Trajectory Data Mining in the Design of Intelligent Vehicular Networks

by

Roniel Soares de Sousa

Thesis submitted
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

Vehicular networks are a promising technology to help solve complex problems of modern society, such as urban mobility. However, the vehicular environment has some characteristics that pose challenges for wireless communication in vehicular networks not usually found in traditional networks. Therefore, the scientific community is yet investigating alternative techniques to improve data delivery in vehicular networks. In this context, the recent and increasing availability of trajectory data offers us valuable information in many research areas. These data comprise the so-called "big trajectory data" and represent a new opportunity for improving vehicular networks. However, there is a lack of specific data mining techniques to extract the hidden knowledge from these data.

This thesis explores vehicle trajectory data mining to design intelligent vehicular networks. In the first part of this thesis, we deal with errors intrinsic to vehicle trajectory data that hinder their applicability. We propose a trajectory reconstruction framework composed of several preprocessing techniques to convert flawed GPS-based data to road-network constrained trajectories. This new data representation reduces trajectory uncertainty and removes problems such as noise and outliers compared to raw GPS trajectories. After that, we develop a novel and scalable cluster-based trajectory prediction framework that uses enhanced big trajectory data. Besides the prediction framework, we propose a new hierarchical agglomerative clustering algorithm for road-network constrained trajectories that automatically detects the most appropriate number of clusters. The proposed clustering algorithm is one of the components that allow the prediction framework to process large-scale datasets.

The second part of this thesis applies the enhanced trajectory representation and the prediction framework to improve the vehicular network. We propose the VDDTP algorithm, a novel vehicle-assisted data delivery algorithm based on trajectory prediction. VDDTP creates an extended trajectory model and uses predicted road-network constrained trajectories to calculate packet delivery probabilities. Then, it applies the predicted trajectories and some proposed heuristics in a data forwarding strategy, aiming to improve the vehicular network's global metrics (i.e., delivery ratio, communication overhead, and delivery delay). In this part, we also propose the DisTraC protocol to demonstrate the applicability of vehicular networks to detect traffic congestion and improve urban mobility. DisTraC uses V2V communication to measure road congestion levels cooperatively and reroute vehicles to reduce travel time.

We evaluate the proposed solutions through extensive experiments and simulations. For that, we prepare a new large-scale and real-world dataset based on the city of Rio

de Janeiro, Brazil. We also use other real-world datasets publicly available. The results demonstrate the potential of the proposed data mining techniques (i.e., trajectory reconstruction and prediction frameworks) and vehicular networks algorithms.

List of Publications

1. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "Vehicle-assisted Data Delivery based on Trajectory Prediction," *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2022.

2. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "On the Reconstruction of Vehicular Trajectories from Low-Sampling-Rate GPS Datasets," *IEEE Transactions on Intelligent Transportation Systems*, 2022. Second Round.

3. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "On the Applications of Vehicle Trajectory Similarity Measures," *IEEE Vehicular Technology Magazine*, 2022. Under review.

4. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "On the prediction of large-scale road-network constrained trajectories," *Computer Networks*, vol. 206, no. 108337, pp. 1-13, Jul. 2021, doi: 10.1016/j.comnet.2021.108337.

5. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "A Cluster-based Framework for Predicting Large Scale Road-Network Constrained Trajectories," *Proceedings of the 17th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks (PE-WASUN)*, Nov. 2020, pp. 1-8, doi: 10.1145/3416011.3424751.

6. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "Vehicle Trajectory Similarity: Models, Methods, and Applications," *ACM Computing Surveys*, vol. 53, no. 5, pp. 1-32, Sep. 2020, doi: 10.1145/3406096.

7. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "A Map Matching based Framework to Reconstruct Vehicular Trajectories from GPS Datasets," *Proceedings of the IEEE International Conference on Communications (ICC)*, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9148732.

8. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "A Distributed and Low-Overhead Traffic Congestion Control Protocol for Vehicular Ad Hoc Networks," *Computer Communications*, vol. 159, pp. 258-270, 2020,
doi: 10.1109/ISCC47284.2019.8969603.

9. R. S. de Sousa, A. Boukerche and A. A. F. Loureiro, "DisTraC: A Distributed and Low-Overhead Protocol for Traffic Congestion Control Using Vehicular Networks," *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, 2019, pp. 1-6, doi: 10.1109/ISCC47284.2019.8969603.

