REPUBLIC OF TURKEY

AKDENİZ UNIVERSITY

CREATING AGILE AND OPTIMAL TRANSPORTATION SYSTEM

Batuhan BULUT

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER OF SCIENCE THESIS

APRIL 2021

ANTALYA

REPUBLIC OF TURKEY

AKDENİZ UNIVERSITY

CREATING AGILE AND OPTIMAL TRANSPORTATION SYSTEM

Batuhan BULUT

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER OF SCIENCE THESIS

APRIL 2021

ANTALYA

REPUBLIC OF TURKEY
AKDENİZ UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

CREATING AGILE AND OPTIMAL TRANSPORTATION SYSTEM

Batuhan BULUT

DEPARTMENT OF COMPUTER ENGINEERING

MASTER OF SCIENCE THESIS

This thesis was accepted unanimously by the jury on 28/04/2021.

Prof. Dr. Melih GÜNAY (Supervisor)
Assoc. Prof. Dr. Alper BİLGE
Asst. Prof. Dr. Kamer ÖZGÜN

ÖZET

ÇEVİK VE OPTİMUM ULAŞIM SİSTEMİ OLUŞTURMAK

Batuhan BULUT

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Melih GÜNAY

NİSAN 2021; 72 sayfa

Bu çalışma Antalya şehrinin toplu taşıma sistemini bilimsel bir zemin üzerine oturtarak yenilenmiş bir şekilde önermeyi amaçlar. Bu doğrultuda dikkat edilen unsurlar durak yoğunlukları, yoğun bölgeler arası mesafeler ve taşınan yolcu sayılarıdır. Bir milyon kadar biniş verisi, bu tezin yazıldığı sırada Antalya'nın toplu taşımasını üstlenen şirket tarafından sağlanmış olup diğer tüm ilişkili veri söz konusu şirketin API servisinden bir .NET Core komut satırı uygulamasıyla otomatize şekilde toplanıp Azure'da öğrenci kredisiyle açılmış bir PostgreSQL veritabanına ilişkisel olarak kaydedilmiştir. Veri üzerinde çalışmalar için geliştirme ortamı olarak Google Colab (bir Jupyter Notebook servisi) kullanılmıştır. Bu geliştirmeler Python dilinde yapılmıştır. Verilerin doğruluğu; rota, durak konumları, yoğunluk haritası gibi çeşitli değerlerin Google Maps ve KeplerGL üzerinde görselleştirilmesi ve incelenmesinin ardından DBSCAN ve K-Means algoritmalarıyla coğrafik data üzerinde kümeleme yapılmış ve Uniform Cost Search ile düğümler verim odaklı bağlanmıştır. Taşınan yolcu arttıkça pozitif, mesafe uzadıkça negatif etki eden bir fonksiyon tanımlanmıştır. Tüm üçlü, dörtlü ve beşli rota permutasyonları arasından genetik algoritma kullanarak rotalar seçilmiş ve tanımlı fonksiyona göre verimi yüksek olan çözüm elde edilmiştir.

ANAHTAR KELİMELER: Genetik Algoritma, Kümeleme, DBSCAN, KMeans, Toplu Taşıma, Uniform Cost Search

JÜRİ: Prof.Dr. Melih GÜNAY
Doç.Dr. Alper BİLGE
Dr. Öğr. Üyesi Kamer ÖZGÜN

ABSTRACT

CREATING AGILE AND OPTIMAL TRANSPORTATION SYSTEM

Batuhan BULUT

MSc Thesis in COMPUTER ENGINEERING
Supervisor: Prof. Dr. Melih GÜNAY
APRIL 2020; 72 pages

In today's world, increased and wide-spread population have increased the demand for public transportation. This study takes stop density, stop layout and passenger population of those stops into consideration and offers a better regulated public transportation network design that can satisfy the increased demand. In this study, boarding data is provided by the company that is in charge of Antalya's public transportation system. Remaining inputs are automatically taken from company's API service using .NETCore command line application and saved into a PostgreSQL database that is hosted on Azure. Google Colab, a Jupyter Notebook service, is then used as the development environment to process the data using Python language. After visualizing inputs such as bus routes, stop layout and passenger density on Google Maps and KeplerGL, with the use of DBSCAN and K-Means algorithms, data is clustered and a new way of connecting clusters is offered as a result of Uniform Cost Search. In the cost function, shortening the distance between clusters is assumed as a cost amplifier and increasing the passenger count is assumed to lower the cost. A good solution respect to cost function found with genetic algorithm from all triple, quadruple and quintet permutations of routes.

KEYWORDS: Genetic Algorithm, Clustering, DBSCAN, KMeans, Public Transportation, Uniform Cost Search

COMMITTEE: Prof.Dr. Melih GUNAY
Assoc.Prof.Dr. Alper BILGE
Asst.Prof.Dr. Kamer OZGUN

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Dr. Melih GÜNAY for his guidance and would like to thank all my teachers for their precious lessons also. I am especially grateful to my family for their support. Most importantly, I thank to my girlfriend, Günay PAYLI. Tolerated and helped me all the way throughout this process with her kindness and skills. I'm happy to have her.

LIST OF CONTENTS

TEXT OF OATH

I declare that this study "Creating Agile and Optimal Transportation System", which I present as master thesis, is in accordance with the academic rules and ethical conduct. I also declare that I cited and referenced all material and results that are not original to this work.

28/04/2021

Batuhan BULUT

ABBREVIATIONS

csv	: Comma Separated Values
json	: Javascript Object Notation
ID	: Identifier
SQL	: Structured Query Language
http	: Hyper-text Transfer Protocol
API	: Application Programming Interface
IP	: Internet Protocol
4NF	: Fourth Normal Form
ORM	: Object Relation Mapper
SDK	: Software Development Kit
EF	: Entity Framework
DBSCAN	: Density Based Spatial Clustering of Applications with Noise
k-nn	: k-nearest neighbors
UTM	: Universal Transverse Mercator
VM	: Virtual Machine
TNDP	: Transit Network Design Problem
RNDP	: Road Network Design Problem
UTNDP	: Urban Transport Network Design Problem
GA	: Genetic Algorithm
DC-Elbow	: Double Center Elbow

List of Figures

List of Tables

1. INTRODUCTION

Public transport system designs are systems that need updating depending on the changing urban structure. In the case of an urban renewal is concerned, updating the system becomes necessary as an area becomes a residential area with the technology and population changes. In such cases, adding and subtracting can be done by hand in some systems and this situation brings the necessity of optimization. Although the system is based on a scientific basis or designed intuitively, advanced optimization can be applied.

In the content of this thesis, the transportation system of the city of Antalya will be discussed and it has been suggested how to follow a path in case of designing a new system.

Data is important and very valuable for an optimization work to be done in public transportation systems. Otherwise intuitively identify each variable can cause a misleading study. For this reason, the first step is to collect as much data as possible. For this study, Antalya's current public transport contractor firm was agreed as the provider. The first batch of data that comes with some delay contains two csv files as content. While one of them contains the line code, the displayed line code, the line start point, the line end point data; line code, line code representation, route code and direction (departure or return) columns. Since the contents have common values, the study started by checking the integrity of these data. The integrity errors found were solved by testing with assumptions.

Google Colab (a Jupyter Notebook service) has been used so that other people who will study with the data in the project, can jointly examine and run the code. In this way, those who want to access the data do not need to install any program or install an infrastructure on their local computers.

The content of the data received as the second party is the boarding data. The json file containing the boarding data contains ID, CardNo, BusId, BoardingTime, RouteCode, BusStopId information for each record.

In order to provide more details and data relations, the service offered by the provider as a website was discovered and the http request and http response packages were examined. In line with this analysis, a solution was made to collect and combine the data in

order to associate the data obtained with the available data and for easy access.

Since the API offered by the service is protected by authorization, the authorization token is obtained with a user and the information provided by the API methods is determined using the Postman program. Then, by writing a command line application under the framework of .NET Core, these calls to the API were autonomized and calls were made for all the requested information serially. The written code imports the csv containing the line and route ids into the database, analyzes the csv content containing the boarding data and adds it to the database by separating the valid and invalid ones, handles all the web requests for the data to be taken from the API, logs the traffic into text files and most importantly extracts the data and saves it in a format suitable for the PostgreSQL database.

While designing the database, the fourth normalized form (4NF) was used for the data that was thought to be important and frequently used, but some data were included in case it was necessary, and not much importance was given to its place in the design. For this reason, a data may be repeated sometimes in the same table.

The database obtained was moved to a PostgreSQL server on Azure. Access to the server from external IPs has been opened which can be considered a design flaw for production environments but this is just research. In this PostgreSQL server, due to the danger of changing / deleting the data that can be created by the administrator user, a user with only reading feature has been defined and the information of this user has been given to those who want to benefit from the data.

After the data was obtained, some studies were carried out by accessing the database via Colab. Since this study will propose a method of renovating the transportation system, only those related to this will be mentioned. The density of the boarding passes according to the region is of great importance during design. For this reason, temperature maps have been created with various methods and an idea has been obtained through them.

In line with these ideas, it was thought that it would be logical to start with clustering and a segmentation was tried to be obtained by using the DBSCAN algorithm. Although the outputs have some meanings, it is concluded that they are not sufficient for the purpose. After that, clustering was done using KMeans. Unlike DBSCAN in KMeans algorithm, the number of clusters can be given as a parameter to the algorithm. This is

a benefit in terms of directing the system by dictating the number of attraction centers to the clustering algorithm. As a result of different clustering studies made with KMeans, significant clusters were obtained among the busiest places of the city.

The clusters obtained have been evaluated as attraction centers where lines will pass through each other. These clusters were graphed and fully connected to each other. Then the roads in between were pruned according to a distance proportional to the city dimensions.

On the graph that connects the city attractions, the Uniform Cost Search is used to determine which edges to choose between each binary node. In this method, while designing the cost function, a formula that penalizes the distance and gives a reward according to the density of the nodes it has passed is chosen.

Different routes were created from the obtained nodes to suggest optimistic routes. From these route permutations, the closest route solution to optimum was searched using genetic algorithm.

Public transportation systems require ample updating due to ever-changing city structures. When urban transformation comes into question, areas where reconstruction is not permissible often end up being reconstruction permissible with the help of technology. That results in a wide-spread population and makes it so that existing system should be readjusted. These adjustments can be done manually in simple steps or they can be done in a scientific manner that makes use of service frequency, stop density and stop layout. Both approaches can be improved with further optimization. This study is based on Antalya's transportation network and offers a way to design a better system. Data being used in a design a public transportation system is naturally very crucial. In the case of faulty or not ample data, trying to guess variables intuitively would yield a badly designed system. For this reason alone, first step of this study is to get as much as trusted data. This is done by having Antalya's public transportation company's as data provider. First batch of data includes two csv files.

2. LITERATURE REVIEW

2.1. Transit Network Problem

In context of public transport systems, the transit network design problem (TNDP) aims to find a set of optimal routes regarding costs of the users and the operator. Since the quality of service highly depends on result of this design, TNPD, which is an NP-hard problem, has been studied for last five decades (J. Yang and Jiang 2020). While users expects cheap and direct transfer, comfortable terminal and vehicles, and bus frequency, operators' objective is making a system brings as much profit as possible (Guihaire and Hao 2008).

There are works with different goals to achieve and those works optimizes different parameters. T. L. Magnanti and R. T. Wong 1984 review them but since the comparison is old more newer version can be found in the paper by Crainic 2000 in which author reviews various state-of-art integer programming solutions of it's time to design network. A branch and cut algorithm to solve incapacitated fixed charge network flow problem has suggested (Ortega and Wolsey 2003). The genetic algorithm used in Ukkusuri, Mathew, and Waller 2007, to address robust network design problem.

In an old paper author approach to the problem with three related ways which are assigning passengers and vehicles to the routes as two different problems and finding optimal path in graph network as another problem. The number of vehicles are fixed to set a constraint to the problem. The next step is finding optimum routes and vehicle distribution on those routes while minimizing travel costs for passengers (Christoph E Mandl 1980a).

Sáez Aguado 2008 used Lagrangian relaxation based heuristic solve fixed charge transportation network design problem. The author divides the process to three phases in outlining aspect.

1. Apply lagrangian relaxation to get reduced costs of all variable.

2. Core problem, in which reduced cost of pair calculated, binary search algorithm applied, Lagrangian relaxation or decomposition applied to obtain heuristic.

3. Branch and Cut.

4. Iterate above.

When said cost effective transit planning, the literature assesses, this process usually divided into five elements Guihaire and Hao 2008; Ceder and Wilson 1986:

1. route design
2. frequency setting
3. building timetables
4. bus scheduling
5. crew scheduling

The first two of these elements are considered as mathematical programming problems which can be solved with simulations and/or heuristic algorithms. But in practice they are solved with intuition and experience of a local expert as pointed out by Deng and Yan 2019. Authors commented combining both is helpful to address network design complexity. Route design indices has provided as number of stops, route length, route directness (Equation 2.1 where u represents labels of bus routes), bus and metro overlapping, residential (Equation 2.2 where g is label of bus stop on route u, c is total number of bus stops on route u) and employment coverages, bus and metro connectivity, operation speed (Equation 2.3), annual average daily ridership while frequency design indices has provided as bus count, driver count, daily operation time, frequency, on-time arrival rate, annual average daily ridership.

$$route\ directness_u = \frac{route\ length_u}{Euclidean\ distance_u} \tag{2.1}$$

$$residential\ coverage_u = \sum_{g=1}^{c} \frac{residential\ heat\ value_{ug}}{c} \tag{2.2}$$

$$operation\ speed_u = \frac{route\ length_u}{average\ running\ time_u} \tag{2.3}$$

In a review made by Guihaire and Hao 2008, it is assessed that, steps of transit planning process and their combinations are referred with variety of names in the literature, while even many articles did not give any explicit name to the problem.

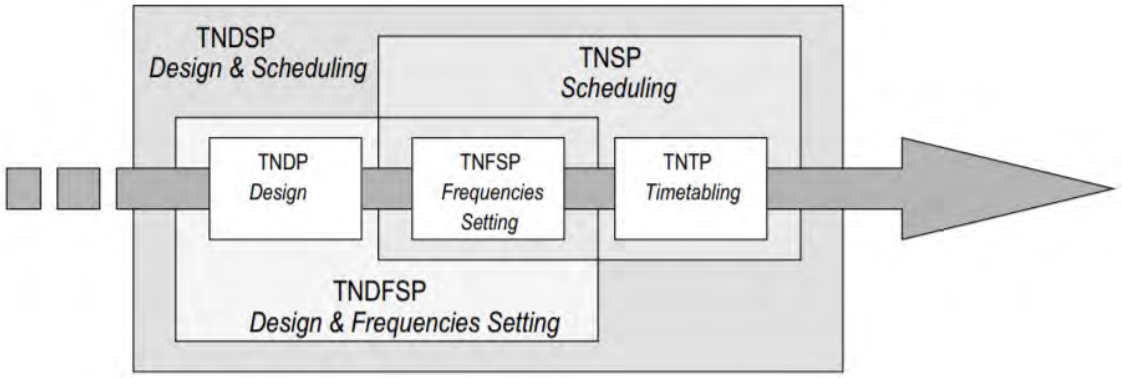

Figure 2.1. Proposed structure of transit network problems

Among these five elements, Deng and Yan 2019 reports how little attention given in previous studies for the route design. And J. Yang and Jiang 2020 emphasizes on the route set size parameter, since it is not only directly proportionate to fuel cost, but also may lead more direct services for passengers.

2.1.1. Transit Network Design Problem

Public transport, especially in urban areas, has increased with the increasing interest in improving transport systems. Although approaches to finding preferred bus routes in the urban public transport system were published until 1967, it has since been focused on this topic, especially Silman, Barzily, and Passy 1974, Van Vliet 1978, Hasselström 1979 Christoph E. Mandl 1980b, Baaj and Mahmassani 1991, A. Wren and D. O. Wren 1995, Nguyen, Pham, and Ha 2016, Canca et al. 2017 and Owais and Osman 2018 did. J. Yang and Jiang 2020 states that TNDP contains two parts: initialization and optimization.

Transit network design problem predicates on determination of a set of intermittent routes with bus stops between two terminals (Sáez Aguado 2008). Ceder and Wilson 1986 proposes that there are inputs and outputs for a network design, which are route performance indicators, demand and supply data as inputs, and route changes, new routes, operating strategies and service frequencies as outputs. Using an algorithm based on Kuhn-Tucker conditions, Furth and Wilson 1981 devised a method to optimize the allocation of buses to routes by maximizing net social benefit. Constantin and Florian 1995 formulated a mixed integer programming model to optimize frequencies to minimize pas-

senger travel and waiting time. Pattnaik, Mohan, and Tom 1998 proposed a genetic algorithm to solve transit routes and frequency optimization problem; thus minimizing both operator costs and travel time of users, and genetic algorithm was used to create transit routes. Claessens, Dijk, and Zwaneveld 1998 developed a programming model that minimizes operating costs subject to services and capacity requirements to solve the optimal rail route problem. Goossens, Hoesel, and Kroon 2006 proposed several models in which train wagons and stops of each line can also be designed by calculating the frequencies of each line to solve the railway line planning problem. Z. Yang, Yu, and Cheng 2007 proposed an optimization model for a bus network design problem that aims to maximize the number of direct passengers per unit distance subject to some restrictions; They proposed a parallel Ant Colony Optimization algorithm to solve the problem. Marín and García-Rodenas 2009 presented a two−level model for the optimal location of infrastructure in urban rail networks in order to minimize the travel time of passengers. They thought the rail network overlapped with other modes (buses, private cars, walking, etc.) and used a Logit model to simulate the multi-modal. Two network design criteria were considered: maximizing the coverage of the rail crossing network and minimizing the private vehicle transport network. Yu, Z. Yang, and Yao 2010 proposed a genetic algorithm that aims to minimize the total travel time of passengers subject to overall fleet size constraint to solve a two-level bus frequency optimization problem. Farahani et al. 2013 presented a comprehensive review of the definitions, classifications, goals, constraints, network topology decision variables and solution methods of the Urban Transport Network Design Problem (UTNDP), which includes both the Road Network Design Problem (RNDP) and Public Transport. And also about this UTNDP-RNDP relations have been published by Boyce 1984, Thomas L Magnanti and Richard T Wong 1984, Friesz et al. 1992, H. Yang and H. Bell 1998, Ngamchai and Lovell 2003, Dong and J. Wu 2003, Hu et al. 2005, Fan and Machemehl 2006, Pacheco et al. 2009, Kepaptsoglou and M. Karlaftis 2009, Saeidi 2014, Pternea, Kepaptsoglou, and M. G. Karlaftis 2015, S. B. Jha, J. K. Jha, and Tiwari 2019 and Chao Wang, Ye, and W. Wang 2020. Capacity of constraints is discussed by Zhu et al. 2020 with the correlation of multi-modal transportation and route.

Discrete road network design problem, capacitated multi commodity network desing problem, mixed network design problem, path based multi-commodity capacitated net-

work design, robust network design problem, capacitated single commodity network design problem, time constraint multi commodity network design, time-space charge network design problem and uncapacitated fixed charge network flow problem are not the concern of this work. The work in this thesis has different approach than the others since it has two steps which are clustering and constraint satisfaction with hyperparameter optimization. As a result, come up with a solution including new macroscopic routes.

2.2. Clustering GeoSpatial Data With DBSCAN

DBSCAN mostly intended to be used geographic data (Schubert et al. 2017a). It is an unsupervised machine learning algorithm to cluster spatial data with noise and not limited with euclidean distance like KMEANS. Geographic coordinate system in which consists latitude and longitude lines is not appropriate to use with euclidean distancing. The geodesic distance is the shortest distance on the surface of an ellipsoidal model of the earth. Geodesic distance calculated with geopy module for python programming language which implements geodesic algorithm in given method by Karney 2013.

