

MODELING STATIC AND DYNAMIC DIAL-A-RIDE PROBLEM



M.Sc. THESIS

Dilek EKİZ

Department of Computer Engineering

Computer Engineering Programme

MARCH 2019

MODELING STATIC AND DYNAMIC DIAL-A-RIDE PROBLEM



M.Sc. THESIS

Dilek EKİZ
(504141505)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Sanem SARIEL

MARCH 2019

**MÜŞTERİ ROTALAMA PROBLEMİNİN
STATİK VE DİNAMİK OLARAK MODELLENMESİ**

YÜKSEK LİSANS TEZİ

**Dilek EKİZ
(504141505)**

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Assoc. Prof. Dr. Sanem SARIEL

MART 2019

Dilek EKİZ, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 504141505 successfully defended the thesis entitled “MODELING STATIC AND DYNAMIC DIAL-A-RIDE PROBLEM”, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Assoc. Prof. Dr. Sanem SARIEL**
Istanbul Technical University

Jury Members : **Assoc. Prof. Dr. Sanem SARIEL**
Istanbul Technical University

Assoc. Prof. Dr. Ayşe Şima Uyar
Istanbul Technical University

Assoc. Prof. Dr. Fatih Amasyalı
Yıldız Technical University

Date of Submission : **06 March 2019**

Date of Defense : **21 March 2019**





To my family,



FOREWORD

First of all, i would like to indicate that i am very proud to be a part of Istanbul Technical University (ITU). During the years i had in ITU, i have always been feeling myself very lucky to be here.

I could not complete this thesis without the support and assistance of my advisor, Assoc. Prof. Dr. Sanem Sariel. She is one of the most inspiring women i have ever met in my personal and academical life. I would like to thank her for the endless and unlimited understanding, continuous support in any conditions and inspiration. The time that we spent together were more than just being a student and an advisor. I am so grateful that i could meet with Dr. Sariel and have benefited from her knowledge and experience.

I specifically thank to my parents, Aysel and Yılmaz Arıkan, brother, Onur Baran Arıkan and husband, Serdar Ekiz for their support, patience and belief in me and my works. They always supported me without any doubt of failure. I am very thankful them to have their love and make my soul happy.

March 2019

Dilek EKİZ

TABLE OF CONTENTS

	<u>Page</u>
FOREWORD.....	ix
TABLE OF CONTENTS.....	xi
ABBREVIATIONS	xiii
SYMBOLS.....	xv
LIST OF TABLES	xvii
LIST OF FIGURES	xix
SUMMARY	xxi
ÖZET	xxv
1. INTRODUCTION	1
1.1 Purpose of Thesis	2
1.2 Literature Review	3
1.2.1 Static dial-a-ride problem	3
1.2.2 Dynamic dial-a-ride problem	5
1.3 Contribution of the Thesis	8
1.3.1 Modeling the problem	8
1.3.2 Static dial-a-ride problem	8
1.3.3 Dynamic dial-a-ride problem	8
2. PROBLEM STATEMENT	11
2.1 Purpose	11
2.2 Dial-a-Ride Problem.....	11
2.2.1 Objective function	13
2.2.2 Static and dynamic optimization models of DARP	14
2.3 Notation	15
2.4 Formulation	16
2.4.1 Static dial-a-ride problem	16
2.4.1.1 Mixed integer programming formulation	17
2.4.1.2 Heuristic formulation.....	18
2.4.2 Dynamic dial-a-ride problem	20
3. STATIC DIAL-A-RIDE PROBLEM	21
3.1 Definition.....	21
3.2 Methods	22
3.2.1 Exact methods	22
3.2.1.1 Linear programming	23
3.2.1.2 Mixed integer programming	23
3.2.2 Linear and mixed integer programming solvers.....	23
3.2.3 Heuristic methods.....	24
3.2.3.1 Cluster-first route-second approach.....	24

3.2.3.2 Genetic algorithm	24
3.2.3.3 Nearest neighbor heuristic	25
3.3 Implementation.....	25
3.3.1 MIP implementation.....	26
3.3.2 Heuristic implementation	28
3.3.2.1 Clustering via genetic algorithm.....	28
3.3.2.2 Routing via space-time nearest neighbor heuristic algorithm	30
4. DYNAMIC DIAL-A-RIDE PROBLEM	33
4.1 Definition	33
4.2 Methods	34
4.3 Implementation.....	34
4.3.1 Overview of the solution method	34
4.3.2 Components of the solution mechanism.....	37
4.3.2.1 Requests.....	37
4.3.2.2 Vehicles.....	37
4.3.2.3 Event manager	38
4.3.2.4 Decision algorithm.....	38
4.3.2.5 Optimization algorithm.....	38
4.3.3 Implementation of method	39
5. EMPIRICAL ANALYSIS and RESULTS.....	41
5.1 Data.....	41
5.2 Empirical Analysis	42
5.2.1 Static dial-a-ride problem	42
5.2.1.1 Experiment setup	43
5.2.2 Dynamic dial-a-ride problem	43
5.2.2.1 Experiment setup	43
5.3 Results	44
5.3.1 Static dial-a-ride problem.....	44
5.3.2 Dynamic dial-a-ride problem	46
5.3.2.1 Case study on dynamic DARP.....	46
6. CONCLUSION and DISCUSSION	49
6.1 Future Work.....	50
REFERENCES.....	53
APPENDICES.....	57
APPENDIX A.1	59

ABBREVIATIONS

B2C	: Business to Customer
DARP	: Dial-a-Ride Problem
TSPTW	: Traveling Salesman Problem with Time Windows
VRPPD	: Vehicle Routing Problem with Pick-up and Delivery
MIP	: Mixed Integer Programming
NNH	: Nearest Neighbor Heuristic
VRP	: Vehicle Routing Problem
TSP	: Traveling Salesman Problem
LP	: Linear Programming
MILP	: Mixed Integer Linear Programming
GA	: Genetic Algorithm
TSP	: Traveling Salesman Problem
VRP	: Vehicle Routing Problem
ADARTW	: Advanced Dial-a-Ride with Time Windows
GRASP	: Greedy Randomized Adaptive Search Procedure



SYMBOLS

n	: The number of customer requests
$P = \{1, \dots, n\}$	: Pick-up coordinates of requests
$D = \{n+1, \dots, 2n\}$	: Drop-off coordinates of requests
$N = P \cup D \cup \{0, 2n+1\}$	: Pick-up, drop-off, origin and destination coordinates
A	: Arcs connecting coordinates
$G = (N, A)$	: Directed graph on which the problem is constructed
K	: Set of carriers
Q_k	: Load amount allowed for carrier k
T_k	: Ride time limit for the carrier k
q_i	: Amount loaded by a carrier at node i
d_i	: Service duration at node i
$[e_i, l_i]$	: Time window for node i
L	: Ride time limit for a request
c_{ij}	: Cost of traveling between the nodes i and j
t_{ij}	: Required time to travel between the nodes i and j
V	: Set of carriers used in the solution satisfying $V \subset K$
m	: Number of carriers used in solution, i.e. $ V =m$
x_{ij}^k	: Variable taking the value 1, if carrier k is going from node i to node j ; 0 if not
B_i^k	: Time point that carrier k visits node i
Q_i^k	: Amount of load left in the carrier's vehicle after it visits node i
L_i^k	: Total ride time of request i by carrier k
w_1	: Weight on the customer transportation time
w_2	: Weight on the number of vehicles used
w_3	: Weight on the route duration
w_4	: Weight on the customer excess ride time
w_5	: Weight on the waiting time for a customer



LIST OF TABLES

	<u>Page</u>
Table 5.1 : Sample data set.....	42
Table 5.2 : The results for the generated instances.	45
Table 5.3 : Sample data set for the test of dynamic DARP algorithm.	47





LIST OF FIGURES

	<u>Page</u>
Figure 3.1 : Chromosome representation used in GA.....	29
Figure 3.2 : The average and best cost values change with the number of generations.	30
Figure 3.3 : The constructed route for carrier 1 from Figure 3.1.....	31
Figure 4.1 : Solution architecture of dynamic DARP implementation.	35





MODELING STATIC AND DYNAMIC DIAL-A-RIDE PROBLEM

SUMMARY

As consumers expect faster and cheaper services from the retailers, same day delivery services gained importance in the last years. The same-day delivery companies ensure that the user has the fastest and the cheapest delivery while optimizing the costs and delivery process. There exist different concepts of delivery services that are identified by the main purpose of the service. The main purpose could be the transportation of the disabled and handicapped people as well as to deliver the urgent needs of the people.

This thesis is based on a real-world scenario and it deals with modeling the scheduling process of an on-demand instant delivery start-up company called Kapgel. Located in Istanbul, Turkey, Kapgel aims to pick-up the products that are ordered by the customers via a mobile application from a certain location and deliver them to a certain location. Orders arrive ideally online and processed as soon as they are generated. Transportation of the products are performed by the vehicles owned by the company. Vehicles start and end their routes at a depot location. Products ordered by the customers are first picked-up from a point such as shop or coffee house and they are dropped-off to a requested delivery point. The company claims to deliver the orders in one hour at a fixed price. The problem that is to deliver products to the customers from their corresponding pick-up points to the desired drop-off points within a guaranteed time interval is modeled as a Dial-a-Ride Problem (DARP).

DARP mainly deals with finding the optimal routes for the vehicles while optimizing the costs and user satisfaction and obeying the time and capacity constraints. It has been seen in the literature that there are works on the different versions of DARP. In this thesis, the focus was to work on two versions : multi-vehicle single-objective DARP and multi-vehicle multi-objective DARP. Multi-vehicle single-objective DARP is to find routes for multiple vehicles while optimizing a single objective function such as costs or user satisfaction. Multi-vehicle multi-objective DARP finds also routes for multiple vehicles but optimizes a multi-objective function which includes several different criteria such as costs, user satisfaction, number of vehicles, ride time violation etc.

In this thesis, DARP is formulated as multi-vehicle single-objective DARP to optimize only the costs of the transportation which is the total distances that are travelled by the all vehicles to deliver the all orders in route and it is formulated as multi-vehicle multi-objective DARP to optimize costs and user satisfaction together. User satisfaction is identified by the ride time violations and so the waiting time for the customer. Problem is constrained so that each vehicle starts and ends the route at the depot, orders are delivered ideally in 1 hour, a maximum ride time is applied to all vehicles which makes the drivers not work more than a defined period of time, each vehicle has a maximum capacity and there are certain number of vehicles.

Since a real-world case is investigated in this thesis, DARP is analyzed as static and dynamic DARP. In the static mode of an optimization problem, the information regarding the orders such as pick-up and delivery time and location is known beforehand. The program is not interrupted during the execution time since all the input data is already given and the solution is generated according to the already known orders. In the dynamic mode of an optimization problem, on the other hand, scheduling starts within the arrival of the first request and continues with the arrival of the next requests dynamically. Therefore the optimization algorithm is interrupted at the point that a new request is detected and the new request is adapted into the system to be processed without violating the constraints and optimization of the objectives.

Static DARP is investigated under two different frameworks in this thesis. First, an exact method is implemented to solve static DARP to see if it is adequate to solve the problem in question considering the scale of the real-world scenario. Multi-vehicle single-objective DARP formulation is used and Mixed Integer Programming (MIP) is applied as a Linear Programming (LP) method to solve the static DARP. Second, a heuristic method is implemented to obtain faster solutions using multi-vehicle multi-objective DARP formulation. The problem is modeled heuristically using cluster-first, route-second algorithm. The algorithm solves the scheduling problem as generating first the clusters and then the routes for the clusters. In the clustering part of the algorithm, Genetic Algorithm (GA) is used as a heuristic method and the orders are assigned to different vehicles, i.e. clustered while in the routing part the routes are found for the different clusters that are identified in the clustering part using Nearest Neighbor Heuristic (NNH) algorithm.

To solve the dynamic DARP, the mechanism is modeled so that the program could handle with the dynamic structure of the problem. Since the time becomes an important issue when solving a dynamic optimization problem, an Event Manager that handles with the time is modeled. The time is split into 1-minute intervals to detect and welcome the dynamically arriving requests by the Event Manager. In case a new request is detected, a Decision Algorithm takes the duty to find an available vehicle for this request without violating the optimization. Insertion Heuristic makes a calculation and provides an insertion cost for the request that should be inserted into the scheduling system so that it can be determined which vehicle provides the cheapest cost for the request. Insertion cost is calculated according to the distance between the pick-up point of the new request and the current positions of each vehicle or the location of the new vehicle, if there is any available vehicle. Once the request that has the cheapest insertion cost is assigned into a cluster, Optimization Algorithm finds the best route for the cluster using Nearest Neighbor Algorithm (NNH). Because the location and time information at the time that the insertion cost is calculated is needed to calculate the insertion cost of a new request and assign it to a vehicle, time and location information is kept in the memory in the dynamic mode assuming that the vehicles move with a certain speed without distracting by the environmental factors.

Static and dynamic DARP algorithms are tested on a simulated data that is provided by the company. The data includes 167 requests and all of the requests belong to one day. The results of the exact and heuristic methods of static DARP implementation are compared in terms of objective values and runtime. It has been seen that exact algorithm provides the best cost values but only up to a certain number of requests. After 14 requests, exact method was not able to find feasible solutions in a reasonable

time with the given resources. On the other hand, cost values that are quite close to the ones given by exact algorithm are obtained from heuristic algorithm. Heuristic algorithm was able to solve the all the instances in a very short runtime. The results of the dynamic DARP implementation was satisfactory in terms of modeling the dynamic architecture. The algorithm was able to handle with all the online requests, assign them to a vehicle without violating the requests in transit and do the optimization using heuristic algorithms.

The main focus of this research was to provide a qualitative solution to a real-world problem. The real-world problem is modeled successfully for this purpose as a DARP and several algorithms are implemented and tested. Although the problem of scheduling requests to be picked-up and delivered from predetermined locations by the vehicles is implemented as static and dynamic DARP successfully, improvement of the heuristics and implementing different methods to obtain better solutions is left as future work.





