OPTIMAL ROUTING AND SCHEDULING IN E-COMMERCE LOGISTICS
USING CROWDSOURCING STRATEGIES

by

Eman Mohamed

A Thesis presented to the Faculty of the
American University of Sharjah
College of Engineering
In Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Engineering Systems Management

Sharjah, United Arab Emirates

July 2017

Approval Signatures

We, the undersigned, approve the Master's Thesis of Eman Mohamed.

Thesis Title: Optimal Routing and Scheduling in E-commerce Logistics using Crowdsourcing Strategies.

Signature

Date of Signature

(dd/mm/yyyy)

Dr. Malick Ndiaye
Associate Professor, Department of Industrial Engineering
Thesis Advisor

Dr. Mahmoud Awad
Assistant Professor, Department of Industrial Engineering
Thesis Committee Member

Dr. Steven Formanek
Assistant Professor, School of Business Administration
Thesis Committee Member

Dr. Moncer Hariga
Director, Engineering Systems Management Graduate Program

Dr. Ghaleb Hussein
Associate Dean for Graduate Affairs and Research
College of Engineering

Dr. Richard Schoephoerster
Dean, College of Engineering

Dr. Mohamed El-Tarhuni
Vice Provost for Graduate Studies

Acknowledgement

Foremost, I'm thankful and grateful to Allah for all the blessings bestowed upon me and for giving me patience to make it through despite all challenges.

I cannot fully express my appreciation to my advisor and mentor Professor Malick Ndiaye. His guidance, support, patience and expertly professional and caring communication have made the process amazingly smooth. His professionalism, enthusiasm and knowledge has been a major factor that helped me find my way through the different work obstacles. I'm so fortunate to be able to have his help, I couldn't have succeeded without his help every step of the way. Oceans of gratitude to him for all the extra efforts he has done, his hard work is very much appreciated!

I would like to express my very profound gratitude to my mother (Dr. Safia Mahgoub Karar), my father (Dr. Bukhari Ibrahim Mohamed), my sisters (Dr. Maisoon and Ms. Huda), and my brothers for being my biggest supporters to reach the utmost success. Their love and support at the toughest times and their faith in me throughout the challenging situations has made the journey easy. I can't thank them enough for all what they did.

I must express my gratitude for all my colleagues at the American University of Sharjah and specially Mustafa Assaf for all the favors he did for me. I'm grateful for his help, assistance and continued support. His contribution to my research success will forever remain in my heart.

Finally, I would like to thank the American University of Sharjah, the Department of Industrial Engineering and the Engineering Systems Management Program for the Graduate Teaching Assistantship, the experience I gained and the valuable lessons I learned.

Dedication

To Them ...

Abstract

The internet penetration has transformed the way people shop and has enlarged online shopping popularity. From 2014 to 2016, electronic commerce has increased by 50% and the forecast predicts a threefold increase by 2020. This growth creates various challenges for logistics providers when it comes to home deliveries. Researches showed that the last mile delivery accounts for 13% to 75% of logistics costs and is currently regarded as the most problematic and most polluting section of the entire supply chain. Till now, there is no straight forward solution but technology is paving the way to optimize this part of the logistical process. At the same time, the popularity of mobile applications is an opportunity for companies as they create valuable connections between companies and customers. This research aims to benefit from these connections to finish the last leg of delivery through the use of crowdsourcing. The idea is to use a pool of citizens to deliver products from several lockers distributed around the city to consumers' doorsteps. Deliveries occur during a set time window for each customer taking into consideration drivers' capacity and availability time. The problem as described is proven to be NP-Hard, thus its solution should be verified using exact and heuristic-based techniques for large size instances. We propose two Integer Linear programming (ILP) formulations with the objective of minimizing the total reward paid to crowd-workers through optimum assignment of parcels to drivers and optimum routing for each driver. The proposed models are first solved using Lingo software and their run time compared. Furthermore, a Variable Neighborhood Search (VNS) algorithm that uses a Variable Neighborhood Descent (VND) in its local search phase is developed using C++ programming language to solve large-size problems in a reasonable time. The proposed algorithm returned exact and near-optimal solutions within few seconds depending on the problem's size and all results were obtained within an average of 20 seconds. A sensitivity analysis is conducted by varying the main model parameters to determine inputs' their impact on the problem. Although there was a clear correlation between some parameters and the total delivery cost, some other experiments were inconclusive.

Search Terms: *Electronic commerce; home deliveries; last mile delivery; crowdsourcing; parcel lockers; Variable neighborhood search, variable neighborhood descent;*

Table of Contents

Abstract	6
Table of Contents	7
List of Figures	10
List of Tables	12
List of Abbreviations	13
Chapter 1. Introduction	14
1.1. Foreword	14
1.2. Electronic Commerce	15
1.2.1. Categories of E-Commerce	16
1.3. Last Mile Delivery Problem	19
1.3.1. Absence of parcels recipients	20
1.3.2. Delivery Windows.	21
1.3.3. Customers Density	21
1.3.4. Environmental challenges	22
1.4. Crowdsourcing in Logistics	23
1.5. Vehicle Routing Problem (VRP)	26
1.5.1. Capacitated Vehicle Routing Problem (CVRP)	26
1.5.2. Open Vehicle Routing Problem (OVRP)	27
1.5.3. Multi-depot Vehicle Routing Problem (MDVRP)	27
1.5.4. Travelling Salesman Problem (TSP).	27
1.6. Problem Statement	28
1.6.1. Delimitations	30
1.6.2. Assumptions	30
1.7. Thesis Objectives and Contributions	30
1.8. Research Significance	31

1.9. Methodology	32
1.10. Thesis Organization	32
Chapter 2. Literature Review	33
2.1. Last Mile Delivery Problem.....	33
2.2. Crowdsourcing in Last Mile Delivery	37
2.3. Vehicle Routing Problem.....	40
Chapter 3. Methodology	45
3.1. Modelling Environment	45
3.1.1. Drivers.....	45
3.1.2. Lockers.....	45
3.1.3. Customers.	45
3.1.4. Modelling Environment Setting.....	46
3.2. Optimization Models Formulation.....	47
3.2.1. Model I Formulation	49
3.2.2. Model II Formulation.....	54
3.3. Proposed Models Lingo™ Experimentation	56
3.3.1. Model I Experimentation	57
3.3.2. Model II Experimentation.....	59
3.4. Comparison of The Two Models	61
3.5. Possible Variations of Model I	63
Chapter 4. Variable Neighborhood Search Algorithm for the Proposed ILP Model...	66
4.1. Variable Neighborhood Search (VNS)	66
4.1.1. Initialization	70
4.1.2. Shaking Phase.	74
4.1.3. Local Search Phase by VND.....	82
4.1.4. Evaluation and Termination.....	86
Chapter 5. Experimentation, Results and Analysis.....	88

5.1. VNS Validation.....	88
5.1.1. Comparison between Lingo™ and the Variable Neighborhood Search Algorithm.....	89
5.2. VNS Experimentation.....	91
5.3. Sensitivity Analysis	93
5.3.1. Number of Lockers Variation.	94
5.3.2. Number of Customers Variation.	96
5.3.3. Number of Drivers Variation.	96
5.3.4. Drivers Capacity Variation	98
5.3.5. Fixed Cost to Variable Cost Ratio Variation.	99
Chapter 6. Conclusion and Future Work	100
References.....	102
Vita.....	113

List of Figures

Figure 1.1: Categories of E-commerce	19
Figure 1.2: The Last Mile	20
Figure 1.3: Crowdsourcing Ingredients	24
Figure 1.4: Single-Parcel Delivery Example	29
Figure 3.1: Distance Matrix	47
Figure 3.2: Proposed ILP Model I	49
Figure 3.3: Model I Distance Matrix	53
Figure 3.4: Proposed ILP Model II	54
Figure 3.5: Model I Total Delivery Cost, Lingo™	58
Figure 3.6: Model I Elapsed Time, Lingo™	59
Figure 3.7: Model II Total Delivery Cost, Lingo™	60
Figure 3.8: Model II Elapsed Time, Lingo™	61
Figure 3.9: Models I and II Total Delivery Cost, Lingo™	62
Figure 3.10: Models I and II Elapsed Time, Lingo™	62
Figure 4.1: Steps of the basic VNS	67
Figure 4.2: Steps of the basic VND	68
Figure 4.3: VNS Algorithm with VND	69
Figure 4.4: Implemented VNS Flowchart	69
Figure 4.5: Initial Solution	71
Figure 4.6: Construction of Initial Solution	72
Figure 4.7: Feasibility Check	73
Figure 4.8: Total Delivery Cost Calculations	74
Figure 4.9: Shaking Phase ($k = 1$) Example	75
Figure 4.10: Shaking Phase ($k = 1$)	76
Figure 4.11: Shaking Phase ($k = 2$) Example	77
Figure 4.12: Shaking Phase ($k = 2$)	78
Figure 4.13: Shaking Phase ($k = 3$)	79
Figure 4.14: Shaking Phase ($k = 4$) Example	80
Figure 4.15: Shaking Phase ($k=4$)	81
Figure 4.16: Local Search Phase ($l = 1$) Example	82
Figure 4.17: Local Search Phase ($l = 1$)	83

Figure 4.18: Local Search Phase ($l = 2$) Example	84
Figure 4.19: Local Search Phase ($l = 2$).....	85
Figure 4.20: Solution Evaluation	86
Figure 5.1: Total Delivery Cost, VNS	89
Figure 5.2: Elapsed Time, VNS	89
Figure 5.3: VNS Percentage Deviation from Lingo™	90
Figure 5.4: Total Delivery Cost VNS Experimentation.....	92
Figure 5.5: Elapsed Time, VNS Experimentation	92
Figure 5.6: VNS Elapsed Time Summary	93
Figure 5.7: Number of Lockers Variation I	94
Figure 5.8: Number of Lockers Variation II.....	95
Figure 5.9: Number of Customers Variation	96
Figure 5.10: Number of Drivers Variation I	98
Figure 5.11: Number of Drivers Variation II.....	98
Figure 5.12: Drivers Capacity Variation.....	99

List of Tables

Table 3.1: Proposed Models Notations	48
Table 3.2: Model I Number of Nodes	57
Table 3.3: Model I Lingo™ Objective Value and Elapsed Time	57
Table 3.4: Model II Number of Nodes.....	59
Table 3.5: Model II Lingo™ Objective Value and Elapsed Time.....	60
Table 5.1: VNS Objective Value and Elapsed Time	88
Table 5.2: Elapsed Time, Lingo™ vs. VNS	90
Table 5.3: VNS Experimentation.....	91
Table 5.4: VNS Elapsed Time Summary.....	93
Table 5.5: Number of Lockers Variation I.....	94
Table 5.6: Customers Allocation to Lockers	95
Table 5.7: Number of Lockers Variation II	95
Table 5.8: Number of Customers Variation.....	96
Table 5.9: Number of Drivers Variation I	97
Table 5.10: Number of Drivers Variation II	97
Table 5.11: Drivers Capacity Variation	98
Table 5.12: Cost Ratio Variation	99

List of Abbreviations

OECD	Organization for Economic Cooperation and Development
ISO	International Organization for Standardization
ICC	International Chamber of Commerce
B2B	Business-to-Business
B2C	Business-to-Consumers
B2G	Business-to-Government
G2G	Government-to-Government
C2C	Consumers-to-Consumers
C2B	Consumers-to-Business
C2G	Consumers-to-Government
LMD	Last Mile Delivery
VRP	Vehicle Routing Problem
CVRP	Capacitated Vehicle Routing Problem
OVRP	Open Vehicle Routing Problem
MDVRP	Multi-Depot Vehicle Routing Problem
TSP	Travelling Salesman Problem
mTSP	multiple Travelling Salesman Problem
MmTSP	Multi-depots multiple Travelling Salesman Problem
UCC	Urban Consolidation Center
ILP	Integer Linear Programming
VNS	Variable Neighborhood Search
VND	Variable Neighborhood Descent

Chapter 1. Introduction

In this chapter, electronic commerce and its categories, the last mile delivery problem, crowdsourcing and its application in logistic are briefly discussed. A brief introduction for the vehicle routing problem and some of its different variants is also given. Then, we present the problem under investigation as well as the thesis objectives, contributions and significance. Afterwards, the methodology of the research is explained and finally, the general organization of the thesis is presented.

1.1. Foreword

With today's fierce market competition, companies are trying to optimize their operations and speed up their processes so as to maximize their profit and reduce their costs. As a result, products can be delivered to customers at the lowest possible price which in turn increases customers' satisfaction and retention. Therefore, companies are implementing various strategies to optimize their operational efficiency and their entire supply chain. For customers, the success of this supply chain is judged through orders delivery experience which has a major impact on their future purchasing decisions. Inaccurate or late deliveries lower the efficiency of the supply chain whereas effective and cost-efficient logistics can be a real competitive advantage. When talking about the efficiency of the supply chain, the world has achieved a great success to maximize it; but when it comes to the last leg of delivery the road ahead is still long.

The internet penetration has transformed the way people shop and has enlarged online shopping popularity. Traditional shopping is losing its gleam and online shopping is becoming more favored as it is more convenient, easier and less stressing. With one click, and from where they are, shoppers can get enormous variety of products from different sellers without worrying about physical boundaries, shopping time constrains or even geographical areas. Online shopping experience will never disappoint customers as even the hard to get items can be easily found online; but what these customers really seek is an efficient, cheap, convenient and rapid home delivery. With this online shopping on the rise, retailers have to give sufficient and great attention to their distribution channels and delivery means. As technology keeps advancing and population gets larger, online purchasing will become a central part of people's life and will become more sophisticated with each year. There is no doubt that companies have

to optimize their distribution and delivery to compete in this new battleground or they will struggle to survive.

The internet penetration coupled with the popularity of mobile applications creates an opportunity for companies to save money and deliver their products faster. Some people have the desire and the time to be involved in products delivery and others have the necessity to enhance their income and earn some extra money. The existing technology tools can be used to leverage the daily routes of those people who are already on the road to make products deliveries. Several shipping challenges; delivering single product for example, can be overcome by matching deliveries to individuals' itineraries. Thus, through leveraging the internet and the crowd, companies can strike the correct balance.

The last leg of the delivery journey will continue to be the most crucial and problematic part of the supply chain. While there is no straight forward solution to it right now and no one has quite figured it out, technology is paving the way for companies to optimize this part of the logistical process. In this work, we aim to implement the ingenious innovation "crowdsourcing" to provide a new solution for this crucial element.

1.2. Electronic Commerce

It's no secret that the booming of the internet and the growth of World Wide Web (WWW) have changed the way people live, the way they work, the way they socialize and even the way they shop. Over and above, the change in society and the fact that more and more people are becoming active in the labor market has created difficulty for people to find convenient times to shop for their necessities. During the last decade, the internet has evolved into a well-established trading environment and shoppers have become much more comfortable with this new method of purchasing. In its early days, the internet was considered as a medium for advertising and marketing but with these changes it's becoming a preferred way for purchasing products, goods and services. The emergence of the internet coupled with the changes in social behaviors gave rise to what is called "Electronic Commerce".

Electronic Commerce or e-commerce hereinafter, in the narrow sense, is a type of business in which the process of buying and selling goods and services is performed over the internet. By common consensus, E-commerce traces back to 1979 when the British entrepreneur and inventor Michael Aldrich piloted a system to process a business and consumer transaction [1]. This system was the first demonstration of e-commerce concept followed by the appearance of two of the most dominant e-commerce players, Amazon and E-bay, almost two decades later [2]. According to Vladimir Zwass, Editor-in-Chief of International Journal of Electronic Commerce, “Electronic commerce is the sharing of business information, maintaining business relationships, and conducting business transactions by digital means over telecommunications networks”. In the literature, there is no universally accepted and standard definition of e-commerce and the term appeared in several contexts to match different circumstances. However, some important multinational corporations and international organizations that play a leading role in international economic activities gave some definitions for the term. The Organization for Economic Cooperation and Development (OECD) defined e-commerce as “an electronic transaction which is the sale or purchase of goods or services between businesses, households, individuals, governments and other public or private organizations, conducted over computer mediated networks” [3]. Moreover, the International Organization for Standardization (ISO) referred to e-commerce as “a general term for the exchange of information and requirements between companies, or between companies and their customers” [3]. Broadly, the International Chamber of Commerce (ICC) considered e-commerce as “the digitalization of all phases of trading activities in the entire trade process” [3]. All definitions agree that E-commerce is the use of internet to accomplish the trading process; however different models can be developed depending on the parties involved.

1.2.1. Categories of E-Commerce. According to Qin [4], E-commerce can be categorized into the following main categories based on the transaction. Those categories are:

1.2.1.1. Business-to-Business (B2B) E-commerce. B2B refers to the type of commerce where both parties involved in the e-commerce transaction are companies or businesses rather than businesses and individual consumers. Some businesses require

different types of goods and services that should be procured from other businesses and thus B2B e-commerce is an important mean of providing these needs. In general, any type of transaction that exists between two businesses; a wholesaler and a retailer or a manufacturer and a wholesaler for example, is referred to as Business-to-Business commerce. Studies show that B2B commerce will become the dominative share in e-commerce in future and it will take up the majority of total e-commerce trading volume [4].

1.2.1.2. Business-to-Consumers (B2C) E-commerce. While B2B e-commerce refers to transactions between businesses, Business-to-Consumers, or B2C, is the commerce model in which businesses connect, communicate and conduct all transactions with consumers who are the end-users of the products and services. In generic term, any process, from online retailing to online banking and travel services, in which businesses sell their products and services to consumers for their personal use falls under B2C e-commerce. There are many restrictions and factors that make this category take up a small proportion of e-commerce, however, B2C is developing rapidly and is considered as the main driver for e-commerce development [4].

1.2.1.3. Business-to-Governments (B2G) E-commerce. This type of e-commerce concerns with businesses selling their products and services to governments and governing bodies. It focuses on service exchange between businesses and governments such as the purchase of goods and services that government may need. Moreover, it gives businesses the opportunity to bid on government projects through real-time bidding. Broadly speaking, B2G e-commerce refers to the use of internet for trade and information exchange and all government-related operations. The size of Business-to-Government e-commerce is insignificant as a component of the total e-commerce and the businesses between governments and enterprises is still in its experimental phase [4].

1.2.1.4. Governments-to-Governments (G2G) E-commerce. This G2G e-commerce refers to the electronic sharing of data and business transactions between government departments and organizations as well as between two different governments. Government-to-Government e-commerce generally aims to support all governmental initiatives by improving efficiency and communication. In addition, it

aims to improve inter-government processes by streamlining coordination and cooperation [5].

1.2.1.5. Governments-to-Consumers (G2C) E-commerce. All electronic commerce activities performed between governments and their citizens are referred to as G2C e-commerce. The main focus of this model is to provide citizens with the necessary electronic resources to address their needs and make their transactions easier and less time consuming. The adoption of G2C e-commerce improves public services by giving customers instant and convenient access to government services and information at any time from everywhere. Moreover, the overcome of possible geographic and time barriers facilitates and increases citizen's interaction and participation in government [5].

1.2.1.6. Consumers-to-Consumers (C2C) E-commerce. This e-commerce model refers to the online transaction between consumers with one consumer selling goods or services to another consumer. In C2C e-commerce, two individuals, not businesses, electronically transact or conduct business with each other directly. Usually there exists an intermediary/third party to facilitate the transaction and to provide a platform for the people to connect to each other [6]. C2C e-commerce is the oldest e-commerce form and it mainly differs from B2B and B2C models in the direct interaction between consumers [6].

In contrast to these main categories, other e-commerce models reverse the traditional commercial transaction and allow businesses and governments to extract values from consumers. Consumers-to-Business (C2B) and Consumers-to-Government (C2G) are e-commerce models wherein consumers can behave as buyers and sellers at the same time. In C2B and C2G, individual consumers create value and businesses and governments consume that value. This value can be a service, where individuals perform a specific service and get paid, a product sold to businesses and governments at a cost, or an information, i.e. survey or idea, with a financial reward. Figure 1.1 shows all these categories of e-commerce.

Figure 1.1: Categories of E-commerce

The growth of various e-commerce models will create various challenges for logistics providers especially when it comes to delivering goods to end-customers. The expansion of B2C, C2C and G2C e-commerce will raise the desire for innovative delivery solutions and flexible delivery options. With this issue in mind, businesses, governments and consumers will strive to find ways to achieve success and to address what is known as the “last mile delivery” problem.

1.3. Last Mile Delivery Problem

The increase of online shopping increases the importance of home deliveries as it substantially generates additional parcels; packages to be delivered, for distribution. Companies that come up with creative ways to avoid late deliveries, unmet expectations and broken promises through timely and cost-effective deliveries will have the edge. In logistics, the delivery of goods to their final destination is what is referred to as the last mile delivery.

Last Mile is a term used in supply chain to describe the movement of goods from fulfillment centers or transportation hubs to their final destinations. In other terms, the last mile is the last leg of a product’s trip before it arrives on customers’ doorstep and it’s considered as the moment that matters [7]. In e-commerce environment, last mile delivery (hereinafter referred to as “LMD”) is the problem of transport planning

for delivering goods from e-retailers' hubs to their final destination in the area; end consumers' homes for instance [8]. Figure 1.2 illustrates the last mile.

