Universidade de Aveiro
2015

Departamento de Electrónica, Telecomunicações e Informática

Diogo Marques Martins

Técnicas de agrupamento de trajetórias com aplicação à recomendação de percursos
Trajectory clustering techniques with application to route recommendation

Universidade de Aveiro
2015

Departamento de Electrónica, Telecomunicações
e Informática

Diogo Marques Martins

Técnicas de agrupamento de trajetórias com aplicação à recomendação de percursos
Trajectory clustering techniques with application to route recommendation

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática (M.I.E.C.T.), realizada sob a orientação científica do Prof. Doutor José Moreira, Professor do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e Talel Abdelssalem, Professor Associado da Télécom ParisTech, França.

o júri

Presidente/president	Profª. Drª. Pétia Georgieva Professora Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro
Vogais/Examiners Committee	Prof. Dr. Viriato Marques Professor Coordenador do Instituto Superior de Engenharia de Coimbra
	Prof. Dr. José Manuel Matos Moreira Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

É com imenso apreço que aproveito esta oportunidade para agradecer o apoio a todos os que estiveram presentes ao longo deste meu percurso académico.

Um especial agradecimento ao meu orientador Professor Doutor José Manuel Matos Moreira por toda a disponibilidade prestada ao longo de todas as fases deste trabalho. Desejo tambem agradecer à Télécom ParisTech pelo acolhimento e orientação do Professor Talel Abdessalem.

Um muito especial agradecimento à minha namorada Alexandra por estar sempre presente, por toda a força e apoio prestado. Um agradecimento, também ele especial, aos meus pais que de uma forma incondicional prestaram o devido apoio para a obtenção do meu sucesso académico.

acknowledgments

It is with great appreciation that I take this opportunity to thank the support of all who were present in my academic journey.

A special thanks to my mentor Professor José Manuel Matos Moreira for the availability provided during all the phases of this work. I wish also to thank Télécom ParisTech in particular Professor Talel Abdessalem for the reception and orientation.

A very special thanks to my girlfriend Alexandra to be always present and for the strength and support provided. Finally, I want to thank my parents who unconditionally provided the necessary support to achieve my academic success.

Palavras-chave

Objetos móveis, Dados espácio-temporais, Agrupamento de trajetórias, Sistemas de recomendação

Resumo

O uso generalizado de dispositivos capazes de obter e transmitir dados sobre a localização de objetos ao longo do tempo tem permitido recolher grandes volumes de dados espácio-temporais. Por isso, tem-se assistido a uma procura crescente de técnicas e ferramentas para a análise de grandes volumes de dados espácio-temporais com o intuito de disponibilizar uma gama variada de serviços baseados na localização.

Esta dissertação centra-se no desenvolvimento de um sistema para recomendaSr trajetos com base em dados históricos sobre a localização de objetos móveis ao longo do tempo. O principal problema estudado neste trabalho consiste no agrupamento de trajetórias e na extração de informação a partir dos grupos de trajetórias. Este estudo, não se restringe a dados provenientes apenas de veículos, podendo ser aplicado a outros tipos de trajetórias, por exemplo, percursos realizados por pessoas a pé ou de bicicleta.

O agrupamento baseia-se numa medida de similaridade. A extração de informação consiste em criar uma trajetória representativa para cada grupo de trajetórias. As trajetórias representativas podem ser visualizadas usando uma aplicação web, sendo também possível configurar cada módulo do sistema com parâmetros desejáveis, na sua maioria distâncias limiares. Por fim, são apresentados casos de teste para avaliar o desempenho global do sistema desenvolvido.

Keywords Moving objects, Spatio-temporal data, Trajectory clustering, Recommendation systems

Abstract

The widespread use of devices to capture and transmit data about the location of objects over time allows collecting large volumes of spatio-temporal data. Consequently, there has been in recent years a growing demand for tools and techniques to analyze large volumes of spatio-temporal data aiming at providing a wide range of location-based services.

This dissertation focuses on the development of a system for recommendation of trajectories based on historical data about the location of moving objects over time. The main issues covered in this work are trajectory clustering and extracting information from trajectory clusters. This study is not restricted to data from vehicles and can also be applied to other kinds of trajectories, for example, the movement of runners or bikes.

The clustering is based on a similarity measure. The information extraction consists in creating a representative trajectory for the trajectories clusters. Finally, representative trajectories are displayed using a web application and it is also possible to configure each system module with desired parameters, mostly distance thresholds. Finally, case studies are presented to evaluate the developed system.

Contents

CONTENTS I

LIST OF FIGURES III

LIST OF TABLES IV

LIST OF ABBREVIATIONS AND ACRONYMS V

INTRODUCTION 1

1.1 OBJECTIVES 2
1.2 OUTLINE OF THE DISSERTATION 3

OVERVIEW ON SPATIAL AND TRAJECTORY DATA ANALYSIS 5

2.1 SPATIAL AND TRAJECTORIES CLUSTERING 5
2.1.1 Classical Spatial Clustering algorithms 6
2.1.2 Density-based algorithms 6
2.1.3 Hierarchical algorithms and Moving Clusters 8
2.1.4 Graph-based algorithms 8
2.1.5 Partitioning algorithms 13
2.1.6 Applications of Clustering Trajectories 14
2.1.7 Delaunay Triangulation 18
2.2 TRAJECTORY SIMILARITY AND DISTANCE MEASURES 21
2.2.1 Distance function for time series on free moving objects 22
2.2.2 Similarity of Trajectories in Spatial Networks 26
2.3 COMPARISON OF CLUSTERING ALGORITHMS 30
2.4 SUMMARY 31

METHODOLOGY 33

3.1 PROBLEM FORMULATION 34
3.2 IMPLEMENTATION 35
3.2.1 Trajectories Filtering Step 36
3.2.2 Trajectory Clustering Step 38
3.2.2.1 Distance Function 38
3.2.2.2 Clustering Algorithm 40
3.2.3 Representative Trajectory Step 43
3.2.3.1 Positions Connectivity Extraction 43
3.2.3.2 Centroid Position Representation 45
3.2.3.2.1 Neighbors Closeness 46
3.2.3.2.2 Centroid Assignment 46
3.2.3.3 Representative Trajectory Algorithm 47
3.3 IMPLEMENTATION 48
3.3.1 Technical Specifications 48
3.3.2 Data Visualization 49
3.3.3 Interface (Front-End) 50

RESULTS 53

4.1 ACCURACY TESTS 53
4.1.1 Test case one 53
4.1.2 Test case two 55
4.1.3 Test case three 58
4.1.4 Test case with modified parallel weight 60
4.2 PERFORMANCE TESTS 61

CONCLUSION **65**

5.1 DISCUSSION 65
5.2 CONTRIBUTION 66
5.3 FUTURE WORK 66

BIBLIOGRAPHY **68**

List of Figures

Figure 1: Trajectories found by Google Maps with traffic information. 2
Figure 2: DBSCAN clustering of three point datasets. (Ester, Kriegel et al. 1996) 7
Figure 3: Clusters problem associated to distance and density: (a) to (e) are distance-separated cluster problems and (f) to (h) density-separated. (Zhong, Miao et al. 2010). 9
Figure 4: (a) Graph constructed by Delaunay Triangulation; (b) Distribution of final feature values of D1; (c) The data point's boundary of D1 detected by TRICLUST; (d) Clustering result of D1. (Liu, Nosovskiy et al. 2008) 10
Figure 5: Clustering results of one simulated dataset (x represents the noise): (a) DBSC algorithm; (b) K-Means algorithm; (c) CURE algorithm; (d) GDBSCAN algorithm; (e) Geo-SOM algorithm; (f) ASCDT algorithm. (Liu, Deng et al. 2012) ... 11
Figure 6: Spatial clustering using DBSC algorithm in Jiangsu Province per capita road area data (the x symbol represents noise). (Liu, Deng et al. 2012) 12
Figure 7: Trajectory clustering in the partition-and-group framework. (Lee, Han et al. 2007) 13
Figure 8: Trajectory partitions. (Lee, Han et al. 2007) 13
Figure 9: The three components of the distance function for line segments Li and Lj. (Lee, Han et al. 2007) 14
Figure 10: Trajectory clustering using different distance functions result. (Rinzivillo, Pedreschi et al. 2008) 15
Figure 11: Traffic on May 5 and on August 7, at midnight. (Liu, Zheng et al. 2011) 16
Figure 12: Architecture of smart itinerary recommender. (Yoon, Zheng et al. 2010) 17
Figure 13: TCMM framework. (Li 2010)The representative line segment from the cluster is computed in each one micro-cluster. The distance function is adapted from a similarity measure used in pattern recognition (Chen, Leung et al. 2002) also used in (Lee, Han et 18
Figure 14: Delaunay Triangulation example. 18
Figure 15: (A) The GPS data points in the map. (B) A selection of five trajectories represented on the map. (C) The representative points (blue color). (D) Approximated representative trajectories. (Guo, Liu et al. 2010) 19
Figure 16: Density trajectories at different timestamps. (Guo, Liu et al. 2010) 20
Figure 17: (a) Two clusters of trajectories. (b) The blue cluster has 94 trajectories (c) the red cluster contains 136 trajectories, mainly in the south region. (d) 46 trajectories are in both regions. (Guo, Liu et al. 2010) 21
Figure 18: Representations of trajectories in unconstrained and constrained contexts. (Tiakas, Papadopoulos et al. 2006).... 21
Figure 19: Euclidean distance. (Yanagisawa and Satoh 2006) 23
Figure 20: (a) Difference between Euclidean distance and DTW; (b) DTW. (Yanagisawa and Satoh 2006) 24
Figure 21: Example of time-series for ranking. (Chen and Ng 2004) 25
Figure 22: Distance example between Euclidean space and network space. (Hwang, Kang et al. 2005) 26
Figure 23: Spatio-Temporal trajectories. (Hwang, Kang et al. 2005) 27
Figure 24: Hausdorff distance d on road network; (a) TRA and TRB; (b) TRC and TRD. (Hwang, Kang et al. 2005) 27
Figure 25: Temporal and spatial attributes of trajectories. (Xia, Wang et al. 2010) 29
Figure 26: The five hundred trajectories derived from 25 moving objects. 33
Figure 27: Representations of four trajectories crossing a given origin and destination. 35
Figure 28: High-level conceptual model. 36
Figure 29: Set of trajectories crossing an origin and a destination. 37
Figure 30: The three measures types between two line segments with the corresponding intermediate points. 38
Figure 31: Line distance measures. (Chen, Leung et al. 2002) 40
Figure 32: Two similar trajectories with the corresponding line segments. 41
Figure 33: Representation of two visually clusters. 42
Figure 34: Representation of regions of neighbor positions. 44
Figure 35: Delaunay Triangulation of the cluster. 44
Figure 36: Representative Trajectory. 47
Figure 37: Logical Architecture component diagram. 48
Figure 38: GeoJSON output example structure. 50
Figure 39: Web Application general structure. 51
Figure 40: Test case 1 trajectories. 54
Figure 41: Test case 1, trajectories cluster and corresponding representative trajectory (black). 55
Figure 42: Test case 2 trajectories. 56
Figure 43: White representative trajectory illustrated on the Greece map roads. 58
Figure 44: Test case 3: sub-trajectories (1) Cluster 2 zoomed (yellow). 59
Figure 45: Cluster C2 is decomposed in two clusters. 60
Figure 46: Trajectories Filtering Step execution times. 61
Figure 47: Clustering step execution time. 62
Figure 48: Representative trajectory sub-tasks execution times. 63
Figure 49: Sum of all steps times depending on the trajectories number. 63

List of Tables

Table 1: Statistical information for per capita road area data clusters (units: m2) (Liu, Deng et al. 2012)............................12
Table 2: Overall of the similarity measures. (Chen, Tamer et al. 2005) 25
Table 3: A comparison of spatial clustering algorithms. (Liu, Deng et al. 2012) 31
Table 4: Comparison of different trajectory clustering approaches.(Mahrsi 2013)..31
Table 5: Triangles resulted from DT...45
Table 6: The vertices connections...45
Table 7: Groups of neighbor vertices...46
Table 8: Groups of representative centroids. 46
Table 9: Input parameters of test case 1..54
Table 10: Input parameters of test case 2..57
Table 11: Input parameters of test case 3..59

List of abbreviations and acronyms

CRP	Centroid Representation Position.
CRS	Coordinate Reference System.
DM	Data Mining.
DT	Delaunay Triangulation.
DTW	Dynamic Time Warping.
ED	Edit Distance.
EDR	Edit Distance on Real.
ERP	Edit Distance with Real Penalty.
ES	Euclidean Space.
HD	Hausdorff Distance.
JSF	JavaServer Faces.
LCSS	Longest Common SubSequences.
LM	Lehmer Mean.
LS	Line Segment.
MDL	Minimum Description Length.
MO	Moving Objects.
PCE	Positions Connectivity Extraction.
POI	Point of Interest.
RS	Road Segment.
RT	Representative Trajectory.
ST	Synchronized Traverse.
TI	Triangle Inequality.
TWD	Time Wrapping Distance.
WAR	Web Application Archive.

1. Chapter

Introduction

The recent technological advances allowed to obtain large amounts of location data using devices such as Smartphone, GPS or RFID tags. Moreover, the widespread use of these location-aware devices has encouraged many researchers and developers to explore this huge amount of information in a diversity of applications such as fleet management systems, GPS-tracking and logger applications. *My Tracks* and *RunKeeper* (Android applications) are two examples of solutions that allow recording trajectories data such as, time, location, speed, distance travelled and elevation, as well as estimating fuel consumption or calories burned during physical activity. Other works are focused on spatial-temporal data streams analysis, for example, using surveillance data, for monitoring and study animal's trajectories (in case of hibernation), earthquakes, hurricanes and touristic itineraries (Yoon, Zheng et al. 2010). The solutions proposed in these works consider that there are no or very few constraints on the spatio-temporal behavior of the represented entities. However, another important class of applications deal with the analysis of data about networked Moving Objects (MO), e.g., the movement of taxis in roads networks.

One of today's major environmental concerns is the pollution generated by vehicles in low rotations caused by traffic congestion. Several applications were developed to detect congested roads. Successful examples are *Waze (Shinar 2009)*, which is a collaborative platform used to register aperiodic live events and traffic information and V-Traffic [1] a French traffic real-time information system. These are live applications based on user collaboration, information feedback or aggregation of several sources of information, which do not predict or recommend alternative routes based on historical data.

Recently Google Maps has released a service to search for the path between two predefined locations using real-time traffic information (Figure 1).

[1] Source: v-traffic.com

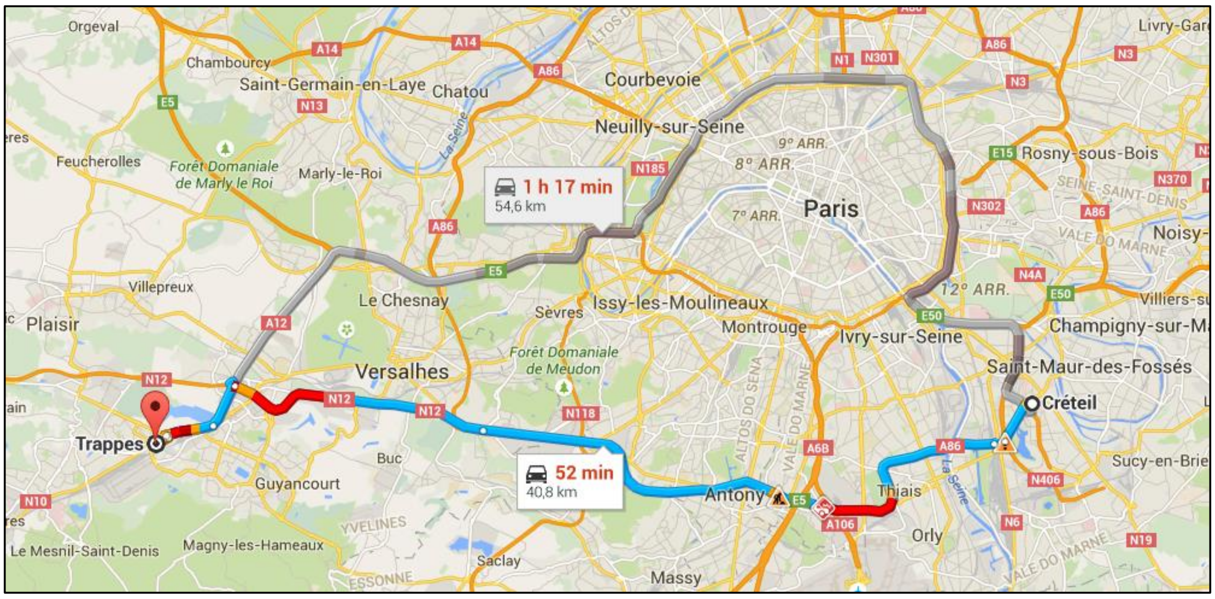


Figure 1: Trajectories found by Google Maps with traffic information.

Currently, real-time traffic information is only available at certain places and this information is also dynamic, which means that it may change significantly along time, e.g., because drivers have different behavior patterns in rush hours or they change decisions when traffic jams occur. Hence, the analysis of historical data can also play an important role in this context, to extract spatio-temporal data patterns for traffic data and use that information for the recommendation of trajectories.

1.1 Objectives

Spatial Data Mining (DM) is a process to extract meaningful information from a source, normally a dataset, and transform it into something understandable. There are three main dimensions explored in moving objects Cluster analysis: spatial, temporal and semantic. The spatial domain is related with coordinates, specifically with the MO position. The temporal domain consists of timestamps. Finally, the semantic domain refers to places, i.e., positions that can be associated to a semantic tag, such as churches, metro stations, pubs, etc.

The focus of this work is on clustering unconstrained trajectories data, i.e., it is considered that objects move freely in space (e.g., ships in the ocean), and if spatial constraints exist (e.g., road networks) they are ignored.

The objective is to develop a trajectories recommendation system that should be able to answer questions such as:

1. "Which trajectories exist from a given origin to a given destination?"
2. "Which trajectories are similar?"
3. "What useful information can be extracted from a cluster of similar trajectories?"

4. “What is the best trajectory from an origin to a destination?”

The data sources are GPS positions recording the trajectories of moving objects along time. A trajectory is considered as a set of GPS positions and the system must be able to extract trajectories in a dataset and prepare them for data analysis. It must be possible to discover trajectories that share an origin and a destination and to find clusters of similar trajectories using a similarity measure. Finally, it must be possible to extract information from trajectories clusters and use this information to recommend a trajectory between an origin and a destination. The generated results must be presented through a web application.

The main issues studied in this work are:

- Clustering: how to create clusters of similar trajectories.

- Outlier Detection: outliers are positions that are highly unlikely to occur in a trajectory. The outliers and anomalies are rare and usually they indicate faulty device collection. This kind of anomalies make the process of finding similar trajectories more difficult.

- Representative Trajectory (RT): usually real-world entities do not have a random behavior, hence it is possible to estimate a representative trajectory from a set of similar trajectories.

1.2 Outline of the Dissertation

The contents of this document is organized in five chapters:

- **Chapter 1** – **Introduction**, presents the motivation and purpose of this work.
- **Chapter 2** – **Overview on spatial and trajectory data analysis,** presents related work on clustering spatio-temporal data with particular focus on moving object trajectories. It presents an overview of clustering algorithms, trajectory clustering applications, Delaunay Triangulation, and trajectories similarity including distance measures and clustering comparison.
- **Chapter 3 – Methodology**, describes the conceptual framework. It starts with the problem formulation, followed by a description of the four main steps required to implement the solution proposed in this work. It also presents the implementation components, web application, technical specifications and end user interface.
- **Chapter 4** – **Results**, presents three test cases for the evaluation of the proposed system.
- **Chapter 5** – **Conclusion**, presents a brief overview of this work. It discusses the limitations and presents an outline of future developments and recommendations to deal with the limitations of the solutions proposed in this work.

2. Chapter

Overview on spatial and trajectory data analysis

This chapter presents an overview on clustering spatio-temporal data, with particular emphasis on trajectories of moving objects. The aim of this technique is to partition spatial data into groups, called clusters or subclasses. Each cluster has several features (e.g., geometrical shapes) in common, i.e. the spatial objects or the trajectories in a cluster are similar to each other and in general dissimilar to the rest of the clusters. The similarity concept is very important in clustering and the results depend largely on the data source. For instance, the sampling rate, used to record the trajectories of moving objects, is of great importance on the similarity measures.

Section 2.1 describes the main techniques for clustering spatial data, Section 2.2 presents recent solutions for trajectory clustering, Section 2.3 presents a comparison of clustering algorithms and Section 2.4 presents a summary of the main achievements in this area.

2.1 Spatial and Trajectories Clustering

Clustering is largely used in the field of spatial data analysis. According to (Mahrsi 2013) and (Han, Kamber et al. 2000), it is important to take into consideration the following characteristics when using clustering techniques:

Application objective

The choice of a clustering algorithm is based on the goal of the application, i.e., if it is based, on positions density or trajectories, probably a type of algorithm could be more suitable in terms of performance and accuracy. If the objective is the clustering of trajectories, the use of algorithms or methods based on sub-trajectories or line segments is more interesting.

Speed versus quality

The clustering of large datasets raises a major problem: computation time. This means that the clustering of data in a quickly and accurately way is not always possible.

Data Characteristics

The type of data to be clustered is another important aspect. Usually, the algorithms compute numeric features, for example, the similarity (distance measures) between objects (for example, trajectories) in the Euclidean space. Some clustering algorithms take also into consideration spatial, temporal, or both, and semantic attributes, depending on the goal. Another concerning to be aware of, are outliers or noise in the data sets, because some of clustering algorithms are very sensitive to this abnormalities.

2.1.1 Classical Spatial Clustering algorithms

The spatial clustering algorithms can be classified into seven categories (Han, Kamber et al. 2000), (Yang and Cui 2009) and (Liu, Nosovskiy et al. 2008): density-based, hierarchical, grid-based, graph-based, partitioning, model-based and combinational algorithms. In this section we briefly review only the most relevant for our study, which are: density-based, hierarchical, graph-based, partitioning and clustering with Delaunay Triangulation (DT). The latter two, are important, because they are directly involved in solving the clustering and representative trajectory problems.

The variety of algorithms developed during the past decade is huge, and the best choice depends on the problems that we want to solve. In (Liu, Deng et al. 2012), several algorithms are compared, in order to find the best solutions for distinct problems in spatial clustering, namely: (i) discovery of arbitrary shaped clusters; (ii) discovery of clusters with uneven density; (iii) robustness to noise; (iv) heavily reliant on prior knowledge; (v) consideration of both spatial proximity and attribute similarity. It was known, that an algorithm cannot solve all the requirements, and a new density-based algorithm, called DBSC, was developed to deal with two important issues in spatial clustering: (1) the construction of spatial proximity relationships and (2) the cluster of spatial objects with similar attributes. To manage the first issue, the authors propose using a Delaunay Triangulation, which will be discussed in Section 2.1.7, as it is used in the generation of the representative trajectory of clusters.

2.1.2 Density-based algorithms

These types of algorithms are commonly used to identify regions which are separated from other regions by low-density regions (Bäcklund, Hedblom et al. 2011), i.e., the clusters are defined as regions of higher density compared with the entire regions in the dataset. The existence of sparse areas, i.e., regions with low density, are usually considered as noise, and these are discarded by the algorithm. There are two well-known density-based algorithms in the literature, which are: DBSCAN (Ester, Kriegel et al. 1996), and a variant of this, known as OPTICS (Ankerst, Breunig et al. 1999). Other algorithms, such as DENCLUE (Xie, Chang et al. 2007) or CURD (Ma, Wang et al. 2003), also integrate this category of density-based algorithms. This section, only focuses on the most relevant: DBSCAN and OPTICS.

The DBSCAN data clustering algorithm was proposed by (Ester, Kriegel et al. 1996) to deal with the following issues: (i) to reduce à priori knowledge, in terms of points density, to make it easier to determine the input parameters in a large dataset (ii) to discover clusters of arbitrary shapes and (iii) efficiency when dealing with large datasets.

DBSCAN algorithm uses two parameters, which are *ε-eps* denoting a maximum distance radius and *MinPts* denoting the minimum number of points required to create density clusters. It is sensitive to these two parameters, and a trial-and-error approach is mandatory, to find a good combination of these parameters.

Figure 2 depicts three density clustering examples, using three datasets obtained from the SEQUOIA 2000 [2] benchmark database. This algorithm is fully described in (Ester, Kriegel et al. 1996) and (Bäcklund, Hedblom et al. 2011).

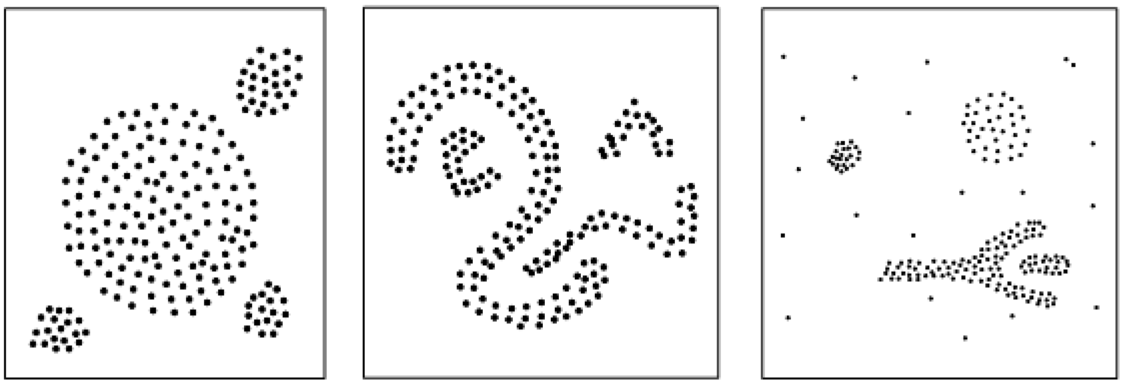


Figure 2: DBSCAN clustering of three point datasets. (Ester, Kriegel et al. 1996)

The DBSCAN algorithm has many applications such as the analysis of satellites images, anomaly detection in temperature data or even x-ray crystallography. Classifying large spatial datasets is complex, computationally heavy and the DBSCAN algorithm can address these requirements well.

The OPTICS algorithm (Ankerst, Breunig et al. 1999) and (Han, Kamber et al. 2000), is a generalization of DBSCAN to multiple ranges, by considering a maximum search radius. As DBSCAN, this algorithm also requires two parameters (*Eps* and *MinPts*), but instead of producing a clustering, it produces an ordering of the dataset points to identify the clustering structure. The aim is to addresses one of DBSCAN's major weaknesses: the problem of detecting meaningful clusters in data of varying density.