MÜŞTERİ ROTALAMA PROBLEMİNİN STATİK VE DİNAMİK OLARAK MODELLENMESİ

ÖZET

Hızlı teslimat şirketlerinin, müşterilerin en hızlı ve en uygun fiyatlı hizmeti alma taleplerinin artmasına bağlı olarak son yıllarda önem kazandığı görülmektedir. Bahsi geçen, aynı günde sipariş teslimatı yapan şirketler, kullanıcılara en hızlı ve en ucuz teslimatın garantisini vermekle birlikte; müşteri memnuniyetini, teslimat sürecini ve toplam maliyeti optimize etmeyi amaçlamaktadırlar. Amaca yönelik olarak, bu şirketlerin birbirinden farklı konseptlerde hizmet sunduğu görülmektedir. Hızlı teslimat, hasta veya engelli kişilerin bir yerden bir yere ulaştırılması olabileceği gibi acil ihtiyaçların teslimatı da olabilmektedir. Buna göre, hasta veya engelli kişileri bir noktadan başka bir noktaya ulaştırmayı amaçlayan teslimat şirketleri ile sipariş edilen bir ürünün bir noktadan alınarak başka bir noktaya teslimatını amaçlayan şirketler arasında problemin modellenmesi açısından fark bulunmaktadır. Birinci durumdaki bir örnekte, randevu bir gün veya daha öncesinde oluşturulabilmekte, hatta bu servisi sağlayan şirketler en geç bir gün öncesinde randevunun oluşturulmuş olması şartını koyabilmektedirler. Bu durumda optimizasyonun yapılarak rotalamanın oluşturulduğu anda, tüm siparişler önceden biliniyor olmaktadır. İkinci durumda ise, siparişler acil ihtiyaçlara yönelik olduğundan, dinamik olarak yoğunlukla çevrimiçi bir arayüz aracılığı ile alınmakta ve siparişin oluşturulduğu andan belirli bir süre sonrasına kadar teslimatın tamamlanması gerekmektedir. Bu durumda tasarlanacak olan optimizasyon algoritmasının dinamik olarak gelen siparişleri karşılayacak ve anında çözümler üretebilecek bir algoritma olması gerekmektedir.

Bu çalışma bir gerçek dünya problemini çözmek üzere yapılmış olup Kapgel adlı şirketin kullanımına yönelik müşterilerin rotalanması probleminin çözülmesi amaçlanmıştır. İstanbul'da kurulmuş olan bir şirket olan Kapgel, müşteriler tarafından bir mobil uygulama aracılığı ile alınan siparişlerin belirli bir noktadan alınarak müşterinin belirttiği başka bir noktaya teslimatını yapmaktadır. Siparişler çevrimiçi olarak alınmakta ve alındığı andan itibaren teslimat süreci başlamaktadır. Ürünlerin teslimatı şirkete ait olan araçlarla yapılmaktadır. Her aracın kendi rotasına sabit bir depo lokasyonundan başlayıp rotayı aynı lokasyonda tamamlaması gerekmektedir. Sipariş edilen ürünler öncelikle bir servis noktasından alınır ve sonra müşterinin belirlediği adrese teslim edilir. Şirket ürünlerin teslimatının 1 saat içerisinde sabit bir fiyata yaptığının garantisini vermektedir. Proje kapsamında, Kapgel için kuryelerin lojistik olarak siparişleri alıp dağıtabilmesini sağlayacak şekilde yönlendirilmesini gerçekleştirecek olan yazılımın geliştirilmesi hedeflendiğinden, çoklu araç rotalama problemleri incelenmiştir. İncelenen problemlerdeki kısıtlar ve Kapgel'in ihtiyaçları göz önüne alınarak, Müşteri Rotalama Problemi (Dial-a Ride Problem, DARP) model olarak kullanılmıştır.

Müşteri rotalama probleminde (Dial-a-Ride Problem, DARP), kuryelerin siparişleri belirli bir başlangıç noktasından (pick-up noktası) belirli bir varış noktasına (drop-off

noktası) verilen bir zaman penceresi içerisinde taşıması formüle edilir. Amaç, tüm kısıtlar sağlanırken, araçlar için en düşük maliyetli rotaları bulan bir model tasarlamaktır. Problem, zaman penceresi kısıtı içerdiğinden dolayı Başlangıç ve Varış noktası olan Araç Rotalama Problemi'nin (Vehicle Routing Problem with Pick-up and Delivery, VRPPD) ve ayrıca Zaman Penceresi olan Gezgin Satıcı Problemi'nin (Traveling Salesman Problem with Time Window, TSPTW) genelleştirilmiş bir versiyonudur. Bu sebepten, NP-Hard bir problem olarak sınıflandırılır. Literatürde, müşteri rotalama probleminin farklı versiyonları üzerine çalışmalar yapıldığı görülmüştür. Bu çalışmada, çoklu-arac tek-objektif müşteri rotalama problemi (multi-vehicle single-objective DARP) ve çoklu-arac çoklu-objektif müşteri rotalama problemleri incelenmiştir. Çoklu-arac tek-objektif müşteri rotalama probleminde maliyet veya müşteri memnuniyeti gibi yalnız tek bir kriterin optimize edilmesi amaçlanır. Çoklu-arac çoklu-objektif müşteri rotalama probleminde ise maliyet, müşteri memnuniyeti, araç sayısı, sürüş süresi ihlali gibi birden çok kriterin içerildiği bir objektif fonksiyon optimize edilir.

Bu çalışmada müşteri rotalama problemi, tüm kuryelerin siparişleri teslim etmek için katettiği mesafelerin toplamını ifade eden, maliyetleri optimize edilecek şekilde tek-objektif fonksiyonlu olarak ve maliyet ve müşteri memnuniyeti birlikte optimize edilecek şekilde çoklu-objektif fonksiyon kullanılarak iki ayrı şekilde formüle edilmiştir. Müşteri memnuniyeti, rota ve sipariş süresi ihlalleri baz alınarak hesaplanmıştır. Problem kısıtları; her aracın kendi rotasına başlangıç noktasından başlayıp rotayı başlangıç noktasında bitirmesi, siparişlerin ideal durumda bir saat içerisinde teslim edilmesi, tüm araçlar için belirli bir maximum sürüş süresi uygulanması, her aracın bir kapasitesinin olması ve belirli bir sayıda aracın kullanılabilir olmasından oluşmaktadır.

Bu tezin amacı, bir gerçek dünya sorununu çözmek olduğundan, müşteri rotalama problemi statik ve dinamik olmak üzere iki ayrı alt başlıkta incelenmiştir. Statik modellemede, rotalama için gerekli olan tüm veriler önceden bilinmektedir. Program, çalışma zamanı sırasında bölünemez ve yeni bir sipariş mevcut rotalamaya eklenemez. Dinamik modellemede ise, rotalama ilk siparişin gelmesi ile başlar ve peşisıra dinamik olarak yeni siparişlerin gelmesi ile devam eder. Program, çalışma zamanında yeni bir siparişin oluşturulması ile bölünür, sipariş mevcut kuryelerin durumuna göre herhangi bir kısıtı ihlal etmeyecek şekilde bir araca atanır ve rotalar tekrar optimize edilir.

Statik müşteri rotalama problemi iki farklı yöntem ile incelenmiştir. Öncelikle, lineer programlama ile Karışık Tamsayı Programlama (Mixed Integer Programming) yöntemi kullanılmıştır. Bu yöntemde problem, çoklu-arac tek-objektif Müşteri Rotalama Problemi olarak formüle edilmiştir. İkinci olarak, problem çoklu-arac çoklu-objektif Müşteri Rotalama Problemi olarak formüle edilmiş ve sezgisel yöntemlerle problemin çözülmesi amaçlanmıştır. Bu amaçla, problem önce kümelerle ayırma, sonra rotaların oluşturulması (cluster-first, route-second) yöntemi kullanılarak modellenmiştir. Kümelerin oluşturulması için genetik algoritma kullanılmış, oluşturulan kümelerin rotilanması için ise en yakın komşu algoritması kullanılmıştır.

Dinamik müşteri rotalama problemini çözmek için öncelikle problemin dinamik yapısına uygun şekilde bir modelleme yapılmış ve mimari oluşturulmuştur. Dinamik modellemede en önemli faktör zaman olduğundan, zaman ile ilgili sorunları çözecek şekilde bir Olay Yöneticisi tasarlanmıştır. Dinamik olarak oluşturulan siparişlerin tespit edilebilmesi için zaman, Olay Yöneticisi tarafından 1 dakikalık aralıklara

bölünmüştür. Her 1 dakikada yeni bir sipariş olup olmadığı kontrol edilmekte, yeni bir siparişin tespit edilmesi ile birlikte Karar Algoritması devreye girerek yeni siparişin atanacağı uygun bir aracı, mevcut atamaları bozmadan ve kısıtları ihlal etmeden bulmaya çalışmaktadır. Ekleme Algoritması ise yeni siparişin tüm araçlar için ve eğer boşta bekleyen araç var ise bu araç için maliyetini hesaplamaktadır. Maliyet hesabı Ekleme Algoritması tarafından, yeni sipariş ile teslimatta olan kuryelerin o andaki mevcut lokasyonları ve depoda beklemekte olan araçlara olan mesafelerin hesaplanması ile yapılır. En ucuz ekleme maliyetini sağlayan araca yeni siparişin ataması yapılır veya mevcutta depoda beklemede olan araç varsa ve yeni araç eklenmesi daha ucuz ise sisteme yeni bir araç eklenir. Yeni siparişin ataması yapıldıktan sonra, Optimizasyon Algoritması, değişiklik olan küme için optimum rotayı tekrar bulmak üzere, statik algoritmanın rotalama kısmında olduğu gibi, en yakın komşu algoritmasını kullanır. Tüm hesaplamaların yapılabilmesi ve yeni siparişlerin sisteme eklenebilmesi, kuryelerin mevcut lokasyonlarının ve mevcut zamanın bilinmesini gerektirdiğinden; lokasyon ve zaman bilgisi hafızada tutulur ve sürekli olarak ideal durumların yaşandığı ve kuryelerin herhangi bir dış etken tarafından müdahale edilmeden sabit bir hızla planlanmış rotalarında hareket ettiği varsayılarak, güncellenir.

Statik ve dinamik müşteri rotalama algoritmaları, şirket tarafından temin edilen veri üzerinde test edilmiştir. İlgili veri 167 adet sipariş içermektedir ve tüm siparişler tek bir güne aittir. Statik problemi çözmek için kullanılan lineer programlama ve sezgisel yöntemler tarafından elde edilen sonuçlar, maliyet ve programın çalışma zamanı baz alınarak karşılaştırılmıştır. Karşılaştırma sonucunda lineer programlama ile elde edilen sonuçların en iyi maliyet sonucunu verdiği ancak belli bir müşteri sayısından büyük veri içeren örnekler için, kabul edilebilir bir süre zarfında verilen kaynaklar ile çözüm bulamadığı görülmüştür. Bu yöntem ile ancak 14 siparişe kadar olan veri için çözüm bulunabilmiştir. Sezgisel yöntemden elde edilen sonuçların ise maliyet bakımından lineer yöntemle oldukça yakın sonuçları gerçek-dünya probleminin kabul edebileceği çalışma zamanında üretebildiği görülmüştür. Sezgisel yöntem, verilen tüm veri örnekleri için çözüm üretebilmiştir. Problem dinamik olarak ise başarılı bir şekilde modellenmiş, çevrimiçi olarak oluşturulan siparişler anında rotalamaya dahil edilerek optimize rotalar oluşturulmuştur. Geliştirilen model ve algoritma, oluşturulan tüm yeni siparişleri, mevcutta teslimat sürecinde olan siparişleri bozmadan ve kısıtları ihlal etmeden, sezgisel yöntemler kullanarak sisteme ekleyebilmiştir.

Bu çalışmanın amacı, bir gerçek dünya problemine uygulanabilir çözümler üretmektir. Problemin kısıtları ve ilgili şirketin ihtiyaçları göz önüne alınarak, problem Müşteri Rotalama Problemi olarak modellenmiş, hedeflere ulaşabilmek amacıyla statik ve dinamik olarak iki farklı alt başlıkta incelenmiş ve çözüm yöntemleri uygulanmıştır. Uygulanan yöntemlerden başarılı sonuçlar elde edilmesine rağmen gelecekte, daha iyi sonuçlar elde etmek amacıyla, sezgisel yöntemlerin geliştirilmesi ve farklı yöntemlerin denenmesi amaçlanmıştır.



1. INTRODUCTION

Same day delivery services have increased their popularity since they mean faster B2C delivery, more time for users and easier shopping. Companies such as Postmates¹, Google Express², Instacart³, Amazon Prime Now⁴, Uber Delivery⁵ and many others claim that they will pick up the desired product from the desired location at an extra cost and deliver it within a certain period of time to any desired location. Most of these companies serve in a certain area, mostly in city centers and each of them usually work for certain areas. Postmates, one of the fastest growing instant delivery service in America, delivers all kind of fast food products while Amazon Prime Now delivers only its own products within 2 hours. Another similar company, called Drizly⁶, cooperates with the stores that sells alcoholic beverages and it delivers any alcoholic beverage within an hour. To give a local example, the company called "Getir"⁷ can be viewed. Founded in 2015, Getir has grown %400 in its first year. Getir has its own depots and delivers products in many different categories from food to cleaning, from baby products to pet products. Getir claims that it is the only company that can deliver products in only 10 minutes. Such companies usually have their own mobile applications and orders can be taken online through these applications.

Since these instant delivery companies ensure the user for the fastest and cheapest delivery, optimization of costs and time are becoming important. Having their own vehicle networks, instant delivery companies need to minimize their costs while they also need to minimize the time for delivery in order to provide a service as they claimed and prevent the user dissatisfaction. These conditions require that the processes of the mentioned companies to be modeled as an optimization problem. Because different companies have different behaviors and their own specialties, each can be taken as a

¹<https://postmates.com>

²<https://express.google.com>

³<https://www.instacart.com>

⁴<https://primenow.amazon.co.uk/>

⁵<https://www.uber.com/en-GB/drive/delivery/>

⁶<https://drizly.com>

⁷<https://getir.com/>

different problem and designed with a different model according to their objectives and constraints.

This project deals with modeling the transportation process of a start-up instant delivery company named Kapgel⁸ located in Turkey in order to optimize costs and customer satisfaction. Kapgel's greatest aim is to deliver the urgent needs and products that is needed in difficult situations as soon as possible while optimizing costs and fulfilling the orders under the predefined constraints. Kapgel does not have its own depot, rather it picks up an order from a pick-up point e.g. shop, coffee house etc. and delivers it to a requested delivery point. It serves in a certain area of Istanbul, mostly in Beşiktaş, Şişli, Kadıköy, and makes fast delivery of certain products that can be seen through its own mobile application. The company promises to deliver any product that exists in the catalog within 1 hour for a certain fee. Once an order is created via the application, process is started. The most appropriate vehicle picks up the order from the pick-up location and deliver it to the destination location. Kapgel's vehicles with a certain capacity are working at certain times of the day.

This study can be classified as to have three main objectives : First is to model the real-world problem and adapt it to a mathematical formulation; second is to apply the optimization methods to the problem and the third is to test the results on the given data.