Figure 1.2: The Last Mile

The problem with the LMD is that it represents a paradox between speed and cost; customers expect fast same-day or on-demand delivery yet they remain highly price sensitive and tend to prefer cheapest available delivery options. The main challenge for companies is to find ways to minimize shipping expenses while providing customers with rational delivery times and reasonable delivery costs. Above this, there are many other circumstances that increase LMD challenges such as incorrect customers' addresses, cramped customers' locations, drivers' shortages and some adverse economic conditions such as rising fuel prices. In addition to these circumstances, the followings are of the main factors that can adversely affect the efficiency of the last mile delivery:

1.3.1. Absence of parcels recipients. This problem, referred to as unattended delivery, is of increasing concern to e-commerce stakeholders and the proportion of first-time home delivery failure is more than 20% [9]. The fact that most houses are likely unoccupied during the working day along with the growth of leisure activities outside homes increase the probability that no one is actually home to receive the orders. As a result, orders may need to be shipped three or four times before being

successfully delivered. Failure to deliver orders to their intended recipients on the first delivery attempt causes inconvenience to customers, increases costs to delivery service providers and results in high carbon emissions. For logistics operators, returning items to distribution centers or transportation hubs and rearranging these failed deliveries generate additional costs. For customers on the other side, failed deliveries will eliminate the convenience and time saving of online shopping as they may need to handle this last mile of delivery and collect their returned orders from certain locations or alternative addresses. Moreover, a failure rate of 10% increases CO₂ emissions by 15% and the percentage increase can reach 75% with a failure rate of 50% [10]. Parcel recipients are critical factors for the success of home deliveries and delivery failures can be reflected poorly on both companies and delivery service providers.

1.3.2. Delivery Windows. To avoid “not at home” syndrome, companies may need to offer their customers delivery time frames options in which the customer will be available to receive the parcels. Increased requests for arranged deliveries lower LMD efficiency as customers prefer shorter delivery windows while companies favor longer windows to better optimize delivery routes. Customers tend to favor narrow time slots since broad ones will keep them tied to their homes for hours to attend and await their orders. On the other hand, adding restrictions on delivery times imposes huge costs for delivery service providers as costs of deliveries with time windows are considerably higher than for those without time windows [10]. For instance, offering a 3 hour delivery window is 30–45% more expensive than offering normal 9 hour delivery window [11]. At the end, the paradox is that tighter windows are more convenient for customers but costlier for logistics operators. To achieve this balance, some companies such as FedEx offers specific delivery times but at an additional charge over the basic delivery offering to alleviate the increased cost of restricting delivery times through shorter windows [11].

1.3.3. Customers Density. Customer density, which is the number of customers within a specific area, is also considered as one of the key operational and logistical challenges when delivering orders directly to customers [11]. Delivering few orders to low density areas where shipments can contain only one package exacerbates LMD expenses significantly especially with the rising transportation costs. There exists

a strong relationship between customers density and delivery routes efficiency [11]. Deliveries in central city areas results in high efficiencies in terms of number of deliveries per mile travelled. On the other hand, suburban areas and rural settings lower routes efficiency as they usually result in an increased distance per package, that is; very few deliveries per mile travelled. Suburban and rural density problem gives logistics providers less opportunities to optimize their delivery routes and hence results in greater route inefficiencies. Companies can increase market density by restricting deliveries to certain days of the week however this gives customers less choices and increase their inconvenience.

1.3.4. Environmental challenges. The steady increase in freight delivery vehicles result in many environmental problems such as traffic congestion, noise nuisance, harmful emissions and problems with public spaces. Freight vehicles represent around 15% of total traffic in urban areas and they have more significant impact than passenger vehicles due to their size and frequent stops [12]. The emissions from diesel-powered freight vehicles are harmful to climate and public health and hence delivery service providers should exhibit compliance with emission standards. Therefore, e-retailers and logistics providers should not focus only on reducing LMD costs but also on reducing carbon footprint from their freight vehicles. To minimize these emissions, providers should minimize the total number of their trucks and maximize the utilization of existing trucks. Moreover, they can shift to greener delivery types e.g. bikes however at the expense of delivery speed. The increasing requests for environmental friendly transportation forces last mile logistics providers to move into green logistics deals and think about LMD impact from ecological point of view.

The above-mentioned challenges and others dictate that companies should diligently manage the last mile of their deliveries. To meet customers' expectation regarding delivery times and the overall delivery experience, and to operate with the least possible cost, online retailers and businesses should invest in new technological solutions. For years, companies have been tweaking around many strategies and solutions to manage the last mile. Several solutions have been implemented such as consolidation centers, pickup stations, delivery boxes, alternative addresses, near stores and self-service parcel stations. Although these solutions ensure that 100% of the

parcels are delivered, customers are responsible of the final leg of the journey. As the importance of the last mile delivery is increasing, e-commerce giants such as Walmart, Amazon and Google are implementing some new innovative solutions. Amazon is testing delivery drones that deliver parcels to customers wherever they are at the moment and hence it's thinking beyond home delivery [13]. With these drones, and with a "Bring it to me" option from the consumer's mobile device, the customer does not have to remain in one place after placing the order. While delivery drones' solution is promising, there are still specific operation rules and a way to go before regulators open the skies for commercial use. Along with Amazon, Walmart and Google, among others, are currently piloting same-day delivery projects in selected locations [14]. Despite the pressures and costs of expedited transportation, Amazon, Walmart and Google, are considering the adoption of same-day delivery; intrigued by the newly emerging concept of crowdsourcing.

1.4. Crowdsourcing in Logistics

The internet and social media laid the ground for easy and cheap communication and collaboration between organizations and their customers. They have created valuable connections between companies and their users; which give companies access to new ideas and solutions. Likewise, they create an opportunity for companies to add significant values to their products and services according to customers' preferences. This phenomenon gives companies the opportunity to leverage the collective intelligence of these online communities to serve their business goals, optimize their tasks and reduce their costs. Hence, businesses have become concerned with how to build and sustain these online communities in the first place. The development of this web communication has led to the emergence of new models for business and one of these models, related to business and innovation, is known as crowdsourcing [15].

Being first discussed in [16], crowdsourcing is a term coined by Jeff Howe in 2006 to describe outsourcing a task to the crowd. The term is a compound contraction of "crowd" which is defined as a mass of people and "outsourcing" which refers to the shift of processes, functions, and duties to third parties [17]. As a first attempt to conceptualize the term, Howe defined crowdsourcing as "the act of a company or institution taking a function once performed by employees and outsourcing it to an

undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively), but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers” [18]. The use of crowdsourcing permits businesses to increase their value and visibility by involving large group of people from outside the company. However, this crowd becomes an essential connected resource and an integral part of the company. From Howe’s definition and according to [19], the key ingredients of crowdsourcing are:

- An organization that has a task it needs to perform.
- A community (crowd) that is willing to perform the task voluntarily.
- An online environment that allows the work to take place and the community to interact with the organization.
- Mutual benefit for the organization and the community.

Figure 1.3: Crowdsourcing Ingredients

The initial form of crowdsourcing was crowd working wherein a group of external people acted as employees and performed a work that was previously conducted within an enterprise [17]. Since then, crowdsourcing has steadily developed into new contexts across many disciplines and it has been used to eliminate inefficient operations and to invent entirely new ones in different fields ranging from geography to medicine [15].

In logistics sector, crowdsourcing delivery can be an answer to the growing expectations for faster and cheaper delivery services. Delivering products to their last mile is an activity that can be broken down to benefit from the availability and diversity of the crowd; hence the role of crowdsourcing in LMD cannot be overlooked. The transfer of crowd work to logistics sector gave a rise to a new concept known as crowd logistics. Being defined in [20], crowd logistics is “the outsourcing of logistics services to a mass of actors, whereby the coordination is supported by a technical platform, which is hosted and managed by a crowd logistics provider”. The platform acts as a communication medium and can be accessed via multiple ways such as web browsers, mobile phones or generally using the internet. The aim of this platform is to coordinate the demand and the supply for transport services [20]. The application of crowd logistics in the last mile delivery improves the existing delivery practices in terms of flexibility, speed and volume. Moreover, with the growing number of single-parcel deliveries the use of crowdsourcing, in opposition to traditional delivery services, reduces the environmental impact which is gaining a growing attention with the rising requests for green logistics.

Major e-commerce players, Amazon and Wal-Mart, are experimenting new delivery services that rely on crowdsourcing. In 2014, Amazon experimented a far more innovative method of providing same-day delivery to its customers in San Francisco and Los Angeles [21]. After being stung by shipping delays blamed on regional courier services such as UPS and FedEx, Amazon has been exploring various options to have its own delivery vehicles [21]. The company used a taxi-hailing mobile app to ship parcels via licensed taxi cabs providing customers with speedier deliveries and generating secondary revenue for cab drivers during their off-peak hours. As an alternative mode of delivery to speed up shipments while restraining costs, Amazon is considering “on my way” service that uses everyday people to pick up and drop off packages for customers [22]. With this service, average consumers could pick up their own parcels and deliver other parcels to their consumers at their convenience. At the side of Amazon, Wal-Mart is considering crowdsourcing the delivery of online purchases using in-store customers [23]. Their idea is to use in-store customers to drop off online orders packages that are on their route back home. In return, in-store

customers will have a discount on their shopping bills that covers the cost of their gas in return for the delivery of packages.

Crowdsourcing initiatives would undoubtedly be a cheaper alternative for e-retailers who are always looking for better options to get deliveries to customers on time. The broad participation that takes place at relatively no cost is what makes crowdsourcing a powerful way to cut the last mile delivery costs.

1.5. Vehicle Routing Problem (VRP)

Vehicle Routing Problem, referred to as “VRP” hereinafter, is a widely-studied optimization problem that arises in many practical contexts such as transportation, freight collection and distribution, and many others. The problem addresses organizations' concern of determining which orders to serve, by which route, and in what sequence so as to minimize operating and distribution costs. VRP is a generic term given to the whole class of problems that try to find the optimum delivery routes for a fleet of vehicles to serve a number of customers or cities. The goal of VRP is to find a set of least-cost vehicle routes such that each customer is visited exactly once by one vehicle and each vehicle travels only one route, originates and terminates its route at a central depot, and the capacity of the vehicles is not exceeded [24]. The problem was first introduced in 1959 as a “Truck Dispatching Problem” that models how a fleet of trucks can serve the demand for oil of a number of gas stations from a central hub and with a minimum traveled distance [24]. Since then, VRP has grown as an optimization problem commonly encountered in transport and logistics domain and has been extended in many ways that consider real-life characteristics and aspects. The extension of the classical VRP by adding different constraints on vehicles or customers resulted in a large number of variants. Among these variants, we will consider the followings:

1.5.1. Capacitated Vehicle Routing Problem (CVRP). The capacitated vehicle routing problem is an extension for the classical VRP wherein a new constraint regarding vehicle capacities is added. In the CVRP, all vehicles are identical and have a maximum capacity load. In other words, the demand of commodities for each designated route should not exceed the capacity of the vehicle serving that route [25].

1.5.2. Open Vehicle Routing Problem (OVRP). In the open vehicle routing problem, vehicles are not required to return back to the central depot after servicing the last customer in their route. This variant is used in industrial and service applications and is utilized in practice in delivering newspapers and packages to homes [26]. OVRP is mostly adopted by delivery companies that use contractors who are not employees of the company and who use their own vehicles to deliver packages to residential customers. Hence, these companies require neither the contractors nor their vehicles to be returned to the depot.

1.5.3. Multi-depot Vehicle Routing Problem (MDVRP). In real life contexts, companies utilize several depots and warehouses from which delivery vehicles depart when starting their routes. This kind of VRP is called multi-depots vehicle routing problem and it's much more complex than the classical problem. In MDVRP, each depot has a fleet of vehicles and the problem is to optimize the assignment of customers to depots. Each vehicle starts from a depot, service the customers assigned to it and terminates at the same depot [25].

1.5.4. Travelling Salesman Problem (TSP). The travelling salesman problem is one of the widely-used routing optimization problem in the field of operations research. Simply stated, TSP is the problem of minimizing the total distance travelled when a salesman has to visit a number of cities [27]. For a set of cities, given the travel distance between each pair of cities, TSP seeks the shortest possible path for the salesman to visit each city only once and return to his starting point. The TSP has been extended to multiple travelling salesman problem (mTSP) where more than one salesman is considered. Furthermore, mTSP has also been extended to multi-depot multiple travelling salesman problem (MmTSP) in which more than one salesmen depart from several starting depots and return to these depots. In general, since this problem is considered as a special case of the VRP, almost all VRP variants has been studied in TSP.

The vehicle routing problem has many other variants that have been widely studied to find optimum solutions for real life scenarios. The above briefly discussed variants are the ones that are mostly addressed in last mile delivery.

1.6. Problem Statement

As e-commerce continues to grow in a bigger way than before, e-businesses are confronted by a rising issue that could threaten customers' positive experience with online shopping. The increased volume of goods to be delivered coupled with the increased expectations for the delivery experience strengthens the difficulties of managing the last mile. A recent article mentioned that between 13% and 75% of all costs related to logistics are attributable to the last-mile [28]. Hence, online buyers may end up having delivery costs much greater than their actual purchase. The last mile problem is growing in tandem with soaring e-commerce sales and there is a need to provide lasting solutions that can stand up to the growing demand in the coming years.

In this work, we aim to leverage a pool of crowd-workers to finish the last leg of delivery; that is from a collection point to consumers' doorsteps. The problem under consideration can be described as follows: There exists, outside the city, an urban consolidation center (UCC) with unlimited capacity and an eco-friendly fleet of vehicles. At the UCC, goods are consolidated based on their destination and delivered to several parcel lockers distributed in the city. As an attempt to assure a successful delivery, parcels are not delivered to their locations unless the customer places a delivery request. At fixed intervals; every hour for instance, the available crowd-workers, the maximum number of tasks they can deliver, their current locations, their destinations and more importantly their availability time frame, that is; the time at which they are willing to deliver, are identified. If there is a single order to deliver, intuitively it will be assigned to the worker with the minimum additional cost. In case there is more than one request, they will be assigned to drivers in an economical way that minimizes the total reward paid by the company to crowd-workers. Drivers can either accept or reject the delivery requests and hence the rejected parcels should be reassigned. Figure 1.4 illustrates the problem through a single-parcel delivery example and explains the assignment process as follows:

1. When the parcel recipient is available, the customer places a delivery request.
2. As a second step, and at fixed intervals, only the available crowd-workers, their vehicle's capacities, their current locations, their destinations and their availability timeframe are identified.

3. Based on the available workers, the request is assigned to the driver with the minimum additional cost.
4. In case the worker rejects the delivery request, the parcel will be reassigned to the available driver with the second minimum additional cost.
5. In case the worker is willing to handle the delivery, he/she should accept the assigned task.
6. Finally, the assigned worker picks up the parcel from the locker and delivers it to customers' address.

Figure 1.4: Single-Parcel Delivery Example

We assume that the total cost is composed of a fixed delivery fee and a variable delivery fee; the former is a fixed fee rewarded for the driver when he is assigned a delivery task or set of tasks whereas the latter varies with the distance travelled. The objective of the proposed model is to seek optimum assignment of the parcels to drivers who are willing to deliver, as well as the optimum routing for each driver. The solution will maximize trips taken by a single driver thus maximizing rewards for drivers while minimizing company cost.

Since this problem will be modelled as a Vehicle Routing Problem or a Traveling Salesman Problem, it will be considered as an NP-Hard problem. To handle medium-sized and large-size instances, a heuristic method will be developed. Due to its simplicity and soundness in solving such kind of problems, a **Variable Neighborhood Search (VNS)** heuristic will be proposed.

1.6.1. Delimitations. This research tackles the final leg of product journey from parcels lockers to consumers' doorsteps. It takes into consideration neither the optimization of lockers selection nor the delivery from the UCC to the selected stations.

1.6.2. Assumptions. An urban consolidation center (UCC) with unlimited capacity and an eco-friendly fleet of vehicles exists outside the city. Parcels are shipped from the UCC to one of already existing parcel lockers based on their final destination. The selection of the parcel lockers as well as the scheduling optimization of delivery to these lockers is as optimal as possible.

1.7. Thesis Objectives and Contributions

The problem of “last mile” delivery is not a new occurrence but it has increased significantly with the growing electronic commerce. The objective of this work is to offer a lasting solution to the moment that matters and that, if not managed well, could leave customers with bitter disappointment. The main contributions of this research are:

1. Propose and formulate an **Integer Linear Programming (ILP)** crowdsourcing optimization problem that utilizes a pool of crowd-workers to perform the last mile delivery.
2. Present a new way of modelling that combines tasks' assignment and drivers' routing through the use of a distance matrix that includes all problem elements.
3. Propose and develop a **Variable Neighborhood Search (VNS)** heuristic to be used in solving medium-sized and large size problems.
4. Minimize total delivery costs paid by companies through optimal parcels allocation and drivers routing so as to minimize the total additional distance travelled by workers thus reducing the variable cost.

5. Lessen the environmental impact of last mile delivery through the use of urban consolidation center and the elimination of dedicated delivery trucks.
6. For companies, the elimination of freight transportation vehicles eliminates operations, maintenance and other costs such as insurances, benefits ... etc.
7. Reduce the number of missed deliveries substantially by using parcel lockers and delivering to customers only when they are available to receive their parcels.

1.8. Research Significance

From 2014 to 2016, worldwide e-commerce sales increased by almost 50% and the forecast for 2020 predict a threefold increase [17]. Nowadays, it's almost impossible to find a business that doesn't offer online shopping services and few years from now, it's most likely that people will be fortunate enough to order even their groceries from home. Customers, more often than not, expect doorsteps delivery in almost no time. Although these consumers' channels have a hard to predict demand, e-businesses should have quick response delivery systems. The last mile is currently regarded as the most expensive and most polluting section of the entire supply chain and hence the implemented solutions should address these challenges in this crucial part of the journey. Delivery trucks, due to their sizes and frequent stops, increase traffic congestions and air pollution considerably, thus they should be replaced by other alternatives. The utilization of vehicles that are already on the road instead of the use of dedicated freight vehicles reduces the traffic congestion and its associated polluting emissions and costs substantially. Above that, having no vehicles to maintain and no regular salaries and benefits to pay tends to save a lot for companies. For consumers, having a fast delivery at any time without being bound to delivery providers' operating hours is the best possible option. Therefore, crowdsourcing in LMD is a competitive differentiator for companies, a favorable solution for consumers and a direct contributor to sustainability.

The experience of customers at the last mile stage determines their common perception about the business which flatters around the market. The ability to fulfill customers' order on time reflects the success of the business and hence there has never been a better time to start thinking about the power of crowdsourcing in LMD.

1.9. Methodology

To achieve the outcomes of this research, the following steps will be followed:

Step 1: Extensive review of literature on crowdsourcing in last mile delivery, vehicle routing problem and the variants of the travelling salesman problem, and the heuristics methods used in solving such problems.

Step 2: Identify objective function, decision variables and constraints based on model assumptions.

Step 3: Formulate a mathematical model to optimize parcels' assignment and drivers' routings.

Step 4: Code the formulated model using appropriate optimization software.

Step 5: Develop a heuristic method to test the ability of the developed model to solve large instances.

Step 6: Perform sensitivity analysis to assess model's sensitivity to model inputs.

1.10. Thesis Organization

The thesis is organized as follows: Chapter 2 presents the literature review on the last mile delivery problem, the use of crowdsourcing in last mile delivery and the vehicle routing problem. Chapter 3 presents the formulation of the two linear programming models and the experimentations of these models. Chapter 4 illustrates the proposed Variable Neighborhood Search Algorithm and its phases. Chapter 5 presents the experimentations, results and the sensitivity analysis. Chapter 6 presents the conclusion and future research.

Chapter 2. Literature Review

Last mile delivery problem literature is vast and an extensive literature searches were done in the past years to address the interacting but poorly understood problem causes. In recent years, companies started to move toward efficient and reliable logistics that support the fulfillment of customers' promises and hence an impressive amount of literature is done on this problem. Although last mile delivery has been studied broadly with different proposed solutions, we restrict our attention to the application of crowdsourcing. Crowdsourcing is seen as an innovative and promising solution for many problems, yet we primarily focus on its application in this last leg of delivery.

In this chapter, the relevant literature about the last mile delivery problem, the use of crowdsourcing in last mile delivery and the vehicle routing problem will be reviewed.

2.1. Last Mile Delivery Problem

The last mile delivery, which refers to the final leg of the supply chain wherein products are delivered to their final consumers, is the most crucial element in e-businesses. For years, researchers have been trying to solve the economic, social and environmental pressure of the last mile delivery. The most studied problem is "unattended home delivery" when recipients are not at home during the delivery. Many studies such as [10, 29-33] presented alternative solutions for home deliveries to reduce rework which generates extra costs, extra kilometers, and extra emissions. Their idea is to ship the parcels to some collection points instead of customers' locations and the parcels are picked later by the customers at their convenience. Moreover, Van Duin *et al.* [29] disclosed that pre-delivery contact with customers is a promising concept to guarantee efficient delivery. Above and beyond, Agatz *et al.* [33] added that physical in-store pick-up points such as petrol-stations, groceries and supermarkets are also becoming common alternatives for home deliveries. In contrast, some studies argued that there exist some deliveries that require the presence of customers for a reason or another such as the necessity of customers' signature. Of these researchers, just to name few, are Ehmke *et al.* [34] and Agatz *et al.* [35] who studied routing optimization and timeslot management in "attended home delivery" problem respectively. The latter authors tried to identify specific time slots for different areas so as to minimize expected

total delivery cost. Results showed that narrow delivery time slots are convenient for customers but greatly reduce routing efficiency.

