DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

# MULTI PRODUCT INVENTORY LOCATION-ROUTING PROBLEM WITH SHORTAGES

by
Özge ŞATIR AKPUNAR

June, 2019
İZMİR

# MULTI PRODUCT INVENTORY LOCATION ROUTING PROBLEM WITH SHORTAGES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science in Industrial Engineering, Industrial Engineering Program

by
Özge ŞATIR AKPUNAR

June, 2019
İZMİR

**M.Sc THESIS EXAMINATION RESULT FORM**

We have read the thesis entitled **"MULTI PRODUCT INVENTORY LOCATION ROUTING PROBLEM WITH SHORTAGES"** completed by **ÖZGE ŞATIR AKPUNAR** under supervision of **ASSOC. PROF. DR. ŞENER AKPINAR** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Ş. Akpınar

Assoc. Prof. Dr. Şener AKPINAR

Supervisor

Assist. Prof. Dr. Leyla DEMIR

Jury Member

Asist. Prof. Dr. Özgür YALÇINKAYA

Jury Member

Prof. Dr. Kadriye ERTEKİN

Director

Graduate School of Natural and Applied Sciences

## ACKNOWLEDGMENTS

Firstly, I would like to thank my supervisor, Assoc. Prof. Dr. Şener AKPINAR for his support; advice and encouragement that helped me overcome challenges throughout my M.Sc. thesis.

I would like to offer my eternal thanks to my valued partner Anıl AKPUNAR for his endless support, understanding and patient.

Last, but the most, I would like to emphasize my deep thankfulness to my family, Perihan ŞATIR, İbrahim ŞATIR and my sister, Müge HATİP for their love, confidence, encouragement and endless support in my whole life.

Özge ŞATIR AKPUNAR

**MULTI PRODUCT INVENTORY LOCATION ROUTING PROBLEM WITH SHORTAGES**


## ABSTRACT

The design of an effective logistic network generally consists of two combinatorial optimization problems; to determine facility locations among a group of potential candidates, and determine vehicle routes to supply demand points from these locations. The combination of these two problems is known as the Location-Inventory-Routing Problem (LIRP) and this study addresses the multi-product and multi-depot LIRP while considering  defective manufacturing processes. LIRP is an enriched location-routing problem (LRP) with the inventory related issues and determines inventory, location and transportation related decisions at the strategic, tactical and operational levels. This study considers a manufacturing process with the probability of encountering defective items in every supply point. Because of these defective items supply of products may disrupted in supply points and, therefore, backorders may occur. A mathematical model, a combination of a location-routing problem and a joint replenishment economic order quantity (EOQ) model with backorders, is developed to formally describe the problem. Considering the NP-hardness of the proposed model, a hybrid meta-heuristic algorithm,  a combination of the adaptive large neighborhood search (ALNS) algorithm and the variable neighborhood search (VNS) algorithm, is proposed to tackle the problem. Since there is a lack of standard benchmarks related to the considered LIRP problem in the literature the performance of the proposed algorithm is evaluated on a standard benchmark set of LRP instances taken from the related literature, and the proposed algorithm is realized on a real-life problem. The computational results indicate the efficiency of the proposed algorithm in solving the LRP instances  and the proposed algorithm could be successfully used to handle the LIRPs.



**Keywords:** Multi-product multi-depot location-inventory-routing problem, defective items, shortages, hybrid adaptive large neighborhood search algorithm

# KARŞILANAMAYAN TALEPLİ ÇOKLU ÜRÜN ENVANTER YERLEŞİM ROTALAMA PROBLEMİ

## ÖZ

Etkili bir lojistik ağın tasarımı genellikle iki kombinatoryal optimizasyon probleminden oluşur: bir grup potansiyel aday arasında tesis konumlarının belirlenmesi ve bu noktalardan talep noktalarının ihtiyaçlarını karşılamak için araç rotalarının belirlenmesi. Bu iki problemin birleşimi Yerleşim-Envanter-Rotalama problemi (YERP) olarak bilinir ve bu çalışma hatalı üretim süreçlerini göz önüne alarak çoklu ürün ve çoklu depo YERP'sini ele almaktadır. YERP, envanter ile ilgili konularla zenginleştirilmiş bir yerleşim-rotalama problemidir (YRP) ve stratejik, taktiksel ve operasyonel seviyelerde envanter, yerleşim ve ulaşımla ilgili kararları belirler. Bu çalışma, her tedarik noktasında kusurlu ürünlerle karşılaşabilme olasılığı olan bir üretim sürecini ele almaktadır. Bu hatalı ürünler nedeniyle, ürünlerin tedariği, tedarik noktalarında kesintiye uğrayabilir ve bu nedenle de karşılanamayan talep durumu oluşabilir. Problemin tanımı için YRP ile karşılanamayan talepli ekonomik üretim miktarı (EÜM) modeli entegre edilerek bir matematiksel model geliştirilmiştir. Önerilen modelin NP-zorluğu göz önüne alarak, modeli çözmek için adaptif büyük komşuluk arama (ABKA) ve değişken komşuluk arama (DKA) algoritmalarını içeren melez bir meta-sezgisel algoritma geliştirilmiştir. Önerilen algoritmanın performansı, YERP ile ilgili standart kıyaslama örnekleri eksikliği nedeniyle ilgili literatürden alınan klasik YRP örneklerinin standart kıyaslama seti üzerinde değerlendirilmiş ve önerilen algoritma bir gerçek hayat problemine uygulanmıştır. Hesaplamalı sonuçlar, önerilen algoritmanın YRP örneklerini çözmedeki etkinliğini ve YERP problemini çözmek için kullanılabilirliğini göstermektedir.

**Anahtar kelimeler:** Çok ürünlü çok depolu yerleşim-envanter-rotalama problemi, hatalı ürünler, karşılanamayan talep, hibrit adaptif büyük komşuluk arama algoritması

## CONTENTS

**LIST OF FIGURES**

## LIST OF TABLES

**Page**

# CHAPTER ONE
# INTRODUCTION

## 1.1 Objectives and Motivation

Companies have gained the ability of producing their goods in high volumes after the Industrial Revolution. This situation exposed many problems, which must be tackled effectively. The basic problem is to store and distribute the produced items with a cost-effective manner. After years, this problem got more complicated because of the changed market conditions under the effect of globalization. Besides the Industrial Revolution and Globalization, the Just in Time (JIT) philosophy had also effects on the market conditions and production processes. The company, who wants to remain competitive and adapt to the varied market requirements, must design their warehousing, manufacturing, and dispatching processes as efficiently as possible. As described by Chopra & Meindl (2007) “a supply chain consists of all parties involved, directly or indirectly, in fulfilling a customer request. The supply chain includes not only the manufacturers and suppliers but also transporters, warehouses, retailers and even customers themselves.” In other words, the logistic costs constitute a significant part of the annual expenditures of the companies and hence, these activities must be managed in an effective manner. The design of such an effective logistic network involves two hard combinatorial optimization problems, to locate facilities, and set the vehicle routes supplying demand points from these locations (Prodhon & Prins, 2014). The above-mentioned combinatorial optimization problems had been used to tackle separately until the idea of combining these problems was highlighted by Maranzana (1964), Von Böventer (1961) and Webb (1968). This combined problem has been termed as the location-routing problem (LRP) by Laporte (1988). Salhi & Rand (1989) introduced the potential benefits brought by the simultaneously solving the location and routing problems by indicating that solving these problems separately often leads to suboptimal solutions. LRPs typically integrate three different decisions: the number and location of the facilities, the allocation of the demand points to the facilities, and determine the vehicle routes (Lopes, Ferreira, Santos, & Barreto, 2013). Some of the LRP practices involve newspaper and postal delivery (Jacobsen & Madsen, 1980), blood distribution (Or & Pierskalla, 1979), waste collection (Kulcar, 1996), aircraft routing (Argüello, Bard & Yu, 1997), food distribution (Watson-Gandy & Dohrn, 1973), school bus pick-up and delivery (Park & Kim, 2010), mobile health care (Xu, Wermus & Bauman, 2011) etc. For more detailed information about the LRPs, the reader can refer to the review papers of Nagy & Salhi (2007), Melo, Nickel, & Saldanha-da-Gama (2009), Lopes et al. (2013), Prodhon & Prins (2014) and Farahani, Rahidi Bajgan, Fahimnia, & Kaviani (2015) which were handled the LRPs within different perspectives.

As mentioned above, JIT philosophy has also caused many changes within the production processes and thereby within the supply chain management (SCM) by focusing on inventory, kanban, wastes, and lead-time reduction. From the SCM point of view, it must not be wrong to claim that the inventory management is the most important novelty raised with the JIT philosophy. The optimization problems dealing with the inventory management within the SCM perspective, named as location inventory problem (LIP), became more important under the influence of JIT philosophy. In addition, the existing literature on LIP may be classified into four groups; the basic LIP, location-inventory routing problem (LIRP), dynamic-location-inventory problem (DLIP) and inventory-transportation problem (ITP) (Farahani et al., 2015). Most of the studies consider dual combination of inventory, location and routing rather than taking into consideration all simultaneously. Although, these decisions are solved separately as it is hard to solve them at the same time, they must be regarded in a unified model to come close to the real world. Moreover, considering them simultaneously can have significant effect on reducing related costs and improving efficiency.

Since of the importance of solving these three types of problems simultaneously, within the context of this thesis, we deal with the LIRP, which is an enriched LRP that considers the inventory decisions in addition to location and transportation decisions at the supply chain management levels. Decisions that must be considered simultaneously are: 1) the number and location of the facilities 2) inventory decisions of the located facilities 3) allocation of the customers to facilities and 4) vehicle routing decisions.

Most of the studies on LIRP have focused extensively on single-product. However, single-product location inventory routing models are not able to respond the variety of products, which is an outcome of the current consumer-centric market conditions. In addition, defective items could be observed during production process in the real life. It leads to occur shortages and additional costs such as backordering or shortage costs. At this point, the lack of the studies dealing with the consideration of variety of products and defective production process for LIRP stands out in the existing literature.

The main goal of this study is to introduce the multi-product LIRP with shortages, by formally describing the problem, developing solution procedures to tackle the problem, and to show the feasibility of the proposed meta-heuristic algorithm on the LIRPs.

**1.2 Research Methodology**

We developed a mixed integer non-linear programming (MINLP) model, which considers the multi-product with defective rates for LIRP for the first time, in order to formally describe the problem. However, due to the NP-Hardness of the problem the proposed MINLP model is not able to solve large scale problems. Therefore, we developed a hybrid meta-heuristic which is comprised of adaptive large neighborhood search algorithm (ALNS) and variable neighborhood search (VNS) algorithm to solve the problem. The VNS algorithm is realized as an elitist search mechanism therefore as a local search within the ALNS algorithm to provide as satisfactory level of intensification. The effectiveness of these two meta-heuristics has been proven by the research community in solving the variants of the vehicle routing problem individually, and therefore, this proven performance of these algorithms encourages the rationale of combining these algorithms to solve LRP and LIRP.

### 1.3 Outline of the Thesis

The rest of this thesis is organized as follows: the following chapter contains the problem definition, and a literature survey on the LIRPs. This chapter provides the detailed information on the proposed hybrid algorithm, and also a literature survey on the ALNS algorithm.

The developed Mixed Integer Nonlinear Programming (MINLP) model for location-inventory-routing problem with multi-product, defective items, and shortages is given in Chapter 3. In this chapter, detailed information about the adapted inventory model is given with its notations. In addition, chapter three includes the model validation on a sample LIRP problem which is coded in GAMS and solved by Mixed Integer Nonlinearly Constrained Optimization Solver.

Chapter 4 presents the procedure of the proposed hybrid meta-heuristic algorithm and the implementation of the proposed model on a set of classical LRP instances. The proposed solution method is evaluated by using LRP data set taken from literature.

In chapter 5, developed algorithm is implemented on a real-life case. The detailed information on the chosen test problem is also presented in this chapter.

Finally, in addition to summary of the thesis, the contribution of this thesis and future research directions are discussed in the last chapter.

# CHAPTER TWO
# PROBLEM DEFINITION AND LITERATURE SURVEY

This chapter describes the standard location inventory problem firstly. Then, a brief classification of the LRPs and the related literature on the LIRP which is the main focus of this thesis are given. After that, the Large Neighborhood Search (LNS) algorithm that constitutes the basis of the adaptive LNS which is used as the solution procedure within the context of this thesis is explained. Finally, the related literature on the LNS and ALNS algorithms is reviewed.

## 2.1 Location Inventory Routing Problem

The standard LRP consist of three decisions: whether to open a facility or not (facility location problem), customer assignment to the opened facilities (customer allocation problem) and to determine the routes of vehicles that serves the customers assigned to the opened facilities (vehicle routing problem (VRP)). The facility location problem aims at opening facilities among a group of potential candidates to minimize the total facility opening cost as the first decision, while the customer allocation problem aims at determining which customer will be served from which opened facility as the second decision. As the third decision, vehicle routing problem intend to determine the vehicle routes for serving the customers assigned to the facilities to minimize the total travelling costs due to the travelling distances.

These problems consider only the decisions regarding to the selection of the facilities and assignment of the customers to the capacitated vehicles to create the routes. However, inventory decisions are disregarded in LRPs. Inventory decisions, such as order quantity, order frequency, backorder or shortage quantity, etc., influence both the facility location allocation decisions and the routing decisions. For instance, dispatching in higher quantities and smaller frequency causes raising inventory cost while reducing transportation cost. Hence, the total system cost may rise when inventory control decisions are ignored because of the location, inventory and transportation costs are interrelated to each other.

There are comprehensive application areas for LRPs in literature. The practical applications focus not only on distribution of consumer goods, but also medical and military applications. Some of them can be described as newspaper distribution, goods distribution, food and drink distribution, postbox location, military equipment location, waste collection, network designs, and parcel delivery (Nagy & Salhi, 2007).

Within decades, varieties of LRP have been developed taking another logistics decisions into account by researchers. One of the developed models is location inventory problem (LIP). The problem is about to find the best location for supply points. Thus, the suppliers can serve by minimizing inventory holding and transportation costs, also considering the opening costs of the supply points. LIRP is one of the types of LIPs which considers the integration of location, inventory, and routing decisions into one problem. Table 2.1 gives general attributes of LIRP models.

Table 2.1 General attributes of LIRP models (Farahani et al., 2015)

| Attribute | Options |
|---|---|
| Hierarchical Level | Single-echelon |
| | Two-echelon |
| | Three-echelon |
| Demand Type | Deterministic |
| | Stochastic |
| Planning Horizon | Finite |
| | Infinite |
| Number of Supplier | Single source |
| | Multiple sources |
| Supplier Capacity | Finite |
| | Infinite |
| Fleet Size | Single vehicle |
| | Multiple vehicles |
| Fleet Capacity | Finite |
| | Infinite |
| Fleet Type | Homogeneous |
| | Heterogeneous |
| Objective Function | Single objective |
| | Multiple objectives |
| Product | Single product |
| | Multi-product |
| Inventory Decisions | Lost Sales |
| | Back-order |
| | Nonnegative |

In Table 2.1, level refers to the stages of distribution considered from supply point to the last demand point, e.g. Single-echelon represents direct distribution between supply and demand points. Number of suppliers may be varied, while there is one single source serving demand points, the structure could be multiple sources serving several demand points. The planning horizon, supplier capacity and fleet capacity could be regarded either finite or infinite. Demand at end-users may be considered deterministic or stochastic, while deterministic refers to known demand; stochastic refers to random demands generated with given probability distributions. Number of fleets can be either restricted with single vehicle or be allowed several vehicles. The fleet can be either homogeneous (i.e. each vehicle has same capacity) or heterogeneous (i.e. each vehicle has different capacity). In general, main objective function of the LIRP is to minimize total distance, facility opening and inventory costs. In some special cases, it could be dealt with the dual objective function simultaneously, e.g. maximizing number of customers to be served and minimizing total cost. LIRP models could also be formulated for single or multiple items. Inventory decisions determine how inventory management is modeled (Coelho, Cordeau, & Laporte, 2013). When the inventory level is negative, backordering occurs. Then, backorder will be met at the next stage. If the backorders are not met later, the demand is regarded as lost sales. In both cases, the penalty cost occurs. In some cases, it is not allowed to become negative inventory level.

### *2.1.1 Illustrative Example*

An illustrative example is given for an LRP instance which involves 20 customers and 5 suppliers in this section. The location and demand information of customers is shown in Table 2.2. The location information and opening costs of the potential suppliers are listed in Table 2.3. The capacity for each supplier is set to 300, the vehicle capacity is set to 150, and the fixed cost for a route is set to 1000. A visual illustration of the example solution is given in Figure 2.1.

Table 2.2 Location and demand information of customers

| Customer | X | Y | Demand |
|---|---|---|---|
| 1 | 22 | 35 | 14 |
| 2 | 37 | 29 | 17 |
| 3 | 8 | 32 | 16 |
| 4 | 40 | 14 | 17 |
| 5 | 47 | 10 | 17 |
| 6 | 42 | 35 | 12 |
| 7 | 5 | 14 | 20 |
| 8 | 4 | 36 | 15 |
| 9 | 42 | 25 | 15 |
| 10 | 29 | 13 | 12 |
| 11 | 40 | 12 | 13 |
| 12 | 50 | 19 | 15 |
| 13 | 49 | 12 | 11 |
| 14 | 45 | 9 | 18 |
| 15 | 11 | 8 | 13 |
| 16 | 25 | 5 | 13 |
| 17 | 17 | 28 | 19 |
| 18 | 33 | 35 | 18 |
| 19 | 2 | 41 | 19 |
| 20 | 7 | 15 | 14 |

Table 2.3 Opening cost and location information of suppliers

| Supplier | X | Y | Opening cost |
|---|---|---|---|
| 21 | 6 | 25 | 12286 |
| 22 | 1 | 50 | 12031 |
| 23 | 11 | 25 | 6995 |
| 24 | 47 | 42 | 8502 |
| 25 | 3 | 33 | 12790 |

As can be seen from the Figure 2.1, vendors 23 and 24 serve with one and two routes, respectively. Customers 12, 13, 5, 14, 11, 4, 9, and 6 are served by vendor 24. Since the assigned customers' demand do not exceed the capacity of vendor, there is one route for this vendor. Vendor 23 is followed by customers 20, 7, 15, 16, 10, 2, 18, 1 and 17 for the first route. Then, second route starts to serve requests 8, 19 and 3.

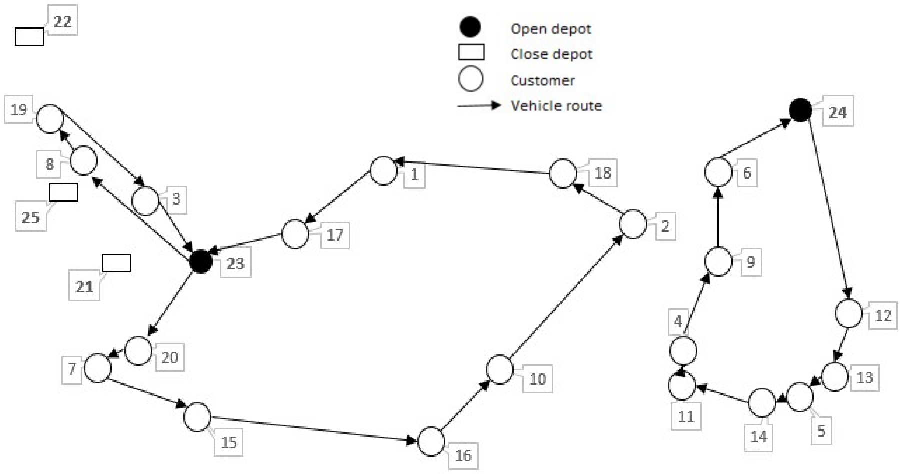


Figure 2.1 Visual illustration of example solution

This study focuses on the location inventory routing problems. It is also considered two significant issues that are not widely taken into consideration by the LIRP studies in the related literature. The first one is the defective manufacturing processes at each supply point. Supply points may have defective rates for each product type produced. These defects could be occurred since lack of raw material and machine breakdowns etc. The other issue is occurring back-order because of the defective rates. These two issues could widely be confronted in real world applications. As far as we are concerned, there are obvious gaps related to these issues in the developed LIRP models in literature. In the following section, we present a literature survey on LIRP to underline the motivation of this thesis.

## 2.2 An Overview on LIRP

According to Farahani et al. (2015) the researches on the LIRP have been a rare occurrence in the related literature. Many researchers integrate only two decisions of the supply chain into one single model (Daskin, Coullard, & Shen, 2002; Shen, Coullard, & Daskin, 2003; Moin & Salhi, 2007; Nagy & Salhi, 2007 and Diabat, Richard, & Codrington, 2013). These models are called as location-inventory, location-routing, and inventory-routing models. On the other hand, the studies that integrate location, inventory and routing decisions at the operational, tactical and strategic levels are the recent works of Miranda & Garrido (2008), Mak & Shen (2009), Miranda & Garrido (2009), Gebennini, Gamberini, & Manzini (2009), Javid & Azad (2010), Sajjadi & Cheraghi (2011), Naseraldin & Herer (2011), Jha, Somani, Tiwari, Chan, & Fernandes (2012), Diabat et al. (2013) and Rafie-Majd, Pasandideh, & Naderi (2018).

LIRP models were firstly introduced by Liu & Lee (2003). They proposed a location- inventory model for single product multiple depots and two-phase heuristic algorithm to find an optimal solution for the problem. They tackled this problem by taking inventory decisions into account.

In another study, Liu & Lin (2005) contrast their proposed method with the other solution techniques. They decomposed the LIRP problem as an inventory-routing problem and a location problem. They proposed a hybrid heuristic integrating tabu search (TS) and simulated annealing (SA). Their hybrid method was able to find global optimum solutions.

Ambrosino & Scutellà (2005) presented two mathematical programming models for the LIRP and they dealt with a case study. They studied distribution problem regarding transportation, facility, depository and inventory decisions with non-homogenous vehicle fleet.

Ma & Davidrajuh (2005) proposed an iterative approach which is applicable for the changing environment conditions as market situations and supply chain processes. Their proposed algorithm composed of a strategic model that is based on a mixed integer linear programming model and a genetic algorithm based tactical model.

Shen & Qi (2007) developed a nonlinear integer programming model for the LIRP in which stochastic customer demands are considered. They designed three-layer supply chain which is comprised of one provider, candidate depots and customers for a single period.

