{i \in V} \sum_{p \in P} q_{iprt}(1 - \beta_{irt}) = 0 \qquad \forall r \in R_t, t \in \mathcal{T} \tag{45}$$

$$S_{ipt} \geq 0 \qquad \forall i \in V, p \in P, t \in \mathcal{T} \tag{46}$$

$$I_{ipt} \geq 0 \qquad \forall i \in V, p \in P, t \in \mathcal{T} \tag{47}$$

$$q_{iprt} \geq 0 \qquad \forall i \in V, p \in P, r \in R_t, t \in \mathcal{T} \tag{48}$$

The objective function of the LP-S (39) aims to minimize the total amount of unsatisfied demands of all customers for all products on all days. Constraints (40-41) are the inventory flow balance constraints. Constraints (42-43) guarantee that storage capacities are not exceeded at customer locations. Constraints (44) guarantee that the capacity dedicated to each product is not exceeded. Constraints (45) ensure that only visited customers on a route can be delivered any products. The remaining constraints are the nonnegativity constraints.

Throughout the algorithm, we regularly use the LP-S model after every destroy operator and within some of the repair operators. Since the LP-S model is a linear programming model, we are affected as little as possible by the potential computational burden that can occur by using a mathematical model this often. However, for the second distribution assumption where each customer is allowed to be visited by one vehicle for each product on a day, the LP-S model cannot guarantee that a

product is delivered to a customer by only one vehicle on a day. In other words, the LP-S model can have a zero objective value even when the solution is infeasible for the second assumption. To prevent this, we need to add a binary variable to the LP-S model which makes it a mixed-integer programming model. Changing to a MIP model would cause a huge computational burden on the algorithm since we use this model frequently. Thus, we decide to create a fixing operation that we use after the LP-S model instead of changing the model for the second assumption. When the LP-S finds an objective value equals to zero, we check the solution to see if there is a violation in deliveries for the second assumption. If we detect any, we apply this fixing operation. The goal of this operation is to combine deliveries of a product to a customer into one vehicle. We consider two cases; (i) we try to combine deliveries in a vehicle that is already being used, (ii) we try to combine deliveries on a new route that makes direct delivery to the customer. For the first case, for all the vehicles visiting that customer, we check the empty spaces dedicated to that product. If there is enough space in a vehicle, then we combine these deliveries in that vehicle. If the first case is not possible, then we check for the second case. If we can add a new route into the solution, we combine these deliveries in this new route. However, if adding a new route would cause a customer to be visited by more than the number of products, then we remove that customer from the route it belongs to, insert it to a new route and combine these deliveries in this new route. It is worth mentioning that, in this fixing operation, we compare two cases where we adjust the capacities dedicated to products (w_{prt}) manually and not. Since the results suggest that adjusting these values manually performs better for the algorithm (see the numerical results presented in §5.3), we continue our algorithm in this way. This fixing operation does not guarantee feasibility. In that case, we assume that the solution is infeasible and continue with our algorithm.

4.2.1 Destroy operators

There are 20 different destroy operators in our ALNS variant, we adopt the first 13 of them from [7] and [35], and we developed the remaining specifically for MCIRP. We describe the destroy operators used in our implementation below:

(i) Randomly remove ρ visits: This operator starts with selecting a random day/period, then selects a random route from that day/period, and removes a randomly selected customer from that route. This operator is repeated ρ times.

(ii) Randomly insert ρ visits: This operator starts with selecting a random day/period, then selects a random route from that day/period, and inserts a randomly selected customer that is not visited on that day/period into that route. It is repeated ρ times.

(iii) Remove worst ρ visits: This operator calculates the cost decrease for each customer in case that customer is removed from their route. Then it removes the customer who yields the highest cost decrease from its route. It is repeated ρ times.

(iv) Insert best ρ visits: This operator calculates the cost increase for each customer if that customer is inserted into a route it's not visited. Then inserts the customer into the route with the lowest cost increase. This operator repeated ρ times.

(v) Shaw removal: This operator starts with selecting a customer from a randomly selected route on a randomly selected day/period. It calculates the distance of the selected customer to a customer who is closest to it and is on the same route. It then removes all customers who are within twice that distance from the selected customer from the route.

(vi) Shaw insertion: This operator selects a random day/period, then selects a customer who is not visited in that day/period. It calculates the distance of the selected customer to the customer who is closest to it among all the customers. It then inserts all customers who are within twice that distance from the selected customer

into the route. The operator inserts customer/s into the route in the most cost-effective way.

(vii) Remove ρ customers: This operator randomly selects a customer and removes it from every day/period it exists. It is repeated ρ times.

(viii) Insert ρ customers: This operator randomly selects a customer. It inserts the selected customer into a route for every day/period that was not present before. The operator inserts the customer into a route that will result in the lowest cost increase. This operator is repeated ρ times.

(ix) Empty one period: This operator randomly selects a day/period and removes all routes within that day/period.

(x) Swap routes: This operator randomly selects two days/periods from the solution and swaps their routes with each other.

(xi) Randomly move ρ visits: This operator starts with selecting a customer from a randomly selected route on a randomly selected day/period. It removes the selected customer from the selected route on the current day/period and then inserts it into a route that is performed on another day/period. The operator inserts the customer into a route that will result in the lowest cost increase. It is repeated ρ times.

(xii) Remove ρ routes: This operator randomly removes ρ routes from the solution.

(xiii) Insert ρ routes: This operator randomly inserts ρ routes to the solution.

(xiv): Divide ρ routes: This operator randomly selects ρ routes from the solution and randomly divides them into two.

(xv): Merge 2ρ routes: This operator selects a random pair of routes from the same day/period and combines the selected pair into one route. It solves the Lin-Kernighan TSP algorithm proposed in [32] to combine the pair in the most cost-effective way. It is repeated ρ times.

