RICH VEHICLE ROUTING: A DATA-DRIVEN HEURISTIC APPLICATION FOR A LOGISTICS COMPANY

by

MUSTAFA SALİH ÇAVUŞ

Submitted to the Graduate School of Management

In partial fulfillment of

the requirements for the degree of

Master of Science

Sabancı University

July 2019

RICH VEHICLE ROUTING: A DATA-DRIVEN HEURISTIC APPLICATION FOR A LOGISTICS COMPANY

Approved by:

Prof. Dr. Burçin Bozkaya (Thesis Supervisor)	[illegible]
Asst. Prof. Dr. F. Tevhide Altekin	[illegible]
Assoc. Prof. Dr. Enes Eryarsoy	[illegible]

Approval Date: July 19, 2019

ABSTRACT

RICH VEHICLE ROUTING: A DATA-DRIVEN HEURISTIC APPLICATION FOR A LOGISTICS COMPANY

MUSTAFA SALİH ÇAVUŞ

Business Analytics, Master's Thesis, July 2019
Thesis Supervisor: Prof. Dr. Burçin Bozkaya

Keywords: E-Commerce Logistics, Customer Satisfaction, Routing, ALNS Heuristic, GIS

Changing online shopping behaviors have resulted in the emergence of different product and services that aim high customer satisfaction. In this thesis, we develop an alternative approach to solve problem of a logistics company, which operates solely for e-commerce transactions, using an Adaptive Large Neighborhood Search (ALNS) heuristic. To understand the nature of the distribution system and for the development of the solution procedure, we create, preprocess and analyze a dataset constructed from company's database that is used for daily operations. The proposed solution provides a prioritization mechanism for the deliveries based on certain specifications related to deliveries. To evaluate the performance of the proposed ALNS, we perform computational experiments using scenarios with real-life instances extracted from the dataset. Our results show that, the proposed ALNS can produce solutions with high quality regarding customer satisfaction.

ÖZET

ZENGİN ARAÇ ROTALAMA: BİR LOJİSTİK FİRMASI İÇİN VERİ ODAKLI SEZGİSEL UYGULAMA

MUSTAFA SALİH ÇAVUŞ

İş Analitiği Yüksek Lisans Tezi, Temmuz 2019
Tez Danışmanı: Prof. Dr. Burçin Bozkaya

Anahtar Kelimeler: E-Ticaret Lojistiği, Müşteri Memnuniyeti, Rotalama, ALNS Sezgisel Yöntemi, GIS

Değişen e-alışveriş alışkanlıkları yüksek müşteri memnuniyetini hedefleyen farklı ürün ve servislerin ortaya çıkmasına neden olmaktadır. Bu çalışmada, yalnızca e-ticaret işlemleri için faaliyette bulunan bir lojistik firmasının dağıtım problemine, Uyarlanabilir Geniş Komşuluklu Arama sezgisel yöntemi kullanılarak alternatif bir çözüm yaklaşımı geliştirilmektedir. Dağıtım sisteminin yapısını anlamak ve çözüm prosedürünün gelişimi için, firmanın günlük operasyonları için kullandığı veri tabanından bir veri seti oluşturulmuş, ön işleme yapılmış ve analiz edilmiştir. Önerilen çözüm, gönderilerin belirli özelliklerine dayanan bir önceliklendirme mekanizması sağlamaktadır. Önerilen Uyarlanabilir Geniş Komşuluk Arama sezgisel yönteminin performansını değerlendirmek için, veri setinden çıkarılan ve gerçek hayattan örnekler içeren senaryolar üzerinden analizler yapılmaktadır. Sonuçlar önerilen Uyarlanabilir Geniş Komşuluk Arama sezgisel yönteminin müşteri memnuniyeti kapsamında, yüksek kalitede sonuçlar üretebileceğini göstermektedir.

To my beloved family

ACKNOWLEDGEMENTS

First of all I would like to thank to Prof. Dr. Burçin Bozkaya and Asst. Prof. Dr. Fatma Tevhide Altekin for their invaluable support and guidance both in academic and in personal life, throughout my undergraduate and graduate studies. Without them, I can say that it would be impossible to have a path in this field and push my self this hard to fulfill requirements of this thesis.

Besides those I would like to express my gratitudes to CEO, CTO and to all of the employees in IT, HR, Operations, Customer Services, Marketing departments for their supports and friendships.

Finally, thanks to all business analytics graduate students and to my family who encouraged and supported me throughout the time of this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In the last decade, online shopping behaviors of consumers have rapidly changed due to developments occurring in e-commerce with the capability of collection and analysis of data in large quantities. Consequently, companies in related industries look for new business models and/or strategies with the aim of increasing their market share in a business environment with a very high competition. Among these related industries, logistics is one of the most important that introduces new products and services with aiming high customer satisfaction.

The main goal of this thesis is to develop an efficient solution procedure for the package delivery distribution problem in Istanbul for a Turkish logistics company. Different from traditional logistics companies, the one in this case operates solely for the deliveries to be made for e-commerce purchases made by consumers. The business model of the company involves a single central depot and nine different city warehouses called crossdocks located in Istanbul, from where goods are delivered to customers by independent couriers who are familiar with the delivery zones in which they operate. As a result of the flexibility obtained with independent couriers and the use of technological infrastructure, the company provides both standard and premium services to its customers. In the standard service, deliveries are to be made within three days whereas in premium services, customers can select either same-day or next-day delivery options within the time slot demanded of the day. All the deliveries are done within the time slots of the day according to their promised date.

In this business model, customer satisfaction is measured with the number of tickets opened by the customers whose services are not fulfilled within the promised time. According to the information given by the company, 80% of the customer tickets are attributed to the customers who purchased premium services. Hence, the company’s primary objective is the fulfillment of premium delivery services to the maximum satisfaction of its customers. Currently, dispatch of the deliveries is arranged manually at the crossdocks by supervisors and to have a route plan, each courier must manually process the addresses of the customers they are assigned to determine where they are located and hence plan a route. This entire manual process takes approximately one hour of work for both supervisors and couriers and consequently, it yields a significant operational inefficiency. Additionally, due to lack of any prioritization mechanism for deliveries, the manual dispatch arrangement of deliveries cannot meet the requirements of the company's primary objective.

For this problem, which is a variant of the famous Vehicle Routing Problem (VRP), we manipulate and analyze a dataset constructed from the company's database with the aim of understanding the nature of the operations. For the next step, we model this distribution problem as a Rich Vehicle Routing Problem (RVRP) and apply an Adaptive Large Neighborhood Search (ALNS) algorithm to solve it. Studies in the recent literature have shown that ALNS performs well especially for the real-life problems with real world constraints, objectives and variables. We then conduct computational experiments to show the effectiveness of our proposed ALNS implementation.

The organization of the remainder of this thesis is as follows: Chapter 2 reviews the literature on applications of ALNS to various VRP types. In Chapter 3, the data preprocessing and descriptive analytics steps are presented. The problem description, model formulation and the proposed ALNS application are presented in Chapter 4, which is followed by computational results in Chapter 5. Finally, in Chapter 6, we conclude the thesis with a summary, present its contributions with some directions for the future work.

CHAPTER 2

LITERATURE REVIEW

The VRP is one of the most studied combinatorial optimization problems in the literature since it was introduced by Dantzig and Ramser (1959). Today, there are numerous variants of VRPs with the aim of bringing solutions to both real life and theoretical transportation questions. According to Golden et al. (2008), solution methods for VRPs can be grouped under five categories: exact approaches like branch and bound method that tries to compute every possible solution, constructive approaches that create routes with an attempt of minimizing cost, two-phase algorithms such as cluster-first and route-second, heuristic methods, and meta-heuristics. Dramatically increased computational power brought by the technological developments in the last two decades shifted attention especially to heuristic and meta-heuristic approaches for new types of VRPs including additional constraints and objectives with a higher degree of complexity that is stimulated by complex characteristics of real-life VRPs. Rich Vehicle Routing Problem (RVRP) is a term used to describe groups of extended real-life VRPs and despite its vague definition in the literature, a RVRP can be identified as a type of VRP that includes partial or complete aspects of real-world applications including constraints, optimization criteria and preferences (Lahyani et al., 2015). One of the recent attempts that provide a unified heuristic model for a large class of VRPs was proposed by Pisinger and Ropke (2007). This heuristic is based on Shaw's (1998) Large Neighborhood Search (LNS) with an addition of multiple sub-heuristics and an adaptive layer which was named as Adaptive Large Neighborhood Search (ALNS). Like most of the local search heuristics, ALNS tries to explore the search space but rather differently, it can make modifications up to 40% of a given solution in only single iteration which eventually leads to the exploration of a larger neighborhood and the achievement of better results. This characteristic of the heuristic approach makes it suitable, especially for the problems with tight constraints and high complexity, which are the two main characteristics observed in RVRPs. Many insights gained from the application of ALNS to a large variety of VRPs, as discussed in the next section, could be extended to other complex problems.

2.1 A Structural Review

VRP problems can be classified based on various attributes like depot characteristics, objective function, vehicle characteristics, among others. The structure of our review is mainly based on the VRP type, which is determined by attributes mentioned above. Since our review is focused on the application of ALNS heuristic to various VRPs, the strengths, weaknesses and differentiating features of the reviewed works that involve ALNS, (see Table 2.1) are examined by focusing on heuristics utilized for the ALNS application and novel features related to the performance of the heuristics.

In order to have insights into the framework of LNS, Shaw's (1998) work should be examined first. In his work, he used constraint programming to solve a capacitated VRP; however, he claimed that "the traditional view of local search does not integrate well with the tree-based view of search with constraint programming", for which he developed a technique referred to as LNS. In his technique, the search over the feasible space is diversified with a heuristic called Shaw's Removal that aims to remove visits with a relatedness measure calculated based on the distance between visits and a binary variable representing whether or not both visits are served by the same vehicle. This removal heuristic constructed the basis for many other approaches created to serve problem specific situations for variety of RVRPs. For the second phase, which is the re-insertion of the removed visits into routes, Shaw's LNS utilizes a branch and bound technique that examines every combination for re-insertion of removed visits and picks the one(s) with the minimum cost.

Table 2.1: Literature Review Summary Table

Author(s) (Year)	Objective Function	VRP Type	Depot Characteristics	Vehicle Characteristics
Shaw (1998)	Minimization of operational cost	CVRP	Single Depot	Unlimited number of identical vehicles
Ropke and Pisinger (2006)	Minimization of operational cost	VRPPD	Multiple Depots	Unlimited number of identical vehicles
Masson et al. (2013)	Minimization of total distance travelled	VRPPD	Multiple Depots	Fixed number of identical vehicles
Qu and Bard (2012)	Minimization of number of vehicles and minimization of the total distance travelled	VRPPD	Single Depot	Identical Vehicles
Emeç et al. (2016)	Minimization of total distribution cost	VRPPD	Single Depot	Fixed number of identical vehicles
Ghilas et al. (2016)	Minimization of total travel cost	VRPPD	Multiple Depots	Heterogeneous fleet with two levels
Grimault et al. (2017)	Minimization of total travel cost	VRPPD	Multiple Depots	Heterogeneous fleet
Azi et al. (2012)	Maximization of total profit	DVRP	Single Depot	Fixed number of identical vehicles
Chen et al. (2018)	Minimization of total travel cost and the fixed cost of used vehicles	DVRP	Single Depot	Soft constrained number of heterogeneous vehicles
Hemmelmayr et al. (2012)	Minimization of total cost	2E-VRP	Single Depot at First Level & Multiple Depots at Second Level	Fixed number of identical vehicles with two levels
Grangier et al. (2016)	Minimization of the fleet size and the travel cost	2E-VRP	Single Depot at First Level & Multiple Depots at Second Level	Fixed number of identical vehicles with two levels
Kovacs et al. (2013)	Minimization of total travel time	MP-VRP	Single Depot	Fixed number of identical vehicles

Table 2.1 (continued)

Dayarian et al. (2016)	Minimization of total fixed vehicle and routing costs	MP-VRP	Multiple Depots	Fixed number of identical vehicles
Mancini (2016)	Minimization of total delivery cost	MP-VRP	Multiple Depots	Heterogeneous fleet composed of vehicles with different characteristics
Demir et al. (2012)	Minimization fuel, emission and driver costs	VRP	Single Depot	Fixed number of identical vehicles
Ribeiro and Laporte (2012)	Minimization of sum of arrival times at the customers	CVRP	Single Depot	Fixed number of identical vehicles
Luo et al. (2016)	Minimization of total expected transportation cost	VRP	Single Depot	Fixed number of identical vehicles
Bozkaya et al. (2017)	Minimization of total transportation cost and security risk of transporting	CVRP	Single Depot	Fixed number of identical vehicles

Vehicle Routing Problem with Pickup and Deliveries (VRPPD) and Extensions

Ropke and Pisinger (2006) are the first ones who developed ALNS and used it to solve a VRPPD with Time Windows (VRPPDTW), having modified Shaw's LNS by using multiple heuristics for both removal (destroy) and insertion (repair) of the visits. Selection of heuristics is made by a roulette wheel selection mechanism in which the probability of a heuristic being selected is determined by an adaptive weighting and a scoring tool which evaluates the past performance of each heuristic. Other than the use of multiple heuristics and adaptive selection mechanism, Ropke and Pisinger utilized simulated annealing for the acceptance criteria because they claimed that tree-based search of Shaw could get trapped in a local minimum. Their proposed ALNS heuristic improved many of the best-known solutions from the literature of that time, but still had some problems like getting trapped in local minimum at some instances or lacking the ability to minimize the number of vehicles by itself and requiring an additional algorithm to overcome it.