Gebennini et al. (2009) developed a model considering the following decisions: the number of suppliers, finding of their optimum locations and allocating of customers to them and combine concerning inventory control decisions, production rates, and service level and safety stock.

Javid & Azad (2010) developed a hybrid heuristic model to solve location–allocation, routing and inventory decisions at the same time. They proposed the meta-heuristics that integrates tabu search and simulated annealing algorithms for large-scale problem instances.

Mete & Zabinsky (2010) presented a case study about a disaster planning problem as a LIRP and they formulated the problem as a stochastic programming (SP) model.

Huang & Lin (2010) dealt with the inventory routing problem with the multi item replenishment consideration. They formulated the problem as an integration of multi item inventory management and vehicle routing problem, and they solved the problem via Ant Colony Optimization algorithm.

Sajjadi & Cheraghi (2011) studied an extension of LIRP model with multi product, and they offered space for selected warehouses if needed in third party logistics.

Dabiri, Tarokh, & Setak (2012) studied multi product inventory problem, and they employed constructive heuristic to solve it.

Lieckens, Colen, & Lambrecht (2013) developed a strategic decision system to identify the optimum system design and the optimum service distribution tactics for a multi-product, multi-level network for repairable service parts. They formulated the problem as a non-linear programming problem and they solved it employing differential evolution algorithm (DEA).

Guerrero, Prodhon, Velasco, & Amaya (2013) presented a multi-period capacitated location-inventory-routing (LIR) problem. The main objective of this study was to minimize the total costs related facility opening, shipping and inventory for depots and customers. The inventory decisions are considered for both supplier and depots.

Nekooghadirli, Tavakkoli-Moghaddam, Ghezavati, & Javanmard (2014) presented a bi-objective LIRP model considering a multi-period, multi-product supply chain environment.

In another study, Guerrero, Prodhon, Velasco, & Amaya (2015) developed a three-layer supply chain (one supplier, depots and customers) and considered minimizing the depot opening, distribution and inventory costs simultaneously. They solved the problem by using heuristic procedures. Their proposed method was able to optimize an integrated model without decomposing to sub-problems.

Tavakkoli-Moghaddam & Raziei (2016) presented a bi-objective multi-period and multi-product LIRP for two echelon supply chain networks. They proposed a fuzzy approach for dealing with the customers' uncertain demands to optimize two fuzzy objective functions: minimizing total costs and minimizing the shortage of products for each customer.

Gholamian & Heydari (2017) considered a location-inventory-routing problem which is modelled via the METRIC (Multi-Echelon Technique for Recoverable Item Control) approach under stochastic conditions. They developed a hybrid method that comprise of a simulated annealing (SA) method to solve the location-routing problem and a genetic algorithm (GA) to solve the inventory problem.

Rayat, Musavi, & Bozorgi-Amari (2017) proposed a LIRP model having two objectives. These objectives were to minimize total network cost and the expected failure cost, respectively. They considered multi-product, multi-period, partial backordering and failure risk during the process. They employed an archived multi-objective simulated annealing (AMOSA) meta-heuristic to solve the problem.

Hiassat, Diabat, & Rahwan (2017) developed a genetic algorithm for a LIRP for perishable products.

Rafie-Majd et al. (2018) presented a three-echelon supply chain in form of an integrated LIRP for perishable products in a finite planning horizon by considering heterogeneous transportation fleet. They used a Lagrangian Relaxation algorithm to solve the proposed mathematical model and introduced a heuristic feasibilizer algorithm.

Solution techniques used in the literature for LIRP models are summarized in Table 2.4. Table 2.4 also shows the differences between current study and previous studies.

Table 2.4 Solution techniques used in literature for LIRP models

| Author(year) | Cost components | | | | | | | | | Solution method |
|---|---|---|---|---|---|---|---|---|---|---|
| | Setup cost | Fixed cost | Transportation cost | Holding cost | Shortage cost | Routing cost | Ordering cost | Defective items | Multi-product | |
| Liu & Lee (2003) | √ | √ | √ | √ | √ | √ | √ | | | Heuristic |
| Liu & Lin (2005) | √ | √ | √ | √ | √ | √ | √ | | | TS – SA |
| Ambrosino & Scutellà (2005) | | √ | √ | √ | | | | | | CPLEX |
| Ma & Davidrajuh (2005) | √ | √ | √ | √ | | | | | | MIP/ Genetic algorithm |
| Shen & Qi (2007) | √ | √ | √ | √ | | | √ | | | Lagrangian relaxation approach |
| Gebennini et al. (2009) | | √ | √ | √ | √ | | | | | MINLP |
| Javid & Azad (2010) | | √ | √ | √ | | √ | √ | | | TS- SA |
| Mete & Zabinsky (2010) | | | √ | √ | √ | | | | | MIP/ SP |
| Huang & Lin (2010) | | | √ | | √ | | | | | ACO |
| Sajjadi & Cheraghi (2011) | | √ | √ | √ | √ | | √ | | √ | SA |
| Dabiri et al. (2012) | | √ | √ | √ | | √ | | | √ | MILP/CPLEX/Heuristic |
| Lieckens et al. (2013) | √ | √ | √ | √ | | | √ | | √ | MILP/DEA |
| Guerrero et al. (2013) | | √ | √ | √ | | √ | √ | | | MIP/Heuristic |
| Nekooghadirli et al. (2014) | √ | √ | √ | √ | | √ | √ | | √ | Hybrid Meta-heuristic |
| Guerrero et al. (2015) | √ | | | √ | | √ | √ | | | Heuristic |
| Tavakkoli-Moghaddam & Raziei (2016) | | √ | √ | √ | √ | √ | √ | | √ | MIP/CPLEX |
| Gholamian & Heydari (2017) | | √ | √ | √ | √ | √ | √ | | | GA- SA |
| Rayat et al. (2017) | √ | √ | √ | √ | √ | √ | √ | √ | √ | AMOSA |
| Hiassat et al. (2017) | | √ | √ | √ | | | | | | MIP/ Genetic algorithm |
| Rafie-Majd et al. (2018) | √ | √ | √ | √ | | | √ | | √ | MINLP/ Lagrangian relaxation algorithm |

According to Table 2.4 the LIRP models presented in literature generally focus on reliable manufacturing process with single product. In this thesis, we consider that each vendor could have defective processes because of predictable or unpredictable cases during the manufacturing process. In this study, it is assumed that every product has defective rate and, there are inspection costs in addition to the other inventory costs. The cases mentioned above can occur in real life.

As stated before, we deal with the LIRP while shortages are allowed in a multi-product environment. A mathematical model, an integration of a LRP and a joint replenishment economic order quantity (EOQ) model with backorders, is developed to formally describe the problem. Additionally, we propose a hybrid meta-heuristic approach called as ALNS-VNS to solve the problem. To the best of our knowledge, there is a lack of the models that consider location, allocation, routing and inventory decisions simultaneously in the related literature. The first gap that we have realized and try to fulfil by the developed model is taking backordering for shortage into consideration. Because of the probability of disruptions during the manufacturing process, shortages could be occurred. These shortages include only backordered demands. It means that all demands meet at the end of the period. The second gap is also taking varieties of products into account. In real life, customers have demands for many products. Supply points must meet these demands for different products.

### *2.3 An Overview on Large Neighborhood Search Algorithm*

This section explains the framework of the neighborhood search algorithms, especially the large neighborhood search (LNS) algorithm which is the basis of the ALNS algorithm. Then, a literature survey on LNS is given briefly.

#### *2.3.1 Neighborhood Search Algorithms*

Neighborhood search algorithms which are also called local search algorithms are based on exploring the neighborhoods of a solution in the search space. It is an improvement algorithm which iteratively tries to improve the solution on hand by searching its ‘neighbors’. The term *neighbor* denotes the similarity between a solution in an iteration and any other solution derived from this solution. This relationship is quoted ‘neighborhood’. The neighborhood involves all solutions that might be created by altering the current solution.

One of the most important issue when designing a neighborhood search algorithm is to decide the size of the neighborhood. Searching a larger neighborhood could lead to consume extra time and get stuck in local optima. In order to avoid such potential disadvantages, the size must be determined carefully.

The neighborhood search algorithms could be categorized as subsets of *Very Large-Scale Neighborhood* (VLSN) search algorithms if the neighborhood search grows exponentially with the problem size (Pisinger & Ropke, 2010). All VLSN algorithms are rely on searching a large neighborhood space in improving the solution. On the other hand, exploring a large search space require more time, it could be prevented by limiting the search space. VLSN could be classified to three classes: 1) methods based upon restrictions of the problem which are solvable in polynomial time, 2) variable depth methods , 3) improvement methods based on network flow techniques, as stated in Ahuja, Ergun, Orlin, & Punnen (2002). For a detailed survey on VLSN algorithms the reader may refer to Ahuja et al. (2002).

### *2.3.2 Large Neighborhood Search*

Large Neighborhood Search (LNS) heuristic, belongs to the class of VLSN algorithms, was firstly introduced by Shaw (1997). The extended version of this algorithm is known as Adaptive Large Neighborhood Search (ALNS) and was proposed by Ropke & Pisinger (2006). The algorithm takes the initial solution and improves it by alternately implementing the destruction and repair procedures. The destruction procedure is based on destroying a part of the current solution using removal operators while the repair procedure reinserts the ruined solution with insertion operators. By using these operators, the algorithm replaces previous solution with a better solution at each iteration. The pseudocode for an LNS algorithm is presented in Figure 2.2.

```
1  Function LNS (x ϵ {solutions}, q ϵ ℕ )
2     solution x_best = x;
3     repeat
4        x'=x;
5        remove q requests from x'
6        reinsert removed requests into x';
7        if (c(x') < c(x_best))  then
8           x_best=x';
9        if accept (x', x)  then
10          x=x';
11    until  stop criterion met
12 return x_best;
```

Figure 2.2 Pseudocode of LNS (Ropke & Pisinger, 2006)

According to the pseudocode given above, an initial solution $x$ is created, and parameter $q$ is given at the initialization phase of the algorithm to determine the scope of the search. The variable $x_{best}$ is the best solution obtained during the search. Lines 5 and 6 are the heuristic mechanism in the algorithm. In line 5, $q$ requests are removed from the current solution $x'$, then removed requests are reinserted into the solution in line 6. The performance and effectiveness of the algorithm is relying on removal and insertion operators' choices. Then, the best solution is updated, and it is decided if the new one will be accepted. This decision is performed with an acceptance criterion mechanism. There are several acceptance criterions in the literature. In this study, simulated annealing's acceptance criterion has been used. The algorithm stops when the stopping condition is met. In this study, the number of iterations is used as a stopping criterion.

The parameter $q \in \mathbb{N}$ determines the *degree of destruction,* as stated in Shaw (1997). It is the most important parameter that must be specified attentively while implementing the removal heuristic as it directly effects the performance of the algorithm. If $q$ is equal to zero, it means that there are no requests to remove. Removing a small part of the solution may lead to have trouble searching. Otherwise, if $q$ is equal to number of total requests in the problem, it causes the problem to be solved from the beginning. In addition, the larger $q$ leads to make insertion heuristic slower. Then, it affects negatively the quality of the solution. Shaw (1997) presented the *degree of destruction* increased step by step while Pisinger & Ropke (2010) proposed the *degree of destruction* which is chosen randomly in each iteration by defining a range dependent on the size of the instance.

Below, we briefly explain several removal and insertion heuristics from the related literature. The removal heuristics are described in §2.3.2.1 and the insertion heuristics are described in §2.3.2.2. LNS algorithm uses only one method for each removal and insertion procedure while ALNS uses multiple removal and insertion methods.

#### *2.3.2.1 Removal Heuristics*

A removal heuristic destructs the current solution. The destruction could be performed by removing defined number of requests from the feasible current solution or closing a supply point, thus, removing the requests served by this supply point from the current solution. It is important to remember that the *degree of destruction* must be well adjusted. This parameter has an important influence on the effectiveness of the algorithm. In the following, eight removal heuristics which are widely used in the literature are briefly explained. The Random Removal heuristic is the simplest method. This heuristic was proposed by Ropke & Pisinger (2006). This operator removes $q$ requests uniformly at random. Another removal heuristic proposed by the same authors is the Worst Removal. The main idea of this heuristic is to remove the worst requests which cause the huge costs in the solution. Similarly, Worst Time Removal which is used by Majidi, Hosseini-Motlagh, & Ignatius (2018) considers the deviation of arrival time of the vehicle at each request and then removes the request with the high deviation. Route Destruction heuristic (Mancini, 2016) removes all the requests from a route which is chosen randomly. Hemmelmayr, Cordeau, & Crainic (2012) proposed a removal method that is based on swapping, opening and removing a satellite facility. Another effective removal method was suggested by Shaw (1997, 1998). It is called as ‘related removal’. It tries to remove a set of requests that have similar characteristics. The idea is to remove the similar requests so that giving a chance to have re-inserted them in a profitable position.

#### *2.3.2.2 Insertion Heuristics*

After applying the removal heuristic, the partial solution must be repaired to make it feasible again. Each repair operator takes a partial solution as input and inserts the requests in removal list to partial solution according to characteristics of itself to return it a complete feasible solution. The most applied repair operators in literature are described in this section. The Basic Greedy heuristic is the simplest insertion heuristic. It tries to insert every removed request from the removal list in the cheapest possible position. Another heuristic is Regret Insertion proposed by Ropke & Pisinger (2006). Unlike greedy insertion, regret heuristic takes the second cheapest, third cheapest etc. into consideration besides minimum cost. The last one, Random Insertion, also searches the feasible positions available for a removed request and then randomly chooses one of the positions for insertion. This is performed until all the requests of the removal list are inserted.

### ***2.3.3 Literature Survey on LNS***

The focus of this section is the recent studies on the LNS and ALNS algorithms in literature. The LNS heuristics have been successfully employed in several fields of combinatorial optimization, especially routing problems.

#### *2.3.3.1 Routing Problems*

There are many applications of LNS to 'Vehicle Routing Problems' and its variants, as from the introduction of the LNS heuristic proposed by Shaw (1997).

Breunig, Schmid, Hartl, & Vidal (2016) used this heuristic to tackle the two-echelon location routing problem. Lian, Milburn, & Rardin (2016) developed a heuristic based on LNS algorithm for multi-objective vehicle routing problem. François, Arda, Crama, & Laporte (2016) published a study for vehicle routing problem which considers that each vehicle has several routes during a period to serve a set of customers.

Eskandarpour, Dejax, & Péton (2017) proposed an LNS algorithm to solve a four-layer single period multi-product supply chain network design problem. Their model included locations of production point and distribution centers as well as the choice of transportation methods. LNS determined location decisions while greedy heuristic determined product flow decisions and transportation modes.

Ho & Szeto (2017) implemented the LNS on the multiple-vehicle bike repositioning problems with considering penalty cost in the objective function. They proposed hybrid algorithm which consists of the LNS and tabu search algorithms.

Suh & Ryerson (2017) developed the LNS in order to optimize the rerouting of flights and minimize the sum of passenger travel time and waiting time. They used this heuristic to decide a coordinated traffic management for diverted flights in a real-world airport outage.

In another study, Hojabri, Gendreau, Potvin, & Rousseau (2018) studied on a synchronization problem related to arrival of vehicles at different customer locations and used the LNS to handle this problem.

The ALNS, extension of the LNS, was developed by Ropke & Pisinger (2006) as described previously in Section 2.3.2. They applied the algorithm to the pickup and delivery problem considering time windows and limited number of vehicles. They applied the heuristic by using several insertion and removal heuristics. The heuristics were selected according to the their past performance. The performance of the proposed algorithm was evaluated with more than 350 benchmark instances. They improved the best-known solutions for more than 50% of the instances.

Belo-Filho, Amorim, & Almada-Lobo (2015) presented a model for operational combined production and distribution planning problem with perishable goods. For this problem, perishability and number of products were hard constraints. To solve the problem, they proposed an ALNS  algorithm.

Mancini (2016) handled a multi-period multi-depot VRP having an objective of minimizing total transportation cost with an ALNS algorithm. The problem considered heterogeneous fleet consisting of vehicles that have different capacity, characteristics and usage costs.

Koç (2016) introduced a periodic location-routing problem with time windows (PLRPTW), along with two particular cases; i.e. heterogeneous and homogeneous fleets. A unified meta-heuristic (U-ALNS) was proposed in order to solve the problem. According to output of this study, the U-ALNS is capable of efficiently solving this problem category without any modification.

Grimault, Bostel, & Lehuédé (2017) suggested an ALNS algorithm for full truckload pickup and delivery problem within time windows in public works companies. The performance of the algorithm was tested on real case instances.

Smith & Imeson (2017) presented an algorithm based on ALNS in order to solve the generalized travelling salesman problem. They proposed a special unified insertion operator which contains random, nearest, and farthest insertion mechanisms, and a unified removal method.

Canca, De-Los Santos, Laporte, & Mesa (2017) modeled and solved the railway network design and line planning problem. They employed the ALNS algorithm for the design of the railway transit network in a real-life instance.

Palomo-Martinez, Salazar-Aguilar, Laporte, & Langevin (2017) solved a real-life distribution problem about daily distribution planning of a fruit supplier. They modeled the problem as a multi-product split delivery capacitated team orienteering problem with incomplete service and soft time windows.

Žulj, Kramer, & Schneider (2018) addressed an order-batching problem. The aim was to optimize the grouping of customer orders by decreasing the total length of tours. They proposed a hybrid meta-heuristic based on ALNS and tabu search to solve larger instances within acceptable run-time.

Majidi et al. (2018) presented a nonlinear mixed integer programming model for the pollution-routing problem within simultaneous pickup and delivery problem. The main was to minimize fuel consumption and emissions as well as total length of the routes. They also proposed an ALNS algorithm including new removal and insertion heuristics to solve the problem.

Related to the concept of this study, we should consider the routing problems which take into inventory decisions account. There are limited studies that consider three decisions (allocation, routing and inventory) simultaneously by implementing the LNS heuristics in literature. Following this section, we present the routing problems which considers inventory constraints.

The periodic inventory routing problem (PIRP) determines the routing and inventory decisions in a periodic time with the minimal cost criterion. There are many studies addressing IRP and its variants and solving the problem via LNS and ALNS algorithms (Coelho, Cordeau, & Laporte, 2012; Song & Furman, 2013; Riquelme-Rodríguez, Langevin, & Gamache, 2014; Aksen, Kaya, Salman, & Tüncel, 2014; Hemmati, Stalhane, Hvattum, & Andersson, 2015; Iassinovskaia, Limbourg, & Riane, 2017).

Riquelme-Rodríguez, Langevin, & Gamache (2016) presented a location arc routing problem with inventory constraints. They developed a mathematical model for vehicle allocation and routing problem based on the location of water depots along the network. They used the ALNS to modify the initial routing of vehicles.

Hemmati, Hvattum, Christiansen, & Laporte (2016) modeled and solved a multi-product short sea inventory-routing problem. The problem considered many-to-many distribution and multiple products. They proposed an iterative hybrid meta-heuristic algorithm called as Hybrid Cargo Generating and Routing (HCGR) to solve the problem. They used the ALNS to solve the ship routing and scheduling problem.

## 2.4 Chapter Conclusions

This chapter described the standard LIRP which is the basis of the addressed problem in this study. Then, a brief classification of the LRPs and the related literature about the LIRP were given. It should be noted that there is no study dealing with multi-product and defective manufacturing process in a supply chain design in existing literature. So, this study aims to fulfill this research gap. Following the chapter, the neighborhood search algorithms are summarized and the LNS algorithm that constitutes the basis of the proposed hybrid ALNS algorithm was defined. Finally, literature on the LNS and ALNS algorithms were summarized briefly.

The following chapter presents the developed mathematical model for the multi-product LIRP with shortages and model validation on a small-scale problem.

# CHAPTER THREE
# PROBLEM FORMULATION

## 3.1 Definition of the Problem

In this section, a mathematical formulation of the problem is presented. The formulation has two parts: an economic production quantity (EPQ) model with shortages backordered presented by L.F. Hsu & J.T. Hsu (2016) and an LRP model. Section 3.1 describes the related EPQ model in details and Section 3.2 describes the full mathematical formulation of the LIRP model, integration of EPQ and LRP models. The notations and their definitions for both the LIRP and EPQ models are presented in Table 3.1.

The presented LIRP considers a distribution network with multi-product and homogeneous fleet (i.e., vehicles with same capacities) assumptions. The MINLP model determines the amount of products sent from the vendors to the customers, and the routes of vehicles to satisfy customers' demands in each period for each type of products.

The main assumptions of the presented model are listed below:

- The location of vendors is known.
- Multi-products are considered.
- Deterministic demands are considered.
- All vehicles are considered to have the same capacity.
- The capacity of vendors and vehicles are limited.
- Route decision between vendors and customers is considered.
- The transportation cost contains the Euclidian distance between the vendors and customers and the vehicle fixed cost.

The aim of the entire model is to determine whether to locate the facilities to the potential supply nodes or not and determine which demand points will be served by which supply node with the aim of minimizing location, routing, production and inventory related costs.