10. R. S. de Sousa, F. S. da Costa, A. C. B. Soares, L. F. M. Vieira and A. A. F. Loureiro, "Geo-SDVN: A Geocast Protocol for Software Defined Vehicular Networks," *Proceedings of the IEEE International Conference on Communications (ICC)*, 2018, pp. 1-6, doi: 10.1109/ICC.2018.8422755.

Acknowledgements

I thank God.

I thank my parents, Neube Fernandes de Sousa and Naildes Soares de Sousa, for their education and always believing in me. I am also thankful to my brother, Rafael Soares de Sousa, for all the help and for always being present in my life.

I am very thankful to my beloved wife, Katharinne Sabrina Nascimento Teixeira Soares, for her love and companionship throughout this stage. My accomplishments are only possible because of your presence, support, and love in all moments. I love you!

I also thank my godparents, Nora Ney Soares de Sousa and João Almiro Lustosa, for being fundamental in my education and always supporting me.

I thank my supervisors, Antonio Loureiro and Azzedine Boukerche, for all the advice, patience, and help in this period of my life. Also, I am thankful to the teachers, staff, and faculty members who supported me directly or indirectly.

Finally, I would like to thank all my colleagues and family members who contributed to me and made this phase of my life much more pleasant.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The changes in the movement aspects of society and the growth of population concentration in large cities bring many challenges for modern civilization, such as urban mobility. According to INRIX, Inc., traffic jams caused a waste of $87 billion in the United States in 2018 [90]. Besides that, the Centre for Economics and Business Research reported that the economy-wide costs across four advanced economies are forecast to rise 46% from 2013 to 2030. Therefore, there is a growing need to develop smart transportation solutions to move people, goods, and animals efficiently.

A notable alternative to improve urban mobility and other aspects of modern society is the development of new technologies and transportation modes, leading to an Intelligent Transportation System (ITS). ITSs use technologies such as inductive loops, Global Positioning System (GPS), and wireless communication to provide better (intelligent) solutions to a city's transportation. In this context, wireless communication in the vehicular environment, which comprises vehicular networks, is prominent in the scientific community.

Vehicular networks, also known as VANETs (Vehicular Ad Hoc Networks), are wireless communication systems that allow the exchange of data between vehicles (V2V communication - Vehicle-to-Vehicle) and between vehicles and infrastructures located along the roadsides (V2I communication - Vehicle-to-Infrastructure). Two vehicles can exchange data when they are within communication range of each other, typically around 300 meters. However, the vehicular environment has some characteristics that present challenges for communication in VANETs, such as the following.

- The communication links frequently change because of the constant movement of vehicles (i.e., vehicular networks present a highly dynamic topology).

- Regions with traffic congestion can concentrate many vehicles within the same communication range, which results in a dense network that can lead to the broadcast storm problem. The broadcast storm problem causes different issues, such as redundancy, contention, and collision, associated with flooding in wireless networks that hinder communication [195].

- Regions with few vehicles result in sparse networks [213], which make communication difficult since vehicles in different regions can be isolated, preventing communication among them [141].

- Proposals for vehicular networks should have a low communication overhead to not adversely impact the applications that are sensitive to the transmission delay (e.g., traffic safety applications) [48].

1.1 Motivation

The challenges that harm communication using vehicular networks are not entirely solved, and a promising approach is to extract mobility knowledge from the increasingly available trajectory data to improve vehicle-assisted data delivery. The so-called "big trajectory data" offer us crucial information to understand movement patterns and plays a pivotal role in research fields such as urban planning, traffic analysis, location-based social networks, vehicular networks, and other data mining-based applications. For instance, we can learn from historical trajectory data to predict mobility and traffic congestion. Besides that, we can use the acquired knowledge to improve the networks' performance and, consequently, disseminate traffic congestion data and reduce vehicle travel time. However, problems such as noises, outliers, and gaps present in most trajectory datasets and the increasing volume of trajectory data demand the development of new preprocessing algorithms and analysis techniques to apply the data in the applications mentioned above efficiently.

In this thesis, we study vehicle trajectory data mining as a tool to design Intelligent Vehicular Networks. In particular, we first investigate how we can preprocess trajectories to enhance data quality, effectively improving the applicability of this data in vehicular networks. Then, we study trajectory prediction based on enhanced big trajectory data. Finally, we design vehicle-assisted data delivery techniques based on trajectory prediction. In addition, we demonstrate how vehicular networks can be applied to estimate and reduce traffic congestion.