Masson et al. (2013) and Qu and Bard (2012) extended the problem to a VRPPD with Transshipment/Transfer (VRPPDT) in which deliveries are brought to a transfer point at the first stage and from the transfer point to customer locations at the second stage. Masson et al. (2013) adapt heuristics existing in literature for insertion and removal of both visits and transfer points as well as includes new ones that can insert requests through transfer points. This novel attribute of insertion through transfer points, deals with the complexity of the problem successfully and yields high-quality results. However, with the newly introduced heuristics, the time required to make a feasibility check for candidate insertions increases dramatically, which results in much longer computation times than solving the VRPPDTW.

Qu and Bard provide a two-phase model in which a Greedy Randomized Adaptive Search Procedure (GRASP) is utilized for construction of routes at the first phase and ALNS to improve a subset of solutions obtained through GRASP at the second phase. There are also two adjustments in their work, which affect the performance of the ALNS: first, some best insertion positions are stored for a given request and a route, and secondly, the number of customers to be removed is changed reactively. ALNS implemented with GRASP and these adjustments successfully overcomes the problem of getting trapped in local optimum but as in Masson's work, computation time increases significantly.

Emeç et al. (2016) propose an ALNS model for an E-grocery Delivery Routing Problem (EDRP) which is a VRPPD with external vendors at multiple locations supplying premium products that are required to be delivered to customers in a single visit together with regular products supplied from a main depot. The proposed model includes in total two insertion, six vendor selection and thirteen removal heuristics, two of which are newly introduced and the remaining ones are adapted from Ropke and Pisinger (2006), Pisinger and Ropke (2007) and Demir et al. (2012). The main contribution of this work is the use of vendor selection/allocation algorithms as auxiliary actors utilized in the repair phase of the ALNS.

Ghilas et al. (2016) utilize ALNS to solve a VRPPDTW and Scheduled Lines problem (VRPPDTW-SL) motivated by a scheduling problem to serve freight requests, in which part of the journey can be carried on a scheduled public transportation line. The model formulated for VRPPDTW-SL involves a heterogeneous fleet with two levels resulting in larger routing costs and a sharply increased problem complexity due to synchronization constraints introduced by the model. Nevertheless, the proposed heuristic yields competitive results thanks to the auxiliary algorithms that make simple but efficient feasibility checks. A similar model is presented by Grimault et al. (2017) in which a heterogeneous fleet of trucks that are synchronized based on unitary loading and unloading resources on pickup or delivery locations, transports goods between the sites. This time, complex synchronization constraints are dealt with a more advanced feasibility check framework that introduces priority rules for the insertion process, consisting of three decisions and several rules for each decision. Additionally, two new removal heuristics are introduced, which aim to relax a given solution focusing on resource constraints.

Dynamic Vehicle Routing Problem (DVRP) and Extensions

DVRPs consist of visits occurring dynamically and must be responded to in real time either with or without predetermined static visits. Azi et al. (2012) study a Dynamic Vehicle Routing Problem with Multiple Delivery Routes (DVRPMDR) containing only dynamic visits that are placed into the model with consideration of a sample of possible scenarios for the occurrence of future requests, obtained through a historical data. So, at each iteration, a given solution includes a mix of true and expected requests. In order to deal with the hierarchical nature of the problem, the proposed ALNS involves removal heuristics at three levels: customer, route and workday.

Chen et al. (2018) propose a similar model for a DVRP with Time Windows (DVRPTW) that involves static and dynamic visits together. An initial solution containing all static visits is optimized with ALNS and then dynamic visits are added to the routes with a three decision-based feasibility check system where, if insertion is done successfully, ALNS reoptimizes the obtained solution. In this model, there is no problem specific insertion or removal heuristic, but differently, heuristics are weighted and evaluated in pairs. The resulting model approach provides better solutions as the number of vehicles decreases and its performance decreases sharply when customer locations are distributed randomly with relatively tight service time windows.

Two-Echelon Vehicle Routing Problem (2E-VRP) and Extensions

2E-VRPs involve deliveries from a central depot to satellite facilities at the first level and from satellite facilities to the customers at the second level. Hemmelmayr et al. (2012) propose a modeling approach which transforms Location Routing Problem (LRP) instances into 2E-VRP instances so that the same heuristics can address both problems. Newly introduced satellite removal and insertion heuristics are used in a hierarchical scheme followed by a local search that checks whether the first level VRP problem can be improved. The whole framework can successfully deal with the multi-level nature of the problem.

2E Multiple-Trip VRP with Satellite Synchronization (2E-MTVRP-SS) is modeled and solved also using ALNS by Grangier et al. (2016). Different from the previously mentioned 2E-VRP, their model first designs second-level routes for a multi-trip multiple-depot problem. Like in Azi et al.'s (2012) work, removal heuristics at three levels are utilized. Additionally, they introduce three distinct greedy insertion heuristics that operate either by inserting into an existing trip, by creating a new trip or by splitting a trip. High computation times due to the complex nature of the problem is dealt with an efficient way that evaluates potential insertions in terms of profitability and feasibility.

Multi-Period VRP (MP-VRP) and Extensions

Kovacs et al. (2013) utilize ALNS to solve a Consistent VRP (ConVRP) in which customer satisfaction is prioritized and each customer is provided with deliveries made by the same driver at the same time of the day. Unlike any other work in this review, the proposed ALNS is based on a template from which the actual daily routes are derived. Another novel feature of this work is the randomization of the route construction procedures, which diversifies the search over the feasible space beyond the levels obtained through simulated annealing. While the template-based ALNS results in sharply increased costs, the authors resolved this issue by allowing delays in departure times.

On the other hand, Dayarian et al. (2016) model a MP-VRP with Seasonal Fluctuations problem that aims to route a single plan for the whole horizon. The performance of ALNS is increased dramatically with the proposed removal heuristics that operate in accordance with the problem specific entities such as the depot, producer and the plant, and with the use of a solution representation scheme encoded by using these entities. Using an efficient data structure in this scheme leads to a constant time for application of each insertion and removal heuristic. Mancini (2016) use an ALNS-based matheuristic to solve an ultra-constrained Multi Depot Multi-Period VRP with a Heterogeneous Fleet. In his work, Mancini combines several neighborhood strategies by using customers-to-route assignment variables, which enable the model to diversify the search neighborhood with a higher degree than LNS. This ALNS-based matheuristic could be extended for other problems with the possibility of exploring the whole search area within reasonably small computation times.

VRPs with Complex Objective Functions

The next of VRPs we review belong to one of the VRP types listed above, but differently, they involve relatively more complex objective functions. Demir et al. (2012) study Pollution-Routing Problem (PRP), which is an extension of the VRPTW. They propose an ALNS-based approach with removal heuristics that operate at the spatial level, which are adapted and improved by many of the ALNS heuristics utilized for VRPs and by some of the works reviewed in this chapter. In their study, the authors develop, in order to improve the quality of the solutions produced by ALNS, a Speed Optimization Algorithm that determines the speed of vehicles which has implications on the objective function composed of fuel consumption, emission and driver costs.

Ribeiro and Laporte (2012) utilize ALNS to solve a Cumulative Capacitated VRP (CCVRP) in which the (minimized) objective function is the sum of the arrival times at the visit locations in a case like natural disasters. In this work, widely accepted insertion and removal heuristics like Shaw and Greedy are improved and utilized at multiple levels. Luo et al. (2016) propose ALNS for a VRP with stochastic demand and weight-related cost. The setting of this problem is similar to that of the previously mentioned DVRPs, so the authors apply a dynamic recourse strategy, which employs several approximation schemes to obtain the minium expected cost.
Lastly, Bozkaya et al. (2017) formulate a CVRP with a bi-objective function for the transportation of valuables in cash-in-transit (CIT) operations and solve it with ALNS proposed by Emeç et al. (2016).

2.2 Discussion

ALNS is utilized for many VRPs with different characteristics in terms of various attributes like objective function, vehicle properties and demand type. No matter what the setting is, each study tries to obtain good quality solutions in reasonable computation time. In order to achieve good quality solutions, problem-specific removal and insertion heuristics are introduced, and the success of these is demonstrated by using them or their modified versions for other problems with similar settings. Secondly, four of the reviewed works present efficient and fast ways of evaluating the feasibility of insertion heuristics, which decrease computation time despite the high complexity of the problems. Despite all the improvements done for ALNS, its main framework did not change that much and the issue of tradeoff between quality and computation time still exists for most of the studies reviewed here. Future researches who wish to employ ALNS might include different acceptance criteria with the use of multiple local search algorithms, design of problem-independent feasibility check mechanisms and the fortification of ALNS with various metaheuristics. In our implementation of ALNS for the Rich VRP of the logistics company, we adapt some of the existing destroy and repair operations in the reviewed literature, and modify them in a convenient way for our purposes in this study.

CHAPTER 3

DATA PREPROCESSING AND DESCRIPTIVE ANALYSIS

In this chapter, we present the dataset constructed from the company’s database using HeidiSQL. This is the actual database utilized by the company for its daily operations. However, the constructed dataset either includes missing/misplaced values or is not enough only by itself to understand the nature of the distribution system. Thus, we preprocess the dataset and introduce new variables that are required for our purposes in this study.

3.1 Data Source

The dataset constructed includes 2,492,956 deliveries completed across nine city warehouses (crossdocks) in İstanbul within a time frame from January 1, 2018 to December 31, 2018. The variable definitions are shown in Table 3.1.

Table 3.1: Constructed Dataset for Study

Variable Name	Description
AD	Date and time in which delivery is added to system
DDP	Date and time promised for the delivery
DD	Date and time in which delivery is completed
B	Unique barcode for the delivery
PT	Product type: “Standard”, “Same Day”, “Next Day”
TN	Town name
XDN	Crossdock name
L1	Y coordinate of the customer
L2	X coordinate of the customer
C.ID	Unique ID for the courier

3.2 Data Preprocessing

In this section, we present the data preprocessing done to prepare the dataset for descriptive analysis. The outputs obtained in this section are utilized also for the creation of the test instances.

3.2.1 Data Cleaning

The logistics company operates in 8 different cities of Turkey. Therefore, the dataset includes the deliveries completed within İstanbul as well as other cities. Deliveries completed in other cities but present in the dataset are defined as misplaced entries. Additionally, there are data entries with missing values for all 10 variables in the dataset. After elimination of data entries with missing values and misplaced deliveries using the R programming language (Appendix A-1), the remaining dataset contains 2,422,916 data entries, which is about 2.8% loss of data.

In order to visualize the dataset geographically, by using the QGIS software, we first partition the shape file of İstanbul into the distribution areas of the nine crossdocks. In order to partition the shapefile correctly, we merge the zones both in the district and the neighborhood levels (Appendix B-1). The final version of the partitioning is shown in Figure 3.1.