The authors of (Birant and Kut 2006) introduce a clustering algorithm based on DBSCAN, called ST-DBSCAN. The goal of this algorithm is to discover clusters using non-spatial, spatial and temporal data.

[2] Sources: s2k-ftp.cs.berkeley.edu/ , asc.llnl.gov/sequoia/benchmarks

A similar approach is followed in (Lee, Han et al. 2008) for the classification of trajectory data. This work considers two types of clustering: (i) region-based clustering and (ii) trajectory-based clustering. The trajectory-based clustering, is particularly relevant to this work, and uses the partition-and-group framework proposed in (Lee, Han et al. 2007) for line segments classification. The main purpose of this type of clustering, is to create clusters of similar partitioned trajectories. This subject is covered in detail in the following sections.

NETSCAN (Kharrat, Popa et al. 2008) is an algorithm also based on DBSCAN, to deal with the clustering of trajectories, with application to traffic-density analysis. This algorithm follows a two-step approach: (i) define the similarity between road segments and aggregate them in dense paths; (ii) compute a similarity measure between trajectories and compose the trajectory clusters around the dense paths.

2.1.3 Hierarchical algorithms and Moving Clusters

Hierarchical algorithms create a decomposition of a given dataset using a *dendrogram*. A *dendrogram,* is a tree, which displays the arrangement of clusters. It can be created in two ways: the agglomerative ("bottom-up") or divisive ("top-down") approaches. The agglomerative approach starts with each object forming a separate group, i.e., each object starts in its own cluster, and pairs of clusters are merged recursively towards the top of the hierarchy. The divisive approach starts with all objects in the same cluster, which is divided recursively as the algorithm moves down in the hierarchy (Han, Kamber et al. 2000), (Yang and Cui 2009).

There are several works in the literature based on these approaches, for example: BIRCH (Zhang, Ramakrishnan et al. 1996), CURE (Guha, Rastogi et al. 1998) and CHAMELEON (George, Karypis et al. 1999). The last developments on hierarchical algorithms are also able to deal complex data, but determining the input parameters of clustering algorithms remains an open issue and the application of these clustering algorithms to spatial data is not straightforward (Liu, Deng et al. 2012).

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) is a notable example, and it consists of an algorithm to compress data objects into small sub-clusters. According to (Zhang, Ramakrishnan et al. 1996), the BIRCH algorithm is capable of finding a good clustering with a single scan of the data. However, the quality of the results can be improved with a few additional scans and the algorithm is also capable of handling "noise" effectively. The main purpose of BIRCH is to solve the problem of lack (limited) of memory, which is usually smaller than the dataset size.

2.1.4 Graph-based algorithms

This type of algorithms are applied in several areas, such as biology, social sciences, technology, economy and traffic management. Typically, data is organized in a graph representation, based on relationships or proximity among objects (Guo 2009). It is possible to discover clusters of arbitrary shapes, but when dealing with datasets associated with noise, this type of algorithm is disadvantageous. There exists a variety of types of graphs, including spatial graphs where the edges represents the proximity (weight) between objects. In the case of trajectory graphs, the edges represent the similarity between trajectories. There are methods using minimum spanning trees (Zhong, Miao et al. 2010), k-nearest neighbors (George, Karypis et al. 1999) or Delaunay Triangulation (DT) (Guo, Liu et al. 2010), (Zheng, Chen et al. 2008) and (Kang, Kim et al. 1997).

The basic approach is to firstly construct the graph and then apply an algorithm to create a set of sub-graphs. For example, the authors of (Zahn 1971) apply methods based on the minimum spanning tree (commonly calculated by *Prim* and *Kruskal* algorithms) to create the clusters and eliminate "inconsistent" edges. *2-MSTClus* (Zhong, Miao et al. 2010) uses a similar approach, applied in pattern recognition to calculate clusters in graphs. The problems of the clusters are classified into two categories: (i) "distance-separated" and (ii) "density-separated". Figure 3, depicts some examples of typical cluster problems.

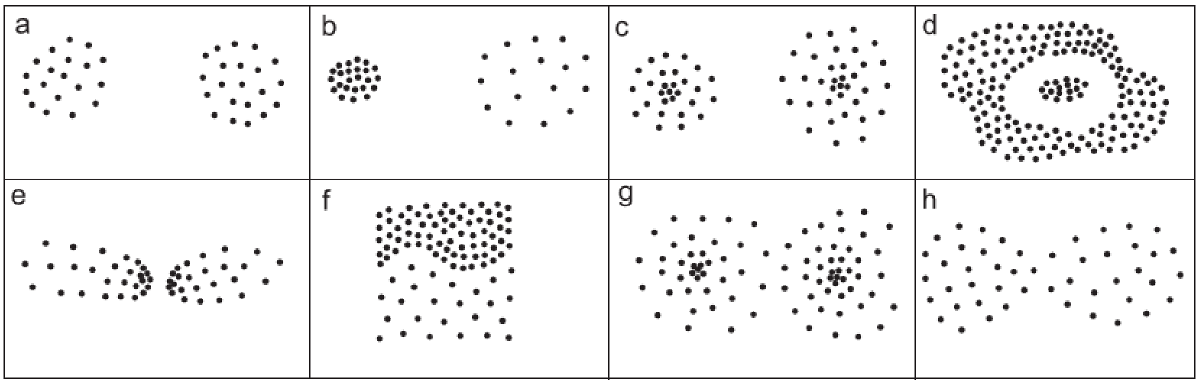


Figure 3: Clusters problem associated to distance and density: (a) to (e) are distance-separated cluster problems and (f) to (h) density-separated. (Zhong, Miao et al. 2010).

Figure 3 (a), shows two independent clusters with similar shapes, density and size. This occurs, because inter-cluster density is higher than the intra-cluster pairwise distance. The remaining distance-separated clusters, (b) to (e), illustrate that there are different kinds of clusters regardless of their size or density. Density-separated clusters, illustrated in figure 3 (f-h), consider a density gradient rather than a distance. The clusters in (g) and (h) differ from (f) because the two clusters touch each other superficially, but they are distinct. These are usually denoted as *touching* clusters. (Zahn 1971)

The methods based on DT rely on the concept of a connected graph where the nodes represent spatial objects and the edges denote the nearest pairs of spatial data points (Liu, Nosovskiy et al. 2008). There are several approaches that have been implemented in the context of geographical information systems (GIS) (Kang, Kim et al. 1997), (Isenburg, Liu et al. 2006), (Liu, Deng et al. 2012), (Yang and Cui 2009) and (Zheng, Chen et al. 2008). According to (Liu, Deng et al. 2012) there are two main issues to perform the spatial clustering: (i) the construction of spatial proximity relationships and (ii) the clustering of spatial objects with similar attributes. The first step consists in capturing the proximity relationships and the second consists in a density-based method. The main purpose of using a DT is to find a quick way to capture the proximity relationships, between the points in the spatial dataset. In other words, a DT is used to identify the natural neighbourhood of spatial objects in $O(nlogn)$ time complexity.

An example of how a DT can be used in spatial clustering problems is presented in (Liu, Nosovskiy et al. 2008). This work presents a clustering algorithm called TRICLUST, based first on a DT construction, followed by a process of feature extraction, from data point's neighborhood relationships. Then, it is defined a criteria function and a *K-Means* [3] method is applied to compute a threshold for the criteria function. The classification of the objects is performed, by applying a threshold. Figure 4, depicts this method, used in two-dimensional testing dataset.

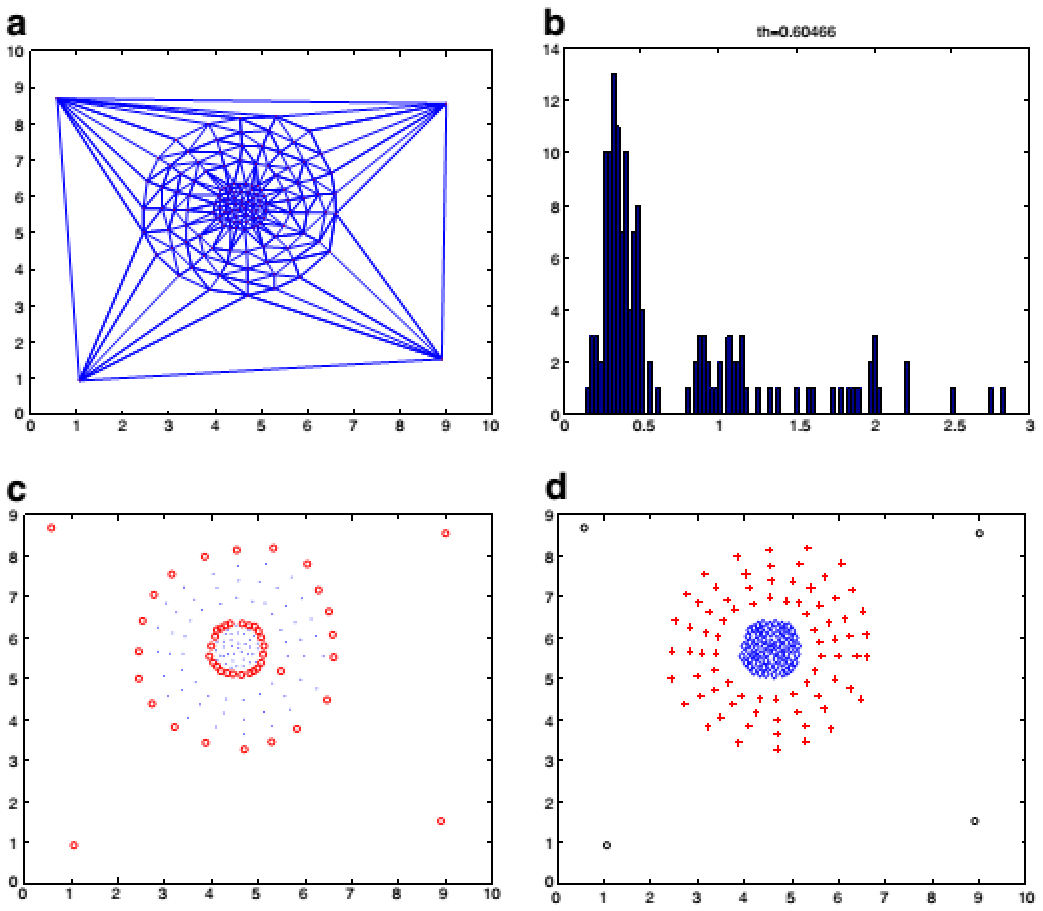


Figure 4: (a) Graph constructed by Delaunay Triangulation; (b) Distribution of final feature values of D1; (c) The data point's boundary of D1 detected by TRICLUST; (d) Clustering result of D1. (Liu, Nosovskiy et al. 2008)

In (Liu, Deng et al. 2012) an algorithm called DBSC is proposed. The first step, consists also in creating a spatial proximity relationships graph, using DT, and the second step is derived from the

[3] Source: en.wikipedia.org/wiki/K-means_clustering

density-based clustering strategy, called ASCDT (Deng, Liu et al. 2011). The latter is used for clustering spatial objects with similar attributes, i.e., considering the distance relationships among objects. The authors have used two simulated datasets (generated in ArcGIS[4]) and three real datasets to validate the algorithm. In addition, DBSC is compared with five spatial clustering algorithms: CURE, K-Means, GDBSCAN, Geo-SOM and ASCDT. Figure 5, presents the clustering results obtained using each algorithm.

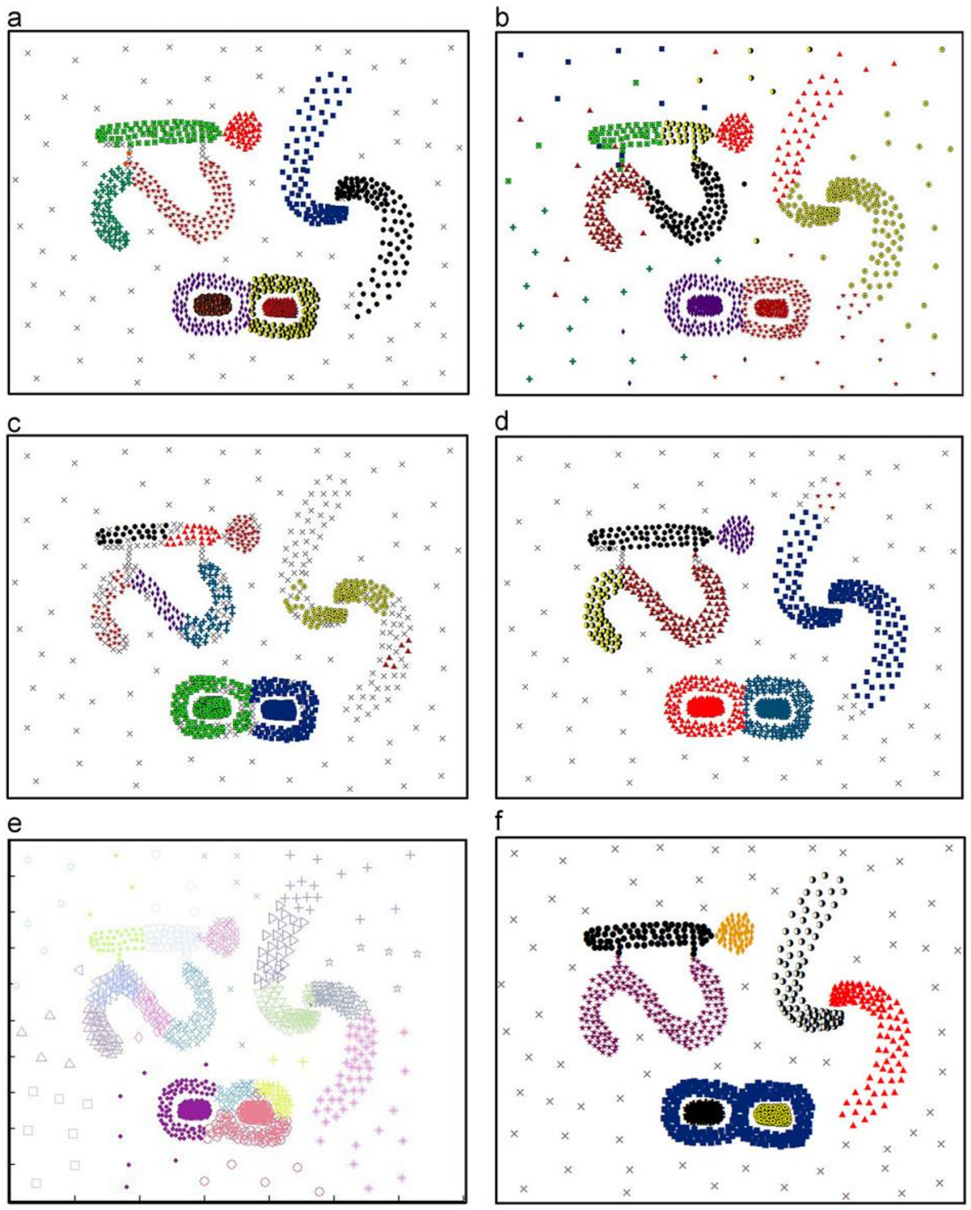


Figure 5: Clustering results of one simulated dataset (x represents the noise): (a) DBSC algorithm; (b) K-Means algorithm; (c) CURE algorithm; (d) GDBSCAN algorithm; (e) Geo-SOM algorithm; (f) ASCDT algorithm. (Liu, Deng et al. 2012)

[4] Source: arcgis.com

The ASCDT algorithm (e) is able to separate the points in dataset into 10 clusters and it was also able to deal with noise successfully.

The three real datasets used in the experiments are: meteorology, environment and urban development. The latter, is a dataset of 853 records from Jiangsu Province in China and represents per capita road area in the Jiangsu Province town for the year 2008. Aiming to study the traffic development level, figure 6 shows the clustering results achieved using the DBSC algorithm, followed by table 1, containing statistical information about the clusters.

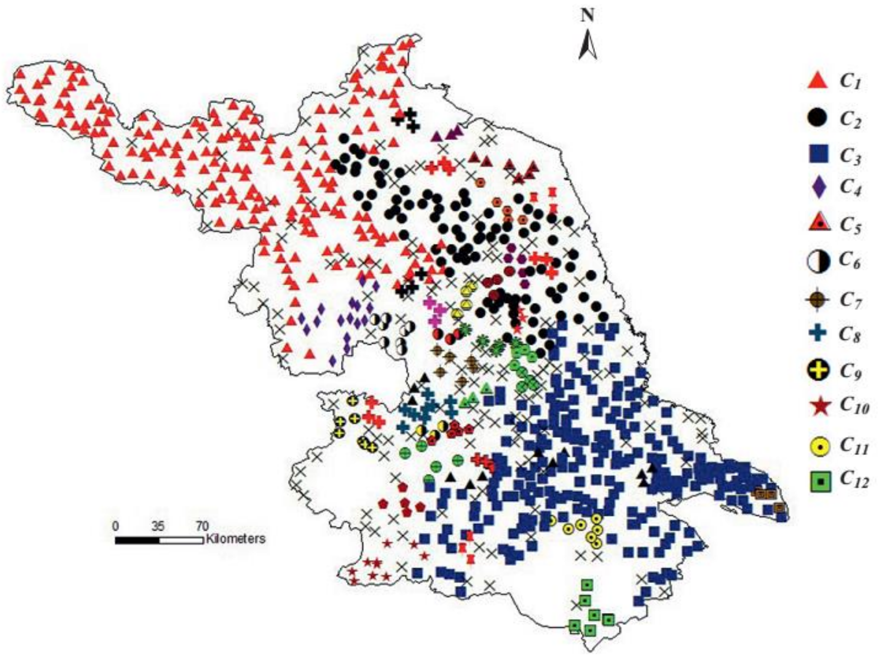


Figure 6: Spatial clustering using DBSC algorithm in Jiangsu Province per capita road area data (the x symbol represents noise). (Liu, Deng et al. 2012)

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
Number	197	92	230	16	6	8	6	9	7	10	7	7
Mean value	17.7	11.2	17.4	8.6	8.7	12.9	14.2	17.4	23.6	14.4	23.8	10.5
Standard deviation	1.9	2.2	2.0	1.9	1.2	2.3	2.5	1.9	1.3	2.0	1.3	0.8

Table 1: Statistical information for per capita road area data clusters (units: m^2) (Liu, Deng et al. 2012)

The DBSC algorithm discovers 12 main clusters, and according to table 1 and figure 6, it is possible to conclude the existence of 3 main clusters (C1, C2 and C3). It is also possible to conclude, that traffic development is more significant from the north to southeast. C2 has an area value lower than the other two main clusters. In the center of the map in figure 6, several clusters are presented, with no statistics relevance, because of noise.

2.1.5 Partitioning algorithms

TRACLUS (Lee, Han et al. 2007) is an algorithm that aims at discovering common sub-trajectories by partitioning trajectories into a set of line segments and then on clustering them based on the similarity between the line segments. The framework consists of two phases, as presented in Figure 7: the first is the partitioning, followed by the line segments grouping phase.

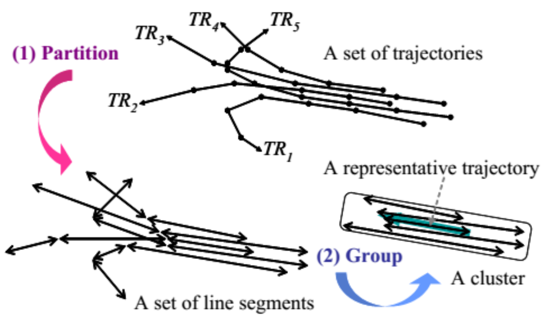


Figure 7: Trajectory clustering in the partition-and-group framework. (Lee, Han et al. 2007)

The partitioning phase transforms each trajectory into a set of feature points, which are discovered by searching for important changes in trajectories positions. The connection of two feature points forms a trajectory partition, which is represented by a line segment, as depicted in Figure 8. The partitioning phase, uses the Minimum Description Length (MDL) (Grnwald, Myung et al. 2005) to capture the optimal *trade-off*, between two main and desirable requisites: (i) preciseness and (ii) conciseness.

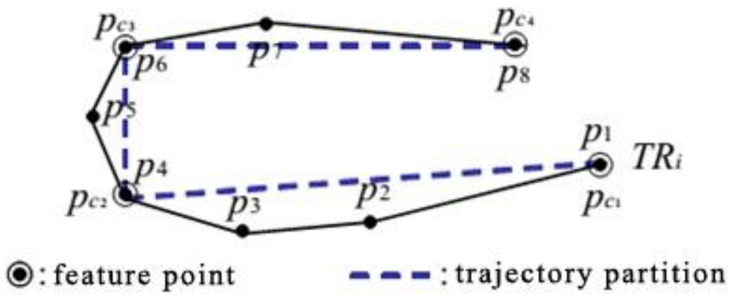


Figure 8: Trajectory partitions. (Lee, Han et al. 2007)

The second phase, consists in grouping similar line segments (trajectory partitions) into a cluster. The clustering proceeds as follows: a distance function is primarily defined, in analogy to a proximity measure, in order to obtain the similar line segments. The distance function consists of three components and it is adapted from similarity measures used in pattern recognition domain (Chen, Leung et al. 2002). Figure 9 illustrates the three components of the distance function: (i) perpendicular distance ($d_\perp$), (ii) parallel distance ($d_\parallel$) and (iii) angle distance (d_θ). These distance function components, estimate the similarity between two line segments in a two-dimensional space, where L_i and L_j represent the two line segments (Figure 9).

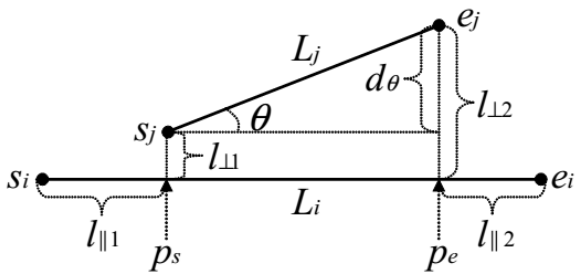


Figure 9: The three components of the distance function for line segments L_i and L_j. (Lee, Han et al. 2007)

The three measures are as follows:

$$d_{\perp}(L_i, L_j) = \frac{{l_{\perp 1}}^2 + {l_{\perp 2}}^2}{l_{\perp 1} + l_{\perp 2}} \tag{1}$$

$$d_{\parallel}(L_i, L_j) = MIN(l_{\parallel 1}, l_{\parallel 2}) \tag{2}$$

$$d_{\theta}(L_i, L_j) = \begin{cases} \|L_j\| \times \sin(\theta), & if\ 0^\circ \le \theta < 90^\circ \\ \|L_j\|, & if\ 90^\circ \le \theta \le 180^\circ \end{cases} \tag{3}$$

The purpose of the distance function is to define the density of line segments. Then a DBSCAN based algorithm is proposed, where the clustering of points is adapted to perform the clustering of line segments. At the end, the representative (RT) of each cluster is computed. This trajectory represents the major behavior or overall movement of the moving objects (line segments) in a cluster. The ordering of the line segments in the representative trajectory is determined by applying a sweep line technique.

There are several works (Lee, Han et al. 2008), (Jiashun 2012) and (Lee, Han et al. 2008), which have been proposed in literature based on this framework. The first work focuses on trajectories outlier detection. In (Jiashun 2012) an algorithm based on TRACLUS, called SPSTC (Shielding Parameters Sensitivity Trajectory Clustering), is proposed to deal with issues concerning input parameters sensitivity. The authors consider parameters sensitivity an important limitation in trajectory clustering, because the results differ according to the input parameters values and thus results are uncertain.

2.1.6 Applications of Clustering Trajectories

This section presents examples of trajectory clustering applications. The authors of (Rinzivillo, Pedreschi et al. 2008) introduce the concept of progressive clustering on large real datasets. The objective is to use clustering techniques in visual exploration of a large number of trajectories. It uses the OPTICS algorithm to cluster the trajectories and several distance functions are provided. The experimentation scenario consists of a dataset composed by GPS-positions of 17.241 vehicles tracked during one week in Milan city. The total number of trajectories is 176.000 trajectories, but the study has used a subset of 6187 trajectories collected in one day. Figure 10 shows a progressive clustering. The authors have obtained the clusters (in the left) by clustering with the distance function “common destination” according to the trip destinations. The algorithm, also uses a distance function named “route similarity” (one of the four distance functions used in the experiments) to find the clusters members. The result is depicted in the middle, and it is shown that most trajectories are short. The right-hand image depicts the results obtained using another distance function “common source and destination” to group the trajectories according to the trips origin and destination.

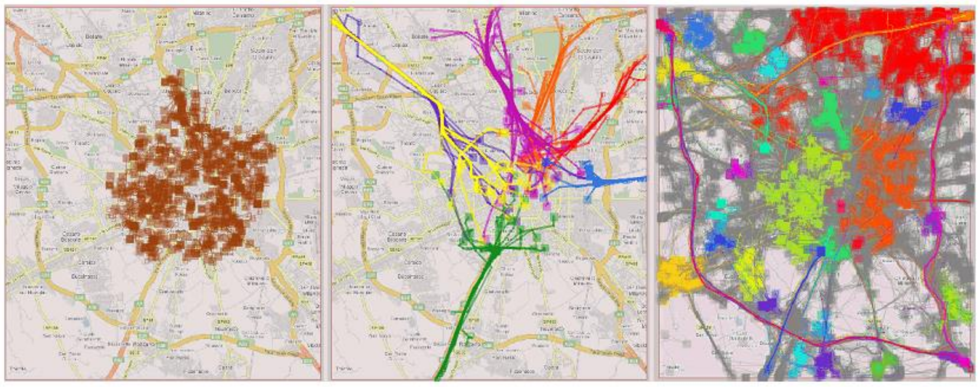


Figure 10: Trajectory clustering using different distance functions result. (Rinzivillo, Pedreschi et al. 2008)

The paper (Yuan, Zheng et al. 2011) presents a recommender system for taxi drivers and persons who wish to take a taxi. The recommendations are based on parking locations obtained from historical taxis GPS positions and status information to determine when taxis are parked, cruising or occupied. The goal is to reduce fuel consumptions and to make peaking a taxi an easier task. The solution is based on a partition-and-group framework, which consists of an offline mining of the trajectories and an online recommendation for taxi drivers and passengers. The proposed solution uses OPTICS to discover the taxis parking zones.