1.2 Objectives

The main objective of this thesis is to investigate data mining techniques to extract mobility knowledge from vehicle trajectory data and improve the efficiency and effectiveness of vehicular networks. We divide our specific objectives into two parts, the first focusing on trajectory data mining and the second on vehicular networks.

First, we aim at investigating techniques based on trajectory data for predicting mobility, travel time, and other information that can be applied to improve data delivery in vehicular networks. For that, we plan to design methods to reconstruct vehicle trajectories (road-network constrained trajectories) from GPS-based data, as most GPS-based trajectory data are unreliable. Thus, we define our specific objectives from the perspective of trajectory data mining as follows:

- Our first objective is to create a framework to reconstruct road-network constrained trajectories from unreliable GPS-based datasets. This objective is essential as most analysis techniques benefit from a better trajectory representation.
- We investigate how we can effectively and efficiently predict long-term trajectories from historical datasets. In other words, we intend to develop a method that predicts the next road segments that a vehicle will travel based on its current partial trajectory and historical trajectory datasets.

In the second part, our objective is to use the knowledge extracted from big trajectory data to improve the vehicular networks' global metrics (i.e., delivery ratio, communication overhead, and delivery delay). We intend to apply the developed data mining techniques (e.g., trajectory prediction algorithm) to design new data forwarding algorithms.

1.3 Contributions

We list the main contributions of this thesis in the following.

- **The Rio Center Dataset.** One of the challenges in evaluating vehicular network applications and the related data mining techniques is the lack of vehicle trajectory datasets. Therefore, we present a novel real-world, large-scale vehicle trajectory dataset based on the city of Rio de Janeiro, called Rio Center Dataset. We prepare different versions of the dataset to evaluate data mining techniques (e.g., trajectory reconstruction and prediction) and vehicular networks applications.

- **Trajectory Reconstruction Framework.** This contribution consists in proposing an efficient framework to reconstruct vehicle trajectories from large-scale GPS datasets. The framework is composed of several preprocessing techniques to allow the reconstruction of the entire real trajectory.

- **Trajectory Prediction Framework.** This contribution proposes a new cluster-based framework for the long-term prediction of road-network constrained trajectories. The framework trains prediction models from clustered historical trajectory datasets. Besides that, we present a new hierarchical agglomerative clustering algorithm for this task, which automatically detects the most appropriate number of clusters.

- **DisTraC protocol.** This contribution consists of the **Dis**tributed and low-overhead protocol for **Tra**ffic **C**ongestion control (DisTraC protocol), which aims at measuring road congestion levels and reducing the average travel time of vehicles by employing V2V communication. The protocol is independent of external infrastructures such as roadside units or cellular network towers as it uses only V2V communication.

- **VDDTP algorithm.** This contribution employs the trajectory reconstruction and prediction frameworks to create the VDDTP algorithm, a novel vehicle-assisted data delivery algorithm based on trajectory prediction. VDDTP demonstrates the application of different data mining techniques to improve data delivery in vehicular networks. First, it creates an extended trajectory model and uses predicted road-network constrained trajectories to calculate packet delivery probabilities. Then, the algorithm employs a data forwarding strategy to improve the delivery ratio and reduce communication overhead and delivery delay.

1.4 Thesis Outline

This thesis is organized as follows. Chapter 2 introduces some definitions and reference models used throughout this thesis. Chapter 3 surveys and classifies the main vehicle trajectory similarity measures, which is a fundamental task of trajectory analysis. Chapter 4 presents the Rio Center Dataset. Chapter 5 proposes the framework to reconstruct vehicle trajectories from GPS datasets. Chapter 6 presents the new cluster-based framework for the long-term prediction of road-network constrained trajectories. Chapter 7

deals with the problem of applying vehicular networks to detect and reduce traffic congestion and presents the DisTraC protocol. Chapter 8 presents the VDDTP algorithm, which employs the trajectory reconstruction and prediction frameworks to improve data delivery in vehicular networks. Finally, Chapter 9 summarizes the contributions of this thesis and presents future research directions.