One of the challenges while using DBSCAN is choosing parameters, $minPts$ and $epsilon$. $k = 2 * dimensionCount$ heuristic is is used initially for $minPts$ (Sander et al. 1998). But very low values of $minPts$ can cause duplicate data problem, it may be necessary to choose larger values for one of noisy, large datasets or containing more duplicates (Schubert et al. 2017b). Since the data in this thesis's scope has contain many duplicates -because of every boarding is a new point at same bus stop location- and has many bus stops that can be in rural areas which leads to algorithm think it is a noise point, it can be categorized as duplicate data problem as stated.

In Ren et al. 2016, authors have used DBSCAN with modifications. Proposes temporal-origin-destination combined with DBSCAN algorithm and forms DBSCAN variant with both spatial and temporal clustering called TOD-DBSCAN. Adopts widely used assumptions:

1. The previous trip destination is nearby the bus stop of next trip origin

2. Passengers return to the first boarding station of the day at the end of the day.

Authors didn't cluster points but they did cluster line segments and time information. The

purpose was clustering boarding points, alighting points and boarding time. As a result 49 customized city bus service trajectories extracted.

2.3. Clustering GeoSpatial Data With KMeans

KMeans is a heuristic algorithm that partitions a data set into K clusters by minimizing the sum of squared distance in each cluster (Singh, Yadav, and Rana 2013). Applied in three main steps:

1. Initialization center points randomly or predefined with given K value
2. Dividing all data points into K clusters
3. Update centroid locations based on formed clusters
4. Repeat step 2 and step 3 until converges

Default implementation of K-means is uses Euclidean distance (MacQueen 1967). Euclidean distance is not a consistent metric for geographic coordinate system. A projected coordinate system is defined on a flat, two-dimensional surface. Unlike a geographic coordinate system, a projected coordinate system has constant lengths, angles, and areas across the two dimensions (Maher 2013). UTM is a projected coordinate system (Buchroithner and Pfahlbusch 2017). Projecting geographic coordinates to a plane is used for working with euclidean distance.

2.4. Search Algorithms

Different search algorithms can be used for a search problem defined as graph or tree structures. Depth First Search, Breadth First Search, Depth Limited Search, Uniform Cost Search, Greedy Best First Search and A* Search are some of them. Depth First Search and breadth first search are most brute force approaches (Tarjan 1972; Bundy and Wallen 1984) while A* (A star) and Greedy Best First Search needs heuristics to run (Chunbao Wang et al. 2015; Xie, Müller, and Holte 2014). Depth Limited Search is a variation of Depth First Search with a depth limit (Korf 1990). Because of this limitation, algorithm may not find the solution because of early termination. Uniform Cost Search preferred

because it calculates cost of every branch in the fringe meantime, changes it's course back to another node if cost of that branch becomes lower than current (Htwe and Naing 2009).

3. MATERIAL AND METHOD

3.1. Line and Route Data Investigation

The data including line and route information was obtained on 31 January 2020 as an excel file named "Hat-Güzergah Bilgileri_20200131172858.xls". In order to increase the validity between all platforms, this file convert to csv file using Google Sheets and saved as "LineRouteInfo.csv". After saving, the file was analyzed on a Colab notebook named "LineRouteInfoAnalysis.ipynb". In order for files such as csv to be accessible by notebooks shared with the team, the collaborative folder on Google Drive has been added as a shortcut to the root folder of Drive. In this way, base path is not differ in file access from user to user, and it was not necessary to change variables in the notebook for each user.

3.1.1. Authorization For Google Drive

Drive package was imported from google.colab module in order to access files in Drive via notebook. Then, by calling the mount method from the drive package, the user's Drive is mounted to the notebook local. During the process, an informative text is displayed as a cell output, accompanied by a link directed by the article and a text box for input (Figure 3.2).

Figure 3.2. Colab, Drive authorization cell

Clicking on the link will take you to the Google account authorization page in a new browser tab. Once authorization is provided, an alphanumeric code appears. When this code is pasted into the Colab cell where the mount method is called, the text that appears in the output field after the call is executed and the Enter key is pressed, access to the Drive is given to the virtual machine where Colab notebook instance is running on. Since the VM has full access to the files until the Drive is unmounted, notebooks with untrusted code should not be given Drive access.

3.1.2. Uniqueness Control of Lines

Acting on the intuitive assumption that each line is unique and has 2 round-trip routes, the following propositions are made.

- There should be no more than two records for each unique line.
- Every line code has to match with only one line name.
- Every line should contain one route for going and one for coming back.

The suggestions tested and failed which indicates the assumptions were wrong at the first place. A line code can be related with more than one going route or more than one coming back route according to observations of test script outputs.

There is a relation between line code and route code in which if you multiply line code with 10 you get route code or route code plus one. The plus one indicates coming back route while minus one indicating going route. So the rows that doesn't match with that rule has removed. The observation was that the line code 15150 (SB15) has only 1 instance which is a going route. So when the file checked one can see coming back route has id of 151401 which conflicts with line code 15140 (SC14). When that route code has changed to 151501 as the rule suggests, everything has become on track.

Route code uniqueness has checked with a python script to double check and assume that every specific route should have only one match in lines list. As a result 609 unique routes found without error message about repetitive route code which legitimize chosen approach to the data.

Also 341 unique line code found while checking the other csv file's integrity.

A difference has detected between distinct line code counts of two csv files.

A cross check applied to find mismatching lines. To do this initially dictionaries filled with line codes as keys for both csv file separately. The findings shows that one of the files has 341 unique line codes while the other has 305. The larger one includes all the line codes the smaller one has. But since the smaller one also has the route code too it was chosen to be used as reference data.

3.2. Data Management

While the future data for the study were expected to be more comprehensive, only the arrival of stop descriptors and time-tagged ride data and the lack of spatial information, which is the most important of these data, required an alternative solution. Data has been harvested from the API for use with the data provided.

3.2.1. Exploration of Service API

The current source of the data is the service provider itself. Users use this service mainly as a mobile application, as well as a web-accessible portal. Authorization is required to use this portal. After obtaining the authorization, the user can call the vehicle with the line code on the service and view the route of this vehicle and the estimated transit times from the stops.

The methods of using the features offered by the portal and the authorization method have been debugged by following the HTTP requests and responses with the tools offered by the browser. During this process, it was determined that the endpoint used to obtain authorization, the GET method was used, and the parameters containing the username and password were sent as URL encoded. Information requests can be made to the service with the authorization code in the returned response. In this study, the line code information displayed to the public transportation users and the information that there is a departure or return route, the endpoint, where the route and stop locations of that line are taken, was used to collect data. For example, when we query the bus known with the code VF01, the content in the API RESPONSE attachment in the attachments section from the service returns. Response is quite long (2500 lines and remainder). For this reason, a brief summary of the returned content is shared below:

- Information on whether the transaction was completed successfully or not
- Line code information not sharing the users
- The explanation of line code shown to the users (VARSAK ALTIAYAK - FAKULTE)
- Array of points expressing the route when combined with each other with sequence information (in geographic coordinate system)
- Departure times depending on the days of the week of the vehicles running on that line

Some of the data mentioned here were not understood at the first stage, but were later resolved by thinking. For example, while coding departure times, it has been determined that there are 3 categories as MTWTFss, mtwtfSs, mtwtfsS and departure times related to them. With some thought, it is assumed that these are Monday, Thuesday, Wednesday, Thursday, Friday, Saturday, Sunday and that the shared departure times are for the day marked with a capital letter. In addition, it was noticed during the visualization studies that, apart from the stop locations, the locations shared with sequential tags in a series called points were also combined with each other to draw the route.

3.2.2. Database Design

Before the data was collected, it was thought to be placed in a database in order to be accessible quickly and easily, to be open to others to work and to increase its sustainability. Accordingly, the database was designed to comply with the Fourth Normal Form (4NF) rules as possible. During this design, the fields that are returned from the service's API and which are not clear are not used in accordance with the 4NF rules.

As a database, PostgreSQL, which is popular among open source databases, has been chosen for its performance, reliability and community. In the future, PostgreSQL can respond to these within the scope of a NoSQL structure requirement.

The designed structure was converted into models with .NET Core SDK in C# language and converted into a database with the Code First technique. Integration of PostgreSQL and Entity Framework Core, which is the ORM tool used by .NET Core, is provided with Npgsql provider.

The database scheme designed in line with the data obtained is reflected on Figure 3.3.

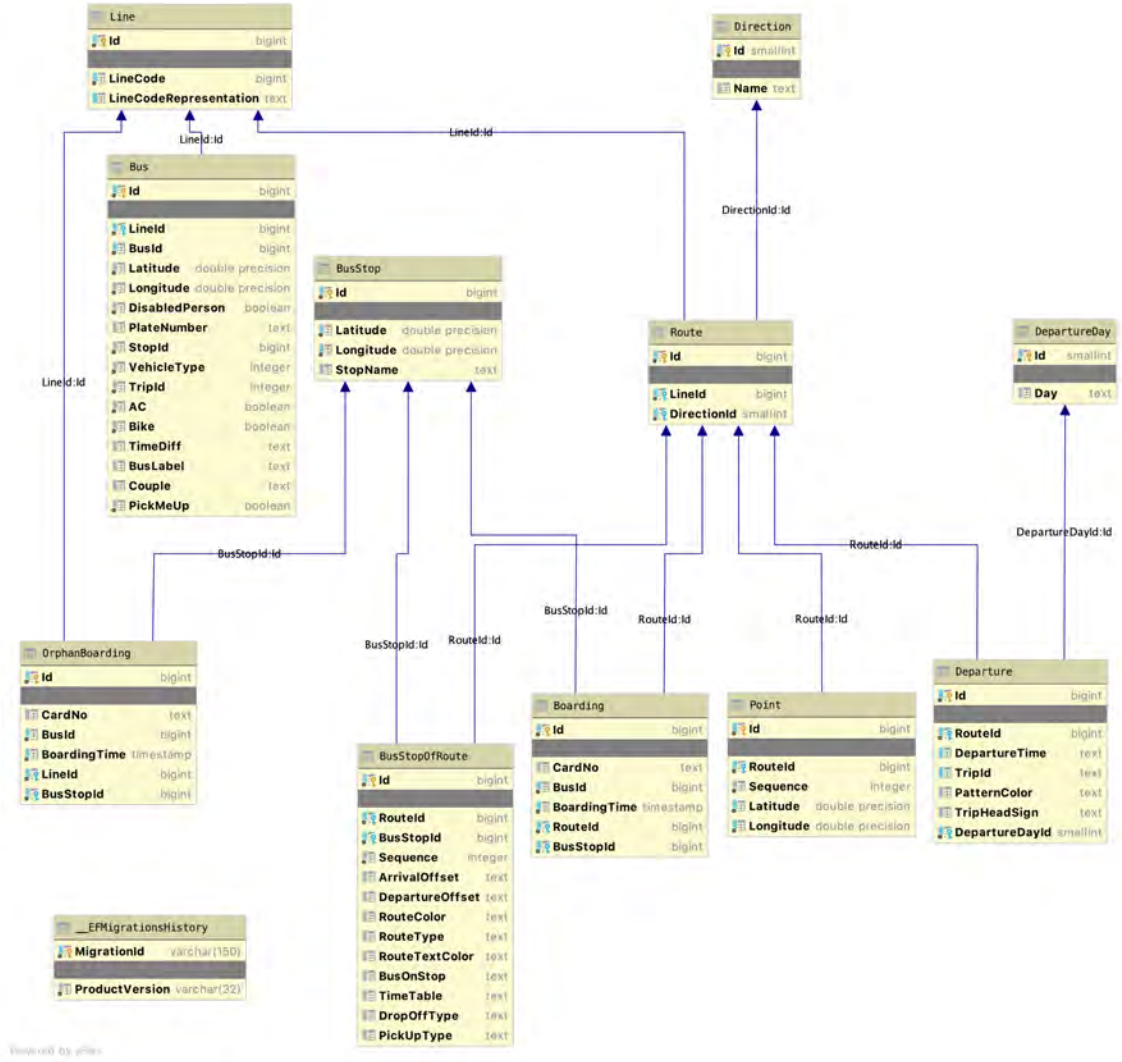

Figure 3.3. Database schema

The database created was then migrated locally to Azure's PostgreSQL service. A free student loan with the Github Education Pack was used to cover the service charges.

3.2.3. Importing Line Code Data

Before adding the csv data provided by the service provider, it is necessary to add the definition of direction and day of the week. Direction definitions are in two types, Going and Return, while the day type definitions of the week are classified as Weekday,

Saturday, Sunday. After the definitions are made, the data are taken into the database.

3.2.4. Data Gathering

Information requests are sent to the service with a .NET Core CLI application that includes all combinations of line codes and going / return parameters written in the database previously provided and created, and the returned responses are parsed.

The command line application prompts the user for the username and password used for the service and gets authorization.

Then, for each added line, two requests are sent, one for the departure route and the other for the return route. BusStop, BusStopOfRoute, Departure and Point tables are filled with the returned responses.

All json objects returned as responses are mapped to related classes with names that are more meaningful or appropriate to the naming convention of the language. An example of this is the following class:

```
public class Data : Response
{
    [JsonPropertyName("pathList")]
    public List<Path> PathList { get; set; }
}
```

Returning responses are saved to the file before being processed so that there is no need to request again in line with the errors that may occur.

3.2.5. Importing Boarding Data

The json file was read and processed into the database including the boarding data provided by the service provider. During this process, logging has also been implemented and is not included in the database in order to observe that some boardings are not from a registered stop or a registered line code. If the data comes from a known stop of a known line but its route is not clear, it is included in the OrphanBoarding table.

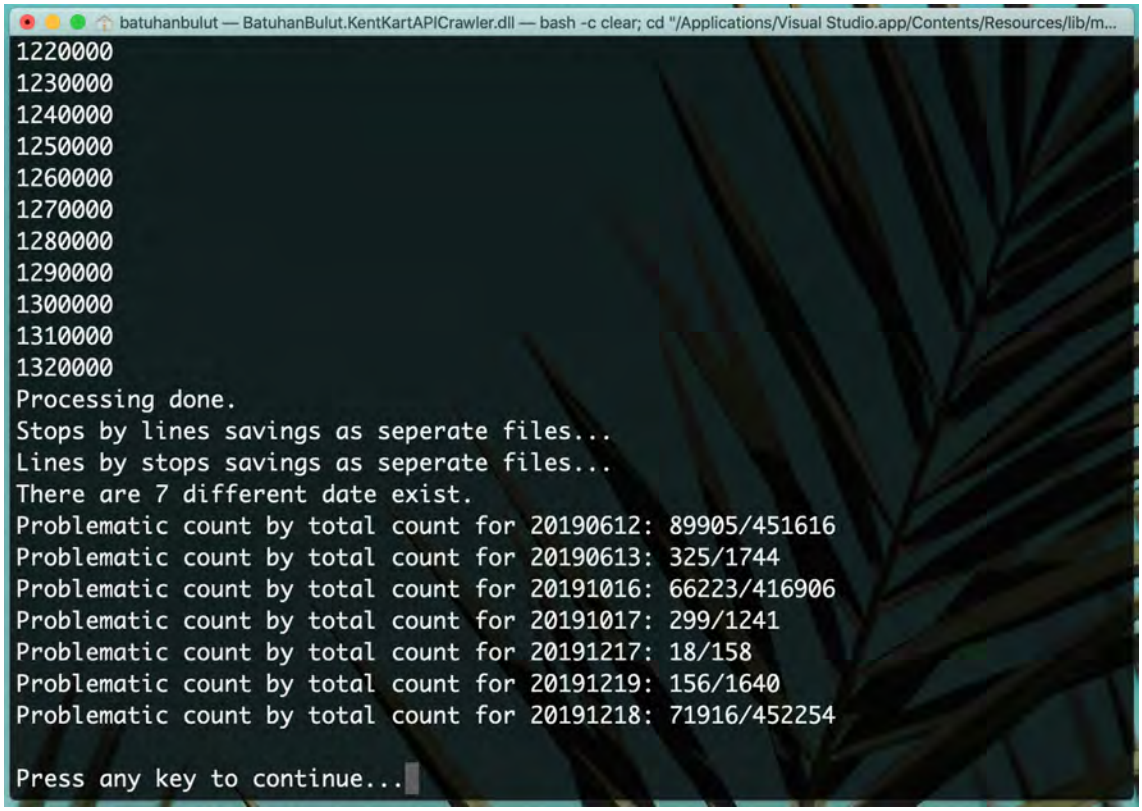

Figure 3.4. Boarding import process output

As a result of the output, there are records for 7 different days, how many of these records are problematic in the Figure 3.4.

3.3. Visualization of Problematic Boardings

In order to examine why problematic data is problematic and perform the first visualization of the data on this occasion, an implementation has been made on Google Colab using Google Maps.

To use the Google Maps API, an API key must first be obtained. This API should be kept as a key secret and should not be shared. Usage above a certain quota is charged.

The class that will hold the series of stops on a route is defined. This class has methods according to need, such as bringing the leading stop, the trailing stop, and the stops in between.

The station location and sequence data, which were previously taken from the database and containing route information, were saved as csv files to be used in this notebook.

Where the map will be centered during its creation is taken as a parameter. Since we want all the stops to fit and center on the visible map, a method that finds the middle point was needed to solve the issue of centering. This method simply averages the latitude and longitude parameters of the coordinates by taking them cumulatively and dividing them by the total number of coordinates. Then, a personal API key was provided to the gmaps module, and the gmaps figure was created with the calculated map center, the zoom level that will include all the stops and the map type by selecting the road map.

After the figure has been created, a method has been defined to draw lines that define the route by combining the drawing layer with the positions. Line thickness and line color can be determined with method parameters. The method draws a line between each successive pair of coordinate points given.

After all the necessary methods are ready, the drawing layer is added on the figure and saved as an html page. It is shown in Colab cell with embedded html pointer.

The cell output is presented on Figure 3.5.

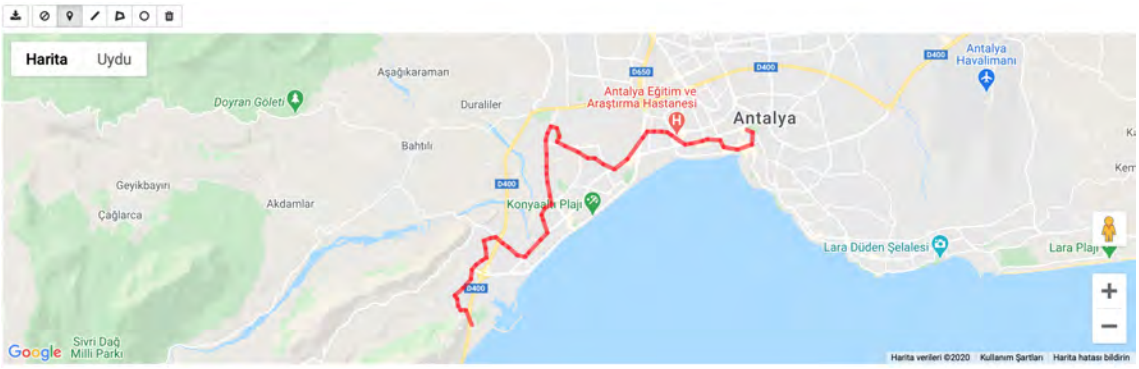

Figure 3.5. Google Maps on Google Colab cell output

The zoomed image is displayed as Figure 3.6.