In (Liu, Zheng et al. 2011) it is proposed a solution to discover abnormal traffic streams on road networks, i.e., unusual patterns of moving objects trajectories. The authors use spatial and temporal features to detect traffic outliers as follows: (i) decompose a city (in this case, the major roads of the city of Beijing) into regions and build a region graph by scanning the Beijing dataset trajectories; (ii) detect outliers in graph links; (iii) create a list of temporal and spatial trees denoting the possible paths, as depicted in Figure 11.

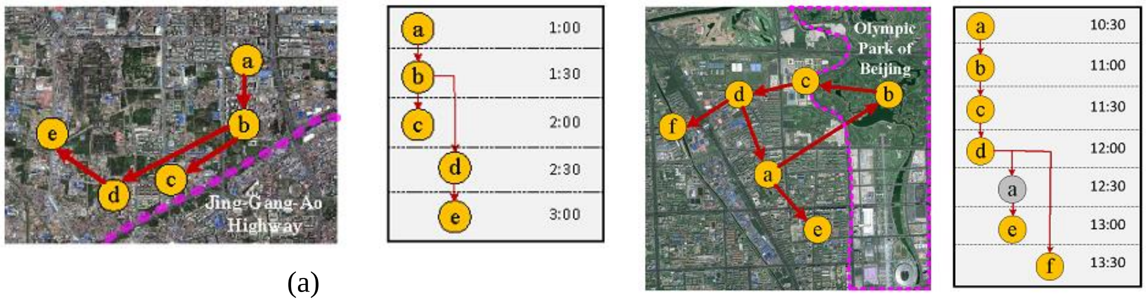


Figure 11: Traffic on May 5 and on August 7, at midnight. (Liu, Zheng et al. 2011)

These examples report two unusual events and put in evidence the kind of causal interactions that are detected by the algorithm. Figure 11 (a) depicts a situation on May 5, at midnight, when the highway (the dotted line) was under traffic control for building viaducts. It is possible to observe the impact caused by traffic in the highway during that period of time. Figure 11 (b) refers to August 7, when access to the Olympic Sports Center of Beijing was free of charge. There is a loop where *a* denotes the origin on the way to the park and later, after the event, *d* becomes a destination of the traffic leaving the park. The experiments were performed using a dataset with GPS trajectories generated by 33.000 taxis during 6 months in 2009. The distance travelled by the taxis is greater than 800 million km.

The main challenge covered in (Yoon, Zheng et al. 2010) is the recommendation of smart itineraries to new travelers based on historical users travel routes and experiences. The aim is to extract relevant information about the locations and the duration of visits from GPS historical data. This work shows how to: (i) create a Location-Interest Graph which is generated offline, containing location-related information (location, interest, stay time, travel time, classical travel sequence), and (ii) define a good itinerary and (iii) how to evaluate and compare recommendations. The solution relies on a density-based algorithm (OPTICS) to cluster *stay points* into unique locations (density regions). The *stay points* are geographical regions where a user stayed at least during a certain minimum time interval. The dataset consists of 17.745 GPS samples from 175 users. Figure 12 illustrates the architecture of this system.

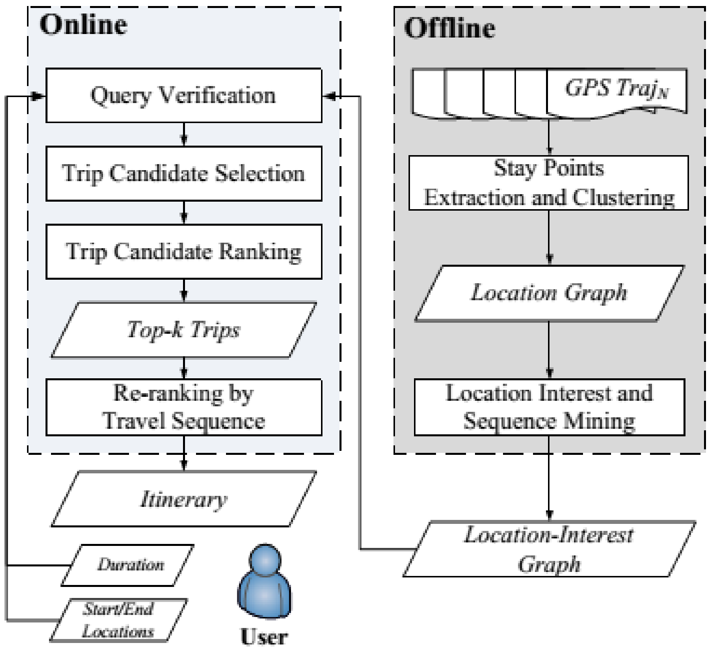


Figure 12: Architecture of smart itinerary recommender. (Yoon, Zheng et al. 2010)

The offline tasks are time-consuming, for example, the analysis of user GPS log files (static information) and the Location-Interest Graph creation. The online tasks, refer to the processing of user queries using the pre-processed offline Location-Interest Graph for the top-k recommendations.

In (Li 2010) another framework for trajectories is presented, which is based on micro-clusters and macro-clusters. The micro-clusters are used to store compact summaries of similar sections of trajectories (line segments) for the purpose of space reduction. The definition of a micro-cluster trajectory is an extension of the cluster feature vector in BIRCH algorithm. By applying this concept, the authors try to solve the clustering problem in terms of computational effort versus cluster time processing. So, they proposed the TCMM (an incremental Trajectory Clustering using Micro and Macro clustering) has the following steps: (i) trajectories are partitioned into line segments in order to find the sub-trajectory clusters; (ii) the micro clusters of partitioned trajectories are computed and maintained incrementally; (iii) micro clusters are used to generate the macro-clusters. Figure 13 illustrates the process, which starts with trajectories simplification followed by the micro-clustering and the macro-clustering tasks. The final result (macro-cluster), shows the representative trajectory derived from micro-clusters line segments.

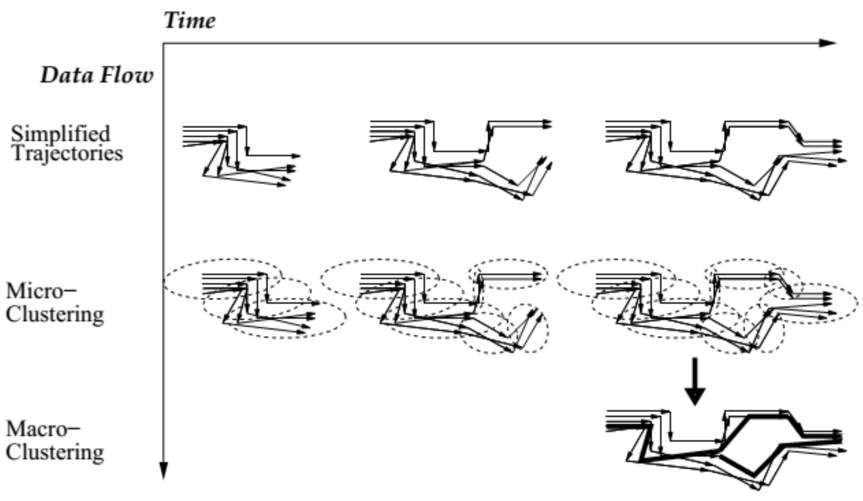


Figure 13: TCMM framework. (Li 2010)The representative line segment from the cluster is computed in each one micro-cluster. The distance function is adapted from a similarity measure used in pattern recognition (Chen, Leung et al. 2002) also used in (Lee, Han et

A similar approach is followed in (Li, Han et al. 2004), where micro-clustering is applied to detect spatio-temporal variations of moving objects in a very large datasets. The concept used is analogous to micro-clustering in BIRCH (Zhang, Ramakrishnan et al. 1996), but it is extended to handle moving micro-clusters that, according to (Li, Han et al. 2004), are a group of moving objects not only similar in terms of spatial (closeness or proximity) and temporal aspects (similar at the same time), but must also have similar movements during a period of time. This solution also uses a *K-Means* algorithm in the experimental evaluation to select objects with similar behavior. The evaluation was made using a collection of synthetic datasets.

2.1.7 Delaunay Triangulation

Delaunay Triangulation (Delaunay 1934) is well known in computational geometry and consists in creating triangles from a set of points in the Euclidean space, where none of this points are inside the circumcircle of any triangle. Figure 14 depicts an example of a DT, with the triangles and all the circumcircles centers (points in red).

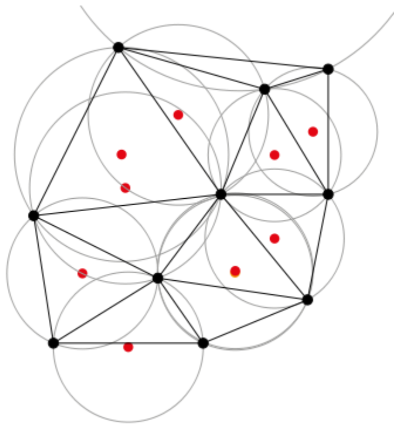


Figure 14: Delaunay Triangulation example.

The solutions proposed in (Guo, Liu et al. 2010) and (Guo 2007) use DT to create clusters of trajectories.

The main challenge, in (Guo, Liu et al. 2010) is to find patterns in trajectories datasets. The authors consider a set of trajectories as a complex network and focus on the analysis of vehicle movements. The first step consists in aggregating GPS points to reduce the number of positions. This step is carried out by creating a DT and the search of neighbors is performed in the DT, followed by the identification of representative points. This step is called the 'smooth phase' and each trajectory is interpolated using the extracted representative points. The clustering is performed using a modified shortest distance measure. Figure 15 shows an example where 112.203 GPS points are reduced to 12.029 representative points.

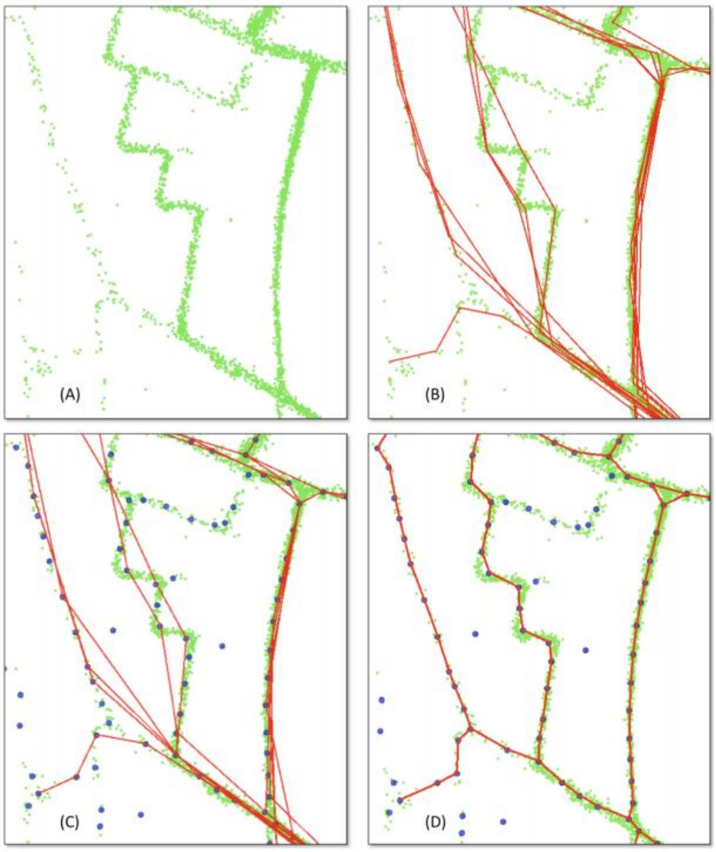


Figure 15: (A) The GPS data points in the map. (B) A selection of five trajectories represented on the map. (C) The representative points (blue color). (D) Approximated representative trajectories. (Guo, Liu et al. 2010)

Figure 16 displays the density of trajectories at different times.

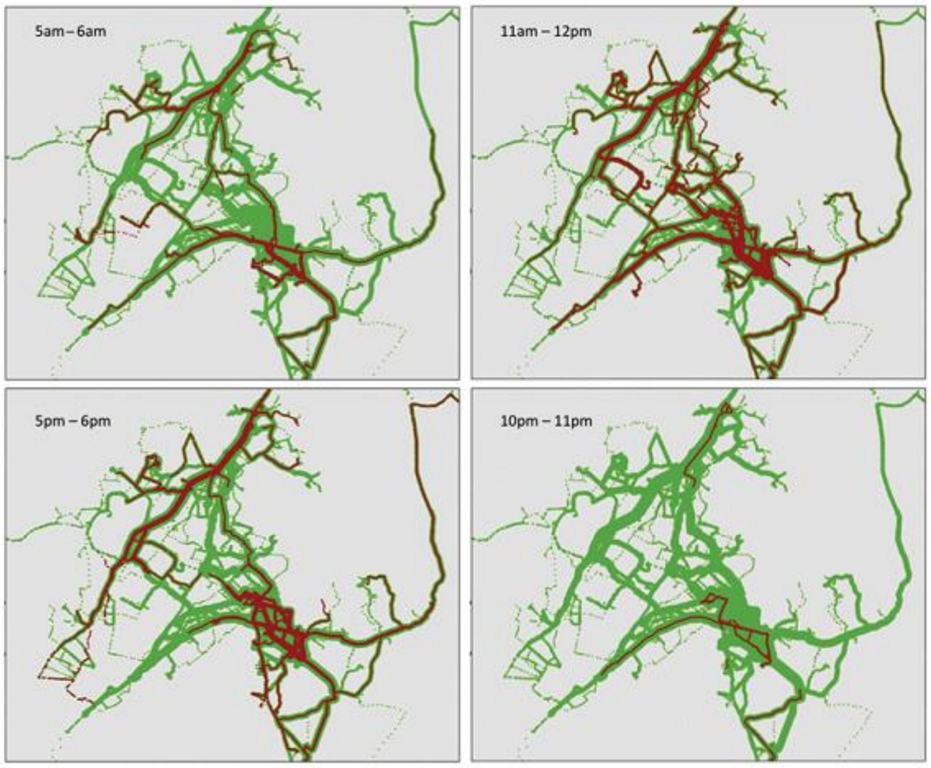


Figure 16: Density trajectories at different timestamps. (Guo, Liu et al. 2010)

To obtain the results in Figure 16, a graph based on the extracted representative points (first step) and the interpolated trajectories (second step), is constructed as follows. The representative points are nodes and they are linked when they correspond to consecutive nodes belonging to the same trajectory. The edges between the nodes represent the weight, i.e., the total number of trajectories that they share. Knowing this, a spatially constrained graph partitioning method (Guo 2009) is applied to discover a hierarchy of regions, where locations inside a region share trajectory connections with each other. The clustering of trajectories is considered to be region-based. Figure 17, represents two regions of clustered trajectories.

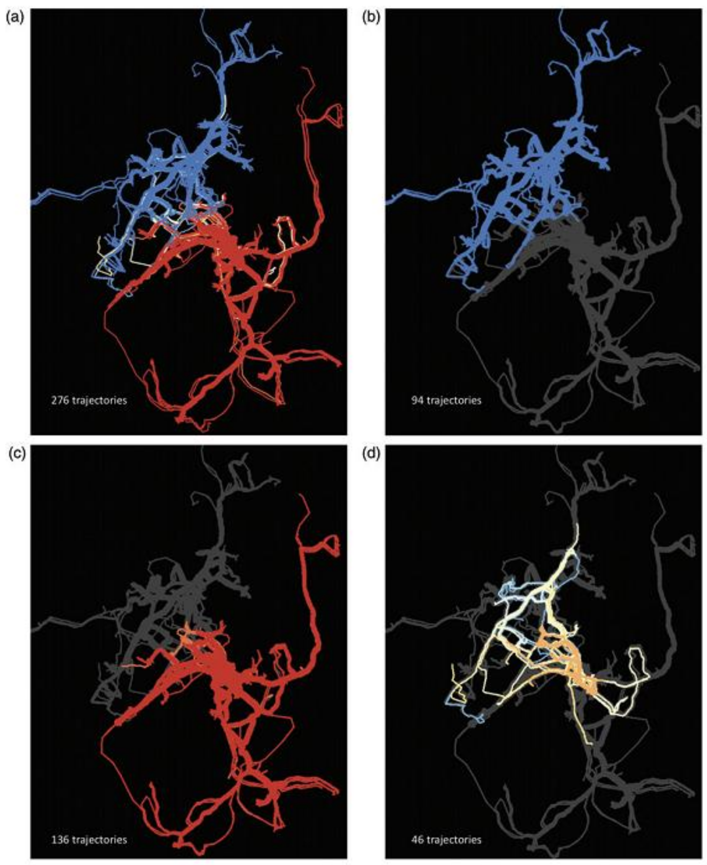


Figure 17: (a) Two clusters of trajectories. (b) The blue cluster has 94 trajectories (c) the red cluster contains 136 trajectories, mainly in the south region. (d) 46 trajectories are in both regions. (Guo, Liu et al. 2010)

2.2 Trajectory Similarity and Distance Measures

This section presents solutions to determine the similarity between trajectories, focusing on (1) free movements in Euclidean Space and (2) network-constrained movements, where similarity is applied in a spatial network context, usually represented by a graph (Tiakas, Papadopoulos et al. 2006), (Mahrsi 2013), as depicted in Figure 18.

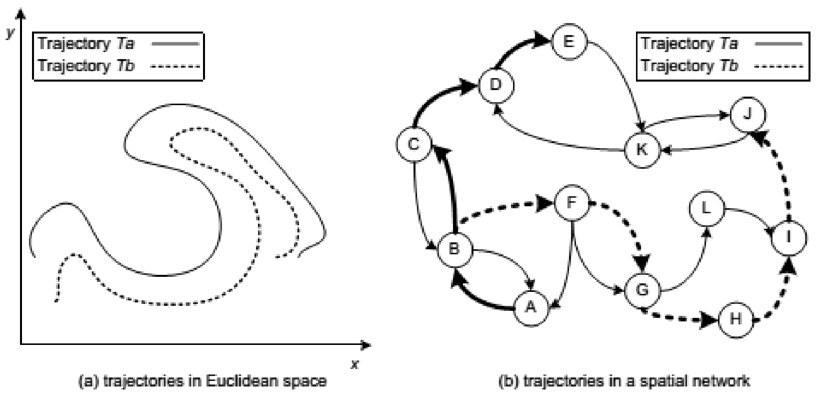


Figure 18: Representations of trajectories in unconstrained and constrained contexts. (Tiakas, Papadopoulos et al. 2006)

The similarity between moving objects in a roads network space is gaining relevance, but so far, few studies have been developed in that domain. On the other hand, several works exist concerning the similarity of moving object trajectories in the Euclidean space.

2.2.1 Distance function for time series on free moving objects

A time series $S = [s_1, s_2, \ldots, s_n]$ is defined as a sequence of real values, where each value s_i is a sample at a specific timestamp and n is the length of S (Chen and Ng 2004). This sequence is called the raw representation of the time series. Trajectories can be treated as time series data as well, but it is necessary to change the distance functions to deal with multidimensional values.

The definition of distance function between trajectories based on similarity measures raises several issues (Chen, Tamer et al. 2005) (Vlachos, Gunopoulos et al. 2002): (i) The presence of outliers or noise, caused by sensors anomalies, due to GPS positions precision, or even due to human failures. These issues have an impact in the clustering phase, because similar trajectories in real world may be considered dissimilar. (ii) Different sampling rates, i.e., the time interval between observations tends to be constant, but failure in sensors can cause 'shifts' in data and trajectories shapes will be different from the expected. (iii) Similar motions in different space regions, i.e., the movement of two objects can be similar but they can differ in the space they move (Vlachos, Gunopoulos et al. 2002).

In most cases, the similarity between trajectories is focused on shape, i.e., it considers the spatial attributes but it ignores the temporal dimension (Chen, Tamer et al. 2005). Although, the similarity between trajectories can be classified into the following categories (Dodge, Weibel et al. 2009): (i) spatial similarity: the focus is on the similarity of the geometric shape of moving objects trajectories; (ii) temporal similarity: it is usually implemented using Time Wrapping Distance (TWD) to retrieve similarity between two trajectories (Yi, Jagadish et al. 1998) (Yanagisawa, Akahani et al. 2003); (iii) spatial-temporal similarity: spatio-temporal and some dynamic features, like speed, are taken into account; (iv) semantic similarity: trajectories are associated with one or more semantic tags. These semantic tags can denote airports, restaurants, shops, hotels, etc. in the form of $(\mathrm{x1}, \mathrm{y1}, \mathrm{t1}, \mathrm{S1})$, where $(\mathrm{x}_1, \mathrm{y}_1)$ is a position, t_1 is a timestamp and S_1 is a semantic tag (Liu and Schneider 2012).

The Euclidean Distance [5] is often used to compare trajectories, but it has several drawbacks. Formula 4, describes how to calculate the distance between two points, where n denotes the number of dimensions. The trajectories must have the same length, as depicted in Figure 19, where the first position of trajectory $\lambda 1$ is compared with first position of trajectory $\lambda 2$, and so on. This measure is also sensitive to noise. Hence, some similarity measures were studied in the past years to address these issues: *Edit Distance* (ED) (Wagner and Fischer 1974), *Edit Distance with Real Penalty* (ERP) (Chen and Ng 2004), *Dynamic Time Warping* (DTW) (Chen, Gu et al. 2013), (Senin 2008) and (Muller 2007), *Longest*

[5] Source: en.wikipedia.org/wiki/Euclidean_distance

Common SubSequences (LCSS) (Vlachos, Gunopoulos et al. 2002), and finally the *Edit Distance on Real sequence* (EDR) (Chen, Tamer et al. 2005).

$$d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \quad (4)$$

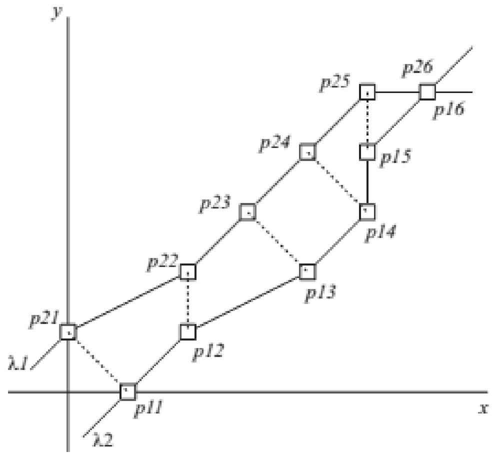
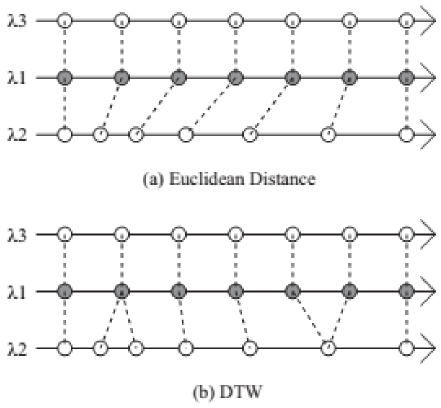
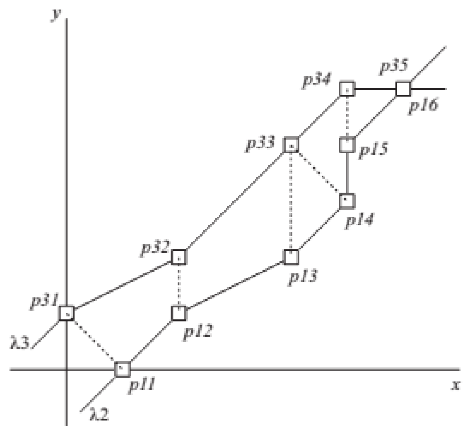


Figure 19: Euclidean distance. (Yanagisawa and Satoh 2006)

The Edit Distance (ED) is an old method used to calculate the minimum cost of all sequences of edit operations (substitution, insertion and deletion) required to transform one string into another. In moving objects trajectories context, ERP and EDR were developed based on ED. ERP supports time shifting and it is a metric distance function (*L1-norm*), but like in Euclidean distance it is vulnerable to noise. On the other hand, EDR is more robust, because it can reduce the effect of noise. In (Yuan and Raubal 2012), the authors used a modification of Edit Distance algorithm to measure trajectories similarity of mobile phone users.

The DTW (Yi, Jagadish et al. 1998), is a distance function usually applied in speech recognition (Sakoe and Chiba 1990), gesture recognition, handwriting matching, surveillance (Zhang, Huang et al. 2006), (Berndt and Clifford 1994), as well as in moving objects trajectories (Chen, Gu et al. 2013), to compare and align two sequences. It is not mandatory to have trajectories with the same length and this technique is also capable of handling time shifting, but it is sensitive to noise like ERP and Euclidean distance. Figure 20, illustrates the comparison between Euclidean Distance and DTW and shows that DTW can deal with trajectories of different lengths.





(a) (b)

Figure 20: (a) Difference between Euclidean distance and DTW; (b) DTW. (Yanagisawa and Satoh 2006)

The LCSS (Yanagisawa and Satoh 2006), (Zhang, Huang et al. 2006), is another variant of the ED used to match two sequences and then to define the distance of sub-sequences. This method do not requires rearranging the sequence of the elements and allows elements to be unmatched, i.e., it only considers the matched sub-sequences and ignores the unmatched portions of the trajectories. This similarity measure is applied on video tracking trajectories (Vlachos, Gunopoulos et al. 2002), surveillance (Buzan, Sclaroff et al. 2004), trajectory semantics (Liu and Schneider 2012) and it is not vulnerable to noise, but the fact of being non-metric difficult its usage in clustering algorithms. This method cannot hold the Triangle Inequality[6] (TI). The TI theorem states that the sum of the lengths of two edges of a triangle must be greater or equal than the length of the remaining edge (Formula 5). In the Euclidean geometry the edges of the triangle are vectors**.**

$$\|x+y\| <= \|x\| + \|y\| \tag{5}$$

Distance functions robust to noise, normally violate the triangle inequality theorem.

According to (Chen, Tamer et al. 2005), EDR needs three pruning techniques to hold the triangle inequality. In addition, this is the most accurate and robust similarity measure with or without noise. LCSS can deal with the existence of gaps in the sequences defining the shapes to be compared and EDR assigns penalties according to the sizes of the gaps. Table 2, shows the differences among the five similarity measures, according to what the measures are able to handle and the computational costs of each one.

[6] Sources: en.wikipedia.org/wiki/Triangle_inequality, mathworld.wolfram.com/TriangleInequality.html

	Support			Computation cost
	Local Time Shifting	Sensitive to Noise	Metric	
Euclidean distance	No	Yes	Yes	$O(n)$
ERP	Yes	Yes	Yes	$O(m * n)$
EDR	Yes	No	Yes (pruning techniques)	$O(n^2)$
DTW	Yes	Yes	No	$O(n^2)$
LCSS	Yes	No	No	$O(n^2)$

*m and n are the length of the two trajectories.