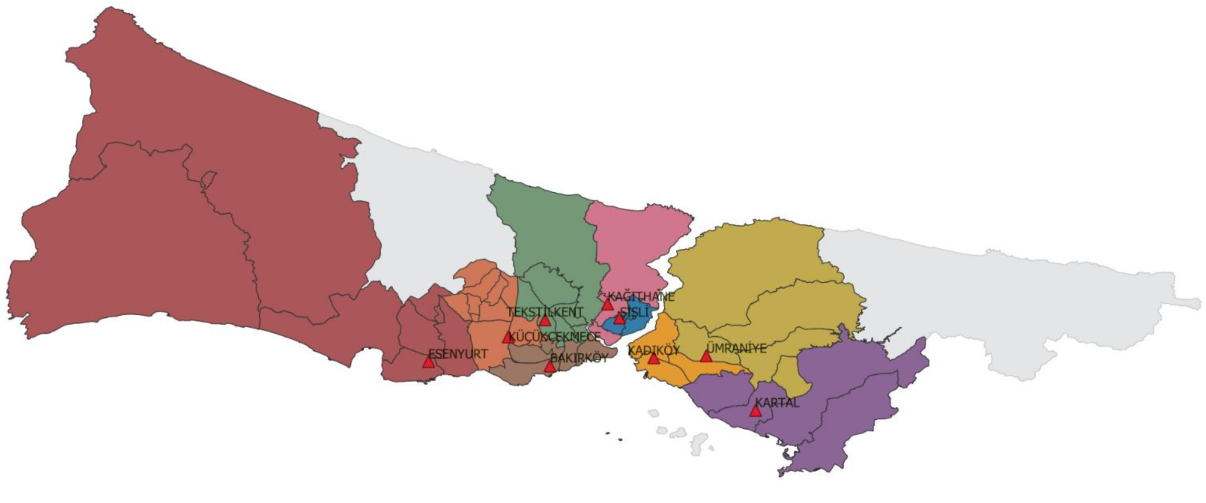


Figure 3.1: Partitioning of İstanbul into Distribution Areas by Crossdocks

As a second step, we utilize the geoprocessing tools of the software to remove data entries with wrong GPS coordinates (Appendix B-2) such as a delivery registered on the Küçükcekmece crossdock with the GPS coordinates from the Esenyurt crossdock's distribution area. In total 67,853 data entries with wrong GPS coordinates are removed from the dataset, and Figure 3.2 shows the final output of the shape file with 2,355,063 deliveries.

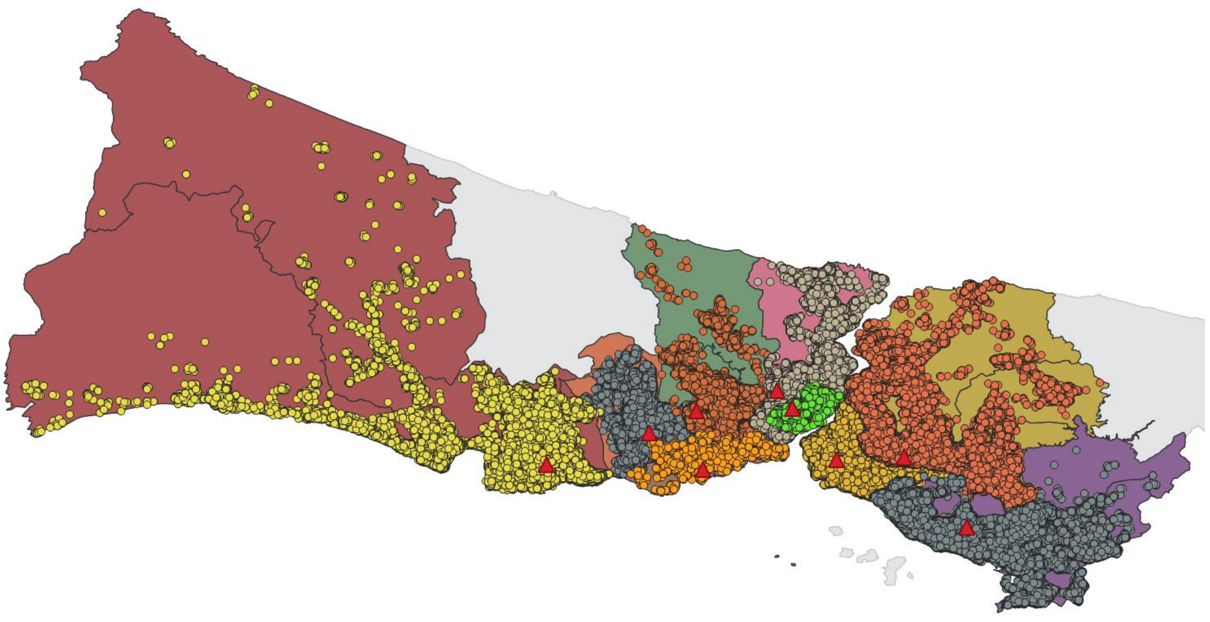


Figure 3.2: Locations of Deliveries and Depots after Geoprocessing

3.2.2 Variable Creation

Even though the constructed dataset has 10 variables about the deliveries, it has no information on whether a delivery is done within the promised time (On Time) or not (Delayed). Thus, by using "DDP" and "DD" variables, we create a new variable called "Delay Status" (Appendix A-2). Secondly, to identify the time slot of each delivery, by using "DD" variable, we create a new variable called "Time Slot" denoting one of the time slots of the day; "morning", "noon" or "evening" (Appendix A-3). All operations are done by using R programming language and a sample of obtained output can be seen in Table 3.2.

Table 3.2: Sample Output after Variable Creation

DDP	DD	Delay Status	Time Slot
10/31/2018 23:59	11/1/2018 14:39	Delayed	Noon
10/31/2018 23:59	11/1/2018 9:39	Delayed	Morning
10/31/2018 23:59	11/1/2018 9:19	Delayed	Morning
11/1/2018 23:59	11/1/2018 8:59	On Time	Morning
11/1/2018 23:59	11/1/2018 9:02	On Time	Morning
11/1/2018 23:59	11/1/2018 9:04	On Time	Morning
11/1/2018 23:59	11/1/2018 21:09	On Time	Evening
11/1/2018 13:00	11/1/2018 20:10	Delayed	Evening
11/1/2018 13:00	11/1/2018 9:10	On Time	Morning

3.3 Descriptive Analysis

In this section, we report outputs of our descriptive analysis describing the nature of the distribution system which is important for the development of the solution procedure. For our purposes in this study, we consider both “Same Day” and “Next Day” deliveries within the same product type “Premium”. Figure 3.3 shows the distribution of deliveries according to product types amongst nine crossdocks. The capacity of each crossdock is different based on the number of couriers utilized, which is proportional to the population’s demand within the distribution area. Despite the major differences in total delivery counts caused by the differences in capacity, portions of the premium deliveries are close to one other.

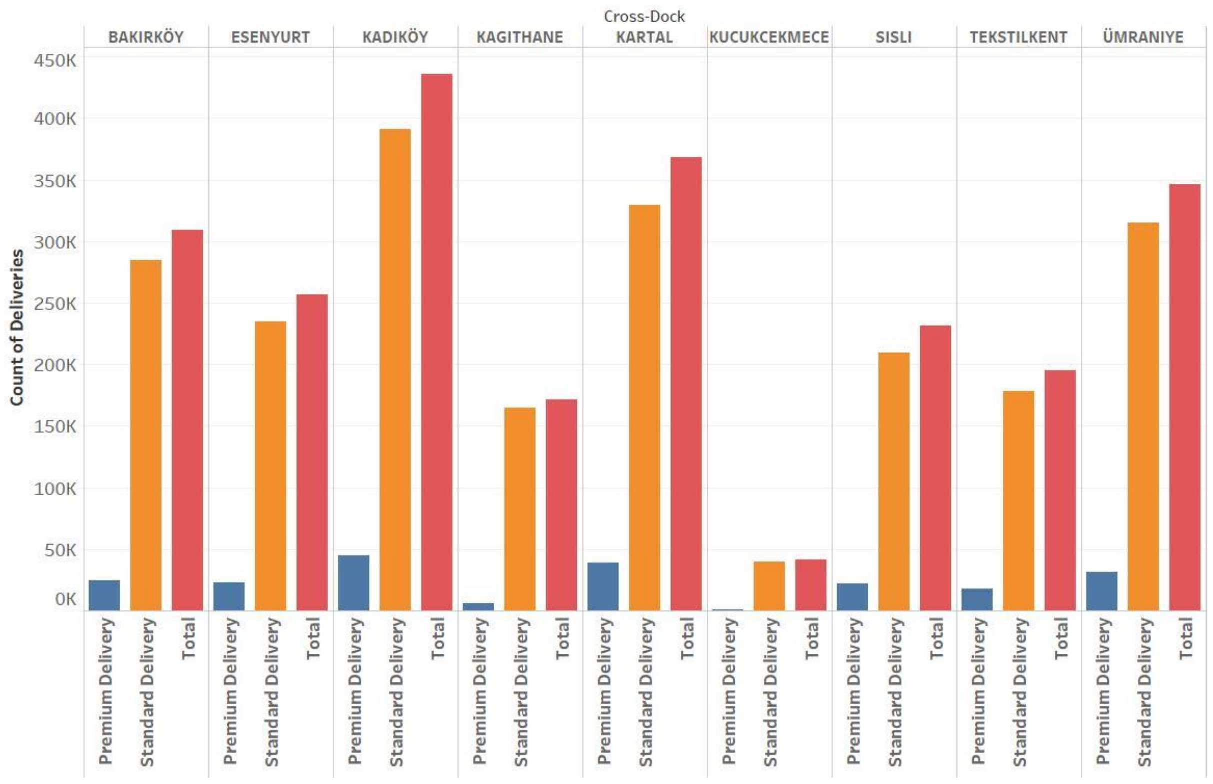


Figure 3.3: Delivery Counts By Product Type across Crossdocks

“Delay Status” is the only variable within this dataset that represents the performance of the crossdocks. Table 3.3 shows the percentage of deliveries done within promised time (On Time) and percentage of delayed ones (Delayed). Except the Ümraniye crossdock, delayed deliveries percentages of crossdocks are very similar to each other.

Table 3.3: Delay Status Percentages across Crossdocks

CROSSDOCK	PERCENTAGE DELAYED	PERCENTAGE ON TIME
BAKIRKÖY	0.08	0.92
ESENYURT	0.08	0.92
KADIKÖY	0.07	0.93
KARTAL	0.09	0.91
SISLI	0.06	0.94
TEKSTILKENT	0.07	0.93
ÜMRANIYE	0.15	0.85
KAGITHANE	0.05	0.95
KUCUKCEKMECE	0.07	0.93

In order to observe the distribution of deliveries into the time slots of the day, deliveries are grouped according to time slot values; “Morning”, “Noon” or “Evening”. The percentage values for times slots across the nine crossdocks are shown in Figure 3.4. Results illustrate that in any time slot of the day, crossdocks are faced with a similar amount of demand proportional to their capacities.

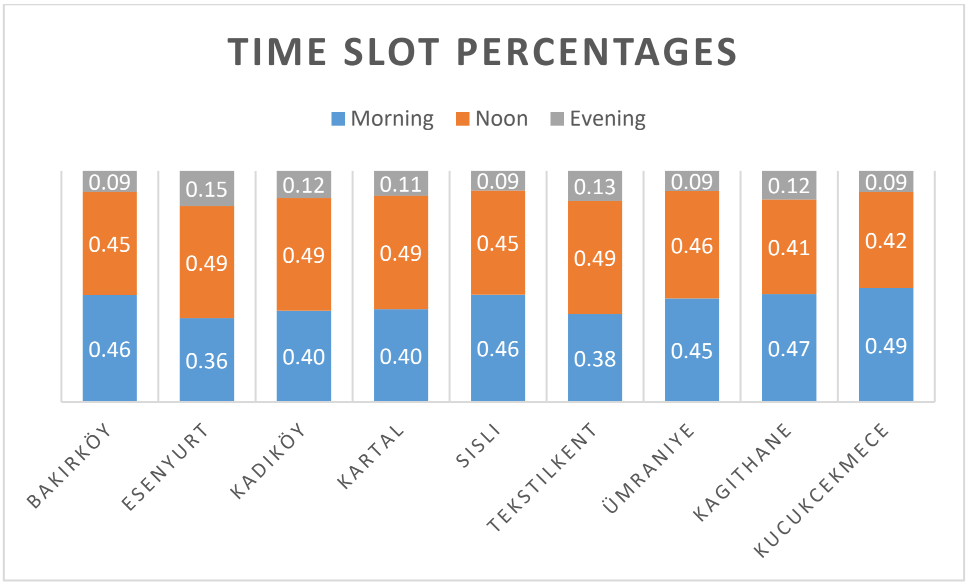


Figure 3.4: Time Slot Percentages of Deliveries across Crossdocks

Lastly, we analyze the distribution of deliveries with respect to the weeks of the year and present it in Figure 3.5. For simplicity, we present just two of the crossdocks. There are some major dips in the dataset caused by special ocassions such as religious holiday in summer and the Black Friday in December. As it can be seen clearly trends follow a similar pattern across the crossdocks during the weeks of the year.