(xvi): Decrease ρ compartments: This operator randomly selects a route (r') from a randomly selected day/period (t'). It decreases the capacity of a randomly selected product by U. Here U is taken as a positive multiple of B. It then calculates the capacities of other products ($\in OP$) with the help of an updated version of the MIP-IFS. The update aims to ensure that the model uses only the w_{prt} variables in the selected route. To achieve this, we replace Constraints (31) with the following constraint. Here w'_{prt} is a parameter that denotes the previous number of unit compartments that product p is assigned on the selected route r' on the selected day/period t'. It is repeated ρ times.

$$\sum_{p \in OP} w_{pr't'} = \sum_{p \in OP} w'_{pr't'} + U \tag{49}$$

(xvii): Increase ρ compartments: This operator, on the contrary to the (xvi), increases the capacity of a randomly selected product by U. It calculates the capacities of other products ($\in OP$) with the updated version of MIP-IFS, however for this operator we update MIP-IFS by replacing Constraints (31) with the following constraint. It is repeated ρ times.

$$\sum_{p \in OP} w_{pr't'} = \sum_{p \in OP} w'_{pr't'} - U \tag{50}$$

(xviii): Decrease worst ρ compartments: This operator reduces the size of the capacity dedicated to a product with the highest number of unused unit compartments by the number of its empty unit compartments. It uses the same modified MIP-IFS model described in (xvi) to calculate the capacities dedicated to other products at the same route (Here, U equals the number of empty unit compartments dedicated to the regarding product). It is repeated ρ times.

(xix): Increase best ρ compartments: This operator increases the size of the capacity dedicated to a product with the highest capacity utilization rate by U. It uses the same modified MIP-IFS model described in (xvii) to calculate the capacities dedicated to other products at the same route. It is repeated ρ times.

(xx): Delete ρ compartments: This operator randomly selects a route and randomly removes one of the product's all dedicated compartments. It uses the same modified MIP-IFS model described in (xvi) to calculate the capacities dedicated to other products at the same route (Here, U equals the number of unit compartments dedicated to the regarding product). It is repeated ρ times.

We explain all of the above operators as if they were to be used in the first distribution assumption examined in our problem, which ensures that each customer can only be visited by one vehicle in a given period. While applying this assumption in the algorithm, we tailor the operations so that they insert a customer into a solution by following this assumption. In other words, for the operators (ii), (iv), (vi), (viii), (xi), and (xiii), we ignore the solutions where a customer is visited more than once in a day/period. For the second distribution assumption, where each customer is allowed to be visited by one vehicle for each product on a period, we update these operators so that they can insert a customer into a day/period at most as the number of products. For the third distribution assumption, where each customer can be visited by more than one vehicle on a period, we update these operators so that they can insert a customer into a day/period as much as it needs.

4.2.2 Repair operators

At each iteration, our ALNS algorithm destructs the solution with a destroy operator. The solution obtained by this destruction may not be feasible for the MCRIP and may need a repair operation for the algorithm to continue. Therefore, we check the feasibility of every solution after applying a destroy operator by solving the LP-S Model. If the objective function value of this model is zero, we skip the repair operators since the solution is already feasible and apply improvement operators. If it is greater than zero, then it means the solution is infeasible, and we apply repair operator(s) to the obtained solution. We have three repair operators; (i) MIP-S, (ii)

Minimum ratio insert, and (iii) Break routes.

In the case where the solution obtained after a destroy operator is infeasible, we first apply an updated version of the LP-S model, which we call MIP-S. MIP-S is formulated by adding the integer variable w_{prt} to the LP-S model, which makes it a mixed-integer programming model. For the w_{prt} variable to perform correctly, we add the following constraints to the LP-S model.

$$\sum_{p \in P} w_{prt} \leq H x_{rt} \qquad \forall r \in R_t, t \in \mathcal{T} \tag{51}$$

The MIP-S model determines the distribution quantities for the customers and the number of unit compartment sizes allocated to products. It does not have a chance to increase or decrease the cost of the solution, as it does not affect routing decisions. The main reason we use this model as the first repair operator is that it can make the solution feasible without increasing the cost of the solution.

For the second distribution assumption where each customer is allowed to be visited by one vehicle for each product on a day/period, we update the MIP-S model. First, we introduce a new binary variable f_{iprt} that takes 1 if product p is delivered to customer i by route r on day t. Secondly, we add the following two constraints to the MIP-S model.

$$q_{iprt} \leq BH f_{iprt} \qquad \forall i \in V, p \in P, r \in R_t, t \in \mathcal{T} \tag{52}$$

$$\sum_{e \in R_t \setminus \{r\}} q_{ipet} \leq BH(1 - f_{iprt}) \qquad \forall i \in V, p \in P, r \in R_t, t \in \mathcal{T} \tag{53}$$

For the third distribution assumption where each customer can be visited by more than one vehicle on a day/period, we don't need to update the MIP-S model.

The second repair operator is the Minimum ratio insert operator which is applied when the solution obtained by the first repair operator (MIP-S) is not feasible. The main goal of this operator is to insert customers with unsatisfied demands into the solution in a cost-effective way. The operator calculates a ratio for every customer that

has stock-out by dividing the increase in the cost of inserting the customer into a route by the decrease in the stock-out amount. The operator calculates the cost increase of inserting the customer with the cheapest insertion option. The operator calculates the decrease in the stock-out amount by taking the minimum of the total amount of the customer's out-of-stock products and the amount of free space in the unit compartments reserved for the products in that vehicle. The operator also considers creating a new route and making a direct delivery to the selected customer when calculating this ratio. It selects the customer-route pair with the lowest ratio and inserts the customer into that route. After each iteration of insertion, the operator solves the LP-S model to calculate the current stock-out amounts of customers. If the objective function value of this model is zero, then the solution is feasible and we terminate this operator. If it is greater than zero, then it means the solution is infeasible. Thus we continue with this operator by trying to insert a customer with the lowest ratio. We terminate this operator also when it cannot insert a customer into any day/period.