The environmental and economic impact of last mile delivery have been subject to much discussion as these impacts result in air pollution, noise and traffic congestion due to the low utilization of delivery vehicles. Handoko *et al.* [36] and Browne *et al.* [37], along with others, introduced Urban Consolidation Centers (UCC) to minimize delivery trucks and costs and to avoid vehicle-type restrictions in city centers. The idea is that orders from various carriers are consolidated at the UCC based on their destinations into the city, and then carried out by eco-friendly fleet of vehicles owned by the UCC for the actual last-mile delivery. One limitation of UCC that was highlighted in [37] is that they suit neither perishable and time sensitive goods nor goods that require special distribution and handling. Inspired by the need to reduce traffic congestion, air pollution and resources wastage, Liakos *et al.* [12] and Petrovic *et al.* [28] studied this problem as well. The former authors introduced a freight-pooling system that orchestrate shipments of multiple shippers and transfer their goods using significantly less resources thus providing an economically and environmentally efficient delivery. The latter authors investigated the opportunities to enhance end-to-end information flow by utilizing mobile communication systems to increase the efficiency of the shipment process as a whole. Their proposed approaches include the establishment of direct communication channels with recipients, interactive Communication channels between all participants and the integration of information flow of the sender and the recipient.

To put it briefly, last mile delivery is a crucial challenge for logistics services and it has become one of the bottlenecks of e-logistics [38]. The problem has been studied by many more researchers who tried to alleviate low efficiency and service dissatisfaction yet there is no room to be comprehensive.

Solution Approaches: Last Mile Delivery Problems are NP-hard problems, which means that it would be difficult to find and prove the existence of an optimal solution using exact algorithms within short computation time [39]. A study in Singapore revealed that courier companies may need to deliver more than 3,000 parcels each day to different locations and hence with these large problem sizes, it is necessary to

develop a heuristic approach to achieve higher efficiency and better service levels [39]. Several researchers, [39-47] to name a few, have developed different heuristics to solve last mile delivery problems in different contexts. For instance, Gunawan [39] and Isler *et al.* [46] developed Genetic Algorithms (GA) to solve a last mile delivery problem in smart cities and in automotive parts distribution problem respectively. In [39], the author developed a mathematical model to minimize the total waiting times for customers and carriers with narrow time windows and same-day delivery requests. Due to the difficulty to solve large instances, the authors then proposed two heuristic algorithms; a Greedy Algorithm to find the minimum number of vehicles to fulfil the deliveries, and a Genetic Algorithm to find the optimal routes for vehicles to minimize their waiting time considering customer's narrow time windows as hard constraints. The author further assumed that the number of fleet is enough and it consists of homogeneous vehicles with the same volume and weight capacity. In [46], the authors developed the model with the objective of determining the best number of delivery regions, assigning clients to each region to minimize the total transportation cost and then routing a set of vehicles within each region. They divided the problem into two stages; firstly, clients are clustered into groups by solving a capacitated p-median problem using a heuristic based on a novel Genetic Algorithm, and secondly, vehicles are routed for each cluster by solving a series of Capacitated Vehicle Routing Problems. The main goal of the hybrid Genetic Algorithm is to determine clusters of clients that are compact and homogeneous in size. As the resulting number of clients assigned to each cluster was not large, the Capacitated Vehicle Routing Problem was modelled as an integer programming formulation and solved to optimality.

On the other hand, Araújo *et al.* [40] implemented a simulation-based approach that combines an iterated local search metaheuristic with Monte Carlo Simulation. The authors believe that metaheuristics are applied to simplified scenarios where real-life uncertainty is not considered. However, when dealing with stochastic components an approach combining optimization with simulation is more appropriate. They also believe that the increase in mobility patterns in cities requires new services based on resource sharing such as the horizontal cooperation (HC) which increase resource utilization and reduce costs. In their context, the first part of the solution is the customer

allocation to depots, and the second one is to find the set of routes to serve the corresponding customers. Comparison results showed the power of the proposed simheuristic over heuristics and metaheuristics. Unlike [40], Sungur *et al.* [41] modelled uncertainty in a real-life courier delivery problem, in which customers appear probabilistically and their service times are uncertain, using normal heuristics. The authors used scenario-based stochastic programming to model uncertainty in customers and robust optimization for service times' uncertainty. The problem was formulated as a mixed-integer program model that maximizes customers' coverage and minimizes the total time spent by the couriers and the total earliness and lateness penalty. To solve large-scale problem instances, an insertion-based two-phase heuristic was developed and a tabu search improvement procedure was adopted afterwards.

Some researchers such as Schneider *et al.* [42] and Coelho *et al.* [43] developed heuristics to solve the delivery problem when vehicles have intermediate stops or pauses in their routes. The former authors in [42] developed an adaptive variable neighborhood search heuristic which combines ideas of variable neighbor search and adaptive large neighborhood search when vehicles have stopping requirements at intermediate facilities to replenish the goods to be delivered or to unload collected goods. The stops have different service times depending on the load level which imposes additional route duration constraints on the routing model. The objective of the proposed model is to minimize a total cost composed of travel cost and fixed cost for the deployment of vehicles. The latter authors in [43] tackled a delivery problem in furniture industry when drivers must take breaks pauses during their shift. They proposed a multi-start randomized local search heuristic in which customers have time windows and drivers' pauses must be within an interval in each route. Furthermore, the authors assumed a heterogeneous fleet of vehicles that route within a specific interval and has a maximum working time. The challenge in the problem context is that neither pauses locations nor the number of pauses are known as the number of routes is typically a decision variable. The objective is to minimize the total length of each route and the total travel distance. The results of the computational experiments demonstrated the importance of considering drivers' pauses in the optimization phase.

Balcik *et al.* [47] came up with a heuristic algorithm for the distribution of emergency relief supplies from local distribution centers to several demand locations

affected by disasters in humanitarian relief chain. The main decisions are the allocation of the relief supplies at the distribution centers among the demand locations and the determination of the delivery schedules and routes for each vehicle. The proposed model determines delivery schedules for vehicles considering vehicles capacity and delivery time restrictions with the objective of minimizing transportation costs and maximizing benefits to aid recipients. The model provides decisions about the customer delivery times, the number of items to be delivered at each visit, and the delivery routes. The authors assumed that the location of the distributions centers is predetermined and their capacity is sufficient to serve their service region.

To sum up, heuristics and meta-heuristics are generally used on last mile delivery in location-routing problems that seek to determine the optimal network configuration, i.e. location of consolidation facilities and vehicle routing sequences [45]. Logistics operators implement these methods to minimize the operating and fixed costs of vehicles and facilities, and to satisfy customers' demand and systems constraints, which are generally related to vehicles capacity or service times [45].

2.2. Crowdsourcing in Last Mile Delivery

In recent years, the use of crowdsourcing in last mile delivery has become a prolific and active area of research. Devari [48] investigated, through a study in city of Alexandria – Virginia, the benefits of using social networks in last mile delivery to reduce delivery times and costs, number of trucks in use and total miles travelled. The author studied the minimum level of friendship required for people to accept delivering products to peers, maximum extra time for which people are willing to accept participation in delivery and the monetary compensation. Results showed that retailers could save up to 8600 USD per day and achieve up to 55% reduction in pollutants with an average of 10 minutes extra time per delivery. Several researchers studied the problem with different criteria to optimize; for instance, [49] and [50] minimized the total reward paid by the logistic companies to crowd-workers and maximized the total payoff from tasks completed respectively. Wang *et al.* [49] proposed a crowdsourcing model that takes less than 10 seconds to find the optimal assignment of 2000 parcels to a pool of 500,000 crowd-workers. Parcels are initially sent to pop-stations and thereafter each parcel is assigned to a crowd-worker based on historical trajectory

patterns. After the successful delivery, workers are rewarded a certain amount of money based on the additional distance travelled. This model was formulated as a min-cost flow problem and was solved using Network simplex algorithm. Unlike our work, the authors assumed that the crowd-workers will always accept the delivery and the workers are rewarded only on the additional distance travelled. Moreover, in their model, the authors assumed a single destination and solved only parcels' assignment without finding the optimum routing. Chen *et al.* [50] introduced a centralized and more coordinated crowd-tasking approach that was initially formulated as an integer linear programming. The model tended to be intractable and hence a greedy heuristic was used to give an initial solution followed by an iterated local search heuristic to improve the quality of the initial solution. The authors concluded that the proposed centralized model outperforms the existing decentralized approaches in terms of tasks performed and detours overhead. The model proposed in [49] surpasses the one in [50] in terms of scalability as the latter tended to be intractable even for small-sized problems whereas the former handles millions of crowd-workers. Moreover, the use of heuristics in [50] gives approximate solutions while the model developed in [49] is very efficient and returns optimal results.

Another greedy algorithm was used in [51] to minimize total travel cost through minimizing total travel distance when workers' exact locations are obfuscated. The model first assigns tasks to workers based on their cloaked locations that follow certain probability distributions. Each participant then fine-tunes his tasks assignment based on his exact locations. The proposed approach proved its soundness in addressing locations uncertainty and achieving high task coverage with a low cost. Besides, Lee *et al.* [52] came up with a framework that minimizes an expected cost comprised of earliness and tardiness penalties, fuel consumption and carbon emissions while maximizes expected revenue based on distances traveled. The framework was modeled as a Markov decision process with a dynamic arrival of parcel delivery requests and a real-time integrated dynamic algorithm was developed for route scheduling afterward. Similar to our work, the authors assumed that the arrived requests can be either accepted or rejected based on expected cost, revenue and vehicle load capacity. Unlike our work, the algorithm dynamically establishes and updates routing sequence based on existing

and continuously arriving requests during the vehicle operation in a real-time manner. Another work similar to our context is in [53] in which the authors proposed a rolling horizon framework and developed an exact solution approach for a peer-to-peer platform that automatically creates matches between parcels and ad-hoc drivers. The platform also includes a set of backup vehicles to deliver the requests that cannot be served by ad-hoc drivers. The developed framework assigns tasks to ad-hoc drivers and backup vehicles and determines the associated delivery routes with the objective of minimizing the sum of the amount paid to ad-hoc drivers and the routing cost of dedicated vehicles. The amount paid to ad-hoc drivers is composed of a fixed fee for each task and a per-mile fee for distance travelled. The cost of using the dedicated backup vehicle depends on the distance travelled. In contrast to our study, the authors considered a problem setting in which tasks and drivers dynamically arrive over time whereas we consider a static problem in which all tasks are known in advance. Similar to our problem, the authors consider a setting in which parcels need to be delivered within a certain time window. The problem was solved in two stages with the first one being tasks assignment and the second one being vehicles routing. In our study, we combine tasks assignment and drivers routing in a single stage as separating assignment from routing may not yield the optimal solution to the problem.

Additionally, Chen *et al.* [54] proposed an ILP formulation to solve a multi-driver multi-parcel matching problem which incorporates drivers' maximum detour, their capacity, and the option of transferring parcels between drivers. The problem considers multiple hops and hence drivers and parcels can be matched without the requirement of sharing a similar or close destination as a parcel can move towards its destination one hop at a time. A driver with a completely different destination can still take the parcel to an intersection where it could be transferred to other vehicle that travels to the destination. The authors' objective is to determine the optimum matching between drivers and parcels, the optimum path of each driver and each parcel and the time schedule for drivers and parcels. The authors assumed an offline setting in which all drivers and delivery requests are known in advance before assignment. Two heuristics; the time compatibility heuristic and the time-expanded graph heuristic, have been developed for solving non-trivial problem instances. Results showed that an increasing number of drivers is beneficial for the company and socially desirable due

to reduction in traffic congestion and CO2 emissions. Relying on the same concept of ride sharing, some studies such as [8] and [55] explored the benefits of people and freight integrated transportation systems in which people and parcels are handled in an integrated way by the same taxi network. Chen *et al.* [8] illustrated through a methodological research that the problem is usually handled in two stages with the first being an offline taxi trajectory mining and the second being an online package routing and taxi scheduling. Furthermore, the authors acknowledged that the proposed solution utilizes occupied taxis while sending passengers and hence it's a green, economic and eco-friendly solution. Another interesting and innovative use of crowdsourcing can be seen in [56]. Pan *et al.* [56] proposed a sustainable solution to the e-commerce returns collection problem with this being the first qualitative and quantitative study of the concept. The authors developed an optimization-based simulation model that uses taxis to collect and deliver e-commerce returns from consumers back to retailers.

To cut it to the chase, the idea of using crowdsourcing in last mile delivery has been studied by a considerable number of researchers. However, to the best of our knowledge, the number of researchers who addressed it as a real-world optimization problem is scarce thus this problem is still considered as a relatively new trend.

2.3. Vehicle Routing Problem

Vehicle Routing Problem (VRP) has been a fruitful and vigorous area of research for decades and it has attracted the attention of large number of researchers who studied different variants of the problem. One of the most widely studied variants is the Capacitated Vehicle Routing Problem (CVRP) in which vehicles are identical and have a specific capacity. Several studies [25, 57-63] developed different heuristics under different conditions to design the least cost routing for vehicles starting and ending at the same depot. Of those researchers, Leggieri *et al.* [58] developed a mixed-integer linear programming formulations and a novel matheuristic to minimize the total transportation cost for the Asymmetric CVRP in which the cost matrix is asymmetric. Besides, [59-61] investigated the Cumulative CVRP in which the objective is to minimize the sum of arrival times at customers rather than total routing cost or distance. The authors concluded that this problem constitutes a good way to model situations where arrival time at customers is very important such as after natural disasters.

Moreover, Lei *et al.* [62] and Eufinger *et al.* [63] incorporated some uncertainty in their studies; the former authors developed a neighborhood search heuristic to solve the problem with stochastic demands while the latter authors came up with a robust approach that takes travel times uncertainty into account. Results confirmed the superiority of the proposed heuristics over an alternative solution approaches. Another similar variation to the CVRP is the Vehicle Routing Problem with Time windows (VRPTW) which imposes restrictions on the service time. VRPTM is based on the fact that customers usually expect their orders within a specified time slot and hence time windows identify earliest and latest times in which a vehicle can arrive at the customer to service. Studies in [64-67] presented memetic algorithm, Ant Colony Optimization, exact algorithms and Particle Swarm Optimization respectively to address this problem. In addition to them, Agra *et al.* [68] believe that delivery delays are frequent and should be accounted for and hence they developed an integer programming model that gives a robust feasible solution when travel times are not known with precision. Belhaiza *et al.* [69] considered the problem with multiple time windows when customers are offered several choices of delivery periods and a new hybrid variable neighborhood-tabu search heuristic was used to solve the problem.

The third category that has been studied in [70-74] is the Vehicle Routing Problem with Multiple Trips (VRPMT) wherein vehicles can perform multiple routes and can be used more than once during the working period. Authors developed different algorithms, heuristics and metaheuristics to find vehicles' routes and schedules that minimize the total length of the trip. Of these studies, [71, 73, 74] added time restrictions to the problem and assumed that each fixed-size vehicle can serve each customer within a specific time frame and it is not allowed to serve before or after this frame. Since some customers may not be served due to this time restrictions, these studies aim to minimize the number of unvisited customers as well as the total distance travelled. Wang *et al.* [73] concluded that incorporating time windows in VRPMT makes it very challenging problem since some of the visible routes may tend to be invisible due to the violation or overlap of customers' time windows. On the other side, some researchers [75-82] studied the Vehicle Routing Problem with Pickup and Delivery (VRPPD) in which vehicles not only deliver goods from central depot to

customers but also pick up some goods from them back to the depot. [75-79, 82] Highlighted the problem of simultaneous pickup and delivery where customers have a delivery and pickup demand which should be satisfied simultaneously by the same vehicle. Further, Polat *et al.* [76] added some time limits for the delivery and pickup process and assumed that each vehicle has a maximum service time for the client and a maximum total travel and service time before returning to the central depot. Similarly [77, 78] and [80] considered time windows and assumed that goods should be delivered and picked up within specific pre-defined time window. Above and beyond, Li *et al.* [82] extended the problem to include multi-depots and developed an Iterated local search adaptive neighborhood selection approach to solve multi-depots VRPPD. The authors concluded that considering joint distribution of multiple depots instead of a single depot results in more savings of cost.

Ant Colony Optimization, Genetic Algorithm along with other heuristics have been developed in [26, 83-86] to study the Open Vehicle Routing Problem (OVRP). In OVRP, vehicles are not required to return to their starting depot after delivering goods, and if they did, they must go through the same route in a reverse order. Sedighpour *et al.* [26] illustrated that the objective in OVRPs is usually to find the minimum number of vehicles required to serve customers as well as the least cost routings. Cao *et al.* [84] introduced a robust optimization model to solve OVRP when customers' demand is not known with certainty and belongs to specific uncertainty set. The authors assumed several scenarios to model uncertainty and developed an improved differential evolution algorithm to find a robust solution that is immune to data variations. Furthermore, Liu *et al.* [85] and Lalla-Ruiz *et al.* [86] extended the problem to multi-depots OVRP in which vehicles start from numerous depots. The former authors proposed a hybrid genetic algorithm to minimize the total travel cost and the latter authors enhanced the model suggested in [85] by changing sub-tour elimination constraints and adding some new constraints.

Finally, a widely studied problem that falls under VRP is the Travelling Salesman Problem (TSP) which is concerned with finding the best possible way for a salesman to visit several cities and return to his starting point with minimum distance and/or cost. Lots of studies highlighted this problem and some of them, to name a few,

can be seen in [87-96]. TSP has been studied with many variations; for instance, [90] added time windows within which customers must be visited and [91] considered time-dependent service time. Taş *et al.* [91] believe that in practice service times depends on several factors during the day such as the availability of parking in customers' area. Thus, the authors developed a mathematical model to minimize the total route duration "total travel time + total service time" and the model was then solved using CPLEX. One extension of this TSP is the multiple travelling salesman problem (mTSP) which includes several salesmen and each salesman travels to a unique set of cities before returning back to his starting point. Arya *et al.* [92] and Hosseinabadi *et al.* [93] developed a modified genetic algorithm and a hybrid metaheuristic algorithm respectively to find the least distance needed for each salesman to visit each city only once before returning to the starting place. Another extension is the multi-depot multiple traveling salesmen problems (MmTSP) in which more than one salesman depart from several starting cities and return to their starting city at the end. Ghafurian *et al.* [94] proposed an ant colony algorithm for MmTSP to find the routes for all salesmen to visit their customers while minimizing the total routes cost. Based on our research, and to the best of our knowledge, mTSP and MmTSP haven't received much attention despite their relevance to real-world problems. Last but not lastly, some researchers such as Bao *et al.* [95] and Ahmed *et al.* [96] studied the clustered travelling salesman problem (CTSP) in which points or cities, except the starting point, are divided into clusters. Different algorithms with different constraints have been proposed for solving CTSP; for instance, Bao *et al.* [95] proposed an approximation algorithm. The objective is to find the shortest or the minimum cost tour so that all cities are visited, cities of each cluster are visited consecutively and clusters are visited in a pre-specified order. In their study, the authors assumed that there is no specific starting and ending points for each cluster.

In conclusion, with the VRP being widely studied in the field of operations research, the literature in this problem and its variants is so vast with dozens of articles appear each year and we can't pretend to be comprehensive. A state of the art classification and review on the vehicle routing problem and its variants can be seen in [24].

Based on the conducted research, LMD problem in general and the use of crowdsourcing in particular have been widely studied from management point of view. The majority of the research highlighted the unattended deliveries and researchers introduced different ideas that consider delivering to an alternative collection points rather than directly to homes. The use of alternative pick up points proved its soundness in minimizing the number of failed deliveries. However, to the best of our knowledge, the majority of solutions require that the customer has to do the last mile of the journey which is not a convenient alternative. Moreover, those who considered crowd delivery from pickup points to customers' doorsteps didn't take into consideration neither the absence of recipients nor what crowd workers can do for the missed delivery. From this, we see an opportunity to fill the gap and examine the proposed solution that considers both the delivery from parcel lockers to consumers' homes and the availability of the parcel recipients. In addition, to the best of our knowledge, none of the researchers modelled the problem in a way that combines requests' assignment and drivers' routing in the same model. Separating the assignment part from the routing part may not give the optimum solution as routing the drivers, considering their current locations and destinations, may require different assignments. Above and beyond, up to our knowledge, modelling the problem by using a single distance matrix that considers all problem elements as nodes is not presented in the literature.

Chapter 3. Methodology

This work aims to solve a last mile delivery problem using a pool of crowd-workers. This chapter describes the model environment and presents the two optimization models developed. The proposed models are validated using Lingo™ Solver that finds the exact solutions.

3.1. Modelling Environment

The problem environment consists of drivers, parcel lockers and customers. The drivers are crowd-workers who are willing to deliver some requests on their way. Parcel lockers are the collection points that contain the requests to be picked up and delivered. The customers are the entities that have requests in the parcel lockers to be delivered by the drivers.

3.1.1. Drivers. The environment is made of D drivers $D = \{D_1, D_2, \dots, D_D\}$ where $D_i = \{cap_i, T_{i,d}, T_{i,u}, T_{i,max}\}$, and

$$i = 1, 2, \dots, D.$$

cap_i : Maximum number of tasks can be delivered by driver i .

$T_{i,d}$: Total delivery time for driver i .

$T_{i,u}$: Time taken by driver i to travel one distance unit.

$T_{i,max}$: Maximum availability time of driver i .

3.1.2. Lockers. The set of parcel lockers from which requests are picked up and delivered can be presented as a set of L lockers $L = \{L_1, L_2, \dots, L_L\}$.

3.1.3. Customers. The requests from customers can be represented as a set of C requests $C = \{C_1, C_2, \dots, C_C\}$ where each request can have different characteristics presented as $C_j = \{l_j, T_{j,a}, T_{j,b}, T_{j,e}\}$, and

$$j = 1, 2, \dots, C.$$

l_j : Locker that contains the request belongs to customer j .