Table 3.1 Notations and their definitions used for the presented model

| Notation | | Definition |
|---|---|---|
| **Indices** | | |
| $f$ | | Supply points |
| $i, j, k, l$ | | Demand points |
| $p$ | | Products |
| **Sets** | | |
| $F$ | | Set of potential supply points |
| $I$ | | Set of demand points to be serviced |
| $P$ | | Set of products |
| **Inventory Model** | $Q$ | The production quantity per lot |
| | $DF$ | The annual demand |
| | $PR$ | The annual production rate |
| | $B$ | The maximum backorder quantity per lot |
| | $c$ | The production cost per unit |
| | $S$ | The setup cost per production run |
| | $c_i$ | The inspection (screening) cost per unit |
| | $c_h$ | The holding cost per unit per year |
| | $c_b$ | The backordering cost per unit per year |
| | $\gamma$ | The defective rate (i.e. the fraction of defective items at the time of production) |
| | $TC$ | The total annual cost |
| **Mathematical Model** | **Parameters** | |
| | $O_f$ | Fixed opening cost of supply point *f* (include fixed vehicle cost) |
| | $D_{ip}$ | Annual demand of customer *i* for product type *p* |
| | $C_{ij}$ | Travelling cost from demand point *i* to demand point *j*, assumed symmetric |
| | $T_{if}$ | Travelling cost from demand point *i* to supply point *f*, assumed symmetric |
| | $V$ | Vehicle capacity |
| | $\alpha_p$ | Volume/Size unit coefficient of product *p* |
| | $c^{pf}$ | The production cost per unit of product *p* at the supply point *f* |
| | $c_i^{pf}$ | The inspection (screening) cost per unit of product *p* at the supply point *f* |
| | $c_h^{pf}$ | The holding cost per unit of product *p* at the supply point *f* per year |
| | $c_b^{pf}$ | The backordering cost per unit of product *p* at the supply point *f* per year |
| | $S^{pf}$ | The setup cost per production run of product *p* at the supply point *f* |
| | $PR^{pf}$ | The annual production rate of product *p* at the supply point *f* |
| | $\gamma^{pf}$ | The defective rate of product *p* at the supply point *f* |

Table 3.1 continues

| | | |
|---|---|---|
| | **Decisions Variables** | |
| | $Z_f \in \{0,1\}$ | Equals to 1 if supply point *f* is used, 0 otherwise |
| | $X_{if} \in \{0,1\}$ | Equals to 1 if customer *i* assigned to supply point *f*, 0 otherwise |
| | $Y_{ijf} \in \{0,1\}$ | Equals to 1 if customer *i* immediately precede customer *j* on the route of supply point *f*, 0 otherwise |
| | $E_{if} \in \{0,1\}$ | Equals to 1 if customer *i* is firstly visited on the route of supply point *f*, 0 otherwise |
| | $L_{if} \in \{0,1\}$ | Equals to 1 if customer *i* is lastly visited on the route of supply point *f*, 0 otherwise |
| | $DF^{pf}$ | The total annual demand of product *p* supplied by supply point *f* |
| | $Q^{pf}$ | The production quantity of product *p* at supply point *f* per lot |
| | $B^{pf}$ | The maximum backorder quantity of product *p* at supply point *f* per lot |
| | $t_f$ | The total number of tours taken by the vehicle of factory *f* while serving the demand points |
| | **Auxiliary Variables** | |
| | $w_{ijf} \in \{0,1\}$ | Equals to 1 if demand point *i* is formerly visited than demand point *j* on the route of supply point *f*, 0 otherwise |

## 3.2 Backordered EPQ Model

The inventory model used in this study is adapted from the model presented by L.F. Hsu & J.T. Hsu (2016), and the notations of this inventory model with their definitions are given in Table 3.1. Their EPQ model considers a facility producing one product and determines the optimal production lot size and backorder quantity for the supplier under an imperfect production process. Additionally, the defective items thought to be kept in stock and sold at the end of the production period within each cycle (see Figure 3.1). The EPQ model developed by L.F. Hsu & J.T. Hsu (2016) has the following basic assumptions.

- The demand rate is known, constant and continuous.
- The production rate is constant and greater than the demand rate.
- The production process is imperfect with a defective rate $\gamma$ which is known with a constant value.
- During the production period, the production and demand processes occur concurrently, and the inventory is built up at a rate of $PR\ (1-\gamma) - DF$.

- To guerentee that the manufacturer has enough production capacity to meet the customers' demand, it is assumed that the maximum possible value of $\gamma$ is less than $1 - DF/PR$.
- Once an item is produced, it is checked with an inspection cost of $c_i$ , and the check procedure is error free.
- All shortages are backordered.
- All requests are eager to wait for a new distribution if a shortage condition occurs.

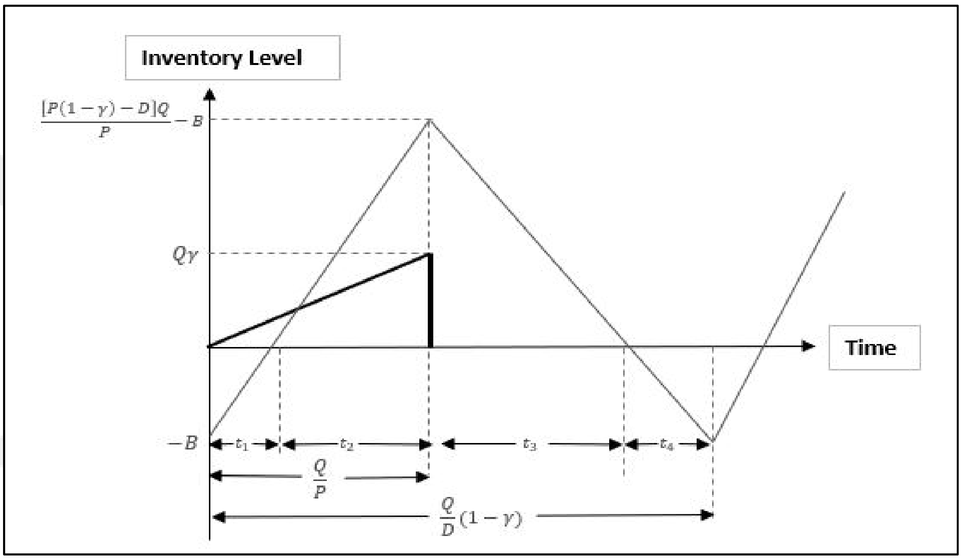


Figure 3.1 Behavior of inventory level (L.F. Hsu & J.T. Hsu, 2016)

In this study, we implement this EPQ model for the multi-manufacturer multi-product case and we try to optimize the production process as well as the distribution process within a cost-effective manner.

Referring to (L.F. Hsu & J.T. Hsu, 2016) the total annual cost can be obtained via Equation (3.1).

$$TC(Q,B) = c\frac{DF}{(1-\gamma)} + c_i\frac{DF}{(1-\gamma)} + S\frac{DF}{Q(1-\gamma)} + \frac{1}{2}c_h\frac{\left(Q\left(1-\gamma-\frac{DF}{PR}\right)-B\right)^2}{Q\left(1-\gamma-\frac{DF}{PR}\right)} + \frac{1}{2}c_b\frac{B^2}{Q\left(1-\gamma-\frac{DF}{PR}\right)} + \frac{1}{2}c_h\frac{Q\gamma DF}{PR(1-\gamma)} \tag{3.1}$$

As can be seen from Equation 3.1 the total annual cost has 6 components: the total annual production cost, the total annual setup cost, the total annual holding cost, the total annual holding cost of defective items, the total annual backordering cost, and the total annual inspection cost. By taking the first derivative of $TC(Q, B)$ with respect to $Q$ and $B$ and setting the results to zero, they derived the optimal production lot size $Q$ and maximum backorder quantity $B$ as given by Equations 3.2 and 3.3, respectively.

$$Q = \sqrt{\frac{2DFS}{c_h} \frac{c_h + c_b}{\left(c_b(1-\gamma)\left(1-\gamma-\frac{DF}{PR}\right)+(c_h+c_b)\gamma\frac{DF}{PR}\right)}} \tag{3.2}$$

$$B = Q\left(1 - \gamma - \frac{DF}{PR}\right)\frac{c_h}{(c_h+c_b)} \tag{3.3}$$

The above mentioned EPQ model assumes that a factory produces one product; however, we deal with the multi-factory, multi-customer and multi-product case within the context of this study. There is a set of potential supply points each is able to produce a set of products for serving a set of demand points. That is to say, each utilized supply point serves a sub-set of demand points via its own EPQ model. And therefore, each utilized supply point will have its own total annual cost function, maximum backorder quantity level and specific production lot size. Hence, the inventory model given above required some modifications so as to be suitable for the multi-factory, multi-customer and multi-product case. As a result, Equations (3.1), (3.2) and (3.3) can be rewritten as Equations (3.4), (3.5) and (3.6) respectively.

$$TC^{pf}(Q^{pf}, B^{pf}) = c^{pf}\frac{DF^{pf}}{(1-\gamma^{pf})} + c_i^{pf}\frac{DF^{pf}}{(1-\gamma^{pf})} + S^{pf}\frac{DF^{pf}}{Q^{pf}(1-\gamma^{pf})} +$$
$$\frac{1}{2}c_h^{pf}\frac{\left(Q^{pf}\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)-B^{pf}\right)^2}{Q^{pf}\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)} + \frac{1}{2}c_b^{pf}\frac{\left(B^{pf}\right)^2}{Q^{pf}\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)} + \frac{1}{2}c_h^{pf}\frac{Q^{pf}\gamma^{pf}DF^{pf}}{PR^{pf}(1-\gamma^{pf})} \tag{3.4}$$

$$Q^{pf} = \sqrt{\frac{2DF^{pf}S^{pf}}{c_h^{pf}} \frac{c_h^{pf}+c_b^{pf}}{\left(c_b^{pf}(1-\gamma^{pf})\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)+\left(c_h^{pf}+c_b^{pf}\right)\gamma^{pf}\frac{DF^{pf}}{PR^{pf}}\right)}} \tag{3.5}$$

$$B^{pf} = Q^{pf}\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)\frac{c_h^{pf}}{\left(c_h^{pf}+c_b^{pf}\right)} \quad (3.6)$$

Equation 3.4 calculates the total annual cost of related supply chain as the summation of the individual total annual costs of the utilized supply point, while Equation 3.5 determines the production lot sizes of each utilized supply point $f$ for each product $p$, separately. Finally, Equation 3.6 determines the maximum backorder quantities for each product $p$ at each supply point $f$.

### 3.3 Mathematical Model

In this section, the inventory strategy presented in Section 3.1 will be developed in the form of MINLP model and then integrated with the presented LRP model.

Model assumptions are given following;

- There is one vehicle for each opened vendor. Each vehicle can take more than one trip.
- The total demand served by a vehicle can not exceed the capacity of a vehicle.
- Each customer must be visited once and only once by exactly one vehicle.
- Each customer's demand must be fulfilled.
- The total demand of customers assigned to a vendor cannot exceed the capacity of the vendor.
- A route starts from the vendor and visits customers in an ordered sequence before returning to the vendor.

The LIRP can be formulated as the following nonlinear mixed integer programming problem:

### *3.3.1 Objective Function*

$$
\begin{aligned}
Min \quad & \sum_{f=1}^{F}\left(O_f Z_f\right) + \sum_{f=1}^{F}\left(t_f\left(\sum_{i=1}^{I}\sum_{j=1}^{I}\left(C_{ij}Y_{ijf}\right) + \sum_{i=1}^{I}\left(T_{if}E_{if} + T_{if}L_{if}\right)\right)\right) + \\
& \sum_{f=1}^{F}\sum_{p=1}^{P}\left(c^{pf}\frac{DF^{pf}}{(1-\gamma^{pf})}\right) + \sum_{f=1}^{F}\sum_{p=1}^{P}\left(c_i^{pf}\frac{DF^{pf}}{(1-\gamma^{pf})}\right) + \\
& \sum_{f=1}^{F}\sum_{p=1}^{P}\left(S^{pf}\frac{DF^{pf}}{Q^{pf}(1-\gamma^{pf})}\right) + \frac{1}{2}\sum_{f=1}^{F}\sum_{p=1}^{P}\left(c_h^{pf}\frac{\left(Q^{pf}\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)-B^{pf}\right)^2}{Q^{pf}\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)}\right) + \\
& \frac{1}{2}\sum_{f=1}^{F}\sum_{p=1}^{P}\left(c_b^{pf}\frac{(B^{pf})^2}{Q^{pf}\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)}\right) + \\
& \frac{1}{2}\sum_{f=1}^{F}\sum_{p=1}^{P}\left(c_h^{pf}\frac{Q^{pf}\gamma^{pf}DF^{pf}}{PR^{pf}(1-\gamma^{pf})}\right)
\end{aligned}
\tag{3.7}
$$

The objective function (3.7) presents the total cost consisting of eight terms. The first term is the  total opening cost of vendors. Term 2 express transportation costs from the vendor to requests, term 3 is the total annual production cost, term 4 is the total annual inspection (screening) cost, term 5 is the total annual setup cost, term 6 is the total annual holding cost, term 7 is the total annual backordering cost and term 8 is the annual holding cost of defective items for each vendor and product, respectively.

The total annual production and the inspection costs are independent of the production lot size $Q$ and backorder quantity $B$, the related costs to defining the lot size $Q$ and backorder quantity $B$ involve only the setup, holding and backordering costs.

### *3.3.2 Assignment constraints*

$$\sum_{i=1}^{I} X_{if} = 1 \qquad f \in \{1, \ldots, F\} \tag{3.8}$$

Constraint (3.8) assigns each request to exactly one vendor.

### *3.3.3 Vendor opening constraints*

$$Z_f - \frac{\sum_{i=1}^{I} X_{if}}{i} \geq 0 \qquad f \in \{1, \ldots, F\} \tag{3.9}$$

Constraint (3.9) ensures that at least one request must be assigned in order to open the vendor.

### *3.3.4 Routing constraints*

$$w_{iif} = 0 \qquad f \in \{1, \dots, F\};\ i \in \{1, \dots, I\} \tag{3.10}$$

$$w_{ijf} + w_{jif} \le \left(1 - X_{if} + X_{jf}\right) \qquad i, j \in \{1, \dots, I\} | j \ne i;\ f \in \{1, \dots, F\} \tag{3.11}$$

$$w_{ijf} + w_{jif} \le X_{if} + X_{jf} \qquad i, j \in \{1, \dots, I\} | j \ne i;\ f \in \{1, \dots, F\} \tag{3.12}$$

$$w_{ijf} + w_{jif} \le \left(1 + X_{if} - X_{jf}\right) \qquad i, j \in \{1, \dots, I\} | j \ne i;\ f \in \{1, \dots, F\} \tag{3.13}$$

$$w_{ijf} \ge w_{ikf} + w_{kjf} - 1 \qquad i, j, k \in \{1, \dots, I\};\ f \in \{1, \dots, F\} \tag{3.14}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{I} w_{ijf} = \sum_{i | \left(i < \sum_{k=1}^{I} X_{kf}\right)}^{I} i \qquad f \in \{1, \dots, F\} \tag{3.15}$$

$$\sum_{j=1}^{I} \sum_{f=1}^{F} Y_{ijf} \le 1 \qquad i \in \{1, \dots, I\} \tag{3.16}$$

$$Y_{ijf} + Y_{jif} \le 1 \qquad i, j \in \{1, \dots, I\};\ f \in \{1, \dots, F\} \tag{3.17}$$

$$\sum_{f=1}^{F} Y_{iif} = 0 \qquad i \in \{1, \dots, I\} \tag{3.18}$$

$$\left|X_{if} + X_{jf} - 1\right| - \left|\sum_{k=1}^{I} \left(w_{ikf}\right) - \sum_{l=1}^{I} \left(w_{jlf}\right) - 1\right| \le Y_{ijf} \qquad i, j \in \{1, \dots, I\};\ f \in \{1, \dots, F\} \tag{3.19}$$

$$X_{if} - \left(\sum_{j=1}^{I} w_{ijf}\right) \le L_{if} \qquad i \in \{1, \dots, I\}; f \in \{1, \dots, F\} \tag{3.20}$$

$$X_{if} - \left(\sum_{j=1}^{I} w_{jif}\right) \le E_{if} \qquad i \in \{1, \dots, I\}; f \in \{1, \dots, F\} \tag{3.21}$$

Constraint (3.10) prevents sorting a request with itself. Constraints (3.11, 3.12, 3.13, 3.17 and 3.19) ensures that two requests would be sorted by priority if both have been assigned to the same vendor.

Constraint (3.14) guarantees that three requests would be sorted if all of them have been assigned to the same vendor. If request *i* precedes from request *j* and there is the same relationship between request *k* and *j*, request *i* precedes from request *k*.

Constraint (3.15) is the sub-tours elimination constraint of vehicle routes. Constraint (3.16) ensures that it can be most one request to visit after request *i*. Constraint (3.18) guarantees that each request in any vendor would not have a processor from itself. Constraints (3.20) and (3.21) is about that first and last requests.

### *3.3.5 Capacity constraints*

$$\frac{\left(\sum_{i=1}^{I}\sum_{p=1}^{P}\left(\alpha_p D_{ip} X_{if}\right)\right)}{V_f} \leq t_f \qquad f \in \{1, \ldots, F\} \tag{3.22}$$

$$\sum_{i=1}^{I}\left(D_{ip}X_{if}\right) - PR^{pf}\left(1-\gamma^{pf}\right) \leq 0 \qquad f \in \{1, \ldots, F\}\,;\; p \in \{1, \ldots, P\} \tag{3.23}$$

$$PR^{pf} - Q^{pf} \geq 0 \qquad f \in \{1, \ldots, F\}\,;\; p \in \{1, \ldots, P\} \tag{3.24}$$

Constraint (3.22) ensures that the remaining demand must not exceed vehicle capacity at any time. Constraint (3.23) ensures that each request's demand for each product must be satisfied from each vendor. And constraint (3.24) guarantees that the total demand of requests assigned to a vendor must not exceed the capacity of the vendor.

### *3.3.6 The annual demands of factories*

$$DF^{pf} = \sum_{i=1}^{I}\left(D_{ip}X_{if}\right) \qquad f \in \{1, \ldots, F\}\,;\; p \in \{1, \ldots, P\} \tag{3.25}$$

Constraint (3.25) ensures that the total demand of requests assigned to a vendor must not exceed the vendor's total annual demand.

### *3.3.7 The production quantity per lot*

$$Q^{pf} = \sqrt{\frac{2DF^{pf}S^{pf}}{c_h^{pf}} \frac{c_h^{pf} + c_b^{pf}}{\left(c_b^{pf}(1-\gamma^{pf})\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right) + \left(c_h^{pf}+c_b^{pf}\right)\gamma^{pf}\frac{DF^{pf}}{PR^{pf}}\right)}}$$

$$f \in \{1, \dots, F\}; \;\; p \in \{1, \dots, P\} \quad (3.26)$$

Constraint (3.26) shows that the production lot sizes of each utilized supply point *f* for each product *p*.

### *3.3.8 Backorder Quantities*

$$B^{pf} = Q^{pf}\left(1-\gamma^{pf}-\frac{DF^{pf}}{PR^{pf}}\right)\frac{c_h^{pf}}{\left(c_h^{pf}+c_b^{pf}\right)} \quad f \in \{1, \dots, F\}; \;\; p \in \{1, \dots, P\} \qquad (3.27)$$

Constraint (3.27) shows that the maximum backorder quantities for each product *p* at each supply point *f*.

## 3.4 Model Validation

The mathematical model was coded in GAMS and solved by Mixed Integer Nonlinearly Constrained Optimization Solver of SCIP on neos-server.org/neos/. To verify the proposed model, a sample of LIRP problem sets is used as a test problem.

### *3.4.1 Data sets and parameter settings*

The problem data used to validate the model is created randomly. All routes and vendors have limited capacities. The number of vendors, customers and products equal to 6, 6 and 2 respectively, and the vehicle capacity is set to 1000. Each demand is constant and integer for each product. Capacity of the vendors and inventory costs are also integers except travelling costs. Travelling costs are also generated randomly. Tables 3.2, 3.3 and 3.4 show all the problem data.

Table 3.2 Values of the parameters for product type

| | | Vendors | | | | | |
|---|---|---|---|---|---|---|---|
| **Products** | **Parameters** | **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | $D_{ip}$ | 350 | 355 | 360 | 365 | 370 | 395 |
| | $c^{pf}$ | 15 | 10 | 15 | 10 | 10 | 15 |
| | $c_i^{pf}$ | 2 | 5 | 3 | 4 | 5 | 3 |
| | $c_h^{pf}$ | 4 | 5 | 5 | 4 | 5 | 4 |
| | $c_b^{pf}$ | 12 | 8 | 10 | 10 | 9 | 8 |
| | $S^{pf}$ | 100 | 150 | 110 | 120 | 160 | 140 |
| | $PR^{pf}$ | 1500 | 1750 | 1600 | 1550 | 1700 | 1650 |
| | $\gamma^{pf}$ | 0.04 | 0.04 | 0.03 | 0.05 | 0.05 | 0.05 |
| 2 | $D_{ip}$ | 445 | 440 | 435 | 430 | 425 | 400 |
| | $c^{pf}$ | 20 | 20 | 25 | 15 | 20 | 30 |
| | $c_i^{pf}$ | 4 | 4 | 2 | 4 | 3 | 4 |
| | $c_h^{pf}$ | 5 | 4 | 4 | 4 | 4 | 5 |
| | $c_b^{pf}$ | 10 | 12 | 10 | 9 | 9 | 8 |
| | $S^{pf}$ | 125 | 145 | 105 | 115 | 135 | 155 |
| | $PR^{pf}$ | 1000 | 1950 | 1050 | 1850 | 1900 | 1800 |
| | $\gamma^{pf}$ | 0.04 | 0.05 | 0.03 | 0.03 | 0.04 | 0.04 |
| $O_f$ | | 8000 | 8150 | 8030 | 8045 | 8100 | 8075 |

Table 3.3 Travelling cost from request *i* to request *j*

| ***Cij*** | Request *j* | | | | | |
|---|---|---|---|---|---|---|
| Request | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 0 | 5.58 | 3.16 | 3.96 | 1.19 | 0.82 |
| **2** | 5.58 | 0 | 3.77 | 7.1 | 6.33 | 5.98 |
| **3** | 3.16 | 3.77 | 0 | 3.32 | 4.32 | 3.92 |
| **4** | 3.96 | 7.1 | 3.32 | 0 | 4.95 | 4.7 |
| **5** | 1.19 | 6.33 | 4.32 | 4.95 | 0 | 0.4 |
| **6** | 0.82 | 5.98 | 3.92 | 4.7 | 0.4 | 0 |

Table 3.4 Travelling cost from request *i* to vendor *f*

| $T_{if}$ | Vendors | | | | | |
|---|---|---|---|---|---|---|
| Requests | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 3.71 | 3.33 | 5.51 | 2.98 | 3.17 | 6.81 |
| **2** | 4.68 | 4.9 | 7.36 | 6.89 | 8.63 | 4.56 |
| **3** | 5.14 | 1.16 | 7.72 | 5.83 | 6.24 | 7.96 |
| **4** | 7.43 | 2.21 | 9.48 | 6.87 | 5.72 | 5.07 |
| **5** | 3.45 | 4.52 | 4.68 | 1.92 | 2.3 | 2.82 |
| **6** | 3.36 | 4.16 | 4.83 | 2.18 | 2.65 | 2.55 |

### *3.4.2 Obtained Solution*

The obtained solution is given in Figure 3.2. As can be seen from the Figure 3.2, vendors 4 and 5 serve their customers with one route. Customers 3, 2 and 4 are served by vendor 4. Vendor 5 is followed by customers 5, 1 and 6.