Chapter 2

Preliminaries, Definitions, and Reference Models

This Chapter introduces some definitions and reference models used throughout this thesis. We organize the Chapter as follows. Section 2.1 presents some basic definitions to introduce the graph-based road-network model and trajectory representations necessary for elaborating the problems addressed in this thesis. Section 2.2 discusses the trajectory representations, their advantages, disadvantages, and applicability in more detail. Finally, Section 2.3 concludes this Chapter.

2.1 Road Network and Vehicle Trajectory Models

Definition 1. (Road Network) A road network is a single directed graph $\mathcal{N} = (\mathcal{V}, \mathcal{E})$, where the set of arcs $\mathcal{E}$ represents the roads (for simplicity, we use the term road as a synonymous of road segment), and the set of nodes $\mathcal{V}$ represents the road intersections. Each arc $e \in \mathcal{E}$ connects a start node e.from $\in \mathcal{V}$ to an end node e.to $\in \mathcal{V}$, and each node $v \in \mathcal{V}$ is endowed with its coordinates v.lng and v.lat (i.e., longitude and latitude), representing its position on Earth. Besides that, the road network graph can have additional information, such as the maximum allowed speed and number of lanes in a road. Finally, each arc $e \in \mathcal{E}$ stores a *linestring* (i.e., an ordered set of coordinates) representing its shape, where the first and last coordinates are equal to the positions of e.from and e.to, respectively.

Definition 2. (GPS-based Trajectory) A GPS trajectory (also called raw trajectory), denoted by $\mathrm{T}_{gps} = (\rho_1, \rho_2, \ldots, \rho_n)$, is a series of sampling points ordered

by timestamps with $\rho_i = (\rho_i.\text{lng}, \rho_i.\text{lat}, \rho_i.\text{t})$, where $\rho_i.\text{lng}$, $\rho_i.\text{lat}$, and $\rho_i.t$ represent the longitude, latitude, and timestamp of the sampling point, respectively. A GPS-based trajectory may have other attributes associated with each point, such as the vehicle's speed $\rho_i.\text{spd}$ and direction $\rho_i.\text{dir}$.

Definition 3. (Road-Network Constrained Trajectory) A road-network constrained trajectory $\text{T} = (t_1, t_2, \ldots, t_n)$ is an ordered set of connected roads $t_i \in \mathcal{E}$ that represent the path traveled by a vehicle. That is, $\forall t_i \in (t_1, \ldots, t_{n-1})$, $t_i.to$ must be equal to t_{i+1}.from. Besides, t_1.from and t_n.to represent the vehicle's initial and final positions, respectively.

2.2 Trajectory Representations

A trajectory is a trace generated by the movement of some entity (e.g., person or vehicle) over a specific time window. Although the movement is continuous, traces are usually represented by a sample of the object's real movement due to limitations of location positioning devices [191], so it is not always possible to have a trajectory representation that fully captures the movement of the object. In the context of vehicles, the most common representations of trajectories are *GPS-based trajectories* and *Road-network constrained trajectories*. Data representation formats are closely related to distance measures and thus are intended to support a particular set of data mining tasks, such as similarity-based search. Furthermore, accurate representations of vehicle trajectories are essential for experiments of vehicular networks.

2.2.1 GPS-Based Trajectories

A GPS-based trajectory, also called a raw trajectory, is a trajectory representation directly obtained from the logs generated by GPS-equipped devices. As defined in Section 2.1, a GPS-based trajectory is an ordered set of sampling points, where each point contains a set of attributes, such as the timestamp and vehicle's coordinates.

A problem of raw trajectories is the trajectory uncertainty, which occurs because positioning-devices record data at discrete time intervals and, therefore, we do not know the real movement of a vehicle between a trajectory point and the following one. For example, when a vehicle is moving at 60 kilometers per hour (1 km/min) and reporting its location every 30 seconds, there is a movement gap of 500 meters, which means that the actual position of the vehicle is unknown for most of this time interval. Most

trajectory datasets contain such gaps, since location-acquisition devices usually record the positions with a sampling interval of at least 30 seconds to reduce communication loads and storage costs [30, 19]. On the one hand, some applications need to increase even more the uncertainty of a trajectory to protect the user's privacy. On the other hand, the uncertainty in trajectories hinders various data mining tasks, such as calculating the distance between them. Figure 2.1 illustrates two different trajectories (blue: A1-A2-A3, and red: B1-B2-B3) that look similar if we take into consideration only the points of their GPS-based representations. This misreading happens because of the gaps generated by a low-sampling rate. Thus, many studies attempt to address the uncertainty of trajectories by reducing or removing the gaps, modeling the uncertainty of trajectories, or inferring the actual path of the trajectory. We discuss some of these methods below.