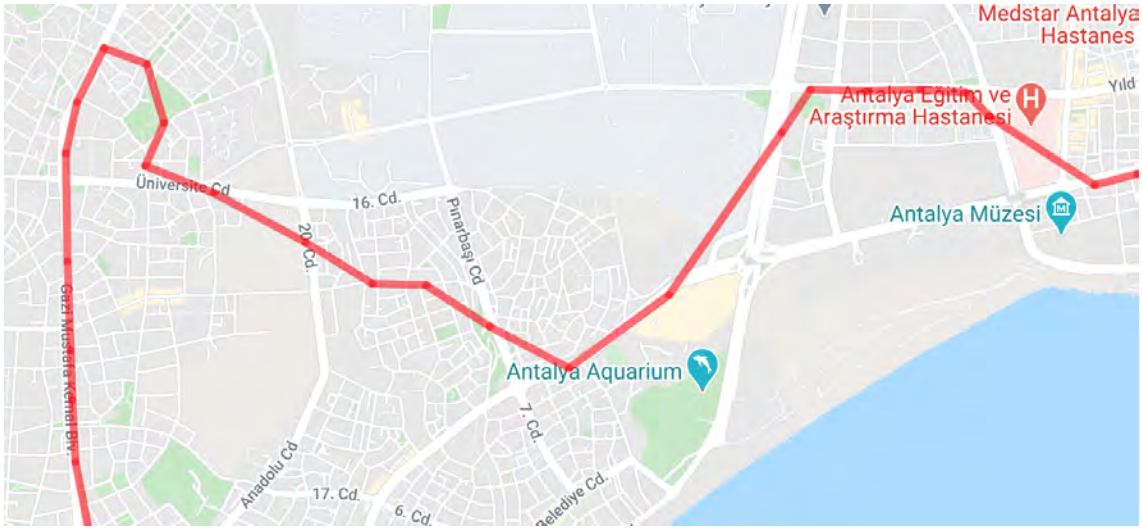

Figure 3.6. UC32 route built with stops

Due to the fact that the stops are not sufficient in drawing the route, the relevant data from the Point table has been drawn as csv to be used in this notebook. A point table is a set of points placed more frequently than stops in order to define a route more precisely. When combined, it defines the route.

With the csv obtained, a higher resolution route drawing was obtained, which was reflected by means of Figure 3.7, by applying the same process as previously applied.

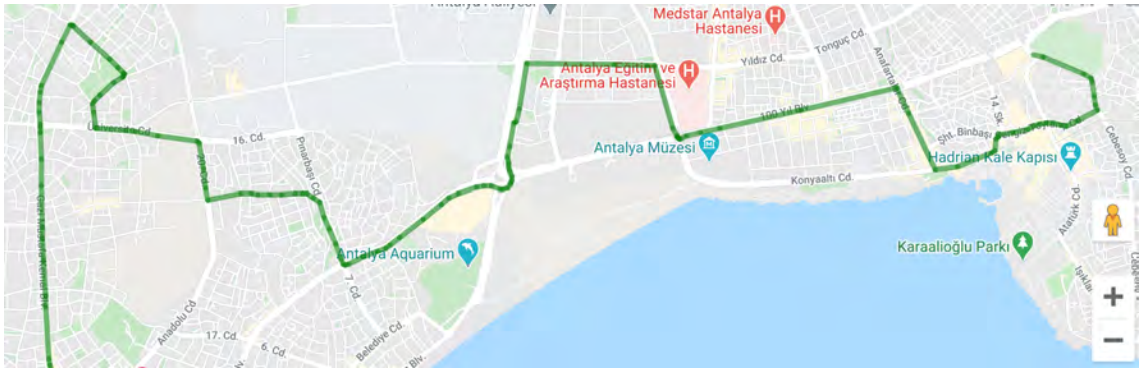

Figure 3.7. UC32 route built with points

The difference between the route (red) created by combining the stops and the route (green) created by combining the points is shown with Figure 3.8.

Figure 3.8. UC32 route built with stops (red) vs points (green)

After the determine the route, the display of the stops obtained from the API on the route is also provided and shown as 3.9.

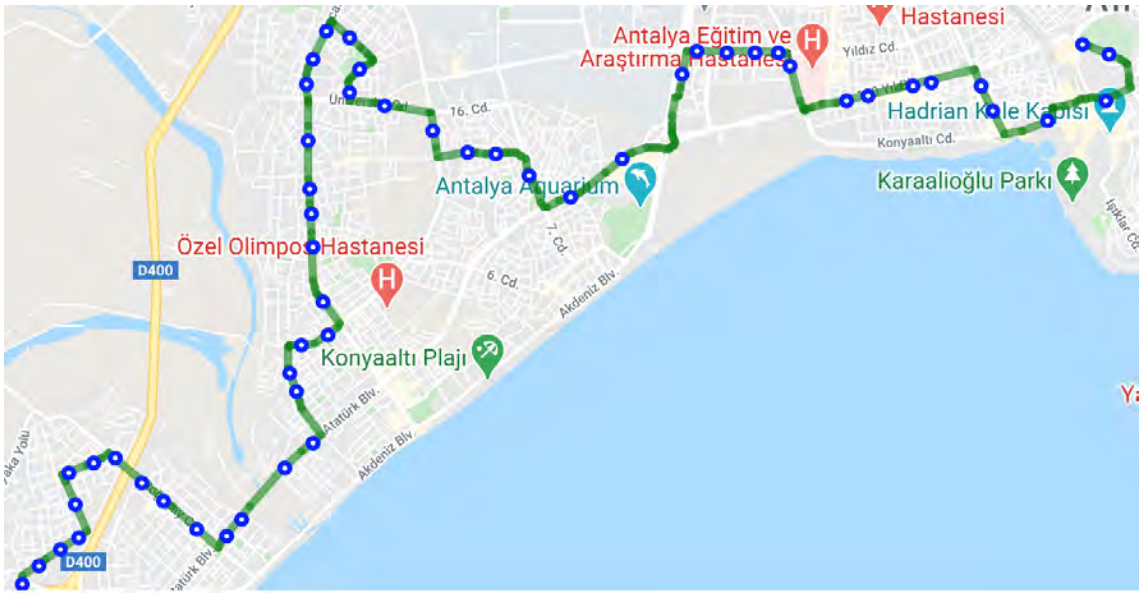

Figure 3.9. UC32 route and stops

When the stops with problematic boarding stops are added to this route, the result is shown as red stops on Figure 3.10 for the direction of departure and return respectively.

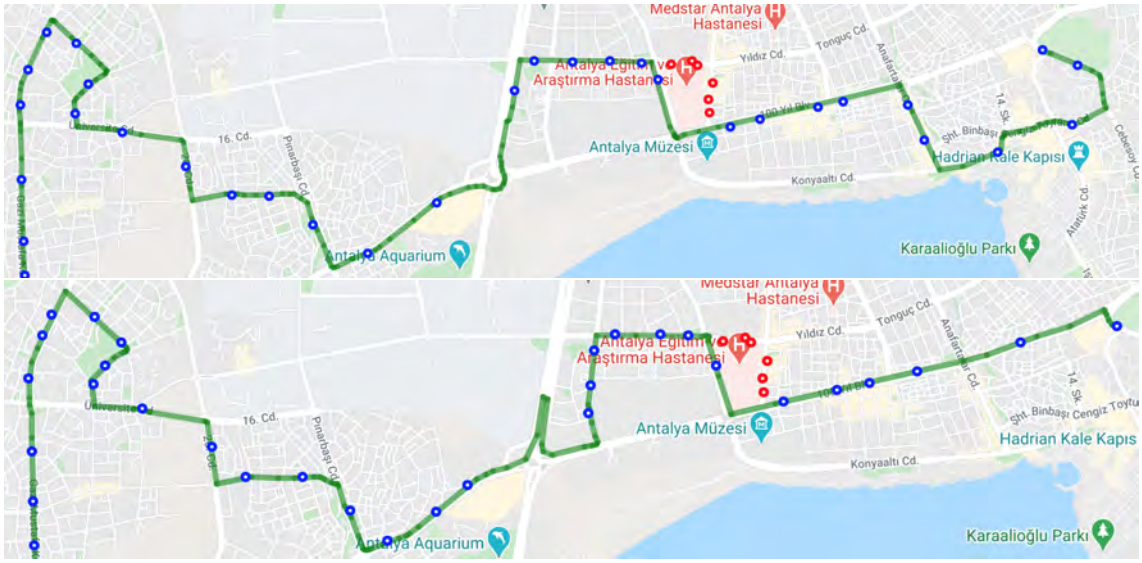

Figure 3.10. UC32 problematic stops on going route

As it can be understood, the problematic stops for the line considered represent an old route. The differences between the date of taking the routes and stops and the date of the boarding data and the arrangement of the stops led to this conclusion that is most logical. Since the neighborhood did not cause a change in the number of boarding passes, it was considered that this difference would not be of great importance during the study and this error was decided to be ignored.

3.4. Connecting PostgreSQL Database from Google Colab

Due to the problems such as the frequent need for data in different ways, the cumbersome need to prepare csv each time; Access method notebook was prepared for the convenience of team access to data and studies from the common center and the ability to run the code by someone else without the need for extra dependencies, with the aim of making the central database directly accessible on Colab notebooks. In order not to affect the data due to an error, only a user named read_only_user, who has read privileges, is opened in the database.

3.5. Boarding Heatmap

It is shown on the map in heatmap form in order to have an idea about the density distribution of the boarding passes. This process was carried out on a Colab notebook.

The data was pulled from the PostgreSQL server with the cheapest plan open with free credit in Azure. For this, the module used for postgresql access to the notebook was installed. In addition, the Google Maps module has been installed.

The data has been pulled to the pandas Dataframe to take the central districts from the database. Districts with no significant traffic are excluded by latitude and longitude restrictions. In the query, the total number of boarding passes that could not be connected to the route and the boarding passes with a route were taken on the basis of stops.

To understand the structure of the data, the first 5 records are shown on Table 3.1.

Table 3.1. First 5 rows of boarding counts by stop

#	Latitude	Longitude	totalsum
0	36.860943	30.624914	189
1	36.860028	30.625748	504
2	36.858166	30.623222	174
3	36.853399	30.618057	1662
4	36.850912	30.615393	146

While the maximum value of the total sum was seen as 26402, the mean value was measured as 374.27093425605534.

The general view and the view of the key region with some zoom is shown on Figure 3.11. It is understood that state hospitals, shopping centers, business centers, bazaars and universities are the busiest areas. The intensity is gradually decreasing in the form of red, green and blue, with the white color being the most intense.

The same data is displayed on Kepler.gl as Figure 3.12. The image has added a little more resolution to the understanding with the 3D view capability and increased the possibilities of the researcher to establish relationships.

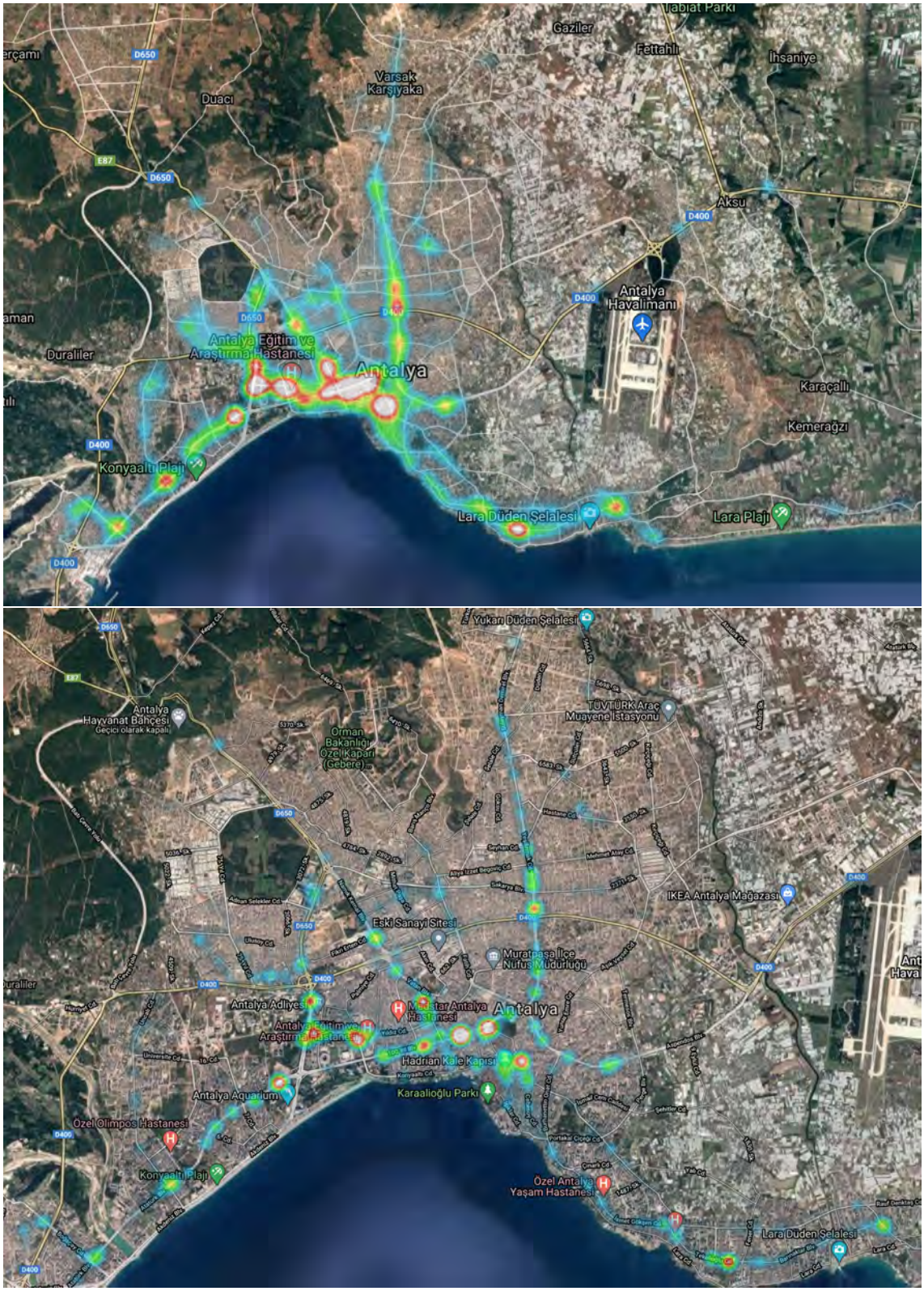

Figure 3.11. Heatmap on Google Maps

Figure 3.12. Heatmap on Kepler.gl

3.6. Clustering Boarding Data

The centers of regional densities must be identified in order to propose a new transport. The number of these centers and their distances from each other and the number of boardings of these centers are the parameters that play an important role in this determi-

nation.

KMeans and DBSCAN algorithms have been applied to create these regions. The technique that can create the most successful result has been selected by examining the results.

When applying DBSCAN, firstly the selected distance metric distance matrix is obtained to determine the epsilon parameter. The distance metric chosen for this problem is the geodesic distance due to the latitude and longitude of the data. Meter is used as the unit, therefore the epsilon value to be determined is also in meters. Run k-nn (such that k = MinPts) with the matrix obtained. A one-dimensional array is obtained by taking the furthest points from the k closest points of each obtained point. This sequence defines the distance of each point to the nearest kth point. The dots are arranged in ascending order and plotted. The epsilon parameter of the DBSCAN algorithm is selected from the image with the heuristic method called the knee or elbow method.

The clusters obtained were evaluated with their centers and the weights of the centers were determined depending on the number of boarding passes. These centers are connected to each other in graph structure and each center has become a node. The connection between 2 nodes is called edge and the value is expressed as cost, and this cost is calculated with a function proposed in this thesis. While this function proposal penalizes the distance with an increase, it also rewards it with an increase in the average of the 2 node densities in question.

After the graph was created, the most suitable path to go from one point to another was found with the Uniform Cost Search algorithm. All 2-node combinations are input to this algorithm via a loop and all travel edges between nodes are determined.

3.6.1. DBSCAN

Before the clustering algorithm is applied, the data of the districts outside are excluded so that they do not affect the whole. This is provided as in the example below:

```
cur = conn.cursor()
cur.execute("""
  SELECT
    BS."Latitude",
```

```
    BS."Longitude",
    (SELECT count(*) FROM "OrphanBoarding" OB
      WHERE OB."BusStopId" = BS."Id") +
    (SELECT count(*) FROM "Boarding" B
      WHERE B."BusStopId" = BS."Id") AS TOTALSUM
  FROM "BusStop" BS
  WHERE BS."Latitude" > 36.831178
    AND BS."Latitude" < 37.011045
    AND BS."Longitude" > 30.586582
    AND BS."Longitude" < 30.918065;
""")
```

While each stop is included in the system as a point, the number of boardings made from that stop is used as the weight of that point.

Distance matrix is calculated only once, in cases where the same distance metric is used, it saves a lot of time while experimenting with different parameters. Apart from this, since there is no geodesic distance among the predefined metrics used in the implementation of the DBSCAN algorithm in the sklearn module, it is necessary to calculate and give the distance matrix separately.

K-nn is run with the calculated distance matrix. When choosing the k value, the formula $k = 2 * dimensionCount$ is used as rule of thumb (Sander et al. 1998).

Only the largest (last index) of each point is taken from the nearest k point.

When we combine these distance values with the table showing the boarding amount at the stops and look at the first 5 elements, we see it in the form of Table 3.2.

Table 3.2. First 5 rows of boarding counts by stop

#	Latitude	Longitude	totalsum	k_{dist}
0	36.860943	30.624914	189	168.146104
1	36.860028	30.625748	504	159.209047
2	36.858166	30.623222	174	343.095967
3	36.853399	30.618057	1662	333.539491
4	36.850912	30.615393	146	246.898525

These distance values are listed in ascending order and the graph is drawn. The plotted graphic will help us find the critical number called knee or elbow.

As seen on Figure 3.13, elbow is between 1000 and 400 meters.

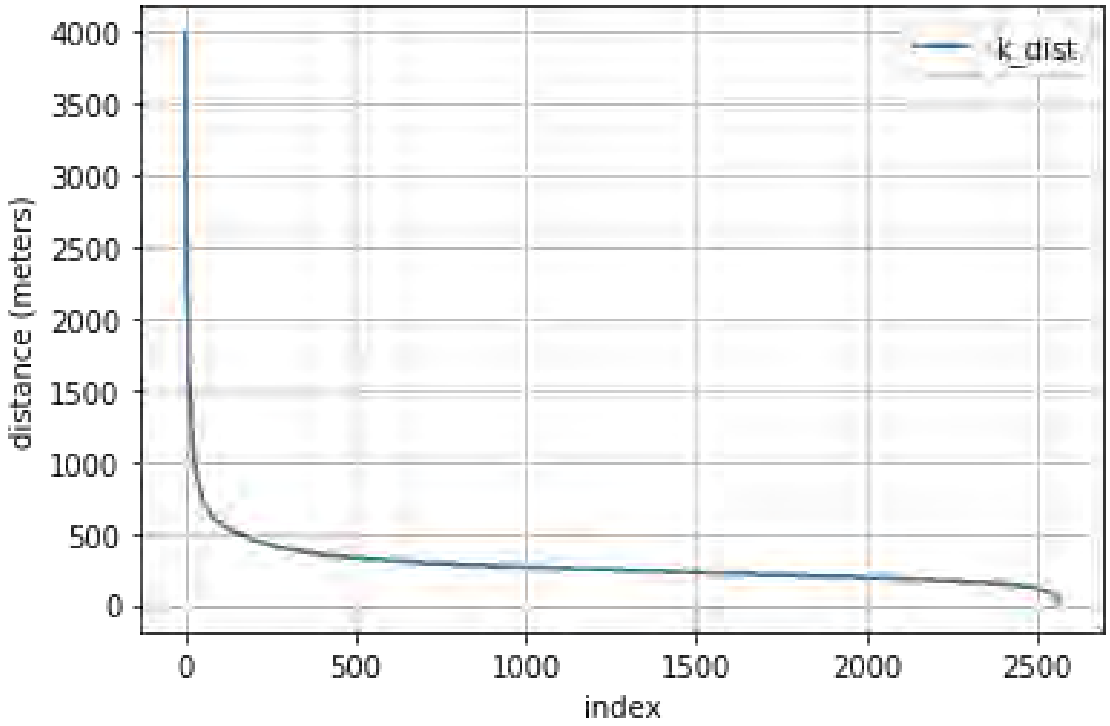

Figure 3.13. KNN max distances ordered descending with $k = 4$

After the heuristic epsilon value is selected with elbow method, DBSCAN algorithm is operated.