1.1 Purpose of Thesis

In this project, optimization of costs and user satisfaction for an on-demand instant delivery start-up company is aimed to be realized. Problem that is to deliver products to the customers from their corresponding pick-up points to the desired drop-off points within a guaranteed time interval [1] is modeled as a Dial-a-Ride Problem (DARP). Main goal from the company's point of view is to optimize the costs and user satisfaction while satisfying the time window, duration, capacity and maximum ride time constraints.

DARP, being a generalization of Traveling Salesman Problem with Time Windows (TSPTW) and Vehicle Routing Problem with Pick-up and Delivery (VRPPD), is an

⁸<http://www.kapgel.com>

NP-Hard problem which solves the problem of finding the best routes in terms of costs and customer satisfaction given aforementioned constraints. In DARP, routes for the vehicles are generated so that requests are taken from customer-specified pick-up points and delivered to the destination points within a period of time, i.e. 1 hour specifically for this project.

Since the purpose of this project is to provide a solution and contribute to a real-world business process, considering that the requests are taken from the customers dynamically, i.e. online, works on real-time solutions to the dynamic DARP as well as static DARP is included within the scope of this project.

In this project, DARP is addressed in two different concepts. First, DARP is considered as a static problem and investigated under two different frameworks. Second, DARP is considered as a dynamic (online) problem.

1.2 Literature Review

The purpose of this section is to give an overview of the background work that is related to the topics covered in this thesis. Dial-a-ride problems might be investigated according to static and dynamic modes. For this reason, works reviewed here are analyzed according to the sub-topics Static Dial-a-Ride Problem and Dynamic Dial-a-Ride problem. In each sub-topic, after an explanation of the method, formerly applied methods are presented as a summary.

1.2.1 Static dial-a-ride problem

A static DARP is to find the optimal solution for an objective function given constraints from all feasible solutions on a set of given data. All the data such as coordinates of pick-up and drop-off points, earliest pick-up times and latest drop-off times are given in advance. None of this information is changed during the time. Method to solve static DARP is applied on this formerly given data. Static DARP can be solved using exact methods, e.g. linear programming, or with heuristic methods. Linear programming methods such as MIP, looks at the whole problem space over all feasible solutions and finds the best solution. Objective function and constraints in exact methods are mostly strict and cannot be relaxed. Therefore, it is not always possible to find the solutions using exact methods in a reasonable time and owned resources. In heuristic methods,

on the other hand, some of the constraints can be softened using relaxation methods in a reasonable manner and objective function is adjusted to comply with the relaxed constraints to get not the best but acceptable solutions in a reasonable time. Heuristic methods can be improved within applied methods and can be combined with linear methods to optimize the results.

Bodin and Sexton (1986) developed an algorithm to solve the multi-vehicle subscriber DARP [2]. The problem that they worked for was to schedule requests which has certain pick-up and delivery locations. In the model, each request had either a certain pick-up time or a delivery time. If the request had a certain pick-up time, it could not be picked-up earlier than this time. If the request had a certain delivery time, it could not be delivered later than this time. Objective of the problem was to minimize the user inconvenience. Bodin and Sexton was proposed the swapper algorithm and space-time heuristic to solve the problem. First the customers are randomly partitioned into the different clusters. Swapper algorithm tries to move the customers in the clusters to obtain better objective values. Space-time heuristic creates routes for each of these sets that include the customers and forms a sequence using Nearest Neighbor Heuristic (NNH). The algorithm was tested on a data that is taken from the Project Mobility Baltimore which aims to automatize the transport of elderly and handicapped people between their homes and hospitals, clinics etc. With the algorithm, 3 minutes spent to solve a 85-customers problem. Since the requests should be done 24 hours before the pick-up time according to the project, as a static DARP, results were satisfying and much better than the manual solution.

Most of the works in the literature for multi-vehicle DARP are based on heuristic solutions. Exact methods are usually applied on single-vehicle DARP [3]. The mathematical formulation in the work of Cordeau and Laporte (2003b) where a tabu search heuristic is applied on DARP is accepted and used as the standard DARP formulation in most of the recent works as well as in this thesis [3]. Cordeau and Laporte defined the DARP formulation, performed relaxation and preprocessing on the formulation and applied branch and cut algorithm to the DARP. Before applying the algorithm, an upper bound is generated using tabu search that is proposed in the former work of Cordeau and Laporte (2003b) [4]. The algorithm is implemented using C++ and IBM ILOG Cplex on a randomly and independently generated data set. It has

been seen that the CPU time and number of nodes in the branch and bound tree are both decreased with the method but authors also specified that this algorithm may not be operable on the large-scaled instances.

Jorgensen, Larsen and Bergvinsdottir (2004) worked on genetic algorithms to solve DARP [5]. The mathematical formulation that is used in the research was from the work of Cordeau and Laporte (2003b) [3]. The classical cluster-first, route second approach is applied on DARP and the algorithm was tested on a data set given by Cordeau and Laporte (2003b) [3].

Ropke with Cordeau and Laporte (2007) published an article to model branch and cut algorithms for pick-up and delivery problems with time windows [6]. Two new formulations are proposed in the article. Tests are performed on an instance with up to 8 vehicles and 96 customers and optimal results are achieved.

1.2.2 Dynamic dial-a-ride problem

A dynamic DARP is to find the optimal solution for an objective function given constraints while the requests are arriving dynamically into the system. Unlike the static case, the algorithm allows to accept new orders during the execution. In the static mode, algorithm starts to run to optimize the objective and create the routes at a time $t=0$ with the existing requests that are known beforehand and no change is allowed until the algorithm terminates and all routes are created. In the dynamic mode, on the other hand, algorithm starts with the existing requests as like in the static mode, but also accepts the newly incoming requests during the execution, take them into the account and updates the routes to provide the optimal solution including the new requests in the routes. Dynamic DARP differs from static DARP in terms of accepting the new orders and being able to update the planned routes according to the new situation. Objective function and the constraints remains the same as in the static mode.

In real world, the difference between static and dynamic DARP remains blurry according to the Borndörfer et al. (1997) [7, 8] for the reason that the requests may be cancelled and so it is easy to include the new requests in the transportation using the static algorithm. Also the dynamic DARP rarely exists in the real world since most of the time all the requests are known in advance [8]. Borndörfer et al. (1997) [7] have developed an algorithm to solve the bus scheduling problem of Telebus Berlin which

is a transportation system for elderly and handicapped people using DARP. Since the Telebus system guarantees the requests to be serviced in case that they are scheduled one day before, it is meaningful to make no distinction between the static and dynamic case of DARP according to the nature of the scenario in practice. Borndörfer et al. (1997) [7] used set partitioning approach including a branch and cut algorithm to solve the problem in two steps. In the first step sets are determined and then the requests are chained according to the sets that are identified in the first step. A version of the system that is developed in this paper is used by Telebus since 1995.

Psaraftis (1980) has worked on static and dynamic modes of single-vehicle DARP in his two-part research [9]. Route is created only for single vehicle for the requests to be picked-up and dropped-off. Objective function to be optimized was a minimization of the weighted sums of total service time and user dissatisfaction in terms of waiting times of users for a delivery. In the second part of the research, dynamic DARP is investigated and solved using the algorithm that is developed in the static part of the same research. Assuming to be at time t during the execution of the program when a new request comes in, algorithm that is used in static mode is executed again to generate a new optimal route including the new request. Since reoptimization of the program to adapt the new request into the system also has an execution time, time origin after the new request arrived starts after some time at the time from the point t . Therefore, the program runs at time t' as it begins from the origin. To adapt the static algorithm to the dynamic mode, some modifications to the algorithm has performed by Psaraftis (1980).

A heuristic algorithm called Advanced Dial-a-Ride with Time Windows (ADARTW) for the advanced-request multi-vehicle many-to-many DARP is introduced by Jaw et al. (1986) [10]. "Advanced-request" means that the orders should be taken a reasonable time before the routing. In the scenario that is used in the research, requests should arrive either with earliest pick-up time or with latest delivery time specified. Only one new request can be added into the route among the all newly-incoming requests according to the algorithm. This insertion is determined by an insertion cost associated to the new requests. Insertion cost considers on two main objectives : the request itself to be assigned to the vehicle and additional cost that will emerge on existing requests in the vehicle in question created by the new request. ADARTW is tested on both real and

simulated data to obtain the performance. Although, there were no other solutions to be compared with the ADARTW, the algorithm is found sufficiently efficient in terms of computational time [10].

Colomi and Righini (2001) described the algorithms that are implemented on their system called DARIA that is developed to solve static and dynamic DARP [11]. They explained in the paper different models of DARP and algorithm that are developed to solve DARP. They used an iterative algorithm based on a local search in order to eliminate the disadvantage of reoptimization. The algorithm includes clustering and routing phase. Although the authors mention in the paper that the algorithms are tested and used in their system DARIA, results are not provided in the paper [8].

Santos and Xavier (2015) worked on dynamic ride and taxi sharing and modeled the problem as DARP [12]. The problem was to make a scheduling for the passengers who want to share taxis. The origin and destination points, earliest pick-up time and latest drop-off time together with maximum accepted delay are given by the customer. In the solution, Santos and Xavier (2015) divided the day by time periods and solved the static DARP for each time period in order to obtain a solution to the dynamic DARP. To solve the static DARP for each time period, greedy randomized adaptive search procedure (GRASP) is used [12]. Algorithm is tested on real data. Also Santos and Xavier (2015) has compared the results to their former work [13] and mentioned that they obtained better results.

Lois and Ziliaskopoulos (2016) have recently developed an online regret-based DARP (OR-DARP) algorithm and tested it on a data provided by municipality in Greece [14]. The algorithm is developed under two submodules : an online insertion algorithm to handle with the online requests, regret optimization algorithm. Original regret algorithm which searches for better options and exchanges the requests depending on their profit is used. The most important part of the algorithm is given in the paper as that the algorithm uses every single idle time as an opportunity to improve the optimization. The algorithm is tested on a real data provided by a demand responsive transit system in Greece.

1.3 Contribution of the Thesis

In this MSc research, the focus is to create a solution for a real-world instant delivery company to optimize the costs and create the optimal routes. Therefore, the main contributions of this research are to model the Kapgel's problem as an optimization problem, applying methods to solve static and dynamic DARP for Kapgel's problem and obtaining usable results.

1.3.1 Modeling the problem

First step of the research is to formulate the Kapgel's problem as an optimization problem. Looking at the objective and the constraints of the problem, DARP is chosen to be adapted and formulated for Kapgel. The DARP formulation from Cordeau [3] is updated to suit the objective and constraints of the Kapgel's problem. DARP is investigated in this thesis as static and dynamic DARP. Different methods are applied to solve static and dynamic DARP accordingly and the algorithms developed are tested on the data provided by the company Kapgel.

1.3.2 Static dial-a-ride problem

Static DARP is applied under two different frameworks as a first step of the implementation of the problem. Considering the scale of the problem, first Mixed Integer Programming (MIP) is applied to see if it can generate solutions up to an acceptable number of requests. Since MIP was not able to solve the instances that company requires, a heuristic method is applied as a second method. The focus when working on the methods for static DARP was to optimize the costs while satisfying the constraints and objectives of the scenario that is provided by the company. For this reason, two different methods applied effectively and the results are compared in terms of objective value and the runtime.

1.3.3 Dynamic dial-a-ride problem

Since the problem investigated in this research is a real-world business problem, requests arrive online during the day by the customer. Therefore, a dynamic

implementation of DARP is also performed in this research to be able to generate online solutions within the arrival of the new requests. Although known methods are applied to solve static DARP, a custom model is developed to solve the dynamic DARP. Solution mechanism is modeled to solve dynamic DARP so that it includes several different components. Algorithms for the different components of the architecture are chosen from the known heuristic methods. The algorithm to solve the dynamic DARP is applied so that it can be used by a real-world problem.





2. PROBLEM STATEMENT

2.1 Purpose

In this chapter, problem that is investigated in this M.Sc. thesis is presented, notation and formulation of the problem based on mainly early research of Jean-François Cordeau and Gilbert Laporte [3] is given.

2.2 Dial-a-Ride Problem

In this chapter, DARP is explained with its characteristics and with the scenario behind. DARP is an NP-Hard problem that can be considered as a generalization of TSPTW and VRPPD [1]. In DARP, orders taken from customers are picked-up from a specified pick-up point and delivered to a specified drop-off point within a period of time, 1 hour in this thesis. The objective of DARP is to minimize travel costs if all of the constraints are set as hard constraints and to minimize both travel costs and user dissatisfaction in case some of the constraints are relaxed, i.e. allowed to be violated. Transportation performed by vehicles having a constant or variable speed of movement. The constraints of DARP can be generalized as time and load constraints. Time starts counting when an order is created by a user. Any order should ideally be delivered within 1 hour once it was created. A maximum ride time applies to all vehicles to ensure that the vehicles work no more than 9 hours. However, each vehicle is allowed to carry some previously determined amount of load since they have a certain capacity. This capacity can be variable depending on the order in other cases. If the system is to transport elderly or handicapped people, it may be the case that some customers need more space. In Kapgel, since only the limited number of orders are transported, orders are assumed to have similar sizes in terms of volume. This made us to set a certain order capacity for the vehicles, e.g. 10 pieces. Vehicles start their routes at the depot and returns back to the depot after completing all the tasks that they are assigned to. Depot is an origin point where all the vehicles start and end their

routes. Although number of available vehicles are given internally to the program, not all of the vehicles are necessarily to be used. Routes are generated only for the used vehicles. Scheduling is done so that the cheapest routes are created. For this reason, it is allowed for a vehicle to go from depot to depot, meaning that the vehicle is not used. Since DARP is a problem to describe a transportation system where the requests are picked-up from some location and delivered to another location, a service time is needed for each request. Within the service time, driver of the vehicle (carrier) will leave the vehicle, pick-up the order and come back to the vehicle to continue with his/her route. In Kapgel, orders are given in advance once it is created in the system by the customer, before the carrier arrives to the pick-up location. For this reason, a 5-minute average service duration associated to each request is determined to be enough. For other usage types, for example transportation of elderly and handicapped people, more or less time can be needed depending on the need of the customer.

DARP is investigated as static DARP and dynamic DARP in this thesis. In the static mode of an optimization problem, all the information, i.e. the requests with their pick-up and drop-off coordinates, the arrival time of the requests and the latest drop-off time of the requests are known in beforehand and the solution is generated according to the known requests. In the dynamic mode, on the other hand, requests arrive within time, i.e. online, and taken into the queue to be included in a solution. Routes are created with the currently existing requests in the dynamic mode and can be adjusted within the arrival of the new requests.