We tailor the Minimum ratio insert operator for each distribution assumption we examine. For the first assumption, we ignore inserting a customer to a day/period that is already visited. For the second assumption, we limit the number of visits to a customer on a day/period by the number of products. And for the third assumption, we don't limit anything in the operator.

Our algorithm applies the Break routes operator when the first and second repair operators cannot obtain a feasible solution. The Break routes operator aims to obtain a feasible solution by removing a customer that has stock-out product/s and making a direct delivery to it on the same day/period. The operator calculates a similar ratio to the previous repair operator for each customer with unsatisfied demand(s). Dividing the cost increase occurred by removing the customer from its current route and making direct delivery to it, by the decrease in the stock-out amount gives us the

ratio. In this operator, unlike the previous one, we use the LP-S model to calculate the decrease amount in the stock-out when we remove that customer from its route and make a direct delivery to it. If the LP-S model finds feasible solutions during the calculation of the ratio of each customer, the operator selects the customer-route pair with the lowest ratio among these feasible solutions and terminates itself, and continues with the feasible solution found. If it doesn't find any feasible solution during the ratio calculation, then the operator selects the customer-route pair with the lowest ratio among all solutions found. After removing the selected customer from its route and making a direct delivery to it, the operator continues calculating ratios for the remaining customers that have stock-out product/s. The operator terminates when there are no customers left with unsatisfied demands.

4.2.3 Improvement operators

Our algorithm seeks to decrease the total cost of the solution by applying improvement operators at each iteration. We use the improvement operators, which are (i) Remove route operator and (ii) Remove customer operator, either after a destroy operator if the solution obtained is feasible or after a repair operator that successfully repaired a solution.

We use the MIP-IFS model as the Remove route operator for the first step of the improvement phase. Our goal in developing this operator is to eliminate the routes that can be removed from the solution without ruining the feasibility of the solution. To achieve this goal, we provide the model with the route set containing only the ones that are currently being used in the solution. Thus, the model is able to eliminate routes from the solution. While updating the operator for the second and third distribution assumption, we make the same changes to the MIP-IFS model. For the second assumption, we exclude Constraints (24) from the MIP-IFS, and include Constraints (36) and (38) into it. And for the third assumption, we only exclude

Constraints (24).

The second step of the improvement phase of our algorithm is handled by the Remove customer operator. This operator aims to decrease the total cost of the solution by eliminating customers from the solution while ensuring feasibility. To figure out whether the solution will be feasible when a customer is removed from its route, we need to solve a mathematical model. For this, we compare using LP-S and MIP-S models, and we decide on using the MIP-S model since it performs better for the algorithm (see the numerical results presented in §5.3).

However, we observe that the computational burden of solving this model for every customer affects the algorithm negatively. Therefore, we minimize the number of MIP-S models used in the operator to reduce this computational burden. The operator tries to predict whether a customer can be removed from the solution without ruining the feasibility. For this, it creates a candidate list of promising customers that can be removed from their routes. To make this prediction, we first calculate the maximum inventory amount that can emerge from the days before the current day. Then, we calculate the minimum inventory required for the days after the current day. We make both of these calculations using the unused capacities in the compartments. If the maximum inventory amount that can emerge is greater than the minimum inventory amount required, the operator adds that customer-route pair to the promising list. After calculating this for every customer-route pair, the operator obtains a list of candidate customers to remove from their routes. The operator solves the MIP-S model to determine the feasibility when each customer in the list is removed from its route. To take into account the solutions where two customers are removed from the same solution, we also consider the cases where each pair of customers on the list is removed from the same solution and compare the results in §5.3. The solution that yields the highest cost decrease whilst ensuring the feasibility is selected and that customer or customer pair is removed from its/their route(s). If there is

no feasible solution, the operator is terminated without any improvements. While updating the operator for the second and third distribution assumption, we update the MIP-S model. To update this operator for the second and third distribution assumption, we make changes to the MIP-S model. For the second assumption we introduce f_{iprt} and we add to Constraints (52) and (53) to the MIP-S model. For the third assumption, we don't need to update the MIP-S model.

4.2.4 Parameter settings and pseudocode

In our ALNS, we use a roulette-wheel mechanism to select which destroy operator to apply to the solution at each iteration. Destroy operators are associated with probabilities of being selected, which are proportional to their performance (i.e., weight). Each destroy operator starts a segment with its score (π_i) being zero and its score is increased by σ each time it yields an improvement on the solution. The calculation of the weight of a destroy operator in the next segment has two components (54). The first component is the weight of the destroy operator i in the previous segment j (w_{ij}). The second component is the total score destroy operator i collects within that segment divided by the number of times the destroy operator i is used within that segment (θ_i). We combine these two components by some proportion r, which is the reaction factor ($0 \leq r \leq 1$). We take r as 0.7 in our algorithm.

$$w_{ij+1} = w_{ij}(1-r) + r\frac{\pi_i}{\theta_i} \tag{54}$$

The probability of destroy operator i being selected in segment j is denoted by p_{ij} and calculated as in (55). Initially, all the weights are taken as zero and probabilities are equal for all operators.

$$p_{ij} = \frac{w_{ij}}{\sum_{i \in I} w_{ij}} \tag{55}$$

Different than the classical applications, we employ a second roulette wheel to give a second chance for the destroy operators whose probabilities of being selected

have converged to zero, specifically have dropped below 0.1%. While the first roulette wheel is called with 90% chance, the second roulette wheel is called with 10% chance.