$T_{j,a}$: Arrival time at customer j .

$T_{j,b}$: Beginning of delivery time window of customer j .

$T_{j,e}$: Ending of delivery time window of customer j .

3.1.4. Modelling Environment Setting. The modelled last mile delivery problem using a pool of crowd-workers is based on the following limiting assumptions to reduce the problem complexity:

- The total available capacity is always greater than the total number of requests to be delivered.
- The total availability time for all drivers is enough.
- There are no missed deliveries and no return since delivery will be within customers' specified delivery time windows and requests will be delivered only after the customer places a delivery request.
- Each customer has requests located in only one locker.
- Each driver can pick requests from multiple lockers.
- All drivers and requests are known in advance before assignment; there is no dynamic arrival of drivers and requests.
- Fixed cost, that is the delivery fee rewarded for the driver when assigned, is the same for all drivers.
- Variable cost, that is the delivery fee rewarded for the driver per unit of distance travelled, is the same for all drivers.
- Service time at customer locations and pickup times at lockers are not considered

As shown in figure 1.4, customers place a delivery request whenever they are available. These requests are available in one of the parcel lockers based on their final destination. At every interval, the requests ready for delivery and their delivery time window are known. The model assigns these requests to the available drivers based on their available capacity and total delivery time. It then routes the drivers to deliver the assigned requests in a way that minimizes the sum of the total fixed cost of using drivers and the total variable costs of distance travelled.

3.2. Optimization Models Formulation

The way the problem was modelled is that we built an $N \times N$ matrix with the first D nodes being drivers' current locations, the second L nodes being lockers' nodes, the third C nodes being customers' nodes and the last D nodes being drivers' destinations. Thus, each driver starts from his "D" node, go to one or more "L" node, go to one or more "C" nodes and other "L" nodes and finally go to his destination node. Then the mathematical model is developed to minimize the total delivery cost. Figure 3.1 shows how the developed distance matrix looks like.

	D_1	D_2	...	D_D	L_1	L_2	...	L_L	C_1	C_2	...	C_C	D_1	D_2	...	D_D
D_1																
D_2																
$\vdots$																
D_D																
L_1																
L_2																
$\vdots$																
L_L																
C_1																
C_2																
$\vdots$																
C_C																
D_1																
D_2																
$\vdots$																
D_D																

Figure 3.1: Distance Matrix

The problem is formulated in two different ways; in the first model drivers' movement is enforced or eliminated with the distance matrix by assigning a very high distance (M) between nodes that should not be connected and a normal distance between other nodes and hence this eliminate the need of adding some constraints. In the second model, the distance matrix is normal and movements are forced or eliminated with several constraints. Both models give the same value of the objective function with a difference in Lingo elapsed time.

Table 3.1 illustrates the notations used in the formulation of the two proposed models

Table 3.1: Proposed Models Notations

Notation	Description
FC_i	Fixed cost awarded to driver i if used for a delivery.
VC	Variable cost per unit of distance travelled by a driver.
y_i	Decision variable. Is 1 if driver i is used and is 0 otherwise.
x_{ijk}	Decision variable. Is 1 if driver i goes from node j to node k and is 0 otherwise.
d_{jk}	Distance between each pair of nodes j and k .
M	A very large number.
cap_i	Maximum number of tasks can be delivered by driver i .
$T_{i,d}$	Total delivery time for driver i .
$T_{i,u}$	Time it takes driver i to travel one distance unit.
$T_{i,max}$	Maximum availability time of driver i .
$U(j)$	Number of nodes visited from driver location to node j .
$U(k)$	Number of nodes visited from driver location to node k .
l_j	Locker that contains the request belongs to customer j .
l_k	Locker that contains the request belongs to customer k .
$T_{j,a}$	Arrival time at customer j .
$T_{k,a}$	Arrival time at customer k .
$T_{k,b}$	Beginning of delivery time window of customer k .
$T_{k,e}$	Ending of delivery time window of customer k .

Given these notations, the proposed formulations are as follows:

3.2.1. Model I Formulation

$$\begin{aligned}
 \min TCOST &= \sum_i FC_i * y_i + \sum_i \sum_j \sum_k VC * d_{jk} * x_{ijk} & (1) \\
 \text{s.t.} & \\
 \sum_i y_i &\leq D & \forall i & (1.1) \\
 \sum_i \sum_k x_{ikj} &\leq D & \forall j > D \text{ and } \leq (D + L) & (1.2) \\
 \sum_i \sum_k x_{ijk} &= \sum_i \sum_k x_{ikj} & \forall j > D \text{ and } \leq (D + L) & (1.3) \\
 \sum_j \sum_k x_{ijk} &\leq M y_i & \forall i & (1.4) \\
 \sum_j \sum_k x_{ijk} &\leq cap_i & \forall i, k > (D + L) \text{ and } \leq (N - D) & (1.5) \\
 \sum_i \sum_j x_{ijk} &= 1 & \forall k > (D + L) \text{ and } \leq (N - D) & (1.6) \\
 \sum_{k>D} x_{iik} &= y_i & \forall i & (1.7) \\
 x_{ijk} &\leq x_{iil_k} + \sum_j x_{ijl_k} & \forall i, \forall l, \forall j \text{ and } k > (D + L) \text{ and } \leq (N - D), l_j \neq l_k & (1.8) \\
 \sum_j x_{ijk} &\leq 1 & \forall i, \forall k > D \text{ and } \leq (D + L) & (1.9) \\
 \sum_j x_{ijk} &= \sum_j x_{ikj} & \forall k, \forall i & (1.10) \\
 x_{ijj} &= 0 & \forall i, \forall j & (1.11) \\
 x_{ijk} + x_{ikj} &\leq 1 & \forall i, \forall j, \forall k & (1.12) \\
 U(j) - U(K) + N * x_{ijk} + (N - 2) * x_{ikj} &\leq N - 1 & \forall i, \forall j > D, \forall k > D, k \neq j & (1.13) \\
 x_{ijk} &= 0 & \forall i, \forall j \leq D, \forall k \leq D \text{ OR } \geq (D + L) & (1.14) \\
 x_{ijk} &= 0 & \forall i, \forall j > D \text{ and } \leq (D + L), \forall k \leq D \text{ OR } > (N - D) & (1.15) \\
 x_{ijk} &= 0 & \forall i, \forall j > D \text{ and } \leq (D + L), \forall k > (D + L) \text{ and } \leq (N - D), l_k \neq j & (1.16) \\
 x_{ijk} &= 0 & \forall i, \forall j > (D + L) \text{ and } \leq (N - D), \forall k \leq D & (1.17) \\
 x_{ijk} &= 0 & \forall i, \forall j > (D + L) \text{ and } \leq (N - D), \forall k > D \text{ and } \leq (D + L), l_j \neq k & (1.18) \\
 x_{ijk} &= 0 & \forall i, \forall j > (N - D), \forall k \neq i & (1.19) \\
 T_{i,d} &= \sum_j \sum_k d_{jk} * T_{i,u} * x_{ijk} & \forall i & (1.20) \\
 T_{i,d} &\leq T_{i,max} & \forall i & (1.21) \\
 T_{k,a} &\geq T_{j,a} + (d_{jk} * T_{i,u}) - (1 - x_{ijk}) * M & \forall i, \forall j, \forall k > D & (1.22) \\
 T_{k,a} &\geq T_{k,b} & \forall k > (D + L) \text{ and } \leq (N - D) & (1.23) \\
 T_{k,a} &\leq T_{k,e} & \forall k > (D + L) \text{ and } \leq (N - D) & (1.24) \\
 x_{ijk} &\in \{0,1\}, i = 1, \dots D, j = 1, \dots N, k = 1, \dots N & & (1.25) \\
 y_i &\in \{0,1\}, i = 1, \dots D & & (1.26)
 \end{aligned}$$

Figure 3.2: Proposed ILP Model I

The objective function 1 focuses on minimizing the total delivery cost (TCOST) which is comprised of a total fixed cost that is considered as the sum of all fixed charges awarded to each driver if the driver is assigned for delivery and a total variable cost which is calculated as the sum of the total distance travelled by all drivers multiplied by the variable cost per one unit of distance travelled.

The proposed model is subject to the following constraints:

Constraints 1.1 and 1.2 are to assure that the total assignment and the total number of drivers entering the lockers for pickup is no more than the total number of drivers available for delivery. Constraint 1.3 guarantees that when a driver enters a locker he leaves that locker.

Constraints 1.4 and 1.5 are to limit drivers' delivery assignment to the maximum number of tasks they are willing to deliver. From the way we are modelling the problem, x_{ijk} will equal to 1 whenever a driver enters or leaves any node. However, only entrance to customers nodes affects the assignment capacity and hence in the constraint 1.5, nodes k are restricted to customers nodes only.

Constraint 1.6 is to ensure that each customer is visited exactly once whereas constraint 1.7 forces the selected drivers to start their trips from their current locations.

For each customer request, the request cannot be delivered unless it is picked up from the locker that contains it. By assigning a very large distance "M" between drivers' locations and any customer node and a normal distance between drivers' locations and any locker node, the solution will force any driver to go to the locker first. The movement between two different customers whose requests are in the same locker has no restrictions since the driver has already passed through the locker. However, the movement between two customers whose requests are not satisfied from the same locker is managed with constraint 1.8. In this constraint, j and k are customers' nodes with the locker of j being different from the locker of k ; hence, driver i cannot go from customer j to customer k unless he visited the locker of k at some point.

Constraint 1.9 restricts drivers to go to the same locker only once to pick all requests once.

Constraints 1.10 to 1.13 are for flow conservation and route continuity, preventing drivers from going to the same node, preventing drivers from going to only one node and the well-known Subtour Elimination Constraints (SEC) respectively. To account for our open path case, the distance from drivers' destinations back to their current locations is assigned as zero in the distance matrix so it doesn't affect the value of the objective function.

Constraints 1.14 to 1.19 are used to generate cuts in the optimization model. As each pair of nodes with a very large distance "M" should not be connected, we can set the values of x_{ijk} for these nodes to zero so that the model excludes these decision variables which reduces the time needed to return the optimum solution.

Constraints 1.20 to 1.24 are the time window constraints. The first two constraints assure that the total delivery time for each driver is less than the total available delivery time. Constraint 1.23 dictates that if customer k is visited directly after customer j then the arrival time at k is at least the arrival time at j plus the time needed to travel from j to k . The service time at customers' nodes and the pickup time at lockers' nodes are not taken into consideration. The last two constraints bound the delivery to customers to be no earlier than the beginning and no later than the ending of the time window.

Constraints 1.25 and 1.26 are the simple binary constraints.

3.2.1.1. Distance Matrix. The way the distance matrix was built in this model can be explained as follows:

3.2.1.1.1. Drivers' Locations Nodes: For each one of the first D nodes there is:

- A very high distance "M" to any other driver's location node as there is no need to pass through these nodes.
- A normal distance to any locker node.

- A very high distance “M” to customers nodes and drivers’ destinations nodes to force each driver go to lockers first.

3.2.1.1.2. Lockers Nodes: For each one of the second L nodes there is:

- A very high distance to drivers’ locations nodes as drivers should not go back to their starting location after visiting the lockers.
- A normal distance to any other locker node.
- A normal distance to customers nodes whose demand should be satisfied from the specific locker.
- A very high distance “M” to customers nodes whose demand is not satisfied from the specific locker as these are considered unnecessary trips.
- A very high distance to drivers’ destinations nodes to prevent drivers from going directly to their destination without delivering the picked requests.

3.2.1.1.3. Customers Nodes. For each one of the third C nodes there is:

- A very high distance “M” to drivers’ locations nodes as drivers should not go back to their starting location.
- A very high distance “M” to the corresponding locker node as drivers should not go back to an already visited locker.
- A normal distance to non-corresponding lockers nodes as drivers can go to other lockers to pick more requests.
- A normal distance to drivers’ destinations nodes.

3.2.1.1.4. Drivers’ Destinations Nodes. For each one of the last D nodes there is:

- A zero distance to the corresponding driver starting point to account for the open path in modelling.
- A very high distance to any other node as the last node in the tour should be the driver destination.

Figure 3.3 illustrates the distance matrix for ILP model I.

	D_1	D_2	D_3	L_1	L_2	C_1	C_2	C_3	C_4	D_1	D_2	D_3
D_1	M	M	M	d_{ij}	d_{ij}	M	M	M	M	M	M	M
D_2	M	M	M	d_{ij}	d_{ij}	M	M	M	M	M	M	M
D_3	M	M	M	d_{ij}	d_{ij}	M	M	M	M	M	M	M
L_1	M	M	M	d_{ij}	d_{ij}	d_{ij}	d_{ij}	M	M	M	M	M
L_2	M	M	M	d_{ij}	d_{ij}	M	M	d_{ij}	d_{ij}	M	M	M
C_1	M	M	M	M	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}
C_2	M	M	M	M	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}
C_3	M	M	M	d_{ij}	M	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}
C_4	M	M	M	d_{ij}	M	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}	d_{ij}
D_1	0	M	M	M	M	M	M	M	M	M	M	M
D_2	M	0	M	M	M	M	M	M	M	M	M	M
D_3	M	M	0	M	M	M	M	M	M	M	M	M

Figure 3.3: Model I Distance Matrix

Figure 3.3 is built with the assumption that the first 2 requests are fulfilled from the first locker and the second 2 requests are fulfilled from the second locker. Also, we assume that there are 3 available drivers to deliver and 2 lockers containing the 10 requests.

3.2.2. Model II Formulation

$$\begin{aligned}
 \min \text{TCOST} &= \sum_i FC_i * y_i + \sum_i \sum_j \sum_k VC * d_{jk} * x_{ijk} & (2) \\
 \text{s.t.} & \\
 \sum_i y_i &\leq D & \forall i & (2.1) \\
 \sum_i \sum_k x_{ikj} &\leq D & \forall j, j > D \text{ and } j \leq (D + L) & (2.2) \\
 \sum_i \sum_k x_{ijk} &= \sum_i \sum_k x_{ikj} & \forall j, j > D \text{ and } j \leq (D + L) & (2.3) \\
 \sum_j \sum_k x_{ijk} &\leq M y_i & \forall i & (2.4) \\
 \sum_j \sum_k x_{ijk} &\leq cap_i & \forall i, k > (D + L) \text{ and } \leq (N - D) & (2.5) \\
 \sum_i \sum_j x_{ijk} &= 1 & \forall k > (D + L) \text{ and } \leq (N - D) & (2.6) \\
 \sum_k x_{iik} &= y_i & \forall i, k > D \text{ and } \leq (D + L) & (2.7) \\
 \sum_k x_{ikj} &= 0 & \forall i, \forall j \leq D \text{ and } \neq i & (2.8) \\
 \sum_k x_{ijk} &= 0 & \forall i, \forall j \leq D \text{ and } \neq i & (2.9) \\
 \sum_j x_{ijk} &= y_i & \forall i, \forall k = \text{destination}(i) & (2.10) \\
 x_{iki} &= \sum_j x_{ijk} & \forall i, \forall k = \text{destination}(i) & (2.11) \\
 x_{ikj} &= 0 & \forall i, \forall k = \text{destination}(i), j \neq i & (2.12) \\
 \sum_j x_{ijk} &= 0 & \forall i, \forall k > (N - D) \text{ and } \neq \text{destination}(i) & (2.13) \\
 x_{iik} + \sum_j x_{ijk} &\leq x_{iil_k} + \sum_j x_{ijl_k} & \forall i, \forall l, \forall k > (D + L) \text{ and } \leq (N - D) & (2.14) \\
 x_{ijk} &\leq x_{iil_k} + \sum_j x_{ijl_k} & \forall i, \forall l, \forall j \text{ and } k > (D + L) \text{ and } \leq (N - D), l_j \neq l_k & (2.15) \\
 x_{ilk} &= 0 & \forall i, \forall l, \forall k > (D + L) \text{ and } \leq (N - D), l \neq l_k & (2.16) \\
 \sum_j x_{ijk} &= 1 & \forall i, \forall k > D \text{ and } \leq (D + L) & (2.17) \\
 \sum_j x_{ijk} &= \sum_j x_{ikj} & \forall k, \forall i & (2.18) \\
 x_{ijj} &= 0 & \forall i, \forall j & (2.19) \\
 x_{ijk} + x_{ikj} &\leq 1 & \forall i, \forall j, \forall k & (2.20) \\
 U(j) - U(K) + N * x_{ijk} + (N - 2) * x_{ikj} &\leq N - 1 & \forall i, \forall j > D, \forall k > D, k \neq j & (2.21) \\
 T_{i,d} &= \sum_j \sum_k d_{jk} * T_{i,u} * x_{ijk} & \forall i & (2.22) \\
 T_{i,d} &\leq T_{i,max} & \forall i & (2.23) \\
 T_{k,a} &\geq T_{j,a} + (d_{jk} * T_{i,u}) - (1 - x_{ijk}) * M & \forall i, \forall j, \forall k > D & (2.24) \\
 T_{k,a} &\geq T_{k,b} & \forall k > (D + L) \text{ and } \leq (N - D) & (2.25) \\
 T_{k,a} &\leq T_{k,e} & \forall k > (D + L) \text{ and } \leq (N - D) & (2.26) \\
 x_{ijk} &\in \{0,1\}, i = 1, \dots D, j = 1, \dots N, k = 1, \dots N & & (2.27) \\
 y_i &\in \{0,1\}, i = 1, \dots D & & (2.28)
 \end{aligned}$$

Figure 3.4: Proposed ILP Model II

The objective function 2 minimizes the total delivery cost (TCOST) to deliver all requests using the available drivers. It consists of a total fixed cost that is considered as the sum of all fixed charges awarded to each driver if the driver is assigned for delivery and a total variable cost which is calculated as the sum of the total distance travelled by all drivers multiplied by the variable cost per one unit of distance travelled.

The proposed model is subject to the following constraints:

Constraints 2.1 and 2.2 are to assure that the total assignment and the total number of drivers entering the lockers for pickup is no more than the total number of drivers available for delivery. Constraint 2.3 guarantees that when a driver enters a locker he leaves that locker.

Constraints 2.4 and 2.5 are to limit drivers' delivery assignment to the maximum number of tasks they are willing to deliver. From the way we are modelling the problem, x_{ijk} will equal to 1 whenever a driver enters or leaves any node. However, only entrance to customers nodes affects the assignment capacity and hence in the constraint 2.5, nodes k are restricted to customers nodes only.

Constraint 2.6 is to ensure that each customer is visited exactly once.

Constraints 2.7 to 2.9 are to force the selected drivers to start their trips from their current locations. The first constraint assures that whenever a driver is selected it starts the route from the corresponding "D" node whereas the last two constraints eliminate the other non-corresponding "D" nodes from driver's route.

Constraints 2.10 to 2.12 force the selected drivers to end at their destinations. Similarly, constraint 2.13 eliminates other destinations from driver's route.

For each customer request, the request cannot be delivered unless it is picked up from the locker that contains it. This condition is forced through constraints 2.14 to 2.16. Constraint 2.14 ensures that a driver cannot go to customer node k either directly or through any other node unless he visit the locker that contains request k first. The movement between two different customers whose requests are in the same locker has no restrictions since the driver has already picked up the request. However, the movement between two customers whose requests are not satisfied from the same

locker is managed with constraint 2.15. In this constraint, j and k are customers' nodes with the locker of j being different from the locker of k ; hence, driver i cannot go from customer j to customer k unless he visits the locker of k at some point before. Similarly, constraint 2.16 prevents drivers from going from a locker node to any customer node whose request is not fulfilled from the visited locker as this is considered as an unnecessary trip.

Constraint 2.17 restricts drivers to enter the same locker only once.

Constraints 2.18 to 2.21 are for flow conservation and route continuity, preventing drivers from going to the same node, preventing drivers from going to only one node and the well-known Subtour Elimination Constraints (SEC) respectively. To account for our open path case, the distance from drivers' destinations back to their current locations is assigned as zero in the distance matrix so it doesn't affect the value of the objective function.

Constraints 2.22 to 2.26 are the time window constraints. The first two constraints assure that the total delivery time for each driver is less than the total available delivery time. Constraint 2.24 dictates that if customer k is visited directly after customer j then the arrival time at k is at least the arrival time at j plus the time needed to travel from j to k . The service time at customers' nodes and the pickup time at lockers' nodes are not taken into consideration. The last two constraints bound the delivery to customers to be no earlier than the beginning and no later than the ending of the time window.

Constraints 2.27 and 2.28 are the binary constraints.

This model has a symmetric distance matrix with normal distances between each pair of nodes.

3.3. Proposed Models Lingo™ Experimentation

The proposed models have been validated using Lingo 16.0 which gives the exact solution that contains the optimum value of the objective function along with the values of the decision variables. To ensure that the models are valid for any instance and all the constraints are always satisfied, several experiments were conducted with

different data sets from the online TSP library “TSPLIB”. The experiments vary in the total number of nodes in the distance matrix. All the experiments were conducted using Intel(R) Xeon(R) CPU @ 3.50GHz with 16 GB RAM machine under Windows 7 (64 bit). The below sections detail the experiments run for each model.

3.3.1. Model I Experimentation. Several experiments were run to test the validity of the proposed ILP model I. Table 3.2 shows the data sets from the online TSP library “TSPLIB” with different number of drivers, number of lockers, number of customers and total number of nodes for each experiment run.