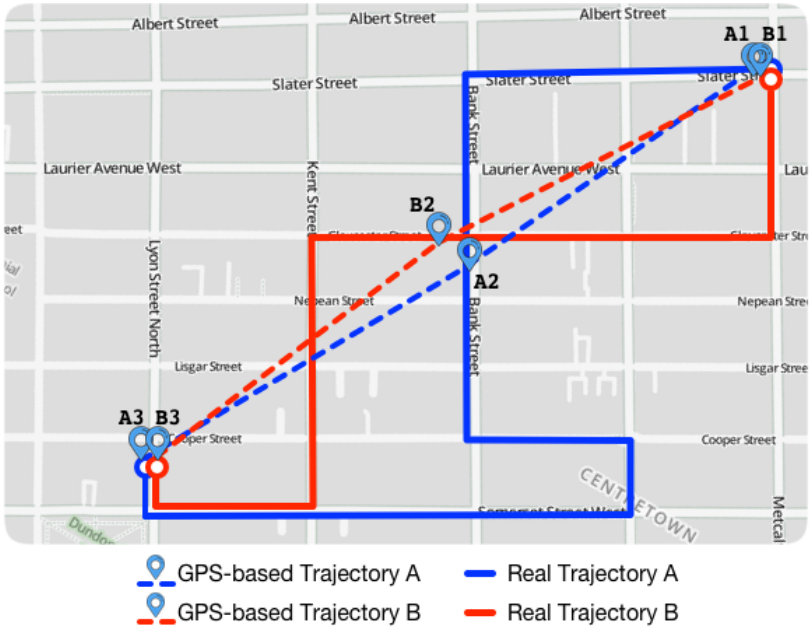


Figure 2.1: Two different GPS-based trajectories that look similar because of the gaps in the data.

Filling the Gaps

Interpolation is the most straightforward idea to remove the gaps in vehicular trajectories. For instance, assuming that the vehicle moves straight from a sample to another, we can apply linear interpolation [185] to represent the unknown movement. However, in most situations, this technique is ineffective because it is common for vehicles to follow

curved paths. A better option is to use the road network to find the shortest path between consecutive points and thereby improve the interpolation [120]. Other advanced methods produce more accurate results by using historical trajectory data from the same dataset of the analyzed trajectory [30, 19, 176]. Su et al. [176] and Celes et al. [30] proposed an approach composed of a reference system and a calibration method. They use anchor points from historical GPS datasets to construct the reference system, and then the calibration method finds trajectory data in the reference system to fill the gaps. Bedogni et al. [19] proposed a methodology to derive entire trajectories of individual vehicles considering sparse and inaccurate samplings. Using that approach, it is possible to reconstruct a given trajectory.

A further step would be to use the geographic constraints of the road network so as not to insert points outside of the roads. Recent approaches use probabilistic models to verify all possible sequences of modal activities (e.g., acceleration and deceleration) between the trajectory points and then reconstruct a trajectory that has one sample per second [80, 192].

Modeling the Uncertainty of Trajectories

Another approach is to model the uncertainty of trajectories and incorporate it into the trajectory representation. In the past few years, many authors studied Markov chains [60, 142, 157, 209] and other stochastic models [153, 186, 187] to represent better the uncertain positions of moving objects. Pfoser and Jensen [153] proposed a technique that first obtains the positions in-between the sampling points by using interpolation, and then, computes two measurement errors, one about each position in time, and a global worst-case error. They assume that the distribution of the measurement error of GPS points is Gaussian. Besides that, they proposed a trajectory representation that includes the movements as linearly interpolated positions and the parameters of the error distributions associated with the movements. One drawback of this approach is that it uses linear interpolation, which is not accurate to represent vehicle movement.