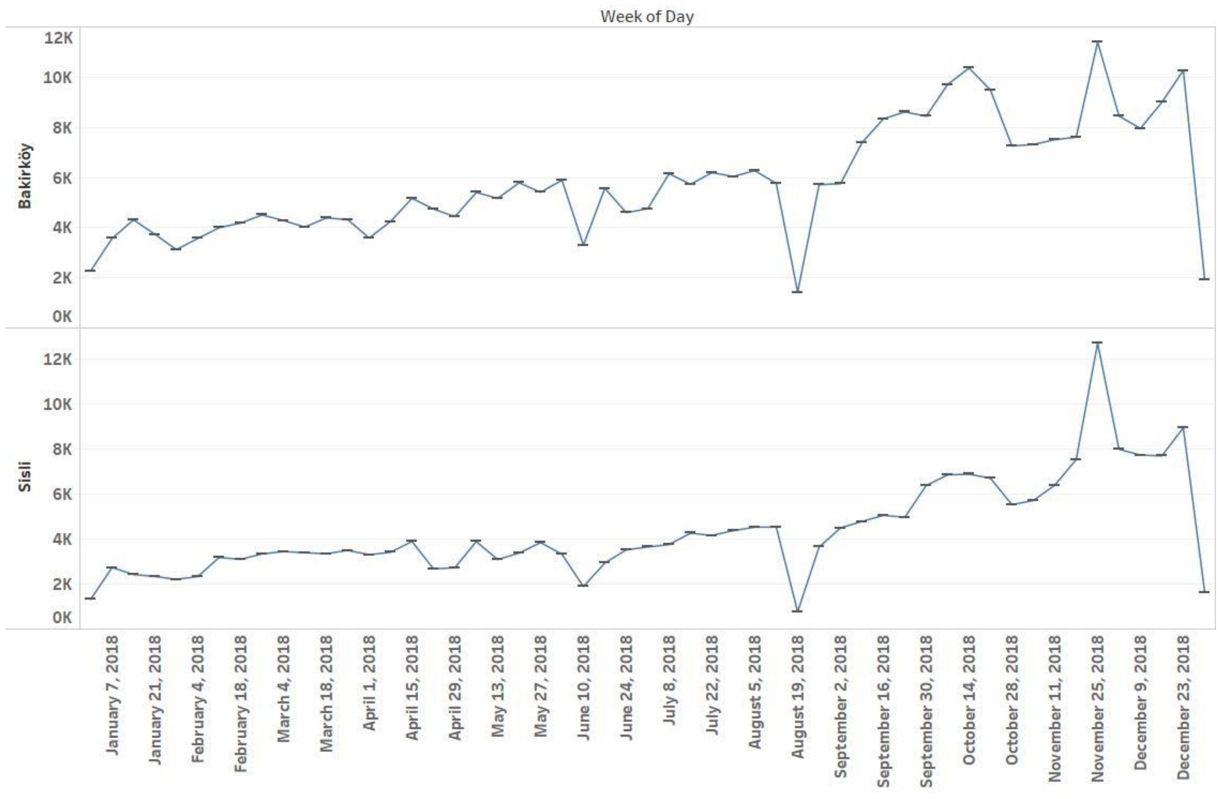


Figure 3.5: Weekly Distribution of Deliveries for Bakırköy and Şişli Crossdocks

Results and insights obtained in this chapter are presented to the company and later utilized in Chapter 4 for the development of the model and for the determination of the prioritization between deliveries by setting parameters regarding delay status and the product type.

CHAPTER 4

PROBLEM DESCRIPTION, MODEL FORMULATION AND THE PROPOSED ALNS APPLICATION

In this chapter, we explain the RVRP model for the delivery distribution problem for a single crossdock within a single time slot of the day and present a mixed integer linear programming formulation.

4.1 Problem Description

The RVRP can be defined as follows: Let $G = (N, A)$ be a directed complete graph network, $N = C \cup \{0\}$ be the set of nodes where "0" represents the single depot (crossdock), $C = \{1,2 \dots, n\}$ denote the set of customers to visit and $A = \{(i,j): i, j \in N \; and \; i \neq j\}$ be the set of arcs. Each node is associated with a continuous variable a_{ik} which represents the arrival time at node i by vehicle k. The "cost" of the travel between nodes i and j is reflected in a travel time matrix and is denoted by c_{ij}. Each customer has a service time s_i and an associated priority value p_i calculated based on the delivery type b_i ($b_i = 1$, if delivery status is premium, 0 otherwise), the number of delayed days g_i. Accordingly, the priority value p_i is calculated as follows:

$$p_i = \lambda^{b_i} \vartheta^{g_i}$$

where $\lambda \geq 1$ and $\vartheta \geq 1$ are the parameters for delivery type and the number of delayed days, respectively.

Finally, let $K = \{1,2,..,m\}$ be a fixed set of homogeneous vehicles with corresponding routes $R = \{r_1, r_{2,}.., r_m\}$ where each route serves its customers within a time slot of $T_{max} = 240$ minutes.

Sets and Parameters

$N = C \cup \{0\}$ set of the nodes where “0” is the depot

$C = \{1,2,\ldots,n\}$ customers to visit

$K = \{1,2,..,m\}$ vehicles

$R = \{r_1, r_2, .., r_m\}$ set of the routes

c_{ij} cost of traveling (travel time) from node i to node j

s_i service time of the customer i

p_i priority value of the customer i

b_i type of the delivery of the customer i

g_i number of delayed days for the delivery of the customer i

Decision Variables

x_{ijk} 1 if vehicle $k \in K$ travels from node $i \in$ N to node $j \in$ N, 0 otherwise

a_{ik} arrival time at node i by vehicle k

4.2 MILP Formulation

Maximize $F = \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} p_i\, x_{ijk}$

Subject to:

$$x_{iik} = 0, \forall i \in N, \forall k \in K \quad (1)$$

$$\sum_{i\in C} x_{0ik} \leq 1, \forall k \in K \quad (2)$$

$$\sum_{k\in K}\sum_{i\in N} x_{ick} \leq 1, \forall c \in C \quad (3)$$

$$\sum_{i\in N} x_{jik} = \sum_{i\in N} x_{ijk}, \forall j \in N, \forall k \in K \quad (4)$$

$$a_{ik} + s_i + c_{ij} \leq a_{jk} + M(1 - x_{ijk}), \forall i,j \in N, \forall k \in K \quad (5)$$

$$T_k = \sum_{i\in N}\sum_{j\in N} c_{ij}x_{ijk} + \sum_{i\in C}\sum_{j\in C} s_i x_{ijk}, \forall k \in K \quad (6)$$

$$T_k \leq T_{\max} \forall k \in K \quad (7)$$

$$x_{ijk} \in \{0,1\}, \forall i,j \in N, \forall k \in K \quad (8)$$

$$a_{ik} \geq 0, \forall i \in N, \forall k \in K \quad (9)$$

The objective function maximizes the total priority value associated with the visited customer nodes. Constraint (1) ensures that no vehicle travels from any node back to itself. Constraint (2) satisfies that every vehicle has at most one arc starting from the depot whereas constraint

(3) satisfies the limitation that no customer is visited more than once. Constraint (4) is the flow balance constraint at each node, and together with constraint (2), it ensures that each one of the routes terminates at the depot. Constraint (5) is required for subtour elimination by the calculated and ordered arrival times at each visited node. Constraints (6) and (7) enforce that the total time used in a route is no more than the total time available. Constraints (8) and (9) define the decision variables.

Since it is a variant of the classical vehicle routing problem, the modeled RVRP is NP-Hard (Toth and Vigo, 2002). As number of deliveries and couriers increase, the time required to find an optimal solution increases exponentially. Most studies in the literature propose different types of heuristic solutions for this kind of problems. Solomon (1987) proposes a type of insertion heuristic in which the route is created starting by a seed node and others are added based on feasibility criteria. Secondly, heuristic approaches with solution improvement methods is firstly described by Lin (1967) in which edges in the current solution are exchanged. Potvin and Rousseau (1987) improve Lin's edge exchange method and propose 2-opt algorithm. Finally, there are matheuristic approaches which are obtained through hybridization of heuristics and mathematical programming algorithms (Maniezzo and Caserta, 2010).

4.3 Proposed ALNS Application

In this section, we present the proposed ALNS framework for the RVRP.

4.3.1 Main Components

The proposed ALNS application is composed of five main components:

General Flow

Let S be a feasible solution obtained at the beginning as initial solution. At each iteration of the ALNS, the current solution S is modified by means of destroy (d) and repair (r) operations selected dynamically through a roulette-wheel selection mechanism where each operation pair is assigned with a weight (ω_{dr}) depending on its past performance. Pairwise evaluation of operations was tried by Kovacs et al. (2012) and has yielded better results. The calculation for selection probabilities (ρ_{dr}) is done as follows:

$$\rho_{dr} = \frac{\omega_{dr}}{\sum_{d=1}\sum_{r=1}\omega_{dr}}$$

Adaptive Scoring

Each pair of destroy d and repair r operations has an associated score π_{dr} that represents the performance of the pair. The search procedure is divided into segments composed of an equal number of iterations ($\varkappa_s$). The score of each pair is set to 0 at the beginning of each segment, and is increased based on three conditions (with $\sigma_1 > \sigma_2 > \sigma_3$):

- If a generated solution is a new global best solution, then the score of each member d and r of the pair used in the segment is increased by σ_1
- If a generated solution is better than the current one, then the score of each member d and r of the pair used in the segment is increased by σ_2
- If a generated solution is worse but still satisfies the acceptance criteria, then the score of each member d and r of the pair used in the segment is increased by σ_3

Adaptive Weight Adjustment

After the completion of each segment, the weight of each destroy and repair pair (ω_{dr}) is updated based on number of times the pair has been used (φ_{dr}), according to the following formula:

$$\omega_{dr} = \begin{cases} \rho\,\dfrac{\pi_{dr}}{\varphi_{dr}} + (1-\rho)\omega_{dr} & \varphi_{dr} \neq 0 \\ \omega_{dr} & \varphi_{dr} = 0 \end{cases}$$

The term $\rho\,\epsilon[0,1]$ is the reaction factor which creates control mechanism, if $\rho = 0$ weights do not change and if $\rho = 1$, only the scores obtained during current segment are considered.

Initial Solution

The proposed neighborhood search algorithm starts with a cheapest insertion solution produced using the algorithm by Rosenkrantz et al. (1974) as the initial solution. Each route r_k starts out with only the depot as both the initial and final stop. At each iteration and for each available route r_k, a customer node with the lowest insertion cost, which is calculated as the increase in the total route time after insertion, is considered for insertion into $r_1, r_2, .., r_m$, in that order. If, at any iteration, no more nodes can be added to a route, insertions will continue with the next available route. Insertion operations continue until no more nodes can be added to any available route.

Acceptance and Stopping Criteria

As in Ropke and Pisinger (2006), the acceptance criteria for the proposed ALNS is based on a simulated annealing decision criteria. It works as follows:

If a newly found solution S' is better than S, then it is accepted, otherwise it is accepted with a probability $\exp\frac{(z(S')-z(S))}{T}$, where $T > 0$ is an initial temperature decreased in every iteration by a decreasing coefficient Ψ called cooling rate, where $0 < \Psi < 1$. The entire ALNS execution stops after a pre-determined number $(\varkappa_s)$ of iterations in segment s.

4.3.2 Destroy Algorithms

At each iteration, the proposed ALNS framework uses one of the ten destroy algorithms described below. For seven of the algorithms, we set a remove parameter δ that determines the removal process for nodes in C.

1-) Random Destroy (RD):

The algorithm randomly selects $y \in [1, N]$ nodes (where N is the number of avaliable nodes) in C and removes them from their respective routes. The goal is to introduce a random component into the search procedure to avoid getting trapped in local optima.