Table 3.2: Model I Number of Nodes

Experiment No.	Problem	No. of Drivers	No. of Lockers	No. of Customers	Total No. of Nodes
1	gr17	3	2	9	17
2	gr21	4	3	10	21
3	gr24	4	3	13	24
4	bays29	4	3	18	29

For each experiment run, Lingo provides the solution in terms of which requests will be assigned to which driver as well as the routing for each driver. The solution also gives the value of the objective function which is the total delivery cost. As mentioned before, the total delivery cost is composed of a total fixed cost rewarded to drivers for accepting delivery and a total variable cost which depends on the total distance travelled by drivers. Moreover, the solver gives the total time taken by Lingo to provide a solution for the proposed model. Table 3.3 illustrates the results obtained from solving the proposed model using Lingo solver for different number of nodes.

Table 3.3: Model I Lingo™ Objective Value and Elapsed Time

Experiment No.	No. of Drivers	No. of Lockers	No. of Customers	Total No. of Nodes	Objective Value	Lingo Elapsed Time (sec)
1	3	2	9	17	10,118	35.38
2	4	3	10	21	11,868	2998.29
3	4	3	13	24	16,040	3889.21
4	4	3	18	29	20,636	118820.31

Figure 3.5 demonstrates the optimum objective values provided by Lingo. The experiment number from Table 3.3 is presented in the x-axis, and the total delivery cost

(the objective value) in the y-axis. As shown in the figure, the total delivery cost increases with the increase in the number of customers. This is because either more drivers will be used to cover all customers while satisfying capacity restrictions or more distance will be travelled to visit all customers. The increase in the number of drivers increases the total fixed cost whereas the increase in the total distance travelled increases the total variable cost. Nevertheless, the delivery costs values shown in the figure are the minimum possible values for delivering all available requests using the minimum number of drivers.

Figure 3.5: Model I Total Delivery Cost, Lingo™

As a tool for finding an exact solution, Lingo solver takes a very long time to find the optimum solution. This time increases with the increase in the number of nodes to an extent that makes it neither logical nor practical to use Lingo to find solutions for large instances. Figure 3.6 shows the exponential increase in the elapsed time (y axis) with the increase in the number of nodes (x axis). The solver took 33 hours when the number of nodes is 29. This instance includes only 4 drivers, 3 lockers and 18 customers.

Figure 3.6: Model I Elapsed Time, Lingo™

3.3.2. Model II Experimentation. Similarly, several experiments were run to test the validity of the proposed ILP model II. Table 3.4 shows the number of drivers used, the number of lockers, the number of customers and the total number of nodes for each experiment run using different data sets from the online TSP library “TSPLIB”.

Table 3.4: Model II Number of Nodes

Experiment No.	Problem	No. of Drivers	No. of Lockers	No. of Customers	Total No. of Nodes
1	gr17	3	2	9	17
2	gr21	4	3	10	21
3	gr24	4	3	13	24
4	bays29	4	3	18	29

For each experiment run, Lingo provides the solution in terms of which requests will be assigned to which driver as well as the routing for each driver. The solver provides the value of the objective function and the total time taken by Lingo to provide a solution for the proposed model. Table 3.5 illustrates the results obtained from solving the proposed model using Lingo solver for different number of nodes.

Table 3.5: Model II Lingo™ Objective Value and Elapsed Time

Experiment No.	No. of Drivers	No. of Lockers	No. of Customers	Total No. of Nodes	Objective Value	Lingo Elapsed Time (sec)
1	3	2	9	17	10,118	118.85
2	4	3	10	21	11,868	3391.98
3	4	3	13	24	16,040	6186.15
4	4	3	18	29	20,636	N/A

Figure 3.7 demonstrates the optimum objective values provided by Lingo. The experiment number from Table 3.5 is presented in the x-axis, and the total delivery cost in the y-axis. As shown in the figure, the total delivery cost increases with the increase in the number of customers. This is because either more drivers will be used to cover all customers while satisfying capacity restrictions or more distance will be travelled to visit all customers. The increase in the number of drivers increases the total fixed cost whereas the increase in the total distance travelled increases the total variable cost. Nevertheless, the delivery costs values shown in the figure are the minimum possible values for delivering all available requests using the minimum number of drivers.

Figure 3.7: Model II Total Delivery Cost, Lingo™

Figure 3.8 shows the exponential increase in the elapsed time (y axis) with the increase in the number of nodes (x axis). The solver took around 2 hours when the number of nodes is 24. For the 4th experiment, when the number of nodes increased to

29, Lingo showed the value of the best objective within several hours however it has been running for more than 5 days without reaching this best objective from the starting objective bound and thus it didn't return the final global optimum solution, i.e. the final values of the decision variables. This instance includes only 4 drivers, 3 lockers and 18 customers.

Figure 3.8: Model II Elapsed Time, Lingo™

3.4. Comparison of The Two Models

As mentioned earlier, from a formulation point of view, the difference between Model I and Model II is that in the first model drivers' movement is forced or eliminated with the distance matrix by assigning a very high distance (M) between nodes that should not be connected and a normal distance between other nodes. In the second model, the distance matrix is normal and movements are forced or eliminated with several constraints. The two proposed models provide as expected, the same objective value. Figure 3.9 illustrates the optimum objective values provided by Lingo for the two models with the experiment number in the x-axis and the total delivery cost (the objective value) in the y-axis.

Figure 3.9: Models I and II Total Delivery Cost, Lingo™

In terms of the elapsed time, Lingo took different times to provide the optimum solution of the two models. Figure 3.10 illustrates the difference in the elapsed time for the two models.

Figure 3.10: Models I and II Elapsed Time, Lingo™

It can be clearly noted that Model I is superior to Model II in terms of solver run time and hence from here on, our focus will be on the first formulation. However, even for the first model, the time taken to find the solution is extremely large for trivial

number of requests “29 nodes with 18 requests and 4 drivers” and this dictates the importance of developing a heuristic method for solving large instances.

3.5. Possible Variations of Model I

The proposed mathematical model is formulated in a way that offers flexibility in modelling different problems by changing some constraints. In this section, we test two possible variations of the developed model.

- **Controlling the Number of Drivers Used.** In the current model, constraints 1.1 and 1.2 assign the delivery requests to any number of a maximum of “D” available drivers. Using constraint 1.1, the total number of drivers assigned can be controlled by simply changing the right-hand side of the equation to the maximum number of drivers to be used. The model will then find the best possible combination from the total number of available crowd-workers that provides enough capacity to deliver all customers’ requests.

Constraints 1.1 and 1.2:

$$\sum_i y_i \leq D \quad \forall i \quad (1.1)$$

$$\sum_i \sum_k x_{ikj} \leq D \quad \forall j > D \text{ and } \leq (D + L) \quad (1.2)$$

Modified Constraint 1.1:

$$\sum_i y_i \leq D' \quad \forall i \quad (1.27)$$

Where D' is the maximum number of drivers to be used from the total available drivers D .

- **Using UCC Vehicles as Extra Drivers.** Another extension to the problem could be to use the vehicles that deliver requests from the UCC to the lockers to deliver some requests directly to customers if their capacity and/or total delivery time permit. From modelling point of view, the formulation is valid to handle this variation with slight changes in some constraints that can be highlighted with the following simple example. For simplicity and illustration, we assume

that the set of D drivers consists of a subset of crowd-workers D_1 and another subset of UCC vehicles D_2 .

- The first difference between the crowd-workers and the UCC vehicle is that not necessarily all crowd-workers are assigned for delivery but all UCC vehicles must be assigned to visit at least all lockers if their delivery time doesn't permit to visit other customers' nodes. Thus, constraint 1.7 should be modified to exclude the UCC driver " D_2 " and a new constraint should be added to force the use of this driver. The new constraints are:

Modified Constraint 1.7:

$$\sum_{k>D} x_{iik} \leq 1 \quad \forall i, i \notin D_2 \quad (1.28)$$

Additional Constraint:

$$\sum_{k>D} x_{iik} = 1 \quad \forall i \in D_2 \quad (1.29)$$

- For the UCC vehicle, the constraint that forces the drivers to go to lockers first before going to any customer node doesn't hold as for the UCC driver no requests require any pickup from lockers. Hence, UCC driver " D_2 " should be excluded from constraint 1.8 and thus the new constraint is:

$$x_{ijk} \leq x_{iil_k} + \sum_j x_{ijl_k} \quad \forall i \notin D_2, \forall j \text{ \& } k > (D + L) \text{ \& } \leq (N - D), l_j \neq l_k \quad (1.30)$$

- As the UCC vehicles must visit all lockers' nodes to deliver the requests and visit customers' nodes only if their time window permits, the priority in their route is to go to all lockers nodes and hence a new constraint must be added to force this condition.

$$\sum_j x_{ijk} = 1 \quad \forall i \in D_2, \forall k > D \text{ \& } \leq (D + L) \quad (1.31)$$

- Unlike other drivers, UCC driver node has a normal distance to any customer node since there is no restrictions on visiting customers nodes directly. Hence some cut equations should exclude the UCC driver " D_2 ". Constraints 1.14 and 1.17 are modified to:

Modified Constraint 1.14:

$$x_{ijk} = 0 \quad \forall i, \forall j \leq D \text{ and } j \notin D_2, \forall k \leq D \text{ OR } \geq (D + L) \quad (1.32)$$

Modified Constraint 1.17:

$$x_{ijk} = 0 \quad \forall i, \forall j > (D + L) \text{ and } \leq (N - D), \forall k \leq D \text{ and } \notin D_2 \quad (1.33)$$

- Finally, as crowd-workers should end their route at their final destination, UCC vehicles must go back at the end of their route to their depot which is modelled as the same node as the starting location of the driver.

To end with, this problem is a multi-depots multiple travelling salesman problem (MmTSP) and hence its considered as an NP-Hard class of problems. Indeed, when only one driver is used and all the requests are satisfied from a single locker, the problem is reduced to a TSP which is known to be NP-Hard [86]. In [86], it has been proved that NP-hard problems cannot be solved to optimality within a bounded computational time and therefore heuristics are used to find near-optimal solutions. All heuristics cannot guarantee that the problem will be solved to optimality and it may not even be possible to state how close to optimality the solution is. Nevertheless, heuristics seek good, high quality solutions at a practical computational time. Therefore, in the next chapter we propose a Variable Neighborhood Search (VNS) algorithm to give near optimum solutions in a reasonable amount of time.

Chapter 4. Variable Neighborhood Search Algorithm for the Proposed ILP Model

Real-life distribution problems in logistics and Supply Chain Management are NP-hard, which means that exact solution methods can provide optimum solutions only for small instances [40]. Therefore, heuristics and metaheuristics methods are usually developed to provide near-optimal solutions to mid- to large-sized instances in a reasonable time [40]. In our problem environment, thousands of requests will be available for delivery and hundreds of crowd-workers will be used and hence it's not feasible to make the decision in a very long time. Thus, in this chapter we implement a Variable Neighborhood Search Algorithm to find near optimal solutions to the proposed mathematical optimization model. It has been shown in [97] that the Variable Neighborhood Search Algorithm is an effective method for getting approximate solutions for large single-objective optimization problems including scheduling problems, TSP and all its variants. Results of testing algorithm's solvability in benchmark symmetric Travelling Salesman Problem instances showed that the algorithm gives good quality solutions within up to 28.5% from the best-known solution in an execution time of 218 seconds when the problem size is 226. It also proved that changes of the order of neighborhoods makes a significant difference in the solution. We validated our proposed algorithm by comparing the solutions obtained to the exact solutions provided by Lingo™. The developed algorithm is proposed to minimize the total delivery cost, satisfying all constraints, while keeping drivers' maximum available time and customers' delivery time windows into consideration.

4.1. Variable Neighborhood Search (VNS)

Variable Neighborhood Search or VNS hereinafter is a metaheuristic algorithm proposed by Mladenović and Hansen [98] for solving combinatorial optimization problems. The basic idea of the VNS is that it explores different neighborhoods of an initial solution and moves from this current incumbent solution to a new one only if an improvement is made. The algorithm repeatedly applies a local search routine that uses local search methods to get from neighboring solutions to local optima. Local search methods are the simplest and oldest heuristics that start from an initial candidate solution and proceeds by applying a sequence of local changes in the initial solution

that improves each time the value of the objective function. The algorithm moves iteratively from a solution to a neighbor solution until no further improvements can be made or a stopping condition is met. The VNS algorithm exploits systematically the idea of neighborhood change relying on the following findings [99]:

- A local minimum with respect to one neighborhood structure is not necessarily an optimal with respect to another neighborhood.
- A global minimum is a local minimum with respect to all neighborhood structures.
- Local minima with respect to several neighborhoods are relatively close to each other.

Figure 4.1 illustrates the general idea of the Variable Neighborhood Search Algorithm that is used in finding the solutions for the proposed ILP model.

Initializations. Select the set of neighbourhood structures $N_k, k = 1, \dots, k_{max}$, that will be used in the search; find an initial solution x ; choose a stopping condition;
Repeat the following until the stopping condition is met:
 (1) Set $k \leftarrow 1$;
 (2) Until $k = k_{max}$, repeat the following steps:
 (a) **Shaking.** Generate x' at random from the k^{th} neighborhood of x ($x' \in N_k(x)$);
 (b) **Local search.** Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;
 (c) **Move or not.** If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with N_1 ($k \leftarrow 1$); otherwise, set ($k \leftarrow k + 1$);

Figure 4.1: Steps of the basic VNS [100]

The algorithm has three main steps; shaking or diversification, local search or intensification, and evaluation. The process starts by identifying a finite set of neighborhood structures N_k ($k = 1, \dots, k_{max}$) that will be used in the search, with $N_k(x)$ being the set of solutions that can be obtained from the k^{th} neighborhood of x . A stopping condition is identified and this condition can be a maximum elapsed time, a maximum number of iterations or a maximum number of repetitions with no improvement. An initial solution x is generated randomly and the algorithm then starts with the first pre-specified neighborhood structure ($k = 1$) to generate another solution. The idea of the shaking phase is to diversify the search through making several random moves to escape from local optima. With the generated shaking solution x' , the

algorithm seeks an improvement in the neighborhoods of the current solution through applying a local search method as the local optimum within one neighborhood is not necessarily an optimum one within another. At the end, the algorithm performs an evaluation to accept only the improved solutions compared to the current solution.

The basic VNS can be implemented easily using any local search algorithm as a subroutine in the local search step [98]. Studies showed that local optima trap problem can be resolved by applying a deterministic change of neighborhoods. As a result, a Variable Neighborhood Descent algorithm, VND hereinafter, can be used as a local search routine and hence it's usually used within metaheuristics. VND hereinafter, is the simplest variant of the basic Variable Neighborhood Search in which the change of the neighborhoods is performed in a deterministic way [101]. The idea of the VND is that it explores different pre-specified neighborhoods structures until a local optimum for all considered neighborhoods is reached. The algorithm starts with an initial solution and explores different neighborhood structures until a local optimum is found. After that, the algorithm switches to a different neighbor structure that may lead to further progress. VND is frequently used as a local search routine in other metaheuristics' contexts [101]. Figure 4.2 illustrates the general idea of the VND.

Initializations. Select the set of neighborhood structures $N'_k, k = 1, \dots, k'_{max}$, that will be used in the descent; find an initial solution x ;

Repeat the following until no improvement is obtained:

- (1) Set $k \leftarrow 1$;
- (2) Until $k = k'_{max}$, repeat the following steps:
 - (a) **Exploration of neighborhood.** Find the best neighbor x' of x ($x' \in N'_k(x)$);
 - (b) **Move or not.** If the solution thus obtained x' is better than this local optimum is better than x , set $x \leftarrow x'$; otherwise, set $k \leftarrow k + 1$;

Figure 4.2: Steps of the basic VND [100]

In our work, we implemented a VNS algorithm that uses the VND in the local search phase. Figure 4.3 illustrates the VNS with VND used.

Initializations. Select the set of neighborhood structures $N_k, k = 1, \dots, k_{max}$, that will be used in the shaking phase, and the set of neighborhood structures $N_l, l = 1, \dots, l_{max}$ that will be used in the local search; find an initial solution x ; choose a stopping condition;

Repeat the following sequence until the stopping condition is met:

- (1) Set $k \leftarrow 1$;
- (2) **Repeat** the following steps until $k = k_{max}$:
 - (a) **Shaking.** Generate x' at random from the k^{th} neighborhood of x ($x' \in N_k(x)$);
 - (b) **Local search by VND.**
 - (b1) Set $l \leftarrow 1$;
 - (b2) **Repeat** the following steps until $l = l_{max}$:

Exploration of neighborhood. Find the best neighbor x'' of x' in $N_l(x')$;

Move or not. If $f(x'') < f(x')$ set $x' \leftarrow x''$ and $l \leftarrow 1$; otherwise set $l \leftarrow l + 1$;
 - (c) **Move or not.** If the local optimum x'' is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with N_1 ($k \leftarrow 1$); otherwise, set ($k \leftarrow k + 1$);

Figure 4.3: VNS Algorithm with VND [99]

Based on Figure 4.3, Figure 4.4 represents a flowchart for the proposed VN

Figure 4.4: Implemented VNS Flowchart

The proposed VNS algorithm is implemented in the following phases:

4.1.1. Initialization. In this phase, the sets of neighbourhood structures that will be used in the shaking and local search phases are specified. Also, an initial solution is generated at random and a feasibility check is carried out as an acceptance criteria for the generated solution. The below subsections detail the initialization phase.

4.1.1.1. Neighborhood Structures. Neighborhood structures for both the shaking phase and the local search phase are selected with $k_{max} = 4$ and $l_{max} = 2$. For the shaking phase, the neighborhood structures are as follows:

- Neighborhood N_1 ($k = 1$): Swapping two different drivers between two different lockers-customers sequences.
- Neighborhood N_2 ($k = 2$): Swapping customers of the same locker between two different drivers.
- Neighborhood N_3 ($k = 3$): Swapping customers of the same locker between two different drivers.

Neighborhood N_4 ($k = 4$): Swapping any two customers and inserting customer locker before any of the two customers if necessary.

For the local search phase, the neighborhood structures are as follows:

- Neighborhood N_1 ($l = 1$): Swapping two customers within the same driver and from the same locker.
- Neighborhood N_2 ($l = 2$): Swapping lockers' order with their customers' sequence within the same driver.

4.1.1.2. Initial Solution. In order to generate an initial solution, a random procedure is used to build solution nodes and a feasibility check is then conducted to accept only feasible solutions. At the beginning, a driver is selected at random and then, starting from driver's current location, a locker is selected at random as a second node since drivers should visit lockers before going to customers' nodes. Afterwards, more lockers nodes, that have remaining unvisited customers, and non-visited customers nodes are added randomly and progressively taking into consideration drivers'

capacity. When the capacity is reached and no more customers can be added, driver's final destination node is added and a new driver is selected at random. The new driver route is constructed in the same way and the process repeats until all customers are visited. Each customer node should be visited exactly once after its locker node is visited and each driver can go to the same locker only once. Figure 4.6 illustrates the construction of the solution. The process iterates many times and a feasibility check is performed each time to accept the feasible solution. When a feasible solution is generated, the process stops and a cost calculation is performed afterwards to find the value of the total delivery cost of the initial solution.

Figure 4.5 shows the structure of the initial solution. In the figure, 10 customers' requests " C_1 to C_{10} " are to be satisfied from 2 lockers " L_1 and L_2 " using 3 drivers whose current locations are denoted by " D_1 to D_3 " and whose final destinations are denoted by " F_1 to F_3 ". The first 5 customers are satisfied from locker 1 whereas the second 5 are satisfied from locker 2. From hereinafter, same structure will be used to explain the phases of our implemented VNS.