The remaining two parameters we use in our algorithm are ρ and U values which are both selected randomly in the interval [1, 3].

Next, we present the steps of our ALNS algorithm (Algorithm 1).

Algorithm 1: *ALNS algorithm for MCIRP*

All weights are set to 1 and all scores are set to 0.
s_{best} = Initial solution
$iterations = 0$
$time = 0$
while $iterations \leq 5000$ and $time \leq 7200$ seconds **do**
  $k = 1$
  while $k \leq K$ **do**
    Select a destroy operator ($d \in D$) using the roulette-wheel mechanism based on the current weights.
    $\theta(d) = \theta(d) + 1$
    $s' = d(s_{best})$
    s' = LP-SO(s')
    if s' is not feasible **then**
      $s'' = R(s')$
      $s''' = I(s'')$
    else
      $s''' = I(s')$
    end if
    if $z(s''') \leq z(s_{best})$ **then**
      $s_{best} = s'''$
      $\pi(d) = \pi(d) + \sigma$
    end if
    $k = k + 1$
    $iteration = iteration + 1$
  end while
  update the weights of all operators and reset their scores.
end while
return s_{best}

CHAPTER V

COMPUTATIONAL STUDY

5.1 Test Instances

While creating our test instances, we adapt the pattern used in [11] by altering some of the points to better reveal the complexity of the problem. We use a 100x100 map by taking the origin point as the center of the map. We assume that the depot resides on the origin point. We generate instances with 12 different sizes. For each instance size, we generate two test instances, which add up to a total of 24 instances. The smallest size instance has 20 customers, and up until 50 we increase the size by 5; after 50, we increase the size by 10 up to 100 customers. We calculate the distances between any nodes based on Euclidean distance. We have three different product types and consider 3-, 5-, and 7-day long planning horizons. For each customer, product, and day, demand (d_{ipt}) takes a value between [1,2] tons with %40 probability, between [2,3] tons between %40 probability, and between [3,6] tons with %20 probability. For each customer and product, the storage capacity (C_{ip}) is 40 tons. For some instances, the initial inventory levels used in [11] suffice to meet a significant portion of some customers' overall demands, which is a situation that reduces the difficulty of the problem. Hence, for each customer and product, we set the initial inventory levels (J_{ip}^0) randomly between [0,3] tons. Rather than an upper limit on the route lengths, [11] has an upper limit on the number of customers to be visited on a route, which they set as three. We, on the other hand, set the maximum route length (L) to 300 units, so that a vehicle leaving the depot can visit at most half of the map periphery. Unlike our flexible compartment structure, the MCIRP examined in [11] assumes to have compartments with fixed capacities. Therefore, they use a

parameter denoting the capacity of compartments dedicated to each product. The smallest vehicle used in their setting has an overall capacity of 18 tons, while the largest one has an overall capacity of 30 tons. Since it is likely to visit more than three customers within a distance of 300 units, we decide to use a larger vehicle than in [11] and assume that vehicles have 36 unit compartments, whereas the capacity of a unit compartment is one. We evaluate the performance of our ALNS algorithm in the test instances described above and perform numerical analysis on a system with a 2.80 GHz i7-7700HQ processor and 16GB RAM. We code our algorithm in Python and use CPLEX 12.8 with default parameter settings.

5.2 Benchmark Algorithm

Our benchmark algorithm is proposed by the authors in [1]. In this study they examine an MCVRP with flexible compartments as in our problem and similar to our second distribution assumption, they assume that each customer can only be visited once for each product on a day/period. They solve this problem with a Large Neighborhood Search (LNS) algorithm. Different than ALNS, operators in the LNS are not associated with a probability of being selected proportional to their performance. To find an initial feasible solution, they use the savings algorithm proposed in [36]. They use two operators within the algorithm; (i) shaw removal and (ii) regret-k insertion. Shaw removal selects a random customer-product pair and calculates its similarity to all other customer-product pairs. Based on these similarity measures, r number of customer-product pairs are removed from the solution. The customer-product pair with a higher similarity measure is more likely to be selected. Regret-k insertion, on the other hand, inserts the customer-product pairs that are removed by the shaw removal operator back into the solution. For this, it calculates a regret value for all the removed customer-product pairs and inserts the pair with

the highest regret value into the solution. This operation continues until all customer-product pairs are inserted into the solution. An iteration is completed by applying these two operations. If the objective value of the solution achieved in an iteration is less than the best solution found so far with D added to its value, this solution is accepted and the algorithm continues with this solution. If the objective value of the solution achieved in an iteration is more than the best solution found so far with D added to its value, the algorithm continues with the previous solution. If no better solution is found than the best solution in 500 iterations in a row, the number of customer-product pairs to be removed from the solution in the shaw removal operator is increased by one for one time only. If the solution found in 2000 consecutive iterations is worse than the best solution, the algorithm is terminated. Since the authors examine an MCVRP, the LNS algorithm is designed to find solutions for a single period. For this algorithm to find a solution for the MCIRP, we run the LNS for each day of the planning horizon separately. We subtract the initial inventory levels starting from the first day's demand to generate the daily demand parameters for each customer-product pair.

To modify the LNS algorithm so that it complies with our first and third distribution assumptions, we make the following adjustments. For the first assumption, where a customer can only be visited by one vehicle on a day/period, we ensure that instead of considering each customer-product pair separately, the algorithm considers customers solely and applies the removal and insertion operators only on the customer basis. For example, in a scenario with 20 customers and 3 products, the LNS algorithm developed for the second assumption examines 60 customer-product pairs separately and calculates similarity and regret for each. However, for the first assumption, the modified algorithm makes the calculations for the 20 customers. As expected, the LNS algorithm modified for the first assumption runs faster than its original state.