3.6.2. K-Means

The KMeans implementation of the sklearn module does not support receiving pre-computed distances matrix and the distance metric is euclidean distance. In order to be suitable for this, latitude and longitude data have been projected to 2 dimensional UTM coordinates. Thus, Euclidean distance is provided to be a logical application.

The values obtained are separated with their weights to be used as new coordinates. The converted values are shown on Table 3.3.

Table 3.3. First 5 rows of Latlon projection convertions to UTM

#	Latitude	Longitude	totalsum	Easting	Northing	ClusterLabel
0	36.860943	30.624914	189	288272.834183	4.082080e+06	10
1	36.860028	30.625748	504	288344.625935	4.081977e+06	10
2	36.858166	30.623222	174	288114.296693	4.081775e+06	6
3	36.853399	30.618057	1662	287640.599560	4.081258e+06	6
4	36.850912	30.615393	146	287396.163881	4.080988e+06	6

Elbow method has been applied to find the correct cluster number. In order to do this, the kmeans algorithm has been trained with the same data from 2 clusters to 30 clusters. The inertia value of each model is plotted based on the corresponding k value.

In determining the k value of KMeans, a new method has been developed due to the inadequacy of the standard elbow method in terms of reaching sufficient resolution. This method examines the distance between the centers of gravity and geometric centers of clusters. The number of clusters is increased to the point where the centers of gravity and geometric centers of the clusters stop converging significantly. The number of clusters is found when it comes to the point where the geometric and gravity centers does not become closer significantly even if k increased. Here the weighted center phenomenon comes from different weights of the elements in a cluster. As an example in the Figure 3.14 we can see there is no improvement after 24 clusters.

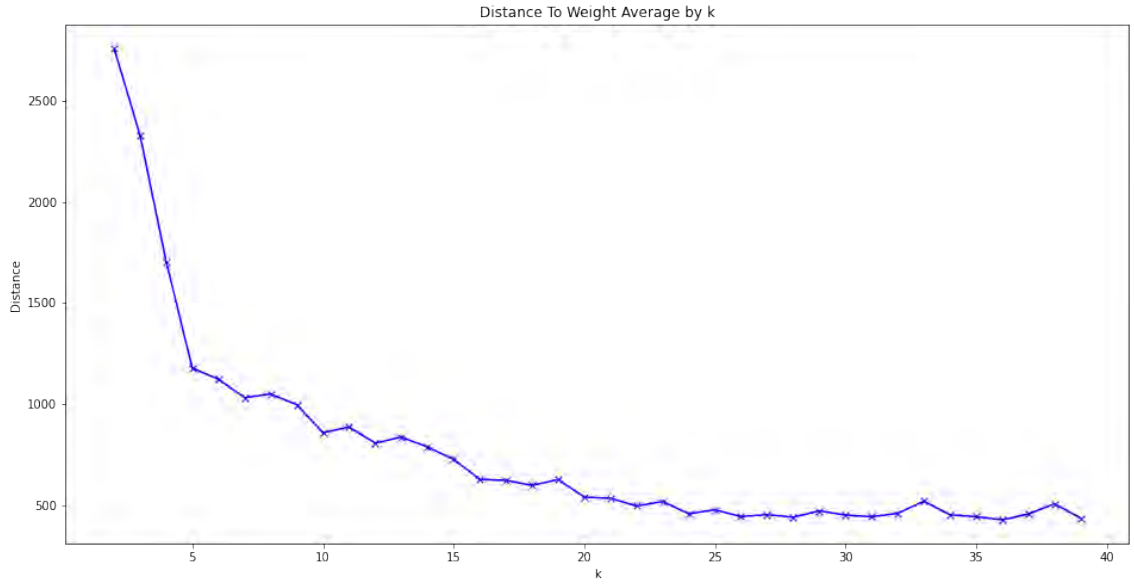

Figure 3.14. DC-Elbow method for best k selection

3.7. Pathfinding

As the last step, the clusters created are treated as nodes, connected to each other with edges between them and converted into a graph. After a pruning operation on this graph, the route of going from one point to another is defined with a cost function, with the Uniform Cost Search search algorithm.

3.7.1. Building Graph

First of all, a Node class is defined, it takes location, boarding count, stop count and name parameters. The name parameter was later used to match the input of the most logical path algorithm between 2 nodes. The Edge class represents the bidirectional path between 2 nodes and the length of this route in km.

Nodes were created for clusters found with these defined classes. Latin alphabet has been used to give nodes unique names among themselves. Node locations are taken as the center of gravity of clusters.

After the nodes are created, the distances between nodes are calculated to also create the edges. In order for not all nodes to make a connection between each other, no edge has been drawn between nodes above a certain distance threshold. This is chosen as the

average of all distances between threshold nodes.

After taking out the list of distances and examining the graphic, edges are created and the successor of each node is connected to it. It was observed that the total number of edges created was 153.

The efficiency function between two clusters is designed to affect the sum of the boarding count values of the clusters in direct proportion, while the distance between them is designed to affect inversely. Although the division operation is in the range of $[0, 1]$, 1 is added to the result and the efficiency function definition is made simply.

$$Efficiency = 1 + \frac{(Node1_{potential}) + (Node2_{potential})}{Distance}$$

The distance between each binary node is calculated with the efficiency function and efficiency matrix is generated.

Edges are represented as lines on the drawing layer in order to show the graph on google maps. The letters assigned as names to the cluster center of gravity were added on the marker and reflected on the map as seen on Figure 3.15.

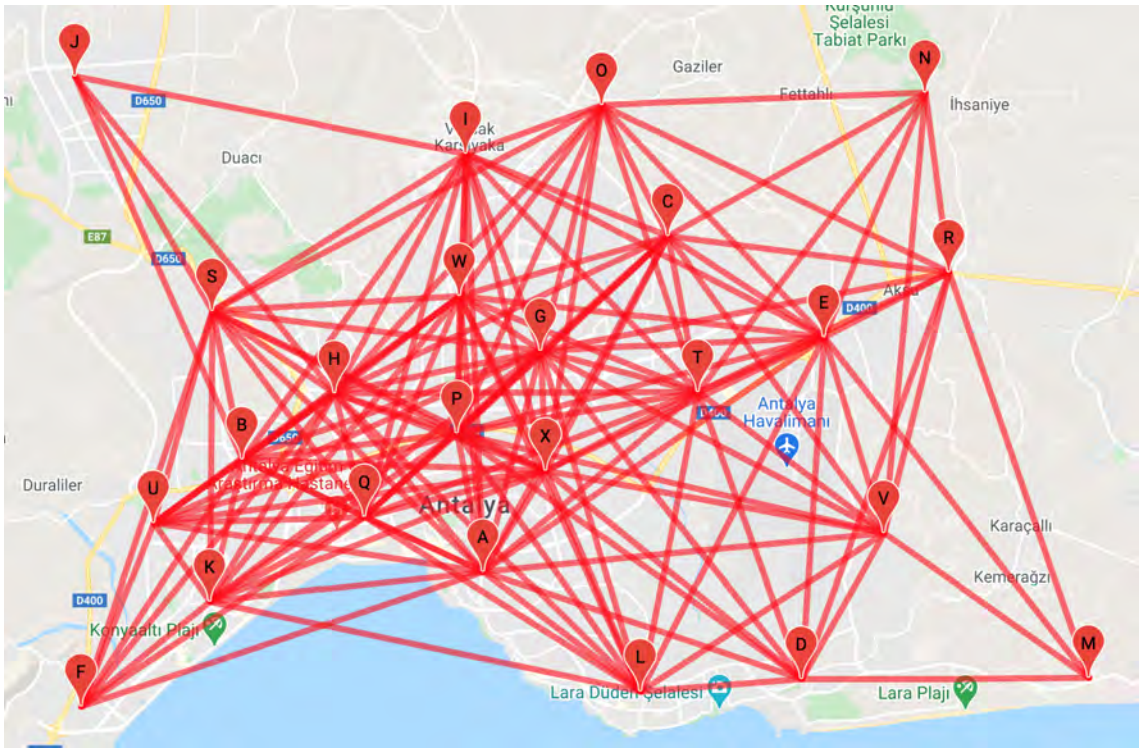

Figure 3.15. Cluster graph on maps

3.7.2. Uniform Cost Search

The Uniform Cost Search search algorithm has been used to determine which route to follow in case a journey is required between two nodes. The algorithm has been implemented in python language. As the cost function, the inverse of the efficiency function according to multiplication is taken so that as the efficiency increases, the cost decreases. Despite the possibility that the Efficiency value is zero, precautions have been taken against the undefined division by zero in the form of maximum cost.

3.8. Genetic Algorithm

Since the processing burden of choosing the best solution includes routes that are efficient and coherent among combinations of all routes obtained by connecting the nodes to each other in quintet permutations is very high. This is the reason why genetic algorithm preferred.

The new efficiency function defined as below:

$$Efficiency_2 = f(x) \times \frac{\sum_{i=0}^{N} nodes[i].potential}{\sum_{i=0}^{N-1} distance(nodes[i], nodes[i+1])}$$

$$f(x) = \begin{cases} y, & y > 0.8 \\ 0.9y, & 0.7 < y \leq 0.8 \\ 0.8y, & 0.6 < y \leq 0.7 \\ 0.6y, & 0.5 < y \leq 0.6 \\ 0.2y, & y \leq 0.5 \end{cases}$$

$$y = \frac{X_{euclidean}}{\sum_{i=0}^{N-1} distance(nodes[i], nodes[i+1])}$$

The $nodes$ parameter represents the nodes of a specific route. The formula basically sums up the node potentials to find route potential. Then divides it to total distance which is sum of road lengths between sequential route nodes. Then multiplying it with directness factor which is calculated by dividing euclidean distance between route start node to route end node (X) with an easing function (f) multiplication to total distance.

The routes out of 10km-30km range are eliminated. Then, all remaining routes were examined programmatically to make sure that each node was found at least once in the problem space.

For a genome to represent the solution, each chromosome in the genome represents the presence or absence of a route in that solution. For this, each chromosome is a boolean value and the genome length is equal to the total number of routes which is 17145. When creating the first population, the maximum number of active chromosomes in the genomes was limited to 1/3 of the total number of nodes. While generating the genome, random chromosomes are activated with a %50 probability.

The fitness function sums up the $Efficiency_2$ score of each active route within the genome. If there is an edge in the same direction between 2 nodes in the route suggestions, 1 point removed from fitness score for each encounter. 10 points are deducted for each node that is not included in solution.

A selection function is defined to select the candidates to crossover from among the population. Each time this function is called, it randomly selects two parents from the population. During this random selection, the probability of being selected was defined in direct proportion to their fitness scores.

The crossover function describes the way the crossover process between 2 genomes is performed. The minimum active index and the maximum active indexes were found for 2 genomes and random numbers were generated in this range. From the point of this random number generated, the genomes were split in two and each piece was combined to correspond with the other partner. Thus, two new genomes were produced. Here, instead of directly producing value between zero and chromosome number for random selection, the process of finding the minimum and maximum value range is that the ratio of active chromosome number to total number of genomes is very small. This creates the possibility that all active genes remain to the right or left of the chosen point when the range is not determined.

The identified mutation function has a %50 probability of logically inverting a random gene. It's performed this experiment three times in a row.

The algorithm has been tested with different parameters and the values have been tweaked. Initial population member count is set to 30. At first, algorithm is set to stop

after 100 generations if it cannot exceed the desired fitness value. But a patience counter implemented which increases when the solution score is equal to previous and resets when greater was a better approach since it stops iterations when no improvement seen.

3.9. Domain Migration of Current System and Analysis

The routes of the existing system has converted into routes in the proposed system domain. To do this, every stop in the current system is included to nearest node in the proposed system. Converted routes with less than 3 nodes eliminated because of external connections out of problem boundary. For forward direction (there are two directions: forward and backward) there are 249 lines which consists 117 unique converted routes. After the elimination filter applied 125 lines left with 98 unique converted routes. Efficiency, directness and transfer counts of active and proposed system compared. Algorithm proposed for adding extra routes to the proposed system in need of decreasing transfer counts.

4. RESULTS AND DISCUSSION

4.1. DBSCAN

It was created by taking $k = 4$ of the DBSCAN method, and over the ordered k-nn values reflected on the Figure 3.13, $epsilon = 600$ was taken to get a heuristic meaning with the elbow method and 17 classes except noise were formed. Cluster distributions were not as expected as seen on Figure 4.16. DBSCAN has gathered most of the city in the red cluster.

The black dots represent the weighted centers of the clusters where they are located, while the colored dots represent the stops. Colors allow us to distinguish the cluster to which they are attached.

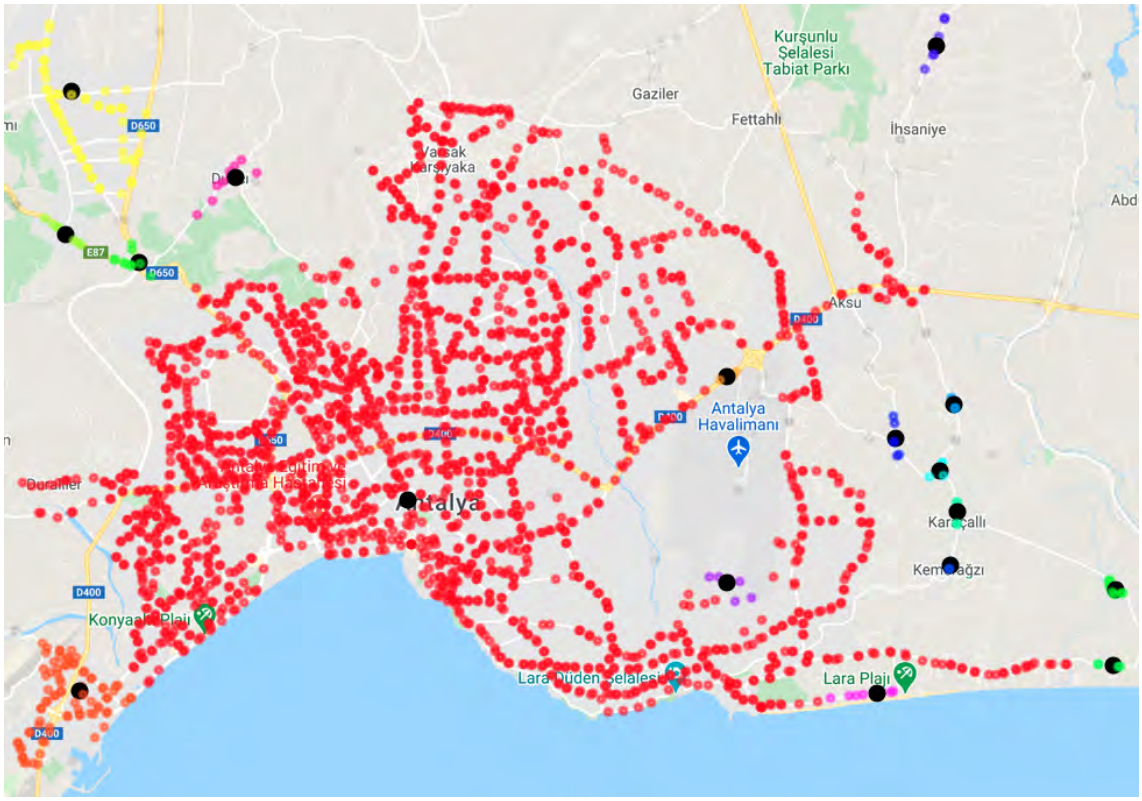

Figure 4.16. DBSCAN with $ininPts = 4$ and $epsilon = 600$

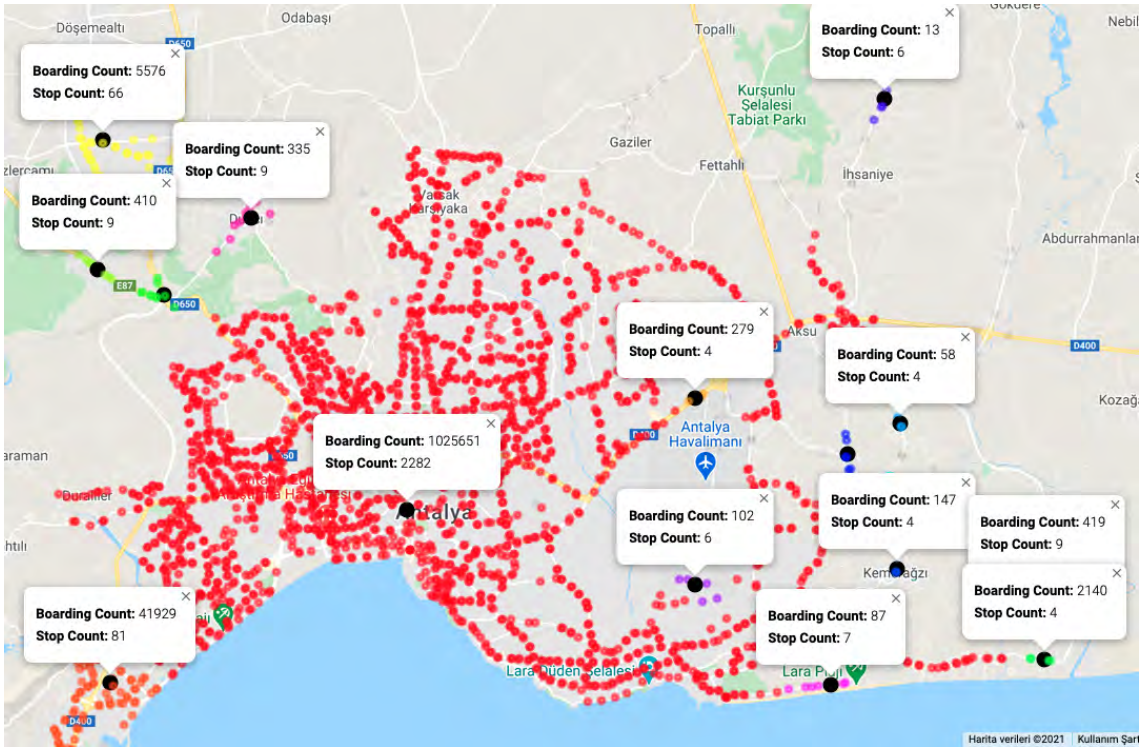

Figure 4.17. DBSCAN $MinPts = 4$, $epsilon = 600$ with boarding info

One of the reasons why the cluster, expressed in red, occupies a large density, may be to have the algorithm weighted. In other words, since $k = MinPts = 4$ is selected, having 4 trips at one stop is enough for that stop to be a cluster alone. This problem is referred to as duplicate data problem. To fix that it may be necessary to choose larger values for very large data, for noisy data or for data that contains many duplicates (Schubert et al. 2017b) which indicates our case because of every boarding from same bus stop (same location) is a data point.

In this case, after MinPts is determined, it is recalculated with the formula $k = MinPts - 1$ ((Schubert et al. 2017b). With the new calculated k value, epsilon was calculated and DBSCAN was applied again with new parameters.