Figure 4.5: Initial Solution

Input: Drivers Nodes (drivers), Drivers Capacity (drivers_cap), Lockers Nodes (lockers), Customers Nodes (customers), Drivers Final Destination (drivers_final_dest), Total Number of Nodes (total_nodes)

Output: Initial Solution (init_sol), Solution Length (sol_length)

```

for (i=1 to total_nodes*2) do
    if (i=1)
        selected driver= choose random driver
        init_sol[i] = selected driver
        sol_length++
    if (i=2)
        selected locker= choose random locker
        init_sol[i] = selected locker
        sol_length++
    if (i >= 3)
        if (selected driver capacity >= drivers_cap[selected driver])
            init_sol[i] = drivers_final_dest[selected driver]
            sol_length++
            i++
            selected driver= choose random driver
            for (j=1 to i) do
                if (init_sol[j] = selected driver)
                    choose another driver
            end for
            init_sol[i] = selected driver
            sol_length++
            i++
            selected locker= choose random locker
            if (selected locker has no remaining customers)
                choose another locker
            else
                init_sol[i] = selected locker
                sol_length++
                i++
            else
                while (true)
                    is_customer = false, is_locker=false
                    Step 1: selected node= choose random locker or customer
                    if (selected node is locker)
                        if (selected node is not visited by the selected driver)
                            init_sol[i]= selected node
                            is_locker=true
                            sol_length++
                        else
                            go to Step 1
                        if (selected node is customer)
                            if (selected node is not visited before & selected
                                node locker is visited)
                                init_sol[i]= selected node
                                is_customer=true
                                customers_visited++
                                sol_length++
                            else
                                go to Step 1
                            if (is_customer = true or is_locker = true) break;
                        end while
                    if (customers_visited = customers)
                        i++
                        init_sol[i] = drivers_final_dest[selected driver]
                        sol_length++
                        break
                end for

```

Figure 4.6: Construction of Initial Solution

4.1.1.3. Feasibility Check. Several initial solutions are generated at the beginning but the algorithm accepts only the feasible ones. The feasibility check is based on drivers' maximum available delivery time and customers' delivery time window. For each driver, the total delivery time; that is the time until the driver reaches his final destination, should be less than the total available delivery time for that driver. Total delivery time can be calculated from the total distance that the driver travelled and the time needed to travel one distance unit. Similarly, for all customers, the arrival time at each node should be no less than the beginning of customer's delivery window and no more than the ending of the customers' delivery window. If any solution violates these time constraints, the solution is considered as not feasible. Figure 4.7 illustrates initial solutions feasibility check.

```

Input: Initial Solution (init_sol), Solution Length (sol_length), Drivers Maximum Time (drivers_maxtime), Arrival Time (arrival_time), Customer Begin Time (begin_time), Customer End Time (end_time), Drivers Final Destination (drivers_final_dest)
Output: Solution Feasibility

for (i=1 to sol_length) do
    drivers_ontime = false
    if (init_sol[i] is driver)
        selected driver = init_sol[i]
        driver_index = i
        for (j=driver_index to drivers_final_dest[selected driver]) do
            calculate arrival time to each node
        end for
        if (selected driver time to reach final destination <= drivers_maxtime [selected driver])
            drivers_ontime= true
            customers_ontime=false
            for (k=driver_index to drivers_final_dest[selected driver]) do
                selected node = init_sol[k]
                if (selected node is customer)
                    if (arrival_time[selected node] > begin_time[selected node] & arrival_time[selected node] < end_time[selected node])
                        customers_ontime = true;
                    end if
                end for
            end for
        end for
    if (drivers_ontime = true & customers_ontime = true)
        init_sol is feasible
    else
        init_sol is not feasible
    end if
end for

```

Figure 4.7: Feasibility Check

4.1.1.4. Total Delivery Cost Calculation. For each generated solution, the algorithm calculates the total travel cost which is the sum of the total fixed cost and the total variable cost. When a new driver is selected, a fixed charge is added to the delivery cost while a variable charge is added for any unit of distance travelled between every two nodes. The algorithm then chooses the solution with the minimum total cost for the next phases. Figure 4.8 illustrates total travel cost calculations.

```

Input: Solution (sol), Solution Length (sol_length), Distance Array (distance), Drivers Final
Destination (drivers_final_dest), Fixed Cost, Variable Cost
Output: Total Cost

for (i=1 to sol_length) do
    if (init_sol[i] is driver)
        driver_index = i
        total cost = total cost + fixed cost
        for (j=driver_index to drivers_final_dest[selected driver]) do
            selected node 1 = init_sol[j]
            selected node 2 = init_sol[j+1]
            total cost = total cost + (variable cost * (distance[node1] [node2]))
        end for
    end for

```

Figure 4.8: Total Delivery Cost Calculations

4.1.2. Shaking Phase. In this phase, several random moves from one neighbor structure to other neighbor structure are performed to escape from local search local optima. The four-implemented different neighborhood structures are as follows:

4.1.2.1. Neighborhood Structure N_1 ($k = 1$). The first neighbourhood structure in the shaking phase is to swap two different drivers while keeping the rest of the sequence as it is. Since there are capacity constraints, not all drivers can be swapped and hence the number of customers visited by each driver is taken into consideration. The algorithm begins with the first driver in the solution, and then search for the first next driver with a total number of visited customers less than or equal to the driver's capacity. If there is no capacity violation, the drivers are swapped and the solution feasibility is checked. For the feasible solutions, the total travel cost is calculated to be compared later and the process repeats with another selected driver. When all swapping possibilities are explored, the algorithm stops and the feasible solution with the

minimum cost is used in the next local search phase ($l = 1$). Figure 4.9 presents a simple example.

Before the Swap:

After the Swap:

Figure 4.9: Shaking Phase ($k = 1$) Example

The process begins with the first driver in the sequence D_1 ; the capacity of the driver is 4 while the capacity of the two other drivers is 3 and hence this driver cannot be swapped with any other driver. Since the second driver in the sequence is D_3 with a capacity of 3, this driver can be swapped with driver D_2 as the total number of visited customers by D_2 is also 3 and hence D_2 and D_3 are swapped. The process stops here since there is no other swapping possibilities. Figure 4.10 details this first stage in the shaking phase.

Input: Initial Solution (init_sol), Solution Length (sol_length), Drivers Nodes (drivers), Drivers Capacity (drivers_cap), Drivers Final Destination (drivers_final_dest), Distance Array

Output: Shaking Solution (shake_sol), Shaking Solutions (shake_sols), Shaking Solutions Arrival Time (shake_sols_arrival), Shaking Solutions Cost (shake_sols_cost), Number of Shaking Solutions (num_shake_sols)

```

shake_sol = init_sol
num_shake_sols = 1
for (i=1 to drivers) do
    driver1 = drivers[i]
    driver1_index = i
    for (j=1 to sol_length) do
        if (shake_sol[j] = driver1)
            for (k= driver1_index to drivers_final_dest[driver1]) do
                count the number of customers visited by driver1
            end for
        end for
    end for
    for (m=1 to sol_length) do
        if (shake_sol[m] is driver & not equal to driver1)
            driver2 = L1_sol[m]
            driver2_index = m
            for (L=driver2_index to sol_length) do
                count the number of customers visited by driver2
            end for
            if (customers visited by driver2 <= drivers_cap[driver1] && customers visited
by driver1 <= drivers_cap[driver2])
                //swap the two drivers
                shake_sol[driver1_index] = driver2
                shake_sol[drivers_final_dest[driver1]] = drivers_final_dest[driver2]
                shake_sol[driver2_index] = driver1
                shake_sol[drivers_final_dest[driver2]] = drivers_final_dest[driver1]

                shake_sols[num_shake_sols] = shake_sol
                calculate arrival time for this solution using Distance Array
                add arrival time to shake_sols_arrival[num_shake_sols]
                calculate the cost of delivery for this solution
                add cost to shake_sols_cost[num_shake_sols]
                check feasibility
                num_shake_sols ++
                break
            end for
        end for
    end for
    keep the solution with the minimum cost as the shaking solution
    set L = 1

```

Figure 4.10: Shaking Phase ($k = 1$)

4.1.2.2. Neighborhood Structure N_2 ($k = 2$). This neighbourhood swaps the customers of the same locker between two different drivers when the locker is visited by more than one driver. The algorithm randomly selects two drivers that has at least

one common locker in their route and then starts to swap customer by customer until all locker's customers are swapped. The algorithm then moves to the next common locker and repeats the same process for all common lockers between the two selected drivers. Afterwards, another two random drivers are selected and the total number of iterations is half the total number of drivers. As in the shaking phase, the feasibility of the generated solution after every swap is checked and feasible solutions are kept for further consideration. Figure 4.11 presents an example that clarifies the process.

Before the Swap:

After the Swap:

Figure 4.11: Shaking Phase ($k = 2$) Example

At the beginning, drivers D_1 and D_2 are selected since they have two lockers in common. The first locker in D_1 route is L_2 and hence all customers that belongs to L_2 will be swapped one at a time with the customers that belongs to L_2 in D_2 route; only C_7 in this case. After that, the same process is repeated for customers of the second locker which is L_1 in this case. Thus, C_3 and C_5 from D_1 route will be swapped one at a time with C_2 and C_4 from D_2 route. After each swap, the total travel distance of the new solution is calculated and the minimum of $N_2(x)$ solutions will be used in the next steps of the VNS. Subroutine 4.12 presents this stage of the VND.

Input: Initial Solution (init_sol), Solution Length (sol_length), Drivers Nodes (drivers), Drivers Capacity (drivers_cap), Drivers Final Destination (drivers_final_dest), Distance Array, Customer Locker (cust_lock)

Output: Shaking Solution (shake_sol), Shaking Solutions (shake_sols), Shaking Solutions Arrival Time (shake_sols_arrival), Shaking Solutions Cost (shake_sols_cost), Number of Shaking Solutions (num_shake_sols)

```

shake_sol = init_sol
num_shake_sols = 1
for (i=1 to drivers/2) do
    driver1 = select random driver
    driver2 = select random driver & not equal to driver1
    for (j=1 to sol_length) do
        if (shake_sol[j] = driver1)
            driver1_index = j
        else if (shake_sol[j] = driver2)
            driver2_index = j
        end if
    end for
    for (k = driver1_index to drivers_final_dest[driver1]) do
        if (shake_sol[k] is locker)
            selected_locker = shake_sol[k]
            n = driver2_index
            for (m = k+1 to drivers_final_dest[driver1]) do
                if (shake_sol[m] is customer & cust_lock[shake_sol[m]] = selected_locker)
                    driver1_customer = shake_sol[m]
                    for (n to drivers_final_dest[driver2]) do
                        if (shake_sol[n] is customer & cust_lock[shake_sol[n]] = selected_locker)
                            driver2_customer = shake_sol[n]
                            // swap the two customers
                            shake_sol[m] = driver2_customer
                            shake_sol[n] = driver1_customer
                            shake_sols[num_shake_sols] = shake_sol
                            calculate arrival time for this solution using Distance Array
                            add arrival time to shake_sols_arrival[num_shake_sols]
                            calculate the cost of delivery for this solution
                            add cost to shake_sols_cost[num_shake_sols]
                            check feasibility
                            num_shake_sols ++
                            n++
                            break
                        end if
                    end for
                end for
            end for
        end for
    end for
    keep the solution with the minimum cost as the shaking solution
    set L = 1

```

Figure 4.12: Shaking Phase (k = 2)

4.1.2.3. Neighborhood Structure N_3 ($k = 3$). This neighbourhood structure is similar to N_2 ($k = 2$) except that the swapping is not performed for all common lockers. After selecting two drivers that has at least one common locker in their route, the algorithm randomly selects half of the lockers to swap their customers. Afterwards, another two random drivers are selected and the process repeats until the total number of iterations is half the total number of drivers. Figure 4.13 shows this last structure in the local search phase.

```

Input: Initial Solution (init_sol), Solution Length (sol_length), Drivers Nodes (drivers), Drivers
Capacity (drivers_cap), Drivers Final Destination (drivers_final_dest), Distance Array, Customer
Locker (cust_lock), Lockers Nodes (lockers)
Output: Shaking Solution (shake_sol), Shaking Solutions (shake_sols), Shaking Solutions Arrival
Time (shake_sols_arrival), Shaking Solutions Cost (shake_sols_cost), Number of Shaking
Solutions (num_shake_sols)

shake_sol = init_sol
num_shake_sols = 1
for (i=1 to drivers/2) do
    while(true)
        driver_chosen = false, locker_chosen = false
        driver1 = select random driver
        driver2 = select random driver & not equal to driver1
        driver_chosen = true
        for (j=1 to sol_length) do
            if (shake_sol[j] = driver1)
                driver1_index = j
            else if (shake_sol[j] = driver2)
                driver2_index = j
            end if
        end for
        for (k = 1 to lockers/2) do
            selected_locker = select random locker that is visited by driver1 and driver2
            locker_chosen = true
            n = driver2_locker_index
            for (m = driver1_locker_index to drivers_final_dest[driver1]) do
                if (shake_sol[m] is customer & cust_lock[shake_sol[m]] = selected_locker)
                    driver1_customer = shake_sol[m]
                    for (n to drivers_final_dest[driver2]) do
                        if (shake_sol[n] is customer & cust_lock[shake_sol[n]] = selected_locker)
                            driver2_customer = shake_sol[n]
                            // swap the two customers
                            shake_sol[m] = driver2_customer
                            shake_sol[n] = driver1_customer
                            shake_sols[num_shake_sols] = shake_sol
                            calculate arrival time for this solution using Distance Array
                            add arrival time to shake_sols_arrival[num_shake_sols]
                            calculate the cost of delivery for this solution
                            add cost to shake_sols_cost[num_shake_sols]
                            check feasibility
                            num_shake_sols ++
                            n++
                            break
                        end if
                    end for
                end for
            end for
            if (driver_chosen = true & locker_chosen = true) break
        end while
    end for

    keep the solution with the minimum cost as the shaking solution
    set L = 1

```

Figure 4.13: Shaking Phase ($k = 3$)

4.1.2.4. Neighborhood Structure N_4 ($k = 4$). The last neighbourhood structure is a combination of swapping and insertion. In this step, two customers from any driver and any locker are selected at random to be swapped. After the swap, the algorithm assures that the constraint of visiting lockers before visiting their customers is met for each customer. Thus, for each one of the swapped customers, the algorithm checks whether the locker is visited before customer node. In case the locker is visited before, neither shifting nor insertion is performed. In case the locker is in the driver route but after the customer, the locker node is shifted back to be visited before the customer. If the locker doesn't exist in the driver route, it is inserted directly before the customer node. After any shifting and any insertion, the feasibility is checked and in case the solution is feasible the total travel cost is calculated and the solution is saved for further comparison. Figure 4.14 illustrates shifting and insertion of two lockers.

Before the Swap:

After the Swap with shifting and insertion:

Figure 4.14: Shaking Phase ($k = 4$) Example

At the beginning, two customers are selected at random; we consider C_7 and C_1 . After swapping the two customers, Locker L_1 will be visited after its customer C_1 and hence its shifted one node before. At the same time, customer C_7 will be added to driver 2 route which does not include locker L_2 and hence this locker is inserted in the node

before C_7 . The algorithm then chooses other two different customers at random and the process repeats until the total number of iterations is half the number of customers. At the end, the algorithm chooses the feasible solution with the minimum cost among all generated solutions $N_4(x)$ to be used in the local search phase ($l = 1$). Figure 4.15 details this neighbourhood structure.

```

Input: Initial Solution (init_sol), Solution Length (sol_length), Customers Nodes (customers),
Customer Locker (cust_lock), Distance Array
Output: Shaking Solution (shake_sol), Shaking Solutions (shake_sols), Shaking Solutions Arrival
Time (shake_sols_arrival), Shaking Solutions Cost (shake_sols_cost), Number of Shaking
Solutions (num_shake_sols)

num_shake_sols = 1
shake_sol = init_sol
for (i=1 to customers/2) do
    customer1_index = select random index that contain customer
    customer1 = shake_sol[customer1_index]
    customer2_index = select random index that contain customer and not equal to
customer1_index
    customer2 = shake_sol[customer2_index]

    customer1_driver_index = index of driver visited customer1
    customer2_driver_index = index of driver visited customer2

    //swap customers
    shake_sol[customer1_index] = customer2
    shake_sol[customer2_index] = customer1
    customer1 = customer2 & customer2 = customer1
    customer1_locker = cust_lock[customer1]
    customer2_locker = cust_lock[customer2]

    if (customer1_locker is visited before customer1 & customer2_locker is visited before
customer2)
        shake_sols[num_shake_sols] = shake_sol
    else if (customer1_locker is visited after customer1 or customer2_locker is visited after
customer2)
        shift the locker before the customer
        shake_sols[num_shake_sols] = shake_sol
    else if (customer1_locker is not found or customer2_locker is not found)
        add a new node before the customer and insert the customer locker in that node
        increase the sol_length with the number of added noded
        shake_sols[num_shake_sols] = shake_sol
        calculate arrival time for this solution using Distance Array
        add arrival time to shake_sols_arrival[num_shake_sols]
        calculate the cost of delivery for this solution
        add cost to shake_sols_cost[num_shake_sols]
        check feasibility
        num_shake_sols ++
end for
keep the solution with the minimum cost as the shaking solution
set L = 1

```

Figure 4.15: Shaking Phase (k=4)

4.1.3. Local Search Phase by VND. For each generated shaking solution, the variable neighborhood search algorithm tries to improve the structure of the solution by applying several local search steps. The three-implemented different neighbourhood structures are as follows:

4.1.3.1. Neighborhood Structure N_1 ($l = 1$) . In this neighborhood, two different customers who are visited by the same driver and whose requests are fulfilled from the same locker are swapped. The algorithm starts with the first visited customer in the driver route, checks its locker and then search for another visited customer from the same locker in the driver's route. When a second customer is found, the swap is performed as shown in figure 4.16.

Before the Swap:

After the Swap:

Figure 4.16: Local Search Phase ($l = 1$) Example

The algorithm starts from driver 1, take the first visited customer C_7 and looks for the first customer in the subsequent sequence whose request is to be satisfied from locker 2 also. Since the first customer is C_{10} , the algorithm will swap those two customers and proceed to the next customer in the route which is C_3 . The process will repeat until all common customers are swapped. After each swap, the feasibility of the new solution is checked and if the solution is feasible, the total travel cost is calculated

and the solution is stored for further comparison. The algorithm then proceeds to the next customers' swap and the process iterates until all swapping possibilities for all drivers and lockers are generated. At the end, the algorithm chooses the minimum feasible solution among all generated shaking solutions $N_1(x)$ and use it in the next phase. Figure 4.17 details this first neighborhood structure.

```

Input: Shaking Solution (shake_sol), Solution Length (sol_length), Drivers Nodes (drivers),
Drivers Final Destination (drivers_final_dest), Customer Locker (cust_lock), Distance Array
Output: L1 Solution (L1_sol), VND Solutions (vnd_sols), VND Solutions Arrival Time
(vnd_sols_arrival), VND Solutions Cost (vnd_sols_cost), Number of VND Solutions
(num_vnd_sols)

L1_sol = shake_sol
num_vnd_sols = 1
for (i=1 to sol_length) do
    if (L1_sol[i] is driver)
        selected driver = L1_sol[i]
        for (j=i+1 to drivers_final_dest[selected driver]) do
            if (L1_sol[j] is locker)
                selected locker = L1_sol[j]
                for (k=j+1 to drivers_final_dest[selected driver]) do
                    swap = false
                    if (L1_sol[k] is customer & cust_lock[L1_sol[k]] = selected locker)
                        customer 1 = L1_sol[k]
                        for (m=k+1 to drivers_final_dest[selected driver]) do
                            if (L1_sol[m] is customer & cust_lock[L1_sol[m]] = selected locker)
                                customer 2 = L1_sol[m]
                                //swap the two customers
                                L1_sol[k] = customer 2
                                L1_sol[m] = customer 1
                                vnd_sols[num_vnd_sols] = L1_sol
                                calculate arrival time for this solution using Distance Array
                                add arrival time to vnd_sols_arrival[num_vnd_sols]
                                calculate the cost of delivery for this solution
                                add cost to vnd_sols_cost[num_vnd_sols]
                                check feasibility
                                num_vnd_sols++
                                swap = true
                                k = m+1
                                break
                            else
                                swap = false
                                m++
                        end for
                    if (swap = false)
                        j++
                        break
                    end for
                else
                    j++
                end for
            i = j+1
        end for

keep the solution with the minimum cost as L1 solution

```

Figure 4.17: Local Search Phase ($l = 1$)

4.1.3.2. Neighborhood Structure N_2 ($l = 2$). The second neighbourhood structure is to swap the order in which driver visits two different lockers and the sequence following each locker. The process begins with randomly selecting a driver who visits two or more lockers and then randomly selecting two of the visited lockers. After that, the two selected lockers along with customers' sequence that starts after each locker and ends at either another locker or a final destination are swapped. The algorithm then checks the swapped customers to ensure that each customer in the sequence is visited after visiting the locker in which the customer order exists. In case there is a violation for this constraint, the customer will be replaced by any other customer whose locker is visited before this location. If there exists no other customer that can be replaced, the swapping of the lockers is not performed and different two lockers are selected at random for the next iteration. The process stops when the total number of iterations is half the number of drivers. Figure 4.18 illustrates this process through a simple example.