2-) Destroy Single Vehicle (DSV):

The algorithm first randomly selects a route $r_k \in R$ and then removes δ nodes in C again randomly, which are on the selected route r_k.

3-) Destroy Random Sequential (DRS):

Let $i \in \mathrm{C}$ be a randomly selected node from any of the available routes $r_k \in R$. Then, this algorithm removes δ sequential nodes in C, starting by node i.

4-) Destroy Random from Vehicle Max Time (DRVMT):

Let $r_k \in R$ be the route with the maximum time utilized. The algorithm selects δ nodes $\in C$ randomly from the route r_k and removes them.

5-) Destroy Random from Vehicle Max Fitness (DRVMF):

Let $r_k \in R$ be the route with the maximum fitness value, which is the summation of the priority values of the customer nodes. The algorithm selects δ random nodes in C that are on route r_k and removes them from r_k.

6-) Shaw Destroyer (SD):

In order to remove similar nodes which would allow more insertion movements, Shaw (1998) introduced this algorithm. Similarly, we introduce a relatedness measure (Ω) for nodes $i, j \in C$ and define it as follows,

$$\Omega = \beta_1(c_{ij}) + \beta_2(v_{ij})$$

where c_{ij} denotes the travel time cost between nodes i and j and v_{ij} is a binary variable where $v_{ij} = 1$ if the two nodes i and j are served by the same vehicle, 0 otherwise.

The removal operation starts by choosing a random node in C and then it constructs a list composed of relatedness measures for the remaining nodes in C in comparison to the already chosen one. In the constructed list, the node in the $(y^p \cdot N)^{th}$ position gets removed from the current solution where $y \in [0,1]$ is a random number, N is the number of nodes in the constructed list and $p \geq 1$ is a Shaw removal determinism factor. The algorithm iterates until δ nodes get removed from the constructed list.

7-) Destroy Max Priority (DMaxP):

The algorithm constructs a list of randomly selected δ nodes in C for each available route $r_k \in R$. Each list is sorted according to the priority values of nodes in descending order and from each list, the node with the highest priority value gets removed from its respective route.

8-) Destroy Min Priority (DMinP):

This algorithm works in the same principle as *DMaxP* but instead, it removes nodes in C with the lowest priority value.

9-) Destroy Outlier (DO):

For each available route $r_k \in R$, the algorithm selects a random number, $y_k \in [1, N_k]$ where N_k is the number of customer nodes on route r_k. Then from each route r_k, the algorithm removes y_k nodes whose within-route-distance are highest. The within-route-distance for node i_k is calculated as $\sum_{j \in L(k)} c_{ij}$ where $L(k)$ is the list composed of nodes on r_k excluding node i.

10-) Geographical Destroy (GD):

Let $\mathcal{E}_k$ be the average time spent for route $r_k \in R$ calculated as $\mathcal{E}_k = \frac{total\ time\ r_k}{number\ of\ nodes \in C\ on\ r_k}$.
For each node $i \in C$, a value $f_i = s_i + c_{ii^*}$ is calculated where i^* is the successor of node i. Then, all nodes with $f_i \geq \mathcal{E}_k$ are removed from their respective routes.

4.3.3 Repair Algorithms

At each iteration, the proposed ALNS framework uses one of the nine repair algorithms described below. For three of the algorithms, we set an insertion parameter η that determines the number of nodes in C to insert. For each insertion, we check the feasibility of route by checking whether the time constraint is violated or not.

1-) Repair Cheapest (RC):

For each available route $r_k \in R$, the algorithm finds η nodes in C with the lowest insertion cost and inserts them into the respective route and position.

2-) Repair Cheapest Complete (RCC):

For each available route $r_k \in R$, the algorithm inserts nodes in C with the lowest insertion cost into selected route and position until no more nodes can be inserted into any available route.

3-) Repair N Cheapest (RNC):

Among all the feasible insertions, the algorithm inserts η nodes from C with the lowest insertion cost into their respective routes and positions.

4-) Repair Best Fitness (RBF):

The algorithm sorts all the feasible insertions according to their priority value and starting from the node with the highest priority value, it inserts η nodes from C into their best position.

5-) Repair Regret 2 (RR2):

In order to avoid myopic behavior introduced by the greedy insertion heuristics such as the cheapest insertion technique, this algorithm calculates a regret value for each feasible insertion by taking the difference of the insertion cost between the best position and the second-best position (Ropke and Pisinger, 2006). It then inserts the node with the highest regret value to its best position, until no more nodes can be added to any available route $r_k \in R$.

6-) Repair Regret 3 (RR3):

The working principle of this algorithm is similar to *RR2*, but this time the regret value is calculated by taking the difference of insertion cost between the best position and the third-best position.

7-) Repair Worst Fitness (RWF):

The algorithm sorts all feasible insertions according to their priority values and starting from the node with the lowest priority value, it inserts η nodes at their best position.

8-) Repair Random Best (RRB):

The algorithm constructs a list composed of possible insertions and sorts them according to their priority values in descending order. Then the algorithm selects a random integer $y \in [1, N_L]$ where N_L is the number of nodes in the constructed list and inserts a node from C in the yth place of the list at the best position in a route.

9-) Repair Outlier (RO):

Let $f(i)$ be the farness value of node $i \in C$ that is calculated as $f(i) = \sum_{j \in N} c_{ij}$. Starting from the node with the highest farness value, the algorithm inserts nodes at their best positions in a respective route until no more nodes can be added to any available route $r_k \in R$.

The Pseudo code of ALNS Algorithm

```
1:  $Set\ all\ weights\ \omega_{dr}\ equally\ and\ all\ scores\ \pi_{dr}\ to\ 0$
2:  $Generate\ an\ initial\ solution\ S\ by\ using\ the\ Cheapest\ Insertion\ Heuristic$
3:  $S_{best} \leftarrow S$
4:  $iterations \leftarrow 0$
5:  $\boldsymbol{while}\ \varkappa_i > iterations$
6:      $Select\ a\ destroy\ repair\ operation\ pair\ dr\ based\ on\ weights$
7:      $Generate\ S'\ by\ applying\ selected\ operation\ pair\ to\ S$
8:      $Update\ number\ of\ times\ used\ \varphi_{dr}\ for\ the\ selected\ pair$
9:      $\boldsymbol{if}\ z(S') > z(S_{best})\ \boldsymbol{then}$
10:         $S_{best} \leftarrow S'$
11:         $increase\ score\ of\ selected\ pair\ by\ \sigma_1$
12:     $\boldsymbol{elif}\ z(S') > z(S)\ \boldsymbol{then}$
13:         $S \leftarrow S'$
14:         $increase\ score\ of\ selected\ pair\ by\ \sigma_2$
15:     $\boldsymbol{else}$
16:         $\boldsymbol{if}\ S'\ is\ accepted\ by\ the\ simulated\ annealing\ criterion\ \boldsymbol{then}$
17:             $S \leftarrow S'$
18:             $increase\ score\ of\ selected\ pair\ by\ \sigma_3$
19:     $\boldsymbol{if}\ iterations\ reached\ \varkappa_s\ \boldsymbol{then}$
20:         $update\ weights\ of\ pairs\ and\ set\ scores\ to\ 0$
21:     $iterations \leftarrow iterations + 1$
22: $\boldsymbol{end\ while}$
23: $\boldsymbol{return}\ S_{best}$
```

CHAPTER 5

COMPUTATIONAL RESULTS

In this chapter, we solve the Rich Vehicle Routing Problem (RVRP) for the distribution process of a single crossdock within a single time slot using the proposed Adaptive Large Neighborhood Search (ALNS) algorithm. To evaluate the performance of the algorithm, we utilize scenarios with the real-life instances extracted from the dataset constructed for this study. All the algorithms are coded in Python 3.6 programming language and the mathematical model is constructed and solved using Gurobi 8.1.1. The experiments are performed on a computer equipped with Intel Core i7 2.9 GHz CPU (7500U) and 16 GB RAM.

5.1 Extraction of the Test Data

For extraction of the test data, we have selected the Bakırköy crossdock, which has a median number of deliveries among all nine crossdocks. Five random consecutive days are selected throughout the year. Figure 5.1 shows geographical locations of the depot (red triangle) and of the delivery points of the one of the selected days.

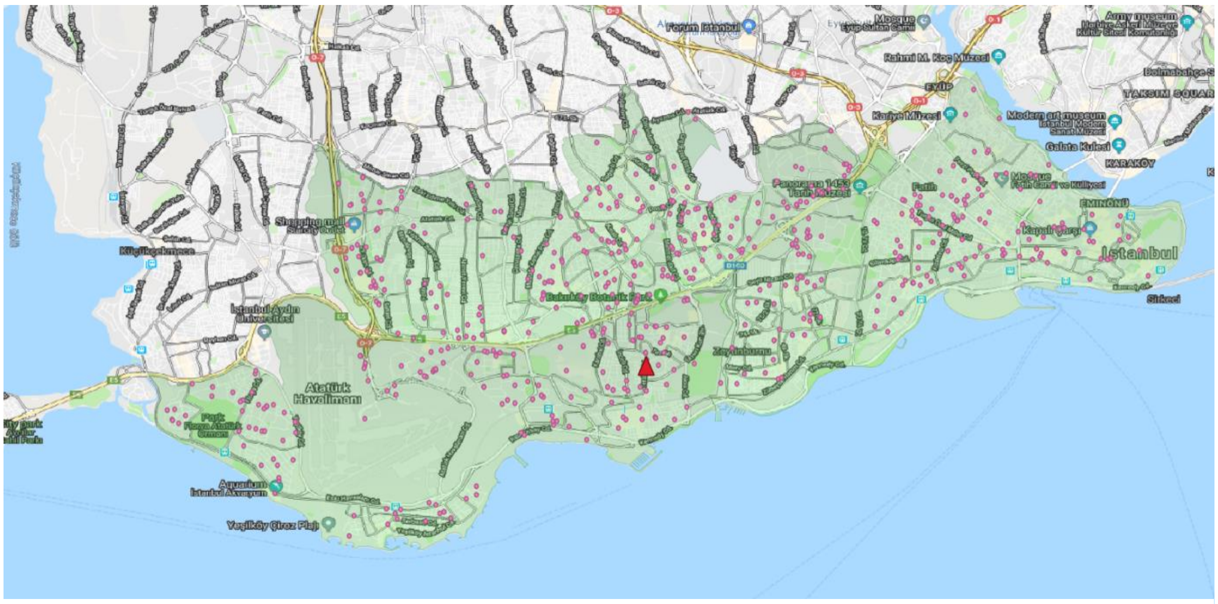


Figure 5.1: Locations of Deliveries and Depot for Bakırköy Crossdock

As the next step, the deliveries of each day are grouped by the time slot of the day (M = morning, N = noon, E = evening) and by the distinct courier ids (Appendix A-4). The resulting output gives the number of deliveries completed by each courier within the time slots of the day and is shown in Table 5.1.

Table 5.1: Delivery Counts By Time Slot and Distinct Courier ID

	D1			D2			D3			D4			D5		
C.ID	**M**	**N**	**E**	**M**	**N**	**E**	**M**	**N**	**E**	**M**	**N**	**E**	**M**	**N**	**E**
160	25	0	0	40	0	0	30	31	0	43	23	0	56	46	7
163	30	41	6	19	36	2	0	0	0	31	0	0	21	0	0
168	25	35	0	22	30	0	30	17	3	26	24	8	27	47	0
194	48	32	0	39	34	0	50	11	0	0	0	0	48	51	8
206	47	63	0	21	35	0	25	28	0	24	15	0	26	24	3
242	18	26	5	19	19	3	21	10	3	21	28	0	57	50	0
253	34	34	0	31	19	0	40	10	1	46	24	0	43	38	0
260	23	43	3	0	0	0	19	13	0	19	22	5	18	19	0
295	46	37	0	50	27	0	0	0	0	48	3	0	79	55	0
312	38	65	10	31	30	5	30	32	2	36	37	0	32	41	0
313	54	18	4	0	0	0	45	11	0	27	12	0	20	36	0
317	29	26	5	43	2	4	31	6	0	10	59	1	27	69	0
318	24	43	0	33	29	0	41	13	0	31	9	1	58	40	0
351	35	46	0	28	27	0	0	37	0	34	21	0	35	38	0
369	46	19	0	44	23	0	18	22	0	41	22	0	14	61	0

To test the performance of the ALNS, we first test instances for five days with 30 and 40 deliveries and one courier. For instances with higher number of deliveries and couriers, overall 20 different scenarios are generated for the distribution problem. The scenarios for each day differ in the number of deliveries and total priority values associated with the deliveries. The instance names reflect the specifications mentioned above. For instance, D1-1-54 represents the scenario for the first day with 54 deliveries and one courier whereas D3-4-176 represents the scenario for the third day with 176 deliveries and four couriers. Lastly, to test the prioritization mechanism of our approach, we create five different scenarios where each is composed of five days and the uncompleted deliveries are transferred to the next day with an update in their delayed days and hence their priority values. The travel time matrix for each test instance is generated using ArcMAP 10.2 NetWork Analyst.