MinPts were re-obtained by taking the average of boarding passes on the basis of stops.

```
min_pts = int(weights.mean())
min_pts
```

When found as $MinPts = 420$, it becomes $k = 419$ from the formula $k = MinPts - 1$.

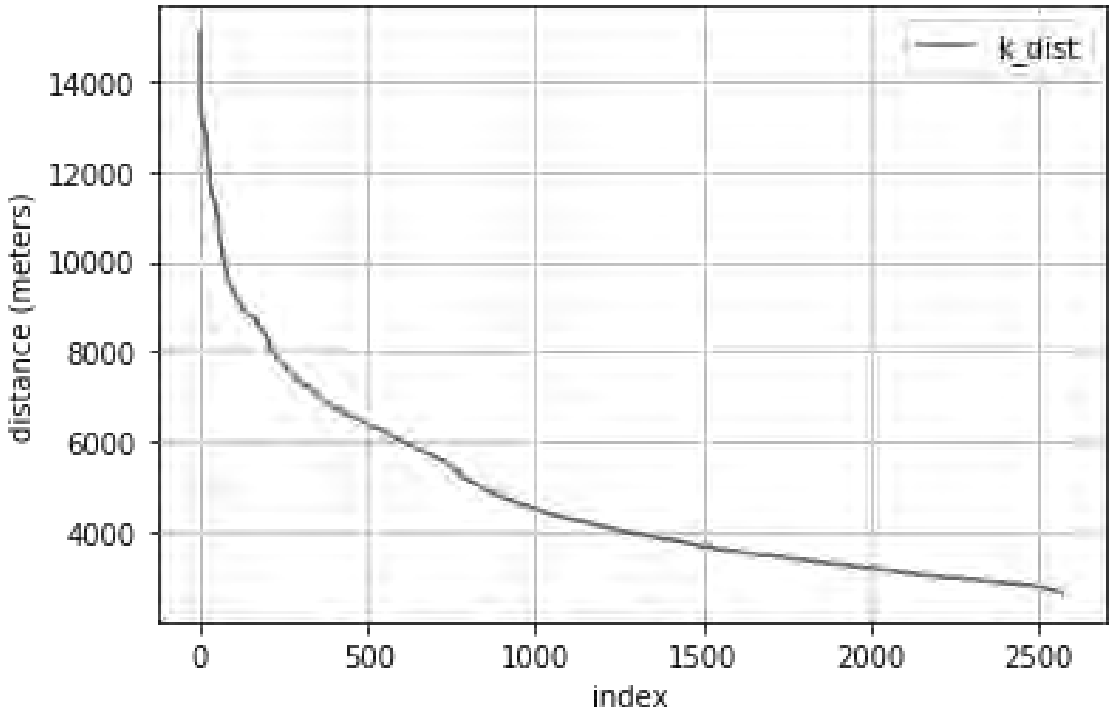

Figure 4.18. KNN max distances ordered descending with $k = 419$

When interpreting on the Figure 4.18, it can be thought of the elbow 9000, 7000 and 5000 meters. When an action is taken for each of these values, the result is that clustering cannot be performed.

```
min_pts = int(scipy.stats.gmean(
    weights.totalsum.to_numpy(),
    axis=0, dtype=None))
min_pts
```

It is aimed to reduce the effect of extreme radical values by taking the geometric means. When $MinPts = 82$, it becomes $k = 81$ from the formula $k = MinPts - 1$.

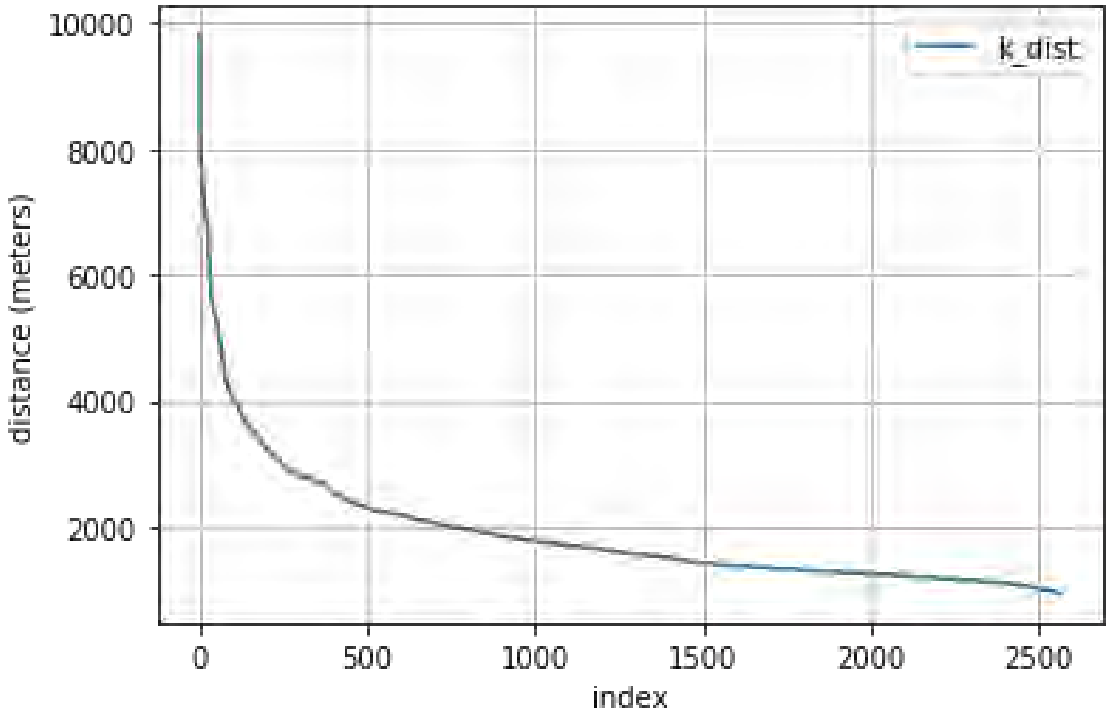

Figure 4.19. KNN max distances ordered descending with $k = 81$

When interpreting on the Figure 4.19, elbow is selected as 3000. When there was no result, different epsilon values were tried, and 5 clusters were obtained in 1100. Clustering results 4.20 and information shown on Figure 4.21.

Figure 4.20. DBSCAN with $MinPts = 82$ and $epsilon = 1100$

Figure 4.21. DBSCAN $MinPts = 82$, $epsilon = 1100$ with boarding info

Since the clustering result obtained contains less cluster and takes a lot of data points as noise, the more logical MinPts value was searched and the elbow method was used again for epsilon calculation. When $MinPts = 12$, the k-dist plot is as shown in Figure 4.22.

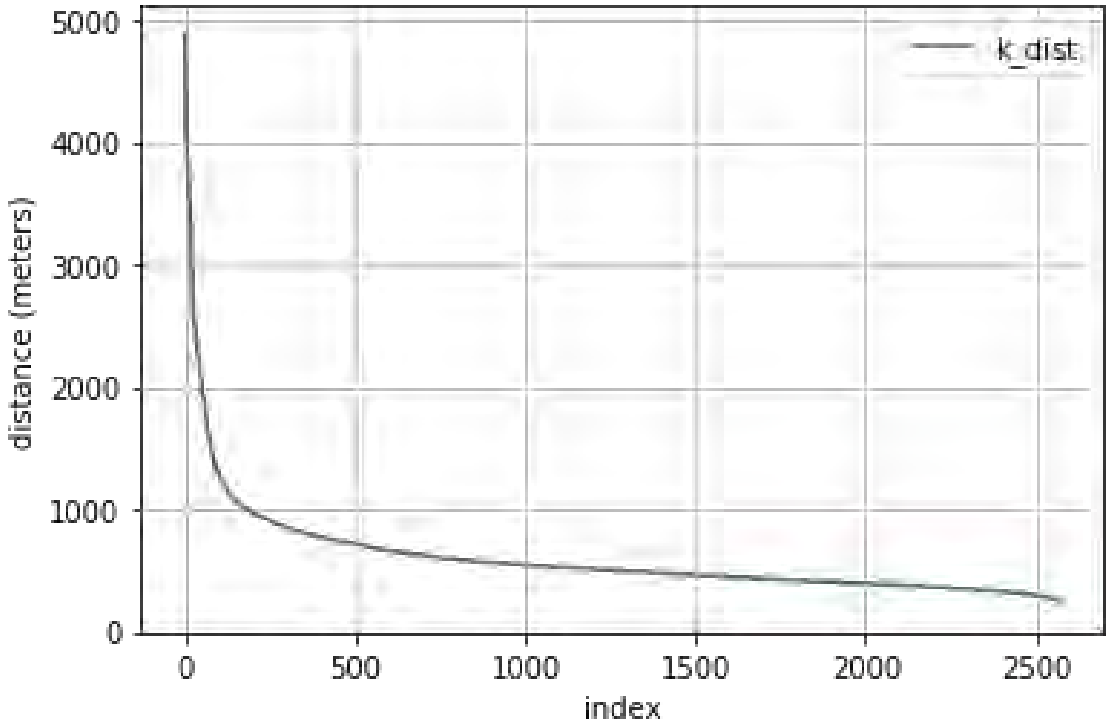

Figure 4.22. KNN max distances ordered descending with $k = 11$

According to the Elbow method, it was first selected as $epsilon = 1100$, this caused the middle parts of the city to be included in a single cluster. In order to overcome this problem, the epsilon value was decreased gradually (100 to 100 each). The value $epsilon = 400$ provided a higher number of clusters, and the result is shown on Figure 4.23.

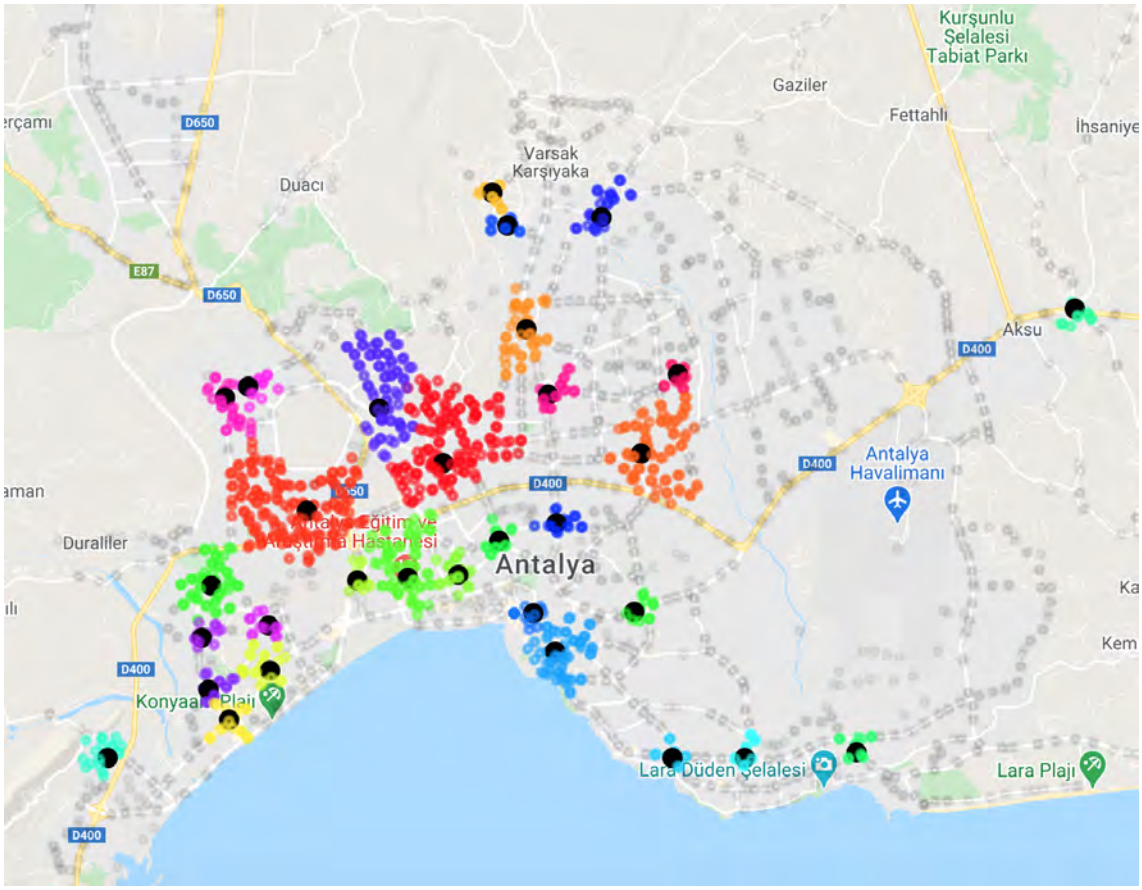

Figure 4.23. DBSCAN with $MinPts = 12$ and $epsilon = 400$

When the result obtained was examined, it was determined that too many data points were marked as noise.

4.2. K-Means

When the Elbow method is applied between $k = 2$ and $k = 30$, a curve in the form of Figure 4.24 occurs.

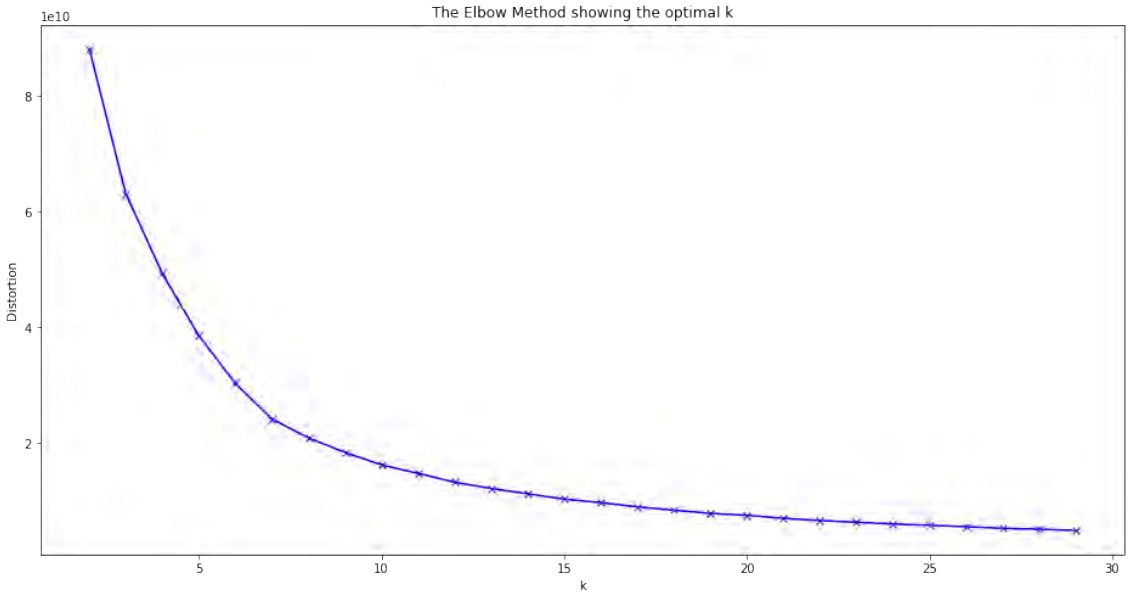

Figure 4.24. Elbow search

First approximation is $k = 7$ (Figure 4.25), followed by $k = 15$, $k = 20$ (Figure 4.26) and $k = 25$ (Figure 4.27) have been tested. It has been observed that even if the k value determined by the Elbow method is exceeded, meaningful clusters continue to occur for the problem.

The black dots represent the weighted centers of the clusters where they are located, while the colored dots represent the stops. Colors allow us to distinguish the cluster to which they are attached.

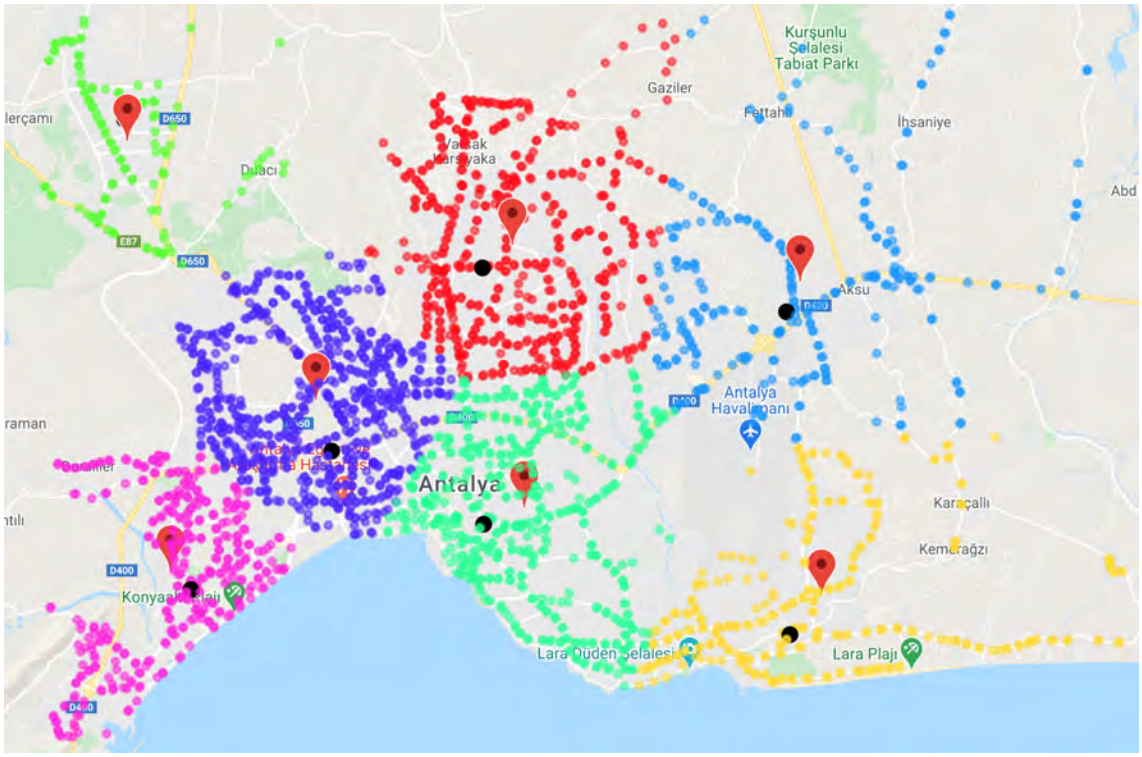

Figure 4.25. KMeans with $k = 7$

The $k = 7$ selection with elbow method looking fine. Clusters almost as big as districts. It might be a good choice of k if we want to analyze it in district traffic scope. But having more clusters creates a chance for having more resolution in the solution.

Pins are representing the centroids of clusters while black dots are weighted centers. We can see the distance between two centers in same cluster are not close respect to cluster geometry. This can be an indication of insufficient amount of clusters.

Increasing the k value is a need up to a point. Which arises another question. To what extent? Because as the elbow method suggests, if you go further in linearity in elbow curve, you don't except any valuable improvement but it doesn't specify where to stop if your valuable clusters are increasing in linearity zone.