In the static case, DARP is analysed under two different frameworks. First, DARP is implemented as single-objective multi-vehicle DARP [15] using Mixed Integer Programming where the objective is only to minimize costs. All of the constraints are considered to be hard, i.e. time window and capacity violation is not allowed. Second, DARP is implemented as multi-objective multi-vehicle DARP [5] using classical cluster-first, route-second [5] approach. The reason of that multiple objectives are used in the latter case is that since some of the constraints are relaxed, it is needed to add them into the objective function as an expression to be minimized with a penalty. Dynamic DARP is also implemented using the multi-objective multi-vehicle formulation.

Since two different DARP formulation are going to be given in formulation, all the variables belonging to the both are given in notation.

2.2.1 Objective function

The objective of DARP in the literature is either to minimize total routing costs or maximize user satisfaction or both depending on the model of the problem. DARP can be formulated as single-objective or multi-objective. In this thesis, both formulations are used in different implementation methods. If the problem is single-objective, either routing costs or user satisfaction can be used as an objective. In this thesis, single-objective DARP is used in MIP implementation of static DARP. Here the objective function includes only to minimize the total routing costs. If the problem is multi-objective, other optimization criteria can also be added into the objective function. The multi-objective function includes more than one functions to be optimized. This can be formulated as in the Equation 2.1.

$$\text{Min/Max } O(k) = \text{Min/Max } \begin{bmatrix} O_1(k) \\ O_2(k) \\ \dots \\ O_n(k) \end{bmatrix} \quad (2.1)$$

where each $O_i(k)$, $i = 1, 2, \dots, n$ is another function to be optimized. In this thesis, multi-objective DARP formulation is used in the heuristic method and dynamic implementation. Therefore, multi-objective function defined as to minimize total routing costs and user dissatisfaction. Multi-objective function is used in the formulation as a sum of different objectives. This brings a requirement to add a weight value to each of different objectives. If weight values are not used, they are assumed to be the same. A classical approach that is used in the optimization problems for multi-objective function is to write it as a sum of weighted functions as follows.

$$\text{Min/Max } \sum_{i=1}^n w_i O_i(k) = \text{Min/Max } (w_1 O_1(k) + w_2 O_2(k) + \dots + w_n O_n(k)), w_i > 0 \quad (2.2)$$

Values of weights, i.e. $w_i, i = 1, \dots, n$ depends on the needs of the model and importance of the different objectives. If minimizing the total routing costs are more important than minimizing the user dissatisfaction, which is the case in this thesis, belonging weight value for the function to minimize the costs is chosen to be higher than the weight of

the other functions. Weight values are usually chosen to be a value between 0 and 1 and the sum of the all weight values equals to the 1.

2.2.2 Static and dynamic optimization models of DARP

Transportation problems such as VRP, VRPPD, TSP, TSPTW or DARP can be applied according to the two different modes : static or dynamic.

In the static model, incoming data is known in advance. Pick-up and drop-off coordinates, earliest pick-up time and latest delivery time is given before the algorithm is executed. This model is used in the applications do not need to work in real-time. Since all the data is given beforehand, algorithm is applied and routes are created for the vehicles. None of the routes are changed within time, no new requests arrive and none of the requests are changed during time. In the work of Bodin and Sexton (1986) [2], problem was to model a transportation system for the Project Mobility Baltimore, which aims to operate the transportation of elderly and handicapped patients. According to the rules of the project, requests should be created 24 hours before the requested pick-up time which makes all the required information for the next day to be known in advance. But depending on the nature of the problem, this is not applicable for all real cases.

In the dynamic model, algorithm starts with no or partly known requests. Orders are created online by the customers and should be operated real-time. Application can start as like a static implementation in the beginning with the existing information. But since the new requests will arrive within time, or even sometimes it is allowed to make a change in the request, routes should be consistently updated in real-time. Dynamic implementation of the optimization problems can be performed as to run static implementation with a reasonable period so that the new requests are included in the new run and new routes are created as to include the new requests. For this reason, there are more work on static methods in the literature than the dynamic methods. This method is not efficient for a real case since it does not allow to modify the formerly operated requests and takes more time to run static method every time. Another method to implement dynamic mode is to start with the given data, check the new data online and modify and constructs the routes without damaging the current situation and timing of the vehicles and current orders. Since this method is less time-consuming, it is

considered to be a better technique for the real-world case that is investigated in this thesis.

2.3 Notation

The variables that are used in objectives and constraints of DARP is adopted from [1,3]. Notation used in single-objective and multi-objective DARP formulations is given in Symbols.

n denotes the number of requests arrived from the customers. $G = (N, A)$ is the graph that the DARP is defined on. The sets P and D include pick-up and drop-off points, respectively. N is the set including the origin point, destination point and pick-up and drop-off points. 0 is the origin node and $2n + 1$ is the destination node. Carriers start their routes at the origin node and go to the destination nodes after they deliver all the requests they are responsible for. K is the set of carriers. Q_k is the load amount that is allowed for carrier k . T_k is the ride time limit for carrier k and none of the carriers are allowed to exceed this time limit. In this paper, T_k is defined as 9 hours and identical for each k . q_i is the loaded amount at node i , and each $i \in N$ is associated with a load q_i satisfying the equations $q_0 = q_{2n+1} = 0$ and $q_i = -q_{n+i}$ ($i = 1, \dots, n$). These equations mean that an order picked-up at node i should be dropped-off at node $n + i$, the load amount of each carrier at the origin and the destination points is zero. d_i is the service duration at node i . Since there is no service at the origin and destination nodes, service duration at the origin and destination nodes are zero. The equation $d_0 = d_{2n+1} = 0$ is satisfied. Time window is given as $[e_i, l_i]$ for each $i \in N$, where e_i represents the order time, l_i represents the latest time that the request is allowed to be delivered. Time window can not exceed the duration defined as L . This duration is defined as 1 hour by the company. For each arc $(i, j) \in A$, c_{ij} is the cost and t_{ij} is the time duration to traverse the arc. The average speed for the carriers is defined as 30 km/h, and t_{ij} is calculated depending on this speed. Note that in the dynamic version of the problem, these values are subject to change.

In the heuristic algorithm, V is the set of carriers determined by the optimization procedure. The less the carriers are used, the more the objective function is minimized, and the better solution we have. m , is introduced as a necessity for the heuristic

algorithm depending on the new variable V representing the number of carriers used in the solution (i.e., $|V| = m$ and is to be minimized inside the objective function).

Four decision variables are used in the MIP for the single-criterion multi-vehicle DARP model. x_{ij}^k is the variable that takes the value 1 if the carrier k is going to travel from node i to node j and it takes the value 0 otherwise. The variable B_i^k is the point in time that the carrier k visits node i . Q_i^k is defined as the amount of load that is left in the carrier k 's vehicle after it visits the node i . Total riding time of a certain request i by carrier k is identified by the variable L_i^k . Among these decision variables, x_{ij}^k is defined as a binary variable since it only takes 0 or 1. Other variables, B_i^k , Q_i^k and L_i^k are fractional variables [4].

In the heuristic version of the DARP, some relaxations to the constraints are required. While the constraints are relaxed, the objective function is expanded with a new mathematical expression. For the multi-criteria multi-vehicle DARP model, the following weight values are introduced to be used as coefficients for each expression to be minimized [5]. w_1 is the weight on the customer transportation time, w_2 is the weight on the number of used vehicles, w_3 is the weight on the route duration, w_4 is the weight on the customer's excess ride time and lastly w_5 is the weight on the waiting time for a customer.

2.4 Formulation

Given the above notation, two different mathematical formulation of DARP is given depending on different frameworks that are applied in this thesis. As mentioned before, DARP can be formulated as single-objective where there is only one function to be optimized or multi-objective where there are more than one functions to be optimized. In this thesis, single-objective multi-vehicle DARP formulation is used in MIP method to solve static DARP. In the other methods that are presented in this thesis such as heuristic implementation for static DARP or dynamic DARP are all based on multi-objective multi-vehicle DARP formulation.

2.4.1 Static dial-a-ride problem

For the static case, DARP is investigated under two different framework. First, DARP is modeled as single-objective multi-vehicle problem to be used in Mixed Integer

Programming (MIP). Second, DARP is modeled as multi-objective multi-vehicle problem to be used in heuristic implementation. Both formulations are presented in this section.

2.4.1.1 Mixed integer programming formulation

MIP formulation as single-objective multi-vehicle DARP is adopted from [3]. This formulation is accepted as standard DARP formulation in most of the works as of 2006 when we look at the literature. Mathematical formulation as well as the explanation of objective function and constraints as follows [3].

Objective Function:

$$\min \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij}^k x_{ij}^k \quad (2.3)$$

constraints:

$$\sum_{k \in K} \sum_{j \in N} x_{ij}^k = 1 \quad \forall i \in P \quad (2.4)$$

$$\sum_{j \in N} x_{ij}^k - \sum_{j \in N} x_{n+i,j}^k = 0 \quad \forall i \in P, \forall k \in K \quad (2.5)$$

$$\sum_{j \in N} x_{0j}^k = 1 \quad \forall k \in K \quad (2.6)$$

$$\sum_{j \in N} x_{ji}^k - \sum_{j \in N} x_{ij}^k = 0 \quad \forall i \in P \cup D, \forall k \in K \quad (2.7)$$

$$\sum_{i \in N} x_{i,2n+1}^k = 1 \quad \forall k \in K \quad (2.8)$$

$$B_j^k \geq (B_i^k + d_i + t_{ij})x_{ij}^k \quad \forall i, j \in N, \forall k \in K \quad (2.9)$$

$$Q_j^k \geq (Q_i^k + q_j)x_{ij}^k \quad \forall i, j \in N, \forall k \in K \quad (2.10)$$

$$L_i^k = B_{n+i}^k - (B_i^k + d_i) \quad \forall i \in P, \forall k \in K \quad (2.11)$$

$$B_{2n+1}^k - B_0^k \leq T_k \quad \forall k \in K \quad (2.12)$$

$$e_i \leq B_i^k \leq l_i \quad \forall i \in N, \quad \forall k \in K \quad (2.13)$$

$$t_{i,n+i} \leq L_i^k \leq L \quad \forall i \in P, \quad \forall k \in K \quad (2.14)$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{Q_k, Q_k + q_i\} \quad \forall i \in N, \forall k \in K \quad (2.15)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i, j \in N, \forall k \in K \quad (2.16)$$

Objective function of the DARP that is used in MIP formulation includes only one expression which minimizes total routing costs using the variables c_{ij}^k and x_{ij}^k [1,3]. A route for each vehicle is created by assigning the requests to the appropriate vehicles. Traveling from depot to depot means that the vehicle is not used at all and does not contribute to the total routing cost. Objective function calculates sum of route costs of all vehicles. Route costs are taken as total travelled distance. Constraint 2.4 guarantees that each request is served exactly once. Constraint 2.5 ensures that a request picked up by a vehicle should be dropped off by the same vehicle. Constraint 2.6 ensures that each vehicle starts its route from the origin point while constraint 2.8 ensures that it ends its route at the destination point. Origin and destination points are set as depot in this thesis. A vehicle visiting a node should be leaving the same node by constraint 2.7. Time spent between picking up and dropping off a request should be sum of service time that is spent during pick up and duration taken to travel from pick up point to the drop off point (constraint 2.9). Each request should ideally be delivered in a certain period of time by constraints 2.11 and 2.13. Constraint 2.12 ensures that maximum ride time for a route is limited. Constraint 2.13 applies the time window for a given request. Load limitations are represented by constraints 2.10 and 2.15. x_{ij}^k is defined as a binary variable with constraint 2.16 [1].

2.4.1.2 Heuristic formulation

The solution method used to solve the DARP with a heuristic algorithm is based on cluster-first, route-second principle [2,4,5]. This method is divided into two parts. In the first part, a clustering process is performed in order to distribute the requests to the carriers. A genetic algorithm (GA) based clustering method is used in the first part where each request is assigned to one and only one carrier. In the second part, routes are created for carriers depending on the requests assigned to them. The modified space-time Nearest Neighbor Heuristic (NNH) [5] is used for the routing part. The nearest neighbor heuristic uses the pick-up and drop-off locations and time windows. After the routing schedule is completed for each carrier, the costs are calculated.

The mathematical formulation of the multi-objective optimization formulation for the heuristic method is adopted from [5] as follows :

Objective Function:

$$\begin{aligned}
\min \quad & w_1 \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij}^k x_{ij}^k + w_2 m + \\
& w_3 \sum_{k \in V} (B_{2n+1}^k - B_0^k) + \\
& w_4 \sum_{k \in V} \sum_{i \in P} (B_{n+i}^k - d_i - B_i^k - t_{i,n+i}) + \\
& w_5 \sum_{k \in V} \sum_{i \in P \cup D} L_i^k (Q_i^k - q_i)
\end{aligned} \tag{2.17}$$

Instead of the belonging constraints that are used in the MIP formulation, the following constraints are used for the heuristic algorithm formulation [5].

additional constraints:

$$\sum_{j \in P \cup 2n+1} x_{0j}^k = m \quad \forall k \in V \tag{2.18}$$

$$\sum_{i \in D \cup 0} x_{i,2n+1}^k = m \quad \forall k \in V \tag{2.19}$$

$$(B_i^k + d_i + t_{ij}) x_{ij}^k \leq x_{ij}^k (B_j^k - W_j^k) \quad \forall i, j \in N, \forall k \in V \tag{2.20}$$

$$(Q_i^k + q_j - Q_j^k) x_{ij}^k = 0 \quad \forall i, j \in N, \forall k \in V \tag{2.21}$$

$$t_{i,n+i} + W_j^k - (B_i^k + d_i) \leq B_{n+i}^k - (B_i^k + d_i) \quad \forall i \in P, \forall k \in V \tag{2.22}$$

$$B_{n+i}^k + B_i^k \leq L \quad \forall i \in P, \forall k \in V \tag{2.23}$$

Since it is difficult to solve the DARP with an exact function, an LP relaxation has been performed on the following constraints: Maximum route duration constraint, maximum ride times constraint, time windows constraints [5]. An acceptable modification for these constraints is made on the model in order to solve the problem using heuristic algorithms. Within these relaxations, the softened constraints are added to the objective function to be minimized and the DARP formulation introduced in the MIP formulation is turned out to be a multi-objective DARP with the new multi-criteria objective function.

Objective function (Equation 1) used in the heuristic formulation of the DARP minimizes both the total cost and the user inconvenience [5]. The total cost is measured with the total distance travelled by all carriers in order to serve all requests, the number of carriers used in the final solution, and the total time taken by the carriers in order to serve all requests. The details about user inconvenience expressions are as introduced

in the original paper [5]. Since the constraints are very similar to the ones used in the MIP approach, the rest of the formulation will not be repeated here.