5.2 Test Results

For the experiments on test instances described in section 5.1, we utilize parameters adapted from Emeç et al. (2016) and Pisinger and Ropke (2007), which we present in table 5.2.

Table 5.2: Parameters Used by the Proposed ALNS

Parameter Description	Parameter Value
Total number of iterations ($\varkappa_i$)	10000
Number of iterations for the segment completion ($\varkappa_s$)	200
Reaction factor for the roulette selection (ρ)	0.9
Score for the new best solution (σ_1)	10
Score for the better solution (σ_2)	5
Score for the worse but accepted solution (σ_3)	2
Initial temperature (T)	25
Cooling rate (Ψ)	0.99
Lower limit of the deliveries to remove (δ)	$\min\{0.1\|N\|, 30\}$
Upper limit of the deliveries to remove (δ)	$\min\{0.4\|N\|, 60\}$
Number of deliveries to insert (η)	5
Shaw parameter for time cost (β_1)	1
Shaw parameter for vehicle usage (β_2)	9
Shaw removal determinism factor (p)	5
Premium parameter for delivery (λ)	8
Delay parameter for delivery (ϑ)	2

To evaluate the performance of ALNS on instances with 30 and 40 deliveries and a single courier, we set a run time limit of 1 hour and present the following:

- $MILP$: Best feasible objective value obtained by the MILP model
- $MILP_{CPU}$: The CPU time of MILP in seconds
- $ALNS$: Objective value obtained by the ALNS
- $ALNS_{CPU}$: The CPU time of ALNS run in seconds
- $GAP_{ALNS-MILP}$: The difference between MILP solution and ALNS in percentage

Table 5.3: Test Results on Small Instances

$Instance$	$MILP$	$MILP_{CPU}$	$ALNS$	$ALNS_{CPU}$	$GAP_{ALNS-MILP}$
D1-30	54	7	54	40	0%
D2-30	60	6	60	15	0%
D3-30	62	8	62	16	0%
D4-30	52	3	52	15	0%
D5-30	104	5	104	12	0%
D1-40	56	72	56	87	0%
D2-40	64	143	62	40	3%
D3-40	66	176	65	66	2%
D4-40	66	158	64	41	3%
D5-40	108	1389	106	140	2%

Results for the small instances show that, even though ALNS obtained optimal results, MILP model outperforms it for the instances with 30 deliveries in terms of run time. For the instances with 40 deliveries, ALNS generated solutions with approximately 2% of gap and outperforms MILP model in terms of run time.

For larger instances we set a time limit of one hour for the instances with one courier, two hours for the instances with two or three couriers and four hours for the instances with four couriers. MILP model cannot produce a feasible solution within allowed time limits for the instances more than one courier, hence we solve MILP model with LP relaxation for these instances and present the following:

- LP_B: Best bound found by LP relaxation
- GAP_{MILP}: Difference between the upper and lower bound in percentage
- $MILP_{CPU}$: The CPU time of MILP in seconds
- $ALNS$: Objective value obtained by ALNS
- $ALNS_{CPU}$: The CPU time of ALNS run in seconds
- $GAP_{ALNS-LP}$: Difference between best bound found by LP relaxation and ALNS in percentage

Table 5.4: Test Results on Larger Instances

$Instance$	LP_B	GAP_{MILP}	$MILP_{CPU}$	$ALNS$	$ALNS_{CPU}$	$GAP_{ALNS-LP}$
D1-1-54	90	0%	248	90	149	0%
D2-1-50	100	0%	2217	100	82	0%
D3-1-50	104	0%	36	102	88	2%
D4-1-48	116	0%	232	114	53	2%
D5-1-79	162	3%	3600	156	355	4%
D1-2-102	265	6%	7200	256	309	4%
D2-2-94	248	4%	7200	246	215	1%
D3-2-95	254	10%	7200	244	191	4%
D4-2-94	264	7%	7200	256	350	3%
D5-2-137	334	8%	7200	320	450	4%
D1-3-149	492	29%	7200	476	428	3%
D2-3-137	350	20%	7200	350	448	0%
D3-3-136	480	29%	7200	462	357	4%
D4-3-137	360	36%	7200	346	470	4%
D5-3-194	448	292%	7200	410	700	9%
D1-4-195	744	49%	14400	726	1105	2%
D2-4-177	534	51%	14400	534	1113	0%
D3-4-176	546	87%	14400	546	975	0%
D4-4-178	442	96%	14400	442	921	0%
D5-4-250	616	926%	14400	580	3712	6%

For the larger instances, MILP model generated feasible solutions only for the instances with number of deliveries up to 50. Results in Table 5.4 shows that ALNS can generate feasible solutions with an average of 3% of gap with best bounds found by the MILP model. The run time for ALNS varies between 82 and 3712 seconds based on specifications of problem such as number of deliveries and number of utilized couriers.

Table 5.5 illustrates the uncompleted deliveries for the first scenario where we test the prioritization mechanism. To trace the distribution process the ID, Delivery Type (DT) and Delayed Days (DD) variables are presented and each uncompleted delivery is colored with color of the respective day. Results show that, the priotiziation mechanism within our proposed ALNS can obtain solutions in which premium deliveries are completed in the same day and standard deliveries are completed in at most three days. Tables for the remaining scenarios are presented in Appendix C.

Table 5.5: Test Results on Prioritization Scenario 1 (uncompleted deliveries)

D1			D2			D3			D4			D5		
ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD
15	0	0	41	0	0	81	0	0	129	0	0	161	0	0
16	0	0	48	0	0	87	0	0	130	0	0	166	0	0
22	0	0	69	0	1	92	0	0	134	0	0	169	0	1
32	0	0	74	0	0	93	0	0	138	0	0	170	0	0
34	0	0	77	0	0	99	0	0	143	0	0	183	0	0
			32	0	1	100	0	0	144	0	1	184	0	0
			34	0	1	101	0	1	146	0	0	186	0	0
			16	0	1	103	0	0	149	0	0	191	0	0
			15	0	1	104	0	0	151	0	0	192	0	0
						110	0	0	152	0	0	194	0	0
						115	0	0	157	0	0	195	0	0
						116	0	0	110	0	1	196	0	0
						117	0	1	93	0	1	197	0	0
						77	0	1	116	0	1	198	0	0
									115	0	1	199	0	0
									92	0	1	129	0	1
									103	0	1	130	0	1
												134	0	1
												138	0	1
												143	0	1
												146	0	1
												149	0	1

We finally present the average weights for the destroy and remove operation pairs, representing performance regarding the improvement in total priority value associated with the deliveries. We indicate the 30 best performing pairs in bold in Table 5.6. We observe that the operation pairs that involve heuristics related to the priority value show more success but, other operation pairs that diversifies the search, are also critical for the achievement of better solutions.

Table 5.6: Average Weights of Destroy Repair Operation Pairs

D/R	**RC**	**RCC**	**RNC**	**RBF**	**RR2**	**RR3**	**RWF**	**RRB**	**RO**
RD	0.0007	0.0025	0.0006	**0.0734**	0.0006	0.0007	0.0020	0.0006	0.0008
DSV	0.0013	0.0014	**0.0063**	**0.1370**	0.0012	0.0008	0.0011	0.0006	0.0025
DRS	0.0004	0.0013	0.0020	**0.0628**	0.0007	**0.0069**	**0.0035**	0.0009	**0.0041**
DRVMT	**0.0186**	0.0022	**0.0045**	**0.0470**	0.0005	0.0011	0.0011	0.0017	**0.0027**
DRVMF	0.0006	0.0010	0.0008	**0.0182**	0.0006	0.0016	0.0009	0.0009	0.0007
SD	**0.0317**	0.0007	0.0015	**0.0184**	0.0013	0.0008	0.0010	0.0016	0.0007
DMaxP	0.0007	0.0022	0.0012	**0.0423**	0.0012	0.0024	0.0018	0.0010	**0.0052**
DMinP	**0.0129**	**0.0286**	**0.0268**	**0.0836**	**0.0081**	**0.0301**	**0.0280**	**0.0566**	**0.0180**
DO	0.0009	0.0008	0.0007	**0.0026**	0.0011	0.0011	0.0011	0.0008	0.0008
GD	**0.0135**	**0.0737**	**0.0354**	**0.0291**	0.0022	0.0014	0.0024	0.0009	0.0016

CHAPTER 6

CONCLUSIONS

In this study, we proposed an alternative solution for the distribution problem of an e-commerce delivery company by developing an adaptive large neighborhood search (ALNS) algorithm. Similar studies in the literature are reviewed for the construction of destroy and repair heuristics that are building blocks of the algorithm, which turn out to be the most effective state-of-the-art approach for the Rich VRP problem under consideration.

For the development of the solution procedure, a dataset is constructed and processed with both descriptive and geographical tools. Thanks to analysis made with the mentioned tools, a wide knowledge about the nature of the distribution system is obtained.

In our proposed MILP formulation, we provide a prioritization mechanism that evaluates deliveries based on two aspects. Firstly, it evaluates deliveries based on their delivery type and secondly based on their delay status.

To evaluate performance of the proposed ALNS, scenarios with real life instances are generated. Increase in total priority value through fulfillment of premium deliveries is our primary objective which is directly related to number of tickets opened by the customers that measures the customer satisfaction. Computational results show that, ALNS can achieve solutions with high quality for the instances with a variety of number of deliveries and couriers.

The scope of this study is limited to the dataset constructed from the company’s database. More information about the relationship between unsatisfied delivery requests and customer complain tickets might help for the development of better and more accurate prioritization mechanisms for the distribution problem. For the future work, fast and efficient insertion feasibility check frameworks might be introduced, so that shorter times for insertion heuristics might be achieved. An extension for this study may contain multi-depots and multi-periods that can provide solutions for all nine crossdocks simultaneously for longer periods such as three, five or seven days.

BIBLIOGRAPHY

Azi, N., Gendreau, M., & Potvin, J.-Y. (2012). A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research, 199*(1), 103-112.

Bozkaya, B., Salman, F. S., & Telciler, K. (2017). An adaptive and diversified vehicle routing approach to reducing the security risk of cash-in-transit operations. *Networks, 69*(3), 256-269.

Chen, S., Chen, R., Wang, G.-G., Gao, J., & Sangaiah, A. K. (2018). An adaptive large neighborhood search heuristic for dynamic vehicle routing problems. *Computers & Electrical Engineering, 67*, 596-607.

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science, 6*(1), 80-91.

Dayarian, I., Crainic, T. G., Gendreau, M., & Rei, W. (2016). An adaptive large-neighborhood search heuristic for a multi-period vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review, 95*, 95-123.

Demir, E., Bektaş, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. *European Journal of Operational Research, 223*(2), 346-359.

Emeç, U., Çatay, B., & Bozkaya, B. (2016). An Adaptive Large Neighborhood Search for an E-grocery Delivery Routing Problem. *Computers & Operations Research, 69*, 109-125.

Ghilas, V., Demir, E., & Van Woensel, T. (2016). An adaptive large neighborhood search heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines. *Computers & Operations Research, 72*, 12-30.

Golden, B., Raghavan, S., & Wasil, E. (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*. Boston, MA: Springer US.

Grangier, P., Gendreau, M., Lehuédé, F., & Rousseau, L.-M. (2016). An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *European Journal of Operational Research, 254*(1), 80-91.

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

Hemmelmayr, V. C., Cordeau, J.-F., & Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Computers & Operations Research, 39*(12), 3215-3228.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2012). Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of Scheduling, 15*(5), 579-600.

Kovacs, A. A., Parragh, S. N., & Hartl, R. F. (2014). A template-based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks, 63*(1), 60-

Lahyani, R., Khemakhem, M., & Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research, 241*(1), 1-14.

Lin, S. (1965). Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal, 44*(10), 2245-2269.