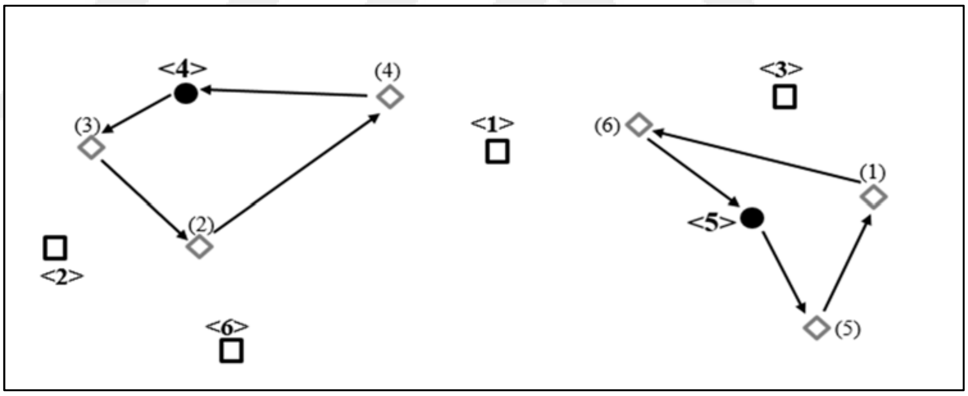


Figure 3.2 Visual illustration of the solution

Demands of vendor 4 are 1080 and 1115 for each product, demands of vendor 5 are 1305 and 1270 for each product type, respectively. Economic production quantities of vendor 4 are 1137 and 1218 for each product type, similarly for vendor 5, the quantities are 1425 and 1291, respectively. Backorder quantities are 82 and 99 for vendor 4, similarly the values are 150 and 116 for vendor 5, for each product type. The total number of tours taken by the vehicles of opened vendors while serving the demand points are 4 trips for each vendor. The result of the objective function is $1.0878E + 5$.

### 3.5 Chapter Conclusions

In this chapter, we aim at developing a MINLP model for the multi-product LIRP with defective rates for the first time. The MINLP provides us to formally formulate the considered LIRP within the context of this study. Moreover, developed MINLP can solve the problem with and without inventory decisions, single product and multi-product. Thus, we can conclude that the developed MINLP formulation is a general model for some of the LIRPs.

Due to the complex nature of the model, it is not able to solve optimal for large-scale problem instances. For that reason, a hybrid meta-heuristic approach is proposed to solve the problem. The details of the proposed algorithm is explained in the next chapter.

# CHAPTER FOUR
# A HYBRID ADAPTIVE LARGE NEIGHBORHOOD SEARCH ALGORITHM FOR THE LIRP

In this chapter, the proposed solution methodology is presented in detail. This section explains the operators of the algorithm, their interactions, the performance evaluation of the proposed solution methodology on benchmark sets, and the computational results.

The proposed ALNS algorithm is explained in the next section. In section 4.2, VNS algorithm that is used for intensification process is explained with its neighborhood structures. Then the performance of the proposed hybrid meta-heuristic is tested on a group of benchmark problems taken from the related literature.

## 4.1 Adaptive Large Neighborhood Search Algorithm

In order to solve the real size problem instances, we propose an ALNS algorithm which has effective algorithmic framework and is capable of tackling vehicle routing problems effectively. The ALNS meta-heuristic was introduced by Ropke & Pisinger (2006). Basically, the ALNS meta-heuristic is an extended version of LNS. Like LNS algorithm, the main idea of ALNS is based on improving an initial solution via the guidance of removal and insertion operators. Unlike LNS, the ALNS algorithm allows multiple removal and insertion operators to be used on the basis of selecting probabilities. These probabilities are adjusted according to the performance of the operators in the previous iterations.

The LIRP is formed by a set of node clusters which represent the locations of customers and potential vendors in the problem. The customers are assigned to the vendors according to their capacities. At each iteration, a node or part of node clusters are randomly revised by way of removal and insertion operators. We define seven operators: three removal and four insertion operators. The removal operators remove a part of the solution while the insertion operators reinsert the removed nodes and extend current clusters. In ALNS algorithm, different heuristics for removal and insertion are used with a learning mechanism guiding the selection of which heuristic to use next. (Derigs et al., 2011) The nodes are selected according to aspects of operators. The operators are selected with a probability according to their score in the past iterations and a new solution is produced. The ALNS algorithm solves the problem of minimizing the system costs of the model by selecting the opened vendors, assigning the requests points and routing vehicles. The algorithm determines whether to replace the current solution with the new one or not. An *acceptance criterion* is used for *accepting* or *rejecting* the new solution.

Figure 4.1 illustrates the main steps of the proposed hybrid ALNS algorithm and the related abbreviations used in this figure are given in Table 4.1. A feasible initial solution $s$ is first constructed. Operators' weights ω and scores ¥ are initialized, and $q$ parameter is chosen as a percentage of the destruction from a predefined interval. The percentage of destruction decreases while the iteration number increases at every iteration. Then, an operator pair which consists of a removal heuristic and an insertion heuristic is probabilistically selected based on their current weight. At each iteration, an infeasible solution is obtained by using the destroy operator on the current solution, this is followed by implementation the repair operator to obtain a new feasible solution $s'$. The new solution $s'$ is then submitted to an acceptance criterion approval. If it has a better objective value it is accepted as a the current solution, otherwise it is applied an acceptance rule to decide whether worse solution will be accepted or not. The solution accepting criterion of the simulated annealing (SA) algorithm is used in this study. The VNS is employed when $K_{non\text{-}improving}$ iterations have been performed without improving. The procedure of VNS will be described in section 4.2. After all, when given a number of iterations is reached, the weights are adjusted. This is repeated until the stopping criteria is met and the best solution $s^*$ is updated.

Table 4.1 Abbreviations related to the proposed hybrid ALNS algorithm

| Notations | Descriptions |
|---|---|
| $s$ | initial solution |
| $s_{best}$ | best solution |
| $\alpha_1, \alpha_2, \alpha_3$ | score factors |
| $q$ | # of removed requests |
| pr | randomization factor |
| т | initial temperature |
| alfa | cooling rate |
| ω | weight |
| ¥ | score |

### *4.1.1 Solution representation*

In the proposed ALNS algorithm, we created a giant tour including vendors and requests. The tour is represented by a vector (Yu, Lin, Lee & Ting, 2010; Ferreira & Queiroz, 2018) that consists of string of numbers of $m$ requests indicated by the set {1, 2, …, $m$}, $s$ potential vendors indicated by the set {$m$+1, $m$+2, …, $m$+$s$} and dummy zeros represents whether a route is terminated (see Figure 4.2). The first number of the string is always beginning with {$m$+1, $m$+2, …, $m$+$s$} denoting the first depot. The solution representation determines which vendor will serve and the customers on its route. Each vendor serves requests between the vendor and the next vendor in the solution representation. If a vendor serves the requests, the route of this vendor starts with the first request after the vendor. Other requests are added until the capacity of vendor reached. Then, a new route for different vendor will be started to serve remaining requests. The solution representation always gives an LIRP solution considering the capacity constraint of the vendor. However, the capacity of the vehicle is ignored in the solution representation. It takes into consideration while calculating objective function value. When a customer is added, if it will exceed the vehicle capacity, the current route is terminated. In same, if there is a dummy zero between two customers, the current route is also terminated.

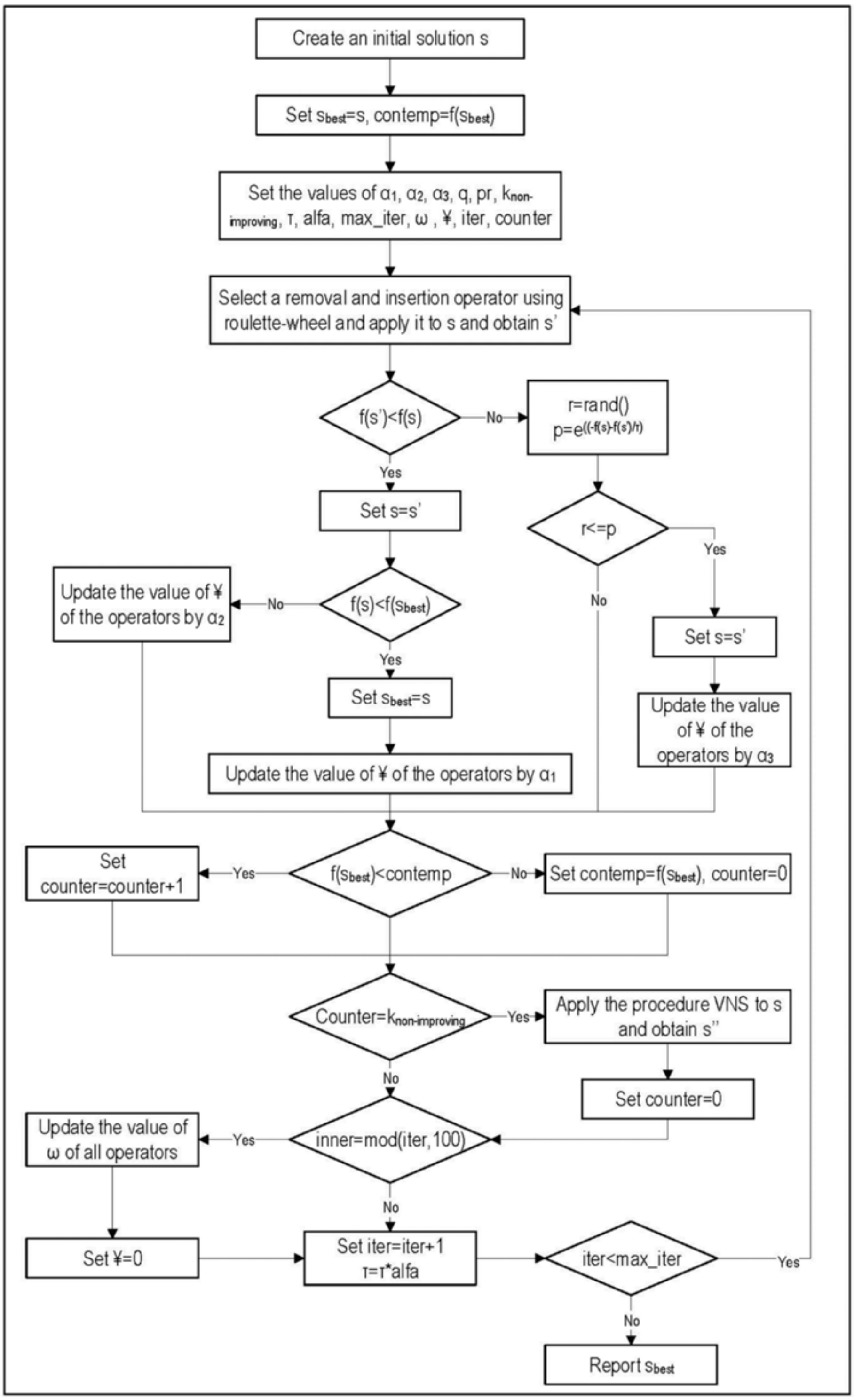


Figure 4.1 Main steps of the proposed hybrid ALNS algorithm

Figure 4.2 represents a solution of an LIRP having 10 requests and five vendors. As can be seen from the Figure 4.2, the first number is vendor 12, followed by a dummy zero and vendor 14. As there are no actual requests between two vendors, vendor 12 doesn't serve any requests. Requests (4, 1, 9, and 6) between vendor 14 and vendor 13 are served by vendor 14. Since assigned requests do not exceed the capacity of vendor, there is one route for this vendor.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 0 | 14 | 4 | 1 | 9 | 6 | 13 | 7 | 5 | 10 | 2 | 0 | 8 | 3 | 15 | 11 |

Figure 4.2 Solution representation for LIRP

Vendor 13 is followed by requests 7, 5, 10 and 2. Request 2 is followed by a dummy zero, therefore the first route is terminated. The second route of vendor 13 then starts to serve requests 8 and 3. Since there are no requests between vendor 15 and vendor 11, addition to after vendor 11, both vendors also do not serve any requests.

The solution representation determines which vendor is served and the requests on each route. In this solution, vendors 14 and 13 serve with one and two routes, respectively.

### *4.1.2 The initialization phase*

The proposed ALNS algorithm starts its search by constructing an initial solution via *K-means Clustering Algorithm* (Tiwari & Solanki, 2013), where "K" refers to the number of clusters defined. The algorithm partitions the sample space into K clusters and assign the requests to the clusters randomly. Then, it determines centroid of clusters. In this study we calculate the centroid of clusters by using center of requests assigned related to cluster. Then, it calculates the distance from the vendors to the centroid of clusters and assigns the requests to the vendors starting from the closest distance considering the capacity constraints. It continues until all requests are assigned. The *K-means Clustering Algorithm* is explained as follows:

1- Cluster the sample space into $k$ groups where $k$ is predefined.

2- Assign the requests to the clusters randomly.

3- Calculate the centroid of clusters using center of requests related cluster.

4- Assign the requests to their closest vendor center.

#### *4.1.3 Removal heuristics*

This section describes three removal heuristics, random, worst and Shaw removals, used within the proposed hybrid ALNS algorithm. All heuristics take an initial solution as input and remove a group of requests from the related solution through the guidance of parameter $q$. The output of each heuristic is a partial infeasible solution as a result of removed $q$ requests. Shaw and worst removals include a randomness factor, specified as $p$, destroying different parts of the solution in every iteration for contributing diversity as distinct from the random removal.

##### *4.1.3.1 Random Removal*

Random Removal is the simplest method among the used removal heuristics. It selects $q$ requests randomly and removes them from the solution repeatedly. It provides diversification by removing any request from the solution without considering solution costs. The random removal heuristic is shown in Figure 4.3.

**Function** Random Removal
    ( $s \in \{solutions\}$, $q \in \mathbb{N}$ )
Initialize removal list ($L \leftarrow \emptyset$)
    set of requests: $D = \{r\}$
  **For** $q$ **do**
      choose a route randomly;
      remove a request from the route
          $L \leftarrow L \cup i$
      remove selected request $i$ from $s$
  **end for**

Figure 4.3 Random removal

#### *4.1.3.2 Worst Removal*

This heuristic is based on removing the $q$ worst requests which cause the huge costs in the solution. In order to determine the worst requests, it calculates the difference of the costs of the solution and the costs of the solution without related request for all requests of a solution. This calculation is executed as follows. For request $r \in R$, the difference is calculated as $c(s) - c(s^{-r})$. $C$ is the cost function value for the current solution $s$ and $s^{-r}$ is the current solution without request $r$. It continues repeatedly until $q$ requests are removed. It is noticeable that the cost differences change after every removal.

Worst removal has a randomization factor which is controlled by the parameter $p$. This randomization presents to remove the same requests again and again. The worst removal heuristic is shown in Figure 4.4.

**Function** Worst Removal
    ($s \in \{solutions\}$, $q \in \mathbb{N}$ , $p \in R_+$)
**while** $q > 0$
    Array: $L$ = All planned requests $i$ sorted by descending cost $(i,s)$
    Choose a random number $y$ in the interval [0,1)
      request: $r = L[y^p |L|]$ ;
      remove $r$ from solution $s$;
       $q = q-1;$
**end while**

Figure 4.4 Worst removal (Ropke & Pisinger, 2006)

#### *4.1.3.3 Shaw Removal*

This removal heuristic is suggested by Shaw (1997, 1998), and also named as ‘related removal’. It tries to remove requests that have a relation between each other. The idea is to remove the similar requests so that giving a chance to be re-inserted them in profitable positions. The heuristic utilizes *relatedness* measure in order to apply the removal method. A relatedness measure $R(i, j)$ is defined between any two visits $i$ and $j$. The lower $R(i, j)$ means that more related are the two requests. One of the relatedness criteria for any requests pair could be geographical distance between them. Removing the requests which are close to each other from the solution enables repositioning them to a better one. The pseudo code for Shaw Removal is shown in Figure 4.5.

```
Function Shaw Removal
    ( s ∈ {solutions}, q ∈ ℕ , p ∈ R+)
  request: r = a randomly selected request from s
  set of requests: D = {r}
   while |D| < q do
     r = a randomly selected request from D;
     Array: L = an array containing all request from s not in D;
      sort L such that
            i < j ⇒ R(r, L[i]) < R(r, L[j]);
  choose a random number y from the interval [0,1);
       D = D ∪{L[y^p |L|]};
    end while
 remove the requests in D from s;
```

Figure 4.5 Shaw removal (Shaw, 1998)

Occurring in the same route could also make them related requests. Removing all requests in a route leads to reduce the number of routes.

The heuristic also includes a random element in order to control the diversification. In the algorithm proposed by Shaw (1997), it is realized with parameter $p$. If $p$=1, relatedness is ignored, and requests are removed randomly. If $p = \infty$, relaxed visits are maximally related to some other relaxed visits (Shaw, 1998). According to the previous studies, $p$ works well with range $3 \leq p \leq 5$ (Shaw, 1997).

### *4.1.4 Insertion heuristics*

Similarly, to the removal heuristics, there are very simple, but also effective insertion heuristics. Basic greedy and regret heuristics were proposed by Pisinger & Ropke (2007). Besides these two well-known insertion heuristics a new insertion heuristic is proposed by modifying the basic greedy insertion within the context of this study.

*4.1.4.1 Basic Greedy Heuristic*

The basic greedy heuristic is a simple repair heuristic. This heuristic tries to identify the impact of each removed request on the cost of the partial infeasible solution by reinserting each request individually into the partial infeasible solution. These impacts on the cost of the partial infeasible solution are calculated as the difference between the total cost of the partial infeasible solution and the total cost of the partial infeasible solution after reinserting each request individually. Afterwards, the requests are ranked with the ascending order as per the calculated cost differences and reinserted according to their ranks. This procedure is repeated until all parts are inserted again. $C_i$, is the "cost" of inserting request *i* at its best position. It could be called minimum cost position. The request *i* that minimizes $c_i$ is chosen and inserted at its best position. This process continues until no requests could be inserted as shown in Figure 4.6.

```
Function Greedy Insertion
      ( s ∈ {solutions}, removal list L)
  For i ∈ L do
    For all routes of s do
       For all feasible positions(j,k) do
          Calculate C_i = d_ji + d_ik - d_jk
       end for
     end for
     insert request i in min[C_i] position
     L ← L − i
  end for
```

Figure 4.6 Greedy insertion (Majidi et al., 2018)

Important part to be considered in this heuristic is no need to recalculate insertion costs in all the routes at each iteration. It is enough to change the route which is inserted into request *i*. This property is used to speed up the insertion heuristics.

*4.1.4.2 Greedy Regret with Noise*

Inserting the request at its best place iteratively may cause getting stuck in local optimum. There is a random removal operator among the removal heuristics which has a significant randomization. Additionally, Shaw and worst removal heuristics also derive a randomization with parameter *p.* An insertion operator with noise is developed by adding a noise term to the objective function. Every insertion process, *C* that is the cost of insertion of a request is calculated. It is also added a *noise* term randomly generated from the interval $[-max\ N, \quad max\ N]$ and the modified insertion cost $C' = max\{0, C + noise\}$ (Ropke & Pisinger, 2006). While calculating the value of *max N,* the function $\eta \times \max_{i,j \in V}\{d_{ij}\}$ is used. The parameter $\eta$ controls amount of the noise. It could be selected as a constant or a random number from the interval [0, 1). *max N* is also dependent on the distance between request *i* and *j.* Implementing *noise* factor and randomization may not guarantee to give better results in the solution. However, they can leave positive effect by increasing the chance of searching new regions in the solution space.

#### *4.1.4.3 Regret-k Heuristics*

This heuristic is like the previous one. Regret-n heuristics take the second cheapest, the third cheapest and $k$-cheapest into account instead of considering only the minimum insertion cost.

In simple, let $r_{ik}$ represents the index within the solution for which request *i* has the $k^{th}$ lowest insertion cost, that is $\Delta c_{i,r_{ik}} \leq \Delta c_{i,r_{ik'}}$ , for $k \leq k'$. The regret value $c_i^*$ can be defined as $c_i^* = \Delta c_{i,r_{i2}} - \Delta c_{i,r_{i1}}$ and each iteration the heuristic selects request *i* that satisfies max $\{c_i^*\}$. Then the request is inserted at its minimum cost position.

The heuristic can be extended by considering the cost of the request *i* in its best, *2nd* best, …, $k^{th}$ best position. *k* is a parameter which is defined before. With this, *regret-k* heuristic chooses to insert a request *i* according to max$\{\sum_{j=2}^{k} \Delta c_{i,r_{ij}} - \Delta c_{i,r_{i1}}\}$. Ties are broken when the lowest cost is selected. Basically, we choose a request for inserting that we will regret most if it is not routed to better position. When a request has been inserted, the insertion positions of the remaining unplaced requests must be recomputed by considering the change caused by inserting this request at a position (Hemmelmayr et al, 2012).

*Regret-2* and *Regret-3* heuristics are implemented as re-inserting procedures in this study. The procedure is shown in Figure 4.7. The advantage of regret heuristics is that they recognize the shortage of suitable insertion possibilities for a request earlier than basic greedy heuristic and can prepone the insertion (Lutz, 2014).

**Function** Regret Insertion
    ( $s \in \{solutions\}$, removal list $L$)
**For** $i \in L$ **do**
  **For** all routes of *s* **do**
    **For** first and second cheapest positions **do**
      Calculate *regret value* :
        $c'_i = \Delta C_{i,r_{i2}} - \Delta C_{i,r_{i1}}$
    **end for**
  **end for**
  sort regret value *c*' descending
  insert request *i* with max[$c'_i$] in min[$C_{i,r_{i1}}$] position
  $L \leftarrow L - i$
**end for**

Figure 4.7 Regret insertion (Ropke & Pisinger, 2006)

#### *4.1.4.4 Modified Greedy Insertion*

In this new heuristic, we propose adding the request that have minimum cost change on the fitness value. In Equation 4.1, we firstly calculate the difference between the feasible solution's fitness value *f(s)* and the solution's fitness value $f(s^{-r})$ without *request r* similar to the worst removal heuristic. This difference provides us the cost effect of a request's current position on the solution's fitness value. After this calculation, we find the best position effect on the fitness value by calculating the difference between the solution's fitness $f(s^r)$ adding *request r* to its best position and fitness $f(s^{-r})$.This difference also shows the cost effect on the fitness values and we expect this effect to be low. By ranking these differences ascendingly, we intend to identify an order of requests to reinsert them into the solution.

$$\min\{\sum_{r=1}^{q}[(f(s) - f(s^{-r})) - (f(s^r) - f(s^{-r}))]\} \tag{4.1}$$

The request that has low cost is firstly inserted to its best position in the infeasible solution. The advantage of this new insertion heuristic is to evaluate the cost effects of the current position and the best position of a removed request simultaneously.