2.4.2 Dynamic dial-a-ride problem

Online scheduling algorithms are used to handle with online tasks [14] which comes during daily operation. Online, i.e. dynamic scheduling algorithm should be able to accept incoming tasks and re-optimize the routes including the new tasks while continuously checking if there is a new task. Since it has been seen from previous works [1] that is not always possible to solve DARP with exact optimization methods such as Mixed Integer Programming (MIP) for large number of requests (for more than 12 requests in the work [1]), a heuristic algorithm of which the mathematical formulation is given above is applied in this thesis to solve dynamic DARP. In the dynamic version of DARP, mechanism includes an event manager and an insertion algorithm as an addition to the optimization algorithm used to solve the problem.

3. STATIC DIAL-A-RIDE PROBLEM

This chapter includes the work belonging to the static DARP that is carried out in this thesis. DARP is investigated usually as static DARP in the literature since most of the works are based on theoretical data and since dynamic version of DARP can be obtained adjusting the static DARP. For this reason, largest part of this thesis is composed of static DARP. In this chapter, first a high-level definition of DARP is given. Second, the methods of applying static DARP are explained based on the methods that are used in this thesis to implement static DARP on Kapgel's problem. Then the implementation of the methods are given in the third section. Lastly the empirical analysis and results follow.

3.1 Definition

DARP is a type of pick-up and delivery problem to find routes for many customers using owned vehicles. The objective of DARP is to optimize the costs and user satisfaction subject to some time, capacity and precedence constraints. The problem can be considered as an extended version of Vehicle Routing Problem (VRP) and Traveling Salesman Problem (TSP). VRP is a combinatorial optimization problem to find the optimal (namely cheaper in terms of costs) routes for a set of vehicles. TSP is also a combinatorial optimization problem to find to optimal route for a traveling salesman so that all the cities in the scope are visited. DARP finds routes for the vehicles to pick-up and deliver the products within time windows and capacity constraints. Each request should be delivered in a certain time with the capacitated vehicles. Each vehicle has also a time constraint which makes the related driver make to complete the route within this total time. There are precedence constraints emerged from the nature of the pick-up and delivery problem. Each request should be picked-up first before it is delivered. Objective function is adjusted depending on the method of implementation.

3.2 Methods

In this section, methods that are used in this thesis for static DARP are explained.

In applied mathematics, computer science or in the area of operational research, optimization which also can be called as mathematical optimization or mathematical programming is a way to find the element according to some certain criteria that gives the best result among others [16].

An optimization problem is to optimize i.e. minimize or maximize the value of an objective function using the allowed elements in the solution domain. An optimization problem can be constrained optimization problem where the set of allowed values are identified by the constraints or unconstrained where there are no constraints. The aim of an optimization problem is to find the best possible value of the objective function in the given domain and set of elements that makes the objective function to take this value. There are many sub-fields of optimization each of that can be considered as another area of study. Some of these sub-fields can be listed as Linear Programming (LP), Integer Programming, Non-linear Programming, Stochastic Programming, Combinatorial Optimization, Heuristics and Metaheuristics, Constraint Satisfaction, Dynamic Programming etc. The term 'Programming' here does not state to the software programming in computer science, it refers to mathematical programming which means scheduling in the applied mathematics.

There are different techniques and algorithms used to solve different kind of optimization problems such as exact methods, iterative methods or heuristics. In this chapter, only the types and techniques of mathematical optimization that are used in this thesis are explained.

3.2.1 Exact methods

Exact methods refer to the algorithms that finds the optimal value of an optimization problem. Using exact methods such as LP, MIP, Branch and Bound etc., best i.e. optimal value for an optimization problem is aimed to be found but this is not always possible especially if the problem is based on a real-world scenario because of worst-case running time. Exact methods are mostly extremely time-consuming. LP

and MIP are explained in this section to provide an insight for the methods that are used in this thesis.

3.2.1.1 Linear programming

Linear programming is a way to formulate an optimization problem with linear relationships to find the optimal value of objective function. In a linear optimization problem, objective function and constraints in terms of valid inequalities are expressed as linear functions. Since LP is a method to simplify even very complex problems to linear functions, it is widely used in applied mathematics and operational research. There are many methods as well as software programs (solver) to solve LP problems. An LP problem can be depicted with a graphical representation.

3.2.1.2 Mixed integer programming

Integer programming problem is a problem where some or all of the variables are constrained to be integer values. MIP is often referred as Mixed Integer Linear Programming (MILP). MILP problems are constrained optimization problems where variables are integer, objective function is linear and the constraints are linear inequalities [17]. A lot of real-world problems can be formulated as mixed integer linear programming problem. The real-world problem that is investigated in this thesis is modeled as MILP using the formula in the work of [3]. As mentioned before, in MILP the purpose is to find the optimal value of objective function.

3.2.2 Linear and mixed integer programming solvers

There are many commercial and open-source optimization programs that are designed to solve LP and MIP problems. In this thesis, IBM Ilog Cplex Optimization Studio¹ and Cplex Library are used to solve the single-objective multi-vehicle static DARP. IBM Ilog Cplex Optimizer is designed to solve LP and MIP problems based on Simplex Method [18] which is one of the most powerful and used methods in LP. It offers libraries that can be used embedded with a programming language such as C, C++, Java etc. to solve LP programs. In this thesis, Cplex library is called from Java to solve the problem using Cplex.

¹<https://www.ibm.com/products/ilog-cplex-optimization-studio>

3.2.3 Heuristic methods

Heuristic and metaheuristic methods are powerful methods to solve optimization problems which cannot be solved efficiently in a reasonable time. These methods are flexible search methodologies that aims to find approximate solutions rather than the exact optimal solution. With heuristic and metaheuristic methods, high quality solutions can be produced within acceptable time without guaranteeing to find the optimal value. How close the solution to the optimal value is determined by the parameters that are specified according to the scenario. Depending on the available CPU time and the size of the instances to be solved, satisfactory solutions can be obtained using heuristic methods. A criteria, likelihood, can also be specified to get results with the desired optimality. Heuristics are widely used and provide satisfactory solutions in scheduling problems such as TSP, VRP or variations of these problems.

A wide variety of heuristic algorithms exist. In this thesis, Genetic Algorithm (GA) and Nearest Neighbor Heuristic (NNH) algorithms are used to solve multi-objective multi-vehicle static DARP using Cluster-First Route-Second approach.

3.2.3.1 Cluster-first route-second approach

Cluster-first route-second approach is a two-stage method that is used for routing. In this method, set of vertices are first clustered with an algorithm and then routes are created for different clusters using another algorithm. In this thesis, heuristic algorithm is applied using cluster-first route second approach. Clustering is performed using GA and routing is performed using NNH.

3.2.3.2 Genetic algorithm

GA is a heuristic method that is designed based on Darwin's Theory of Evolution. It is inspired by the natural evolutionary selection process. The algorithm runs within specified number of generations to find the solution which is improved in next generation. According to the algorithm, worse solutions than the current solution are not allowed to be produced in the next generation. Improvement in the solution is performed using bio-inspired operators called mutation, crossover and selection [19]. Mutation and crossover operators are applied on generated (randomly or according to some other algorithm) initial solutions to obtain the individuals in the next

generation. Only the individuals offering better solution are allowed to be included in the next generation using a selection method. Algorithm terminates according to some user-defined criteria. Algorithm can be terminated if the desired number of generations is reached, if there is no improvement in the solution during the last pre-defined number of runs or according to another user-defined criteria. To determine the optimal parameters to be used in the algorithm such as mutation and crossover rate, number of generations, number of iterations etc. algorithm is usually run many times with different values of the related parameter during the tests to have a scheme to see at which value of the parameter the best values are obtained. In this thesis, GA is used for clustering of customers for different vehicles.

3.2.3.3 Nearest neighbor heuristic

NNH is an algorithm which is mostly used to solve TSP. In TSP, a route is created for a traveling salesman so that all the cities in the plan are visited using the shortest route. In this thesis, space-time NNH algorithm [1, 5, 20] is used. NNH starts at a random city (vertex) and chooses the next city to visit according to the distance between current node and the nodes in the neighborhood. A basic version of NNH is k-NN algorithm which finds the k-nearest nodes according to the distance measurement to decide the next vertex [21]. Although NNH is a very fast algorithm, it may fail to find the best solution i.e. the shortest path for routing problems. For this reason, space-time NNH is used together with GA in this thesis as part of the cluster-first route-second approach to find the succeeding moves. The space-time NNH is differentiated from general NNH as it uses the weighted sum of travel time between the locations and the time window violation at the destination [5]. With GA, routes created by NNH in the routing part are evaluated and improved solutions are produced in the next generation until the termination criteria is reached.

3.3 Implementation

The static DARP is investigated in two different frameworks. Since it is not a theoretical but a real-world problem, looking at the scale of the problem, first an exact method is applied to the problem. In the literature, we can see that exact methods usually applied on single vehicle DARP. In this thesis, MIP is applied to

the single-objective multi-vehicle DARP. Objective in the method was to minimize the total routing costs. User satisfaction is ignored in this case and only costs are taken into account in the optimization. Second, a heuristic method is applied to the multi-objective multi-vehicle DARP. Objective function that is used in heuristic implementation includes both minimizing the total routing costs and maximizing the user satisfaction. As explained before, constraints are assumed as hard constraints in the MIP implementation. In the heuristic implementation, some of the constraints are relaxed and softened constraints are added into the objective function to be optimized with a weight value. LP relaxation is applied to the following constraint [5] : Maximum route duration, maximum ride time and time windows constraints.

3.3.1 MIP implementation

Single-objective multi-vehicle DARP is implemented using MIP formulation [3]. Objective is to minimize the total routing costs. Parameters of the problem is adjusted as to fit with the company's requirements. The sets N , P and D include longitude and latitude coordinates for the points i and j . Origin and destination points are fixed and unique. This is considered to be taken as the company's main address where the vehicles start their routes in the real-world implementation. Haversine formula [22] is used to calculate the distance between two nodes, and the values are assigned to the cost matrix c_{ij} . The Haversine formula is an equation important in navigation, giving great-circle distances between two points on a sphere from their longitudes and latitudes [22]. Since the work is on a real-world problem in this thesis, Haversine formula is chosen to be able to make better distance assumption considering the shape of the earth. But because the area that the company servers is a limited area in a certain city, results show that using Haversine formula made no real difference than Euclidean formula. Almost the same distance values are obtained. Since the formulation needs an average speed for the vehicles in order to calculate the ride time between the nodes, average speed for a vehicle is assumed to be 30 km/h for test purposes. Maximum amount of load Q_k is defined as 10 packages for a vehicle. Service duration d_i defined as 5 minutes represents the time that a carrier is used at the pick-up point for each node except origin and destination nodes. The formulation is linearized as introduced in the original paper [5].

MIP implementation is performed using LP solver IBM ILOG Cplex Optimization Studio² and cplex library on Netbeans IDE 8.1³ using the general programming language Java⁴. Using Java programming language and Cplex Concert Technology [23], objective function and constraints are modeled and solved. Pseudocode for MIP implementation using Cplex Concert Technology is given in Algorithm 1.

Algorithm 1 Algorithm for MIP Implementation of DARP

Data: Real-world data obtained from the company

Result: A scheduling of data if feasible solution is found, solution status otherwise
Initialization;

```

while not end of the execution of IloModel do
    Wait for an execution;
    if Cplex.Solve then
        objective value  $\leftarrow$  cplex.getObjValue();
        solution status  $\leftarrow$  cplex.getStatus();
        pick-up and delivery times  $\leftarrow$  cplex.getValue();
        return Obj
    else
        no feasible solution available;
        terminate;
    end
end

```

In the algorithm, following variables are declared first: number of requests, number of customers, number of nodes, pick-up and delivery times and coordinates. In this thesis, these variables are read from the data file that is provided by the Kapgel. Cplex model is defined and the constraints in the LP formulation are given into the model. Each request is served only once, the request is delivered by the same vehicle that it was picked up, each vehicle starts at origin and ends at destination, vehicle visiting a node should leave the same node, maximum duration of a service for a request is identified by the service time and the travel time, maximum amount of load on a vehicle and maximum total route duration is limited. A function to calculate Haversine distance is defined. Decision variables and objective function are declared. Model is solved using cplex library. Cplex checks the status of the solution after the program is terminated.

²<http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/>

³<https://netbeans.org/>

⁴<https://www.java.com>

Decision variables and the value of the objective function can be obtained if the status of the solution has success.

3.3.2 Heuristic implementation

The heuristic algorithm is applied using cluster-first, route-second approach. Clustering is performed using GA and routing is performed using space-time NNH algorithm. Both approaches are explained in this section in detail.

3.3.2.1 Clustering via genetic algorithm

Clustering involves grouping the set of requests into the vehicles that they will served with. To form the clusters, requests are assigned to the vehicles via GA. A request that is assigned to a cluster is not allowed to be assigned to another cluster since it should be picked up and delivered only once and these operations should be realized by the same vehicle.

Clustering the requests is carried out using binary chromosome representation where each element is allowed to take either the value 0 or 1. Binary chromosomes are described as a matrix form where each row represents a vehicle together with it's associated requests. A request is marked as 1 in the associated vehicle if it is assigned to this vehicle and 0 otherwise. In the binary chromosome representation as a matrix form, number of the rows are identified by the number of the available vehicles summed up with depot and number of columns are identified by the number of the requests. In the matrix, each request appears only once without specifying pick-up and drop-off operations. For this reason, the sum of each column in the matrix should be equal to the 1 except the depot column. This information is used in the GA to verify the feasibility of the generated matrices. Chromosomes are generated as a matrices where number of rows equals to the number of available vehicles, number of columns equals to the number of requests, sum of each row except depot column is equal to the 1 and sum of the depot row equals to the number of vehicles [1]. A feasible chromosome is generated only if all of these constraints are satisfied. Clustering is completed when feasible chromosomes are generated.

A sample representation for 8 requests and 4 carriers is given in Figure 3.1 [1]. Rows are the clusters that are generated by the GA. Each row represent a route for a vehicle.

Columns represent the requests. It can be seen in the figure that the sum of each column except the depot equals to the 1 since each request is only allowed to be assigned to one vehicle. Sum of depot column is equals to the number of the available vehicles since each vehicle should start and end their routes in the depot which means that the depot should be included once in each route, i.e. cluster. As seen in the sample chromosome representation, there are 8 requests to be served and 4 available vehicles to be used for serving the requests.

depot	Req1	Req2	Req3	Req4	Req5	Req6	Req7	Req8
1	0	0	1	0	1	0	0	1
1	1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	1	0
1	0	0	0	0	0	1	0	0

Figure 3.1 : Chromosome representation used in GA.