Table 2: Overall of the similarity measures. (Chen, Tamer et al. 2005)

The next example depicts the ranking differences among five similarity measures: L1-norm, EDR, ERP, DTW and LCSS. Figure 21, illustrates a simple time series Q, and other five time series ($T1 - T5$) with time shifting and noise. Time series $T1$ is shifted from Q, $T2$ has noise in position 4, and the others are also variations of the initial one.

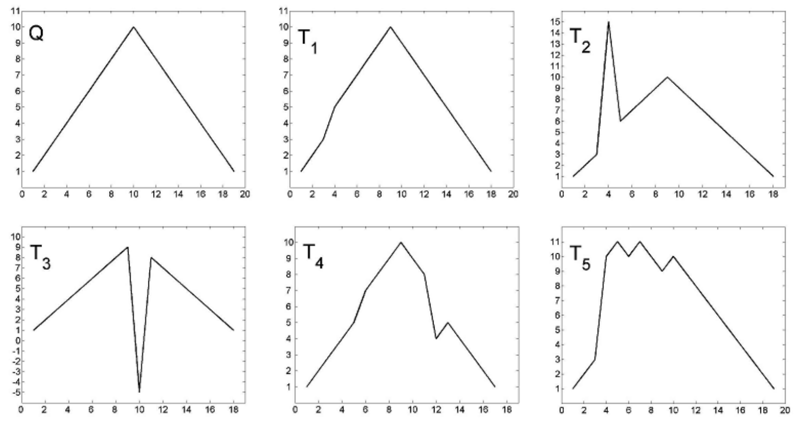


Figure 21: Example of time-series for ranking. (Chen and Ng 2004)

The rankings according to the similarity with Q (defined as the time series reference) it is as follows:

L1-norm: $T1, T4, T5, T3, T2$

ERP: $T1, T2, T4, T5, T3$

EDR: $T1, \{T2, T3\}, T4, T5$

DTW: $T1, T4, T3, T5, T2$

LCSS: $T1, \{T2, T3, T4\}, T5$

L1-norm is sensitive to noise data and considers $T2$ as the worst match. LCSS focus only on the matched parts of the time series and ignores the unmatched parts, and considers $T5$ as the worst match. EDR ranked $T2$ and $T3$ equally, higher than $T4$, which the authors consider more similar to Q than $T3$. DTW is sensitive to noise but not to local shifts and $T2$ is the worst ranked. According to the authors, ERP is the most "natural" order (Chen and Ng 2004).

2.2.2 Similarity of Trajectories in Spatial Networks

Network-constrained moving objects are studied in (Hwang, Kang et al. 2005), (Xia, Wang et al. 2010), (Chang, Bista et al. 2007), (Tiakas, Papadopoulos et al. 2006). This has become a hot research topic, because in many real applications, moving objects move in road networks rather than in the Euclidean space. Notable examples are vehicles (road networks) and train (railroad networks) but the same may also hold when considering invisible air and sea routes. A road network is modelled by a graph $\mathrm{G(V, E)}$, where E denotes the set of edges representing the smallest unit of a road segment, and V is the set of vertices (nodes) representing road junctions (intersections).

The Euclidean distance is an inappropriate measure to apply to road networks, because a distance is a sum of roads lengths between two positions in the network. Figure 22 shows that the distance between positions a and b, using the Euclidean distance is 4 km, but in reality, the sum of the road segments to travel from a to b is 9 km.

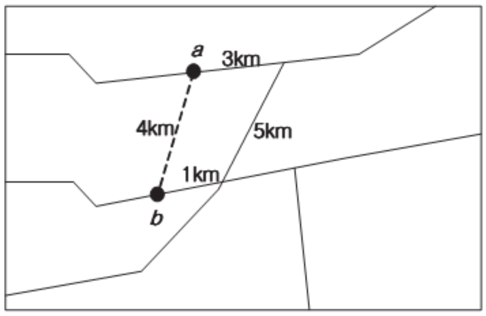


Figure 22: Distance example between Euclidean space and network space. (Hwang, Kang et al. 2005)

In addition, the Euclidean distance considers only spatial similarities and do not takes advantage of spatial-temporal aspects (Hwang, Kang et al. 2005). Figure 23 shows an example of trajectories (TR_A, TR_B, TR_C) in three-dimensional space $\mathrm{(x, y, t)}$. The trajectories TR'_A, TR'_B, TR'_C are projected only in two-dimensional $\mathrm{(x, y)}$. The trajectories TR_A and TR_B have an equal trace, but the temporal components are different. If a two-dimensional $\mathrm{(x, y)}$ space is considered, we can conclude that TR_A and TR_B are similar to each other. However, considering the temporal and spatial dimensions TR_A and TR_C are closer than TR_B.

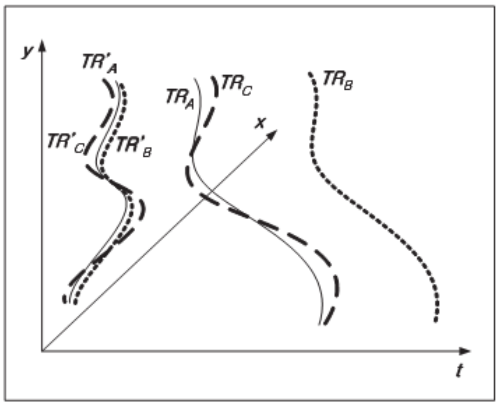


Figure 23: Spatio-Temporal trajectories. (Hwang, Kang et al. 2005)

In (Hwang, Kang et al. 2005) three methods to search for similar trajectories are proposed: (i) using the spatio-temporal distance between trajectories; (ii) filtering trajectories based on temporal similarity and refining similar trajectories based on spatial distance; (iii) filtering trajectories based on spatial similarity and refining similar trajectories based on temporal distance. The conclusion is that the best method is third one, which is based on the *Hausdorff* Distance (HD), defined as:

$$\mathrm{dist}_H(\mathrm{l},\mathrm{m}) = \max_{a\in l}\{min_{b\in m} dist(a,b)\} \tag{6}$$

where l and m are curves and $dist(a,b)$ is the distance between two points. However, this distance measure proved to be a poor choice, because it classifies trajectories such as those depicted in Figure 24 as similar: $\mathrm{dist}_H(TR_A, TR_B) = \mathrm{dist}_H(TR_C, TR_D) = d$. The distances are determined using the pair of points p and q.

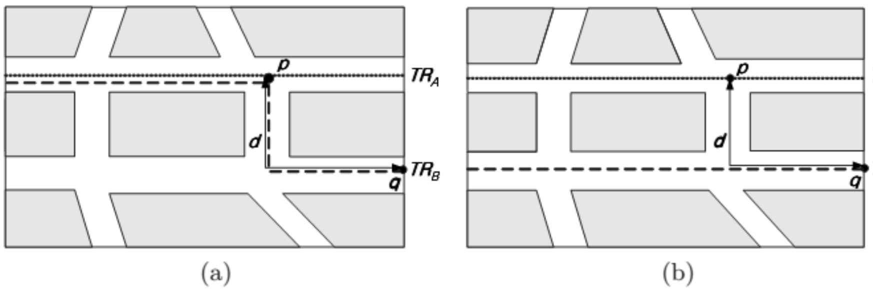


Figure 24: Hausdorff distance d on road network; (a) TR_A and TR_B; (b) TR_C and TR_D. (Hwang, Kang et al. 2005)

Hence, the authors have proposed a similarity measure based on the concept of POI (Points of Interest) instead of using HF, where two trajectories are considered similar if they cross the same POI. The measure is defined as the difference between the timestamps when two objects have crossed the same POI. The temporal distance between TR_A and TR_B for a set of POIs is as follows:

$$\mathrm{dist}_T(TR_A, TR_B, P) = L_p(TR_A, TR_B, p) = (\sum_{i=1}^{k} |p_i(TR_A) - p_i(TR_B)|^p)^{\frac{1}{p}} \tag{7}$$

where P is the set of POIs, $p \in P$ and k is the number of POIs.
The algorithm has two phases: (i) the filtering phase based on spatial similarity and (ii) refining phase to search for similar trajectories based on a temporal distance.

In (Tiakas, Papadopoulos et al. 2006), it is proposed a similarity model for trajectories in spatial networks, based on the distance between trajectories. It is assumed that the trajectories have the same length and are represented using a graph $G(V, E)$. A trajectory T is defined as sequence (Formula 8), where v_m denotes a node in the graph, t_m is the time instance and m is the trajectory length:

$$T = \big((v_1, t_1), (v_2, t_2), \dots, (v_m, t_m)\big) \tag{8}$$

The similarity of two trajectories is not directly related with the number of shared nodes, but with the cost associated to each transition between nodes in trajectories, i.e., the similarity measure must consider the trajectories proximity. The distance is based on a transition cost between two nodes (Formula 9), where the $c(v_s, v_d)$ represents the cost to travel from a source node v_s to a destination node v_d.

$$d(v_s, v_d) = \begin{cases} 0, c(v_s, v_d) = 0 \wedge c(v_d, v_s) = 0 \\ \dfrac{min\{c(v_s, v_d), c(v_d, v_s)\}}{max\{c(v_s, v_d), c(v_d, v_s)\}}, otherwise \end{cases} \tag{9}$$

Formula 10, gives the network distance between two trajectories T_a and T_b , where m is the length of both trajectories.

$$D_{net}(T_a, T_b) = \frac{1}{m} \cdot \sum_{i=1}^{m} \big(d(v_{ai}, v_{bi})\big) \tag{10}$$

However, time information is also important in traffic analysis and so, it is necessary to consider the time required to travel from one node to the next one (Formula 11).

$$D_{time}(T_a, T_b) = \frac{1}{m-1} \cdot \sum_{i=1}^{m-1} \frac{|(T_a[i+1].t - T_a[i].t) - (T_b[i+1].t - T_b[i].t)|}{max\{(T_a[i+1].t - T_a[i].t), (T_b[i+1].t - T_b[i].t)\}} \tag{11}$$

Formula 12 combines the two measures using weighs.

$$D_{total}(T_a, T_b) = W_{net} \cdot D_{net}(T_a, T_b) + W_{time} \cdot D_{time}(T_a, T_b) \tag{12}$$

Trajectories of different lengths are decomposed into sub-trajectories.

The solution proposed in (Xia, Wang et al. 2010) starts with the reconstruction and partition of raw trajectories to suppress useless positions in data and to reduce storage requirements. It uses *feature points* (Lee, Han et al. 2007) to perform the partitioning of the trajectories into sub-trajectories. Then, a spatio-temporal similarity measure is used, followed by a normalization, to classify the similarity as 0 (non-similar trajectories) or 1 (similar trajectories). The method is based on *Jaccard similarity coefficient*, where similarity between trajectories is calculated as the ratio of the common parts divided by the summation of the common and uncommon parts, as follows:

$$Sim(TR_i, TR_j) = \frac{L_c(TR_i, TR_j)}{L(TR_i) + L(TR_j) - L_c(TR_i, TR_j)} \tag{13}$$

The term $L_c(TR_i, TR_j)$ is the total length of the common part, between TR_i and TR_j, $L(TR_i)$ represents the length of the trajectory, which can be spatial or temporal. The authors compare their method with (Tiakas, Papadopoulos et al. 2009) in order to find similar trajectories, considering spatial and temporal features. The authors search for similarity in terms of temporal and spatial, i.e., spatio-temporal neighbors. Figure 25, shows an example of three clusters of trajectories by considering spatial and temporal features, cluster A only has spatial similar trajectories. Nevertheless, cluster A could be refine into cluster 1 and also cluster 2 due to temporal similarity, it demonstrates, trajectories in cluster 1 are spatio-temporal similar.

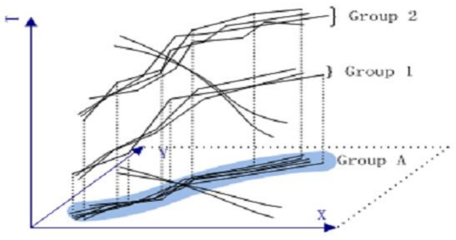


Figure 25: Temporal and spatial attributes of trajectories. (Xia, Wang et al. 2010)

The Formula 14 assumes that spatial and temporal features have the same impact on the similarity between trajectories. The result of this formula is between $[0 - 1]$, i.e., if the result is 1, the two trajectories overlap in the spatio-temporal dimensions.

$$STSim(TR_i, TR_j) = SSim(TR_i, TR_j) \times TSim(TR_i, TR_j) \tag{14}$$

$SSim$ refers to spatial and $TSim$ to temporal similarities. The proposed method proved to be effective and capable to reduce the use of memory.

2.3 Comparison of clustering algorithms

In (Liu, Deng et al. 2012) it is presented, a summary of spatial clustering algorithms (Table 3). This table puts in evidence the major issues in spatial clustering and the main solutions proposed in the literature. The authors conclude that any algorithm can't handle efficiently all the clustering problems.

Algorithm	Problems				
	Discovery of arbitrary shaped clusters	Discovery of clusters with uneven density	Robust to noise	Heavily reliant on prior knowledge	Consideration of both spatial proximity and attribute similarity
k-means	X	X	X	✓	X
CLARANS	X	X	✓	✓	✓
Single-link	✓	X	X	✓	X
Complete-link	X	X	X	✓	X
CURE	X	X	✓	✓	X
BIRCH	X	X	✓	✓	X
AMEOBA	✓	✓(partial solution)	X	X	X
CHAMELEON	✓	✓(partial solution)	X	✓	X
DBSCAN	✓	X	✓	✓	X
GDBSCAN	✓	X	✓	✓	✓
OPTICS	✓	✓(partial solution)	✓	✓	X
DENCLUE	✓	X	✓	✓	X
DBRS	✓	X	✓	✓	✓
ADACLUS	✓	✓	✓(partial solution)	X	X
MST	✓	✓(partial solution)	✓(partial solution)	✓	X
AUTOCLUST	✓	✓(partial solution)	✓(partial solution)	X	X
2-MSTClus	✓	✓(partial solution)	X	✓	X
STING	✓	X	✓(partial solution)	✓	X
WaveCluster	✓	X	✓	✓	X
EM	X	X	✓(partial solution)	✓	X
Geo-SOM	X	X	X	✓	✓

ICC	X	X	X	X	✓
CSM	✓	✓	✓	✓	X

Table 3: A comparison of spatial clustering algorithms. (Liu, Deng et al. 2012)

(Mahrsi 2013) is a recent work based on graph structures oriented towards sampling moving objects datasets and clustering network-constrained trajectories and road segments that also presents a comparison of trajectory clustering algorithms considering constrained (networked) and unconstrained (free) moving objects. Table 4, lists the different approaches in trajectory clustering.

Approach	Dimensions		Clustering granularity	Clustering type	Network-constrained
	Space	Time			
(Li, Han et al. 2004)	Yes	Yes	moving objects	Partitioning	No
(Jensen, Lin et al. 2007)	Yes	Yes	moving objects	Hierarchical	No
(Kalnis, Mamoulis et al. 2005)	Yes	Yes	moving objects	density-based	No
(Vieira, Bakalov et al. 2009)	Yes	Yes	moving objects	-	No
(Jeung, Shen et al. 2008)	Yes	Yes	moving objects	density-based	No
(Lee, Han et al. 2007)	No	Yes	segments	density-based	No
(Guo, Liu et al. 2010)	No	Yes	Data points	Graph-based	No
(Nanni and Pedreschi 2006)	Yes	Yes	trajectories	density-based	No
(Pelekis, Kopanakis et al. 2009)	No	Yes	trajectories	fuzzy-clustering	No
(Kharrat, Popa et al. 2008)	No	Yes	road segments	density-based	Yes
(Kharrat, Popa et al. 2009)	Yes	Yes	road segments	density-based	Yes
(Roh and Hwang 2010)	No	Yes	trajectories	Hierarchical	Yes

Table 4: Comparison of different trajectory clustering approaches.(Mahrsi 2013)

This comparison shows that existing methods are mainly focused on unconstrained moving objects trajectories (e.g., hurricanes, vehicles, hiking, bicycle) and few studies deal with clustering network-constrained (e.g., cars) trajectories. (Guo, Liu et al. 2010) is an exception, which presents a graph-based algorithm rather than a density-based algorithm as in most existing works making it a reference work in graph-based approaches.

2.4 Summary

This chapter presents a literature review on trajectories clustering. It starts with basic concepts on clustering and presents the main algorithms proposed for clustering spatial data. Next, an overview on clustering spatio-temporal data is presented followed by a presentation of solutions for trajectories clustering. Several techniques to determine similarity and distance measures between trajectories are presented. It also puts in evidence the particularities of existing solutions when dealing with network-constrained and unconstrained trajectories.

Research on network-constrained trajectories is less extensive than for unconstrained trajectories, and most works use only synthetic datasets using the Thomas Brinkhoff (Brinkhoff 2002) generator to simulate the movement of cars in road networks. The solutions for dealing with unconstrained trajectories are generic and may also be applied to network-constrained trajectories. However, the modelling of trajectories using an underlying graph may reduce search space and improve the performance and the quality of results.

3. Chapter

Methodology

This chapter presents the methods and the application developed in this work for trajectory clustering and the recommendation of routes between two given locations.

Google maps offers a panoply of features, enabling the user to easily plan driving routes from a point A to a point B. It provides multiple routes choices and the option to choose the desired means of transport. Moreover, in the last years, it also includes real-time traffic information. To provide this functionality, Google has integrated *Waze (Shinar 2009)*, a crowd-sourced traffic data system. Our work focuses on a similar problem, but we only consider the user's historical trajectory data generated by GPS devices. The aim is be able to provide personalized route recommendations using for instance, collaborative filtering techniques. For that purpose, this work focuses on three important sub-problems: trajectory clustering, outlier detection and creating representative trajectories for clusters.

The case study is based on a real dataset tracking the movement of trucks in Athens, Greece[7]. We have only considered a sample composed of five hundred trajectories representing the movement of 25 trucks during 33 days (Figure 26).

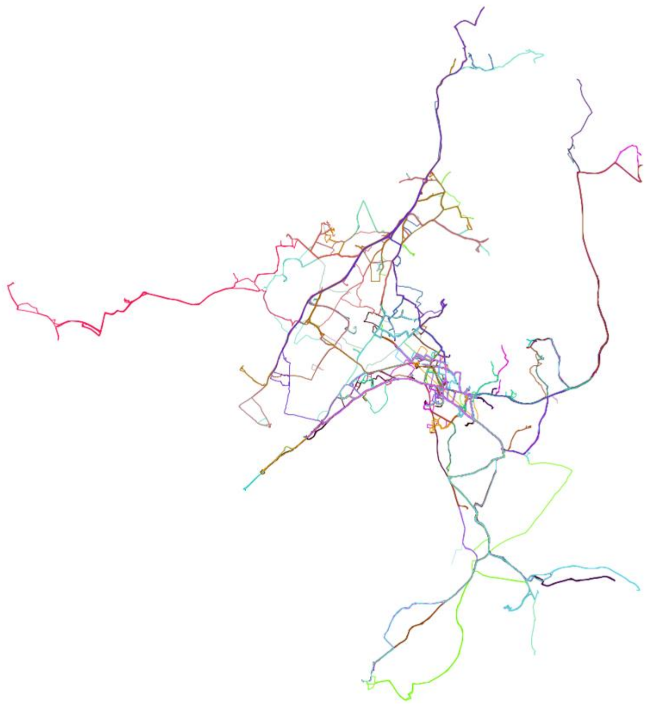


Figure 26: The five hundred trajectories derived from 25 moving objects.

[7] Source: "The Greek Trucks Dataset", www.chorochronos.org

3.1 Problem Formulation

The goal is to develop a framework for recommending routes between a given source and destination using GPS historical data. It is assumed that objects (cars, bicycles, ships, etc.) move freely in space and so this framework should be able to deal with network-constrained and unconstrained trajectories. This requires implementing methods to find which are the representative trajectories between source and destination positions and defining the criteria to recommend one or several alternative (ranked) trajectories (routes) for a specific moving object.

The inputs of the system are the origin and destination positions, and GPS log files. The GPS log files have date and time (timestamp), latitude and longitude coordinates (the WGS84 reference is the standard for GPS data), but some datasets also include coordinates represented using other reference systems, such as UTM or GGRS87.

In the following, it is considered that

Trajectory: a trajectory $T_i = \{p_{i,1}, p_{i,2}, \dots, p_{i,n}\}$, is an ordered sequence of time stamped positions in a two-dimensional Euclidean space, where i denotes a moving object and n is the number of GPS samples. Each point $p_{i,k} = (x_{i,k}, y_{i,k}, t_{i,k})$ corresponds to a GPS sample, such that $(x_{i,k}, y_{i,k})$ represents the coordinates (latitude and longitude) and $t_{i.k}$ is a timestamp.

Line Segments: Line segments are used in the clustering phase and are represented as $RS_{seq} = (p_{i,k}, p_{i,k+1})$, with $1 \leq k \leq n-1$ and seq is an index denoting the order of the line segment in a trajectory or sub-trajectory. This definition is based on the concept of *trajectory partitions* proposed in (Lee, Han et al. 2007). This means that each trajectory is split into line segments such that $T_i = \{RS_{1,} RS_{2,} \dots, RS_{n-1,}\}$.

Cluster: Let $C = \{C_1, C_2, \dots, C_m\}$ denote a set of clusters, where m is the number of clusters. Each cluster $C_j = \{T_1, T_2, \dots, T_p\}$ represents a set of similar sub-trajectories, where p is the number of similar trajectories in a cluster.

Representative Trajectory: A representative trajectory is an approximation or "average" of all trajectories belonging to a cluster, i.e., $R_t = \{p_{r,1}, p_{r,2}, \dots, p_{r,k}\}$, where each $p_{r,k}$ denotes a position (vertex). There are several alternatives to estimate the position of these vertices, e.g., they can be *centroid positions* of an agglomeration of neighbor positions.

As an example, Figure 27 illustrates four trajectories, where the origin is the blue circle and the destination is the red circle. These are the trajectories of four moving objects that have crossed the origin and destination regions within a certain time interval. The trajectory 23 puts in evidence the partitioning of a trajectory into line segments. The clustering result should be two clusters: $C_1 = \{23\}$ and $C_2 = \{28, 29, 30\}$.

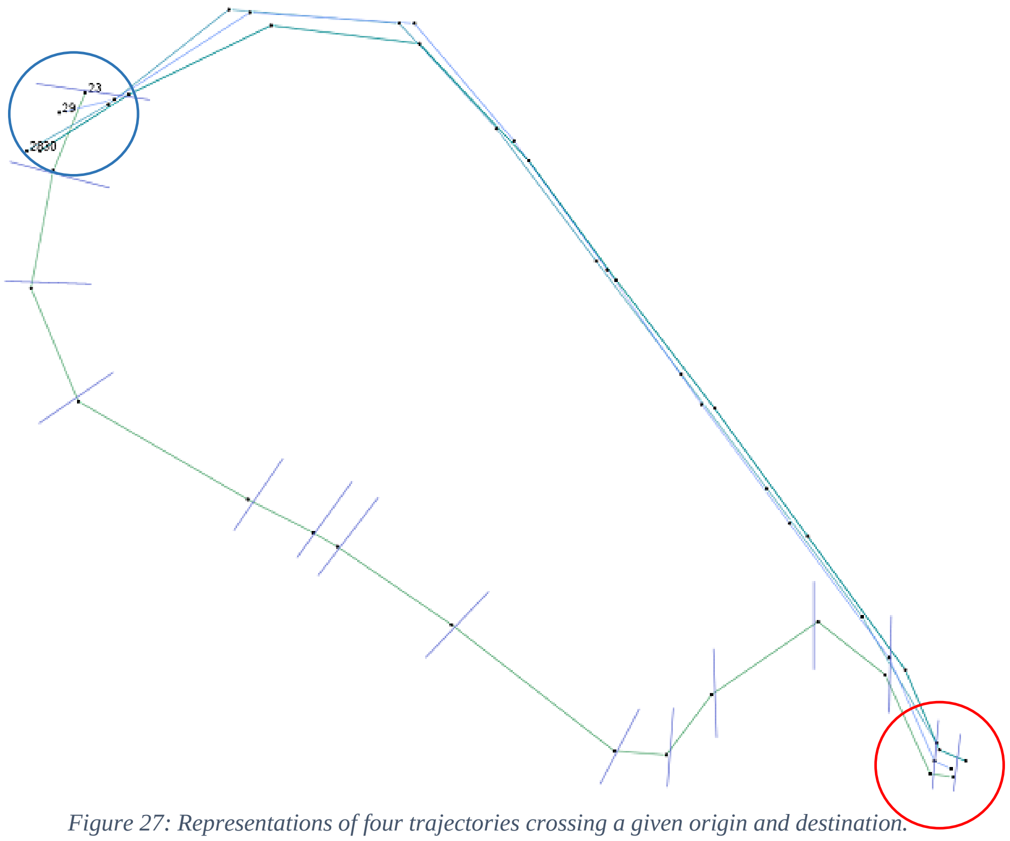


Figure 27: Representations of four trajectories crossing a given origin and destination.

3.2 Implementation

The implementation of this work is organized in four main steps (Figure 28): (1) Trajectories Filtering, (2) Trajectories Clustering, (3) Cluster Representative Trajectory and (4) Trajectories Recommendation.