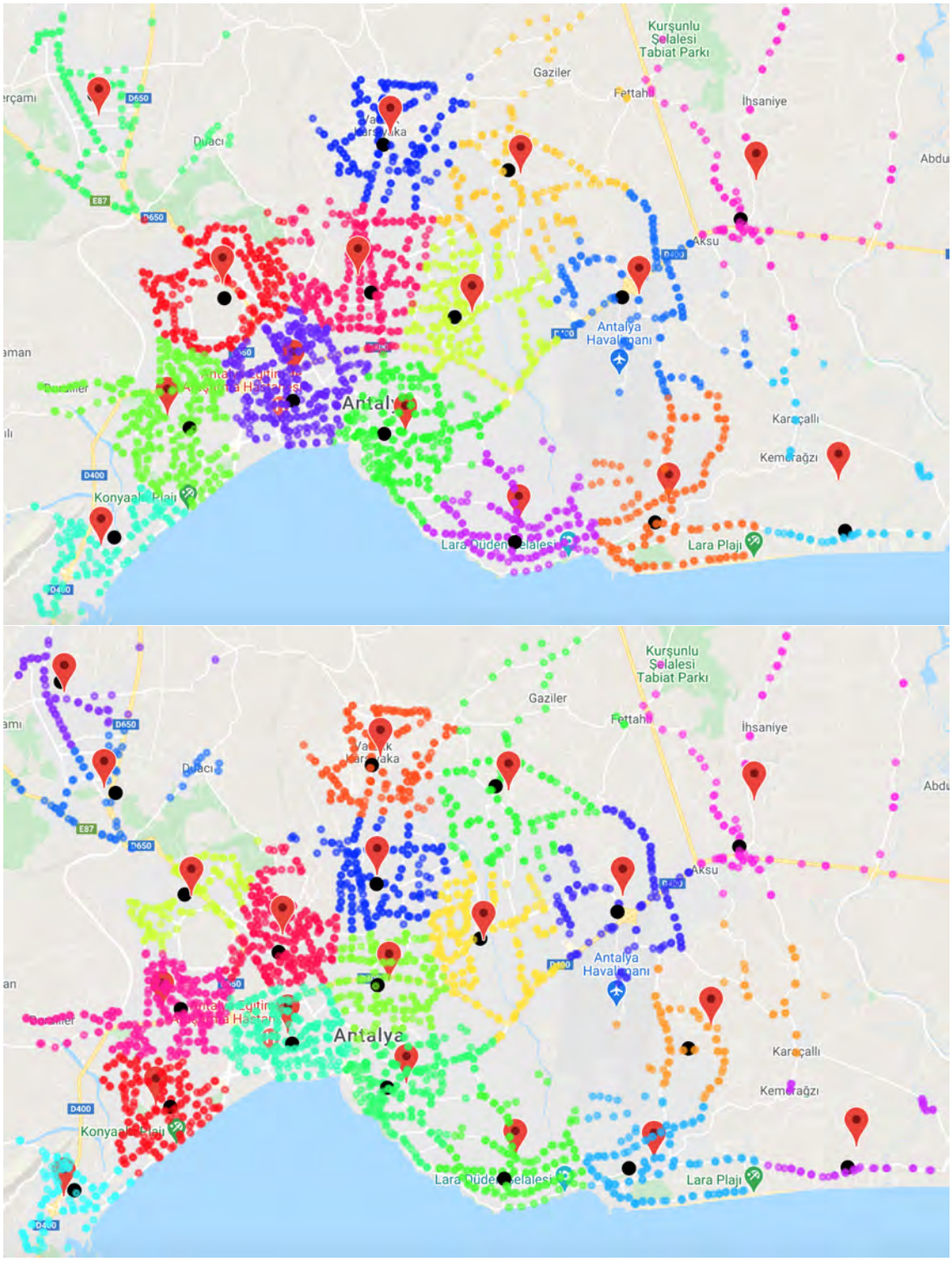

Figure 4.26. KMeans with $k = 15$ then $k = 20$

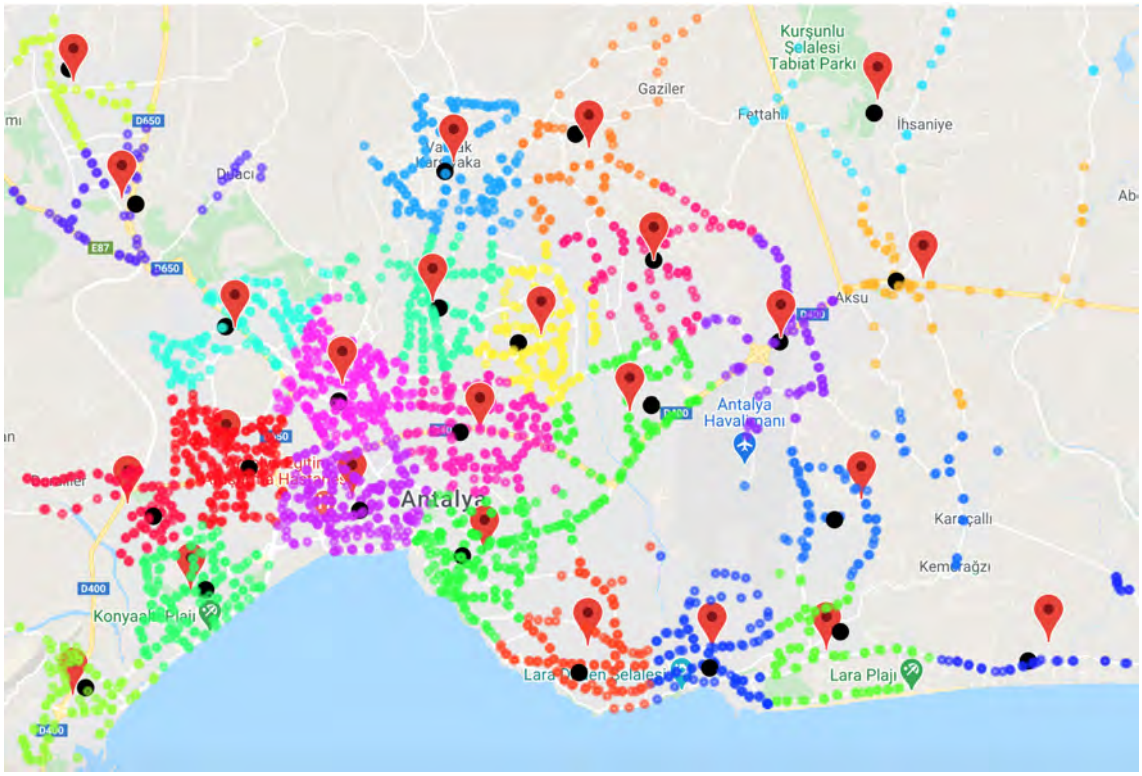

Figure 4.27. KMeans with $k = 25$

It is suggested to take the beginning of the where the average of the convergence of the center of gravity and the spatial center distance made to choose the correct k value between the clusters. So the Figure 4.28 is created and it is suggested to be called as dc-elbow (double center elbow) method.

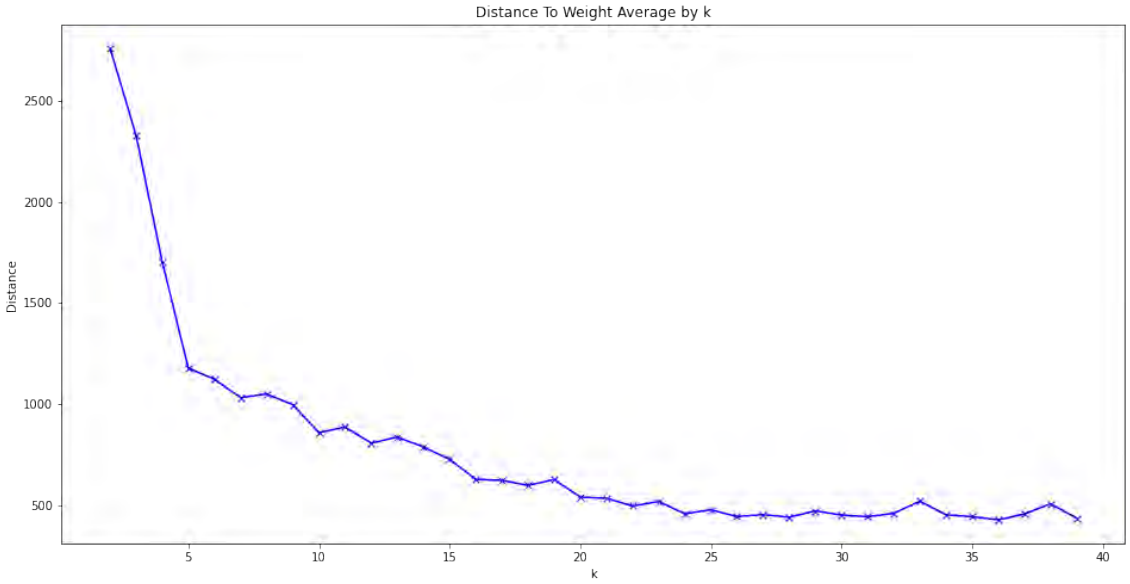

Figure 4.28. DC-Elbow method for best k selection

It is seen on the graph that after the value of 24, there is a flatness and upward fluctuations. This reduces the necessity of dividing into more clusters. For this reason, $k = 24$ is chosen. The result of the clustering created with this selected value is as reflected in Figure 4.29.

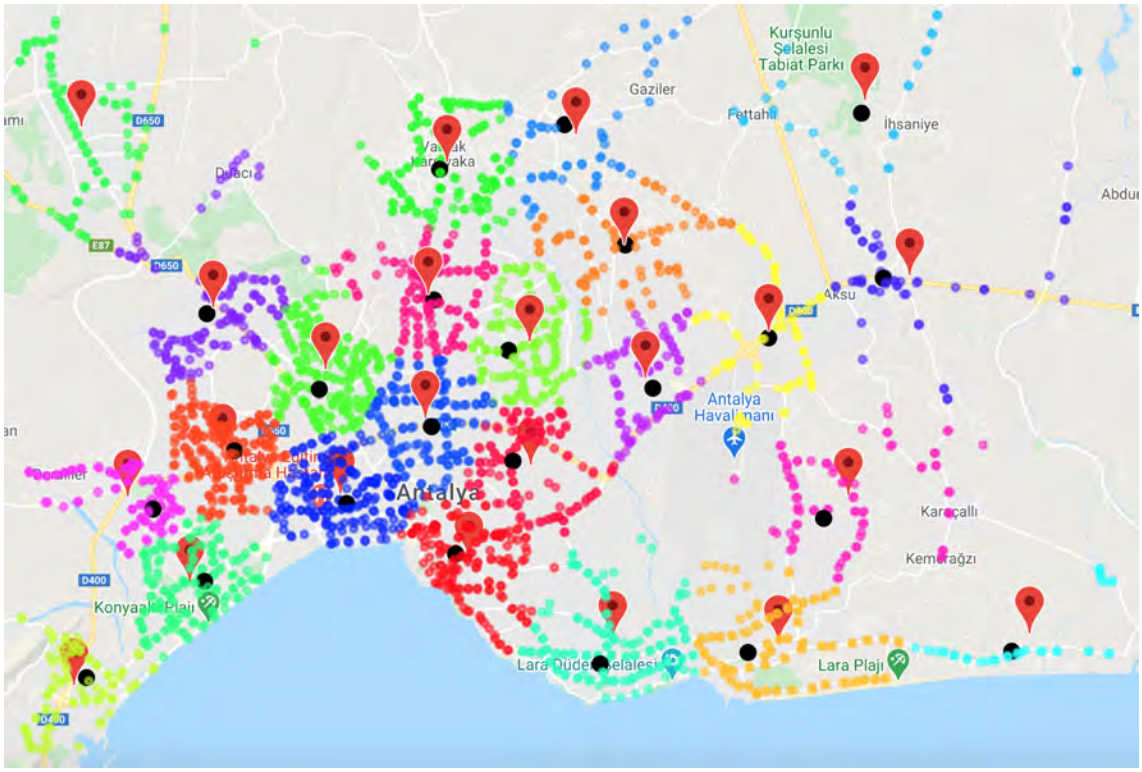

Figure 4.29. KMeans with $k = 24$

4.3. Pathfinding

24 nodes created. The sorted distance graph of these created nodes is shown on Figure 4.30.

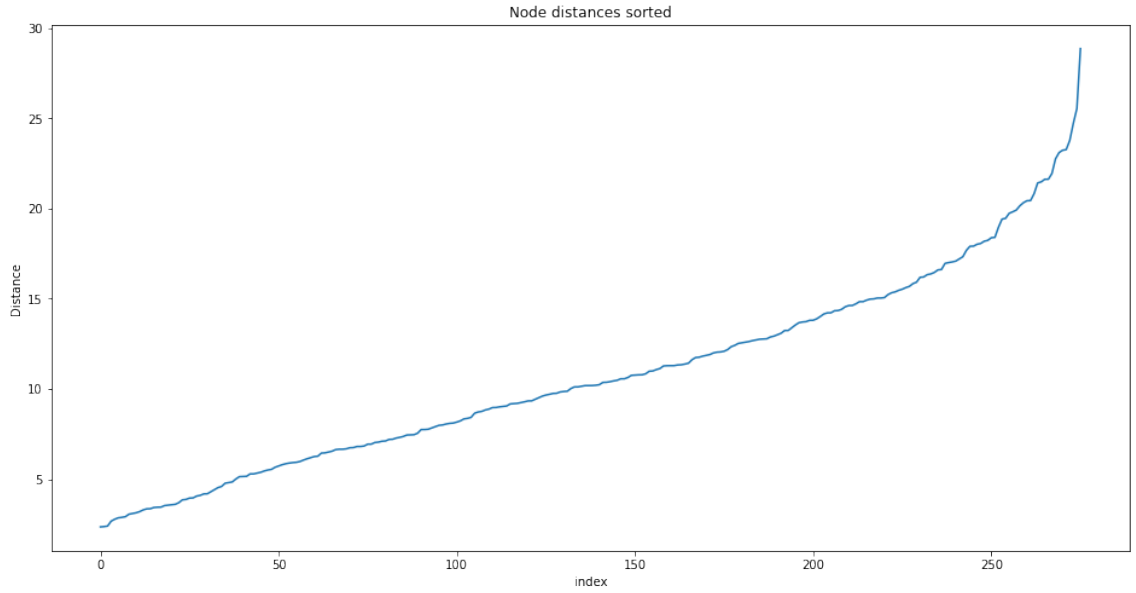

Figure 4.30. Node distances

It has been calculated that the average value is 10.806825560669383. When the average value is taken as the threshold for the distance value of the edge, there are 153 edges.

A few attempts have been made to find the route between 2 clusters on the generated graph. When making the transition between clusters, it has been given in which clusters the passengers will be taken.

Traveling scenario from C cluster to J cluster (Figure 4.31):

Figure 4.31. C $\Longrightarrow$ J

1. C (36.955890508516795, 30.769236424505227) Boarding count: 4497
2. Q (36.89409071069839, 30.685732380557134) Boarding count: 278313
3. B (36.90677908641981, 30.652231736715382) Boarding count: 57017
4. J (36.99099294120758, 30.606066983568223) Boarding count: 6244

Journey scenario from O to U (Figure 4.32):

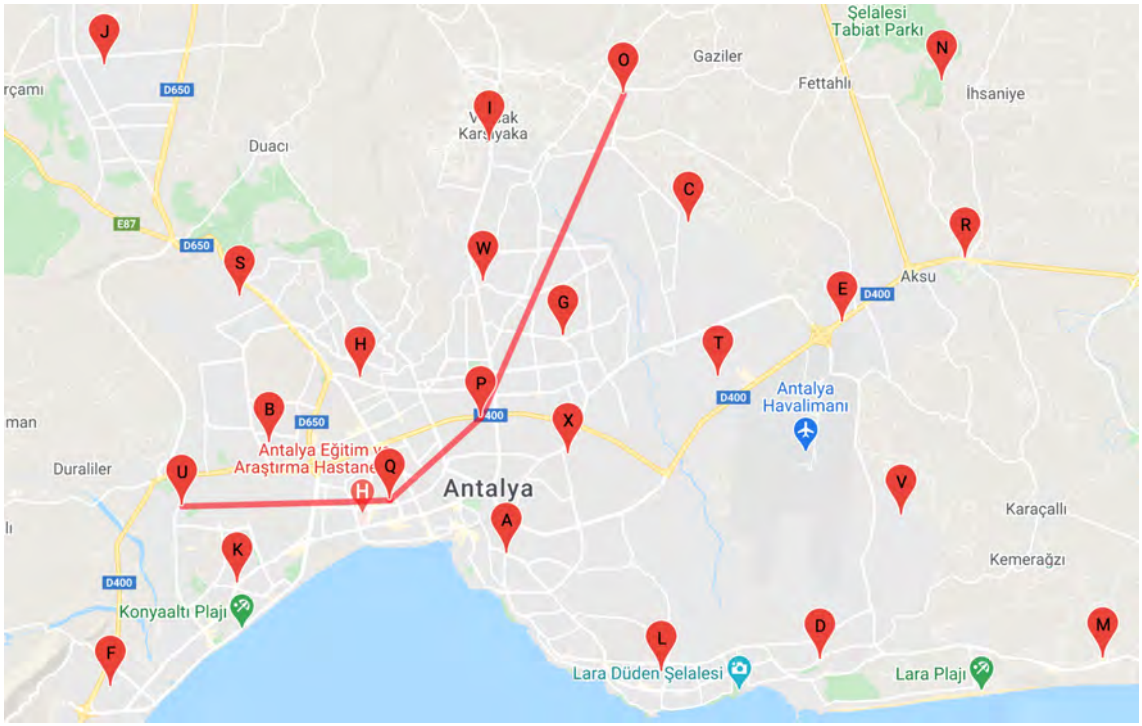

Figure 4.32. O $\Longrightarrow$ U

1. O (36.98474003594996, 30.751104672219263) Boarding count: 3758
2. P (36.91258544977909, 30.711238330218517) Boarding count: 99128
3. Q (36.89409071069839, 30.685732380557134) Boarding count: 278313
4. U (36.89272092439456, 30.6278314798156) Boarding count: 18004

Journey scenario from U to O (Figure 4.33):

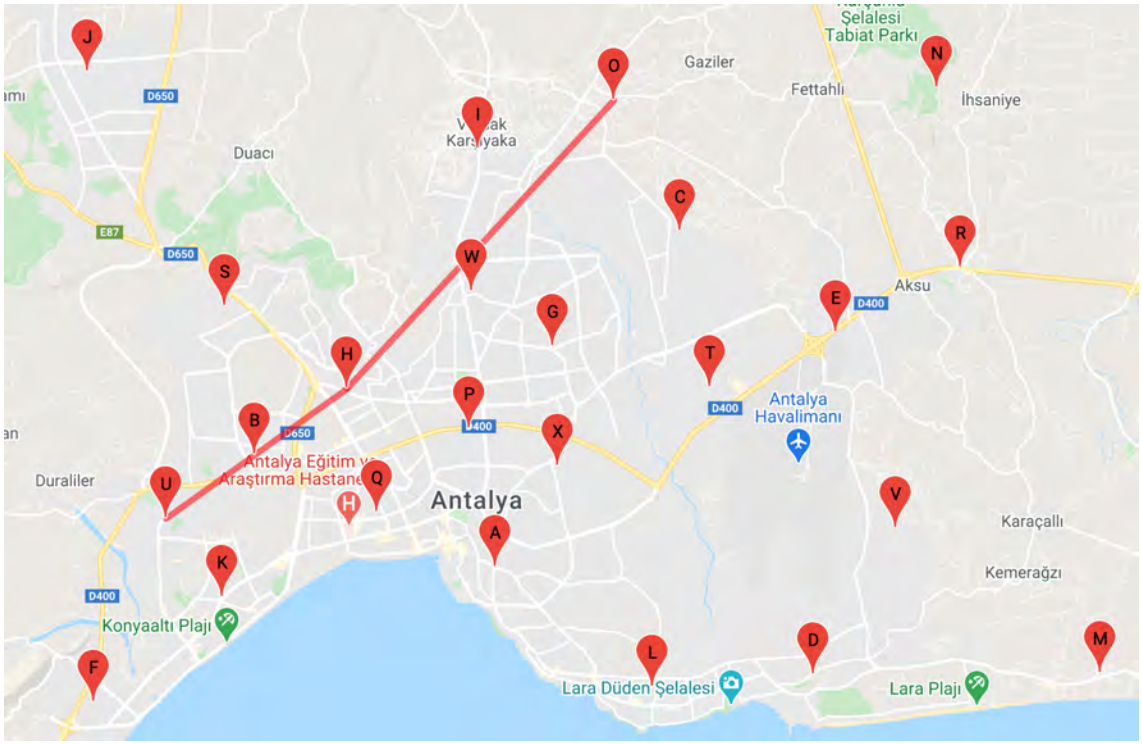

Figure 4.33. U $\Longrightarrow$ O

1. U (36.89272092439456, 30.6278314798156) Boarding count: 18004
2. H (36.92131729985722, 30.677699314666572) Boarding count: 55330
3. O (36.98474003594996, 30.751104672219263) Boarding count: 3758