As for the third assumption, where a product can be distributed to a customer by more than one vehicle on a day/period, to include this flexibility to the LNS, we allow inserting more than one customer-product pair to the solution in the regret-k insertion operator. To enable this, we integrate a similar method we use in our second repair operator (i.e., minimum ratio insert operator) to the regret-k insertion operator. While the customer-product pairs removed in the shaw removal operator are added to the solution in the regret-k insertion operator, even if the empty space in the route to be inserted is lower than the demand of the customer for that product, the operator tries to insert as much as the free space to that route. Considering the case of each customer-product pair being inserted into each route, the ratio of the amount to be deducted from the demand of that pair (which is equal to the free space in that vehicle) by the cost increase that will occur in the routes is calculated. The customer-product pair with the highest ratio is inserted into that route and the feasibility of the resulting solution is checked by solving the LP-S model. The operator continues until all customer-product pairs are inserted into the solution at least once and the solution obtained by the LP-S model is feasible. The LNS modified for the third assumption runs slower than both its original state and the LNS modified for the first assumption due to the use of the LP-S model.

We compare benchmark algorithms modified for every distribution assumption with each other to measure their performances. We run each algorithm for two hours for a 3-day planning horizon in the first 14 instances with 20 to 50 customers. For each algorithm, we calculate the percentage deviation of the solution found from the best solution and present these percentage deviations in Table 3. We name the algorithm adapted for the first assumption as LNS-a1, the second one as LNS-a2, and the third one as LNS-a3.

LNS-a1 achieves the best result for all 14 instances while LNS-a2 deviates 2.64% and LNS-a3 deviates 2.87% from the results found by LNS-a1 on average. The main

Instance		LNS		
#	n	LNS-a1	LNS-a2	LNS-a3
1	**20**	0.00	1.90	3.31
2	**20**	0.00	5.07	6.37
3	**25**	0.00	0.29	0.97
4	**25**	0.00	3.60	3.60
5	**30**	0.00	2.50	2.50
6	**30**	0.00	5.40	5.40
7	**35**	0.00	2.80	2.80
8	**35**	0.00	1.59	1.47
9	**40**	0.00	2.01	2.01
10	**40**	0.00	2.36	2.36
11	**45**	0.00	3.51	3.51
12	**45**	0.00	3.71	3.61
13	**50**	0.00	1.89	1.89
14	**50**	0.00	0.40	0.40
Average		**0.00**	**2.64**	**2.87**

Table 3: Percentage deviation of the solutions found by each benchmark algorithm modified for every distribution assumption from best solution found for that instance

reason why the LNS-a1 performs better than the others is that the LNS-a1 completes much more iterations during the 2 hours of run time. To summarize this difference, we present the number of iterations completed for each modification in Table 4.

As can be seen in Table 4, LNS-a1 can complete 10.67 times more iterations than LNS-a2 and 30.79 times more than LNS-a3. These results support the superior performance of LNS-a1 in Table 3. Due to its performance, we decide to use the LNs-a1 modification as our benchmark algorithm.

5.3 Numerical Results

As mentioned in §4.2.3, the Remove customers operator seeks to reduce the total cost by removing customers from their route. After removing a customer from its route, it is necessary to use a mathematical model to check whether the solution is feasible. The LP-S model, which we often use to check the feasibility of the solutions, or the MIP-S model, which is a version of the LP-S model with the w_{prt} integer decision variable

Instance		LNS		
#	n	LNS-a1	LNS-a2	LNS-a3
1	**20**	6126	572	303
2	**20**	7149	838	263
3	**25**	5407	447	177
4	**25**	4119	432	179
5	**30**	4367	408	140
6	**30**	4228	426	122
7	**35**	3725	294	90
8	**35**	3482	339	114
9	**40**	3265	281	78
10	**40**	2916	262	80
11	**45**	2483	228	60
12	**45**	2913	210	63
13	**50**	1961	186	57
14	**50**	2609	205	52
Average		**3910.7**	**366.3**	**127.0**

Table 4: Comparison of the number of iterations completed by each benchmark algorithm modified for each distribution assumption.

added, are suitable for this purpose. We generate two different ALNS algorithms; the first one (ALNS-1) uses LP-S to check feasibility, and the second one (ALNS-2) uses MIP-S to check feasibility. We run each algorithm for two hours with our first 14 instances and compare their results. For both algorithms, we calculate the percentage deviation of the solution found from the best solution and present these percentage deviations in Table 5.

Based on the results in Table 5, we observe that determining the number of unit compartments allocated to products along with the distribution quantities allows us to find better solutions. In other words, ALNS-2 deviates 0.3% from the best solution found while ALNS-1 deviates 2.1% on average.

In the Remove customer improvement operator, we also consider removing more than one customer from the same solution. To take into account the solutions where

Instance		ALNS	
#	n	ALNS-1	ALNS-2
1	**20**	0.1	0.0
2	**20**	0.0	0.0
3	**25**	3.1	0.0
4	**25**	3.1	0.0
5	**30**	0.0	1.0
6	**30**	0.0	0.5
7	**35**	3.1	0.0
8	**35**	0.0	3.1
9	**40**	5.7	0.0
10	**40**	1.8	0.0
11	**45**	2.7	0.0
12	**45**	4.8	0.0
13	**50**	4.5	0.0
14	**50**	0.6	0.0
Average		**2.1**	**0.3**

Table 5: Percentage deviation of the solutions found by ALNS-1 and ALNS-2 from best solution found for that instance

two customers are removed from the same solution, we solve the MIP-S model including the cases where each pair of customers on the promising list can also be removed from the same solution. We name this adaptation as ALNS-3 and we run the algorithm for two hours with our first 14 instances and compare its results with ALNS-2. For both algorithms, we calculate the percentage deviation of the solution found from the best solution and present these percentage deviations in Table 6.