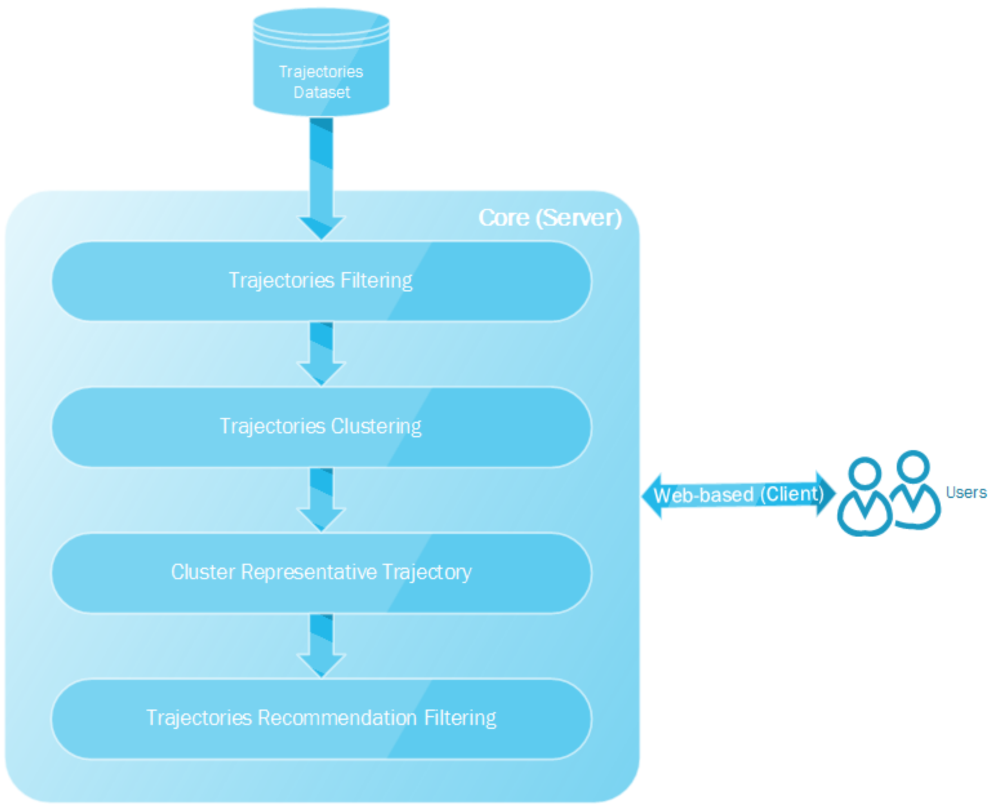


Figure 28: High-level conceptual model.

3.2.1 Trajectories Filtering Step

Before proceeding to trajectory clustering, it is necessary to read the data from the available data source. The source information can be stored in a CSV text file, a PostgreSQL (*PostGIS*[8]) Spatial database, etc. and it is usually organized as a set of GPS positions. So, it is required to define the trajectories in a GPS dataset.

GPS datasets can be easily ordered by object identification and by time. The movement of an object along time can be decomposed into trajectories by scanning the time dimension. In this work it is assumed that each trajectory corresponds to trip. Thus, each trajectory consists of the set GPS positions for which the elapsed time between consecutive samples is less or equal than a given threshold. This means that when this constraint does not hold it assumed that the moving object has stopped, e.g., for a work journey, and the trip ends. It is also necessary to look for inconsistent data to avoid problems such as 'teleportation', i.e., consecutive GPS positions that are miles away from each other. Then, it is necessary to select the set of trajectories T, that have crossed the origin (O) and the destination (D), as illustrated in at the top of Figure 29.

[8] Source: postgis.net/

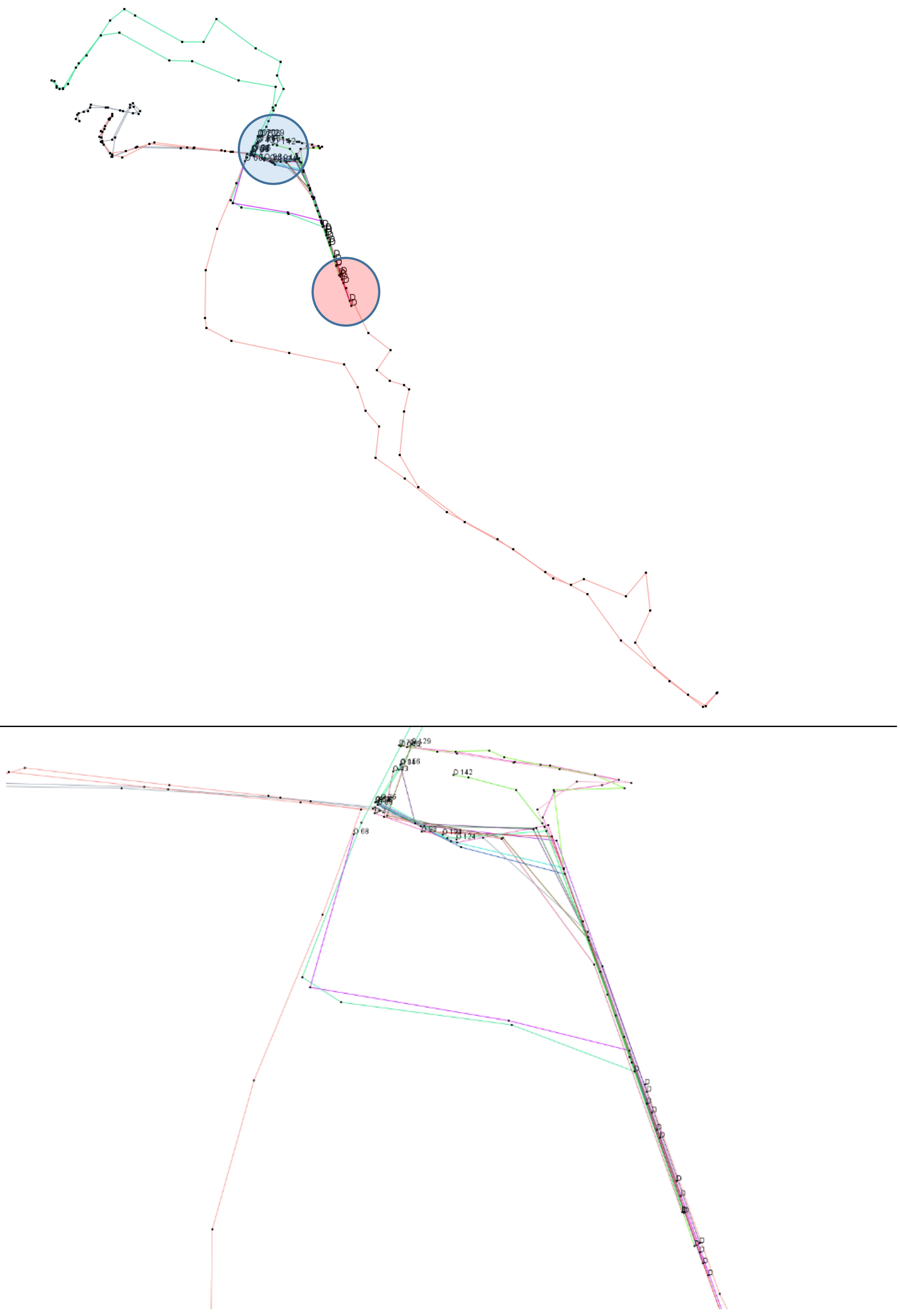


Figure 29: Set of trajectories crossing an origin and a destination.

The selection of these trajectories is performed as follows:

- Select the origin (O), destination (D) and the radius of the circles (d_r) centered at O and D.
- Select the sub-trajectories that intersect the circle centered at O and D. The sub-trajectories of interest are those including at least a line segment (RS_{seq}) that intersects the circle centered at

O and a line segment that intersects the circle centered at D. These distances include RS_{seq} positions to O and D and an aditional perpendicular distance defined as $d_{\perp}(RS_{seq}, P_{O\ or\ D}) \leq d_{r\ thd}$.

- Clip the selected sub-trajectories by deleting all line segments that do not belong to the path between O and D. The sub-trajectories have the same direction from O to D.

The output of this step is a set of sub-trajectories that will define the possible routes between O and D as defined in Section 3.1.

3.2.2 Trajectory Clustering Step

This sections describes the distance function used in similarity calculations and the clustering algorithm.

3.2.2.1 Distance Function

The distance function used in this work comprises three distance measures adapted from (Chen, Leung et al. 2002). All these distances are geometric measures. To the best of our knowledge, the first work to use these measure distances with trajectories is (Lee, Han et al. 2007). Therefore, the clustering step implemented in this work was based on these two works. We have made a small modification by adding intermediate points (the blue dots in Figure 30) to the line segments to improve results when the elapsed time between GPS samples is large.

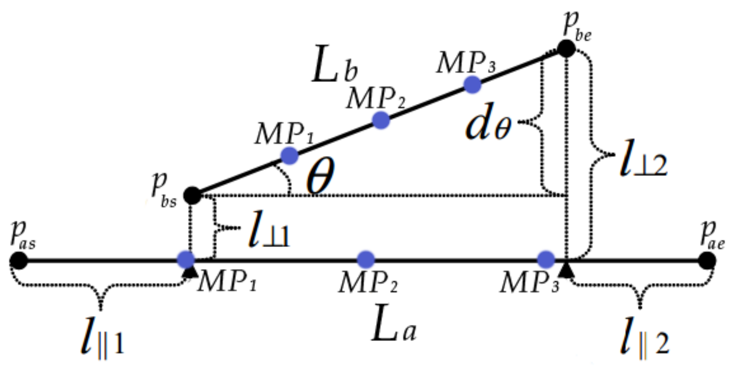


Figure 30: The three measures types between two line segments with the corresponding intermediate points.

The three measures to estimate the similarity between two line segments $L_a = (p_{as},\ p_{ae})$ and $L_b = \{p_{bs},\ p_{be}\}$ are: (i) angle distance (d_θ) (ii) perpendicular distance $(d_\perp)$ and (iii) parallel distance $(d_\parallel)$.

***Perpendicular distance*:** The perpendicular total distance is calculated using the *Lehmer Mean* (LM) concept, which consists of a set of n numbers $(x_k)_{k=1}^{n}$ defined as $(x) = \frac{\sum_{k=1}^{n} x_k^p}{\sum_{k=1}^{n} x_k^{p-1}}$. A reference line segment is chosen and the positions are extracted (start, end and intermediate positions) for comparison with others line segments. Formula 15 defines a generic perpendicular distance between a point and a line segment:

$$l_{\perp_i}(p_{L_a}, L_b) = \frac{\left|(y_e - y_s)x_p - (x_e - x_s)y_p - x_s y_e + x_e\, y_s\right|}{\sqrt{(x_e - x_s)^2 + (y_e - y_s)^2}} \qquad (15)$$

where $L_b = \{(x_s, y_s), (x_e, y_e)\}$ is the line segment to be compared and p_{L_a} denotes a starting, ending or intermediate position from line segment L_a.

The perpendicular total distance (Formula 16) is then the summation of all the perpendicular distances between the positions in L_a and L_b:

$$d_{\perp_{\boldsymbol{Total}}}(L_a, L_b) = \frac{\sum_{i=1}^{n}(l_\perp(p_{L_a}, CL_b)_i)^2}{\sum_{i=1}^{n}(l_\perp(p_{L_a}, CL_b)_i)} \qquad (16)$$

where n is the number line segment positions in L_a.

Angle distance: The angle distance is the angle between two line segments L_a and L_b. Formulas 17, 18 and 19 show how to compute the angle distance in degrees.

$$m1 = \frac{(y_{ae} - y_{as})}{(x_{ae} - x_{as})} \qquad (17)$$

$$m2 = \frac{(y_{be} - y_{bs})}{(x_{be} - x_{bs})} \qquad (18)$$

$$\theta = \tan^{-1}\left(\left|\frac{(m2 - m1)}{(1 + m2 * m1)}\right|\right) \times \frac{180}{\pi} \quad degrees \qquad (19)$$

Formula 20 defines the total angle distance between L_a and L_b, where $||L_b||$ denotes the length of L_b.

$$d_{\theta_{\boldsymbol{Total}}}(L_a, L_b) = \begin{cases} ||L_b|| \times \sin(\theta), & if\ 0^\circ \leq \theta < 90^\circ \\ ||L_b||, & if\ 90^\circ \leq \theta \leq 180^\circ \end{cases} \qquad (20)$$

Parallel distance: The parallel distance is the minimum distance between L_a and L_b and is given by Formula 21.

$$d_{\parallel_{\boldsymbol{Total}}}(L_a, L_b) = MIN(l_{\parallel_1}, l_{\parallel_2}) \qquad (21)$$

The measures $l_{\parallel_1}$ and $l_{\parallel_2}$ are the distances at the start and the end of L_a and L_b (Figure 31). Formulas 22 and 23 represent the distances for each line segment.

$$l_{\parallel_1} = \sqrt{(x_{as} - x_{bs})^2 + (y_{as} - y_{bs})^2} \tag{22}$$

$$l_{\parallel_2} = \sqrt{(x_{ae} - x_{be})^2 + (y_{ae} - y_{be})^2} \tag{23}$$

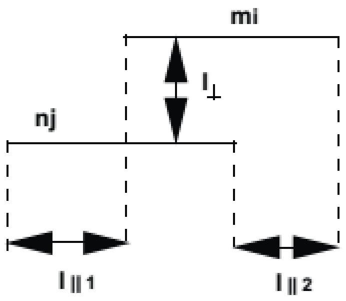


Figure 31: Line distance measures. (Chen, Leung et al. 2002)

Total distance: The total distance is a weighted average of the three distance measures mentioned above (Formula 24).

$$w_{\parallel} = 1, \qquad w_{\theta} = 1, \qquad w_{\perp} = 1$$
$$d(L_a, L_b) = d_{\parallel Total} \times w_{\parallel} + d_{\theta Total} \times w_{\theta} + d_{\perp Total} \times w_{\perp} \tag{24}$$

By default, the weights are equal to 1, but the last test case in this work, the parallel distance is set to 0.5, because if two line segments have very different of lengths, this measure has a strong effect on the distance function value.

3.2.2.2 Clustering Algorithm

The clustering algorithm is based on the similarity between trajectories, which is calculated using Formula 24. After the execution of the filtering step we have the sub-trajectories which share the same origin (O) and destination (D). This section shows how to perform a synchronized traversal of the line segments to compute the similarity between two trajectories and how to implement the clustering algorithm.

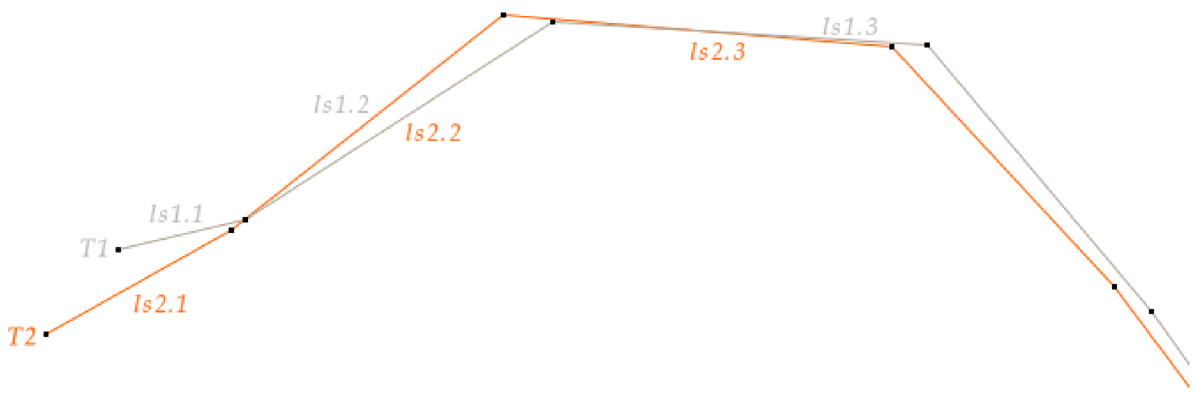


Figure 32: Two similar trajectories with the corresponding line segments.

Synchronized Traversal (ST): The synchronized traversal is a sequential process. The algorithm starts with the two first line segments ($ls1.i$ and $ls2.j$, with $i = j = 1$) of two trajectories or sub-trajectories Ta and Tb (analogous to Figure 32, $T1$ and $T2$), and proceeds as follows:

- Compute the total distance between $ls1.i$ and $ls2.j$. If the total distance (d_{ST}) is less than a given threshold then $ls1.i$ and $ls2.i$ are considered similar, and the process continues;
- To proceed for the next line segments comparison a verification is performed. Is necessqry to chose which of the previous compared line segments is greater. For example, trajectories $T1$ and $T2$ (Figure 32), after the similarity calculation, if both line segments are similar, the algorithm verifies both line segments length and choose the greater one, if $ls1.i > ls2.j$ then add 1 to i, else add 1 to j. In this example, $ls1.1$ is chosen, and will be compared with the next line segment from $T2$, the $ls2.2$.

Clustering: Algorithm 1 is a simplified description of the clustering of trajectories.

Algorithm 1 Clustering trajectories

```
Input: $T$: trajectory set; $d_{ST}$: similarity distance threshold
Output: $C$: set of clusters
1: $T_{OtoD}$ = searchForRoutesOtoD($r_{distance}$, $T$)
2: while exists $T_{OtoD}$ not assigned to $C$ do
3:  create inicial cluster $c_l = \{T_i\}$
4:  for $T_i \in T$ do
5:     for $T_j \in T$ do
6:        get line segment $ls_a$ and $ls_b$ from $T_i$ and $T_j$
7:        if sim($ls_a$, $ls_b$) <= $d_{ST}$ then
8:           add line segment to provisory list
9:           check which line segment end position is greater
10:            increment ls
11:       else
```

```
12:         $increment\ T_j;\ break;$
13:     end if
14:   end for
15:   if all line segments from $T_j$ $are$ in provisory list then
16:     $assign\ T_j\ to\ c_l$
17:   else
18:     $assign\ T_j\ to\ new\ c_n$
19:   end if
20:   $increment\ T_i$
21: end for
22:end while
```

The inputs are a set of trajectories and d_{ST} a similarity distance threshold used to check if two line segments are similar. If two line segments are not similar, or an *outlier* is detected, we stop the line segment similarity process (*line 12*). Otherwise, the line segment is inserted into a provisory list and the loop continues. At the end of the loop, *line 15* checks whether all line segments have been inserted into the provisory list. If so, the trajectory or sub-trajectory is inserted into an existing cluster, otherwise, a new cluster is created.

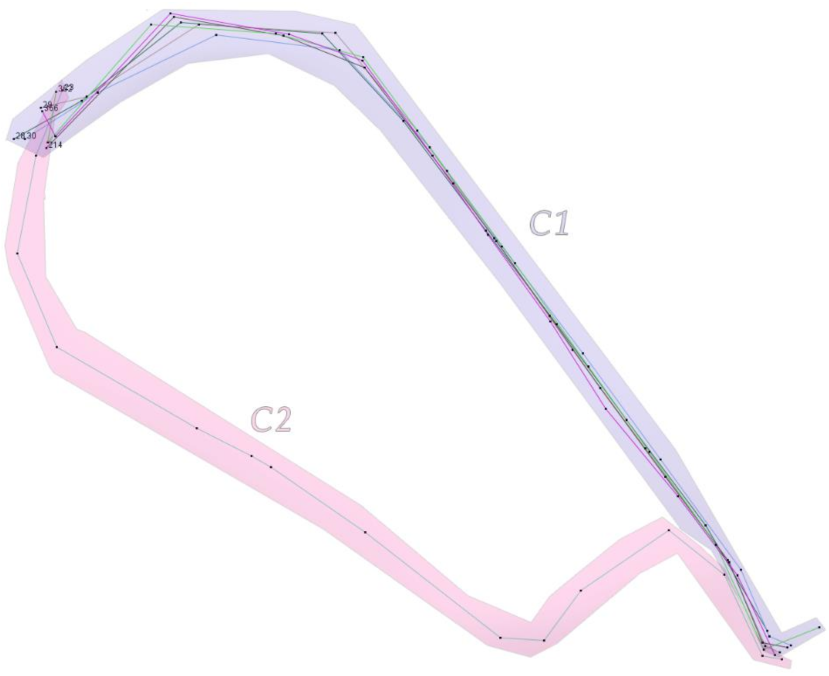


Figure 33: Representation of two visually clusters.

Figure 33 illustrates a set of trajectories $T = \{T_{23}, T_{28}, T_{29}, T_{30}, T_{214}, T_{362}, T_{366}\}$ that can be grouped in two clusters, $C1$ and $C2$. However, our algorithm 1, discovered three clusters, $C1 = \{28, 29, 30, 214, 366\}$, $C2 = \{23\}$ and $C3 = \{362\}$. This occurs due to the algorithm sensitivity to noise and outliers. The authors of (Lee, Han et al. 2007) describe two types of outliers: *positional* and *angular*. *Positional* refers to location while the *angular* is associated with the direction. Our definition of outlier detection is different from other works in the literature, because our goal is to find similar trajectories and not outliers. If algorithm 1 detects an outlier in the similarity process, the sub-trajectory is not associated to the cluster, and a new cluster is created.

3.2.3 Representative Trajectory Step

Finding the representative trajectory of a cluster is a challenge. The representative trajectory should represent the overall movement of the trajectories that belong to a cluster. The solution implemented in this work is based on *representative GPS points extraction* as proposed in (Guo, Liu et al. 2010). This method searches for representative points using a circular window to aggregate GPS positions. The representative points are the *centroids* of the circular windows. The radius of the circles must be fixed in advance. This method also uses a *moving-window smoothing process* based on a Delaunay Triangulation.

The purpose of this feature is to have a unique trajectory representation and retrieve useful information from it, e.g., length, time travel or average speed. This method has two steps: (i) positions connectivity extraction via *Delaunay Triangulation* and (ii) *centroid* representation. These steps are presented in the following sections.

3.2.3.1 Positions Connectivity Extraction

The goal of the Positions Connectivity Extraction (PCE) step is to obtain the positions connections from DT. This step is illustrated in Figure 34 where all trajectories belong to the same cluster. The gray circles represent regions of neighbor positions. The DT is first constructed in a lower level, i.e., instead of using line segments, we construct the DT on the GPS positions directly.

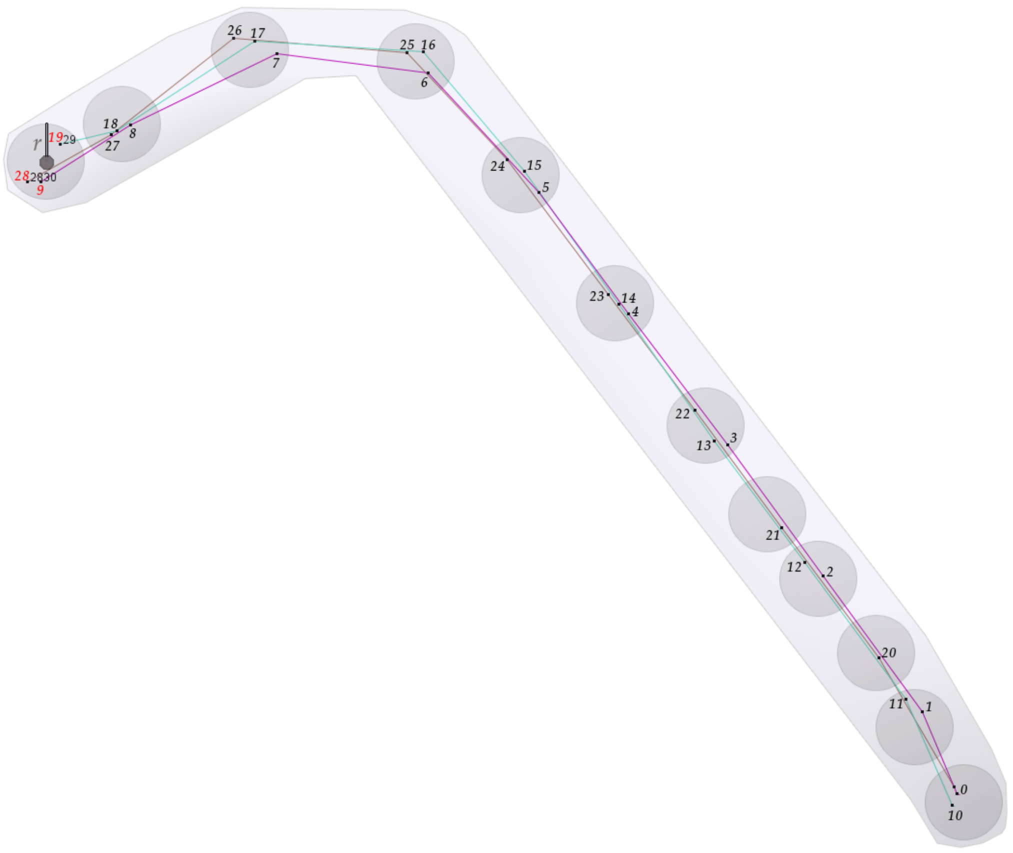


Figure 34: Representation of regions of neighbor positions.

The DT was implemented with the help of a two-dimensional constrained DT library[9]**,** developed in java, which is based on the algorithm of V. Domiter and B. Zalik (Domiter and Zalik 2008). This library allowed us to construct the DT for each cluster based on the trajectories detected on the clustering phase. Figure 35 depicts the triangulation based on the cluster of Figure 34.

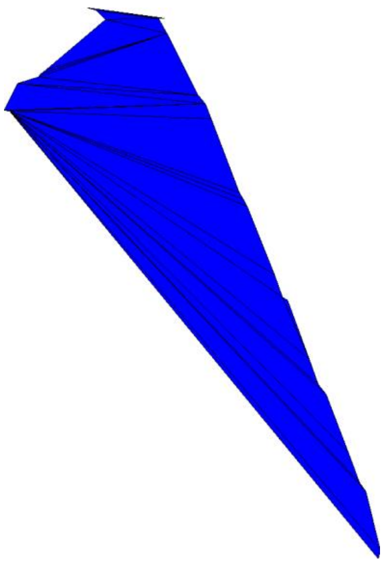


Figure 35: Delaunay Triangulation of the cluster.

[9] Source: code.google.com/p/poly2tri

Table 5 shows the DT triangles with the related positions vertices for the cluster illustrated in figure 35:

$\Delta 1 = [16, 17, 26]$	$\Delta 7 = [27, 6, 18]$	$\Delta 13 = [22, 9, 4]$	$\Delta 19 = [20, 9, 2]$
$\Delta 2 = [25, 17, 16]$	$\Delta 8 = [24, 19, 27]$	$\Delta 14 = [13, 9, 22]$	$\Delta 20 = [11, 9, 20]$
$\Delta 3 = [7, 25, 17]$	$\Delta 9 = [24, 27, 6]$	$\Delta 15 = [21, 13, 3]$	$\Delta 21 = [1, 9, 11]$
$\Delta 4 = [6, 7, 25]$	$\Delta 10 = [23, 9, 5]$	$\Delta 16 = [21, 9, 13]$	$\Delta 22 = [0, 9, 1]$
$\Delta 5 = [8, 6, 7]$	$\Delta 11 = [14, 9, 23]$	$\Delta 17 = [12, 9, 21]$	$\Delta 23 = [10, 0, 9]$
$\Delta 6 = [18, 6, 8]$	$\Delta 12 = [4, 9, 14]$	$\Delta 18 = [2, 9, 12]$	

Table 5: Triangles resulted from DT.