Before the Swap:

After the Swap:

Figure 4.18: Local Search Phase ($l = 2$) Example

The process starts by randomly choosing a driver that goes to two or more lockers; in this case we consider driver 1. The first part from L_2 to C_9 is swapped with

the second part from L_1 to C_{10} . After the swap, customer C_{10} is visited before its locker L_2 in the new sequence and hence it should be replaced by another customer that has a request in the visited locker. Since customer C_5 belongs to locker L_1 , it can be shifted to C_{10} location. After each swap, the feasibility of the generated solution is checked and if the solution is feasible, the total travel cost is calculated and the solution is stored for further comparison. At the end, the algorithm chooses the minimum feasible solution among all generated shaking solutions $N_2(x)$ and use it in the next phase. Figure 4.19 details this second neighbourhood structure.

```

Input: Shaking Solution (shake_sol), Solution Length (sol_length), Drivers Nodes (drivers),
Drivers Capacity (drivers_cap), Drivers Final Destination (drivers_final_dest), Distance Array
Output: L2 Solution (L2_sol), VND Solutions (vnd_sols), VND Solutions Arrival Time
(vnd_sols_arrival), VND Solutions Cost (vnd_sols_cost), Number of VND Solutions
(num_vnd_sols)

num_vnd_sols = 1
for (i=1 to drivers/2) do
    selected driver = select random driver that visited at least two lockers
    if (no driver with at least two lockers)
        go to K+1
        break
    else
        for (j=1 to sol_length) do
            if (shake_sol[j] = selected driver)
                driver_index = j
                break
        end for
        locker1 = select random locker from lockers visited by the selected driver
        locker2 = select random locker from lockers visited by the selected driver & not equal to
        locker1
        for (k=driver_index to drivers_final_dest[selected driver]) do
            if (shake_sol[k] = locker1)
                locker1_index = k
            else if (shake_sol[k] = locker2)
                locker2_index = k
        end for
        count the number of customers visited from locker1
        count the number of customers visited from locker2
        small_index = index of the locker with the small index between the two lockers
        s_length = number of customers visited by the locker with small index
        large_index = index of the locker with the large index between the two lockers
        l_length = number of customers visited by the locker with large index

        copy the shake_sol to the L2_sol till small_index
        in small_index copy the shake_sol from the large_index till l_length to L2_sol
        copy from (small_index+s_length) in the shake_sol to the L2_sol till large_index
        in large_index copy the shake_sol from the small_index till s_length to L2_sol
        copy from (large_index+l_length) in the shake_sol to the L2_sol till sol_length

        check if all customers were visited after their lockers
        if (customer locker is visited after its customer)
            swap the customer with another customer whose locker is visited
            swap = true
        else
            swap = false
        if (swap = true)
            vnd_sols[num_vnd_sols] = L2_sol
        else if (swap = false)
            vnd_sols[num_vnd_sols] = shake_sol
            calculate arrival time for this solution using Distance Array
            add arrival time to vnd_sols_arrival[num_vnd_sols]
            calculate the cost of delivery for this solution
            add cost to vnd_sols_cost[num_vnd_sols]
            check feasibility
            num_vnd_sols ++
            break
    end for
keep the solution with the minimum cost as L2 solution

```

Figure 4.19: Local Search Phase (l=2)

4.1.4. Evaluation and Termination. After each local search step, the algorithm evaluates the solution to decide whether to move from the initial incumbent solution and whether the neighborhood structure should be changed. Initially, the local search solution is compared with the shaking solution, and if it is better than the shaking solution, it is set to be the new shaking solution and it is then compared to the initial solution. In case the new shaking solution is better than the initial solution, the initial solution is updated to be the shaking solution and the search is repeated with the first neighborhood ($k = 1$). If the new shaking solution is not better, the initial solution is kept as it is and the shaking is performed again with a move to the next neighborhood ($k = k + 1$). On the other hand, if the local search solution is not better than the shaking solution, the local search is repeated with the next neighbor ($l = l + 1$) and the process repeats until l_{max} . The process terminates when $k = k_{max}$ or when the number of iterations after improvement ($k = 1$) exceeds 10 iterations. Figure 4.20 details this phase.

```

Input: VND Solution (vnd_sol), Shaking Solution (shake_sol), Initial Solution (init_sol)
Output: Final Solution (final_sol) or Termination Decision

if (vnd_sol is not feasible or vnd_sol_cost > shake_sol_cost)
    if (L = 1)
        go to L = 2
    else if (L = 2)
        if (vnd_sol is not feasible or vnd_sol_cost > init_sol_cost)
            if (K = 1) go to K = 2
            else if (K = 2) go to K = 3
            else if (K = 3) go to K = 4
            else if (K = 4)
                final_sol = init_sol
                final_sol_arrival_time = init_sol_arrival_time
                final_sol_cost = init_sol_cost
                break
            else if (vnd_sol is feasible & vnd_sol_cost < init_sol_cost)
                init_sol = vnd_sol
                init_arrival_time = vnd_arrival_time
                init_sol_cost = vnd_sol_cost
                K = 1
        else if (vnd_sol is feasible or vnd_sol_cost < shake_sol_cost)
            shake_sol = vnd_sol
            shake_sol_arrival_time = vnd_sol_arrival_time
            shake_sol_cost = vnd_sol_cost
            if (shake_sol_cost < init_sol_cost)
                init_sol = shake_sol
                init_arrival_time = shake_arrival_time
                init_sol_cost = shake_sol_cost
                K = 1
            else if (shake_sol_cost > init_sol_cost)
                if (K = 1) go to K = 2
                else if (K = 2) go to K = 3
                else if (K = 3) go to K = 4
                else if (K = 4)
                    final_sol = init_sol
                    final_sol_arrival_time = init_sol_arrival_time
                    final_sol_cost = init_sol_cost
                    break

```

Figure 4.20: Solution Evaluation

The developed algorithm detailed above has been tested on several problems with different sizes to test the returned solution in terms of solution quality and computational time. In the next chapter, all the experiments performed are presented.

Chapter 5. Experimentation, Results and Analysis

In this chapter, we present the experimentations performed to validate the developed Variable Neighbourhood Search Algorithm. We also evaluate the performance of the proposed VNS by comparing the obtained solutions with the exact solutions provided by LingoTM. After that, several experiments were conducted to test the ability of the developed algorithm to solve large instances. At the end, a sensitivity analysis is performed to assess model's sensitivity to model inputs.

5.1. VNS Validation

For testing the solution returned by the VNS algorithm and evaluating how far it is from the exact solution given by Lingo in terms of solution quality and time, the same experiments performed before on different data sets from TSPLIB were conducted. The algorithm is coded using C++ programming language and all the experiments were conducted using Intel(R) Xeon(R) CPU @ 3.50GHz with 16 GB RAM machine under Windows 7 (64 bit). Table 5.1 illustrates the results obtained from finding a solution to the proposed model using the VNS algorithm for different number of drivers, number of lockers, number of customers and different total number of nodes.

Table 5.1: VNS Objective Value and Elapsed Time

Experiment No.	Problem	No. of Drivers	No. of Lockers	No. of Customers	Total No. of Nodes	Objective Value	Elapsed Time (sec)
1	gr17	3	2	9	17	10,118	0.499
2	gr21	4	3	10	21	12,052	2.281
3	gr24	4	3	13	24	17,496	4.080
4	bays29	4	3	18	29	21,748	4.902

Figure 5.1 illustrates the objective function values which are the best solutions provided by the algorithm. The x-axis presents the experiment number from Table 5.1 and the y-axis shows the total travel cost (the objective value). As expected, the total delivery cost increases with the increase in the number of customers.

Figure 5.1: Total Delivery Cost, VNS

Figure 5.2 shows the time taken by the algorithm to find the best solution for the different number of nodes.

Figure 5.2: Elapsed Time, VNS

The algorithm finds the solution very fast when the number of nodes is small and the time increase with the increase in the total number of nodes.

5.1.1. Comparison between Lingo™ and the Variable Neighborhood Search Algorithm. Figure 5.3 presents a comparison between the values obtained from the exact solution (Lingo) and the heuristic. For the first set, the algorithm returned the same solution provided by Lingo and for the other 3 runs the results remained very

close to the exact solution and this validates our proposed algorithm. The comparison couldn't be extended due to Lingo limitations.

Figure 5.3: VNS Percentage Deviation from Lingo™

Similarly, table 5.2 shows the time taken by the algorithm to find a solution. The algorithm is very fast compared to Lingo and it took only 4 seconds to find a near optimal solution for 29 nodes while Lingo took 33 hours to solve the same number of nodes.

Table 5.2: Elapsed Time, Lingo™ vs. VNS

Experiment No.	Lingo Elapse Time (sec)	VNS Elapsed Time (sec)
1	35.38	0.499
2	2998.29	2.281
3	2.281	4.080
4	3889.21	4.902

The comparison presented above validates our proposed algorithm in terms of solution quality and computational time. From this, it can be concluded that our proposed VNS can be used for extending the experiment for larger number of nodes for which exact solutions does not exist. In the next section, several experiments are presented to test the ability of the proposed algorithm to solve large instances.

5.2. VNS Experimentation

To test the effectiveness of the proposed algorithm and to observe how it behaves in term of computational time when increasing the total number of nodes, several experiments were conducted using different data sets from the online TSP Library “TSP Library”. Table 5.3 shows the different problems solved with the different number of drivers, lockers and customers. With the increase in problem size, not only the number of nodes increase but also the distances between each pair of nodes becomes very large and this is why the total delivery cost increased significantly. The elapsed time is taken as an average of 5 runs. All experiments are obtained using Intel(R) Core™ i7 6500U CPU @ 2.50GHz with 4 GB RAM machine under Windows 10 (64 bit).

Table 5.3: VNS Experimentation

Experiment No.	Problem	No. of Drivers	No. of Lockers	No. of Customers	Total No. of Nodes	Objective Value	Elapsed Time (sec)
1	gr48	6	4	32	48	73,492	5.39
2	eil76	8	6	54	76	246,224	7.093
3	rat99	10	7	72	99	584,172	11.24
4	pr136	13	10	100	136	7,124,766	2.28
5	Lin318	16	11	275	318	19,039,828	19.927

Figure 5.4 shows the increase in the total delivery cost with the increase in the total number of nodes whereas figure 5.5 illustrates the total computational time for solving up to 318 instance. The experiment number from table 5.5 is in the x-axis and the objective value and elapsed time are in the y-axis.

Figure 5.4: Total Delivery Cost VNS Experimentation

Figure 5.5: Elapsed Time, VNS Experimentation

It can be clearly noted that the algorithm provides a solution within few seconds regardless of the size of the problem. It took only an average of 20 seconds to return a solution for the largest problem of size 318. This instance includes 16 drivers, 11 lockers and 275 customers.

To end with, table 5.4 provides a summary for all experiments performed for the proposed VNS and the time taken by the algorithm to return a solution. Figure 5.6 concludes that the time needed by the algorithm to find a solution varies between different problems, not necessarily proportional to the problem size, yet it remained within 20 seconds for all the experiments runs.

Table 5.4: VNS Elapsed Time Summary

Experiment No.	Total No. of Nodes	Elapsed Time (sec)
1	17	0.499
2	21	2.281
3	24	4.080
4	29	4.902
5	48	5.39
6	76	7.093
7	99	11.24
8	136	2.28
9	318	19.927

Figure 5.6: VNS Elapsed Time Summary

5.3. Sensitivity Analysis

In this last section, a sensitivity analysis is performed to test the effect of input parameters on the objective function of interest; the total delivery cost. The parameters are changed one at a time with fixing all other parameters and the obtained results are compared with a reference case. Considering Lingo limitations in the number of nodes, the base case is taken as follows:

- Number of drivers: 3
- Number of lockers: 3
- Number of customers: 9

For all experiments, different sets from the online TSP library are used and the size of the matrix is varied by increasing or decreasing the number of nodes to match each trial.

5.3.1. Number of Lockers Variation. In these trials, the number of lockers is increased by one locker each time in order to study the effect of these variations on the total delivery cost. Table 5.5 details the experiment.

Table 5.5: Number of Lockers Variation I

Trial	Number of Drivers	Total Capacity	Number of Lockers	Number of Customers	Total Cost	Elapsed Time (sec)
1	3	12	1	9	12,458	4.45
2	3	12	2	9	14,666	17.55
3	3	12	3	9	17,522	4.59
4	3	12	4	9	17,570	16.75
5	3	12	5	9	18,238	18.93

As expected, figure 5.7 shows the increase in the delivery cost with increasing the number of lockers.

Figure 5.7: Number of Lockers Variation I

However, it has been noted that the delivery cost is not only sensitive to the increase in the number of lockers but also to customers' allocation to lockers and thus the above conclusion is not always valid. Table 5.6 shows the difference between customers' allocation in the two experiments while table 5.7 and figure 5.8 illustrates the second experiment.

Table 5.6: Customers Allocation to Lockers

Trial No.	3		4		5	
Experiment	I	II	I	II	I	II
L1	C1	C1, C2, C3	C1, C2, C4	C4, C5	C1, C2	C3, C4
L2	C2, C3	C4, C5, C6	C5, C8	C1, C2, C3	C3, C4	C1, C2
L3	C4 – C9	C7, C8, C9	C7	C7, C8, C9	C5, C6	C7, C8
L4	N/A	N/A	C6, C9	C6	C7	C5, C6
L5	N/A	N/A	N/A	N/A	C8, C9	C9

Table 5.7: Number of Lockers Variation II

Trial	Number of Drivers	Total Capacity	Number of Lockers	Number of Customers	Total Cost	Elapsed Time (sec)
1	3	12	1	9	12,458	4.45
2	3	12	2	9	14,666	17.55
3	3	12	3	9	14,470	13.40
4	3	12	4	9	14,446	15.87
5	3	12	5	9	14,118	19.10

Figure 5.8: Number of Lockers Variation II

From the details above, the conclusion that can be drawn about the correlation between the increase in the number of lockers and the increase in the total delivery cost is that the increase or decrease in the total costs is sensitive to the allocation of customers to lockers.

5.3.2. Number of Customers Variation. In these trials, the number of customers is increased by two customers each time in order to study the effect of these variations on the total delivery cost. Table 5.8 illustrates the trials performed.

Table 5.8: Number of Customers Variation

Trial	Number of Drivers	Total Capacity	Number of Lockers	Number of Customers	Total Cost	Elapsed Time (sec)
1	3	15	3	5	3,362	3.57
2	3	15	3	7	3,782	3.88
3	3	15	3	9	5,126	8.2
4	3	15	3	11	5,868	34.36
5	3	15	3	13	6,154	181.7

Intuitively, the total delivery cost increases with the increase in the number of customers. This is due to either the increase in the fixed cost that results from using more drivers to cover all customers if the capacity or available time of the existing drivers is not enough or the increase in the variable cost resulting from the additional distance travelled to cover more customers. Figure 5.9 shows this correlation.

Figure 5.9: Number of Customers Variation

5.3.3. Number of Drivers Variation. In these trials, the number of drivers is varied while keeping all other parameters fixed including the total available capacity. At the first time, the total capacity is distributed evenly between all drivers as shown in table 5.9 while in the second trials the distribution was random as presented in table 5.10.

Table 5.9: Number of Drivers Variation I

Trial	Number of Drivers	Capacity Distribution	Number of Lockers	Number of Customers	Total Cost	Elapsed Time (sec)
1	1	12	3	9	8,538	1.61
2	2	6, 6	3	9	8,280	8.95
3	3	4, 4, 4	3	9	9,406	16.07
4	4	3, 3, 3, 3	3	9	8,478	22.88
5	6	2, 2, 2, 2, 2, 2	3	9	13,406	355.17

Table 5.10: Number of Drivers Variation II

Trial	Number of Drivers	Capacity Distribution	Number of Lockers	Number of Customers	Total Cost	Elapsed Time (sec)
1	1	12	3	9	8,538	1.61
2	2	4, 8	3	9	8,824	8.33
3	3	4, 3, 5	3	9	8,516	16.65
4	4	5, 2, 3, 2	3	9	8,594	19.72
5	6	2, 3, 1, 3, 1, 2	3	9	11,440	66.16

Figure 5.10 and 5.11 shows that the total cost can either increase or decrease with the increase in the total number of drivers used. Although the total capacity is fixed, the distribution of capacity among drivers has an unpredictable effect on the total cost. This observation is valid when distributing the capacity either equally or randomly among the drivers. Moreover, in case the total capacity is randomly distributed, the variation in the cost also depends on which driver is assigned what capacity. Thus, the relationship between the number of drivers and the total delivery cost is not conclusive and depends on a confluence of factors; number of drivers, distribution of capacity and definitely drivers' and customers' time windows.

Figure 5.10: Number of Drivers Variation I

Figure 5.11: Number of Drivers Variation II

5.3.4. Drivers Capacity Variation. In this part, the total available capacity is varied by increasing the capacity of one driver while keeping the capacities of the other drivers fixed. Table 5.11 illustrates the several trials performed.

Table 5.11: Drivers Capacity Variation

Trial	Number of Drivers	Total Capacity	Number of Lockers	Number of Customers	Total Cost	Elapsed Time (sec)
1	3	10	3	9	10,050	22.39
2	3	11	3	9	9,810	20.77
3	3	12	3	9	8,920	16.4
4	3	13	3	9	8,168	11.69
5	3	14	3	9	8,168	14.04

Figure 5.12 depicts that the total delivery cost either decrease or remain constant with the increase in the drivers' capacity. This can be because either less drivers are used, and this decrease the total fixed cost, or the same number of drivers is used if the driver with the increased capacity cannot be used to deliver more requests due to delivery time windows restrictions. Both driver's total available time and customers' time windows will have an impact on the correlation between drivers' capacity and total delivery cost.

Figure 5.12: Drivers Capacity Variation

5.3.5. Fixed Cost to Variable Cost Ratio Variation. In these trials, we fixed the variable charge and varied the fixed charge to observe the effect of cost ratio on the total delivery cost. Table 5.12 shows the details.

Table 5.12: Cost Ratio Variation

Trial	Fixed Charge	Variable Charge	Cost Ratio	Total Cost	Elapsed Time (sec)	Number of Drivers Used
1	50	10	5	20,030	11.45	3
2	150	10	15	20,230	14.02	2
3	300	10	30	20,530	11.78	2
4	450	10	45	20,830	10.53	1
5	600	10	60	21,130	12.51	1

Table 5.12 shows that as the fixed charge to variable charge ration increases, the model tends to use less drivers but if the total capacity of the drivers and their total available time permit.

Chapter 6. Conclusion and Future Work

Last mile delivery has recently received considerable attention with the growth of e-commerce and the increase in customers' demand for ever-faster delivery. Technological innovations are offering plenty of options for logistics operators to deliver parcels and products to customers at the last mile. In this thesis, we studied the use of crowdsourcing as a promising solution for this last mile delivery problem. The problem is formulated as an Integer Linear Programming (ILP) problem in two different ways with the objective of minimizing total delivery cost. The modeling environment considered the crowd-workers, the lockers and the customers and the two models are subject to different constraints that restricts drivers' capacities and availability time as well as customers' delivery time windows. The outcome of the two models is the optimal assignment of requests to drivers as well as optimal routing for each driver. To validate the proposed models, Lingo™ was used to give exact solutions to different data sets from the online TSP library. Both models returned similar results for the objective function, yet one of the models proved its superiority to the other in terms of Ling run time. However, Lingo™ proved its inefficiency when the problem size increased as it took an extremely long time to provide a solution for considerably small size problems. Thus, a Variable Neighborhood Search heuristic is developed to give near optimal solutions in reasonable amount of time.

Variable Neighborhood Search Algorithm (VNS) is an effective method for getting approximate solutions for large single-objective optimization problems including scheduling problems and TSP. In our work, we implemented a VNS that uses Variable Neighborhood Descent (VND) in the local search phase with 4 different neighborhood structures in the shaking phase and 2 different neighborhood structures in the local search phase. To test its effectiveness, the proposed algorithm has been coded using C++ programming language and the same Lingo™ experiments were conducted to compare the returned solutions. For small size problems, the algorithm provided the same exact solution in less than one second whereas it returned solutions that are very close to the exact solution in few seconds when the problem size increased. To verify the algorithm expandability, more experiments were performed on larger sets

from the online TSP library (up to 318 nodes) and the algorithm returned solutions within less than 20 seconds.

At last, a sensitivity analysis was conducted to assess the effect of changing input parameters on the total delivery costs. The total number of lockers, total number of customers, total number of drivers and drivers' capacity were changed one at a time and the effect of parameters' variation on the total delivery cost was observed. Although there was a clear correlation between the change in the parameter and the change in the total delivery cost for some parameters, some other experiments were inconclusive.

As a future work, one area of interest will be to consider dynamic environment in which requests are not fully known in advance and drivers can join and disjoin at any point in time. The proposed model can be modified to consider request re-assignment and drivers' re-routing. Another extension could be to use crowdsourcing for both pickup and delivery where crowd-workers can pick some requests on their way and deliver them to customers rather than considering delivery only. Moreover, the neighborhood structures used in the shaking phase and local search phase of the VNS algorithm can be improved in order to enhance the performance of the algorithm.

References

- [1] W. Narwhal, "*The Evolution of E-Commerce*," 21 Aug 2015. [Online]. Available: <https://www.tributemedia.com/blog/the-evolution-of-ecommerce>.
- [2] "*The impact of online shopping on retail property*," 15 Nov 2013. [Online]. Available: <https://www.ampcapital.com/site-assets/articles/insights-papers/2013/2013-11/the-impact-of-online-shopping-on-retail-property>.
- [3] S. Mohapatra, *E-Commerce Strategy: Text and Cases*, New York: Springer US, 2013.
- [4] Z. Qin, Ed., *Introduction to E-commerce*, Berlin: Springer, 2009.
- [5] M. Alshehri and S. Drew, "E-Government Fundamentals," in *Proceedings of the IADIS International Conference ICT, Society and Human Beings* 2010, Freiburg, 2010.
- [6] C. Dan, "Consumer-To-Consumer (C2C) Electronic Commerce: The Recent Picture," *International Journal of Networks and Communications*, vol. 4, no. 2, pp. 29-32, 2014.
- [7] K. Prickett, "*What Is The Last Mile For Ecommerce*," 29 July 2014. [Online]. Available: <https://prospress.com/what-is-the-last-mile/>.
- [8] C. Chen and S. Pan, "Using the Crowd of Taxis to Last Mile Delivery in E-Commerce: a methodological research," in *Service Orientation in Holonic and Multi-Agent Manufacturing*, Switzerland, Springer International Publishing, 2016, pp. 61-70.
- [9] L. Song, T. Cherrett, F. McLeod and W. Guan, "Addressing the Last Mile Problem Transport Impacts of Collection and Delivery Points," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2097, pp. 9-18, 2009.
- [10] M. Dell'Amico and S. Hadjidimitriou, "Innovative Logistics Model and Containers Solution for Efficient Last Mile Delivery," *Procedia-Social and Behavioral Sciences*, vol. 48, pp. 1505-1514, 2012.
- [11] K. Boyer, A. Prud'homme and W. Chung, "The Last Mile Challenge: Evaluating The Effects Of Customer Density And Delivery Window Patterns," *Journal of Business Logistics*, vol. 30, no. 1, pp. 185-201, 2009.