### *4.1.5 Adaptive weight adjustment*

It is implemented an adaptive weight adjustment procedure to record the historical performance of the operators and use this information to bias their selection at the end of each iteration. Selection of the operators is controlled by the roulette wheel mechanism as in the study of Ropke & Pisinger (2006) based on their past successes. At the beginning of each iteration the proposed hybrid ALNS algorithm selects a removal-insertion pair. $\omega_i$ is a weight related to operator pair *i.* In addition, each operator is associated with a score. For *n* operators, the probability of selecting the operator *i* is found as in Equation 4.2.

$$p_i = \omega_i / \sum_{j=1}^{n} \omega_j \qquad i = 1, \ldots . n \tag{4.2}$$

Each *segment*, defined as a set of 100 iterations as in Ropke & Pisinger (2006), the weights are adjusted. At the beginning, all weights are equal to one, so all operators have the same probability and all scores are set to zero. Every time the score of an operator $¥_i$ is increased by factors which are defined as follows:

$\alpha_1$: If the new solution is better than the best global solution.
$\alpha_2$: If the new solution is better than the current solution
$\alpha_3$: If the new solution is worse than the current solution but it is accepted.

The better solution is the solution which has higher score, i.e., $\alpha_1 > \alpha_2 > \alpha_3$. After every *segment*, all scores are reset as zero. The parameter $c_i$ controls the score specifying how many times that the operator *i* is used in one segment. At the end of each segment the weights of operators are determined as in Equation 4.3.

$$\omega_i = \begin{cases} \omega_i & \text{if } c_i = 0 \\ (1-\kappa)\omega_i + \kappa\text{¥}_i / c_i & \text{if } c_i \neq 0 \end{cases} \tag{4.3}$$

Where $\kappa \in [0, 1]$ is a reaction factor which is a controlling parameter how quickly the adjustment algorithm responds variances in the effectivity of the operator. When $\kappa$ is close to 0, update is directed by recent score. Otherwise, when $\kappa$ is close to 1, no significant change in weights.

### *4.1.6 Acceptance and stopping criteria*

Within the context of this study, an acceptance criteria based on the simulated annealing algorithm's one is utilized as proposed by Ropke & Pisinger (2006). The acceptance procedure has two main parameters: the current temperature $\tau>0$ and the cooling rate $0<\varphi<1$. $\tau_{start}$ is the initial temperature and it is decreased with formula $\tau=\tau_{start}*\varphi$ using the cooling rate $\varphi$ throughout the algorithm iterations. $\tau_{start}$ is set according to the initial solution given the current solution $S_{LIRP}$ having an objective value $Z(S_{LIRP})$ and a new neighbor solution $S'_{LIRP}$ is accepted if $Z(S'_{LIRP}) > Z(S_{LIRP})$ and with probability $e^{(Z(S'_{LIRP})-Z(S_{LIRP}))/\tau}$ . The stopping criteria can be controlled with two different methods ending the running: the maximum number of iterations and the final temperature $\tau_{final}$ is reached. In this study, the algorithm stops when the maximum number of ALNS iterations is reached. The simulated annealing procedure (SA) is shown in Figure 4.8.

Input: New solution $X_{new}$; Current Solution $X_{cur}$; Best Solution $X_{best}$; Initial temperature $\tau_{start}$; cooling rate $\varphi$
Adjust temperature $\tau = \tau_{start}$
**If** $Z(X_{new}) < Z(X_{cur})$ **then**
    $X_{cur} = X_{new}$
**Else**
    Let $\nu = e^{-(Z(X_{new})-Z(X_{cur}))/\tau}$
Generate a random number $r \in [0, 1]$
    **If** $r<\nu$
        $X_{cur} = X_{new}$
    **Else**
$\tau = \tau \cdot \varphi$
**End**

Figure 4.8 SA procedure

## 4.2 Improvement Phase

Although the ALNS algorithm is capable of creating high quality solutions for small and medium scale instances within short computational times, it is required to design the hybrid approaches to tackle large scale problems without trapped in local optima and intensify the search space. Besides, it is apparent that hybrid approaches combining the different metaheuristics enables to more powerful and efficient search methods (Bent & Van Hentenryck, 2004; Zachariadis, Tarantilis, & Kiranoudis, 2007). Therefore, to be escaped the local optimum, VNS is employed in this study.

VNS (Mladenovic & Hansen, 1997) is a meta-heuristic search method which uses parameterized neighborhood structures. Within the context of this study, VNS is used for two main purposes; intensifying at the promising solution space for the current or the best solution contained with ALNS and making a considerable change in the current solution using local search. Local search uses five neighborhood structures taken from literature in random order. We implement neighborhood structures when the counter is equal to the $K_{non\text{-}improving}$ parameter. $K_{non\text{-}improving}$ is the maximum number of inner iterations during which the best fitness value is not improved. Every time an improvement occurs during a neighborhood structure is implemented, it is continued to run in same neighborhood structure until the solution is not improved. The main steps of VNS algorithm are given in Figure 4.9.

```
Procedure VNS
Input: s , Neighborhood Structures N_route, s_best
Output: s", s_best
VNS_sol ← s ;
m ← 1;
while m≤max_m
      Initialize the Neighborhood List (NL);
     while NL≠0
       dec=0;
         while dec==0
             s* ← N_route(s );
             if f(s*) < f(VNS_sol)
                   s  ← s* ;
                   VNS_sol ←   s *;
             else
                   dec ← dec+1;
                   remove the N_route from the NL;
             end
            end
         end
   m ← m+1;
  if f(VNS_sol) < f(s_best)
     s_best ← VNS_sol ;
   end
end
```

Figure 4.9 Steps of the proposed VNS algorithm

Two types of intra-route and three types of inter-route structures are employed. After explaining all types of the structures, illustrative neighborhood the structures are given in Figure 4.10. This figure presents an example with 2 depots and 9 customers. The route sequencing of the depot A is given as 065240, while 0831970 for the depot B.

### *4.2.1 Intra-route neighborhood structures*

This type of move involves the deletion of two arcs, and the generation of two new arcs. *2-opt* and *invert* are employed as *intra-route* structures.

*4.2.1.1 2-opt*

In this structure, 2-opt, exchanges adjacent node pairs by way of swapping outmost nodes and reversing the nodes lying between outmost nodes. In Figure 4.10 (a), the non-adjacent nodes (3, 7) are swapped and the nodes between them are reversed.

*4.2.1.2 Invert*

This structure reverses the route direction. It selects a non-adjacent node and flip the arc which consist of non-adjacent nodes and all nodes lying between them. In Figure 4.10 (b), the non-adjacent nodes (5, 4) are selected. Then, they and the nodes between them (4, 2, 5) are flipped.

### ***4.2.2 Inter-route neighborhood structures***

This move type is applied between the route pairs. Every pair of routes is considered. *Swap*, *insertion* and *cross* exchanges are used as *inter-route* structures.

*4.2.2.1 Swap*

The first type of structure exchanges the positions any node pair located in different routes. In Figure 4.10 (c), the positions of nodes 4 and 8 are exchanged.

*4.2.2.2 Insertion*

The insertion move relocates a node from its current route to another route. In Figure 4.10 (d), the node 1 is removed and inserted between nodes 5 and 6.

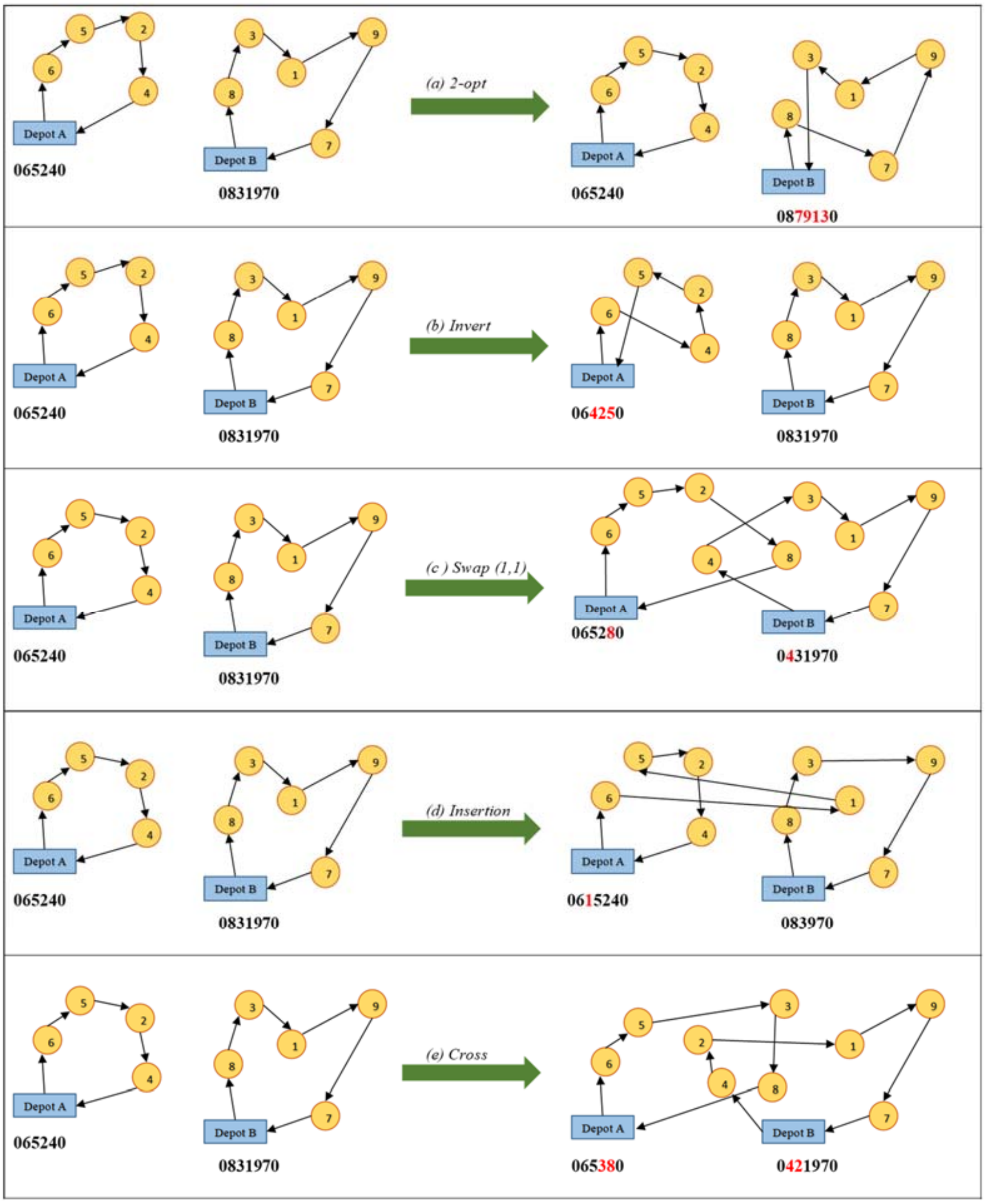


Figure 4.10 Neighborhood structures

*4.2.2.3 Cross*

Cross move proposed by Taillard, Badeau, Gendreau, Guertin & Potvin (1997) consists of exchanging two segments of different routes. In Figure 4.10 (e), the arc between adjacent nodes 5 and 2, 4 and depot from the first rssoute (Depot A), and the one between 8 and depot, 3 and 9 from the second route, are removed. Then, an arc is inserted linking 5 and 3 and another arc is inserted linking 1 and 9. The important thing for the inter-route movement is not to exceed the capacity of the vehicle. In case of exceeding the capacity, the movement is infeasible.

## 4.3 Computational Study

The proposed meta-heuristic algorithm was coded in MATLAB 2016 and run on Intel core i5, 2.5 GHz processor and 8 GB RAM. In order to verify the algorithm performance, three classical LRP problem sets are selected as test instances.

### *4.3.1 Test instances*

The performance of the proposed solution method is tested on classical LRP benchmark instances taken from literature, since there is not any benchmark set related to multi-product LIRP. Therefore, we have considered three sets of instances. The first problem set consists of 13 LRP instances is introduced by Barreto (2004), involving from 21 to 150 customers. The second problem set belongs to Prins, Prodhon & Wolfler-Calvo (2004). This set is denoted as set "Prodhon" in the literature. 24 instances are generated from 20 to 100 customers. The cost of routes is calculated as Euclidean distances, multiplying 100 and rounding up to the next integer. The third set that is denoted as "Tuzun" in literature is proposed by Tuzun & Burke (1999). It also contains 24 instances with the uncapacitated 10 or 20 depots, capacitated vehicles and 100 and 150 customers. The sets Prodhon and Tuzun have the cost using a vehicle. Usage cost is fixed at 1000 for Prodhon, and 10 for Tuzun.

The proposed hybrid meta-heuristic is compared against the best solutions of six methods given as greedy randomized adaptive search procedure (GRASP) (Prins, Prodhon & Wolfler-Calvo, 2006b), memetic algorithm with population management (MA|PM) (Prins, Prodhon & Wolfler-Calvo, 2006a), Lagrangean relaxation-granular tabu search (LRGTS) (Prins, Prodhon, Ruiz, Soriano &Wolfler-Calvo, 2007), genetic algorithm hybridized with a local search procedure (GAHLS) (Duhamel, Lacomme, Prins & Prodhon, 2008) and Heuristics H1-H2 (Ferreira & Queiroz, 2018) and ALNS (Hemmelmayr et. al, 2012) for each benchmark instance.

The computational experiments consider the average GAP and computing time (CPU) for comparison. The average GAP between the best solution found by the proposed algorithm and the best known solution in literature computed via Equation 4.4 as below

$$GAP = \frac{(best - BKS)}{BKS} \times 100\% \tag{4.4}$$

where the *best* is best solution of proposed algorithm and the BKS is the best-known solution (BKS) reported in the literature. The operating system, the computer program and the compiler affect the solution speed. Therefore, a comparison with CPU time will be unfair. In addition, the complexity of the proposed algorithm can affect on the speed.

The characteristics of the each instance are summarized in Table 4.2. The table lists the characteristics respect to *s*-the number of vendors, *m*-the number of requests, and $K_l$-the capacity of the vehicles, respectively.

#### *4.3.2 Parameter tuning*

This section describes the parameters used for obtaining the result and determines to reveal which part of the heuristics contributes most to the solution quality. All values are based on preliminary tests. Firstly, considering the removal parameters, Shaw and worst removal have a randomization parameter $p$ which is chosen a random number range from 3 to 5 (Ropke & Pisinger, 2006). The noise rate $\eta$ as 0.025 and number of requests to be removed $q$ is chosen percentage of destruction in the given interval. It is set to [1 min{60, 0.4*$m$}] (*m: number of customers*) as suggested in Pisinger & Ropke (2007).

For the acceptance criteria, the starting temperature and the cooling rate $\varphi$ is set to 1000 and 0.99975, respectively. Preliminary tests show that 50,000 iterations usually provide a good solution quality. The best solution of five run is reported to make a fair comparison. In order to evaluate the performance of the operators, *segment* length is set to 100 iterations and the reaction factor $\kappa$ is set to 0.1. The scores are updated with $\alpha_1$=15, $\alpha_2$=10 and $\alpha_3$=3.

Finally, the number of iterations without improvement $K_{non\text{-}improving}$ is set to 200 and maximum iterations (*max_m*) for applying VNS is set to 5000.

Table 4.2 Characteristics of the problem instances

| Set | Instances | $S$ | $m$ | $K_1$ |
|---|---|---|---|---|
| Prodhon | 4 | 5 | 20 | $K_1 \in \{70,150\}$ |
| Prodhon | 8 | 5 | 50 | $K_1 \in \{70,150\}$ |
| Prodhon | 6 | 5 | 100 | $K_1 \in \{70,150\}$ |
| Prodhon | 6 | 10 | 100 | $K_1 \in \{70,150\}$ |
| | | | | |
| Tuzun | 6 | 10 | 100 | 150 |
| Tuzun | 6 | 20 | 100 | 150 |
| Tuzun | 6 | 10 | 150 | 150 |
| Tuzun | 6 | 20 | 150 | 150 |
| | | | | |
| Barreto | 13 | $s \in \{5,8,10,14\}$ | $m \in [21,150]$ | $K_1 \in [160,9000000]$ |

### *4.3.3 Computational results*

The proposed hybrid ALNS-VNS algorithm is compared to the different methods from literature as given in section 4.3.1. Table 4.3 and Table 4.4 show the comparison of the proposed hybrid ALNS-VNS algorithm considering the six methods from the literature for the instances generated by Prodhon (Prins et. al, 2004). Table 4.3 provides the value of the BKS for each instance and the obtained results of the other methods. The proposed hybrid ALNS-VNS with the name of the instances in each row. The solutions that are equal to BKS or better than BKS are depicted by the boldmark.

Table 4.3 Results for the instances of Prodhon (Prins et al., 2004)

| Prob ID (mxs) | BKS | GRASP | MA\|PM | LRGTS | GAHLS | Heuristics H1 | & H2 | ALNS | ALNS-VNS |
|---|---|---|---|---|---|---|---|---|---|
| 20-5-1 (20x5) | 54,793 | 55,021 | **54,793** | 55,131 | **54,793** | **54,793** | **54,793** | **54,793** | **54,793** |
| 20-5-1b (20x5) | 39,104 | **39,104** | **39,104** | **39,104** | **39,104** | **39,104** | **39,104** | **39,104** | **39,104** |
| 20-5-2 (20x5) | 48,908 | **48,908** | **48,908** | **48,908** | **48,908** | **48,908** | **48,908** | **48,908** | **48,908** |
| 20-5-2b (20x5) | 37,542 | **37,542** | **37,542** | **37,542** | **37,542** | **37,542** | **37,542** | **37,542** | **37,542** |
| 50-5-1 (50x5) | 90,111 | 90,632 | 90,160 | 90,160 | **90,111** | **90,111** | **90,111** | **90,111** | **90,111** |
| 50-5-1b (50x5) | 63,242 | 64,761 | **63,242** | 63,256 | 63,469 | **63,242** | **63,242** | **63,242** | **63,242** |
| 50-5-2 (50x5) | 88,298 | 88,786 | **88,298** | 88,715 | 88,709 | **88,298** | **88,298** | 88,443 | **88,298** |
| 50-5-2b (50x5) | 67,308 | 68,042 | 67,893 | 67,698 | 67,353 | **67,308** | **67,308** | 67,340 | **67,308** |
| 50-5-2BIS (50x5) | 84,055 | **84,055** | **84,055** | 84,181 | 84,409 | **84,055** | **84,055** | **84,055** | **84,055** |
| 50-5-2bBIS (50x5) | 51,822 | 52,059 | **51,822** | 51,992 | 51,902 | **51,822** | **51,822** | **51,822** | **51,822** |
| 50-5-3 (50x5) | 86,203 | 87,380 | **86,203** | **86,203** | **86,203** | **86,203** | **86,203** | **86,203** | **86,203** |
| 50-5-3b (50x5) | 61,830 | 61,890 | **61,830** | **61,830** | 62,763 | **61,830** | **61,830** | **61,830** | **61,830** |
| 100-5-1 (100x5) | 274,814 | 279,437 | 281,944 | 277,935 | 281,564 | 275,281 | 275,079 | 275,636 | 275,391 |
| 100-5-1b (100x5) | 213,615 | 216,159 | 216,656 | 214,885 | 219,056 | 214,947 | 214,758 | 214,735 | 214,221 |
| 100-5-2 (100x5) | 193,671 | 199,520 | 195,568 | 196,545 | 197,156 | **193,671** | **193,671** | 193,752 | **193,671** |
| 100-5-2b (100x5) | 157,095 | 159,550 | 157,325 | 157,792 | 159,615 | 157,129 | 157,150 | **157,095** | 157,144 |
| 100-5-3 (100x5) | 200,079 | 204,906 | 203,999 | 201,749 | 201,952 | 200,217 | 200,455 | 200,305 | 200,346 |
| 100-5-3b (100x5) | 152,441 | 154,596 | 153,322 | 154,709 | 154,404 | **152,441** | **152,441** | **152,441** | **152,441** |
| 100-10-1 (100x10) | 287,695 | 323,171 | 316,575 | 291,887 | 325,357 | 288,194 | 287,925 | 296,877 | 290,352 |
| 100-10-1b (100x10) | 230,989 | 271,477 | 270,251 | 235,532 | 274,379 | 232,898 | 232,716 | 235,849 | 232,324 |
| 100-10-2 (100x10) | 243,590 | 254,087 | 245,123 | 246,708 | 248,331 | 244,294 | 244,545 | 244,740 | **243,590** |
| 100-10-2b (100x10) | 203,988 | 206,555 | 205,052 | 204,435 | 208,508 | 204,390 | **203,988** | 204,016 | **203,988** |
| 100-10-3 (100x10) | 250,882 | 270,826 | 253,669 | 258,656 | 264,547 | 251,325 | 251,567 | 253,801 | 253,070 |
| 100-10-3b (100x10) | 203,687 | 216,173 | 204,815 | 205,883 | 211,925 | 204,587 | **203,687** | 205,609 | 204,631 |

Observing the results in Table 4.3, proposed hybrid ALNS-VNS is able to reach the BKS for all the instances with 20 and 50 customers. Besides, ALNS-VNS found the BKS for the instance coord100-5-2 (100x5), the instance coord100-5-3b (100x5), the instance coord100-10-2 (100x10), and the instance coord100-10-2b (100 x 10).