Luo, Z., Qin, H., Zhang, D., & Lim, A. (2016). Adaptive large neighborhood search heuristics for the vehicle routing problem with stochastic demands and weight-related cost. *Transportation Research Part E: Logistics and Transportation Review, 85*, 69-89.

Mancini, S. (2016). A real-life Multi Depot Multi Period Vehicle Routing Problem with a Heterogeneous Fleet: Formulation and Adaptive Large Neighborhood Search based Matheuristic. *Transportation Research Part C: Emerging Technologies, 70*, 100-112.

Maniezzo, V., Stützle, T., & Voß, S. (2010). Matheuristics : Hybridizing Metaheuristics and Mathematical Programming.

Masson, R., Lehuédé, F., & Péton, O. (2013). An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers. *Transportation Science, 47*(3), 344-355.

Mattos Ribeiro, G., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research, 39*(3), 728-735.

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

Pisinger, D., & Ropke, S. (2010). Large Neighborhood Search. In M. Gendreau & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 399-419). Boston, MA: Springer US.

Potvin, J.-Y., & Rousseau, J.-M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research, 66*(3), 331-340.

Qu, Y., & Bard, J. F. (2012). A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment. *Computers & Operations Research, 39*(10), 2439-2456.

Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science, 40*(4), 455-472.

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M. (1974, 14-16 Oct. 1974). *Approximate algorithms for the traveling salesperson problem.* Paper presented at the 15th Annual Symposium on Switching and Automata Theory (swat 1974).

Shaw, P. (1998). *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems*, Berlin, Heidelberg.

Solomon, M. M. (1987). Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research, 35*(2), 254-265.

Toth, P., & Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics, 123*(1), 487-512.

APPENDIX A

R CODES

1- Elimination of data entries with missing values and misplaced ones

```
table(delivery$TN)

table(delivery$XDN)

na.omit(delivery)

delivery <- delivery[!(delivery$TN %in%

c("BALÇOVA","BAYRAKLI","ÇANKAYA","ÇAYIROVA","DILOVASI","GEBZE","IZMIT","O
RHANGAZI")),]

delivery <- delivery[!(delivery$XDN %in%
                        c("GEBZE","IZMIR","BURSA")),]
```

2- Creation of "Delay Status"

```
difference <- difftime(delivery$DDP,delivery$DD,units=c("hours"))

delivery$DeliveryStatus <- difference

delivery$DeliveryStatus <- ifelse(delivery$DeliveryStatus>0,"On time","Delayed")
```

3- Creation of “Time Slot”

```
times <- strftime(delivery$DD, format = "%H:%M:%S")

delivery$TimeSlot <- times

delivery$TimeSlot <- ifelse(qs$Hour < "18:00:00" & qs$Hour>"13:00:00","noon",
ifelse(qs$Hour > "18:00:00","night","morning" ))
```

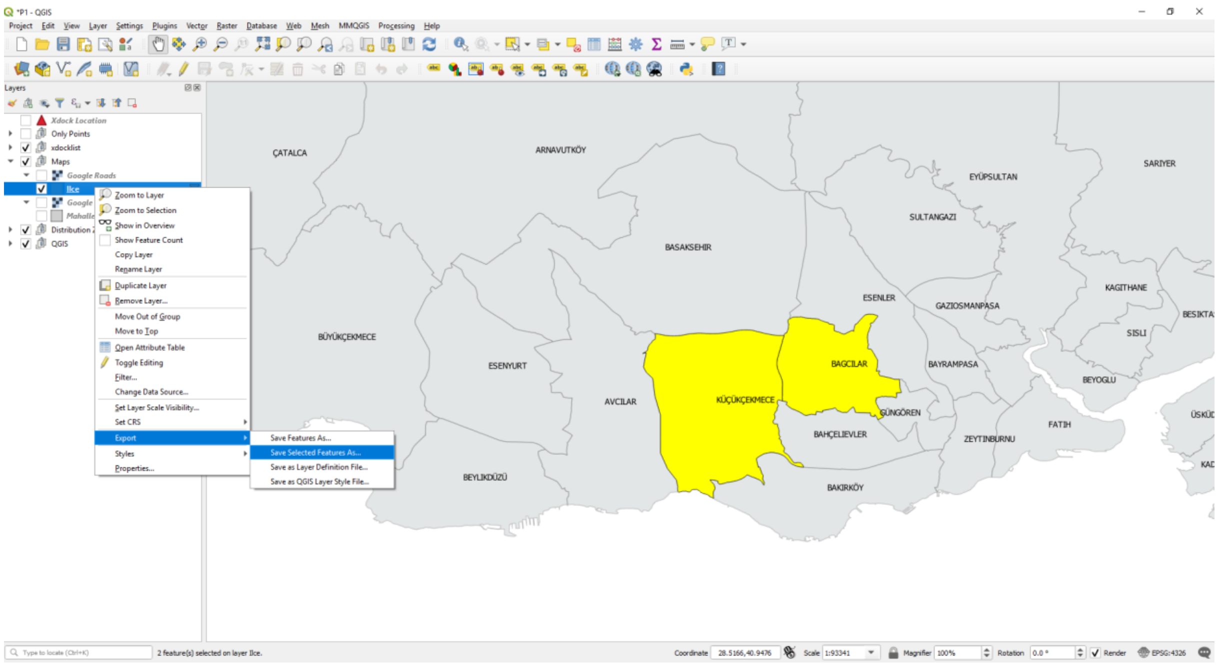
4- Grouping of deliveries by the time slot and by the distinct courier ids

```
by_slot <- D1 %>% group_by(C.ID)

D1C <- by_slot_d1 %>%

summarise(morning=sum(TimeSlot=="morning"),noon=sum(TimeSlot=="noon"),evening=s
um

(TimeSlot=="evening"))
```

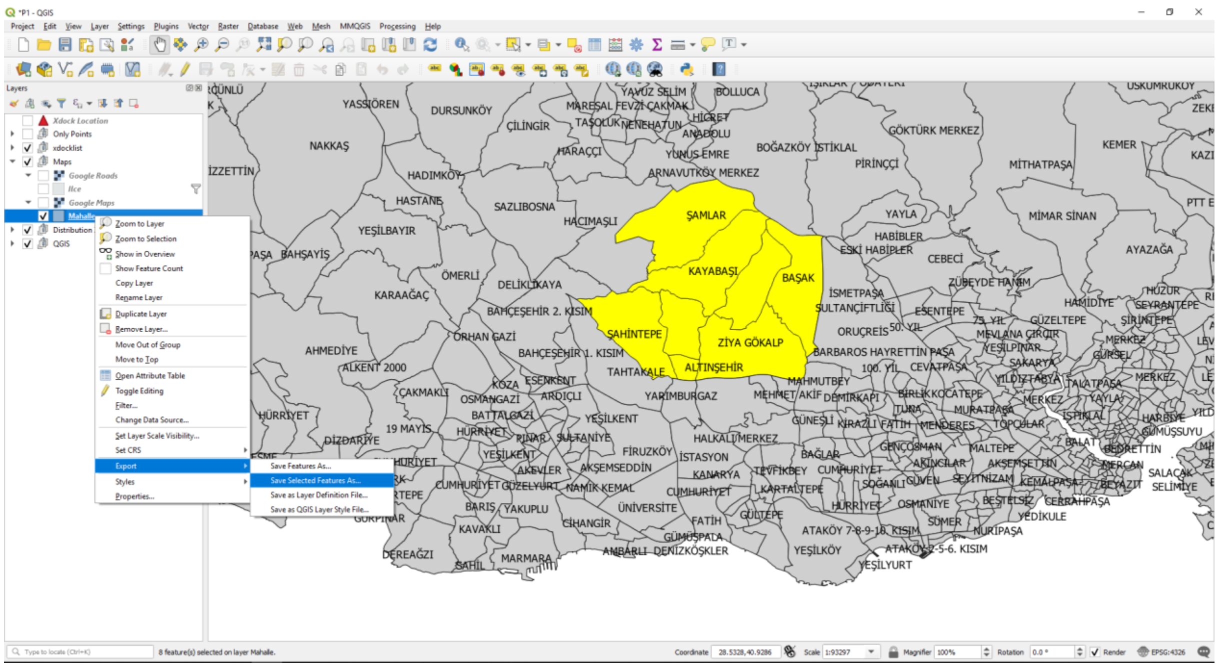
APPENDIX B

QGIS

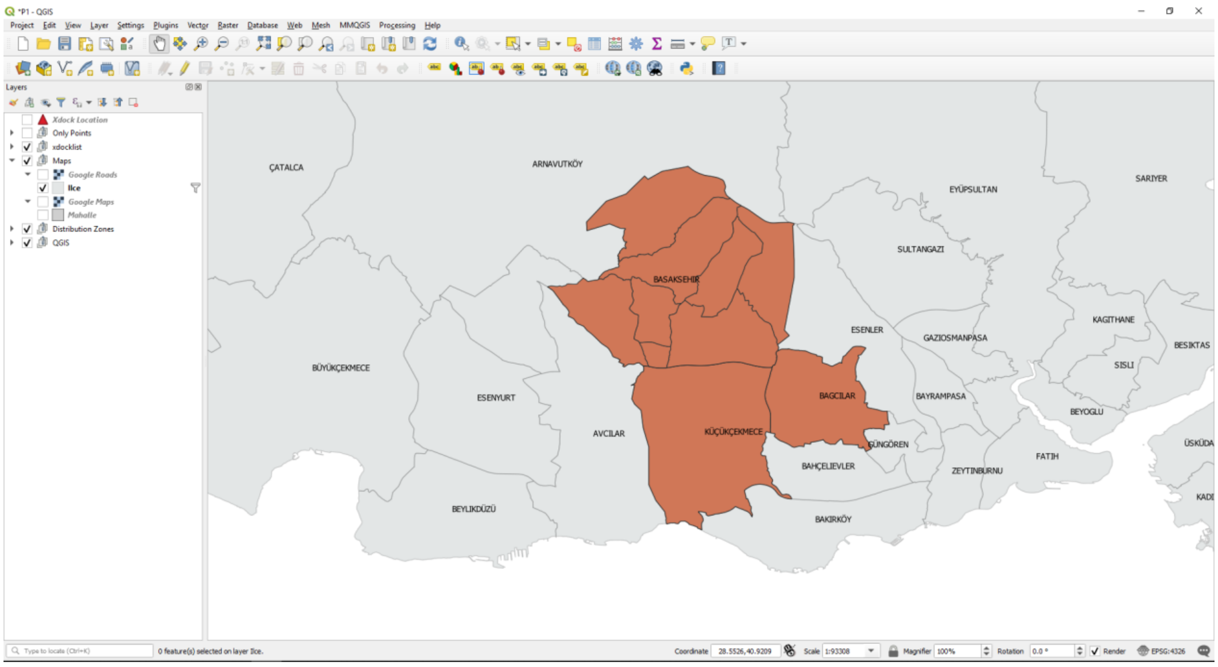
1- Partition into distribution areas of crossdocks

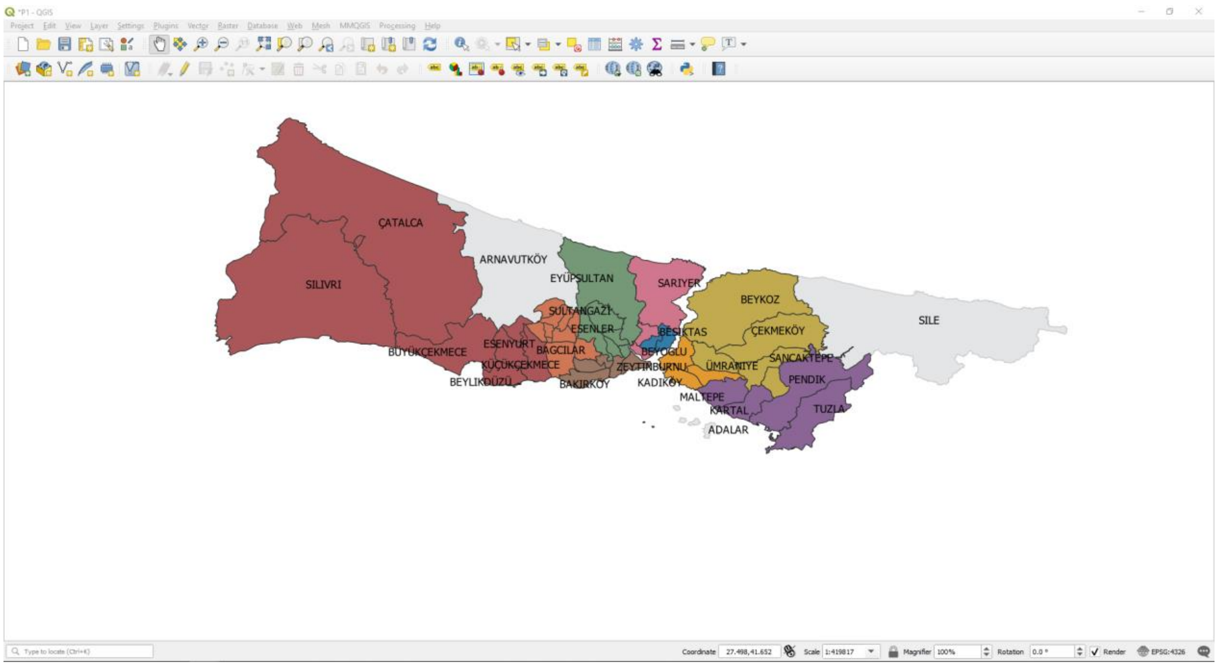


2 feature(s) selected on layer Ilce.