Trajcevski et al. [187] proposed a trajectory uncertainty model based on ideas similar to the method presented by Pfoser and Jensen [153]. However, their uncertainty model is a cylindrical volume in 3D, in contrast with the model give by Pfoser and Jensen [153], which is an ellipse where the endpoints of the segments represent its foci. However, like the trajectory model of Pfoser and Jenson [153], Trajcevski et al. [187] represent a trajectory as a three-dimensional polyline. In other words, it uses linear interpolation and therefore assumes that the movement of the entity between consecutive sample

points follows a straight line at a constant speed. Trajcevski et al. Trajcevski et al. [187] described an uncertainty model based on rectangles intended for scenarios in which we know that the object has a movement restricted to a road network, but its correct lane is unknown. However, they did not examine this method in their paper.

Trajcevski et al. [186] evaluated another model for trajectory uncertainty, which uses the concepts introduced in [85]. In their model, the so-called *beads* bounds the possible whereabouts of a given object between the trajectory sampling points, and the *necklace*, which is a sequence of beads, represents the trajectory. This model has many properties equivalently to the one proposed by Pfoser and Jensen [153]. The model assumes that the only thing known about the moving object in-between two consecutive locations points is that there is a threshold that bounds its maximum speed. Trajcevski et al. [186] demonstrated through experiments that the beads/necklaces model could be used to improve queries of certain types by using pruning strategies.

Lu et al. [125] proposed to reconstruct a complete trajectory of a mobile object given its partial trajectories recorded in different transportation modes (e.g., subway, car, walk). Their solution, called trajectory splicing, must satisfy some expected requirements: disjoint time overlap of the partial trajectories (i.e., they cannot have a time overlap if they refer to the same object); spatial proximity of a sequence of partial trajectories; and a set of partial trajectories should not be contained in a larger set of trajectories. Trajectory splicing helps to build a complete trajectory when there is uncertainty about the different partial trajectories that might comprise the original one.

2.2.2 Road-network constrained trajectories

Vehicular trajectories have an essential feature that the trajectories of most of the other kinds of moving objects do not have: are physically constrained to the road network. Because of this, we can map vehicular trajectories to the roads respecting the turn restrictions, and thus represent the movement of the vehicle better. We call this type of representation a road-network constrained trajectory. When properly constructed, a road-network constrained trajectory is more concise and precise than the GPS-based trajectory (raw trajectory). Figure 2.2 illustrates a road-network constrained trajectory in comparison with a GPS-based trajectory.

Road-network constrained trajectories are generally constructed from GPS-based trajectories. However, GPS-based trajectories contain errors such as noise and outliers, as satellite-based positioning systems do not always produce accurate data. These errors

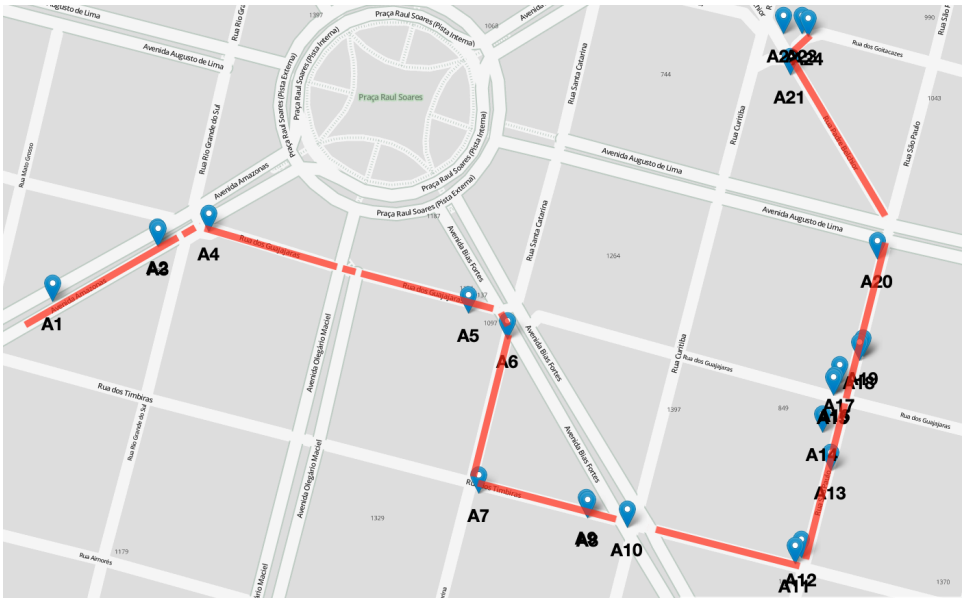


Figure 2.2: A GPS-based trajectory (pins on the map) and its representation as a road-network constrained trajectory (red lines). Unlike the GPS-based trajectory, the road-network constrained trajectory does not have positional errors.

mainly happen because of poor satellite visibility in urban regions, which results in the occurrence of phenomena such as reflection and diffraction [106]. Therefore, some preprocessing techniques (e.g., noise filtering and map matching) are employed to reconstruct a road-network constrained trajectory [215].