Table 4.4 GAPs for the instances of Prodhon (Prins et al., 2004)

| **Prob ID** | **GRASP** | | **MA\|PM** | | **LRGTS** | | **GAHLS** | | **Heuristics H1** | | **& H2** | | **ALNS** | | **ALNS-VNS** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (mxs) | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU |
| 20-5-1 (20x5) | 0.42 | 0.2 | **0.00** | 0.3 | 0.62 | 0.4 | **0.00** | 0 | **0.00** | 6 | **0.00** | 4 | **0.00** | 39 | **0.00** | 0 |
| 20-5-1b (20x5) | **0.00** | 0.2 | **0.00** | 0.3 | **0.00** | 0.2 | **0.00** | 0 | **0.00** | 6 | **0.00** | 4 | **0.00** | 54 | **0.00** | 0 |
| 20-5-2 (20x5) | **0.00** | 0.1 | **0.00** | 0.4 | **0.00** | 0.5 | **0.00** | 0 | **0.00** | 6 | **0.00** | 4 | **0.00** | 38 | **0.00** | 0 |
| 20-5-2b (20x5) | **0.00** | 0.2 | **0.00** | 0.3 | **0.00** | 0.1 | **0.00** | 0 | **0.00** | 6 | **0.00** | 4 | **0.00** | 67 | **0.00** | 0 |
| Avg. | 0.10 | | **0.00** | | 0.15 | | **0.00** | | **0.00** | | **0.00** | | **0.00** | | **0.00** | |
| 50-5-1 (50x5) | 0.58 | 1.8 | 0.05 | 2.6 | 0.05 | 0.3 | **0.00** | 6 | **0.00** | 54 | **0.00** | 23 | **0.00** | 101 | **0.00** | 55 |
| 50-5-1b (50x5) | 2.40 | 1.8 | **0.00** | 3.2 | 0.02 | 1 | 0.36 | 58 | **0.00** | 34 | **0.00** | 15 | **0.00** | 65 | **0.00** | 34 |
| 50-5-2 (50x5) | 0.55 | 2.4 | **0.00** | 3.4 | 0.47 | 1.8 | 0.47 | 35 | **0.00** | 61 | **0.00** | 66 | 0.16 | 99 | **0.00** | 76 |
| 50-5-2b (50x5) | 1.09 | 2.5 | 0.87 | 2.9 | 0.58 | 1.8 | 0.07 | 65 | **0.00** | 258 | **0.00** | 91 | 0.05 | 200 | **0.00** | 59 |
| 50-5-2BIS (50x5) | **0.00** | 1.7 | **0.00** | 3.2 | 0.15 | 2 | 0.42 | 28 | **0.00** | 63 | **0.00** | 19 | **0.00** | 107 | **0.00** | 38 |
| 50-5-2bBIS (50x5) | 0.46 | 2.6 | **0.00** | 4.2 | 0.33 | 0.9 | 0.15 | 27 | **0.00** | 142 | **0.00** | 36 | **0.00** | 98 | **0.00** | 36 |
| 50-5-3 (50x5) | 1.37 | 2.3 | **0.00** | 3.1 | **0.00** | 0.3 | **0.00** | 39 | **0.00** | 178 | **0.00** | 47 | **0.00** | 101 | **0.00** | 60 |
| 50-5-3b (50x5) | 0.10 | 2 | **0.00** | 4.9 | **0.00** | 0.5 | 1.51 | 17 | **0.00** | 17 | **0.00** | 36 | **0.00** | 137 | **0.00** | 86 |
| Avg. | 0.82 | | 0.12 | | 0.20 | | 0.37 | | **0.00** | | **0.00** | | 0.03 | | **0.00** | |

Table 4.4 continues

| | Heuristics | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Prob ID** | **GRASP** | | **MA\|PM** | | **LRGTS** | | **GAHLS** | | **H1** | **&** | **H2** | | **ALNS** | | **ALNS-VNS** | |
| (mxs) | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU |
| 100-5-1 (100x5) | 1.68 | 27.6 | 2.59 | 26.3 | 1.14 | 8.7 | 2.46 | 220 | 0.17 | 1198 | 0.10 | 371 | 0.30 | 520 | 0.21 | 642 |
| 100-5-1b (100x5) | 1.19 | 23.2 | 1.42 | 34.5 | 0.59 | 8.3 | 2.55 | 226 | 0.62 | 801 | 0.54 | 325 | 0.52 | 1190 | 0.28 | 755 |
| 100-5-2 (100x5) | 3.02 | 17.4 | 0.98 | 35.8 | 1.48 | 2.3 | 1.80 | 126 | **0.00** | 1246 | **0.00** | 55 | 0.04 | 463 | **0.00** | 663 |
| 100-5-2b (100x5) | 1.56 | 22.4 | 0.15 | 36.4 | 0.44 | 3.3 | 1.60 | 342 | 0.02 | 742 | 0.04 | 237 | **0.00** | 859 | 0.03 | 245 |
| 100-5-3 (100x5) | 2.41 | 21.6 | 1.96 | 28.7 | 0.83 | 2.4 | 0.94 | 188 | 0.07 | 718 | 0.19 | 290 | 0.11 | 454 | 0.15 | 612 |
| 100-5-3b (100x5) | 1.41 | 20.3 | 0.58 | 33.3 | 1.49 | 2.9 | 1.29 | 291 | **0.00** | 173 | **0.00** | 191 | **0.00** | 684 | **0.00** | 791 |
| Avg. | 1.88 | | 1.28 | | 1.00 | | 1.77 | | 0.15 | | 0.14 | | 0.16 | | 0.11 | |
| 100-10-1 (100x10) | 12.33 | 37.4 | 10.04 | 24.7 | 1.46 | 14.1 | 13.09 | 401 | 0.17 | 1484 | 0.08 | 356 | 3.19 | 210 | 0.92 | 423 |
| 100-10-1b (100x10) | 17.53 | 29.5 | 17.00 | 36 | 1.97 | 14 | 18.78 | 655 | 0.83 | 749 | 0.75 | 83 | 2.10 | 188 | 0.58 | 375 |
| 100-10-2 (100x10) | 4.31 | 39.1 | 0.63 | 24.6 | 1.28 | 14.4 | 1.95 | 306 | 0.29 | 1231 | 0.39 | 106 | 0.47 | 136 | **0.00** | 768 |
| 100-10-2b (100x10) | 1.26 | 29.8 | 0.52 | 31.6 | 0.22 | 10.1 | 2.22 | 801 | 0.20 | 547 | **0.00** | 285 | 0.01 | 261 | **0.00** | 546 |
| 100-10-3 (100x10) | 7.95 | 35.4 | 1.11 | 29 | 3.10 | 13.3 | 5.45 | 176 | 0.18 | 211 | 0.27 | 415 | 1.16 | 202 | 0.87 | 345 |
| 100-10-3b (100x10) | 6.13 | 39.8 | 0.55 | 36.5 | 1.08 | 10.8 | 4.04 | 359 | 0.44 | 1220 | **0.00** | 359 | 0.94 | 224 | 0.46 | 457 |
| Avg. | 8.25 | | 4.98 | | 1.52 | | 7.59 | | 0.35 | | 0.25 | | 1.31 | | 0.47 | |
| Global Avg. | 2.82 | | 1.60 | | 0.72 | | 2.46 | | 0.12 | | 0.10 | | 0.38 | | 0.15 | |

Each row of Table 4.4 has the instance name, with the number of customers and vendors ($m$ x $s$); the GAP and CPU of each method. It can be observed that computing time raised considerably for the large instances. The last row gives the average GAP of each method. Concerning the GAP reported in Table 4.4, Heuristic H2 has the best average GAP, which is 0.10%, followed by H1 with the GAP of 0.12%, behind proposed ALNS-VNS with the GAP of 0.15%, ALNS with the GAP of 0.38% ,while the GAP in LRGTS is 0.72%, MA|PM with 1.60%, GAHLS with 2.46% and GRASP with 2.82%.

Table 4.5 Results for the instances of Tuzun & Burke (Tuzun & Burke ,1999)

| **Prob ID (mxs)** | **BKS** | **GRASP** | **MA\|PM** | **LRGTS** | **GAHLS** | **Heuristics H1** | **& H2** | **ALNS** | **ALNS-VNS** |
|---|---|---|---|---|---|---|---|---|---|
| 111112(100x10) | 1467.68 | 1525.25 | 1493.92 | 1490.82 | 1487.35 | **1467.68** | 1469.43 | **1467.68** | **1467.68** |
| 111122(100x20) | 1449.2 | 1526.9 | 1471.36 | 1471.76 | 1483.48 | **1449.2** | **1449.2** | 1452.14 | **1448.37** |
| 111212(100x10) | 1394.8 | 1423.54 | 1418.83 | 1412.04 | 1444.7 | **1394.8** | **1394.8** | 1394.93 | **1394.8** |
| 111222(100x20) | 1432.29 | 1482.29 | 1492.46 | 1443.06 | 1466.92 | **1432.29** | **1432.29** | 1433.42 | **1432.29** |
| 112112(100x10) | 1167.16 | 1200.24 | 1173.22 | 1187.63 | 1185.45 | **1167.16** | 1167.53 | 1167.53 | **1167.16** |
| 112122(100x20) | 1102.24 | 1123.64 | 1115.37 | 1115.95 | 1115.49 | 1102.62 | **1102.24** | **1102.24** | 1102.62 |
| 112212(100x10) | 791.66 | 814 | 793.97 | 813.28 | 807.85 | **791.66** | **791.66** | **791.66** | **791.66** |
| 112222(100x20) | 728.3 | 747.84 | 730.51 | 742.96 | 737.19 | **728.3** | **728.3** | **728.3** | **728.29** |
| 113112(100x10) | 1238.24 | 1273.1 | 1262.32 | 1267.93 | 1251.01 | 1238.49 | 1239.22 | 1238.7 | 1238.49 |
| 113122(100x20) | 1245.31 | 1272.94 | 1251.32 | 1256.12 | 1260.1 | 1247.27 | 1247.68 | 1246.52 | 1245.9 |
| 113212(100x10) | 902.26 | 912.19 | 903.82 | 913.06 | 909.98 | **902.26** | **902.26** | **902.26** | **902.26** |
| 113222(100x20) | 1018.29 | 1022.51 | 1022.93 | 1025.51 | 1036.86 | **1018.29** | **1018.29** | **1018.29** | **1018.29** |
| 131112(150x10) | 1866.75 | 2006.7 | 1959.39 | 1946.01 | - | 1905.2 | 1897.26 | 1922.7 | 1901.18 |
| 131122(150x20) | 1823.2 | 1888.9 | 1881.67 | 1875.79 | - | 1854.45 | 1840.3 | 1847.93 | **1822.95** |
| 131212(150x10) | 1964.3 | 2033.93 | 1984.25 | 2010.53 | - | 1964.75 | 1967.64 | 1975.83 | 1969.4 |
| 131222(150x20) | 1792.77 | 1856.07 | 1855.25 | 1819.89 | - | **1792.77** | 1801.4 | 1806.31 | 1800.43 |
| 132112(150x10) | 1443.33 | 1508.33 | 1448.27 | 1448.65 | - | 1444.82 | 1444.1 | 1447.43 | 1452.39 |
| 132122(150x20) | 1431.79 | 1456.82 | 1459.83 | 1492.86 | - | **1431.79** | 1432.64 | 1445.32 | 1441.59 |
| 132212(150x10) | 1204.42 | 1240.4 | 1207.41 | 1211.07 | - | 1204.48 | **1204.42** | 1204.98 | 1205.15 |
| 132222(150x20) | 925.07 | 940.8 | 934.79 | 936.93 | - | 925.33 | **925.07** | 931.49 | 931.54 |
| 133112(150x10) | 1694.18 | 1736.9 | 1720.3 | 1729.31 | - | 1696.27 | 1697.27 | 1694.64 | 1700.36 |
| 133122(150x20) | 1392 | 1425.74 | 1429.34 | 1424.59 | - | 1392.18 | 1394.02 | 1400.5 | 1402.23 |
| 133212(150x10) | 1198.19 | 1223.7 | 1203.44 | 1216.32 | - | **1198.19** | 1198.64 | 1198.67 | 1198.31 |
| 133222(150x20) | 1151.8 | 1231.33 | 1158.54 | 1162.16 | - | 1152.37 | **1151.8** | 1152.01 | 1156.88 |

Table 4.6 GAPs for the instances of Tuzun &Burke (Tuzun & Burke ,1999)

| **Prob ID** | **GRASP** | | **MA\|PM** | | **LRGTS** | | **GAHLS** | | **Heuristics H1 & H2** | | | | **ALNS** | | **ALNS-VNS** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | **H1** | | **H2** | | | | | |
| | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU |
| 111112 (100x10) | 3.92 | 32.4 | 1.79 | 31.5 | 1.58 | 3.3 | 1.34 | 628 | **0.00** | 495 | 0.12 | 99 | **0.00** | 153 | **0.00** | 546 |
| 111122 (100x20) | 5.36 | 40.7 | 1.53 | 35.6 | 1.56 | 6.5 | 2.37 | 922 | **0.00** | 1118 | **0.00** | 344 | 0.20 | 148 | **0.06** | 451 |
| 111212 (100x10) | 2.06 | 27.6 | 1.72 | 36.2 | 1.24 | 4.2 | 3.58 | 507 | **0.00** | 306 | **0.00** | 307 | 0.01 | 110 | **0.00** | 312 |
| 111222 (100x20) | 3.49 | 36.2 | 4.20 | 36.4 | 0.75 | 7.4 | 2.42 | 1194 | **0.00** | 1350 | **0.00** | 337 | 0.08 | 144 | **0.00** | 297 |
| 112112 (100x10) | 2.83 | 27.7 | 0.52 | 31.9 | 1.75 | 6.9 | 1.57 | 333 | **0.00** | 267 | 0.03 | 309 | 0.03 | 274 | **0.00** | 234 |
| 112122 (100x20) | 1.94 | 33.4 | 1.19 | 42.7 | 1.24 | 6.8 | 1.20 | 1381 | 0.03 | 1329 | **0.00** | 370 | **0.00** | 178 | 0.03 | 764 |
| 112212 (100x10) | 2.82 | 22.5 | 0.29 | 38 | 2.73 | 5.2 | 2.05 | 557 | **0.00** | 1187 | **0.00** | 333 | **0.00** | 215 | **0.00** | 342 |
| 112222 (100x20) | 2.68 | 37.3 | 0.30 | 49.3 | 2.01 | 5.9 | 1.22 | 959 | **0.00** | 1328 | **0.00** | 123 | **0.00** | 96 | **0.00** | 261 |
| 113112 (100x10) | 2.82 | 21.5 | 1.94 | 36.8 | 2.40 | 4.3 | 1.03 | 877 | 0.02 | 284 | 0.08 | 107 | 0.04 | 157 | 0.02 | 465 |
| 113122 (100x20) | 2.22 | 36 | 0.48 | 47.7 | 0.87 | 6.3 | 1.19 | 1142 | 0.16 | 372 | 0.19 | 124 | 0.10 | 237 | 0.05 | 597 |
| 113212 (100x10) | 1.10 | 20.5 | 0.17 | 35.1 | 1.20 | 4 | 0.86 | 465 | **0.00** | 331 | **0.00** | 107 | **0.00** | 59 | **0.00** | 176 |
| 113222 (100x20) | 0.41 | 38.4 | 0.46 | 62.6 | 0.71 | 4.9 | 1.82 | 1009 | **0.00** | 1354 | **0.00** | 385 | **0.00** | 150 | **0.00** | 247 |
| Avg. | 2.64 | | 1.22 | | 1.50 | | 1.72 | | 0.02 | | 0.04 | | 0.04 | | **0.00** | |

Table 4.6 continues

| Prob ID | Heuristics | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **GRASP** | | **MA\|PM** | | **LRGTS** | | **GAHLS** | | **H1 & H2** | | | | **ALNS** | | **ALNS-VNS** | |
| | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU |
| 131112 (150x10) | 7.50 | 113 | 4.96 | 128.5 | 4.25 | 12.5 | - | - | 2.06 | 2394 | 1.63 | 214 | 3.00 | 360 | 1.84 | 462 |
| 131122 (150x20) | 3.60 | 161.4 | 3.21 | 144.2 | 2.88 | 18.5 | - | - | 1.71 | 265 | 0.94 | 268 | 1.36 | 146 | **0.01** | 316 |
| 131212 (150x10) | 3.54 | 100 | 1.02 | 111 | 2.35 | 11.1 | - | - | 0.02 | 2415 | 0.17 | 688 | 0.59 | 331 | 0.26 | 751 |
| 131222 (150x20) | 3.53 | 132.4 | 3.49 | 144.1 | 1.51 | 15.8 | - | - | **0.00** | 778 | 0.48 | 773 | 0.76 | 155 | 0.43 | 843 |
| 132112 (150x10) | 4.50 | 117.7 | 0.34 | 166.6 | 0.37 | 22 | - | - | 0.10 | 2375 | 0.05 | 679 | 0.28 | 628 | 0.63 | 912 |
| 132122 (150x20) | 1.75 | 166.1 | 1.96 | 154.8 | 4.27 | 28 | - | - | **0.00** | 2586 | 0.06 | 723 | 0.94 | 352 | 0.68 | 754 |
| 132212 (150x10) | 2.99 | 106.7 | 0.25 | 161.4 | 0.55 | 14.6 | - | - | **0.00** | 690 | **0.00** | 681 | 0.05 | 780 | 0.06 | 822 |
| 132222 (150x20) | 1.70 | 142.4 | 1.05 | 196.1 | 1.28 | 13.7 | - | - | 0.03 | 450 | **0.00** | 761 | 0.69 | 431 | 0.70 | 945 |
| 133112 (150x10) | 2.52 | 92.8 | 1.54 | 143.8 | 2.07 | 17.9 | - | - | 0.12 | 2403 | 0.18 | 662 | 0.03 | 612 | 0.36 | 931 |
| 133122 (150x20) | 2.42 | 128.4 | 2.68 | 155.7 | 2.34 | 18.5 | - | - | 0.01 | 2532 | 0.15 | 726 | 0.61 | 457 | 0.73 | 854 |
| 133212 (150x10) | 2.13 | 88.5 | 0.44 | 153.8 | 1.51 | 14.5 | - | - | **0.00** | 2391 | 0.04 | 216 | 0.04 | 624 | 0.01 | 860 |
| 133222 (150x20) | 6.90 | 134.9 | 0.59 | 223 | 0.90 | 14.3 | - | - | 0.05 | 2613 | **0.00** | 762 | 0.02 | 450 | 0.44 | 647 |
| Avg. | 3.59 | | 1.79 | | 2.02 | | | | 0.34 | | 0.31 | | 0.70 | | 0.51 | |
| Global Avg. | 3.11 | | 1.50 | | 1.76 | | 1.72 | | 0.18 | | 0.17 | | 0.37 | | 0.26 | |

Table 4.5 presents the results for the instances of Tuzun & Burke (1999). ALNS-VNS reached the BKS for 8 out of 24 instances, besides improving the BKS for instance 111122( 100 x 20) and instances 131122 (150 x 20).

The results in Table 4.6 shows that the average GAP of Heuristic H2 is the best one with 0.17%, followed by H1 with the GAP of 0.18%, while the proposed hybrid ALNS-VNS has an average GAP of 0.26%, ALNS of 0.37%, MA|PM of 1.50%, and GAHLS of 1.72%, LRGTS of 1.76% and GRASP of 3.11%, respectively. In addition, ALNS-VNS reduce the BKS of the instance 111122 (100 x 20) in 0.06% and the instance 131122 (150 x 20) in 0.01%.

Table 4.7 Results for the instances of Barreto (Barreto, 2004)

| Prob ID (mxs) | BKS | GRASP | MA\|PM | LRGTS | GAHLS | Heuristics H1 | & H2 | ALNS | ALNS-VNS |
|---|---|---|---|---|---|---|---|---|---|
| Gaspelle (21x5) | 424.9 | **424.9** | **424.9** | **424.9** | **424.9** | **424.9** | **424.9** | **424.9** | **424.9** |
| Gaspelle2 (22x5) | 585.1 | **585.1** | 611.8 | 587.4 | **585.1** | **585.1** | **585.1** | **585.11** | **585.1** |
| Gaspelle3 (29x5) | 512.1 | 515.1 | **512.1** | **512.1** | **512.1** | **512.1** | **512.1** | **512.1** | **512.1** |
| Gaspelle4 (32x5) | 562.2 | 571.9 | 571.9 | 587.4 | **562.2** | **562.2** | **562.2** | **562.22** | **562.2** |
| Gaspelle5 (32x5) | 504.3 | **504.3** | 534.7 | 504.8 | **504.3** | **504.3** | **504.3** | **504.33** | **504.3** |
| Gaspelle6 (36x5) | 460.4 | **460.4** | 485.4 | 476.5 | **460.4** | **460.4** | **460.4** | **460.37** | **460.37** |
| Christ50 (50x5) | 565.6 | 599.1 | **565.6** | 586.4 | 584.8 | **565.6** | **565.6** | **565.6** | **565.6** |
| Christ75 (75x10) | 844.4 | 861.6 | 866.1 | 863.5 | 851.8 | 848.85 | 848.85 | 853.47 | 848.85 |
| Christ100(100x10) | 833.4 | 861.6 | 850.1 | 842.9 | 842.4 | **833.4** | **833.4** | **833.43** | 836.043 |
| Min27 (27x5) | 3062 | **3062** | **3062** | 3065.2 | **3062** | **3062** | **3062** | **3062** | **3062** |
| Min134 (134x8) | 5709 | 5965.1 | 5950.1 | 5809 | - | **5709** | **5709** | 5712.99 | **5709** |
| Das88 (88x8) | 355.8 | 356.9 | **355.8** | 368.7 | 355.9 | **355.8** | **355.8** | **355.8** | **355.79** |
| Das150 (150x10) | 43,919.9 | 44,625.2 | 44,011.7 | 44,386.3 | | 44,473.1 | 44,309.6 | 44,309.2 | **43,919.9** |

Table 4.7 shows the results for the instances of Barreto (2004). The proposed ALNS-VNS obtained the BKS for 11 out of 13 instances. For the instance, Christ75 any of the could reach the GAP equal to zero. In Table 4.8 the comparison related to the GAP is shown, between hybrid ALNS and the other methods from literature. According to the table, the average GAP of proposed ALNS-VNS is the best one with the GAP of 0.06%, while H2 is ranked second, with the GAP of 0.11%, H1 is ranked

third with the GAP of 0.14%, behind ALNS, which has the GAP of 0.16%, GAHLS, with 0.49%, GRASP with 1.54%, LRGTS with 1.70% and MA|PM with 2.06%.