GA settings are done according to the need of the real-world case that is planned to be used in the company Kapgel. With the consideration that each request must be assigned to only one carrier, the initial population is generated randomly with 50 chromosomes. The population replacement proportion for the next generation is 0.10 [1].

To determine the number of generations, an empirical analysis is carried out on an instance [1]. Number of requests is chosen to be 20 and the number of vehicles is 5 to obtain a graphical representation to pick up the optimal value for the setting of number of generations that is used in the GA. The average and best cost values are analyzed over the instance with 20 requests and 5 vehicles. Multiple runs are performed in order to decide the optimal value of the number of generations parameter. The program is run 100 times each step while the number of generations are increased with a certain amount and results are obtained. The plot representing the results is given in Figure 3.2. It can be seen in the figure that as of the number of generations to be 10000, the average and best fitness values are almost stable. Since to increase the number of generations to a larger value would affect the performance of the program, the value

for the parameter is determined to be 15000 according to the results. This value is also used as the stopping criteria as the maximum iteration number. The fitness of an individual is solely its objective function value. A simple crossover operator with probability 0.60 is used [5]. The simple crossover is to exchange the genes between two individuals and to create two new offsprings. The offsprings get a section before the crossover point from one parent's chromosome and the section after the crossover point from the second parent. This procedure is repeated until each of the requests is allocated to exactly one vehicle [5]. A standard mutation operator with probability 0.05 is used which moves one random customer from its current cluster to another random cluster (i.e., carrier).

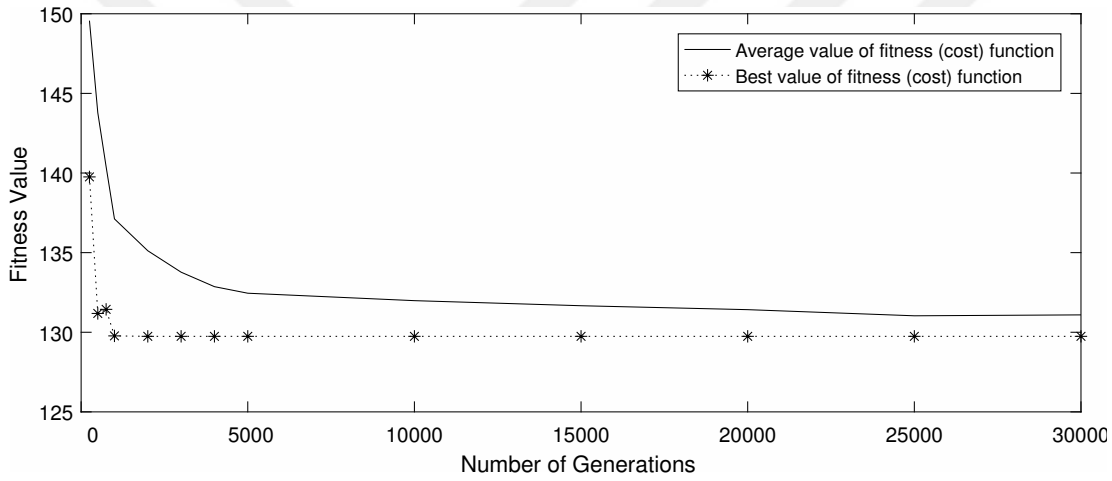


Figure 3.2 : The average and best cost values change with the number of generations.

3.3.2.2 Routing via space-time nearest neighbor heuristic algorithm

The modified space-time nearest neighbour heuristic [15] is used in the routing part to create routes for the carriers [1]. The algorithm generates the routes so that the request having the earliest pick-up time is taken first into the queue. To add the next node into the cluster, costs for the feasible nodes are calculated and the optimal, i.e.cheapest node is chosen. To calculate the cost for a node, weighted sum of travel costs between nodes and the time window violation at the destination node are calculated [1]. Therefore, next node to be visited is the lowest cost unvisited node among others. Routes are created for each vehicle in this way. Routes for the sample that is given in Figure 3.1 can be seen in Figure 3.3. The numbers 1 and 2 that are associated to the request numbers includes the information regarding pick-up and drop-off. If 1 is used

with a request, the action is to pick-up the request from pick-up location. Similarly, if 2 is used with a request, the action is to drop-off the request to the drop-off location. In the sample given in Figure 3.3 [1], associated vehicle picks-up the orders 3, 5 and 8 and drops them of with the order 3, 8 and 5, respectively. The carrier starts and ends its route at the depot.

depot	3.1	5.1	8.1	3.2	8.2	5.2	depot
-------	-----	-----	-----	-----	-----	-----	-------

Figure 3.3 : The constructed route for carrier 1 from Figure 3.1.





4. DYNAMIC DIAL-A-RIDE PROBLEM

This chapter includes the work belonging to the dynamic DARP that is carried out in this thesis. Since the work in this thesis is based on a real-world problem, dynamic implementation is also analysed. In this chapter, dynamic mode of DARP is introduced and differences from static mode are presented first. Second, the methods that are used to implement dynamic DARP in this thesis are explained. Then the implementation of the methods are given in the third section.

4.1 Definition

Since a real-world problem is investigated in this thesis, dynamic DARP is also considered as part of the thesis. As mentioned before, all the data including pick-up and drop-off coordinates, earliest pick-up and latest drop-off times is known in advance in the static DARP. Static DARP can be used for the cases where the orders are created with a guaranteed time before the actual pick-up. If there is no such constraint and the orders should be processed once they arrive, dynamic DARP should be used. In the dynamic DARP, orders arrive randomly during the day and should be processed. In the beginning of the execution, the list of order starts as an empty list. Within creation of new requests, routes are generated simultaneously. This makes the problem to accept new requests into the system during the execution time. Changes in the existing orders such as cancellation of an order or removal of an order because it is delivered is also possible. While the new orders are accepted and added to the available vehicles and delivered orders are removed from the system, all routes should still remain feasible and optimized as much as possible.

In the literature, dynamic DARP does not include the complexity to find the shortest paths between all the possible locations in order to accept the new request [24]. To eliminate this calculation from the execution, distances between all the possible locations are calculated in advance and kept in a matrix to be used in case a new request comes in [24]. In the case of Kapgel, this was not an option, since the locations

of possible new orders are not predictable. Instead, the distance calculations carried out during the execution time after a new request arrived with its pick-up and drop-off coordinates.

The objective and constraints of dynamic DARP remains the same as in the heuristic implementation of static DARP.

4.2 Methods

Optimization methods that are used in the dynamic DARP are included in the methods that are used in static DARP. Heuristic DARP formulation and heuristic methods are used to solve dynamic DARP. Main heuristic algorithm that is used in dynamic DARP implementation is NNH. Version of k-NN algorithm with k value to be 4 is used in the dynamic DARP. As mentioned before, k-NN algorithm is a method that is used to solve TSP-like problems as finding the k-nearest nodes according to the related distance measurement and makes the best choice among the results. Since NNH is applied in static DARP implementation as a heuristic method for the routing part of cluster-first, route-second approach, it is also used to generate the routes dynamically in the dynamic DARP implementation.

4.3 Implementation

The dynamic DARP is investigated in a single framework. Mathematical formulation for multi-objective multi-vehicle DARP that is given under the heuristic implementation is used to implement dynamic DARP. Weight value on the number of vehicles is eliminated in the dynamic DARP implementation since the new vehicle is only added if none of the used vehicles can accept the newly incoming request and this makes the weight function on number of vehicles an expression that cannot be optimized more as part of objective function.

4.3.1 Overview of the solution method

In the static mode of DARP, using the cluster-first route-second approach, known requests are first clustered and then the routes are generated for each clusters. Genetic algorithm is used to recombine the different clusters to test fitness value and once the

optimal cluster that provides the optimal fitness value is found, routes are created for each vehicle using NNH and no change is allowed in the clusters and routes after this point.

In the dynamic mode of DARP, requests arrive one by one at a time and placed in a cluster providing the maximum benefit. Unlike the static DARP, since the requests arrive within execution time, time after the program is started to run becomes an important criteria in dynamic DARP. In the dynamic DARP, time is split into 1 minute intervals in order to manage the incoming requests [12]. Once a request is accepted at a certain time t , acceptance and optimization process is started for the request in question as of t [9]. This execution time to adapt the new request into the system and optimize the existing requests again according to the new situation is costly in terms of time when GA is used to re-optimize the already assigned requests. For this reason, incoming requests are subjected to a decision algorithm in order to be accepted and then the best vehicles according to the objective function value are chosen for the new requests. In this manner, the goal of the dynamic DARP that is applied in this thesis is to find the best vehicle to pick the assigned request up.

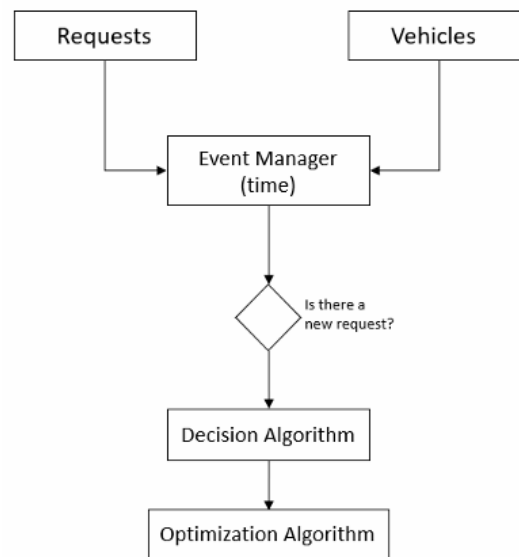


Figure 4.1 : Solution architecture of dynamic DARP implementation.

The architecture of the solution mechanism that is build up to implement the dynamic DARP is given in Figure 4.1. According to the architecture, program initially starts with an empty list An event manager works to split the program into time slices. In

this thesis, time slice is taken as 1 minute depending on the intensity of incoming tasks based on the data that is obtained from Kapgel. If there is a new task made in the time slice that is checked, it assigns the new task either to the one of available vehicles or take a new vehicle into the system to assign this task depending on which option minimizes the total cost and eventually update the routes. Decisions are made based on distance and cost of insertion. Cost is calculated as in the heuristic method of static implementation. According to the formulation, objective function includes the total distances that are travelled and user dissatisfaction in terms of violated delivery times. To adapt the objective function that is from the heuristic implementation, a modification that is to eliminate the weight value regarding the number of vehicles is done in the formulation. According to this modification, objective function is reduced to be as below. Constraints remained as it was in the heuristic implementation of static DARP.

Objective Function:

$$\begin{aligned}
\min \quad & w_1 \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij}^k x_{ij}^k + \\
& w_3 \sum_{k \in V} (B_{2n+1}^k - B_0^k) + \\
& w_4 \sum_{k \in V} \sum_{i \in P} (B_{n+i}^k - d_i - B_i^k - t_{i,n+i}) + \\
& w_5 \sum_{k \in V} \sum_{i \in P \cup D} L_i^k (Q_i^k - q_i)
\end{aligned} \tag{4.1}$$

In real case, geographical coordinates of any vehicle should be taken by Global Positioning System (GPS) because there might be cases which is not ideal and did not go as expected [24]. Since it is not possible to do this in the scope of this project, precise location of the vehicles depending on time is estimated. It is assumed that the real dispatching is performed as how the tasks are scheduled and current positions of vehicles and statuses of requests are found based on vehicle's last known location, speed and next route destination. Status of a task is also updated in event manager and completed tasks are removed from the routes.

In the decision (insertion) algorithm, insertion cost of an incoming task is calculated as the total distance between the new task locations and the closest stops which are already in the route. Vehicles with no tasks are treated as new vehicles. This calculation also includes the robots who are waiting in the depot, i.e. unused but available. If it

is cheaper to add a new vehicle into the system, then a new vehicle is created and the new task is assigned to it.

Routes are created using the same modified space-time Nearest Neighbor Heuristic (NNH) [5] that is used in static DARP implementation [1] as optimization part of the architecture shown in Figure 4.1. Nearest 4 location is calculated to make a decision among the location points. Objective function is used as an evaluation criteria to calculate the best 4 points. Cheapest next move given the four nearest neighbors is returned. After each route is created, total cost including the violations is calculated.

4.3.2 Components of the solution mechanism

Components of the solution mechanism to solve dynamic DARP is given graphically in Figure 4.1. In this section, each component is explained in detail.

4.3.2.1 Requests

In the dynamic mode, requests arrive during the execution time of scheduling program. Because we have the all data set for test purposes, we assume that the program is starts scheduling at the time that the first request is created. Unlike static DARP, requests should be kept with the time and location information from pick-up until drop-off. Therefore, everytime that a route is created, a record of the status of each request together with where the vehicles are, which orders are processed vs. scheduled should be kept.

4.3.2.2 Vehicles

Similar to the static case, there are certain maximum number of capacitated vehicles that are responsible to pick-up and drop-off requests. As it is done for the requests, vehicles are kept with the information of their location in time. In the real world, this information should be taken over a GPS provider since the conditions like traffic, service time, speed of vehicles can be variable. But in this thesis since we have all the data from Kapgel for test purposes as explained before, the location of the vehicles are calculated in the program according to the ideal situation. Current location of vehicles based on their last known location, speed and next route destination are calculated. Therefore, we know where a vehicle is at a certain point in time during delivery

process. This is calculated based on the speed of vehicle, orders that the vehicle has and the scheduling that it should follow.

4.3.2.3 Event manager

Event Manager is a function which works on a time slice structure. It handles with newly incoming requests and manages the insertion process. Time is split into 1-minute intervals in this thesis according to the intensity of the incoming requests. In every 1 minute, it is checked by the Event Manager if there is a new request/requests. When there are no new requests, the event manager calls directly the optimization algorithm. As mentioned, space-time NNH is used to optimize the routes. When the Event Manager detects a new request, it calls the injection method (decision algorithm) to decide what to do with the new request. After the new request is inserted into the system and the routes are generated again using the optimization algorithm, Event Manager keeps calculating all the locations for vehicles and requests depending on the time until a new request is detected.

4.3.2.4 Decision algorithm

This is a subroutine in the program to assign the request to a vehicle or to initiate a new vehicle to assign the request. Insertion cost is calculated as the total distance between the location of new request and the closest stops which are already in the route. If there are vehicles available and it's cheaper to use a new vehicle according to the objective function value, a new vehicle is initiated and the new request is assigned to the new vehicle, otherwise new request is added to the vehicle with the cheapest insertion cost. After finding a suitable vehicle for the new request, then the optimization algorithm starts optimizing the result as a static problem.

4.3.2.5 Optimization algorithm

Optimization algorithm handles with finding the routes and doing the scheduling for the requests and vehicles as it was in static DARP. Space-time NNH is used as an optimization method to generate the routes as in the routing part of static case. The algorithm returns the 4 nearest neighbors and the cheapest move is chosen as a next step to generate the routes.