Then we search for vertices connections (Table 6) presents the result:

$v0 = [9, 1, 10]$	$v9 = [15, 28, 5, 23, 14, 4, 22, 13, 21, 12, 2, 20, 11, 1, 0, 10]$	$v20 = [9, 2, 11]$
$v1 = [9, 11, 0]$	$v10 = [0, 9]$	$v21 = [13, 3, 9, 12]$
$v2 = [9, 12, 20]$	$v11 = [9, 20, 1]$	$v22 = [9, 4, 13]$
$v3 = [21, 13]$	$v12 = [9, 21, 2]$	$v23 = [9, 5, 14]$
$v4 = [9, 14, 22]$	$v13 = [9, 22, 21, 3]$	$v24 = [19, 27, 6, 15]$
$v5 = [9, 15, 23]$	$v14 = [9, 23, 4]$	$v25 = [17, 16, 7, 6]$
$v6 = [7, 25, 8, 18, 27, 24]$	$v15 = [19, 24, 28, 9, 5]$	$v26 = [16, 17]$
$v7 = [25, 17, 6, 8]$	$v16 = [17, 26, 25]$	$v27 = [6, 18, 24, 19]$
$v8 = [6, 7, 18]$	$v17 = [16, 26, 25, 7]$	$v28 = [15, 19, 9]$
	$v18 = [6, 8, 27]$	
	$v19 = [24, 27, 15, 28]$	

Table 6: The vertices connections.

The neighbors are obtained from these vertices connections and this step is explained in detail in the following section.

3.2.3.2 Centroid Position Representation

The Centroid Representation Position (CRP) is the centroid a group of neighbors and it is represented by a *centroid*. This *centroid* will be part of the representative trajectory of the cluster. This methods is divided in two phases: (i) finding neighbors closeness and (ii) *centroid* assignment.

The neighbors closeness step uses the vertices connections from DT to determine which positions are neighbors. This step allows to reduce the search space and so, improve the computational performance of the algorithm

The next step is the *centroid* assignment, which computes the centroid of neighbor positions based on (Guo, Liu et al. 2010).

3.2.3.2.1 Neighbors Closeness

For the closeness step, a neighbors distance threshold (d_{thv}) should be previously defined. The value used in this work is $d_{thr} = 30$ meters. According to (Guo, Liu et al. 2010), this threshold is the error range for GPS datasets. The result of this method for the case illustrated in (table 6) is shown in table 7.

$group1 = [0, 10]$	$group10 = [20]$
$group2 = [1, 11]$	$group11 = [21]$
$group3 = [2, 12]$	$group12 = [22]$
$group4 = [3, 13]$	$group13 = [23]$
$group5 = [4, 14]$	$group14 = [15, 24]$
$group6 = [5, 15]$	$group15 = [6, 16, 25]$
$group7 = [7]$	$group16 = [17, 26]$
$group8 = [8, 18]$	$group17 = [18, 27]$
$group9 = [19]$	$group18 = [9, 28]$

Table 7: Groups of neighbor vertices.

3.2.3.2.2 Centroid Assignment

The centroid assignment is performed for neighbors, i.e., vertices that are close to each other. However, we found that if we check the distance between the pre-calculated *centroid* groups, we can reduce even more the number of groups and thereby improve the representative trajectory. Table 8 shows the result achieved:

$group_{C1} = [9, 19, 28]$	$group_{C7} = [3, 13, 22]$
$group_{C2} = [8, 18, 27]$	$group_{C8} = [21]$
$group_{C3} = [7, 17, 26]$	$group_{C9} = [2, 12]$
$group_{C4} = [6, 16, 25]$	$group_{C10} = [20]$
$group_{C5} = [5, 15, 24]$	$group_{C11} = [1, 11]$
$group_{C6} = [4, 14, 23]$	$group_{C12} = [0, 10]$

Table 8: Groups of representative centroids.

The number of groups was reduced from 18 to 12 and the *centroids* are represented in Figure 36. This gives the representative trajectory.

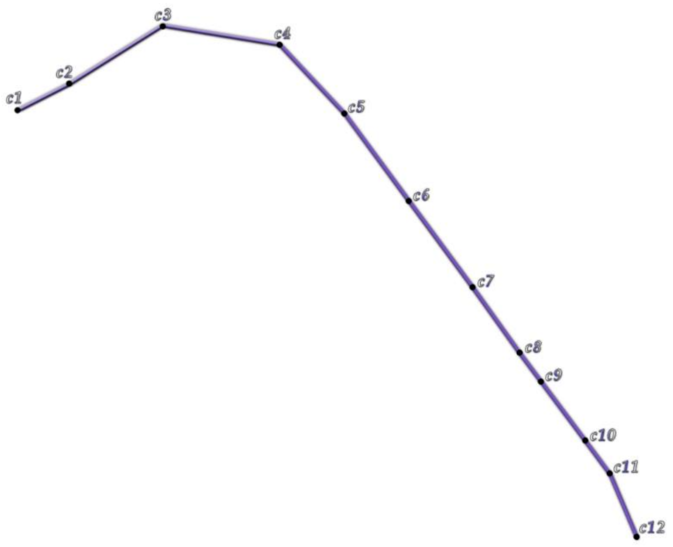


Figure 36: Representative Trajectory.

3.2.3.3 Representative Trajectory Algorithm

Algorithm 2 describes the procedure that joins the previous two phases to obtain the clusters representative trajectories. The input parameters are triangles vertices and a distance threshold. The algorithm iterates for each vertex in the Delaunay Triangulation. The first step (line 2) finds the vertices in the Delaunay Triangulation that are connect to the vertex being processed. Then, the distance between the current vertex and the connected vertices is evaluated (Line 4) to select those that are closer than the given threshold. Next, it is calculated the centroid of the current and selected vertices, and the result is stored into a temporary array (Line 8). Finally, line 11 to 17 simplify the representative trajectory, by checking each pre-calculated centroid to find if there are other candidates for centroids nearby, using also the initial distance threshold. If so, the centroid is merged, i.e., it is computed a new centroid.

```
Algorithm 2 Trajectory Representative
Input: $vTri$: $triangles\ vertices$  $d_{thv}$: neighbors $distance\ threshold$
Output: $CT$: set of representative centroids
1:  for each $vertice\ v \in vTri$ do
      /* Positions Connectivity Extraction */
2:    $vConnections = searchForVerticeConnectionsFromDT(v)$;
3:    for each $connection_{vertice} \in vConnections$ do
      /* Neighbour Closeness: the connected vertice distance to the current vertice $v$ */
4:      if $(d_{connection} \leq d_{thv})$ then
5:        $assign\ connection_{vertice}\ to\ neighbour_{proximityList}$
6:      end if
7:    end for
8:    $centroid = computeCentroid(neighbour_{proximityList})$;
9:  $assign\ centroid\ to\ CT$;
```

```
10: end for
        /* Centroid assignment */
11: for $each$ $pair\ of\ centroid \in CT$ do
12:   $check\ for\ centroids\ proximity$
13:   if $(centroids_{dist} \leq d_{thv})$ do
14:     mergeCentroids();
15      $reassign\ centroid\ to\ CT$;
16:   end if
17: end for
```

3.3 Implementation

To demonstrate all the components working together it was built a web application to support user's interaction. This application allows the user to search for trajectories which share the same origin and destination regions, as well as to define the input parameters for the algorithms. This section presents the components and the technologies used to implement this application.

The web application, allows the user to set the parameters required to generate the clustering and representative trajectories. In fact, users submit a query with specific attributes: (i) for the search trajectory step: origin and destination regions and circle radius, (ii) for the clustering step: user establishes a threshold for the line segments similarity, and (iii) for the representative trajectory step: a threshold for the distance between neighbors.

3.3.1 Technical Specifications

The system development is based on Java language, but to create a bridge between the end-user and the system, it was necessary to use several technologies that are divided into three categories: the back-end business logic integration, data visualization and front-end user interaction.

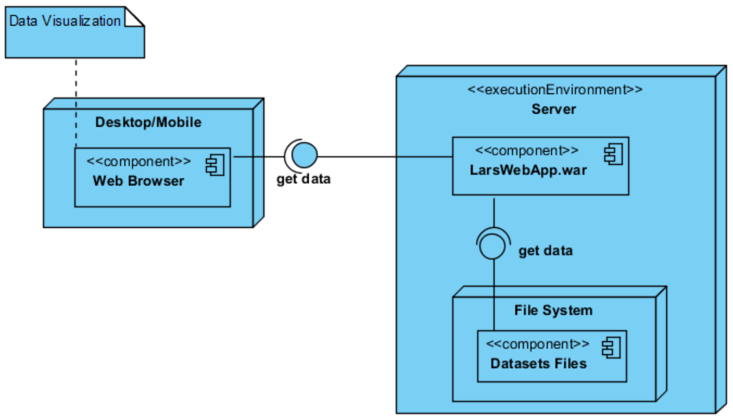


Figure 37: Logical Architecture component diagram.

Figure 37, shows the back-end global logical architecture and components interaction. The diagram consists of a client-side and server-side nodes. Starting with the server-side, the execution environment is based on the open-source application server *GlassFish,* supporting the *Enterprise JavaBeans.* It integrates the *LarsWebApp* web application archive (WAR) and the datasets files. The WAR file contains all the business logic classes and the remaining web resources for the web application. The business logic abstraction is required, to link the core with the other components. Therefore, our back-end is implemented by using *Enterprise JavaBeans* (EJB) to encapsulate the business logic (core).

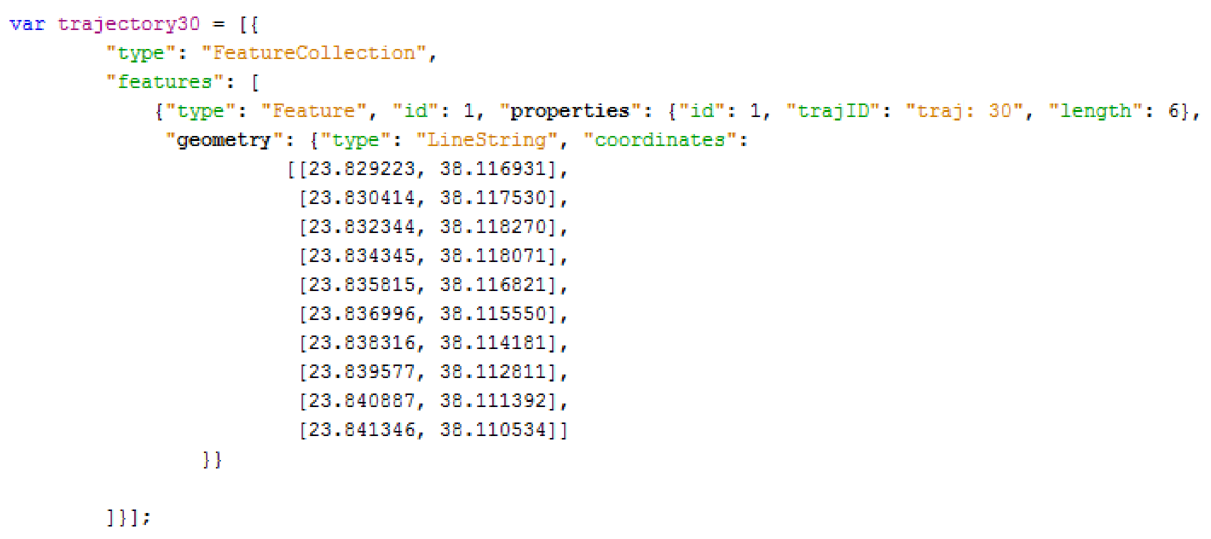
3.3.2 Data Visualization

Data visualization is the means to present the trajectories and clusters on an interactive web map using the World Geodetic System (datum WSG84).

The options considered to implement the spatial data visualization tool were: *ArcGIS*[10], *OpenStreetMap, Foursquare, Google Maps, Yahoo!* and *Bing Maps.* ArcGIS uses a web map powered by ESRI [11] and it is not an open-source solution. Bing and Yahoo! maps use the HERE [12]map platform developed by Nokia, and Google Maps have their own API[13] for geocoding and routing. *Foursquare* integrates an open-source web map application named *MapBox*[14] which allows designing (with *TileMill* studio) maps and customizations. *OpenStreetMap*[15] is a collaborative project developed by a voluntary community which contributes the growth of roads mapping and is fully open-source.

Finally, our choice was an interactive map library called *Leaflet*[16], which is an open-source *JavaScript* library for user interaction providing several features such as map customization, mobile support and the ability to integrate different layers. Apart from the described functionalities, the *Leaflet* community has developed several interesting plugins which can be used freely. This library gained importance due to its versatility in the manipulation of vector layers to present complex spatial data and due to *GeoJSON*[17] integration. The latter, is an extension of *JSON* and is gaining importance in the context of web mapping.

[10] arcgis.com
[11] esri.com
[12] developer.here.com/
[13] developers.google.com/maps/
[14] mapbox.com
[15] openstreetmap.org
[16] leafletjs.com
[17] geojson.org



```
var trajectory30 = [{
        "type": "FeatureCollection",
        "features": [
            {"type": "Feature", "id": 1, "properties": {"id": 1, "trajID": "traj: 30", "length": 6},
             "geometry": {"type": "LineString", "coordinates":
                       [[23.829223, 38.116931],
                        [23.830414, 38.117530],
                        [23.832344, 38.118270],
                        [23.834345, 38.118071],
                        [23.835815, 38.116821],
                        [23.836996, 38.115550],
                        [23.838316, 38.114181],
                        [23.839577, 38.112811],
                        [23.840887, 38.111392],
                        [23.841346, 38.110534]]
                }}

        ]}];
```

Figure 38: GeoJSON output example structure.

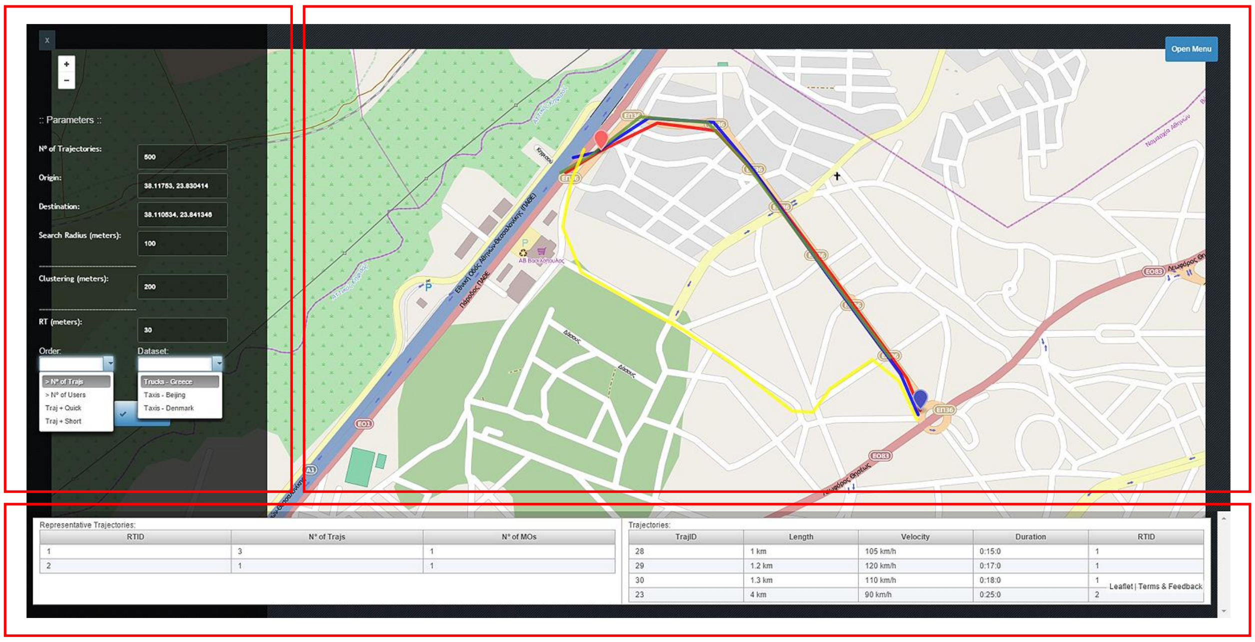
The *GeoJSON* format enables to generate a variety of geometrics types, e.g., points, line string, polygons, etc. Figure 38, shows a collection of positions belonging to a trajectory. In this work, the *GeoJSON* files are used to define the trajectories of the moving objects (trucks) using the line string geometry type, as well as the trajectory identification and length. By default, the Coordinate Reference System (CRS) supported by *GeoJSON* is the WGS84 datum. However, it is possible to define other different CRS, by adding a *crs* object containing a “name” member referencing the CRS in a specific format.

Finally, we use *Leaflet* along with *MapBox* plugin. The *MapBox* is used for displaying layers and controls, and Leaflet for map interaction and data import.

3.3.3 Interface (Front-End)

This section presents the front-end web interface developed using *JavaServer Faces* (JSF) and the user interface components implemented using *PrimeFaces*[18]. This application allows end users to generate trajectory clusters, representative trajectories and to filter inputs. The interface also gives the possibility to enter the desired input values for the different phases of the recommendation. Figure 39 depicts the user interface.

[18] Source: PrimeFaces.org



:: Parameters ::

Nº of Trajectories:	500
Origin:	38.11753, 23.830414
Destination:	38.110534, 23.841346
Search Radius (meters):	100
Clustering (meters):	200
RT (meters):	30

Order: > Nº of Trajs, > Nº of Users, Traj + Quick, Traj + Short

Dataset: Trucks - Greece, Taxis - Beijing, Taxis - Denmark

Representative Trajectories:

RTID	Nº of Trajs	Nº of MOs
1	3	1
2	1	1

Trajectories:

TrajID	Length	Velocity	Duration	RTID
28	1 km	105 km/h	0:15:0	1
29	1.2 km	120 km/h	0:17:0	1
30	1.3 km	110 km/h	0:18:0	1
23	4 km	90 km/h	0:25:0	2

Figure 39: Web Application general structure.

The layout structure is defined by three main containers (highlighted in figure 39): the map in the center is supported by *Leaflet* and contains the two markers, a red marker denoting an origin and a blue marker denoting a destination. The position of the markers can be changed interactively by dragging the marker and the corresponding coordinates in the input text area in the left are updated automatically. Each trajectory cluster has distinct colors. By default, the representative trajectory of each cluster is displayed in black.

The left side container holds the input values: the number of trajectories and thresholds distances for cluster analysis and representative trajectory computation, the radius of the circles centered at the origin and destination for the trajectories filtering step and the other two parameters are distance thresholds for clustering and representative trajectory steps. The origin and destination inputs are obtained by map interaction. The *combo box* allows to choose the desired dataset. The filtering options are mainly based mainly on the Representative Trajectories which can be ranked accordingly to the following options: (i) the Representative Trajectory with the highest number of source trajectories, (ii) the Representative Trajectory with the highest number of moving objects sharing the same route, (iii) the quickest trajectory and (iv) the shortest trajectory.

The two tables in the bottom show detailed information about the trajectories. The table in the left, presents information about the Representative Trajectories: identification, number of trajectories, moving objects and representative trajectory averages (length, duration and velocity), i.e., the average values of all trajectories in a cluster. The table in the right presents information about the source trajectories: identification, length, velocity and duration and the related representative trajectory (RT) identification.

4. Chapter

Results

This chapter presents an evaluation of the method developed in this work under different scenarios. Three test cases are presented using the Trucks dataset. These tests are meant to evaluate the accuracy and the performance of the following steps: trajectories filtering, clustering and representative trajectories creation. It is important to note that the input parameters for the three different steps and the number of trajectories selected have great importance in the performance of the system.

4.1 Accuracy tests

There are specific input parameters for each test case. The weights used to compute the total distance (Formula 24) were ($w_{\parallel} = 1,\ w_{\theta} = 1,\ w_{\perp} = 1$).

4.1.1 Test case one

The input parameters for test case 1 are presented in Table 9. Figure 40 depicts the four trajectories $T = \{T_2, T_{36}, T_{37}, T_{42}\}$ that have traversed the origin (circle in blue) and destination (circle in red), selected in the search phase (filtering step).

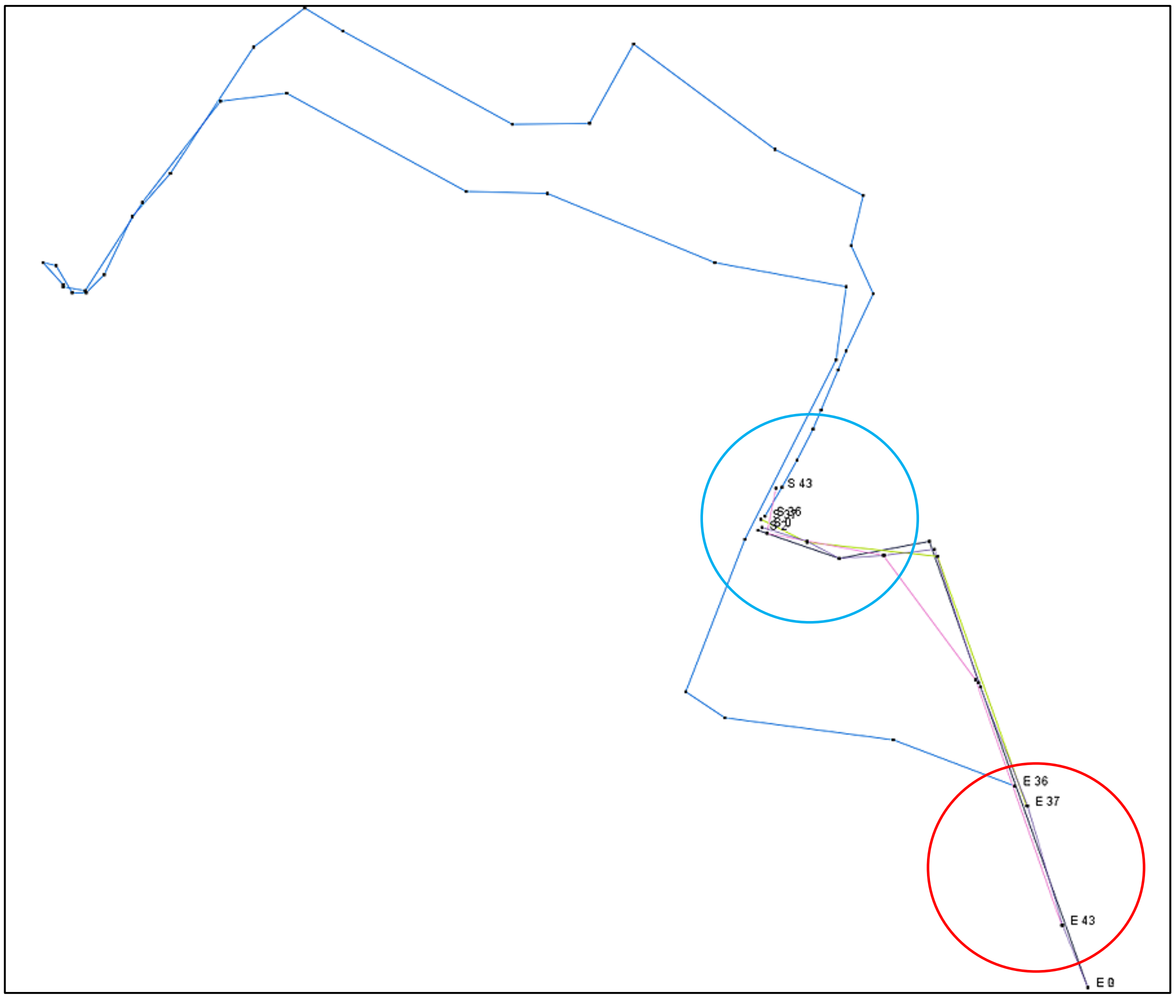


Figure 40: Test case 1 trajectories.

Nº of Trajectories:	50
Search radius:	100 meters
Clustering distance threshold:	300 meters
RT neighbors distance threshold:	30 meters

Table 9: Input parameters of test case 1.

Trajectory T_{36} is an interesting case because the distance traveled by the moving object is much higher than for the other ones. Yet it remains an alternative route and so it is also presented.