Table 4.8 GAPs for the instances of Barreto (Barreto, 2004)

| **Prob ID** | **GRASP** | | **MA\|PM** | | **LRGTS** | | **GAHLS** | | **Heuristics H1** | | **& H2** | | **ALNS** | | **ALNS-VNS** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (mxs) | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU | GAP | CPU |
| Gaspelle (21x5) | **0.00** | 0.20 | **0.00** | 0.30 | **0.00** | 0.20 | **0.00** | 0 | **0.00** | 22 | **0.00** | 8 | **0.00** | 0 | **0.00** | 2 |
| Gaspelle2 (22x5) | **0.00** | 0.20 | 4.56 | 0.30 | 0.39 | 0.20 | **0.00** | 0 | **0.00** | 15 | **0.00** | 5 | **0.00** | 0 | **0.00** | 6 |
| Gaspelle3 (29x5) | 0.59 | 0.40 | **0.00** | 0.80 | **0.00** | 0.40 | **0.00** | 1 | **0.00** | 68 | **0.00** | 27 | **0.00** | 0 | **0.00** | 11 |
| Gaspelle4 (32x5) | 1.73 | 0.60 | 1.73 | 0.80 | 4.48 | 0.60 | **0.00** | 1 | **0.00** | 44 | **0.00** | 14 | **0.00** | 0 | **0.00** | 5 |
| Gaspelle5 (32x5) | **0.00** | 0.50 | 6.03 | 1 | 0.10 | 0.50 | **0.00** | 3 | **0.00** | 37 | **0.00** | 13 | **0.01** | 0 | **0.00** | 25 |
| Gaspelle6 (36x5) | **0.00** | 0.80 | 5.43 | 1.40 | 3.50 | 0.70 | **0.00** | 19 | **0.00** | 43 | **0.00** | 15 | **-0.01** | 0 | **-0.01** | 26 |
| Christ50 (50x5) | 5.92 | 2.30 | **0.00** | 3.80 | 3.68 | 2.40 | 3.39 | 80 | **0.00** | 35 | **0.00** | 26 | **0.00** | 5 | **0.00** | 60 |
| Christ75 (75x10) | 2.04 | 9.80 | 2.57 | 9.40 | 2.26 | 10.10 | 0.88 | 207 | 0.53 | 514 | 0.53 | 93 | 1.07 | 54 | 0.53 | 110 |
| Christ100 (100x10) | 3.38 | 25.50 | 2.00 | 44.50 | 1.14 | 28.20 | 1.08 | 408 | **0.00** | 729 | **0.00** | 278 | **0.00** | 92 | 0.32 | 385 |
| Min27 (27x5) | **0.00** | 0.40 | **0.00** | 1 | 0.10 | 0.30 | **0.00** | 10 | **0.00** | 16 | **0.00** | 6 | **0.00** | 0 | **0.00** | 15 |
| Min134(134x8) | 4.49 | 49.6 | 4.22 | 2.58 | 1.75 | 48.3 | | | **0.00** | 488 | **0.00** | 336 | 0.07 | | **0.00** | 285 |
| Das88 (88x8) | 0.31 | 17.30 | **0.00** | 0 | 3.63 | 17.50 | 0.03 | 582 | **0.00** | 78 | **0.00** | 79 | **0.00** | 69 | **0.00** | 49 |
| Das150 (150x10) | 1.61 | 156 | 0.21 | 0 | 1.06 | 119.20 | | | 1.26 | 227 | 0.89 | 128 | 0.89 | 283 | **0.00** | 563 |
| Global Avg. | 1.54 | | 2.06 | | 1.70 | | 0.49 | | 0.14 | | 0.11 | | 0.16 | | 0.06 | |

It can be seen that proposed ALNS-VNS algorithm reached the BKS more than half of instances and achieved solutions that are as good as the BKS solutions of instances. So, it can be concluded that the proposed ALNS-VNS heuristic, is an efficient solution method for solving LRP. The experimental results also indicate that the ALNS-VNS algorithm has an improving potential even on large sets of problem instances.

### *4.3.4 New solutions*

The proposed ALNS-VNS heuristic improved two instances of Tuzun & Burke (1999). Tables 4.9 and 4.10 show the new solutions of related instances. Each table gives the information about the vendors opened, the associated routes, the total demand for each route and total cost related to the vendor.

Table 4.9 New solution for the instance 111122 (100 x 20 ) in Tuzun & Burke (1999)

| **Vendor 3** (Total cost of 463.63) | | |
|---|---|---|
| Routes | Customers | Demand |
| 1 | 22 82 28 100 65 29 37 6 63 39 | 146 |
| 2 | 14 49 76 7 61 86 52 89 55 | 141 |
| 3 | 13 33 53 8 43 69 2 15 44 | 146 |
| 4 | 93 46 64 | 41 |
| **Total** | 31 | 474 |
| **Vendor 17** (Total cost of 984.74) | | |
| Routes | Customers | Demand |
| 1 | 20 45 5 42 35 79 31 4 24 | 136 |
| 2 | 41 18 30 90 67 78 94 80 83 17 | 148 |
| 3 | 3 99 38 32 12 1 68 95 92 | 150 |
| 4 | 84 26 21 11 66 50 58 71 91 27 51 | 145 |
| 5 | 57 56 81 75 34 9 59 72 70 85 | 150 |
| 6 | 87 25 10 74 98 36 54 88 47 60 | 147 |
| 7 | 16 48 73 40 96 19 23 77 62 97 | 150 |
| **Total** | 69 | 1026 |
| **Overall cost** | 1448.37 | |

Table 4.9 shows the new solution obtained by ALNS-VNS for the instance 111122 (100 x 20) in Tuzun & Burke (1999). It was computed in 450 s and found a solution with the overall cost of 1448.37 ( GAP of -0.06%). The solution has two opened vendors and 11 routes for performing.

Table 4.10 New solution for the instance 131122 (150 x 20 ) in Tuzun & Burke (1999)

| **Vendor 5** (Total cost of 466.01) | | |
|---|---|---|
| Routes | Customers | Demand |
| 1 | 44 123 15 2 128 137 107 140 14 49 | 144 |
| 2 | 64 142 114 143 146 39 93 46 | 115 |
| 3 | 76 136 7 105 61 147 86 52 89 55 | 137 |
| 4 | 22 82 25 100 65 28 37 150 6 63 | 142 |
| **Total** | 38 | 538 |
| **Vendor 13** (Total cost of 467.20) | | |
| Routes | Customers | Demand |
| 1 | 73 40 145 118 96 110 19 23 108 | 142 |
| 2 | 48 16 139 129 102 119 41 18 30 | 143 |
| 3 | 111 120 132 87 78 67 90 60 47 | 145 |
| 4 | 54 36 125 98 10 74 29 124 149 88 | 149 |
| **Total** | 37 | 579 |
| **Vendor 14** (Total cost of 322.79) | | |
| Routes | Customers | Demand |
| 1 | 31 4 24 20 57 56 75 138 45 5 | 149 |
| 2 | 127 148 97 131 62 77 79 35 42 | 136 |
| 3 | 59 9 34 72 70 115 101 134 81 | 141 |
| **Total** | 28 | 426 |
| **Vendor 19** (Total cost of 566.95) | | |
| Routes | Customers | Demand |
| 1 | 51 3 92 95 85 116 109 68 1 | 138 |
| 2 | 99 122 12 38 32 130 104 27 | 112 |
| 3 | 33 144 112 69 133 43 8 53 126 11 | 148 |
| 4 | 26 106 13 17 80 94 83 113 84 103 | 150 |
| 5 | 91 117 58 135 141 71 121 50 66 21 | 149 |
| **Total** | 47 | 697 |
| **Overall cost** | 1822.95 | |

In table 4.10, the new solution is shown for the instance 131122 (150 x 20). The proposed algorithm obtained a solution which has an overall cost of 1822.95 and the GAP of -0.01% in 316 s. This solution requires 16 routes and 4 opened vendors.

### 4.4 Chapter Conclusions

In this chapter, we aim at hybridizing ALNS with VNS to improve the search ability of ALNS for solving the classical LRP problem instances. In the proposed hybrid ALNS-VNS algorithm, VNS is used as an elitist approach after ALNS. To evaluate the performance of the proposed hybrid ALNS-VNS algorithm, three different LRP problem sets with single product is used due to the lack of multi-product LIRP instances in literature. The performance of the proposed hybrid ALNS-VNS algorithm is also compared to the performances of six methods from the existing literature. Computational results indicate that ALNS-VNS can improve the search performance and the proposed ALNS-VNS algorithm has a great potential to solve the considered problem.

In the following chapter, we present an implementation of the proposed hybrid algorithm in a real-life beverage LIRP problem with defective manufacturing.

# CHAPTER FIVE
# CASE STUDY

## 5.1 Introduction

As a real-life implementation, a beverage distribution problem of a retail chain store is handled with the proposed ALNS-VNS algorithm in this section. As mentioned in Section 1.2, this study is focused on the multi-product LIRP. The performance of the proposed algorithm is tested on the classical LRP instances with single product taken from the related literature. Therefore, a numerical example consisting of 50 customers, five potential vendors and five products is solved by the proposed algorithm.

## 5.2 Problem Description and Data Gathering

In this study the beverage products manufacturing company which has five different located vendors in Ankara province is considered. The vendors produce mutually five major products. Since they all have defective rates for each product during the manufacturing process, shortages may occur for each vendor. All shortages are completely backordered. Therefore, we assume that all customers are willing to wait when a shortage occurs. The defective items can be kept in stock to dispose in a time. Fleet of vehicles, location of the vendors and the customers, amount of demands of the customers and the inventory decisions for each vendor is known. The aim of the case study is to evaluate the proposed multi-product LIRP model by designing cost efficient distribution system.

The multi-product LIRP with shortages has many parameters. The values of the parameters considered in this study are listed in Table 5.1. The values are derived in a certain range in accordance with the real values because of the data privacy of the company.

Table 5.1 Values of the inventory parameters for vendors

| | | Vendors | | | | | |
|---|---|---|---|---|---|---|---|
| **Products** | **Parameters** | **1** | **2** | **3** | **4** | **5** | $\alpha_p$ |
| **1** | $c^{pf}$ | 15 | 10 | 15 | 10 | 10 | 1 |
| | $c_t^{pf}$ | 2 | 5 | 3 | 4 | 5 | |
| | $c_h^{pf}$ | 4 | 5 | 5 | 4 | 5 | |
| | $c_b^{pf}$ | 12 | 8 | 10 | 10 | 9 | |
| | $S^{pf}$ | 100 | 150 | 110 | 120 | 160 | |
| | $PR^{pf}$ | 17000 | 12750 | 22000 | 23500 | 22500 | |
| | $\gamma^{pf}$ | 0.04 | 0.04 | 0.03 | 0.05 | 0.05 | |
| **2** | $c^{pf}$ | 19 | 12 | 15 | 12 | 13 | 0.5 |
| | $c_t^{pf}$ | 4 | 4 | 2 | 4 | 3 | |
| | $c_h^{pf}$ | 5 | 4 | 4 | 4 | 4 | |
| | $c_b^{pf}$ | 10 | 7 | 10 | 9 | 9 | |
| | $S^{pf}$ | 105 | 145 | 105 | 115 | 155 | |
| | $PR^{pf}$ | 16500 | 14950 | 25050 | 22200 | 25500 | |
| | $\gamma^{pf}$ | 0.04 | 0.05 | 0.03 | 0.03 | 0.04 | |
| **3** | $c^{pf}$ | 17 | 11 | 17 | 15 | 15 | 0.25 |
| | $c_t^{pf}$ | 4 | 4 | 2 | 2 | 5 | |
| | $c_h^{pf}$ | 5 | 4 | 3 | 2 | 2 | |
| | $c_b^{pf}$ | 10 | 9 | 10 | 7 | 9 | |
| | $S^{pf}$ | 109 | 132 | 150 | 105 | 142 | |
| | $PR^{pf}$ | 17100 | 15000 | 22600 | 22450 | 19950 | |
| | $\gamma^{pf}$ | 0.06 | 0.05 | 0.05 | 0.06 | 0.07 | |
| **4** | $c^{pf}$ | 16 | 10 | 19 | 13 | 15 | 0.5 |
| | $c_t^{pf}$ | 2 | 2 | 3 | 5 | 4 | |
| | $c_h^{pf}$ | 5 | 2 | 4 | 2 | 3 | |
| | $c_b^{pf}$ | 7 | 9 | 9 | 9 | 10 | |
| | $S^{pf}$ | 107 | 141 | 139 | 123 | 132 | |
| | $PR^{pf}$ | 16450 | 14500 | 26500 | 21500 | 21000 | |
| | $\gamma^{pf}$ | 0.02 | 0.03 | 0.04 | 0.02 | 0.04 | |
| **5** | $c^{pf}$ | 17 | 12 | 20 | 14 | 16 | 0.75 |
| | $c_t^{pf}$ | 3 | 2 | 4 | 4 | 2 | |
| | $c_h^{pf}$ | 5 | 2 | 4 | 5 | 5 | |
| | $c_b^{pf}$ | 10 | 9 | 8 | 8 | 6 | |
| | $S^{pf}$ | 108 | 130 | 133 | 125 | 124 | |
| | $PR^{pf}$ | 17000 | 12100 | 24000 | 26870 | 24800 | |
| | $\gamma^{pf}$ | 0.02 | 0.06 | 0.07 | 0.04 | 0.05 | |
| $O_f$ | | 10186 | 13546 | 7001 | 8505 | 12563 | |

The parameters can differ for each product type in each vendor. Among the reasons for this are the distance of the location to the raw material, the geographic characteristics of the location, additional costs related to the workforce etc. can be shown. Moreover, the volume/size unit coefficient, $\alpha_p$, which converts different product sizes to a standard unit volume, is shown in Table 5.1. The demand information for customers can be seen in Table 5.2 where CN stands for 'customer number'. The data of the annual sales is used for deriving customer demands.

Table 5.2 Demand information of customers

| | Product Type | | | | | | Product Type | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **CN** | **1** | **2** | **3** | **4** | **5** | **CN** | **1** | **2** | **3** | **4** | **5** |
| **1** | 467 | 550 | 879 | 760 | 964 | **26** | 345 | 576 | 964 | 960 | 967 |
| **2** | 231 | 345 | 798 | 850 | 875 | **27** | 246 | 421 | 937 | 740 | 894 |
| **3** | 276 | 346 | 876 | 740 | 978 | **28** | 297 | 479 | 980 | 875 | 976 |
| **4** | 240 | 412 | 873 | 735 | 924 | **29** | 234 | 469 | 975 | 865 | 995 |
| **5** | 350 | 342 | 776 | 645 | 837 | **30** | 368 | 435 | 850 | 760 | 976 |
| **6** | 351 | 486 | 750 | 750 | 992 | **31** | 346 | 467 | 971 | 945 | 982 |
| **7** | 126 | 396 | 675 | 590 | 976 | **32** | 379 | 435 | 820 | 890 | 967 |
| **8** | 457 | 423 | 796 | 760 | 871 | **33** | 345 | 672 | 834 | 745 | 896 |
| **9** | 134 | 485 | 976 | 845 | 937 | **34** | 316 | 337 | 756 | 586 | 762 |
| **10** | 356 | 375 | 765 | 765 | 895 | **35** | 675 | 574 | 891 | 795 | 974 |
| **11** | 722 | 657 | 972 | 750 | 978 | **36** | 312 | 475 | 965 | 860 | 812 |
| **12** | 476 | 535 | 967 | 645 | 962 | **37** | 243 | 338 | 786 | 865 | 832 |
| **13** | 124 | 450 | 963 | 850 | 896 | **38** | 321 | 537 | 934 | 840 | 954 |
| **14** | 345 | 385 | 971 | 625 | 974 | **39** | 298 | 435 | 765 | 660 | 785 |
| **15** | 340 | 475 | 940 | 460 | 961 | **40** | 376 | 465 | 821 | 745 | 926 |
| **16** | 367 | 423 | 631 | 760 | 732 | **41** | 329 | 457 | 834 | 685 | 896 |
| **17** | 245 | 385 | 730 | 780 | 862 | **42** | 367 | 467 | 697 | 745 | 945 |
| **18** | 645 | 569 | 725 | 690 | 945 | **43** | 345 | 446 | 895 | 760 | 978 |
| **19** | 242 | 465 | 930 | 820 | 980 | **44** | 321 | 467 | 995 | 645 | 936 |
| **20** | 247 | 432 | 975 | 735 | 968 | **45** | 352 | 576 | 990 | 755 | 947 |
| **21** | 234 | 376 | 896 | 760 | 976 | **46** | 397 | 379 | 960 | 500 | 958 |
| **22** | 267 | 423 | 975 | 830 | 937 | **47** | 346 | 624 | 845 | 740 | 895 |
| **23** | 210 | 467 | 960 | 750 | 978 | **48** | 178 | 446 | 968 | 800 | 879 |
| **24** | 645 | 516 | 850 | 860 | 915 | **49** | 247 | 467 | 970 | 690 | 985 |
| **25** | 349 | 437 | 965 | 870 | 990 | **50** | 254 | 576 | 973 | 700 | 934 |

The location information of the problem is shown Figure 5.1. Distances between any origin and destination are calculated via web-based distance measurement tools (www.maps.google.com) and are used to calculate the transportation costs. Taking real distances instead of those with Euclidean calculation in the coordinate plane provides more realistic results. As shown in Figure 5.1, the triangle stands for vendors, while the four-point star represents the geographical position of the customers. The number of the vehicles is limited to number of the vendors and is homogenous of capacity 10000. The cost of one route is fixed and set to 1000.

As stated in the previous section, the proposed algorithm adopts a SA procedure for an acceptance criterion. The parameters of SA procedure are selected as follows: starting temperature $\tau_{start}$ =1000, and the cooling rate $\varphi$ = 0.99975.

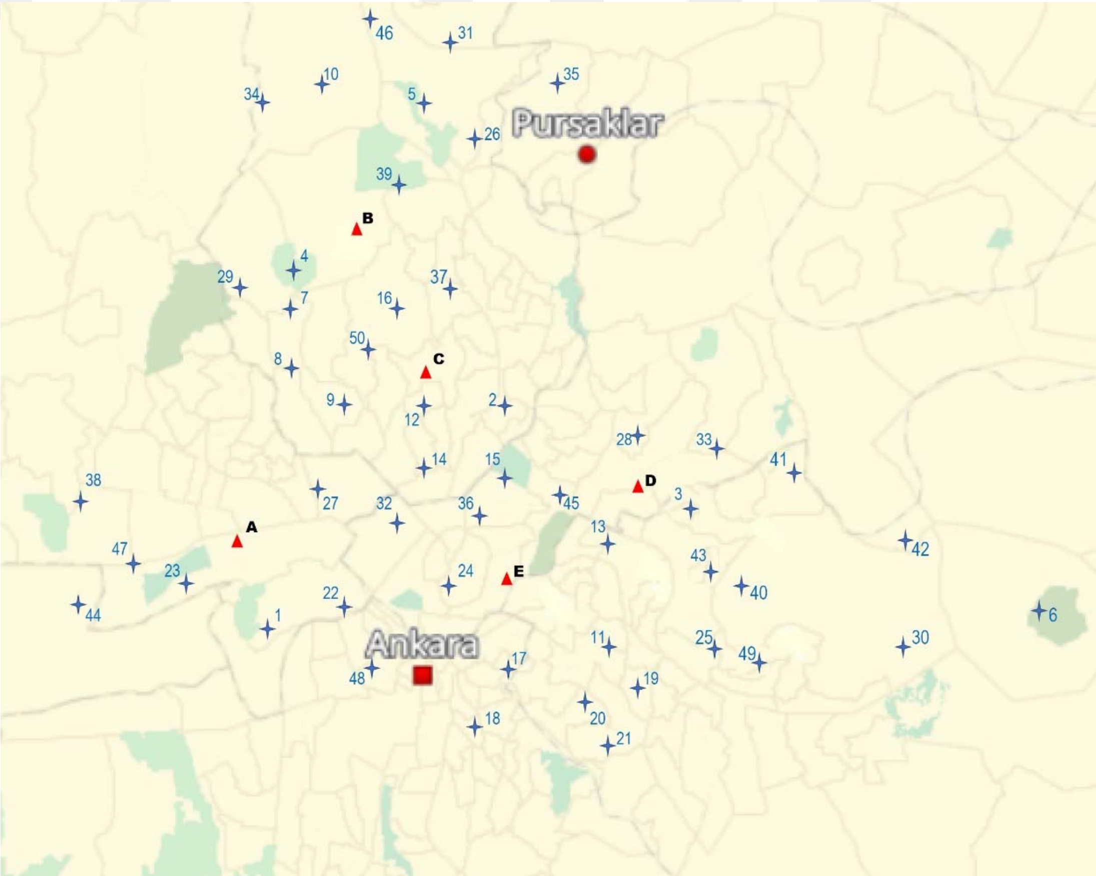


Figure 5.1 The location information of a beverage distribution problem (Retrieved from Google Maps in 2019)

### 5.3 Problem Solution

The proposed hybrid heuristics algorithm run for ten times, with 50.000 iterations in each. The optimized routes for real-life case can be seen in Table 5.3. The *Best Solution* is reported as the output. The output from the algorithm includes three vendors A, C, D, respectively. The vendor A has four routes with total travel distance of 217,79 km. The vendor C has three routes with total travel distance of 213,81 km. Lastly, the vendor D also has three routes with total travel distance of 165,68 km. The graphical illustration of this solution is shown in Figure 5.2. The total system costs of solution are given in Table 5.4.