4.3.3 Implementation of method

Described solution method is implemented as a solution to dynamic DARP on Netbeans IDE 8.2¹ using the programming language Java². Algorithm is executed on a 2.60 GHz Intel Core i7-8850H CPU and 16 GB memory (RAM).

Algorithm starts running as reading the data. Start time of the algorithm is assumed to be the generating time of the first request. Default values such as location of depot, maximum allowed delivery duration, maximum route length, weight values, default speed of vehicles, number of available vehicles, capacity etc. are declared. After having all the required information, algorithm starts to run the event handler. Event Manager splits the time into 1 minute intervals and check if there is a new request in a time interval. If there is no new request, Optimization Algorithm runs to generate optimal routes for the existing requests. If there is a new request, then the Decision Algorithm calculates the insertion cost for the new request and adds this request into a vehicle or initiates a new vehicle to assign this request depending on which option is cheaper in terms of cost. Once the request is added into the vehicle, Optimization Algorithm runs to generate a optimized route for the associated vehicle. Pseudocode for the implementation of dynamic DARP is given in Algorithm 2.

¹<https://netbeans.org/>

²<https://www.java.com>

Algorithm 2 Algorithm for dynamic DARP implementation

Data: Real-world data obtained from the company

Result: Dynamic assignment and scheduling of data

Initialization;

```
while not end of the execution do
    Wait for an execution;
    while not the end of the time do
        split time into 1-minute slices;
        check if new request comes in;
        if new request then
            execute DecisionAlgorithm();
            if availableVehicle > 0 then
                if Insertion Cost is lower then
                    create new vehicle;
                    assign request to a new vehicle;
                    execute Optimization Algorithm();
                else
                    assign request to the cheapest vehicle;
                    execute Optimization Algorithm();
                end
            else
                assign request to the cheapest vehicle;
                execute Optimization Algorithm();
            end
            return scheduling
        else
            repeat;
        end
    end
end
end
```

5. EMPIRICAL ANALYSIS and RESULTS

In this chapter, empirical analysis on the data that is provided by the company is performed using the described methods for static and dynamic DARP and the results and findings are presented. First, the data is described with its characteristics and a part of data is presented as a sample. Second, the empirical analysis is given for static and dynamic DARP and settings for each method that is described before is explained in detail. Third, the numerical results with comparison of methods for static and dynamic DARP follows.

5.1 Data

The company that the problem belongs to is generated a simulated data for test purposes which is very similar to the real-world data [1]. Time stamps that the requests are generated, pick-up and drop-off coordinates are included in the data. Since all the entries in the data file belongs to one day, time stamps include only the time and not the date in the format of hh:mm:ss. The company takes the requests online in the ideal case and it is not possible to order in advance with period such as one day. Therefore, there is no need to provide the date information to the algorithm. Time that is given in the data is the exact time of arrival of request. This information also has the latest delivery time information for the request in question since it is fixed as 1 hour as of ordering. The data file also includes latitude and longitude values for pick-up and drop-off coordinates, respectively. Area that is worked on is a limited area in Istanbul, Turkey. So, the coordinates in the data belongs to this area.

The whole data that we used has 167 successive entries that are ordered by the arrival time. To test the methods for static DARP, data is separated into instances having different number of requests. For the dynamic DARP, on the other hand, whole data set is used as a list of dynamically generated requests and the scheduling is performed based on timestamps of the orders.

The part of the data set as a sample for 14 requests ($n = 14$) is given in Table 5.1. Actual data includes more accurate numbers for latitude and longitude values. Since it does not affect the distance between the locations in a noticeable amount, sensitivity of numbers is limited to be 8 values. Abbreviations *D.Lat.*, *D.Lon.*, *P.Lat.* and *P.Lon.* given in the table represents *Drop-off Latitude*, *Drop-off Longitude*, *Pick-up Latitude* and *Pick-up Longitude*, respectively.

Table 5.1 : Sample data set.

Order Time	D. Lat.	D. Long.	P. Lat.	P. Long.
07:01:00	41.07229665	29.00858021	41.0679290	28.998334
07:02:00	41.07973364	29.00261594	41.0683038	28.995370
07:02:00	41.11138302	29.02498088	41.1103170	29.025015
07:04:00	41.04422181	28.99857499	41.0492424	28.994820
07:07:00	41.03666225	28.98240609	41.0509730	28.992405
07:10:00	41.04648808	29.00000021	41.0509730	28.992405
07:21:00	41.09853598	29.00666582	41.1089660	29.019145
07:21:00	41.04648081	29.00456804	41.0506640	28.994785
07:25:00	41.05389792	29.00851407	41.0478833	28.994217
07:30:00	41.04309879	28.97850662	41.0464000	28.986510
07:31:00	41.03159294	28.99582617	41.0463760	28.986622
07:35:00	41.07401257	28.99822703	41.0611558	28.988009
07:36:00	41.05204563	28.98962944	41.0508389	28.992600
07:41:00	41.12461287	29.05163578	41.1103020	29.032298

5.2 Empirical Analysis

In this section, empirical results of the test that are performed on the data described above are presented. The results are examined in two parts as static and dynamic and tests are performed accordingly for each method described in the former sections.

5.2.1 Static dial-a-ride problem

Static DARP is investigated with an exact method and with a heuristic method [1].

5.2.1.1 Experiment setup

To test the algorithms, two sets of instance are generated from the data. One of them includes 2 vehicles while the other set includes 3 vehicles. The algorithms developed for MIP and GA methods are applied on these instances for test purposes. During the experiment, number of requests is started to be 4 and increased by 2 at each trial until MIP is not able to find a solution in a reasonable time or not at all. The results of the tests for MIP and GA methods are compared in terms of cost value and runtime in seconds. Algorithms for the methods for static DARP is tested on NetBeans IDE 8.1¹ using the programming language Java². Universal LP solver IBM ILOG Cplex Optimization Studio12.6³ and the Cplex libraries are used to implement the MIP method [1]. Programs for both algorithms is executed on a 2.49 GHz Intel Xeon E312xx processor and 4.0 GB memory (RAM) [1]. No limit is defined for runtime of the programs so that it is possible to see the characteristic of the runtime with the increasing number of requests [1].

5.2.2 Dynamic dial-a-ride problem

Dynamic DARP is investigated using a heuristic method.

5.2.2.1 Experiment setup

To test the dynamic algorithm, whole data set including the 167 different requests with own coordinates and creating time is used. It is assumed that the creating time of the first request in the data is the current time of the program. The algorithm that is developed for dynamic DARP is applied once in the whole data and the results are obtained dynamically. Since the time is split into 1 minute intervals in the algorithm, algorithm checks in every minute if there is a new request and proceed according to the result.

¹<https://netbeans.org/>

²<https://www.java.com>

³<http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/>

Algorithm is applied on Netbeans IDE 8.2⁴ using the programming language Java⁵. Program is executed on a 2.60 GHz Intel Core i7-8850H CPU and 16 GB memory (RAM).

5.3 Results

Results are analyzed in terms of static and dynamic DARP.

5.3.1 Static dial-a-ride problem

Results of the tests for static DARP algorithms is summarized in Table 5.2. Table includes the results obtained both from MIP and heuristic implementations to provide a view which is comparable. n , k , Q and L identify the number of requests, the number of vehicles, the capacity of vehicles and the allowed duration to deliver a product, respectively [1]. *MIP Cost*, *MIP RT[s]*, *HA Cost*, *HA RT[s]* mean *Mixed Integer Programming Cost*, *Mixed Integer Programming CPU Runtime in seconds*, *Heuristic Algorithm Cost* and *Heuristic Algorithm CPU Runtime in seconds*, respectively. Here *cost* implies the objective function value obtained from the associated instance [1].

As it can be seen from the Table 5.2, results are obtained and compared in terms of costs and runtime [1]. One can expect to have the an increasing value of cost function within the increased number of requests. It can be clearly seen from the table that the cost (objective value) both for MIP and GA results is increased while the number of requests is increased. In the overall, heuristic algorithm is decided to have better performance. Although the runtime of first 3 trial for each instance set is acceptable for MIP implementation, it is seen that the runtime increases dramatically after a point which is not acceptable in the real time. The cost value, however is kept stable compared to the heuristic implementation. One can see that heuristic algorithm shows a better behaviour than MIP algorithm in terms of cost value for some instances. This is caused by the relaxations that are performed in the heuristic algorithm. Since all the constraints are hard and any violation is not allowed in MIP algorithm, it is ordinary that the cost for MIP implementation is higher than the heuristic implementation for a few number of instances. The heuristic algorithm has also soft constraints such as time

⁴<https://netbeans.org/>

⁵<https://www.java.com>

window constraint. To violate the time window is allowed in heuristic algorithm and this is added as an extra cost to the objective value with a certain weight multiplier.

It can be seen from the table that after the point that the number of requests is 14, MIP algorithm is not able to find a feasible solution. At this point, program is either terminated with an out of memory error or it lasts very long like more than a day which is not acceptable and so terminated manually. Runtime characteristics are also investigated, and it has been seen that the CPU runtime for MIP rise exponentially [1].

The results show that the implementation cannot be done for the company using the MIP algorithm because of excessive runtimes since there is a deadline in time to deliver the products. On the other hand, heuristic algorithm, i.e. GA seems to be applicable since the cost values are quite similar to the ones obtained by MIP implementation as well as the runtime is always kept in an acceptable value in seconds, such as under 30 seconds for 14 requests [1].

Table 5.2 : The results for the generated instances.

Instance	n	k	Q	L	MIP Cost	MIP RT[s]	HA Cost	HA RT[s]
a4-2	4	2	10	60	33.18	0.27	23.52	14.04
a6-2	6	2	10	60	33.18	1.13	33.12	15.24
a8-2	8	2	10	60	35.74	14.28	46.90	15.74
a10-2	10	2	10	60	44.17	223.92	51.26	20.19
a4-3	4	3	10	60	33.18	0.16	23.52	19.35
a6-3	6	3	10	60	33.18	0.97	33.12	18.97
a8-3	8	3	10	60	35.70	3.47	46.90	22.61
a10-3	10	3	10	60	44.17	206	51.26	24.21
a12-3	12	3	10	60	53.71	13940.55	70.24	25.07
a14-3	14	3	10	60	60.95	25456.7	88.82	28.35
a16-3	16	3	10	60	-	-	105.77	63.09
a18-3	18	3	10	60	-	-	117.99	54.23
a20-3	20	3	10	60	-	-	130.83	54.55
a16-4	16	3	10	60	-	-	105.77	73.61
a18-4	18	3	10	60	-	-	118.36	63.54
a20-4	20	3	10	60	-	-	129.74	62.68
a16-5	16	5	10	60	-	-	107.26	77.20
a18-5	18	5	10	60	-	-	119.73	75.16
a20-5	20	5	10	60	-	-	131.48	85.84

5.3.2 Dynamic dial-a-ride problem

In the dynamic DARP implementation, one of the most part to handle is the timing. This includes several aspects : While the requests arrive dynamically in time, they should be processed as meeting the constraints of the problem. Current time according to the program and the location information of the vehicles should be kept in memory during the execution. Because the vehicles are also dynamically moving in time, i.e. they pick up and deliver the requests, once they are not available to pick up a new request, it is possible that they become available again after delivering some of the requests assigned to them. Therefore, it is important to see how the algorithm generates the routes dynamically considering the time and location information in the dynamic DARP implementation.

In this context, the goal of the this part of the thesis is to model a realistic situation of dynamic DARP [12] and to provide an overview how the output looks like in case of a real-world problem. For this reason, a part of the data is used to test and present the results of dynamic DARP and a case is studied over this data with the details of timing and generating the routes for the vehicles. Because the data is considered to be a real-world data, there can be or cannot be new requests at each instant of time. The algorithm behaves as it was explained in Figure 4.1 according to the situation of the checked time interval.

Since we only have a data belonging to a one certain day, we were not able to test the dynamic DARP algorithm on different instances. It is tested on a one-day data which includes 167 different requests arriving randomly during the day, i.e. during the execution time of the program. Current time is identified by the arrival time of the first request and it is updated each time a status is updated within the algorithm. Therefore, the current time information is given for each vehicle in the beginning of their route, end of their route and in between these time at every time that there is an update in the statuses of the requests or vehicles. Output of the algorithm also includes the complete route scheduling and duration for each vehicle.

5.3.2.1 Case study on dynamic DARP

In this section, a sample run is analyzed with the output of the dynamic DARP algorithm. The part of the data that is used to test the dynamic DARP algorithm is

given in Table 5.3. Data includes 10 ($n = 10$) requests that arrive one after another. Abbreviations *Req. ID*, *D.Lat.*, *D.Lon.*, *P.Lat.* and *P.Lon.* given in the table represents *Request ID*, *Drop-off Latitude*, *Drop-off Longitude*, *Pick-up Latitude* and *Pick-up Longitude*, respectively. According to the data, it is assumed that the current time when the algorithm is started to run is the arrival time of the first request, i.e. 05:02. Therefore, current time at the end of the algorithm will be the time when all the requests in the route of the belonging vehicle are delivered.

Table 5.3 : Sample data set for the test of dynamic DARP algorithm.

Req. ID	Order Time	D. Lat.	D. Long.	P. Lat.	P. Long.
R1	05:02:00	41.05064890	28.99910760	41.05100920	29.00131440
R2	05:06:00	41.05988600	28.98332500	41.06356840	28.99153530
R3	05:10:00	41.07532250	29.02175880	41.07638800	29.02602990
R4	05:17:00	41.04213900	28.96627500	41.04227770	28.96639980
R5	05:43:00	41.10240220	29.05141890	41.10843920	29.01780550
R6	05:49:00	41.04611100	28.99341200	41.04922390	28.99852620
R7	05:59:00	41.03990910	28.99236400	41.05311460	28.99435730
R8	06:01:00	40.996R9700	29.10021330	40.99538770	29.10022730
R9	06:11:00	40.98738400	29.02185800	40.98728310	29.02168750
R10	06:13:00	40.97296220	29.10564940	40.98838030	29.10413730

The output of the dynamic DARP algorithm for the data given in the Table 5.3 is given in Listing A.1 in Appendix A.1. As it can be seen, routes are generated dynamically within the arrival of the new requests. Whenever it is cheaper to do so and as soon as there is an available vehicle, a vehicle is generated to assign the request. Therefore, the first step of generating the solution is to define the vehicle and assign the first request to this vehicle to be transferred. In every update, e.g. when there is a new request, the status of the requests that are not delivered yet are also printed out. Once a vehicle is delivered one or more of the requests that are assigned to it, then it can be used again and again according to its current location if it is cheaper. At the end of the output, complete routes for each vehicle for the whole day are given.