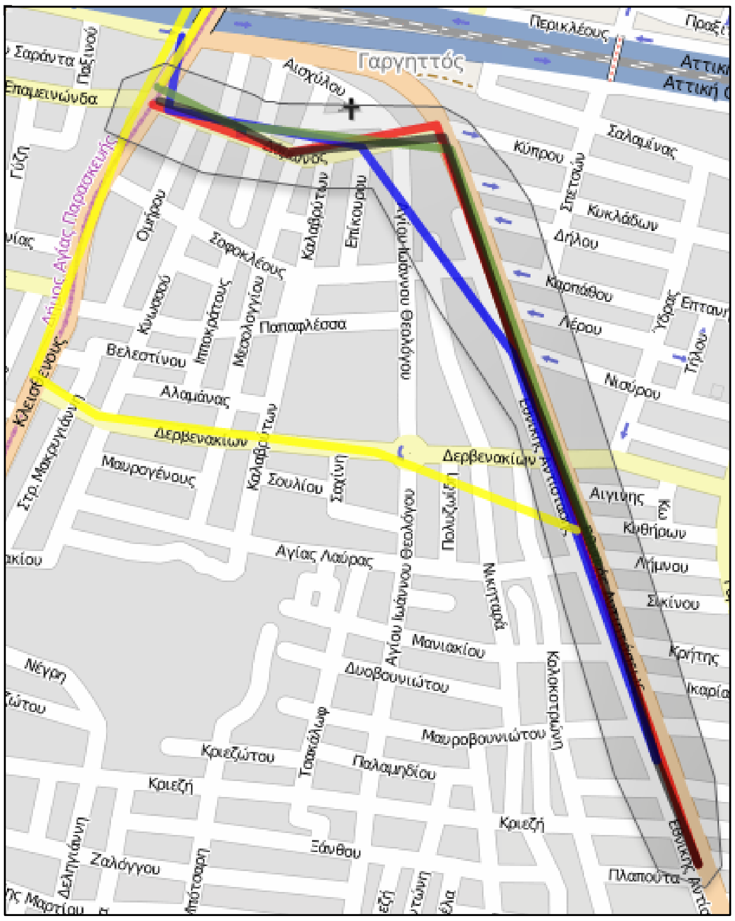
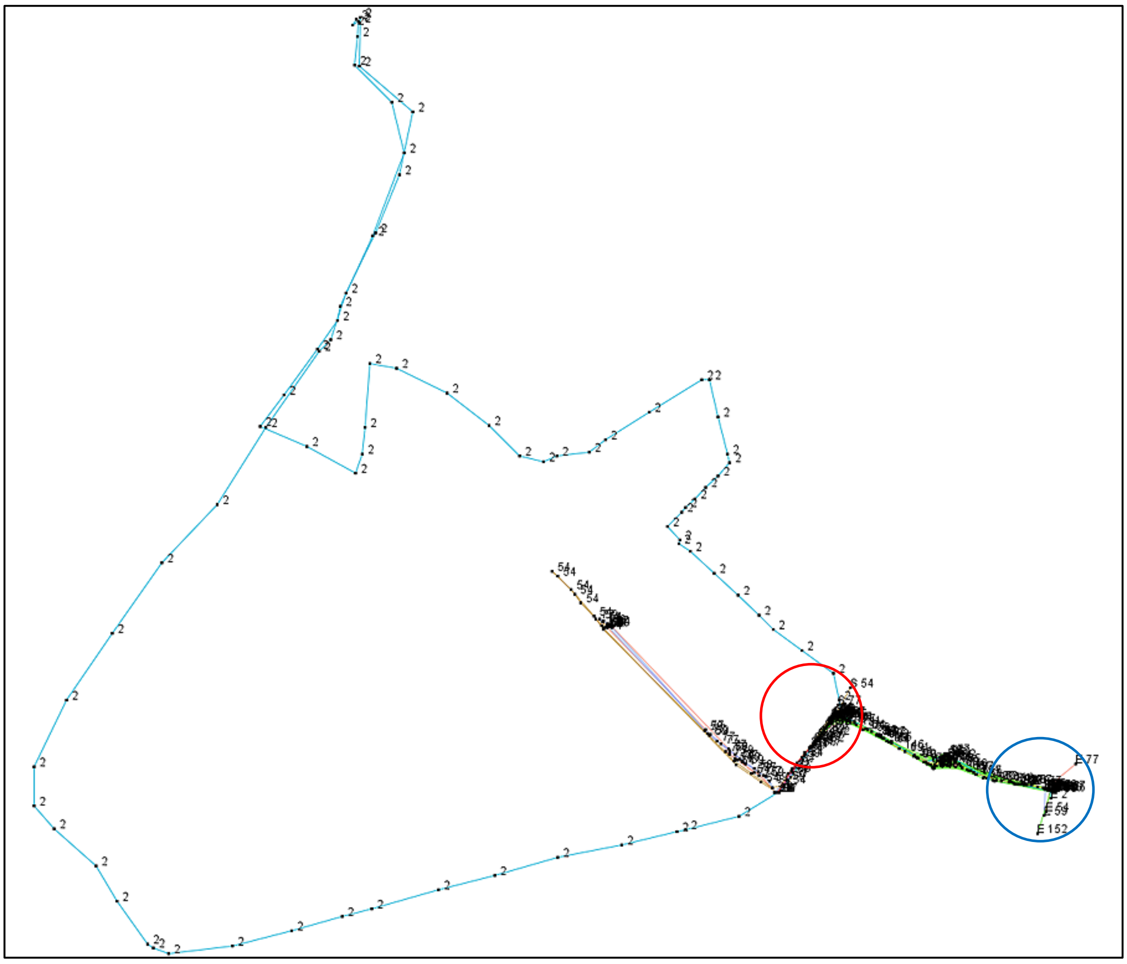


Figure 41: Test case 1, trajectories cluster and corresponding representative trajectory (black).

Two distinct clusters were found in the clustering step: $C_1 = \{T_2, T_{37}, T_{42}\}$ and $C_2 = \{T_{36}\}$. Figure 41 shows the trajectories in C_1 and C_2, where T_{36} is in yellow, T_2 is in red, T_{37} is in green, and T_{42} is in blue. The representative trajectory of cluster C_1 is in black. As it can be seen, the representative trajectory is a good approximation of the three trajectories in cluster C_1.The representation of C_2 is omitted because it contains a single trajectory.

4.1.2 Test case two

In this test case, the number of trajectories is increased to 200, two different clustering distance thresholds are used and the origin and destination positions have changed, as described in table 10. The filtering search step returns the 22 trajectories depicted in Figure 42 (1). Thus the initial set is:
$T = \{T_2, T_{36}, T_{37}, T_{43}, T_{54}, T_{59}, T_{64}, T_{68}, T_{70}, T_{77}, T_{84}, T_{88}, T_{90}, T_{92}, T_{98}, T_{146}, T_{152}, T_{157}, T_{165}, T_{177}, T_{191}, T_{192}\}$.
The first distance threshold for the clustering step was 300 meters and the second one was 200 meters.


(1)

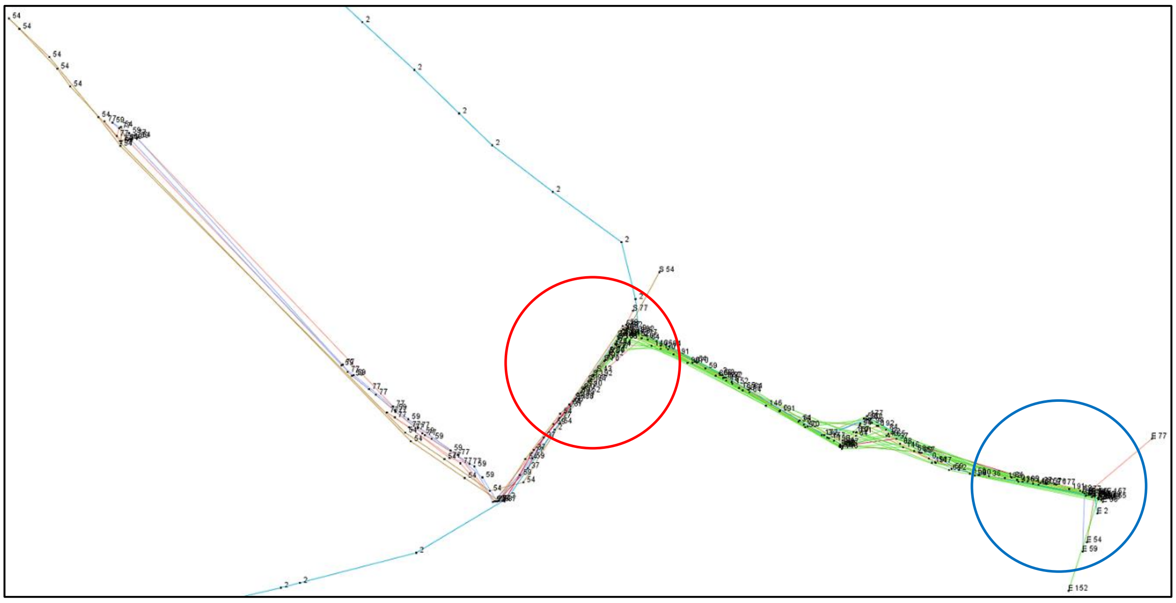

(2)

Figure 42: Test case 2 trajectories.

Nº of Trajectories:	200
Search radius:	100 meters
Clustering distance threshold:	300 and 200 meters
RT neighbors distance threshold:	30 meters

Table 10: Input parameters of test case 2.

The first scenario, using a distance threshold equal to 300 meters returned six clusters:

$C_1 = \{T_2\}$;

$C_2 = \{T_{36}, T_{43}, T_{64}, T_{68}, T_{70}, T_{84}, T_{88}, T_{90}, T_{92}, T_{98}, T_{146}, T_{152}, T_{157}, T_{165}, T_{177}, T_{191}, T_{192}\}$;

$C_3 = \{T_{37}\}$;

$C_4 = \{T_{54}\}$;

$C_5 = \{T_{59}\}$;

$C_6 = \{T_{77}\}$.

The second scenario, using a distance threshold equal to 200 meters resulted in eleven clusters:

$C_1 = \{T_2\}$;

$C_2 = \{T_{36}, T_{43}, T_{70}\}$;

$C_3 = \{T_{37}\}$;

$C_4 = \{T_{54}\}$;

$C_5 = \{T_{59}\}$;

$C_6 = \{T_{64}, T_{68}\}$;

$C_7 = \{T_{77}\}$;

$C_8 = \{T_{84}, T_{88}, T_{192}\}$;

$C_9 = \{T_{90}, T_{92}, T_{146}, T_{157}, T_{165}, T_{177}\}$;

$C_{10} = \{T_{98}\}$;

$C_{11} = \{T_{152}, T_{191}\}$.

The second scenario has more clusters as expected because the distance threshold is more restrictive. However, from a visually analysis of Figure 42 (2), the five clusters identified in the second scenario seem to belong to the same route (the trajectories in green). So, the decision was to create representative trajectories from the first scenario. Figure 43 presents a zoom of the map and shows the trajectories and the representative trajectory for clustersC_1 and C_2.

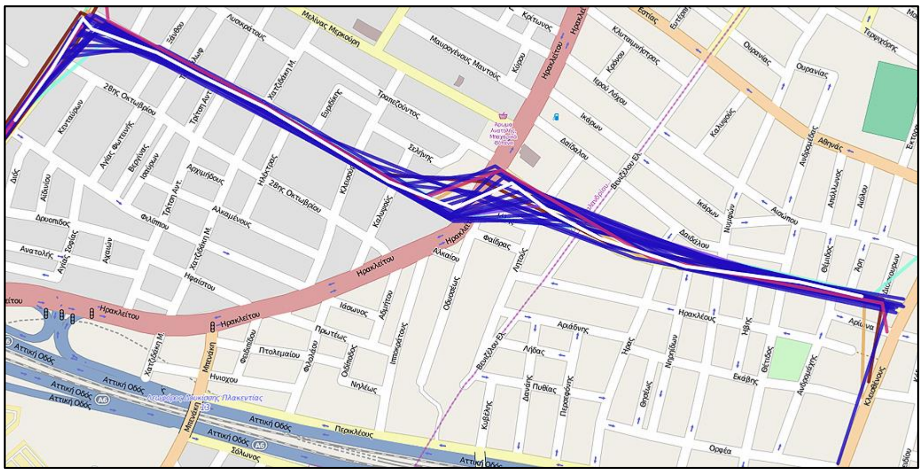
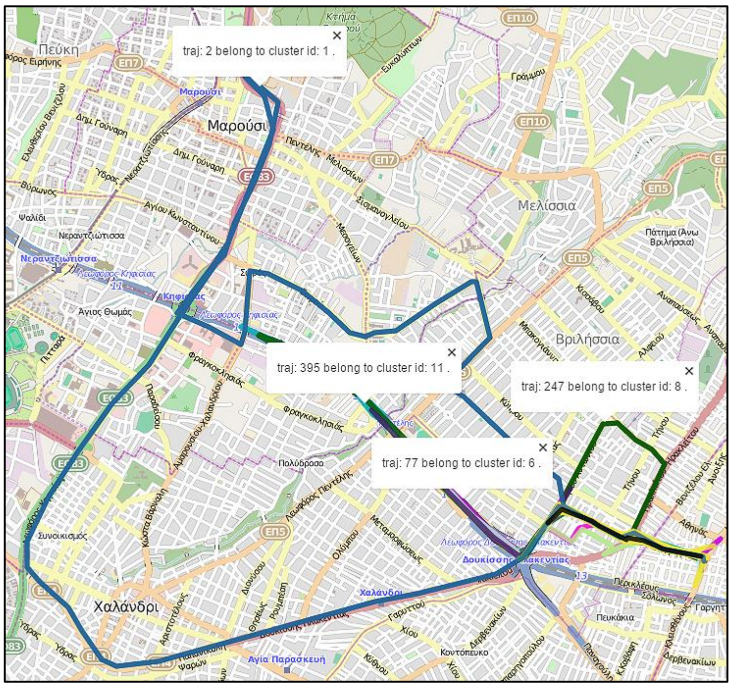


Figure 43: White representative trajectory illustrated on the Greece map roads.

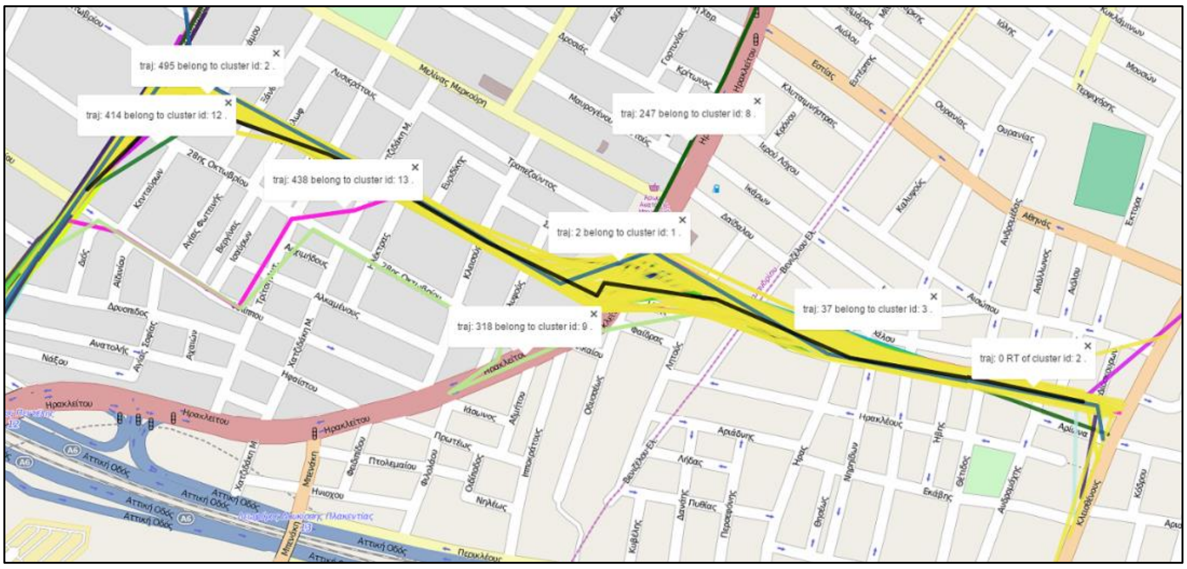
The trajectories belonging to C_2 are in blue and the representative trajectory is in white. The trajectory in C_1 is in red.

4.1.3 Test case three

The third test case uses the same parameters of test case number two but the number of trajectories is increased to 500 (Table 11). The filtering step found 52 trajectories. Figure 44 (1) shows the 52 trajectories and (2) shows a zoom of the map.


traj: 2 belong to cluster id: 1 .
traj: 395 belong to cluster id: 11 .
traj: 247 belong to cluster id: 8 .
traj: 77 belong to cluster id: 6 .

(1)


traj: 495 belong to cluster id: 2
traj: 414 belong to cluster id: 12
traj: 438 belong to cluster id: 13
traj: 247 belong to cluster id: 8
traj: 2 belong to cluster id: 1
traj: 318 belong to cluster id: 9
traj: 37 belong to cluster id: 3
traj: 0 RT of cluster id: 2

(2)

Figure 44: Test case 3: sub-trajectories (1) Cluster 2 zoomed (yellow).

Nº of Trajectories:	500
Search radius:	100 meters
Clustering distance threshold:	300 meters
RT neighbors distance threshold:	30 meters

Table 11: Input parameters of test case 3.

The cluster found in the clustering step are similar to those obtained in test case two:

$C_1 = \{T_2\};$

$C_2 = \{T_{36}, T_{43}, T_{64}, T_{68}, T_{70}, T_{84}, T_{88}, T_{90}, T_{92}, T_{98},$
$T_{146}, T_{152}, T_{157}, T_{165}, T_{177}, T_{191}, T_{192}, T_{226}, T_{248},$
$T_{251}, T_{252}, T_{264}, T_{319}, T_{388}, T_{401}, T_{412}, T_{420}, T_{440}, T_{448},$
$T_{452}, T_{456}, T_{463}, T_{466}, T_{482}, T_{495}\};$

$C_3 = \{T_{37}\};$

$C_4 = \{T_{54}\};$

$C_5 = \{T_{59}\};$

$C_6 = \{T_{77}\};$

$C_7 = \{T_{201}\};$

$C_8 = \{T_{247}\};$

$C_9 = \{T_{318}\};$

$C_{10} = \{T_{384}\};$

$C_{11} = \{T_{395}\};$

$C_{12} = \{T_{414}\};$

$C_{13} = \{T_{438}\};$

$C_{14} = \{T_{449}\};$

$C_{15} = \{T_{455}\};$

$C_{16} = \{T_{465}\};$

$C_{17} = \{T_{497}\}.$

Most trajectories between the origin and the destination belong to cluster C_2 and this could be a good candidate to be the best trajectory for recommendation.

4.1.4 Test case with modified parallel weight

This test case uses the same input parameters defined for test case number two, but the parallel distance weight ($w_{\parallel}$) in Formula 24 was set to 0.5. This change has a significant impact on the clustering step. The following clusters were obtained:

$C_1 = \{T_2\}$;

$C_2 = \{T_{36}, T_{64}, T_{68}, T_{70}, T_{84}, T_{92}, T_{146},, T_{165}, T_{191}\}$;

$C_3 = \{T_{37}, T_{43}, T_{88}, T_{90}, T_{98}, T_{152}, T_{157}, T_{177}, T_{192}\}$;

$C_4 = \{T_{54}\}$;

$C_5 = \{T_{59}\}$;

$C_6 = \{T_{77}\}$.

The cluster C_2 obtained in test case number two was divided into two distinct clusters denoted C_2 and C_3 in this test case. This test case shows that decreasing the weight of the parallel distance increases the similarity between the line segments in a cluster, but it can be harder to obtain clusters of trajectories related to the same route.

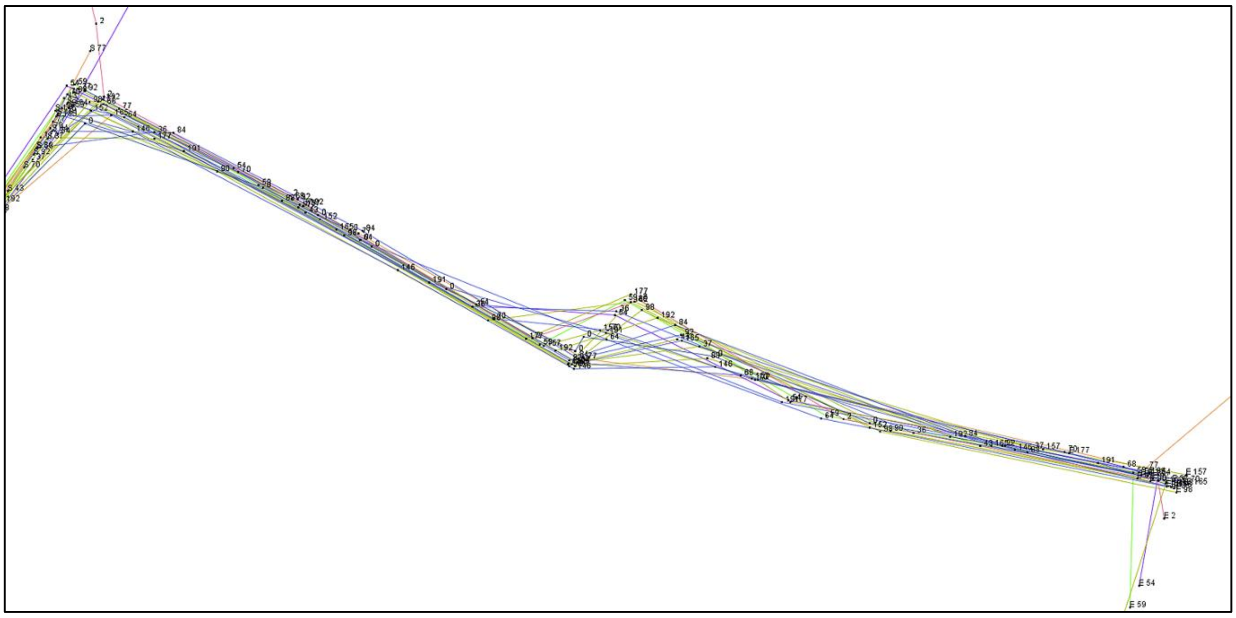


Figure 45: Cluster C_2 is decomposed in two clusters.

Figure 45, shows cluster C_2 divided into two groups with similar sub-trajectories (blue and orange). A visual analysis of Figure 45 shows that the clustering step algorithm could generate accurate clusters between sub-trajectories sharing the same routes.

4.2 Performance tests

The performance measure used in this section is the elapsed time required to answer to a user query. This is an important measure because it is expected to use the methods studied in this work in the context of interactive applications.

The tests were executed in the following environment: Windows 8 64 bit Operating System (OS), Intel Dual Core i7 processor with a base frequency of 2GHz and 4 MB of cache. Memory of 6 GB with a Column Access Strobe (CAS) latency of 9 clock cycles. Finally, the data is stored and accessed from an *mSATA* 128 GB with an average sequential read speed of 477MB/s and write speed of 294 MB/s.

The following figures are related to the different test cases, described in the preceding sections:

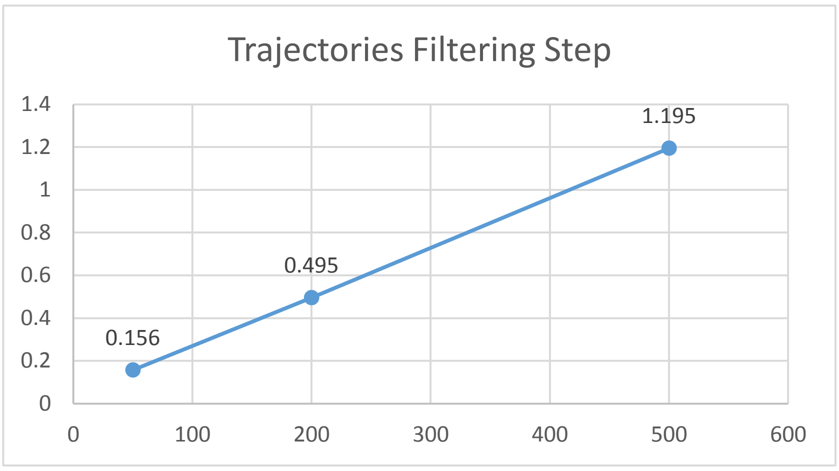


Figure 46: Trajectories Filtering Step execution times.

The filtering step to select the trajectories from an origin towards a destination is an iterative process and depends on the numbers of trajectories, i.e., execution time is proportional to the number of trajectory samples, as shown in Figure 46. The times are in seconds.

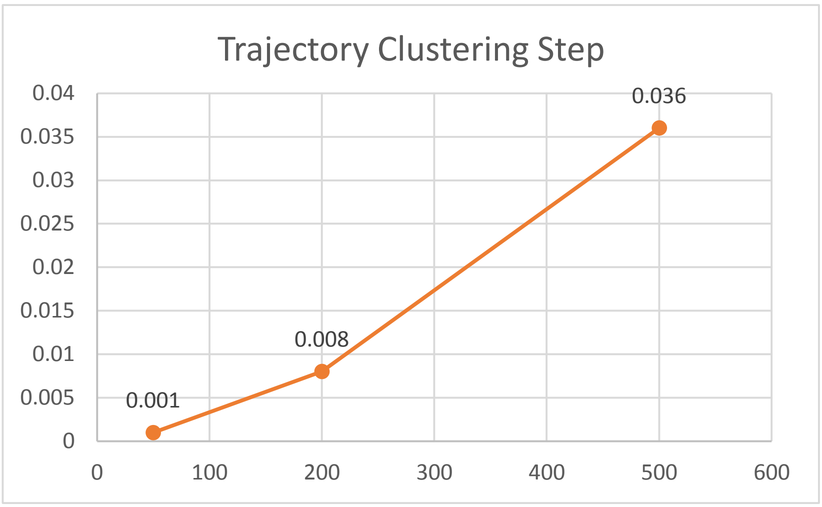


Figure 47: Clustering step execution time.

The clustering step, depicted in figure 47, has a short execution time than the previous step because it must deal with significantly less trajectories. The complexity of this step is closely related to the number of samples and trajectories to be processed, and the execution time is exponential.

This last step consists in creating the Representative Trajectory and is divided into 3 steps: Positions Connectivity Extraction, Centroid Position Representation and Centroid Assignment. The results presented in this section are organized as follows:

- Delaunay Triangulation Construction: measures the execution time for the computations of the triangulations.
- Representative Trajectory Calculation: includes the 3 steps to compute Representative Trajectories.
- Representative Trajectory Conversion: execution time to convert the coordinates from UTM to WGS84 reference system.

Figure 48, shows that the execution times for the Delaunay Triangulation are relatively small and linear. The execution times of the other two steps are more important than the construction of the triangulation, but they are logarithmic.

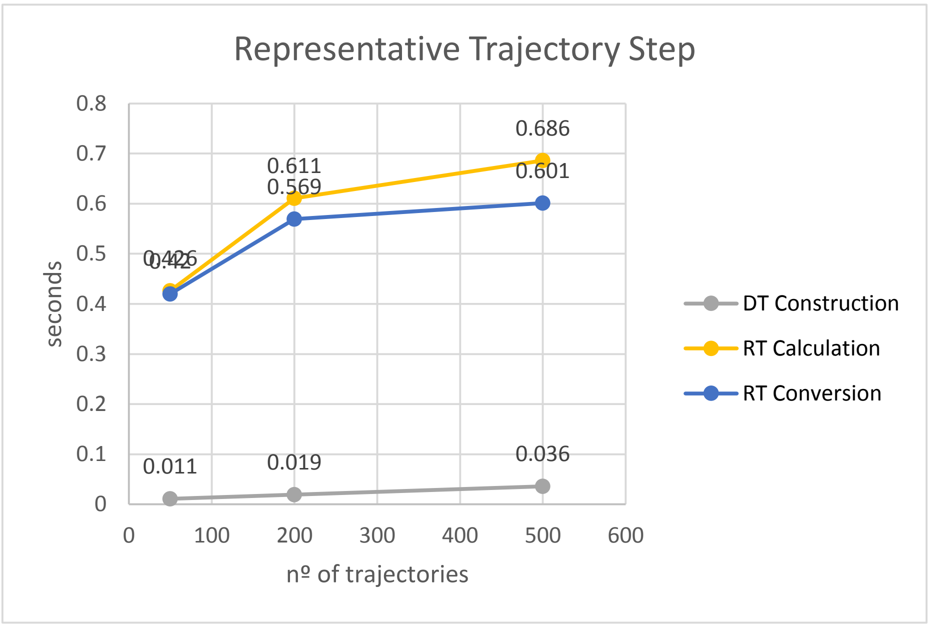


Figure 48: Representative trajectory sub-tasks execution times.