From Table 6, we observe that including the removal of customer pairs from the solution improves the results. ALNS-3 deviates 0.3% from the best solution found, while ALNS-2 deviates 0.8% on average, and as a result, it outperforms ALNS-1 and ALNS-2 for the first 14 instances.

As ALNS-3 gives the best solution overall, we decide to use and update ALNS-3 for the other two delivery assumptions. We name our algorithm that we update for the first assumption as ALNS-3-a1, for the second assumption as ALNS-3-a2, and for the third assumption as ALNS-3-a3.

Instance		ALNS	
#	n	ALNS-2	ALNS-3
1	**20**	0.3	0.0
2	**20**	0.0	0.0
3	**25**	1.5	0.0
4	**25**	0.8	0.0
5	**30**	1.7	0.0
6	**30**	0.0	0.0
7	**35**	3.3	0.0
8	**35**	0.0	0.0
9	**40**	0.0	1.3
10	**40**	0.0	0.1
11	**45**	3.3	0.0
12	**45**	0.0	2.6
13	**50**	0.0	0.2
14	**50**	0.7	0.0
Average		**0.8**	**0.3**

Table 6: Percentage deviation of the solutions found by ALNS-2 and ALNS-3 from best solution found for that instance

As mentioned in §4.2, for the fixing operation that is used after the LP-S model, we compare two cases where we adjust the capacities dedicated to products (w_{prt}) manually and not. We name the algorithm which assumes that capacities dedicated to products are fixed for the fixing operation as ALNS-3-a2, and the algorithm that can manually adjust capacities dedicated to products during the fixing operation as ALNS-3-a2-v2. For both algorithms, we calculate the percentage deviation of the solution found from the best solution and present these percentage deviations in Table 7.

Among these two algorithms we develop for the second assumption, ALNS-3-a2-v2 deviates 0.6% from the best solution found while ALNS-3-a2 deviates 1.4% on average (Table 7). In other words, adjusting the capacities allocated to products (w_{prt}) manually allows us to find better results when trying to make an infeasible solution feasible by combining deliveries in a single vehicle. Based on these results, we decide to use the ALNS-3-a2-v2 for the second assumption.

Instance		ALNS	
#	n	ALNS-3-a2	ALNS-3-a2-v2
1	**20**	0.0	0.0
2	**20**	1.0	0.0
3	**25**	0.0	2.7
4	**25**	0.0	0.8
5	**30**	0.0	1.2
6	**30**	5.1	0.0
7	**35**	1.4	0.0
8	**35**	0.9	0.0
9	**40**	1.6	0.0
10	**40**	0.0	3.0
11	**45**	1.7	0.0
12	**45**	2.8	0.0
13	**50**	3.1	0.0
14	**50**	2.7	0.0
Average		**1.4**	**0.6**

Table 7: Percentage deviation of the solutions found by ALNS-3-a2 and ALNS-3-a2-v2 from best solution found for that instance

To evaluate the performance of our algorithms for every delivery assumption (i.e., ALNS-3-a1, ALNS-3-a2-v2, ALNS-3-a3), we compare the results of these algorithms with the modified benchmark algorithm (i.e., LNS-a1) using the instances we generate for the MCIRP. We run each algorithm in each instance for two hours considering 3-, 5-, and 7-day long planning horizons and present the results obtained in Table 8. The first two columns represent the instance ID and the number of customers in the instance. For each algorithm in each instance, we calculate the percentage deviation of the solution found by our approach from the solution found by the benchmark algorithm and present these percentage deviations in Table 8.

Our algorithm is able to find better solutions than the benchmark algorithm for every instance (Table 8). To be specific, our algorithm finds 24.63% better solutions than the benchmark algorithm for the first assumption, 23.05% for the second assumption, and 26.03% for the third assumption on average.