In recent years, many authors proposed road network models and, consequently, road-network constrained trajectories. For instance, we can use cubic spline interpolation to model the curved shapes of road networks [12]. However, the most common representation of a road-network constrained trajectory is the more straightforward model defined in Section 2.1, which applies concepts of graph theory. From that definition, a road-network constrained trajectory $\mathrm{T} = (t_1, t_2, \ldots, t_n)$ is an ordered set of connected roads (i.e., a path on the graph) that represents a route traveled by a vehicle. Additional data can improve this representation. For instance, the trajectory data can also include the offsets that describe the vehicle's initial position at the first point t_1 and final position at the last point t_n, the times of the beginning and end of the trajectory, and the mean speeds of the vehicle in each road.

2.2.3 Other Trajectory Representations

Trajectories datasets are usually large, which can bring challenges to data analysis tasks when using representations like GPS-based trajectories or road-network constrained trajectories. Therefore, in the last years, some authors have proposed dimensionality reduction, hashing, and other codifications to represent trajectories better and process large scale datasets [33, 114, 154, 167].

Dimensionality Reduction

Road-network constrained trajectories have, by default, a lower dimension when compared with GPS-based trajectories. This difference is because the positions of the former are of only one dimension (i.e., the identifier of the road segment), as opposed to the 2D geospatial coordinates of raw trajectories. Instead of assigning random identifiers to the roads, one can use the Hilbert curve [83] to have a notion of space ordering of the road segments, which can facilitate queries and indexing. This is the idea behind the method proposed by Pfoser and Jensen [154] to reduce the dimensionality of trajectories. First, they sort all network edges based on the Hilbert value of the middle point of the edge. Then, they map all edges into sub-internals according to their ordering. The first edge becomes the first sub-interval, which extends a distance that corresponds to its distance in the road network. The second edge then starts where the first ends, and so on. Finally, they map the trajectories from polylines to this new movement space by identifying the edge where the vehicle was and how much of this edge the vehicle traveled. This method can reduce the size of the trajectories as well as the dead space in the indexes. However, a problem not discussed by the authors is how to map the movements of the vehicles to the road network.

Binary-Encoded Trajectories

We usually express locations in real-world road networks by using hierarchical administrative districts, such as city, road name, and road number. This perception of the hierarchy is not taken into account when we use space-filling curves to represent trajectories. Lee et al. [114] proposed a location encoding method that converts the positions into a binary string and has the notion of hierarchy. Their approach consists of three steps. The first step retrieves the address of the vehicle's location and expresses it as a triplet (district, road, and location on the road). Then, the second step converts the tree address fields into three binary strings, and finally, the last step joins the binary

strings. They use R-tree [76] to find the closest road segments to each position and get their address, and then, the Z-order curve [156] to encode the address as binary strings. One drawback of this representation is that the trajectory data size can be large, as all positions are encoded and concatenated without any compression. Equation 2.1 illustrates a 16-bits binary-encoded location consisting of four components, each one of four bits. When using this representation, for each scenario, one needs to carefully design the size of each component of the string so as not to waste memory. For instance, all road segments should identify the same amount of locations, and thus, we need to divide the longer segments into smaller ones so that every segment has similar sizes.

$$\text{Binary-Encoded Location} = \underbrace{0100}_{\text{Country}}\ \underbrace{0111}_{\text{District}}\ \underbrace{1001}_{\text{Road}}\ \underbrace{0010}_{\text{Location on Road}} \tag{2.1}$$

Hash-based Trajectories

Geohash is a widely-used public domain latitude/longitude geocoding system invented by Gustavo Niemeyer [143]. In summary, a geohash is a hierarchical spatial data system that uses the z-order space-filling curve to map a point to a binary string. It works by recursively dividing the space in half by straight lines, both horizontally and vertically, until the squares bounded by the lines are small enough. The number of divisions performed defines the accuracy of the geohash. Then, one can convert the sequence of bits to an alphanumeric string by using a variant of the base 32 transfer encoding. Among other features, geohashes allow precision control, its prefixes for nearby positions are similar, and one can remove some characters from the end of the geohash to save storage space. Because of its properties, recent proposals use geohash to encode trajectory data [15, 33].