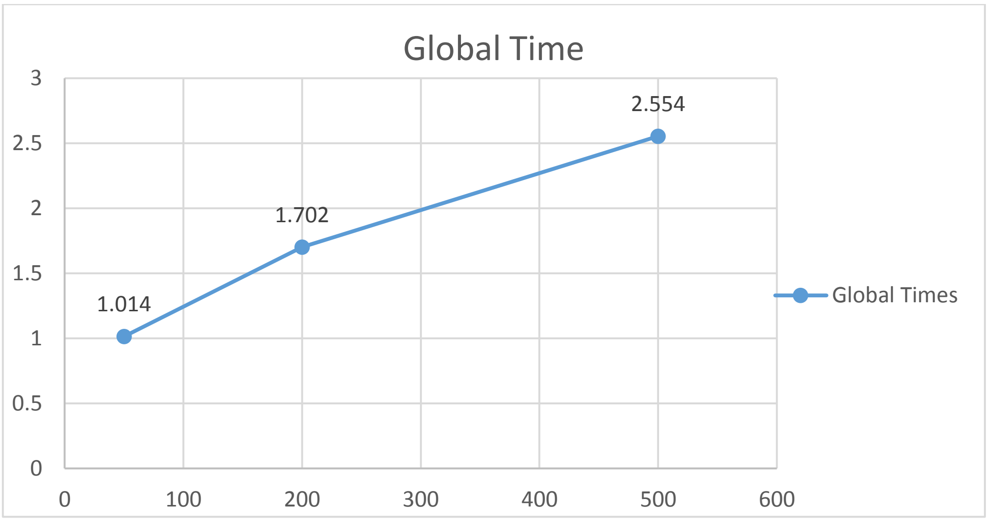


Figure 49: Sum of all steps times depending on the trajectories number.

Finally, Figure 49 presents the overall performance, i.e., the sum of all steps execution times, according to the number of trajectories. It was considered only one cluster in all tests. The overall execution times are 1.014 seconds for test case 1, 1.702 seconds for test case two and 2.554 seconds for test case 3.

The execution times are also dependent on the input parameters. For instance, changing the distance threshold to select the neighbors in the construction of the Representative Trajectories to 50 meters, the Representative Trajectory Conversion time is 0.471 seconds instead of 0.601 seconds and the Representative Trajectory Calculation time is 0.559 seconds instead of 0.686 seconds.

5. Chapter

Conclusion

This chapter presents the main issues studied in this work and presents guidelines for future works regarding moving objects recommendation.

5.1 Discussion

Two main issues were identified during the development of this work. Such limitations are in the process of obtaining clusters and representative trajectories.

The first issue concerns the evaluation of the distance function between line segments in the clustering step. The measures are not accurate when comparing vertical line segments and thus the algorithm tends to generate inadequate clusters. According to (Lee, Han et al. 2007) if two line segments are adjacent, then the parallel distance should be zero, but this may not be the case for the distance measures used in this work. The second issue concerns the Delaunay Triangulation because of the *Poly2Tri* java library used to calculate the triangulation network, does not support self-intersecting trajectories.

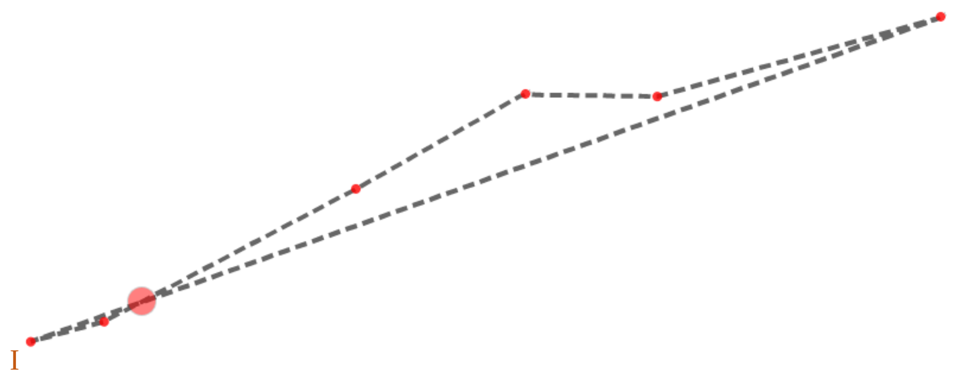


Figure 50: Self-intersecting trajectory.

Figure 50 depicts an example of a trajectory with an intersection (circle in red). In addition, the output generated in the *Centroid* Assignment step (Section 3.2.3.2.2), i.e., the centroids belonging to the representative trajectory, are not always generated in the correct order.

5.2 Contribution

In this dissertation, we propose a solution for trajectories clustering and recommendation, which are accessible through a web-based application. This application allows to discover the alternative routes to go from a given origin to a destination. For that purpose, it is necessary to create clusters of similar trajectories and to create a representative trajectory for each cluster. Ideally, each representative trajectory should correspond to a possible route between an origin and a destination. Then, it is possible to derive features for each possible route (cluster) using data about the trajectories in each cluster, and to recommend routes using several different criteria. The system has four main parts:

- The trajectory filtering step is used to search for trajectories sharing an origin and a destination regions.
- The clustering step groups similar trajectories using three measures (parallel, angular and perpendicular distances) to compute the distance between the line segments defining the moving objects trajectories.
- The representative trajectory step, is responsible to create a representative trajectory for each cluster. This step is based on a Delaunay Triangulation.
- The recommendation step, is basically a filter to suggest one of the representative trajectories according users options.

The solution presented in this work is not restricted to networked moving objects such as cars moving in a roads network, and it is expected that it is also able to deal with other types of trajectories, such as bicycles, boats or hiking, known as footpaths.

These features have been deployed through a web application.

5.3 Future work

The study presented in this dissertation is an exploratory work and there are several issues that need to be studied in the future. Data mining analysis applied to trajectories is challenging, particularly if real datasets are used rather than synthetic datasets.

Real datasets, such as GPS log files typically have approximation errors and outliers that make it more difficult to delimit and analyze moving objects trajectories. Performance is also an important issue when dealing with large trajectory datasets:

- The use distributed systems to divide and submit tasks to different machines. Several approaches are possible, for instance: (i) design a solution from scratch using Java RMI[19] (Remote Method Invocation), or (ii) use the open-source *Apache Hadoop*[20] framework to enable the processing of large datasets across clusters of computers.

[19] Source: docs.oracle.com/javase/7/docs/technotes/guides/rmi
[20] hadoop.apache.org

- The use of spatial databases rather than CSV files to store the geometric objects could improve substantially the performance because it would be possible to use a spatial index (i.e., *R-Tree*), to speed up query operations. For example, *PostgreSQL*[21] is a well-known open source database system, which uses the spatial *PostGIS* extension to add support for geometry data types and other advanced functions.

There also are several developments that can be implemented to improve the work presented in this dissertation. First, the performance of the filtering step to search for trajectories crossing an origin and a destination regions should be improved, because this step has the worst processing time. Second, the recommendation of trajectories in this work is quite simple and it would be interesting to extend this work to use contextual information, e.g., weather data road types, etc., or to use collaborative filtering techniques to be able to perform personalized recommendations.

[21] postgresql.org

Bibliography

Abraham, S. and P. S. Lal (2010). Trajectory similarity of network constrained moving objects and applications to traffic security. Proceedings of the 2010 Pacific Asia conference on Intelligence and Security Informatics. Hyderabad, India, Springer-Verlag**:** 31-43.

Ahmed, M. and C. Wenk (2012). Constructing street networks from GPS trajectories. Proceedings of the 20th Annual European conference on Algorithms. Ljubljana, Slovenia, Springer-Verlag**:** 60-71.

Ankerst, M., M. M. Breunig, et al. (1999). "OPTICS: Ordering Points To Identify the Clustering Structure". Philadelphia, Pennsylvania, USA, ACM.49-60.

C, S. and N. Sao (2012). "Data Mining on Moving Object Trajectories.". IJCSET:1040-1042.

Bäcklund, H., A. Hedblom, et al. (2011). "A Density-Based Spatial Clustering of Application with Noise: DBSCAN". Linköpings Universitet, 8 pages.

Baraglia, R., C. Frattari, et al. (2012). A trajectory-based recommender system for tourism. Proceedings of the 8th international conference on Active Media Technology. Macau, China, Springer-Verlag**:** 196-205.

Berndt, D. J. and J. Clifford (1994). "Using Dynamic Time Warping to Find Patterns in Time Series." AAA1-94 Workshop on Knowledge Discovery in Databases, 359-370.

Berndt, D. J. and J. Clifford (1996). Finding patterns in time series: a dynamic programming approach. Advances in knowledge discovery and data mining. M. F. Usama, P.-S. Gregory, S. Padhraic and U. Ramasamy, American Association for Artificial Intelligence**:** 229-248.

Birant, D. and A. Kut (2007). "ST-DBSCAN: An algorithm for clustering spatial–temporal data." Data Knowl. Eng. Journal, 208-221

Brilhante, I. R., J. A. F. d. Macedo, et al. (2011). Trajectory data analysis using complex networks. Proceedings of the 15th Symposium on International Database Engineering &#38; Applications. Lisboa, Portugal, ACM**:** 17-25.

Brinkhoff, T. (2002). "A Framework for Generating Network-Based Moving Objects." Geoinformatica **6**(2): 153-180.

Bu, Y., L. Chen, et al. (2009). Efficient anomaly monitoring over moving object trajectory streams. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. Paris, France, ACM**:** 159-168.

Buzan, D., S. Sclaroff, et al. (2004). "Extraction and clustering of motion trajectories in video." Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02, IEEE Computer Society, 521-524.

Cao, L. and J. Krumm (2009). From GPS traces to a routable road map. Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. Seattle, Washington, ACM**:** 3-12.

Chang, J.-W., R. Bista, et al. (2007). Spatio-temporal similarity measure algorithm for moving objects on spatial networks. Proceedings of the 2007 international conference on Computational science and its applications - Volume Part III. Kuala Lumpur, Malaysia, Springer-Verlag**:** 1165-1178.

Chen, C.-S., C. F. Eick, et al. (2011). Mining spatial trajectories using non-parametric density functions. Proceedings of the 7th international conference on Machine learning and data mining in pattern recognition. New York, NY, Springer-Verlag**:** 496-510.

Chen, J., M. K. Leung, et al. (2002). "Noisy logo recognition using line segment Hausdorff distance." Pattern Recognition, Volume 36: *943-955*.

Chen, L. and R. Ng (2004). On the marriage of Lp-norms and edit distance. Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. Toronto, Canada, VLDB Endowment**:** 792-803.

Chen, L., M. Tamer, et al. (2005). Robust and fast similarity search for moving object trajectories. Proceedings of the 2005 ACM SIGMOD international conference on Management of data. Baltimore, Maryland, ACM**:** 491-502.

Chen, P., J. Gu, et al. (2013). "A Dynamic Time Warping based Algorithm for Trajectory Matching in LBS." International Journal of Database Theory and Application Vol. 6, No. 3, 39-48.

Delaunay, B. (1934). "Sur la sphère vide." *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800.

Deng, M., Q. Liu, et al. (2011). "An adaptive spatial clustering algorithm based on delaunay triangulation." Computers, Environment and Urban Systems **35**(4): 320-332.

Ding, Y., S. Liu, et al. (2013). "HUNTS: A Trajectory Recommendation System for Effective and Efficient Hunting of Taxi Passengers." 2013 IEEE 14th International Conference on Mobile Data Management 2013, pp. 107-116.

Dodge, S., R. Weibel, et al. (2009). "Exploring movement-similarity analysis of moving objects." SIGSPATIAL Special **1**(3): 11-16.

Domiter, V. and B. Zalik (2008). "Sweep-line algorithm for constrained Delaunay triangulation." Int. J. Geogr. Inf. Sci. **22**(4): 449-462.

Ester, M., H.-P. Kriegel, et al. (1996). "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In Proc. of the 2nd Int’l Conference on Knowledge Discovery in Databases and Data Mining, Portland, Oregon, 6 pages.

Fathi, A. and J. Krumm (2010). Detecting road intersections from GPS traces. Proceedings of the 6th international conference on Geographic information science. Zurich, Switzerland, Springer-Verlag**:** 56-69.

George, Karypis, et al. (1999). "Chameleon: Hierarchical Clustering Using Dynamic Modeling." Computer, Volume 32: 68-75.

Gizem, et al. (2011). A hybrid video recommendation system using a graph-based algorithm. Proceedings of the 24th international conference on Industrial engineering and other applications of applied intelligent systems conference on Modern approaches in applied intelligence - Volume Part II. Syracuse, NY, Springer-Verlag**:** 406-415.

Goldberg, A. V. and C. Harrelson (2005). Computing the shortest path: A search meets graph theory. Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. Vancouver, British Columbia, Society for Industrial and Applied Mathematics**:** 156-165.

Grnwald, P. D., I. J. Myung, et al. (2005). Advances in Minimum Description Length: Theory and Applications (Neural Information Processing), The MIT Press.

Guha, S., R. Rastogi, et al. (1998). "CURE: an efficient clustering algorithm for large databases." SIGMOD Rec. **27**(2): 73-84.

Guibas, L. and J. Stolfi (1985). "Primitives for the manipulation of general subdivisions and the computation of Voronoi." ACM Trans. Graph. **4**(2): 74-123.

Guo, D. (2007). "Visual analytics of spatial interaction patterns for pandemic decision support." Int. J. Geogr. Inf. Sci. **21**(8): 859-877.

Guo, D. (2009). "Flow Mapping and Multivariate Visualization of Large Spatial Interaction Data." IEEE Transactions on Visualization and Computer Graphics **15**(6): 1041-1048.

Guo, D., S. Liu, et al. (2010). "A graph-based approach to vehicle trajectory analysis." J. Locat. Based Serv. **4**(3-4): 183-199.

Han, B., L. Liu, et al. (2012). NEAT: Road Network Aware Trajectory Clustering. Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems, IEEE Computer Society**:** 142-151.

Han, J., M. Kamber, et al. (2000). "Spatial Clustering Methods in Spatial Data Mining: A Survey." Paper presented at the meeting of the Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS, 2001.

Hwang, J.-R., H.-Y. Kang, et al. (2005). Spatio-temporal similarity analysis between trajectories on road networks. Proceedings of the 24th international conference on Perspectives in Conceptual Modeling. Klagenfurt, Austria, Springer-Verlag**:** 280-289.

Isenburg, M., Y. Liu, et al. (2006). "Streaming computation of Delaunay triangulations." ACM Trans. Graph. **25**(3): 1049-1056.

Jörg et al. (1998). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications." Data Min. Knowl. Discov. **2**(2): 169-194.

Jensen, C. S., D. Lin, et al. (2007). "Continuous Clustering of Moving Objects." IEEE Trans. on Knowl. and Data Eng. **19**(9): 1161-1174.

Jeung, H., H. T. Shen, et al. (2008). Convoy Queries in Spatio-Temporal Databases. Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, IEEE Computer Society**:** 1457-1459.

Jiashun, C. (2012). "A New Trajectory Clustering Algorithm Based on TRACLUS." Computer Science and Network Technology (ICCSNT): 783-787.

Kalnis, P., N. Mamoulis, et al. (2005). On discovering moving clusters in spatio-temporal data. Proceedings of the 9th international conference on Advances in Spatial and Temporal Databases. Angra dos Reis, Brazil, Springer-Verlag**:** 364-381.

Kang, I.-S., T.-w. Kim, et al. (1997). A spatial data mining method by Delaunay triangulation. Proceedings of the 5th ACM international workshop on Advances in geographic information systems. Las Vegas, Nevada, USA, ACM**:** 35-39.

Kharrat, A., I. S. Popa, et al. (2008). "Clustering algorithm for network constraint trajectories." 13th International Symposium on Spatial Data Handling, 631-647.

Kharrat, A., I. S. Popa, et al. (2009)."Caractérisation de la densité de trafic et de son évolution à partir de trajectoires d'objets mobiles". Proceedings of the 5th French-Speaking Conference on Mobility and Ubiquity Computing. Lille, France, ACM**:** 33-40.

Knorr, E. M. and R. T. Ng (1998). Algorithms for Mining Distance-Based Outliers in Large Datasets. Proceedings of the 24rd International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.**:** 392-403.

Knorr, E. M. and R. T. Ng (1999). Finding Intensional Knowledge of Distance-Based Outliers. Proceedings of the 25th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.**:** 211-222.

Knorr, E. M., R. T. Ng, et al. (2000). "Distance-based outliers: algorithms and applications." The VLDB Journal **8**(3-4): 237-253.

Lee, J.-G., J. Han, et al. (2008). Trajectory Outlier Detection: A Partition-and-Detect Framework. Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, IEEE Computer Society**:** 140-149.

Lee, J.-G., J. Han, et al. (2008). "TraClass: Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering." Proc. VLDB Endow. 1081-1094.

Lee, J.-G., J. Han, et al. (2007). Trajectory clustering: a partition-and-group framework. Proceedings of the 2007 ACM SIGMOD international conference on Management of data. Beijing, China, ACM**:** 593-604.

Li, Y. a., J. Han, et al. (2004). "Clustering Moving Objects." Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, Seattle, WA, USA, ACM: 617-622.

Li, Z. (2010). "Incremental clustering for trajectories." Proceedings of the 15th international conference on Database Systems for Advanced Applications - Volume Part II, Tsukuba, Japan, Springer-Verlag: 32-46.

Liu, D., G. V. Nosovskiy, et al. (2008). "Effective clustering and boundary detection algorithm based on Delaunay triangulation." Pattern Recogn. Lett. **29**(9): 1261-1273.

Liu, H. and M. Schneider (2012). Similarity measurement of moving object trajectories. Proceedings of the Third ACM SIGSPATIAL International Workshop on GeoStreaming. Redondo Beach, California, ACM**:** 19-22.

Liu, Q., M. Deng, et al. (2012). "A density-based spatial clustering algorithm considering both spatial proximity and attribute similarity." Comput. Geosci. **46**: 296-309.

Liu, W., Y. Zheng, et al. (2011). Discovering spatio-temporal causal interactions in traffic data streams. Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. San Diego, California, USA, ACM**:** 1010-1018.

Lo, S. and C. Lin (2006). WMR--A Graph-Based Algorithm for Friend Recommendation. Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE Computer Society**:** 121-128.

Ma, S., T. Wang, et al. (2003). "A New Fast Clustering Algorithm Based on Reference and Density." 4th International Conference, WAIM 2003, Chengdu, China, 214–225.

Mahrsi, M. K. E. (2013). "Analysis and Data Mining of Moving Object Trajectories." Télécom ParisTech, 143 pages.

Masciari, E. (2009). Trajectory Clustering via Effective Partitioning. Proceedings of the 8th International Conference on Flexible Query Answering Systems. Roskilde, Denmark, Springer-Verlag**:** 358-370.

Meratnia, N. and R. A. d. By (2002). Aggregation and comparison of trajectories. Proceedings of the 10th ACM international symposium on Advances in geographic information systems. McLean, Virginia, USA, ACM**:** 49-54.

Nanni, M. and D. Pedreschi (2006). "Time-focused clustering of trajectories of moving objects." J. Intell. Inf. Syst. **27**(3): 267-289.

O'Mahony, M. P. and B. Smyth (2007). A recommender system for on-line course enrolment: an initial study. Proceedings of the 2007 ACM conference on Recommender systems. Minneapolis, MN, USA, ACM**:** 133-136.

Palma, A. T., V. Bogorny, et al. (2008). A clustering-based approach for discovering interesting places in trajectories. Proceedings of the 2008 ACM symposium on Applied computing. Fortaleza, Ceara, Brazil, ACM**:** 863-868.

Pelekis, N., G. Andrienko, et al. (2012). "Visually exploring movement data via similarity-based analysis." J. Intell. Inf. Syst. **38**(2): 343-391.

Pelekis, N., I. Kopanakis, et al. (2009). Clustering Trajectories of Moving Objects in an Uncertain World. Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, IEEE Computer Society**:** 417-427.

Recio-García, J. A., et al. (2008). Prototyping recommender systems in jcolibri. Proceedings of the 2008 ACM conference on Recommender systems. Lausanne, Switzerland, ACM**:** 243-250.

Rinzivillo, S., D. Pedreschi, et al. (2008). "Visually driven analysis of movement data by progressive clustering." Information Visualization **7**(3): 225-239.

Roh, G.-P. and S.-w. Hwang (2010). NNCluster: an efficient clustering algorithm for road network trajectories. Proceedings of the 15th international conference on Database Systems for Advanced Applications - Volume Part II. Tsukuba, Japan, Springer-Verlag**:** 47-61.

Saglio, J.-M. and J. Moreira (2001). "Oporto: A Realistic Scenario Generator for Moving Objects." Geoinformatica **5**(1): 71-93.

Sakoe, H. and S. Chiba (1990). Dynamic programming algorithm optimization for spoken word recognition. Readings in speech recognition. W. Alex and L. Kai-Fu, Morgan Kaufmann Publishers Inc.**:** 159-165.

Senin, P. (2008). "Dynamic Time Warping Algorithm Review." Information and Computer Science Department, University of Hawaii at Manoa, Honolulu, USA, 23 pages.

Shang, S., R. Ding, et al. (2012). User oriented trajectory search for trip recommendation. Proceedings of the 15th International Conference on Extending Database Technology. Berlin, Germany, ACM**:** 156-167.

Shinar, U. L. a. A. (2009). "“Waze: Free navigation with turn by turn.” ", from http://www.waze.com/.

Shrestha, A., B. Miller, et al. (2013). Storygraph: extracting patterns from spatio-temporal data. Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics. Chicago, Illinois, ACM**:** 95-103.

Su, H., K. Zheng, et al. (2013). Calibrating trajectory data for similarity-based analysis. Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York, New York, USA, ACM**:** 833-844.

Tiakas, E., A. N. Papadopoulos, et al. (2006). Trajectory Similarity Search in Spatial Networks. Proceedings of the 10th International Database Engineering and Applications Symposium, IEEE Computer Society**:** 185-192.

Tiakas, E., A. N. Papadopoulos, et al. (2009). "Searching for similar trajectories in spatial networks." J. Syst. Softw. **82**(5): 772-788.

Vieira, M. R., P. Bakalov, et al. (2009). On-line discovery of flock patterns in spatio-temporal data. Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. Seattle, Washington, ACM**:** 286-295.

Vlachos, M., D. Gunopoulos, et al. (2002). Discovering Similar Multidimensional Trajectories. Proceedings of the 18th International Conference on Data Engineering, IEEE Computer Society**:** 673.

Wagner, R. A. and M. J. Fischer (1974). "The String-to-String Correction Problem." J. ACM **21**(1): 168-173.

Wu, E., W. Liu, et al. (2010). Spatio-temporal outlier detection in precipitation data. Proceedings of the Second international conference on Knowledge Discovery from Sensor Data. Las Vegas, NV, Springer-Verlag**:** 115-133.

Xia, Y., G.-Y. Wang, et al. (2010). Research of spatio-temporal similarity measure on network constrained trajectory data. Proceedings of the 5th international conference on Rough set and knowledge technology. Beijing, China, Springer-Verlag**:** 491-498.

Xie, C., J. Chang, et al. (2007). Hill-down strategy based DENsity CLUstEring and its application to medical image data. Proceedings of the 2nd international conference on Scalable information systems. Suzhou, China, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)**:** 1-2.

Yanagisawa, Y., J.-i. Akahani, et al. (2003). Shape-Based Similarity Query for Trajectory of Mobile Objects. Proceedings of the 4th International Conference on Mobile Data Management, Springer-Verlag**:** 63-77.

Yanagisawa, Y. and T. Satoh (2006). Clustering Multidimensional Trajectories based on Shape and Velocity. Proceedings of the 22nd International Conference on Data Engineering Workshops, IEEE Computer Society**:** 12.

Yang, X. and W. Cui (2009). "A Novel Spatial Clustering Algorithm Based on Delaunay Triangulation." Proc. SPIE 7285, International Conference on Earth Observation Data Processing and Analysis (ICEODPA), 9 pages.

Yi, B.-K., H. V. Jagadish, et al. (1998). Efficient Retrieval of Similar Time Sequences Under Time Warping. Proceedings of the Fourteenth International Conference on Data Engineering, IEEE Computer Society**:** 201-208.

Ying, J. J.-C., E. H.-C. Lu, et al. (2010). Mining user similarity from semantic trajectories. Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks. San Jose, California, ACM**:** 19-26.

Yoon, H., Y. Zheng, et al. (2010). Smart itinerary recommendation based on user-generated GPS trajectories. Proceedings of the 7th international conference on Ubiquitous intelligence and computing. Xi'an, China, Springer-Verlag**:** 19-34.

Yuan, J., Y. Zheng, et al. (2010). An Interactive-Voting Based Map Matching Algorithm. Proceedings of the 2010 Eleventh International Conference on Mobile Data Management, IEEE Computer Society**:** 43-52.

Yuan, J., Y. Zheng, et al. (2011). Where to find my next passenger. Proceedings of the 13th international conference on Ubiquitous computing. Beijing, China, ACM**:** 109-118.

Yuan, N. J., Y. Zheng, et al. (2013). "T-Finder: A Recommender System for Finding Passengers and Vacant Taxis." IEEE Trans. on Knowl. and Data Eng. **25**(10): 2390-2403.

Yuan, Y. and M. Raubal (2012). "Similarity measurement of mobile phone user trajectories - a modified edit distance method." Workshop on "Progress in Movement Analysis - Experiences with Real Data, Zurich, Switzerland, 5 pages.

Zahn, C. T. (1971). "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters." IEEE Trans. Comput. **20**(1): 68-86.

Zhang, T., R. Ramakrishnan, et al. (1996). BIRCH: an efficient data clustering method for very large databases. Proceedings of the 1996 ACM SIGMOD international conference on Management of data. Montreal, Quebec, Canada, ACM**:** 103-114.

Zhang, Z., K. Huang, et al. (2006). Comparison of Similarity Measures for Trajectory Clustering in Outdoor Surveillance Scenes. Proceedings of the 18th International Conference on Pattern Recognition - Volume 03, IEEE Computer Society**:** 1135-1138.

Zheng, M.-q., C. Chen, et al. (2008). An Algorithm for Spatial Outlier Detection Based on Delaunay Triangulation. Proceedings of the 2008 International Conference on Computational Intelligence and Security - Volume 01, IEEE Computer Society**:** 102-107.

Zheng, Y., L. Zhang, et al. (2011). "Recommending friends and locations based on individual location history." ACM Trans. Web **5**(1): 1-44.

Zhong, C., D. Miao, et al. (2010). "A graph-theoretical clustering method based on two rounds of minimum spanning trees." Pattern Recognition (Impact Factor: 2.58). 03/2010; 43(3):752-766.