Journey scenario from P to A (Figure 4.34):

Figure 4.34. P $\Longrightarrow$ A

1. P (36.91258544977909, 30.711238330218517) Boarding count: 99128
2. Q (36.89409071069839, 30.685732380557134) Boarding count: 278313
3. A (36.88219738701961, 30.71851876606596) Boarding count: 148547

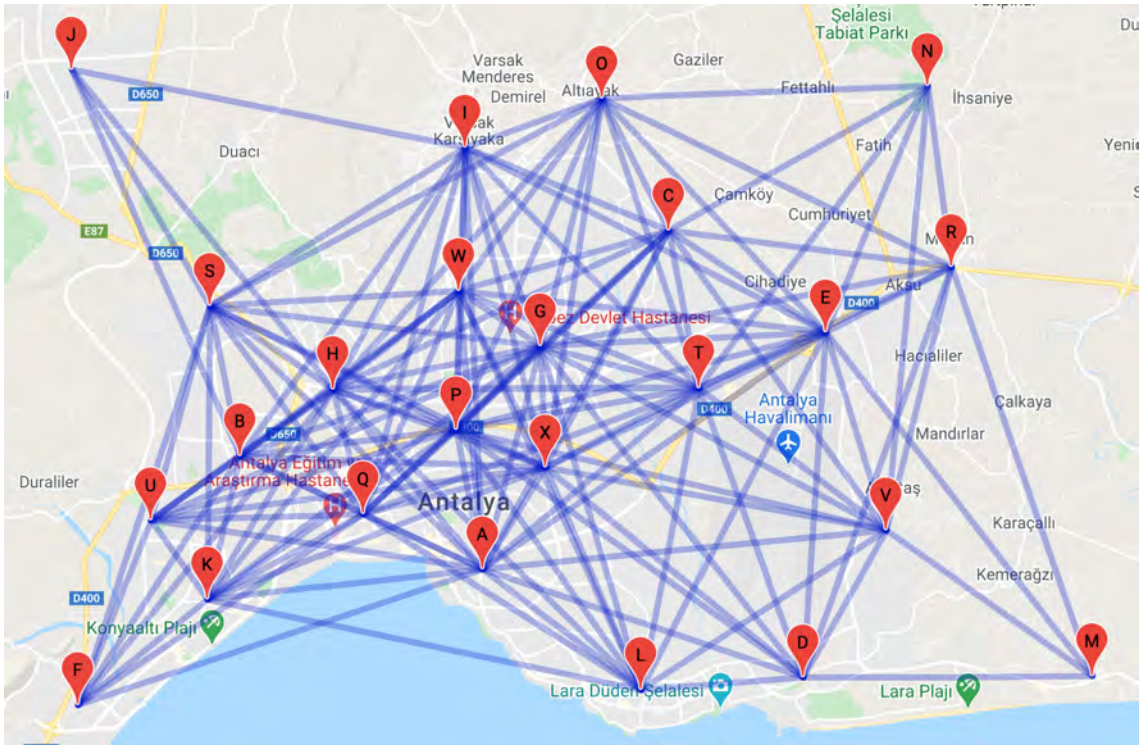

Figure 4.35. Preferred direct transit edges

Figure 4.36. Utilized routes for bus transportation

4.4. Genetic Algorithm

Although the results obtained with Uniform Cost Search show where to visit for the most efficient transition between 2 nodes, it does not offer a holistic solution. While it seems like an optimizing approach, going through a crowded intermediate node on the way from one node to another, this optimization is only route subjective. There is the fact that switching between other points also tends to go through the same crowded node. This is an indication that while a route is found, other routes do not affect the calculation. Solving the problem in question and approaching it in a holistic way increases the complexity of the problem. For this reason, genetic algorithm has been used. After 75 generations max score of the run reached and no further improvement observed on next generations until patience run out. The solution route scores ranged from 0.16 to 15.22. Routes with low score are eliminated until the point that full node coverage rule not be broken. Looking at the results obtained, it is directly seen that a more holistic approach has an effect on routes.

On the Figure 4.36, there is a skeleton created by directly connecting the edges in question, but the route assignments were not made here.The combination of the routes obtained by using the genetic algorithm is also seen on the Figure 4.37. It appears that some transitions are no longer used. Differently, there are only route definitions other than the general skeleton. Routes are drawn in different colors in order to be distinguished, but because of their transparency, the colors of the routes using common roads are mixed and because there are many routes, the routes cannot be clearly distinguished from this view.

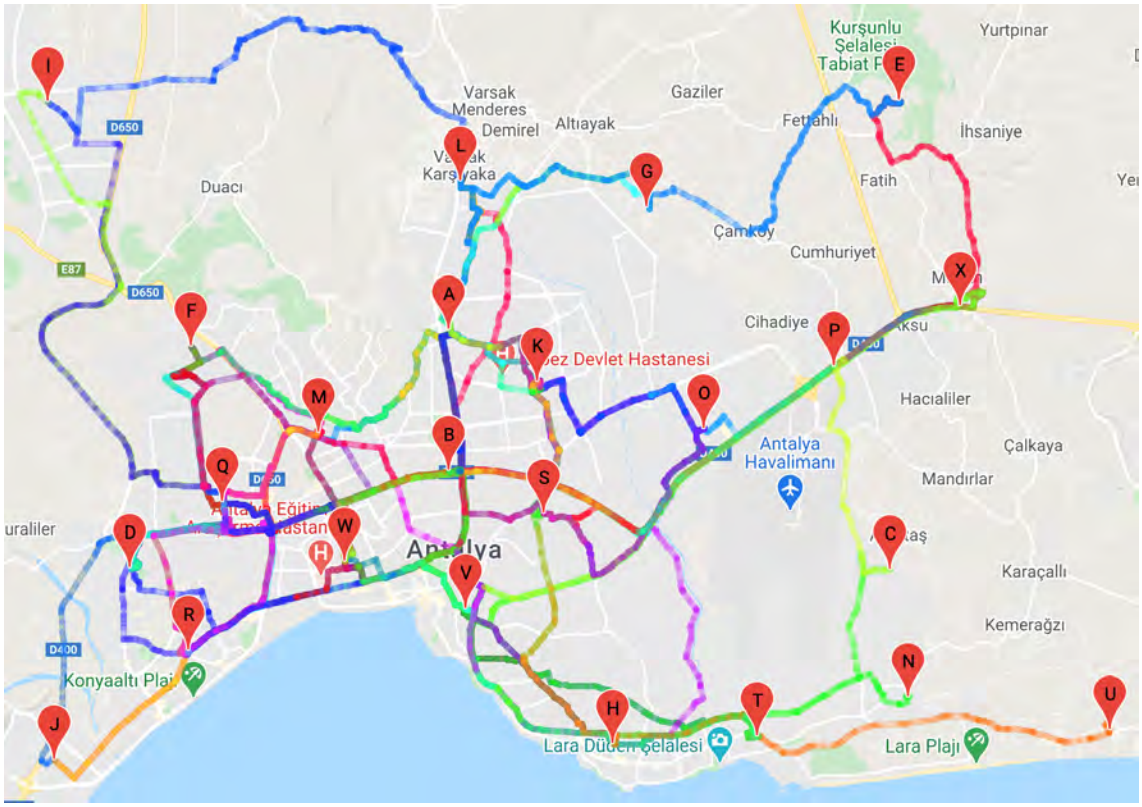

Figure 4.37. Genetic algorithm routes

Edge	Use Count
WQ	12
QB	8
BW	6
QD	5
BQ	5

Top 5 most used edges in routes in the solution listed as seen in the table. Those edges connecting high potential nodes which leads to conclusion of consistency of algorithm output. In this solution, total number of unique edges count is 105 while their use count summation equals to 200. The average edge reuse amount is simply 1.90 from division of 200 to 105.

Table 4.4. Adjacency matrix without $score > 2$ filter

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
A	0	1	0	1	0	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0	0	1	1	0
B	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
D	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0
E	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
F	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
G	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
H	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	0
I	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0
K	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1
L	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
M	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1	0
N	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	1
P	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1
Q	1	1	0	1	0	1	0	0	1	1	1	0	1	0	0	0	0	1	1	0	0	0	1	0
R	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	1	0
S	0	1	0	0	0	0	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	1	1	1
T	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
U	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	1	1	0	0	1	0
W	1	1	0	0	0	0	0	1	0	1	0	1	1	0	1	0	1	1	1	1	0	1	0	0
X	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0

Table 4.5. Adjacency matrix with $score > 2$ filter

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
A	0	1	0	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0	0	0	0	1	0	0
B	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	1	1	1	0	0	1	1	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
D	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
E	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
G	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
H	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	0
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0
K	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1
L	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0
N	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1
P	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1
Q	1	1	0	1	0	1	0	0	1	1	1	0	1	0	0	0	0	1	0	0	0	0	1	0
R	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0
S	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1
T	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
U	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	1	1	0	0	1	0
W	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	1	0	0
X	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0

Table 4.6. First 25 Genetic algorithm generated routes

	Route	Boarding	Length (km)	Duration (min)	Score	Stops
0	HVWQF	569984	20.18	46.67	15.23	817
1	XPBWJ	417525	28.70	41.33	12.05	574
2	XPBWQ	431656	24.48	33.27	11.62	677
3	LABRJ	301584	21.84	38.45	9.42	718
4	UHVBQ	402299	29.84	56.13	8.77	671
5	HSBAL	258484	19.12	41.65	8.70	716
6	XKABW	434242	25.75	39.55	8.19	706
7	RQMAL	272013	19.30	40.00	7.42	907
8	SWQRJ	464956	22.85	40.88	6.88	744
9	VWBPX	536770	25.58	43.63	6.69	657
10	CPVWQ	493081	26.72	44.53	6.53	722
11	GAKBW	434605	21.53	37.08	6.13	758
12	VBWQI	587251	29.79	50.47	6.11	799
13	QBVPX	330960	27.97	43.85	6.06	663
14	FQBWH	506885	26.73	50.95	5.79	794
15	CTHWQ	433094	27.14	54.32	5.56	675
16	TVBQM	411140	24.12	49.17	5.31	852
17	NHSWQ	415364	28.41	57.63	5.18	687
18	LAKSV	272938	17.99	35.75	5.07	782
19	RQBVO	415152	25.45	45.00	5.06	758
20	FABHT	268405	26.25	60.17	5.03	657
21	PMWQD	400214	29.07	50.67	4.89	803
22	PABVR	398928	29.61	51.17	4.52	711
23	QAVHT	372062	27.24	57.15	4.13	731
24	XPBDQ	186833	27.12	37.13	3.64	586

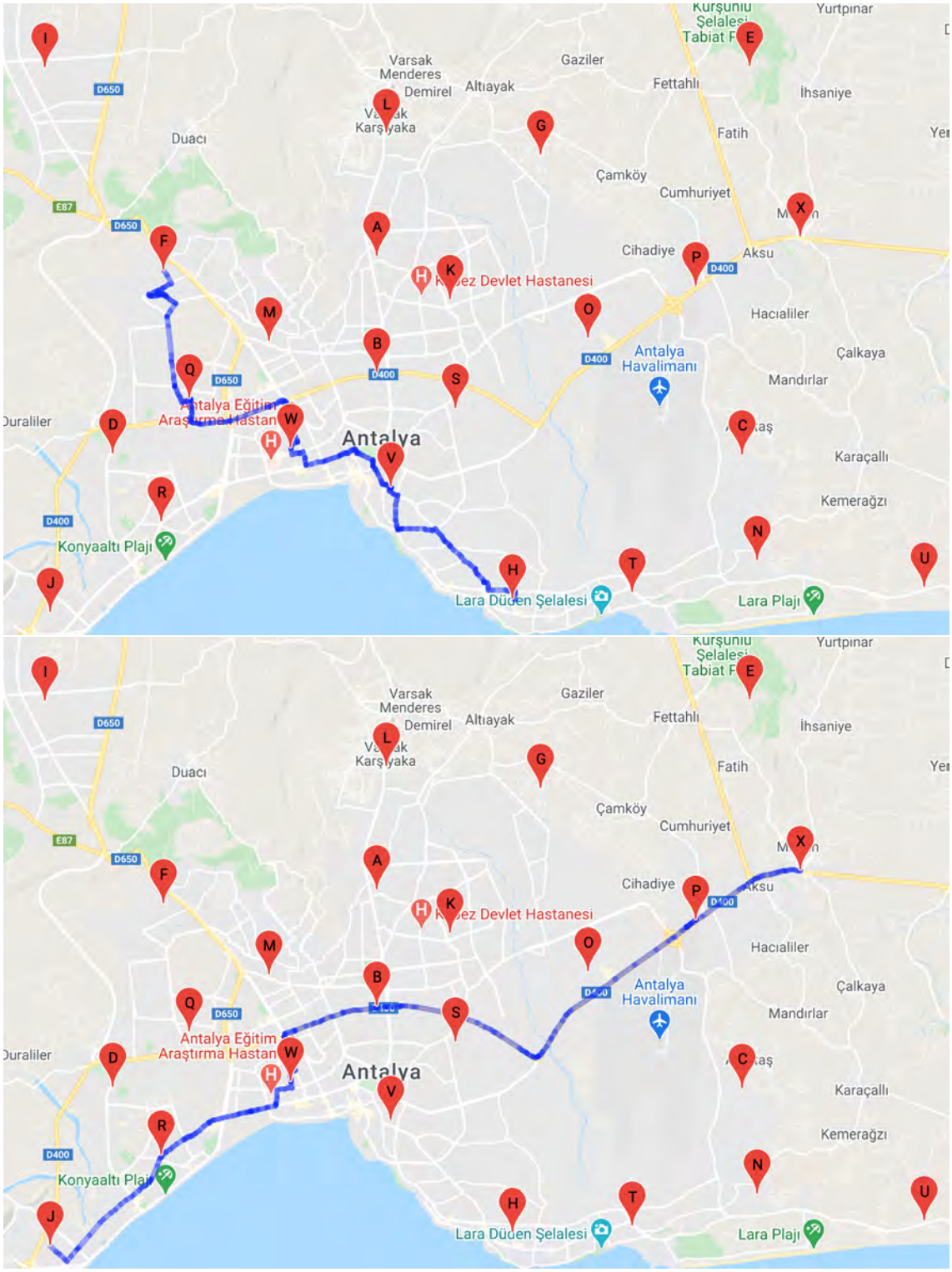

Figure 4.38. HVWQF and XPBWJ routes

4.5. Comparison with Currently Active System

Route repetition ratio for currently active system has found as 0.216 from the $R_r = \frac{R_t - R_u}{R_t}$ formula where R_r is route repetition ratio, R_t is total route count and R_u is unique route count.

Some of routes repeated several times and shown in the Table 4.7 and 4.8.

Table 4.7. Routes with Repetition

Route	Repetition	Route	Repetition
MWQDJ	4	MQWVSOPC	2
MFI	4	JRWVHT	2
MQWRJ	3	FMW	2
FQWVHT	3	UXPOSBWQM	2
THVWRJ	2	VSBWQMFI	2
AKBVSHTNC	2	GLABVHT	2
MQFI	2	MWRJ	2
MWVHT	2	JRM	2
JRWQFI	2	GLABVWQ	2
EGLABV	2	EUM	2
GLAKSBV	2		

Table 4.8. Line Codes of Specific Routes

MWQDJ	MFI	MQWRJ	FQWVHT	THVWRJ
500	DC27	515	LF09	GCK76
521	DC27A	515A	LF10	GCK76A
521A	DM85A	AK03	LF10G	
521B	EA01			

Directness comparison between currently active and proposed system is shown in Table 4.9

Table 4.9. Comparison of Descriptive Directness Analysis

	Active System Directness	Proposed System Directness
count	125	50
mean	0.459282	0.505524
std	0.153265	0.170214
min	0.107524	0.090768
25%	0.354805	0.396579
50%	0.438706	0.525431
75%	0.564154	0.606874
max	0.784852	0.828750

Directness improvement has found as 10% from the $I = \frac{D_1 - D_0}{D_0}$ formula where I is improvement, D_1 is directness value of proposed system and D_0 is directness value of currently active system.

Table 4.10. Overlapping Comparison

	Overlap Count	Total Edge Count	Overlap Ratio
Active Solution	515	611	0.842881
Proposed Solution	95	200	0.475000

The comparison result shows us that the currently active system repeats similar routes and not efficient in that way. In Figure 4.39 we can see the drastic efficiency drop of active system compared to suggested when edge repeat penalty increases.

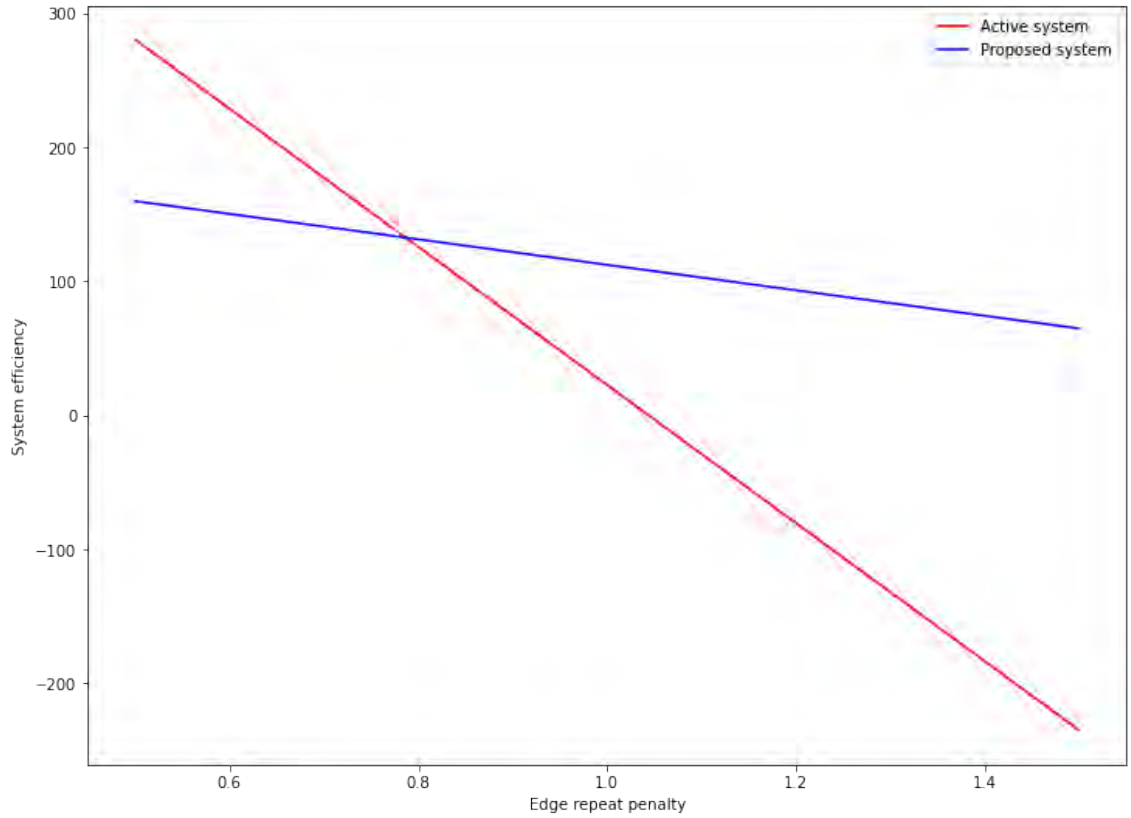

Figure 4.39. Edge Repeat Penalty - Efficiency Relation

Transfer counts measured for both systems by counting every node couples that has not included any route in system. Analysis results shows that the transfer amount in active system is 42 while it is 112 in proposed system. Which is expected since the fitness function doesn't have the knowledge of importance of transfer. Edges that need to transfer to travel shown in Table 4.11.