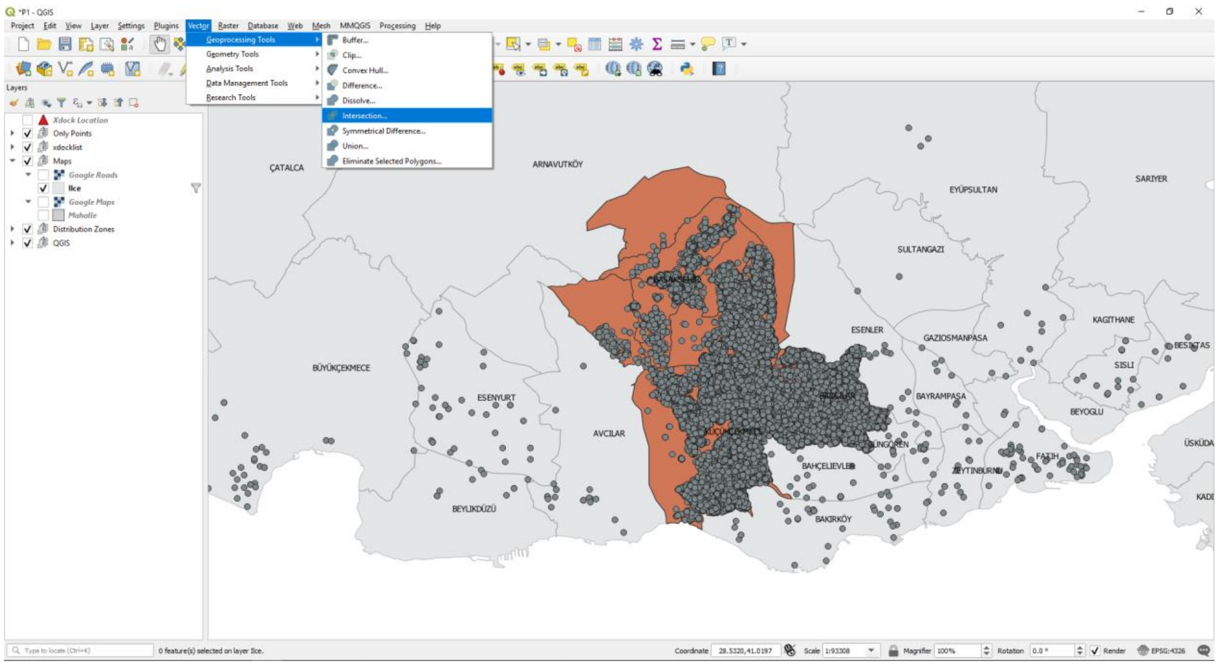


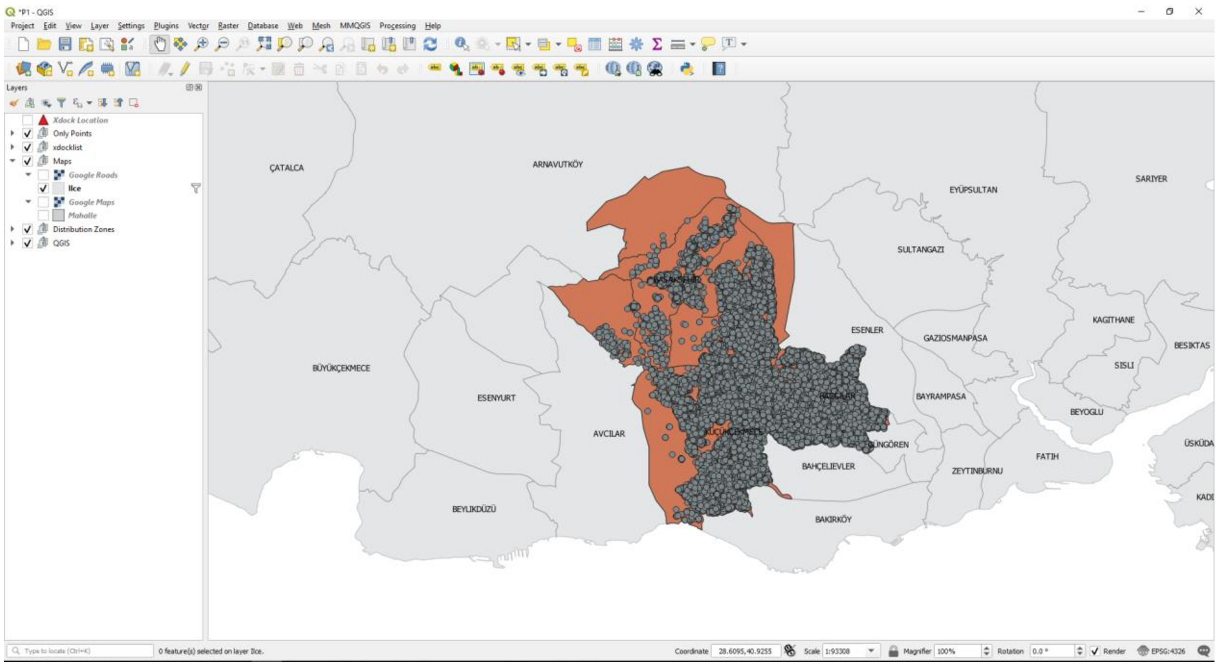
8 feature(s) selected on layer Mahalle.


*P1 - QGIS
Layers
Xdock Location
Only Points
xdocklist
Maps
Google Roads
Ilce
Google Maps
Mahalle
Distribution Zones
QGIS
ÇATALCA
ARNAVUTKÖY
EYÜPSULTAN
SARIYER
SULTANGAZI
BASAKSEHIR
ESENLER
GAZIOSMANPASA
KAGITHANE
BESIKTAS
SISLI
BÜYÜKÇEKMECE
ESENYURT
BAGCILAR
BAYRAMPASA
BEYOGLU
AVCILAR
KÜÇÜKÇEKMECE
GÜNGÖREN
ÜSKÜDA
FATIH
BAHÇELIEVLER
ZEYTINBURNU
KADI
BEYLIKDÜZÜ
BAKIRKÖY
0 feature(s) selected on layer Ilce.
Coordinate 28.5526,40.9209
Scale 1:93308
Magnifier 100%
Rotation 0.0 °
Render
EPSG:4326


*P1 - QGIS
ÇATALCA
SILIVRI
ARNAVUTKÖY
EYÜPSULTAN
SARIYER
BEYKOZ
SILE
SULTANGAZI
ESENLER
BESIKTAS
ÇEKMEKÖY
BÜYÜKÇEKMECE
ESENYURT
BAGCILAR
BEYOGLU
SANCAKTEPE
KÜÇÜKÇEKMECE
ZEYTINBURNU
ÜMRANIYE
BEYLIKDÜZÜ
BAKIRKÖY
KADIKÖY
PENDIK
MALTEPE
KARTAL
TUZLA
ADALAR
Coordinate 27.498,41.652
Scale 1:419817
Magnifier 100%
Rotation 0.0 °
Render
EPSG:4326

2- Geoprocessing tools for removal of data entries with wrong GPS coordinates





APPENDIX C

TABLES FOR THE PRIORITIZATION SCENARIOS

1. Second Scenario involving 40 deliveries and one courier

D1			D2			D3			D4			D5		
ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD
34	0	0	71	0	0	147	0	0	188	0	0	236	0	0
29	0	0	53	0	0	141	0	0	187	0	0	227	0	0
35	0	0	46	0	0	140	0	0	185	0	0	225	0	0
			45	0	0	130	0	0	173	0	0	223	0	0
			43	0	0	124	0	0	167	0	0	219	0	1
						71	0	1	163	0	0	218	0	0
						43	0	1	162	0	0	207	0	0
									141	0	1	203	0	0
									140	0	1	185	1	1
												167	1	1

2. Third Scenario involving 80 deliveries and two couriers

D1			D2			D3			D4			D5		
ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD
20	0	0	142	0	0	177	0	0	293	0	0	376	0	0
59	0	0	105	0	0	186	0	0	257	0	0	359	0	0
49	0	0	131	0	0	166	0	0	306	0	0	336	0	0
19	0	0	109	0	0	176	0	0	244	0	0	323	0	0
28	0	0	106	0	0	201	0	0	245	0	0	372	0	0
31	0	0	122	0	0	220	0	0	282	0	0	340	0	1
48	0	0	104	0	0	171	0	0	307	0	0	325	0	1
			126	0	0	237	0	0	308	0	0	387	0	0
			86	0	0	192	0	0	248	0	0	388	0	0
			121	0	0	168	0	0	313	0	0	321	0	0
			96	0	0	188	0	0	278	0	0	367	0	0
			28	0	1	211	0	2	171	0	1	395	0	0
						109	0	1	188	0	1	329	0	0
						126	0	1	192	0	1	334	0	0
						131	0	1	220	0	1	257	0	1
												282	0	1
												306	0	1

3. Fourth Scenario invoving 80 deliveries and two couriers

D1			D2			D3			D4			D5		
ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD
54	0	0	160	0	0	237	0	0	305	0	0	381	0	0
22	0	0	158	0	0	234	0	0	296	0	0	382	0	0
65	0	0	144	0	0	217	0	0	290	0	0	387	0	0
33	0	0	143	0	0	216	0	0	275	0	0	341	0	0
32	0	0	123	0	0	209	0	0	267	0	1	209	0	1
9	0	0	122	0	1	208	0	0	266	0	0	339	0	0
77	0	0	101	0	0	206	0	0	260	0	0	345	0	0
78	0	0	95	0	0	203	0	0	258	0	0	363	0	0
			83	0	0	184	0	0	246	0	0	335	0	0
			65	0	1	158	0	1	243	0	0	394	0	0
			54	0	1	144	0	1	242	0	0	346	0	0
			22	0	1	143	0	1	241	0	0	328	0	0
						123	0	1	234	0	1	366	0	0
						95	0	1	217	0	1	373	0	0
						83	0	1	216	0	1	372	0	0
									209	0	1	399	0	0
									206	0	1	260	0	1
												296	0	1
												290	0	1

4. Fifth Scenario involving 80 deliveries and two couriers

D1			D2			D3			D4			D5		
ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD	ID	DT	DD
46	0	0	99	0	0	215	0	0	258	0	0	321	0	0
32	0	0	106	0	0	208	0	0	306	0	0	327	0	0
15	0	0	109	0	0	186	0	0	255	0	0	338	0	0
24	0	1	125	0	1	219	0	0	242	0	1	339	0	0
			146	0	0	196	0	1	267	0	0	345	0	0
			89	0	1	202	0	0	244	0	0	347	0	0
			96	0	0	189	0	0	300	0	0	348	0	0
			46	0	1	216	0	1	319	0	1	351	0	0
			32	0	1	169	0	0	247	0	0	353	0	0
						109	0	1	261	0	0	356	0	0
						99	0	1	294	0	0	359	0	0
						96	0	1	256	0	1	367	0	1
									286	0	0	372	0	1
									216	0	2	373	0	0
									186	0	1	380	0	0
									169	0	1	386	0	0
									219	0	1	395	0	0
												300	0	1
												244	0	1
												267	0	1