Table 5.3 The optimized routes of the solution

| **Vendor A** | | |
|---|---|---|
| Routes | Customers | Distance (km) |
| 1 | 24 17 18 48 22 | 69.15 |
| 2 | 1 23 44 47 38 | 49.37 |
| 3 | 8 29 7 50 9 | 68.18 |
| 4 | 32 36 14 27 | 31.09 |
| Total | 19 | 217.79 |
| **Vendor C** | | |
| Routes | Customers | Distance (km) |
| 1 | 26 35 31 46 4 | 89.34 |
| 2 | 37 39 5 10 34 16 | 81.96 |
| 3 | 2 13 45 15 12 | 42.51 |
| Total | 16 | 213.81 |
| **Vendor D** | | |
| Routes | Customers | Distance (km) |
| 1 | 11 20 21 19 25 | 46.03 |
| 2 | 49 30 6 42 41 | 73.92 |
| 3 | 3 43 40 33 28 | 45.73 |
| Total | 15 | 165.68 |

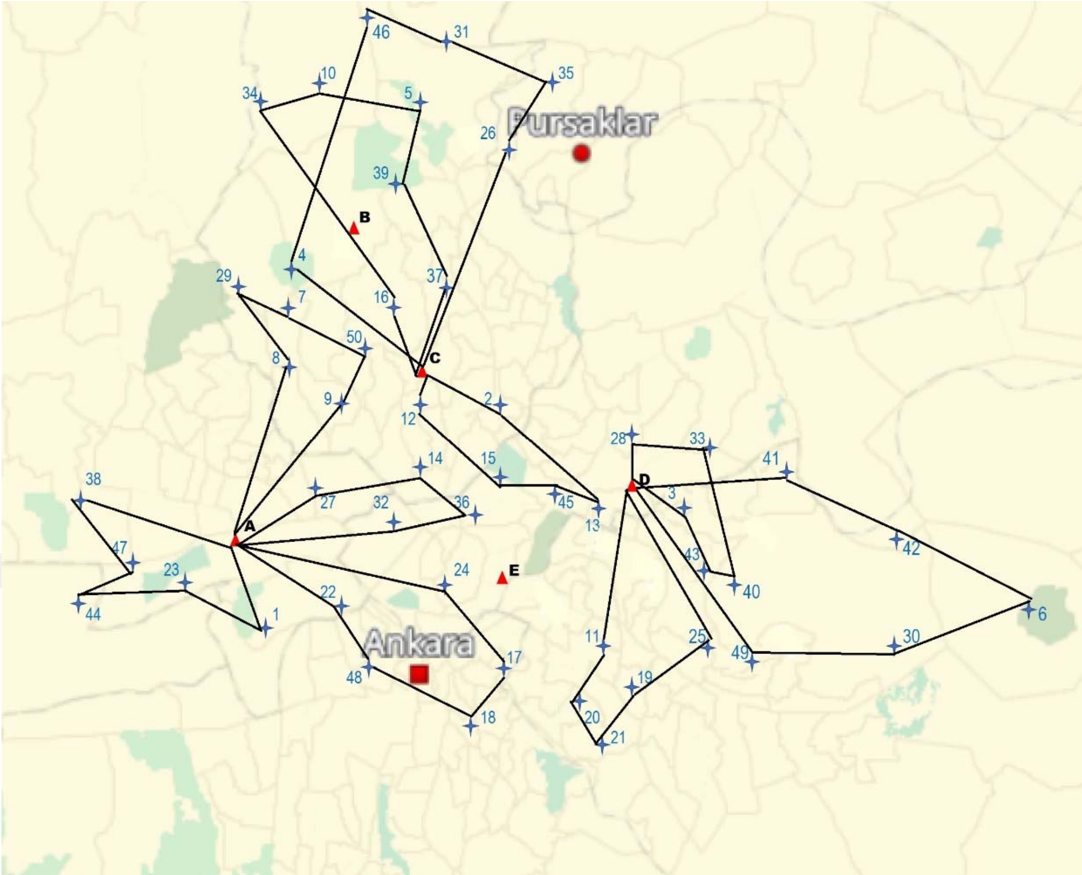


Figure 5.2 The graphical illustration of solution (Drawn on Google Maps picture in 2019)

Table 5.4 The cost components of the solution

| **Inventory** | **Costs** |
|---|---|
| Production cost | 2.4321$E$ + 6 |
| Inspection cost | 6.3510$E$ + 5 |
| Setup cost | 15426 |
| Holding cost | 2.2768 $E$ + 6 |
| Backordering cost | 8.2466$E$ + 5 |
| Holding cost for defective items | 2.3633$E$ + 6 |
| | |
| **Transportation** | |
| Travelling cost | 597.28 |
| Fixed cost | 25692 |
| Route cost | 10000 |
| Total | 8.5685$E$+ 6 |

## 5.4 Chapter Conclusions

In this section, it is aimed to implement the proposed hybrid algorithm on a real-life multi-product LIRP. For this purpose, the proposed algorithm is realized on a problem which is composed of 50 customers, five vendors and five products. As can be seen from Figure 5.2, the obtained solution is reasonable and feasible for real-life. The results endorse that the proposed algorithm is effective for LIRP.

# CHAPTER SIX
# CONCLUSION

## 6.1 Summary of the Thesis

This study defines the multi-product Location-Inventory-Routing problem (LIRP) with shortages as the problem of simultaneously considering the vendor location, vehicle routing and inventory related issues. The multi-product LIRP with shortages is formulated as a mixed-integer nonlinear programming problem. The mathematical model considers a defective manufacturing process in which the backorders might occur. Consequently, the LIRP is a NP-hard problem which could not be solved directly MINLP technique. Therefore, we present a hybrid meta-heuristic algorithm which is consisted of ALNS as a main method and VNS as an intensification procedure. To the best of our knowledge, hybrid meta-heuristic of ALNS and VNS has been used in this study for the first time for solving LIRP.

The performance of the proposed algorithm is evaluated on a standard benchmark set of LRP instances taken from the related literature, since there is a lack of standard benchmarks related to the considered LIRP problem in literature. The experimental results showed that the proposed algorithm achieved the BKS for 35 out of 61 instances. Besidesi it has been observed that ALNS-VNS could improve the already best-known solution of some large instances. The best improvements are 0.06% for the instance 111122 which has 100 customers and 0.01% for the instance 131122 which has 150 customers. Based on these results, it seems that the proposed algorithm can provide good solutions for the multi-product LIRP with shortages.

Moreover, the proposed algorithm is employed on a real-life LIRP having 50 customers, five potential vendors and five products. The algorithm is run for ten times and a reasonable solution is obtained for case study. By taking the real distances between supply chain actors, the solution has more realistic cost values. The case study demonstrates the ability of the algorithm to solve large scale LIRPs.

### 6.2 Contributions

The main contributions of this study is summarized as follows:

- A LIRP model, extended version of LRP, considering defective manufacturing process and inventory costs is proposed.
- A new mathematical model formulation for multi-product and multi-depot LIRP is developed.
- A hybrid meta-heuristic approaches for solving LIRP is proposed.

### 6.3 Future Research Directions

Future research may focus on more constraints and assumptions such as stochastic demands, heterogeneous fleets and time windows in MINLP presented in Section 3.3. As an another proposal, the hybrid ALNS-VNS may also be used for other LRPs and LIRPs that contain multi-echelon supply chains.

## REFERENCES


Ahuja, R. K., Ergun, Ö., Orlin, J. B., & Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, *123*(1-3), 75-102.

Aksen, D., Kaya, O., Salman, F. S., & Tüncel, Ö. (2014). An adaptive large neighborhood search algorithm for a selective and periodic inventory routing problem. *European Journal of Operational Research*, *239*(2), 413-426.

Ambrosino, D., & Scutella, M. G. (2005). Distribution network design: New problems and related models. *European Journal of Operational Research*, *165*(3), 610-624.

Argüello, M. F., Bard, J. F., & Yu, G. (1997). A GRASP for aircraft routing in response to groundings and delays. *Journal of Combinatorial Optimization*, *1*(3), 211-228.

Barreto, S. (2004). Análise e Modelização de Problemas de localização-distribuição. *Analysis and modelling of location-routing problems'),* PhD Thesis, University of Aveiro, Aveiro, Portugal.

Belo-Filho, M. A. F., Amorim, P., & Almada-Lobo, B. (2015). An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products. *International Journal of Production Research*, *53*(20), 6040-6058.

Bent, R., & Van Hentenryck, P. (2004). A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, *38*(4), 515-530.

Breunig, U., Schmid, V., Hartl, R. F., & Vidal, T. (2016). A large neighbourhood based heuristic for two-echelon routing problems. *Computers & Operations Research*, *76*, 208-225.

Canca, D., De-Los-Santos, A., Laporte, G., & Mesa, J. A. (2017). An adaptive neighborhood search metaheuristic for the integrated railway rapid transit network design and line planning problem. *Computers & Operations Research*, *78*, 1-14.

Chopra, S., & Meindl, P. (2007). Supply chain management. Strategy, planning & operation. In *Das summa summarum des management* (3rd ed.) (265-275). Wiesbaden: Gabler.

Coelho, L. C., Cordeau, J. F., & Laporte, G. (2012). The inventory-routing problem with transshipment. *Computers & Operations Research*, *39*(11), 2537-2548.

Coelho, L. C., Cordeau, J. F., & Laporte, G. (2013). Thirty years of inventory routing. *Transportation Science*, *48*(1), 1-19.

Dabiri, N., Tarokh, M. J., & Setak, M. (2012). A constructive heuristic for a multi-product inventory routing problem. *Transportation Research*, *2*(1), 11.

Daskin, M. S., Coullard, C. R., & Shen, Z. J. M. (2002). An inventory-location model: Formulation, solution algorithm and computational results. *Annals of Operations Research*, *110*(1-4), 83-106.

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., & Vogel, U. (2011). Vehicle routing with compartments: applications, modelling and heuristics. *OR Spectrum*, *33*(4), 885-914.

Diabat, A., Richard, J. P., & Codrington, C. W. (2013). A Lagrangian relaxation approach to simultaneous strategic and tactical planning in supply chain design. *Annals of Operations Research*, *203*(1), 55-80.

Duhamel, C., Lacomme, P., Prins, C., & Prodhon, C. (2008). A memetic approach for the capacitated location routing problem. *In Proceedings of the 9th EU/Meeting on Metaheuristics for Logistics and Vehicle Routing,* Troyes, France, 39-47.

Eskandarpour, M., Dejax, P., & Péton, O. (2017). A large neighborhood search heuristic for supply chain network design. *Computers & Operations Research*, *80*, 23-37.

Farahani, R. Z., Rashidi Bajgan, H., Fahimnia, B., & Kaviani, M. (2015). Location-inventory problem in supply chains: A modelling review. *International Journal of Production Research*, *53*(12), 3769-3788.

Ferreira, K. M., & de Queiroz, T. A. (2018). Two effective simulated annealing algorithms for the location-routing problem. *Applied Soft Computing*, *70*, 389-422.

François, V., Arda, Y., Crama, Y., & Laporte, G. (2016). Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, *255*(2), 422-441.

Gebennini, E., Gamberini, R., & Manzini, R. (2009). An integrated production–distribution model for the dynamic location and allocation problem with safety stock optimization. *International Journal of Production Economics*, *122*(1), 286-304.

Gholamian, M. R., & Heydari, M. (2017). An inventory model with METRIC approach in location-routing-inventory problem. *Advances in Production Engineering & Management*, *12*(2), 115.

Grimault, A., Bostel, N., & Lehuédé, F. (2017). An adaptive large neighborhood search for the full truckload pickup and delivery problem with resource synchronization. *Computers & Operations Research*, *88*, 1-14.

*Google Maps* (n.d.) Retrieved May 11, 2019, from www.maps.google.com.

Guerrero, W. J., Prodhon, C., Velasco, N., & Amaya, C. A. (2013). Hybrid heuristic for the inventory location-routing problem with deterministic demand. *International Journal of Production Economics*, *146*(1), 359-370.

Guerrero, W. J., Prodhon, C., Velasco, N., & Amaya, C. A. (2015). A relax-and-price heuristic for the inventory-location-routing problem. *International Transactions in Operational Research*, *22*(1), 129-148.

Hemmati, A., Stålhane, M., Hvattum, L. M., & Andersson, H. (2015). An effective heuristic for solving a combined cargo and inventory routing problem in tramp shipping. *Computers & Operations Research*, *64*, 274-282.

Hemmati, A., Hvattum, L. M., Christiansen, M., & Laporte, G. (2016). An iterative two-phase hybrid matheuristic for a multi-product short sea inventory-routing problem. *European Journal of Operational Research*, *252*(3), 775-788.

Hemmelmayr, V. C., Cordeau, J. F., & Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, *39*(12), 3215-3228.

Hiassat, A., Diabat, A., & Rahwan, I. (2017). A genetic algorithm approach for location-inventory-routing problem with perishable products. *Journal of Manufacturing Systems*, *42*, 93-103.

Ho, S. C., & Szeto, W. Y. (2017). A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem. *Transportation Research Part B: Methodological*, *95*, 340-363.

Hojabri, H., Gendreau, M., Potvin, J. Y., & Rousseau, L. M. (2018). Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. *Computers & Operations Research*, *92*, 87-97.

Hsu, L. F., & Hsu, J. T. (2016). Economic production quantity (EPQ) models under an imperfect production process with shortages backordered. *International Journal of Systems Science*, *47*(4), 852-867.

Huang, S. H., & Lin, P. C. (2010). A modified ant colony optimization algorithm for multi-item inventory routing problems with demand uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, *46*(5), 598-611.

Iassinovskaia, G., Limbourg, S., & Riane, F. (2017). The inventory-routing problem of returnable transport items with time windows and simultaneous pickup and delivery in closed-loop supply chains. *International Journal of Production Economics*, *183*, 570-582.

Jacobsen, S. K., & Madsen, O. B. (1980). A comparative study of heuristics for a two-level routing-location problem. *European Journal of Operational Research*, *5*(6), 378-387.

Javid, A. A., & Azad, N. (2010). Incorporating location, routing and inventory decisions in supply chain network design. *Transportation Research Part E: Logistics and Transportation Review*, *46*(5), 582-597.

Jha, A., Somani, K., Tiwari, M. K., Chan, F. T., & Fernandes, K. J. (2012). Minimizing transportation cost of a joint inventory location model using modified adaptive differential evolution algorithm. *The International Journal of Advanced Manufacturing Technology*, *60*(1-4), 329-341.

Koç, Ç. (2016). A unified-adaptive large neighborhood search metaheuristic for periodic location-routing problems. *Transportation Research Part C: Emerging Technologies*, *68*, 265-284.

Kulcar, T. (1996). Optimizing solid waste collection in Brussels. *European Journal of Operational Research*, *90*(1), 71-77.

Laporte, G., (1988). Location routing problems. In: B.L. Golden, & A.A. Assad, (Ed.). *Vehicle routing: Methods and studies* (2nd ed.) (163-197 ). Amsterdam: North-Holland.

Lian, K., Milburn, A. B., & Rardin, R. L. (2016). An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. *IIE Transactions*, *48*(10), 975-992.

Lieckens, K. T., Colen, P. J., & Lambrecht, M. R. (2013). Optimization of a stochastic remanufacturing network with an exchange option. *Decision Support Systems*, *54*(4), 1548-1557.

Liu, S. C., & Lee, S. B. (2003). A two-phase heuristic method for the multi-depot location routing problem taking inventory control decisions into consideration. *The International Journal of Advanced Manufacturing Technology*, *22*(11-12), 941-950.

Liu, S. C., & Lin, C. C. (2005). A heuristic method for the combined location routing and inventory problem. *The International Journal of Advanced Manufacturing Technology*, *26*(4), 372-381.

Lopes, R. B., Ferreira, C., Santos, B. S., & Barreto, S. (2013). A taxonomical analysis, current methods and objectives on location-routing problems. *International Transactions in Operational Research*, *20*(6), 795-822.

Lutz, R. (2014). Adaptive Large Neighborhood Search. Bachelor Thesis, Ulm University, Germany.

Ma, H., & Davidrajuh, R. (2005). An iterative approach for distribution chain design in agile virtual environment. *Industrial Management & Data Systems*, *105*(6), 815-834.

Majidi, S., Hosseini-Motlagh, S. M., & Ignatius, J. (2018). Adaptive large neighborhood search heuristic for pollution-routing problem with simultaneous pickup and delivery. *Soft Computing*, *22*(9), 2851-2865.

Mak, H. Y., & Shen, Z. J. M. (2009). A two-echelon inventory-location problem with service considerations. *Naval Research Logistics (NRL)*, *56*(8), 730-744.

Mancini, S. (2016). A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. *Transportation Research Part C: Emerging Technologies*, *70*, 100-112.

Maranzana, F. E. (1964). On the location of supply points to minimize transport costs. *Journal of the Operational Research Society*, *15*(3), 261-270.

Melo, M. T., Nickel, S., & Saldanha-Da-Gama, F. (2009). Facility location and supply chain management: A review. *European Journal of Operational Research*, *196*(2), 401-412.

Mete, H. O., & Zabinsky, Z. B. (2010). Stochastic optimization of medical supply location and distribution in disaster management. *International Journal of Production Economics*, *126*(1), 76-84.

Miranda, P. A., & Garrido, R. A. (2008). Valid inequalities for Lagrangian relaxation in an inventory location problem with stochastic capacity. *Transportation Research Part E: Logistics and Transportation Review*, *44*(1), 47-65.

Miranda, P. A., & Garrido, R. A. (2009). Inventory service-level optimization within distribution network design problem. *International Journal of production economics*, *122*(1), 276-285.

*Mixed Integer Nonlinearly Constrained Optimization Solver of SCIP* (n.d.) Retrieved May 11, 2018, from https://neos-server.org/neos/

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097-1100.

Moin, N. H., & Salhi, S. (2007). Inventory routing problems: A logistical overview. *Journal of the Operational Research Society*, *58*(9), 1185-1194.

Nagy, G., & Salhi, S. (2007). Location-routing: Issues, models and methods. *European Journal of Operational Research*, *177*(2), 649-672.

Naseraldin, H., & Herer, Y. T. (2011). A location-inventory model with lateral transshipments. *Naval Research Logistics (NRL)*, *58*(5), 437-456.

Nekooghadirli, N., Tavakkoli-Moghaddam, R., Ghezavati, V. R., & Javanmard, S. H. (2014). Solving a new bi-objective location-routing-inventory problem in a distribution network by meta-heuristics. *Computers & Industrial Engineering*, *76*, 204-221.

Or, I., & Pierskalla, W. P. (1979). A transportation location-allocation model for regional blood banking. *AIIE Transactions*, *11*(2), 86-95.

Palomo-Martínez, P. J., Salazar-Aguilar, M. A., Laporte, G., & Langevin, A. (2017). A hybrid variable neighborhood search for the orienteering problem with mandatory visits and exclusionary constraints. *Computers & Operations Research*, *78*, 408-419.

Park, J., & Kim, B. I. (2010). The school bus routing problem: A review. *European Journal of Operational Research*, *202*(2), 311-319.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, *34*(8), 2403-2435.

Pisinger, D., & Ropke, S. (2010). Large Neighborhood Search. In M. Gendreau, (Ed.). *Handbook of Metaheuristics* (2nd ed.) (399-420). Boston, MA: Springer.

Prins, C., Prodhon, C., & Wolfler-Calvo, R. (2004). Nouveaux algorithmes pour le problème de localisation et routage sous contraintes de capacité. In *Mosim 4*(2), 1115-1122.

Prins, C., Prodhon, C., & Calvo, R. W. (2006). A memetic algorithm with population management (MA| PM) for the capacitated location-routing problem. *In European Conference on Evolutionary Computation in Combinatorial Optimization,* Heidelberg, Germany, 183-194.

Prins, C., Prodhon, C., & Calvo, R. W. (2006). Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. *4OR*, *4*(3), 221-238.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., & Wolfler Calvo, R. (2007). Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, *41*(4), 470-483.

Prodhon, C., & Prins, C. (2014). A survey of recent research on location-routing problems. *European Journal of Operational Research, 238*(1), 1-17.

Rafie-Majd, Z., Pasandideh, S. H. R., & Naderi, B. (2018). Modelling and solving the integrated inventory-location-routing problem in a multi-period and multi-perishable product supply chain with uncertainty: Lagrangian relaxation algorithm. *Computers & Chemical Engineering*, *109*, 9-22.

Rayat, F., Musavi, M., & Bozorgi-Amiri, A. (2017). Bi-objective reliable location-inventory-routing problem with partial backordering under disruption risks: A modified AMOSA approach. *Applied Soft Computing*, *59*, 622-643.

Riquelme-Rodríguez, J. P., Langevin, A., & Gamache, M. (2014). Adaptive large neighborhood search for the periodic capacitated arc routing problem with inventory constraints. *Networks*, *64*(2), 125-139.

Riquelme-Rodríguez, J. P., Gamache, M., & Langevin, A. (2016). Location arc routing problem with inventory constraints. *Computers & Operations Research*, *76*, 84-94.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455-472.

Sajjadi, S. R., & Cheraghi, S. H. (2011). Multi-products location? routing problem integrated with inventory under stochastic demand. *International Journal of Industrial and Systems Engineering*, *7*(4), 454-476.

Salhi, S., & Rand, G. K. (1989). The effect of ignoring routes when locating depots. *European Journal of Operational Research*, *39*(2), 150-156.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *In Fourth International Conference on Principles and Practice of Constraint Programming*, Heidelberg, Germany, 417-431.

Shen, Z. J. M., Coullard, C., & Daskin, M. S. (2003). A joint location-inventory model. *Transportation Science*, *37*(1), 40-55.

Shen, Z. J. M., & Qi, L. (2007). Incorporating inventory and routing costs in strategic location models. *European Journal of Operational Research*, *179*(2), 372-389.

Smith, S. L., & Imeson, F. (2017). GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research*, *87*, 1-19.

Song, J. H., & Furman, K. C. (2013). A maritime inventory routing problem: Practical approach. *Computers & Operations Research*, *40*(3), 657-665.

Suh, D., & Ryerson, M. S. (2017). A large neighborhood search heuristic to establish an optimal ad-hoc hubbing strategy in the wake of a large-scale airport outage. *Journal of Air Transport Management*, *65*, 156-165.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, *31*(2), 170-186.

Tavakkoli-Moghaddam, R., & Raziei, Z. (2016). A new bi-objective location-routing-inventory problem with fuzzy demands. *IFAC-PapersOnLine*, *49*(12), 1116-1121.

Tiwari, S., & Solanki, T. (2013). An optimized approach for k-means clustering. *International Journal of Computer Applications*, *975*, 8887.

Tuzun, D., & Burke, L. I. (1999). A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research*, *116*(1), 87-99.

Von Böventer, E. (1961). Der Einfluß der Aufwertung auf die deutsche Zahlungsbilanz: Eine quantitative Untersuchung. *Weltwirtschaftliches Archiv*, 54-92.

Watson-Gandy, C. D. T., & Dohrn, P. J. (1973). Depot location with van salesmen—a practical approach. *Omega*, *1*(3), 321-329.

Webb, M. H. J. (1968). Cost functions in the location of depots for multiple-delivery journeys. *Journal of the Operational Research Society*, *19*(3), 311-320.

Xu, E., Wermus, M., & Bauman, D. B. (2011). Development of an integrated medical supply information system. *Enterprise Information Systems*, *5*(3), 385-399.

Yu, V. F., Lin, S. W., Lee, W., & Ting, C. J. (2010). A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, *58*(2), 288-299.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2009). A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, *36*(2), 1070-1081.

Žulj, I., Kramer, S., & Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, *264*(2), 653-664.