If we examine how much our ALNS algorithm improves the initial solution, we

		ALNS								
Instance		**First Assumption**			**Second Assumption**			**Third Assumption**		
#	n	$T = \mathbf{3}$	$T = \mathbf{5}$	$T = \mathbf{7}$	$T = \mathbf{3}$	$T = \mathbf{5}$	$T = \mathbf{7}$	$T = \mathbf{3}$	$T = \mathbf{5}$	$T = \mathbf{7}$
1	**20**	-31.5	-40.7	-41.8	-31.9	-39.3	-41.5	-31.9	-42.3	-42.7
2	**20**	-33.6	-40.9	-41.0	-34.8	-38.5	-40.2	-34.8	-43.1	-41.5
3	**25**	-31.3	-33.7	-37.5	-28.1	-32.1	-35.1	-32.8	-35.0	-38.2
4	**25**	-33.2	-33.5	-38.7	-33.4	-32.6	-35.6	-34.6	-37.1	-39.0
5	**30**	-23.2	-32.7	-35.7	-26.1	-32.6	-32.4	-28.0	-34.3	-35.0
6	**30**	-30.9	-34.7	-36.5	-32.6	-34.0	-36.0	-32.7	-34.1	-36.5
7	**35**	-24.6	-29.0	-28.3	-22.5	-22.5	-27.3	-24.9	-28.0	-29.6
8	**35**	-27.1	-31.6	-32.0	-28.6	-27.5	-31.7	-30.9	-29.6	-33.2
9	**40**	-20.6	-26.8	-27.2	-20.1	-25.0	-25.3	-24.7	-22.9	-27.7
10	**40**	-25.2	-30.0	-30.8	-23.9	-28.2	-26.4	-25.0	-31.5	-31.9
11	**45**	-22.4	-21.5	-26.9	-19.3	-20.1	-22.0	-19.4	-22.4	-27.6
12	**45**	-21.1	-24.6	-28.1	-20.5	-28.4	-26.5	-24.9	-31.7	-29.8
13	**50**	-20.3	-22.7	-25.9	-16.8	-23.4	-23.7	-19.4	-26.1	-26.7
14	**50**	-16.7	-23.2	-26.9	-15.4	-21.2	-25.7	-22.3	-25.2	-24.4
15	**60**	-22.2	-26.3	-27.2	-16.7	-22.3	-26.1	-23.8	-24.8	-26.9
16	**60**	-13.7	-19.8	-24.0	-18.0	-19.4	-22.8	-19.6	-25.9	-23.7
17	**70**	-16.4	-20.9	-23.5	-14.6	-21.4	-24.6	-14.2	-24.9	-21.9
18	**70**	-17.0	-23.2	-23.2	-16.1	-20.8	-21.9	-20.5	-25.8	-25.1
19	**80**	-15.7	-18.6	-21.1	-11.0	-15.4	-19.7	-14.4	-20.8	-21.4
20	**80**	-8.3	-14.1	-16.3	-10.2	-16.3	-18.0	-15.4	-17.8	-11.9
21	**90**	-10.4	-16.9	-15.9	-13.6	-10.7	-19.9	-16.8	-19.5	-19.8
22	**90**	-12.5	-17.9	-19.5	-12.1	-10.5	-12.7	-13.0	-19.2	-18.7
23	**100**	-11.6	-16.3	-15.9	-4.2	-13.9	-9.6	-14.8	-12.2	-19.6
24	**100**	-8.9	-12.9	-19.0	-4.6	-12.0	-12.6	-12.8	-16.1	-19.7
Average		**-20.8**	**-25.5**	**-27.6**	**-19.8**	**-23.7**	**-25.7**	**-23.0**	**-27.1**	**-28.0**

Table 8: Percentage deviation of ALNS from the benchmark algorithm for each distribution assumption

observe that ALNS-3-a1 improves the solutions found by the MIP-IFS model by 17.92%, ALNS-3-a2-v2 improves by 18.80%, and ALNS-3-a3 improves by 7.27% on average for each time horizon.

CHAPTER VI

CONCLUSION

In this study, we close an existing gap in the literature by addressing the MCIRP by employing a fleet of vehicles with flexible compartments. Each vehicle consists of a certain number of unit capacities. Here, the supplier is entitled to determine the number of unit capacities allocated for each product on each route. In a setting where there are multiple products to be distributed to customers, whose demands vary on product and period basis, this flexibility benefits the supplier in better coordinating the distribution needs.

The simultaneous decision-making of the distribution amounts, inventory levels, vehicle routes, and capacity allocations yields a very challenging problem, however solving this problem in an effective and efficient way brings a lot of advantages for both the supplier and the customers. To this end, we develop a novel matheuristic where we use mathematical models in an ALNS framework. We consider three different split delivery assumptions and tailor our approach separately for each. We introduce problem-specific operators and strategies to enhance our algorithm.

We generate a set of test instances with up to 100 customers and a 7-day planning horizon. To evaluate the performance of our matheuristic, we select a benchmark study from the literature [1], which adopts a flexible compartment structure as we do, yet examine the problem for a single period and solve the MCVRP. To make a fair comparison, for all the days of our planning horizon, we run their algorithm and observe that our matheuristic finds 24.57% better results than the benchmark algorithm. Here, we do not only show the good performance of our approach, but also the advantages of handling the problem as a multi-period problem.

Bibliography

[1] A. Hübner and M. Ostermeier, "A multi-compartment vehicle routing problem with loading and unloading costs," *Transportation Science*, vol. 53, no. 1, pp. 282–300, 2019.

[2] M. Ostermeier and A. Hübner, "Vehicle selection for a multi-compartment vehicle routing problem," *European Journal of Operational Research*, vol. 269, no. 2, pp. 682–694, 2018.

[3] U. Derigs, J. Gottlieb, J. Kalkoff, M. Piesche, F. Rothlauf, and U. Vogel, "Vehicle routing with compartments: Applications, modelling and heuristics," *Operations Research Spectrum*, vol. 33, no. 4, pp. 885–914, 2011.

[4] T. Henke, G. Speranza, and G. Wäscher, "The multi-compartment vehicle routing problem with flexible compartment sizes," *European Journal of Operational Research*, vol. 246, no. 3, pp. 730–743, 2015.

[5] J. Lenstra and A. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[6] C. Archetti, N. Boland, and M. Speranza, "A matheuristic for the multivehicle inventory routing problem," *INFORMS Journal on Computing*, vol. 29, no. 3, pp. 377–387, 2017.

[7] L. Coelho, J. Cordeau, and G. Laporte, "The inventory-routing problem with transshipment," *Computers & Operations Research*, vol. 39, no. 11, pp. 2537–2548, 2012b.

[8] K. Doerner and V. Schmid, *Matheuristics for rich vehicle routing problems*, 2010. Hybrid metaheuristics. Lecture notes in computer science. *Springer* **6373** 206–221.

[9] D. Ronen, "Marine inventory routing: Shipments planning," *Journal of the Operational Research Society*, vol. 53, no. 1, pp. 108–114, 2002.

[10] M. Christiansen and K. Fagerholt, *Maritime inventory routing problems.*, 2009. *Encyclopedia of Optimization*, C.A. Floudas and P.M. Pardalos, Eds., second ed. Springer-Verlag 1947–1955.

[11] D. Popovic, M. Vidovic, and G. Radivojevic, "Variable neighborhood search heuristic for the inventory routing problem in fuel delivery," *Expert Systems with Applications*, vol. 39, no. 18, pp. 13390–13398, 2012.