It can be seen from the output of the dynamic DARP implementation that the algorithm runs correctly to handle with the dynamic architecture. The results, on the other hand,

are unfortunately not as qualitative as they were in the static implementation. As testing the full data it has been seen that the route time violations exist which makes the routes last longer than the maximum allowed time. Since the GA is not used as in the heuristic implementation of the static DARP and NNH is used for routing, the focus of dynamic DARP implementation was to implement the dynamic architecture. The architecture includes to split the time into intervals, accept the online requests, take the snapshot of the static situations to obtain the locations of vehicle and time information, run insertion and optimization algorithms to generate the routes. Results show that the dynamic DARP is modeled successfully to satisfy the constraints of the problem and the real-case scenario.



6. CONCLUSION and DISCUSSION

DARP is an NP-Hard problem to generate the best routes for the vehicles which make pick-up and deliveries of the orders. Orders are taken online from the customers and in the ideal case, the routes should be scheduled online so that the time and capacity constraints are satisfied and the costs are minimized. This thesis is based on a problem of a real-world company. The problem of the company is modeled as a DARP considering the constraints and the objectives that the company is provided.

The mathematical formulation of the DARP that is used in this thesis is based on the work in [3] which is accepted as the standard definition of DARP and the most used formulation in the literature for the researches on DARP. This formulation is adjusted and adapted to the constraints and objectives of the company. DARP is investigated under two frameworks in this thesis : static DARP and dynamic DARP. In the research of static DARP implementation, the focus was to optimize the costs while satisfying the constraints and generating the routes. For this purpose, an exact method and a heuristic method is implemented and the results are compared in terms of objective values, i.e. costs and runtime. It has been seen that the exact method for static DARP implementation is not able to solve the problem for the large instances although it provides better objective values than the heuristic implementation generally. The heuristic implementation for static DARP has provided the expected results and was able to solve the large instances as well without making a big gap in the objective value than the exact method. In the dynamic DARP implementation, on the other hand, the focus was to implement the architecture to solve the DARP dynamically. A model is developed to solve the dynamic DARP and it is implemented using the heuristic methods. The mechanism in the model to solve dynamic DARP includes an event manager to handle with the timing issues in the dynamic mode, an insertion algorithm and an optimization algorithm.

To obtain the best results and the best performance, cluster-first, route-second algorithm is used to solve the static DARP. Clustering is performed using GA and

routing is performed using NNH. Because the algorithm runs many times to improve the objective value at each generation via GA, results were satisfactory in terms of costs. It can be seen from the Table 5.2 that the exact and heuristic methods of static DARP implementation are compared and the performance of heuristic implementation is quite good comparing to the exact method which finds the best available solution among the feasible solutions. In the dynamic implementation, NNH is used as optimization algorithm, since only the routing is performed after the requests are accepted by the insertion algorithm of the system. Since we only had the data belonging to the one day, we were not able to test the dynamic DARP on different instances. But it has been seen that the algorithm runs correctly and the routes are generated accordingly for the vehicles online. At each step, a snapshot of the current situation with location and time information is taken and saved in dynamic algorithm. The bottleneck in the dynamic algorithm was that during the calculations of time and location information, it is assumed that the program is executed in an ideal environment. This means that the vehicles start the routes as soon as the requests are assigned to them, they transfer the requests in the planned time and they are at a point as it was calculated before. Unfortunately, this is not acceptable in the real-world. In the real-world, the location and delivery information should be taken with a GPS from the vehicles and the drivers, respectively since the speed of the vehicles, service time in the pick-up point, conditions in the traffic can all be variable from time to time and there can also be unexpected situations such as accident or long waiting times during pick-up or delivery. However, as mentioned before, within the usage of a GPS, this problem can easily be eliminated.

6.1 Future Work

Since the larger part of this research is taken by the works done on modeling the real-world problem using DARP and solving the problem as a static DARP so that the costs are optimized, only one model could be applied to solve the dynamic DARP. This makes us to leave the comparison of the heuristic methods for dynamic DARP to be performed in the future. It is possible to use the same architecture that is described for dynamic DARP and apply different heuristic methods both for insertion algorithm and optimization algorithm. Moreover, the heuristic method that is applied to solve

the static DARP can be adapted into the dynamic DARP. This would make it possible to improve the routes for already assigned requests. Therefore, to apply the GA on dynamic DARP is left as a future work.

As it was mentioned before, the environment and the components of the scenario are all assumed to be ideal in the dynamic DARP implementation. To use GPS in order to obtain the correct information of the location of vehicles and the situation of requests should also be considered to be applied in the future. Therefore, to improve the algorithm so that it fits better in the real-world is needed so that it can be used effectively by the real-world instant delivery companies.





REFERENCES

- [1] **Arıkan, D., Öztoprak, Ç. and Sariel, S.** (2017). Empirical Analysis of Optimization Methods for the Real-World Dial-a-Ride Problem, 589–600.
- [2] **Bodin, L.D. and Sexton, T.** (1986). The multi-vehicle subscriber dial-a-ride problem, *TIMS studies in Management Science*, 2, 73–86.
- [3] **Cordeau, J.F.** (2006). A branch-and-cut algorithm for the dial-a-ride problem, *Operations Research*, 54(3), 573–586.
- [4] **Cordeau, J.F. and Laporte, G.** (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem, *Transportation Research Part B: Methodological*, 37(6), 579–594.
- [5] **Bergvinsdottir, K., Larsen, J. and Jørgensen, R.** (2004). *Solving the Dial-a-Ride Problem using Genetic algorithms*, Informatics and Mathematical Modelling, Technical University of Denmark, DTU.
- [6] **Røpke, S., Cordeau, J.F. and Laporte, G.** (2007). Models and a Branch-and-Cut Algorithm for Pickup and Delivery Problems with Time Windows, *Networks*, 49(4), 258 – 272.
- [7] **Borndörfer, R., Grötschel, M., Klostermeier, F. and Küttner, C.** (1999). Telebus Berlin: Vehicle Scheduling in a Dial-a-Ride System, *N.H.M. Wilson, editor, Computer-Aided Transit Scheduling*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.391–422.
- [8] **Cordeau, J.F. and Laporte, G.** (2007). The dial-a-ride problem: models and algorithms, *Annals of Operations Research*, 153(1), 29–46, <https://doi.org/10.1007/s10479-007-0170-8>.
- [9] **Psaraftis, H.** (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem, *Transportation Science*, 14(2), 130–154.
- [10] **Jaw, J.J., Odoni, A.R., Psaraftis, H.N. and Wilson, N.H.** (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows, *Transportation Research Part B: Methodological*, 20(3), 243 – 257, <http://www.sciencedirect.com/science/article/pii/0191261586900202>.
- [11] **Colorni, A. and Righini, G.** (2001). Modeling and Optimizing Dynamic Dial-a-Ride Problems, *International Transactions in Operational Research*, 8(2), 155–166, <https://onlinelibrary.wiley.com/doi/abs/10.1111/1475-3995.00256>,

<https://onlinelibrary.wiley.com/doi/pdf/10.1111/1475-3995.00256>.

- [12] **Santos, D.O. and Xavier, E.C.** (2015). Taxi and Ride Sharing: A Dynamic Dial-a-Ride Problem with Money as an Incentive, *Expert Systems with Applications*, 42(19), 6728 – 6737, <http://www.sciencedirect.com/science/article/pii/S0957417415003024>.
- [13] **O. Santos, D. and Xavier, E.** (2013). Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem, pp.2885–2891.
- [14] **Lois, A. and Ziliaskopoulos, A.** (2017). Online algorithm for dynamic dial a ride problem and its metrics, *Transportation Research Procedia*, 24, 377 – 384, <http://www.sciencedirect.com/science/article/pii/S2352146517303782>, 3rd Conference on Sustainable Urban Mobility, 3rd CSUM 2016, 26 – 27 May 2016, Volos, Greece.
- [15] **Baugh Jr, J.W., Kakivaya, G.K.R. and Stone, J.R.** (1998). Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing, *Engineering Optimization*, 30(2), 91–123.
- [16] **Dantzig, G.B.** *The Nature of Mathematical Programming*.
- [17] **Genova, K. and Guliashki, V.** (2011). Linear Integer Programming Methods and Approaches – A Survey.
- [18] **Horen, J.** (1985). Linear programming, by Katta G. Murty, John Wiley & Sons, New York, 1983, 482 pp, *Networks*, 15(2), 273–274.
- [19] **Mitchell, M.** (1998). *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, USA.
- [20] **JR., J.W.B., KAKIVAYA, G.K.R. and STONE, J.R.** (1998). INTRACTABILITY OF THE DIAL-A-RIDE PROBLEM AND A MULTIOBJECTIVE SOLUTION USING SIMULATED ANNEALING, *Engineering Optimization*, 30(2), 91–123.
- [21] **Jin, L., Koudas, N. and Li, C.** (2004). NNH: Improving Performance of Nearest-Neighbor Searches Using Histograms, **E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm and E. Ferrari**, editors, *Advances in Database Technology - EDBT 2004*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.385–402.
- [22] **Van Brummelen, G.** (2013). *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*, Princeton University Press.
- [23] **Corporation, I.** (2016). *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual Version 12 Release 7*, Copyright IBM Corporation, Armonk, NY, USA.
- [24] **Varone, S. and Janilionis, V.** (2014). INSERTION HEURISTIC FOR A DYNAMIC DIAL-A-RIDE PROBLEM USING GEOGRAPHICAL

MAPS, *MOSIM 2014, 10ème Conférence Francophone de Modélisation, Optimisation et Simulation*, Nancy, France, <https://hal.archives-ouvertes.fr/hal-01166662>, colloque avec actes et comité de lecture. internationale.





APPENDICES

APPENDIX A.1 : Output of the Dynamic DARP Algorithm





APPENDIX A.1

Listing A.1: R output

```
(05:03:00)    Vehicle #V1 created.
(05:03:00)    Request #R1 added to vehicle #V1.
(05:03:00)    Setting route for Vehicle #V1...

(05:07:00)    Vehicle #V2 created.
(05:07:00)    Request #R2 added to vehicle #V2.
(05:07:00)    Setting route for Vehicle #V2...

(05:10:30)    Request #R1 is in transit.
(05:10:56)    Request #R1 has been delivered.
(05:11:00)    Vehicle #V3 created.
(05:11:00)    Request #R3 added to vehicle #V3.
(05:11:00)    Setting route for Vehicle #V3...

(05:13:15)    Request #R3 is in transit.
(05:14:09)    Request #R2 is in transit.
(05:14:07)    Request #R3 has been delivered.
(05:15:59)    Request #R2 has been delivered.

(05:18:00)    Request #R4 added to vehicle #V2.
(05:18:00)    Setting route for Vehicle #V2...

(05:24:28)    Request #R4 is in transit.
(05:24:30)    Request #R4 has been delivered.

(05:44:00)    Request #R5 added to vehicle #V3.
(05:44:00)    Setting route for Vehicle #V3...

(05:50:00)    Request #R6 added to vehicle #V1.
(05:50:00)    Setting route for Vehicle #V1...

(05:52:41)    Request #R6 is in transit.
(05:52:35)    Request #R5 is in transit.
(05:53:57)    Request #R6 has been delivered.
(05:59:15)    Request #R5 has been delivered.

(06:00:00)    Request #R7 added to vehicle #V1.
(06:00:00)    Setting route for Vehicle #V1...

(06:02:00)    Vehicle #V4 created.
(06:02:00)    Request #R8 added to vehicle #V4.
(06:02:00)    Setting route for Vehicle #V4...

(06:02:11)    Request #R7 is in transit.
(06:05:35)    Request #R7 has been delivered.

(06:12:00)    Request #R9 added to vehicle #V1.
(06:12:00)    Setting route for Vehicle #V1...

(06:14:00)    Request #R10 added to vehicle #V4.
(06:14:00)    Setting route for Vehicle #V4...

... continues ..

=====
Current Time:   19:57:00
=====

=====
Vehicle #V1
=====
```

```

Route for Vehicle #V1
-----
Route Start Time: 05:03:00
05:02:00      Request #R1 was added
05:10:30      Request #R1 was picked up
05:10:56      Request #R1 was dropped off
05:49:00      Request #R6 was added
05:52:41      Request #R6 was picked up
05:53:57      Request #R6 was dropped off
05:59:00      Request #R7 was added
06:02:11      Request #R7 was picked up
06:05:35      Request #R7 was dropped off
06:11:00      Request #R9 was added

... continues ...

=====
Vehicle #V2
=====

Route for Vehicle #V2
-----
Route Start Time: 05:07:00
05:06:00      Request #R2 was added
05:14:09      Request #R2 was picked up
05:15:59      Request #R2 was dropped off
05:17:00      Request #R4 was added
05:24:28      Request #R4 was picked up
05:24:30      Request #R4 was dropped off

... continues ...

=====
Vehicle #V3
=====

Route for Vehicle #V3
-----
Route Start Time: 05:11:00
05:10:00      Request #R3 was added
05:13:15      Request #R3 was picked up
05:14:07      Request #R3 was dropped off
05:43:00      Request #R5 was added
05:52:35      Request #R5 was picked up
05:59:15      Request #R5 was dropped off

... continues ...

=====
Vehicle #V4
=====

Route for Vehicle #V4
-----
Route Start Time: 06:30:00
06:01:00      Request #R8 was added
06:13:00      Request #R10 was added
06:29:00      Request #709 was added
06:57:12      Request #R8 was picked up
07:03:20      Request #R10 was picked up
07:09:10      Request #R8 was dropped off
07:11:06      Request #R10 was dropped off

... continues ...

-----
BUILD SUCCESSFUL (total time: 8 seconds)

```

CURRICULUM VITAE

Name Surname: Dilek EKİZ

Place and Date of Birth: Istanbul, 04.01.1989

E-Mail: arikand@hotmail.com



EDUCATION:

- **B.Sc.:** 2013, Istanbul Technical University, Faculty of Electrical and Electronics Engineering, Department of Electronics Engineering (Double Major Program)
- **B.Sc.:** 2012, Istanbul Technical University, Faculty of Science and Letters, Department of Mathematics

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2012 Highest Honor Degree Graduation in Istanbul Technical University, Department of Mathematics Engineering.
- 2010-2011 Erasmus scholarship in Durham University, Department of Mathematical Sciences

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Arıkan, D., Öztoprak, Ç., Sariel, S., 2017. Empirical Analysis of Optimization Methods for the Real-World Dial-a-Ride Problem. *Applications of Evolutionary Computation*, Springer International Publishing, 589–600.