Table 4.11. Transfer Necessity

	from	to
0	A	C, N, U
1	B	C, N
2	C	D, E, F, G, I, J, K, L, M, N, O, R, S, U, X
3	D	E, G, J, L, N, T, U, V
4	E	F, H, I, J, K, N, O, Q, R, T, U, V, W
5	F	G, I, J, K, L, N, P, U
6	G	H, I, J, N, O, P, S, T, U, V, X
7	H	I, J, P, R, X
8	I	J, M, N, P, S, T, U, X
9	J	K, M, N, O, T, U
10	K	N, P, T, U
11	L	N, P, T, U, X
12	M	N, U, X
13	N	O, P, R, T, U, V, X
14	O	P, T, U
15	P	T, U
16	R	T, U, X
17	S	T, U
18	T	U, X
19	U	W, X

Those from-to edges in same row in Table 4.11, connected to each other in fivefold permutations (triple or quadruple if row doesn't have minimum 5 nodes) to decrease transfer count in proposed system. 915726 candidate routes generated. Route efficiency values calculated for every generated route and ordered by efficiency, descending. Those routes iteratively evaluated. All covered nodes kept in a hashset during iteration. If the next route does not have enough nodes that are not covered, it has been discarded.

Accepting routes that doesn't have any node covered results in 14 new routes: COSRJ, KVWRJ, UTHVW, OVWRJ, POSVJ, JRDML, NTHVW, EOVWR, XPOVR, JRMLG,

HVWFI, DVTNU, THVWR, THVWF. When those routes added to proposed system total route count increases from 50 to 64 and transfer count decreases from 112 to 83. Accepting routes that has maximum 1 covered node 39 new routes: QKTNU, CGLMF, JQKTN, THVSG, XOKSR, POSVJ, JRDML, NTHVW, UNHKQ, QOTNU, DVTNU, COSRJ, LVTNU, IFQRJ, EOVWR, XPOVR, UNOKM, JRMLG, JRSXU, JDMLE, HSOPX, UTHVW, OVWRJ, XOKSD, UNPGL, MKTNU, EGLFD, HVWFI, THVWR, THVWF, UNTHS, IFQOT, KVWRJ, JDMLG, XSKMR, JRDMI, XOKMF, NOKMF, MSTNU. When those routes added to proposed system total route count increases from 50 to 89 and transfer count decreases from 112 to 36. Adding the first 25 of 39 increases from 50 to 75 and transfer count decreases from 112 to 46. This values can be seen in Table 4.12.

Table 4.12. Transfer and Length Comparison

Proposed System Transfer Count	Proposed System Route Count	Proposed System Total Length
112	50 +0	1296 km
83	50 +14	1651 km
46	50 +25	2047 km
36	50 +39	2440 km

* Transfer count of active system is 42. Total length of active system is 3428 km.

5. CONCLUSION

This study proposes a method to optimize the type of public transport by balancing maximum passenger transport with minimum distance travel with shared edge restriction between routes. An efficiency function is defined so that the number of passengers taken in heuristic sense is rewarded while the distance covered is penalized. Route directness calculated and used as a parameter in efficiency function. This efficiency function used in genetic algorithm's fitness function to obtain good results. Since transfer amount is not considered as a negative effect, it's not included in efficiency function. New system with new routes proposed with genetic algorithm. Active system entities migrated to new system domain to compare them. Comparison made using different aspects like route amount, transfer amount and efficiency. Transfer count tweaked by adding new efficient routes that eliminates necessary transfers. Additive side routes can be appended to evolutionary solution by hand too. To move forward for better genetic algorithm solutions one must parameterize efficiency function coefficients and fitness function constants and then run hyperparameter optimization to tune the algorithm. Adaptive mutation mechanics may help to escape from local maxima which leads algorithm to stop after the patience limit excess.

As we have seen in the experiments, the uniform cost search system pursues to get passengers from the clusters that contain a large number of people. This will contribute to distributing the density across the routes. And it can be used in dynamic routing algorithms to support proposed static system.

When a clear result cannot be obtained from the elbow method during clustering with KMeans, a method that will work on spatial data is proposed in order to find the k value, ie the maximum number of significant clusters. In this method, the average distance between the cluster center of all clusters and the center of gravity of the cluster is calculated. For a set of k values, this distance is calculated and the sorted chart is drawn, and the k value at the beginning of the line where the value converge is taken. This can represent the maximum number of significant classes.

As can be seen from the heatmap data, it has been observed that the DBSCAN algorithm does not exhibit the appropriate performance due to the fact that the distribution of

the ride over the city is far from homogeneity, and the noise approach is also included in the clustering process. To prevent this situation, a normalization can be done on the data, but this will cause loss of important relations between the clusters and the problem to be transferred to a different domain.

It has been shown how to obtain the missing data required for the research, how to manipulate and maintain the data in accordance with the 4th normalized form.

6. REFERENCES

Baaj, M. Hadi and Hani S. Mahmassani (1991). "An AI-based approach for transit route system planning and design". In: *Journal of Advanced Transportation* 25.2, pp. 187–209. DOI: https://doi.org/10.1002/atr.5670250205. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/atr.5670250205. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/atr.5670250205.

Boyce, David E (1984). "Urban transportation network-equilibrium and design models: recent achievements and future prospects". In: *Environment and Planning A* 16.11, pp. 1445–1474.

Buchroithner, Manfred F and René Pfahlbusch (2017). "Geodetic grids in authoritative maps–new findings about the origin of the UTM Grid". In: *Cartography and Geographic Information Science* 44.3, pp. 186–200.

Bundy, Alan and Lincoln Wallen (1984). "Breadth-first search". In: *Catalogue of artificial intelligence tools*. Springer, pp. 13–13.

Canca, David et al. (2017). "An adaptive neighborhood search metaheuristic for the integrated railway rapid transit network design and line planning problem". In: *Computers & Operations Research* 78, pp. 1–14.

Ceder, Avishai and Nigel HM Wilson (1986). "Bus network design". In: *Transportation Research Part B: Methodological* 20.4, pp. 331–344.

Claessens, MT, Nico M van Dijk, and Peter J Zwaneveld (1998). "Cost optimal allocation of rail passenger lines". In: *European Journal of Operational Research* 110.3, pp. 474–489.

Constantin, Isabelle and Michael Florian (1995). "Optimizing frequencies in a transit network: a nonlinear bi-level programming approach". In: *International Transactions in Operational Research* 2.2, pp. 149–164.

Crainic, Teodor Gabriel (Apr. 2000). "Service Network Design in Freight Transportation". In: *European Journal of Operational Research* 122, pp. 272–288. DOI: 10.1016/S0377-2217(99)00233-7.

Deng, Yiling and Yadan Yan (2019). “Evaluating Route and Frequency Design of Bus Lines Based on Data Envelopment Analysis with Network Epsilon-Based Measures”. In: *Journal of Advanced Transportation* 2019.

Dong, Jingxin and Jianping Wu (2003). “Urban traffic networks equilibrium status recognition with neural network”. In: *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*. Vol. 2. IEEE, pp. 1049–1053.

Fan, Wei and Randy B Machemehl (2006). “Optimal transit route network design problem with variable transit demand: genetic algorithm approach”. In: *Journal of transportation engineering* 132.1, pp. 40–51.

Farahani, Reza Zanjirani et al. (2013). “A review of urban transportation network design problems”. In: *European Journal of Operational Research* 229.2, pp. 281–302.

Friesz, Terry L et al. (1992). “A simulated annealing approach to the network design problem with variational inequality constraints”. In: *Transportation science* 26.1, pp. 18–26.

Furth, Peter G and Nigel HM Wilson (1981). “Setting frequencies on bus routes: Theory and practice”. In: *Transportation Research Record* 818.1981, pp. 1–7.

Goossens, Jan-Willem, Stan van Hoesel, and Leo Kroon (2006). “On solving multi-type railway line planning problems”. In: *European Journal of Operational Research* 168.2, pp. 403–424.

Guihaire, Valérie and Jin-Kao Hao (2008). “Transit network design and scheduling: A global review”. In: *Transportation Research Part A: Policy and Practice* 42.10, pp. 1251–1273.

Hasselström, D (1979). “A method for optimization of urban bus route networks”. In: *Volvo Transportation Systems, Gothenburg*.

Htwe, Khin Mar and Thinn Naing (2009). “Analysis of Searching Deterministic Shortest Path Algorithms (Depth First Search, Breadth First Search, Uniform Cost Search)”. In: Third Local Conference on Parallel and Soft Computing.

Hu, Jianming et al. (2005). “Optimal design for urban mass transit network based on evolutionary algorithms”. In: *International Conference on Natural Computation*. Springer, pp. 1089–1100.

Jha, Shashi Bhushan, Jitendra Kumar Jha, and Manoj Kumar Tiwari (2019). "A multi-objective meta-heuristic approach for transit network design and frequency setting problem in a bus transit system". In: *Computers & Industrial Engineering* 130, pp. 166–186.

Karney, Charles F. F. (2013). "Algorithms for geodesics". In: *Journal of Geodesy* 87. DOI: 10.1007/s00190-012-0578-z. URL: https://doi.org/10.1007/s00190-012-0578-z.

Kepaptsoglou, Konstantinos and Matthew Karlaftis (2009). "Transit route network design problem". In: *Journal of transportation engineering* 135.8, pp. 491–505.

Korf, Richard E (1990). "Depth-limited search for real-time problem solving". In: *Real-Time Systems* 2.1-2, pp. 7–24.

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, pp. 281–297. URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

Magnanti, T. L. and R. T. Wong (1984). "Network Design and Transportation Planning: Models and Algorithms". In: *Transportation Science* 18.1, pp. 1–55. DOI: 10.1287/trsc.18.1.1. eprint: https://doi.org/10.1287/trsc.18.1.1. URL: https://doi.org/10.1287/trsc.18.1.1.

Magnanti, Thomas L and Richard T Wong (1984). "Network design and transportation planning: Models and algorithms". In: *Transportation science* 18.1, pp. 1–55.

Maher, Margaret M (2013). *Lining up data in ArcGIS: a guide to map projections*. Esri Press Redlands, CA.

Mandl, Christoph E (1980a). "Evaluation and optimization of urban public transportation networks". In: *European Journal of Operational Research* 5.6, pp. 396–404.

— (1980b). "Evaluation and optimization of urban public transportation networks". In: *European Journal of Operational Research* 5.6, pp. 396–404. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(80)90126-5. URL: https://www.sciencedirect.com/science/article/pii/0377221780901265.

Marín, Ángel and Ricardo García-Rodenas (2009). "Location of infrastructure in urban railway networks". In: *Computers & Operations Research* 36.5, pp. 1461–1477.

Ngamchai, Somnuk and David J Lovell (2003). "Optimal time transfer in bus transit route network design using a genetic algorithm". In: *Journal of Transportation Engineering* 129.5, pp. 510–521.

Nguyen, Van Son, Quang Dung Pham, and Minh Hoang Ha (2016). "Improving the Connectivity of a Bus System: A Case Study of Ho Chi Minh City". In: *Proceedings of the Seventh Symposium on Information and Communication Technology*. SoICT '16. Ho Chi Minh City, Vietnam: Association for Computing Machinery, pp. 73–78. ISBN: 9781450348157. DOI: 10.1145/3011077.3011094. URL: https://doi.org/10.1145/3011077.3011094.

Ortega, Francisco and Laurence A. Wolsey (2003). "A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem". In: *Networks* 41.3, pp. 143–158. DOI: https://doi.org/10.1002/net.10068. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.10068. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.10068.

Owais, Mahmoud and Mostafa K Osman (2018). "Complete hierarchical multi-objective genetic algorithm for transit network design problem". In: *Expert Systems with Applications* 114, pp. 143–154.

Pacheco, Joaquín et al. (2009). "A tabu search approach to an urban transport problem in northern Spain". In: *Computers & Operations Research* 36.3, pp. 967–979.

Pattnaik, SB, S Mohan, and VM Tom (1998). "Urban bus transit route network design using genetic algorithm". In: *Journal of transportation engineering* 124.4, pp. 368–375.

Pternea, Moschoula, Konstantinos Kepaptsoglou, and Matthew G Karlaftis (2015). "Sustainable urban transit network design". In: *Transportation Research Part A: Policy and Practice* 77, pp. 276–291.

Ren, Yibin et al. (Nov. 2016). "Extracting potential bus lines of Customized City Bus Service based on public transport big data". In: *IOP Conference Series: Earth and*

Environmental Science 46, p. 012017. DOI: 10.1088/1755-1315/46/1/012017. URL: https://doi.org/10.1088/1755-1315/46/1/012017.

Saeidi, Shahram (2014). "A genetic algorithm for route optimization in public transportation problem". In: *2014 International Conference on Business and Information*. Vol. 7.

Sáez Aguado, Jesús (2008). "Fixed Charge Transportation Problems: a new heuristic approach based on Lagrangean relaxation and the solving of core problems". In: *Annals of Operations Research* 172.1, pp. 6–18. DOI: 10.1007/s10479-008-0483-2. URL: https://doi.org/10.1007/s10479-008-0483-2.

Sander, Jörg et al. (1998). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications". In: *Data Mining and Knowledge Discovery* 2.2, pp. 169–194. DOI: https://doi.org/10.1023/A:1009745219419.

Schubert, Erich et al. (July 2017a). "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". In: *ACM Trans. Database Syst.* 42.3. ISSN: 0362-5915. DOI: 10.1145/3068335. URL: https://doi.org/10.1145/3068335.

— (July 2017b). "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". In: *ACM Trans. Database Syst.* 42.3. ISSN: 0362-5915. DOI: 10.1145/3068335. URL: https://doi.org/10.1145/3068335.

Silman, L.A., Z. Barzily, and U. Passy (1974). "Planning the route system for urban buses". In: *Computers & Operations Research* 1.2, pp. 201–211. ISSN: 0305-0548. DOI: https://doi.org/10.1016/0305-0548(74)90046-X. URL: www.sciencedirect.com/science/article/pii/030505487490046X.

Singh, Archana, Avantika Yadav, and Ajay Rana (2013). "K-means with Three different Distance Metrics". In: *International Journal of Computer Applications* 67.10.

Tarjan, Robert (1972). "Depth-first search and linear graph algorithms". In: *SIAM journal on computing* 1.2, pp. 146–160.

Ukkusuri, Satish V., Tom V. Mathew, and S. Travis Waller (2007). "Robust Transportation Network Design Under Demand Uncertainty". In: *Computer-Aided Civil and Infrastructure Engineering* 22.1, pp. 6–18. DOI: https://doi.org/10.1111/j.1467-8667.2006.00465.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8667.2006.00465.x. URL: https:

//onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8667.2006.00465.x.

Van Vliet, Dirck (1978). "Improved shortest path algorithms for transport networks". In: *Transportation Research* 12.1, pp. 7–20. ISSN: 0041-1647. DOI: https://doi.org/10.1016/0041-1647(78)90102-8. URL: https://www.sciencedirect.com/science/article/pii/0041164778901028.

Wang, Chao, Zhirui Ye, and Wei Wang (2020). "A multi-objective optimization and hybrid heuristic approach for urban bus route network design". In: *IEEE Access* 8, pp. 12154–12167.

Wang, Chunbao et al. (2015). "Path planning of automated guided vehicles based on improved A-Star algorithm". In: *2015 IEEE International Conference on Information and Automation*. IEEE, pp. 2071–2076.

Wren, Anthony and David O. Wren (1995). "A genetic algorithm for public transport driver scheduling". In: *Computers & Operations Research* 22.1. Genetic Algorithms, pp. 101–110. ISSN: 0305-0548. DOI: https://doi.org/10.1016/0305-0548(93)E0022-L. URL: https://www.sciencedirect.com/science/article/pii/0305054893E0022L.

Xie, Fan, Martin Müller, and Robert Holte (2014). "Jasper: the art of exploration in greedy best first search". In: *The Eighth International Planning Competition (IPC-2014)*, pp. 39–42.

Yang, Hai and Michael G H. Bell (1998). "Models and algorithms for road network design: a review and some new developments". In: *Transport Reviews* 18.3, pp. 257–278.

Yang, Jie and Yangsheng Jiang (2020). "Application of Modified NSGA-II to the Transit Network Design Problem". In: *Journal of Advanced Transportation* 2020.

Yang, Zhongzhen, Bin Yu, and Chuntian Cheng (2007). "A parallel ant colony algorithm for bus network optimization". In: *Computer-Aided Civil and Infrastructure Engineering* 22.1, pp. 44–55.

Yu, Bin, Zhongzhen Yang, and Jinbao Yao (2010). "Genetic algorithm for bus frequency optimization". In: *Journal of Transportation Engineering* 136.6, pp. 576–583.

Zhu, J. -X. et al. (2020). “A Traffic Assignment Approach for Multi-Modal Transportation Networks Considering Capacity Constraints and Route Correlations”. In: *IEEE Access* 8, pp. 158862–158874. DOI: 10.1109/ACCESS.2020.3019301.

ÖZGEÇMİŞ

BATUHAN BULUT

bthn.bulut@gmail.com

ÖĞRENİM BİLGİLERİ

Yüksek Lisans 2018-2021	Akdeniz Üniversitesi Fen Bilimleri Enstitüsü,Bilgisayar Mühendisliği Bölümü, Antalya
Lisans 2011-2015	Akdeniz Üniversitesi Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Antalya

MESLEKİ VE İDARİ GÖREVLER

Yapay Zeka Uzmanı 2020-Devam Ediyor	EyeCheckup
Yazılım Uzmanı 2020-Devam Ediyor	Ural Telekom
Yazılım Uzmanı 2018-2019	DS Bio ve Nano Teknoloji Ürün Takip Doğrulama Tic. A.Ş.
Yazılım Uzmanı 2016-2017	Dream Harvesters
Yazılım Uzmanı 2016-2016	Imaginite Entertainment Software

ESERLER

Uluslararası hakemli dergilerde yayımlanan makaleler

1- B. Bulut, V. Kalın, B. B. Güneş and R. Khazhin, "Deep Learning Approach For Detection Of Retinal Abnormalities Based On Color Fundus Images," 2020 Innovations in

Intelligent Systems and Applications Conference (ASYU), 2020, pp. 1-6, doi: 10.1109/ASYU50717.2020.9259870.

Ulusal bilimsel toplantılarda sunulan ve bildiri kitaplarında basılan bildiriler

1- ALBAYRAK Y, BULUT B, ASİLKAN Ö. Mobil cihazlarda rtmfp protokolü ile P2P görüntü iletimi. XVI. Akademik Bilişim-AB 2014. 2014.

2- ALBAYRAK Y, Seçkin ŞE, Bulut B. Gömülü Web Sunucusuyla Internet Tabanlı Denetim Uygulaması.