- [12] P. Liakos and A. Delis, "An Interactive Freight-Pooling Service for Efficient Last-Mile Delivery," in *16th IEEE International Conference on Mobile Data Management*, Pittsburgh, 2015.
- [13] R. Marsh, "Amazon Drone Patent Application Imagines Delivery That Comes to you With One Click," 12 May 2015. [Online]. Available: <http://edition.cnn.com/2015/05/12/politics/amazon-patent-drone-delivery/>.
- [14] J. O'Reilly, "Same Day Delivery: The Amazing Race," Feb 2014. [Online]. Available: <http://www.inboundlogistics.com/cms/article/same-day-delivery-the-amazing-race/>.
- [15] J. Howe, "The Rise of Crowdsourcing," 6 January 2006. [Online]. Available: <https://www.wired.com/2006/06/crowds/>.
- [16] F. J. Garrigos-Simon, I. Gil-Pechuán and S. Estelles-Miguel, Eds., *Advances in Crowdsourcing*, Switzerland: Springer International Publishing, 2015.
- [17] V. Frehe, J. Mehmman and F. Teuteberg, "Understanding and Assessing Crowd Logistics Business Models-Using Everyday People for Last Mile Delivery," *Journal of Business & Industrial Marketing*, vol. 32, no. 1, pp. 75-97, 2017.
- [18] J. Howe, "Crowdsourcing: a Definition.," June 2006. [Online]. Available: <http://crowdsourcing.typepad.com/cs/2006/06/>.
- [19] D. Brabham, *Crowdsourcing*, Cambridge: MIT Press, 2013.
- [20] W. Kersten, T. Blecker and C. M. Ringle, "Innovations and Strategies for Logistics and Supply Chains," in *Proceedings of the Hamburg International Conference of Logistics*, Hamburg, 2015.
- [21] "Amazon Tries Out Taxi Deliveries In California Cities, WSJ Reports," 5 Nov 2015. [Online]. Available: <http://www.reuters.com/article/us-amazon-com-taxis-idUSKBN0IP2EM20141105>.
- [22] D. Reilly, "Amazon Ponders Crowdsourcing Deliveries With 'On My Way' Program," 16 Jun 2015. [Online]. Available: <http://fortune.com/2015/06/16/amazon-crowd-source/>.

- [23] A. Barr and J. Wohl, "Exclusive: Wal-Mart May Get Customers to Deliver Packages to Online Buyers," 28 March 2013. [Online]. Available: <http://www.reuters.com/article/us-retail-walmart-delivery-idUSBRE92R03820130328>.
- [24] K. Braekers, K. Ramaekers and I. V. Nieuwenhuyse, "The Vehicle Routing Problem: State Of The Art Classification And Review," *Computers & Industrial Engineering*, vol. 99, pp. 300-313, 2016.
- [25] Y. Xiao, Q. Zhao, I. Kaku and Y. Xu, "Development of a Fuel Consumption Optimization Model for The Capacitated Vehicle Routing Problem," *Computers & Operations Research*, vol. 39, no. 7, pp. 1419-1431, 2012.
- [26] M. Sedighpour, V. Ahmadi, M. Yousefikhoshbakht, F. Didehvar and F. Rahmati, "Solving the Open Vehicle Routing Problem by a Hybrid Ant Colony Optimization," *Kuwait Journal of Science*, vol. 3, p. 41, 2014.
- [27] O. Petrovic, M. J. Harnisch and T. Puchleitner, "Opportunities of Mobile Communication Systems for Applications In Last-Mile Logistics," in *International Conference on Advanced Logistics and Transport*, Sousse, 2013.
- [28] R. Matai, S. P. Singh and M. L. Mittal, "Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches," 30 Dec 2015. [Online]. Available: <https://www.intechopen.com/books/traveling-salesman-problem-theory-and-applications/traveling-salesman-problem-an-overview-of-applications-formulations-and-solution-approaches>.
- [29] J. V. Duin, W. D. Goffau, B. Wiegman, L. Tavasszy and M. Saes, "Improving Home Delivery Efficiency by Using Principles of Address Intelligence For B2C Deliveries," *Transportation Research Procedia*, vol. 12, pp. 14-25, 2016.
- [30] S. Iwan, K. Kijewska and J. Lemke, "Analysis of Parcel Lockers' Efficiency as The Last Mile Delivery Solution—The Results of the Research in Poland," *Transportation Research Procedia*, vol. 12, pp. 644-655, 2016.

- [31] E. Morganti, L. Dablanc and F. Fortin, "Final Deliveries for Online Shopping: the Deployment of Pickup Point Networks in Urban and Suburban Areas," *Research in Transportation Business & Management*, vol. 11, pp. 23-32, 2014.
- [32] J. Edwards, A. McKinnon, T. Cherrett, F. McLeod and L. Song, "The Impact of Failed Home Deliveries on Carbon Emissions: are Collection/Delivery Points Environmentally-Friendly Alternatives?," in *14th Annual Logistics Research Network Conference*, 2009.
- [33] N. A. Agatz, M. Fleischmann and J. A. V. Nunen, "E-Fulfillment and Multi-Channel Distribution—A Review," *European Journal of Operational Research*, vol. 187, no. 2, pp. 339-356, 2008.
- [34] J. F. Ehmke and D. C. Mattfeld, "Vehicle Routing for Attended Home Delivery in City Logistics," *Procedia-Social and Behavioral Sciences*, vol. 39, pp. 622-632, 2012.
- [35] N. Agatz, A. Campbell, M. Fleischmann and M. Savelsbergh, "Time Slot Management in Attended Home Delivery," *Transportation Science*, vol. 45, pp. 435-449, 2011.
- [36] S. D. Handoko, D. T. Nguyen and H. C. Lau, "An Auction Mechanism for The Last-Mile Deliveries Via Urban Consolidation Centre," in *IEEE International Conference on Automation Science and Engineering*, Taipei, 2014.
- [37] M. Browne, A. Woodburn and J. Allen, "Evaluating The Potential for Urban Consolidation Centres," *European Transport / Trasporti Europei*, vol. 35, pp. 46-63, 2007.
- [38] X. Wang, L. Zhan, J. Ruan and J. Zhang, "How to Choose “Last Mile” Delivery Modes for E-Fulfillment," *Mathematical Problems in Engineering*, vol. 2014, 2014.
- [39] Lindawati and A. Gunawan, "Minimizing Waiting Time In Last Mile Delivery for Smart City," in *Production and Operations Management Society*, 2015.
- [40] C. L. Quintero-Araújo, A. A. Juan, J. R. Montoya-Torres and A. Muñoz-Villamizar, "A Simheuristic Algorithm for Horizontal Cooperation in Urban

Distribution: Application to a Case Study in Colombia," in *Winter Simulation Conference*, Washington, 2016.

[41] I. Sungur, Y. Ren, F. Ordóñez, M. Dessouky and H. Zhong, "A Model and Algorithm for the Courier Delivery Problem with Uncertainty," *Transportation Science*, vol. 44, no. 2, pp. 193-205, 2010.

[42] M. Schneider, A. Stenger and J. Hof, "An Adaptive VNS Algorithm for Vehicle Routing Problems with Intermediate Stops," *OR Spectrum*, vol. 37, no. 2, pp. 353-387, 2015.

[43] L. C. Coelho, J.-P. Gagliardi, J. Renaud and A. Ruiz, "Solving the Vehicle Routing Problem with Lunch Break Arising In The Furniture Delivery Industry," *Journal of the Operational Research Society*, vol. 67, no. 5, pp. 743-751, 2016.

[44] D. Falsini, A. Fumarola and M. M. Schiraldi, "Sustainable Transportation Systems: Dynamic Routing Optimization for a Last-Mile Distribution Fleet," in *International Conference Sustainable Development: The Role of Industrial Engineering*, Bari, 2009.

[45] D. E. M. Dueñas, "Transshipment Networks for Last-Mile Delivery in Congested Urban Areas," in *6th International Conference on Information Systems, Logistics and Supply Chain*, Bordeaux, 2016.

[46] C. A. Isler, A. C. Bonassa and C. B. Cunha, "Clustering and Routing for a Real Automotive Parts Distribution Problem," *Journal of Applied Operational Research*, vol. 6, no. 2, pp. 82-97, 2014.

[47] B. Balcik, B. M. Beamon and K. Smilowitz, "Last Mile Distribution in Humanitarian Relief," *Journal of Intelligent Transportation Systems*, vol. 12, no. 2, pp. 51-63, 2008.

[48] A. Devari, "Crowdsourced Last Mile Delivery Using Social Network," M.S. thesis, Dept. ISE. Eng., State University Of New York At Buffalo, New York, 2016.

[49] Y. Wang, D. Zhang, Q. Liu, F. Shen and L. H. Lee, "Towards Enhancing the Last-Mile Delivery: An Effective Crowd-Tasking Model with Scalable Solutions,"

Transportation Research Part E: Logistics and Transportation Review, vol. 93, pp. 279-293, 2016.

[50] L. Pournajaf, L. Xiong, V. Sunderam and S. Goryczka, "Spatial Task Assignment for Crowd Sensing with Cloaked Locations," in *15th International Conference on Mobile Data Management*, Brisbane, 2014.

[51] S. Lee, Y. Kang and V. V. Prabhu, "Smart Logistics: Distributed Control of Green Crowdsourced Parcel Services," *International Journal of Production Research*, vol. 54, no. 23, pp. 6956-6968, 2016.

[52] A. Arslan, N. Agatz, L. G. Kroon and R. A. Zuidwijk, "Crowdsourced Delivery: A Dynamic Pickup and Delivery Problem with Ad-Hoc Drivers," *ERIM Report Series Reference*, 2016.

[53] W. Chen, M. Mes and M. Schutten, "Multi-Hop Driver-Parcel Matching Problem with Time Windows," *Flexible Services and Manufacturing Journal*, pp. 1-37, 2016.

[54] B. Li, D. Krushinsky, H. A. Reijers and T. V. Woensel, "The Share-A-Ride Problem: People and Parcels Sharing Taxis," *European Journal of Operational Research*, vol. 238, no. 1, pp. 31-40, 2014.

[55] S. Pan, C. Chen and R. Y. Zhong, "A Crowdsourcing Solution to Collect E-Commerce Reverse Flows In Metropolitan Areas," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1984-1989, 2015.

[56] J. Jin, T. G. Crainic and A. Løkketangen, "A Cooperative Parallel Metaheuristic for the Capacitated Vehicle Routing Problem," *Computers & Operations Research*, vol. 44, pp. 33-41, 2014.

[57] V. Leggieri and M. Haouari, "A Matheuristic for the Asymmetric Capacitated Vehicle Routing Problem," *Discrete Applied Mathematics*, 2016.

[58] G. M. Ribeiro and G. Laporte, "An Adaptive Large Neighborhood Search Heuristic for The Cumulative Capacitated Vehicle Routing Problem," *Computers & Operations Research*, vol. 39, no. 3, pp. 728-735, 2012.

- [59] L. Ke and Z. Feng, "A Two-Phase Metaheuristic for the Cumulative Capacitated Vehicle Routing Problem," *Computers & Operations Research*, vol. 40, no. 2, pp. 633-638, 2013.
- [60] J. Lysgaard and S. Wøhlk, "A Branch-and-Cut-And-Price Algorithm for the Cumulative Capacitated Vehicle Routing Problem," *European Journal of Operational Research*, vol. 236, no. 3, pp. 800-810, 2014.
- [61] H. Lei, G. Laporte and B. Guo, "The Capacitated Vehicle Routing Problem with Stochastic Demands and Time Windows," *Computers & Operations Research*, vol. 38, no. 12, pp. 1775-1783, 2011.
- [62] L. Eufinger, J. Kurtz, C. Buchheim and U. Clausen, "A Robust Approach to the Capacitated Vehicle Routing Problem with Uncertain Travel Times," 6 December 2016. [Online]. Available: <https://pdfs.semanticscholar.org/9c0a/a775bf472c215bac68cedd2717cfc3c68f14.pdf/>.
- [63] J. Nalepa and M. Blocho, "Adaptive Memetic Algorithm for Minimizing Distance in the Vehicle Routing Problem with Time Windows," *Soft Computing*, vol. 20, no. 6, pp. 2309-2327, 2016.
- [64] Q. Ding, X. Hu, L. Sun and Y. Wang, "An Improved Ant Colony Optimization and its Application to Vehicle Routing Problem with Time Windows," *Neurocomputing*, vol. 98, pp. 101-107, 2012.
- [65] R. Baldacci, A. Mingozzi and R. Roberti, "Recent Exact Algorithms for Solving the Vehicle Routing Problem Under Capacity and Time Window Constraints," *European Journal of Operational Research*, vol. 218, no. 1, 2012.
- [66] Y.-J. Gong, J. Zhang, O. Liu, R.-Z. Huang, H. S.-H. Chung and Y.-H. Shi, "Optimizing the Vehicle Routing Problem with Time Windows: a Discrete Particle Swarm Optimization Approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 2, pp. 254-267, 2012.
- [67] A. Agra, M. Christiansen, R. Figueiredo, L. M. Hvattum, M. Poss and C. Requejo, "The Robust Vehicle Routing Problem with Time Windows," *Computers & Operations Research*, vol. 40, no. 3, pp. 856-866, 2013.

- [68] S. Belhaiza, P. Hansen and G. Laporte, "A Hybrid Variable Neighborhood Tabu Search Heuristic for the Vehicle Routing Problem with Multiple Time Windows," *Computers & Operations Research*, vol. 52, pp. 269-281, 2014.
- [69] M. Cheikh, M. Ratli, O. Mkaouar and B. Jarboui, "A Variable Neighborhood Search Algorithm for The Vehicle Routing Problem with Multiple Trips," *Electronic Notes in Discrete Mathematics*, vol. 47, pp. 277-284, 2015.
- [70] N. Azi, M. Gendreau and J.-Y. Potvin, "An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Routes," *Computers & Operations Research*, vol. 41, pp. 167-173, 2014.
- [71] D. Cattaruzza, N. Absi, D. Feillet and T. Vidal, "A Memetic Algorithm for the Multi Trip Vehicle Routing Problem," *European Journal of Operational Research*, vol. 236, no. 3, pp. 833-848, 2014.
- [72] Z. Wang, W. Liang and X. Hu, "A Metaheuristic Based on a Pool of Routes for the Vehicle Routing Problem with Multiple Trips and Time Windows," *Journal of the Operational Research Society*, vol. 65, no. 1, pp. 37-48, 2014.
- [73] F. Hernandez, D. Feillet, R. Giroudeau and O. Naud, "Branch-and-Price Algorithms for the Solution of the Multi-Trip Vehicle Routing Problem with Time Windows," *European Journal of Operational Research*, vol. 249, no. 2, pp. 551-559, 2016.
- [74] B. Çatay, "A New Saving-Based Ant Algorithm for the Vehicle Routing Problem with Simultaneous Pickup and Delivery," *Expert Systems with Applications*, vol. 37, no. 10, pp. 6809-6817, 2010.
- [75] O. Polat, C. B. Kalayci, O. Kulak and H.-O. Günther, "A Perturbation Based Variable Neighborhood Search Heuristic for Solving the Vehicle Routing Problem with Simultaneous Pickup and Delivery with Time Limit," *European Journal of Operational Research*, vol. 242, no. 2, pp. 369-382, 2015.
- [76] R. Liu, X. Xie, V. Augusto and C. Rodriguez, "Heuristic Algorithms for a Vehicle Routing Problem with Simultaneous Delivery and Pickup and Time Windows in Home

Health Care," *European Journal of Operational Research*, vol. 230, no. 3, pp. 457-486, 2013.

[77] C. Wang, D. Mu, F. Zhao and J. W. Sutherland, "A Parallel Simulated Annealing Method for the Vehicle Routing Problem with Simultaneous Pickup–Delivery and Time Windows," *Computers & Industrial Engineering*, vol. 83, pp. 111-122, 2015.

[78] M. Avci and S. Topaloglu, "An Adaptive Local Search Algorithm for Vehicle Routing Problem with Simultaneous and Mixed Pickups and Deliveries," *Computers & Industrial Engineering*, vol. 83, pp. 15-29, 2015.

[79] M. Mahmoudi and X. Zhou, "Finding Optimal Solutions for Vehicle Routing Problem with Pickup and Delivery Services with Time Windows: A Dynamic Programming Approach Based on State–Space–Time Network Representations," *Transportation Research Part B: Methodological*, vol. 89, pp. 19-42, 2016.

[80] D. Männel and A. Bortfeldt, "A Hybrid Algorithm for the Vehicle Routing Problem with Pickup and Delivery and Three-Dimensional Loading Constraints," *European Journal of Operational Research*, vol. 254, no. 3, pp. 840-858, 2016.

[81] J. Li, P. M. Pardalos, H. Sun, J. Pei and Y. Zhang, "Iterated Local Search Embedded Adaptive Neighborhood Selection Approach for the Multi-Depot Vehicle Routing Problem with Simultaneous Deliveries and Pickups," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3551-3561, 2015.

[82] S. Shamshirband, M. Shojafar, A. A. R. Hosseinabadi and A. Abraham, "An Imperialist-Based Optimization Algorithm for the Open Vehicle Routing Problem," in *International Conference on Hybrid Artificial Intelligence Systems*, 2015.

[83] E. Cao, M. Lai and H. Yang, "Open Vehicle Routing Problem with Demand Uncertainty and its Robust Strategies," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3569-3575, 2014.

[84] R. Liu, Z. Jiang and N. Geng, "A Hybrid Genetic Algorithm for the Multi-Depot Open Vehicle Routing Problem," *OR Spectrum*, vol. 36, no. 2, pp. 401-421, 2014.

- [85] E. Lalla-Ruiz, C. Expósito-Izquierdo, S. Taheripour and S. Voß, "An Improved Formulation for the Multi-Depot Open Vehicle Routing Problem," *OR Spectrum*, vol. 38, no. 1, pp. 175-187, 2016.
- [86] A. Misevičius, "Using Iterated Tabu Search for the Traveling Salesman Problem," *Information Technology and Control*, vol. 32, no. 3, pp. 29-40, 2015.
- [87] T. Saenphon, S. Phimoltares and C. Lursinsap, "Combining New Fast Opposite Gradient Search with Ant Colony Optimization for Solving Travelling Salesman Problem," *Engineering Applications of Artificial Intelligence*, vol. 35, pp. 324-334, 2014.
- [88] J. B. Escario, J. F. Jimenez and J. M. Giron-Sierra, "Ant Colony Extended: Experiments on the Travelling Salesman Problem," *Expert Systems with Applications*, vol. 42, no. 1, pp. 391-410, 2015.
- [89] M. López-Ibáñez, C. Blum, J. W. Ohlmann and B. W. Thomas, "The Travelling Salesman Problem with Time Windows: Adapting Algorithms from Travel-Time to Makespan Optimization," *Applied Soft Computing*, vol. 13, no. 9, pp. 3806-3815, 2013.
- [90] D. Taş, M. Gendreau, O. Jabali and G. Laporte, "The Traveling Salesman Problem with Time-Dependent Service Times," *European Journal of Operational Research*, vol. 248, no. 2, pp. 372-383, 2016.
- [91] V. Arya, A. Goyal and V. Jaiswal, "An Optimal Solution to Multiple Travelling Salesperson Problem Using Modified Genetic Algorithm," in *International Journal of Application or Innovation in Engineering & Management*, vol. 3, no. 1, pp. 425-430, 2014.
- [92] A. A. Hosseinabadi, M. Kardgar, M. Shojafar, S. Shamshirband and A. Abraham, "GELS-GA: Hybrid Metaheuristic Algorithm for Solving Multiple Travelling Salesman Problem," in *14th International Conference on Intelligent Systems Design and Applications*, Okinawa, 2014.
- [93] S. Ghafurian and N. Javadian, "An Ant Colony Algorithm for Solving Fixed Destination Multi-Depot Multiple Traveling Salesmen Problems," *Applied Soft Computing*, vol. 11, no. 1, pp. 1256-1262, 2011.

- [94] X. B. a. Z. Liu, "An Improved Approximation Algorithm for the Clustered Traveling Salesman Problem," *Information Processing Letters*, vol. 112, no. 23, pp. 908-910, 2012.
- [95] Z. H. Ahmed, "The Ordered Clustered Travelling Salesman Problem: A Hybrid Genetic Algorithm," *The Scientific World Journal*, vol. 2014, pp.1-13, 2014.
- [96] C. Chen, S.-F. Cheng, A. Gunawan, A. Misra, K. Dasgupta and D. Chander, "TRACCS: A Framework for Trajectory-Aware Coordinated Urban Crowd-Sourcing," in *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.
- [97] Bingüler, A. H. Erol and S. Bulkan, "New Variable Neighborhood Search Structure for Travelling Salesman Problems," *British Journal of Mathematics & Computer Science*, vol. 6, no. 5, pp. 422-434, 2015.
- [98] Mladenović, Nenad and P. Hansen, "Variable neighborhood search," *Computers & operations research*, vol. 24, no. 11, pp. 1097-1100, 1997.
- [99] Hansen, Pierre, N. Mladenović and D. Urošević, "Variable neighborhood search and local branching," *Computers & Operations Research*, vol. 33, no. 10, pp. 3034-3045, 2006.
- [100] Hansen, Pierre and N. Mladenović, "Variable neighborhood search: Principles and applications," *European journal of operational research*, vol. 130, no. 3, pp. 449-467, 2001.
- [101] Duarte, Abraham, N. Mladenovic, J. Sánchez-Oro and R. Todosijevic, "Variable Neighborhood Descent," in *Handbook of Heuristics*, Springer International Publishing, 2016, pp. 1-27.

Vita

Eman Mohamed was born, in Khartoum, Sudan, where she received her primary and secondary education. She received her B.Sc. degree in Electrical and Electronics Engineering, with an emphasis in power systems, from Khartoum University in 2013. In 2014, she joined DAL Group where she worked as a Value Stream team member in business processes planning and improvement in DAL Food Company. She then joined DAL Design and Construction as a cost control engineer and projects coordinator.

In 2015, she joined the Engineering Systems Management master's program in the American University of Sharjah. During her master's study, she worked as a Graduate Teaching Assistant in the Industrial Engineering Department.