[12] M. Ostermeier, T. Henke, A. Hübner, and G. Wäscher, "Multi-compartment vehicle routing problems: State-of-the-art, modeling framework and future directions," *European Journal of Operational Research*, 2021.

[13] A. El-Fallahi, C. Prins, and R. Calvo, "A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem," *Computers and Operations Research*, vol. 35, no. 5, pp. 1725–1741, 2008.

[14] J. Chen, B. Dan, and J. Shi, "A variable neighborhood search approach for the multi-compartment vehicle routing problem with time windows considering carbon emission," *Journal of Cleaner Production*, vol. 277, 2020.

[15] M. Abdulkader, Y. Gaipal, and T. ElMekkawy, "Hybridized ant colony algorithm for the multi compartment vehicle routing problem," *Applied Soft Computing*, vol. 37, pp. 196–203, 2015.

[16] M. Reed, A. Yiannakou, and R. Evering, "An ant colony algorithm for the multi-compartment vehicle routing problem," *Applied Soft Computing*, vol. 15, pp. 169–176, 2014.

[17] P. Silvestrin and M. Ritt, "An iterated tabu search for the multi-compartment vehicle routing problem," *Computers and Operations Research*, vol. 81, pp. 192–202, 2017.

[18] M. Alinaghian and N. Shokouhi, "Multi-depot multi-compartment vehicle routing problem solved by a hybrid adaptive large neighborhood search," *Omega*, vol. 76, pp. 85–99, 2018.

[19] M. Muyldermans and G. Pang, "On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm," *European Journal of Operational Research*, vol. 206, no. 1, pp. 93–103, 2010.

[20] R. Lahyani, L. Coelho, M. Khemakhem, G. Laporte, and F. Semet, "A multicompartment vehicle routing problem arising in the collection of olive oil in tunisia," *Omega*, vol. 51, no. 1, pp. 1–10, 2015.

[21] M. Vidovic, D. Popovic, and B. Ratkovic, "Mixed integer and heuristics model for the inventory routing problem in fuel delivery," *International Journal of Production Economics*, vol. 147, pp. 593–604, 2014.

[22] L. Coelho and G. Laporte, "Classification, models and exact algorithms for multi-compartment delivery problems," *European Journal of Operational Research*, vol. 242, no. 3, pp. 854–864, 2015.

[23] P. Avella, M. Boccia, and A. Sforza, "Solving a fuel delivery problem by heuristic and exact approaches," *European Journal of Operational Research*, vol. 152, no. 1, pp. 170–179, 2004.

[24] F. Cornillier, F. Boctor, G. Laporte, and J. Renaud, "An exact algorithm for the petrol station replenishment problem," *Journal of the Operational Research Society*, vol. 59, no. 5, pp. 607–615, 2008a.

[25] A. Benantar, R. Ouafi, and J. Boukachour, "An improved tabu search algorithm for the petrol-station replenishment problem with adjustable demands," *INFOR: Information Systems and Operational Research*, vol. 58, no. 1, pp. 17–37, 2020.

[26] F. Cornillier, F. Boctor, G. Laporte, and J. Renaud, "A heuristic for the multi-period petrol station replenishment problem," *European Journal of Operational Research*, vol. 191, no. 2, pp. 2295–2305, 2008b.

[27] F. Cornillier, G. Laporte, F. Boctor, and J. Renaud, "The petrol station replenishment problem with time windows," *Computers & Operations Research*, vol. 36, no. 3, pp. 919–935, 2009.

[28] F. Cornillier, F. Boctor, and J. Renaud, "Heuristics for the multi-depot petrol station replenishment problem with time windows," *European Journal of Operational Research*, vol. 220, no. 2, pp. 361–369, 2012.

[29] Y. Hsu, J. Walteros, and R. Batta, "Solving the petroleum replenishment and routing problem with variable demands and time windows," *Annals of Operations Research*, vol. 1, pp. 1–38, 2018.

[30] M. Coccola, C. Mendez, and R. Dondo, "A two-stage procedure for efficiently solving the integrated problem of production, inventory, and distribution of industrial products," *Computers & Chemical Engineering*, vol. 134, 2020.

[31] J. Oppen, A. Løkketangen, and J. Desrosiers, "Solving a rich vehicle routing and inventory problem using column generation," *Computers & Operations Research*, vol. 37, no. 7, pp. 1308–1317, 2010.

[32] K. Helsgaun, "General k-opt submoves for the lin-kernighan tsp heuristic," *Mathematical Programming Computation*, vol. 1, no. 2–3, pp. 119–163, 2009.

[33] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation Science*, vol. 40, no. 4, pp. 455–472, 2006.

[34] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," *In: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming. Springer, New York*, pp. 417–431, 1998.

[35] D. Aksen, O. Kaya, F. Salman, and O. Tuncel, "An adaptive large neighborhood search algorithm for a selective and periodic inventory routing problem," . *European Journal of Operational Research*, vol. 239, no. 2, pp. 413–426, 2014.

[36] G. Clarke and J. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.

[37] L. Coelho, J. Cordeau, and G. Laporte, "Consistency in multivehicle inventory-routing," *Transportation Research Part C: Emerging Technologies*, vol. 24, no. 1, pp. 270–287, 2012a.

VITA

Ömer Berk Ölmez graduated from Maltepe Anatolian High School in 2012. He earned his B.Sc. degree in Industrial Engineering from Özyeğin University in June 2018. He has been pursuing her M.Sc. degree in Industrial Engineering at Özyeğin University since September 2018, under the supervision of Dr. Ali Ekici and Dr. Okan Örsan Özener. His research interests include supply chain management and logistics.