Chapuis and Garbinato [33] proposed a method called geodab that combines hashing and geohashing [143] to represent trajectories in a way that facilitates the processing of dense trajectory datasets. A geodab is constructed from a sequence of points (i.e., from a GPS-based trajectory) as follows. First, it computes a prefix and a suffix for the geodab. The prefix is the geohash of the area that contains all points of the trajectory, and its objective is to distribute different geodabs on the Z-order curve [156] taking into account the sample locations. The suffix is another hash, and it aims to reflect the ordering of the points. Finally, the hash and geohash are concatenated and encoded as a 32 bits hash. The length of the hash can change according to the application requirements. As it includes both hash and geohash, the 32-bits hash that represents the geodab can

distinguish the trajectories considering both their paths and points ordering. The authors [33] demonstrated how geodabs and a fingerprinting algorithm, called winnowing [169], can index trajectory datasets. Finally, they presented a scalable method to distribute indexes of trajectories across different clusters.

Application-based Trajectories

For specific scenarios, it is better to design the trajectory representation according to the application. Tiesyte and Jensen [181] proposed a representation that is intended, for example, to compare trajectories of collective transport. Their model assumes that the vehicle locations are pre-defined and only takes into account the temporal component of the trajectory. In other words, a trajectory is a function that receives as input a location and provides as output the time the vehicle took to reach this location. Besides that, this representation allows the construction of predictive similarity measures, that is, similarity measures that assume that past similar trajectories are also similar in the future.

Deep Representation Learning

Representation learning is a process that transforms input data to facilitate analysis tasks such as building predictors. The idea is that the new representation has additional properties and preserves most information from the original data. We call deep representation learning the techniques composed by a set of non-linear transformations [22].

Recently, many studies proposed the application of representation learning for trajectories [65, 117, 203]. For instance, Li et al. [117] introduced a novel deep learning-based approach for trajectory representation. Their method, called t2vec, can perform similarity computations of low-quality data accurately and efficiently. The t2vec uses historical GPS-trajectory information and deep learning techniques to infer and describe the route information of a trajectory. After processing the trajectories to learn their representations, the time complexity to calculate the similarity between them is linear. In their experiments, the t2vec framework showed to be more accurate and efficient than the similarity measures EDR [38], LCSS [190], and EDwP [159]. Trajectory Embedding via road networks (Trembr) [65] is another deep learning-based approach for trajectory representation. The main difference between Trembr and other related methods (e.g., t2vec) is that Trembr incorporates the underlying road network. To do this, they use

a map-matching technique to map the sample points of raw trajectories onto the road network, which then facilitates and constrains the learning process.

2.3 Chapter Remarks

In this Chapter, we introduced some basic definitions used in the remainder of this thesis. First, we defined a model to represent the road network and the two most common trajectory representations. Then, we discussed these trajectory representations and alternative representations in more detail, as it is a critical component of trajectory data mining and the evaluation of vehicular networks.

Chapter 3

Vehicle Trajectory Similarity: Models, Methods, and Applications

The increasing availability of vehicular trajectory data is at the core of smart mobility solutions. Such data offer us unprecedented information to develop trajectory data mining-based applications. An essential task of trajectory analysis is employing efficient and accurate methods to compare trajectories. This Chapter presents a systematic survey of vehicular trajectory similarity measures and provides a panorama of the research field. First, we comprehensively review methods to compare trajectories and their intrinsic properties. Then, we classify the methods according to the trajectory representation and features such as metricity, computational complexity, and robustness to noise and local time shift. Last, we discuss the applications of vehicular trajectory similarity measures and some open research problems.

3.1 Introduction

Mobility aspects of society are changing rapidly. As the cities grow and their mobility requirements change, it is necessary to use new technologies and transportation modes to move people and other loads like goods and animals in a smart way, leading to smart transportation solutions [21]. Furthermore, the development of novel Information and Communication Technologies (ICTs) is leading to a rise in the availability of mobility datasets, which are acquired using several data sources. For instance, people can record their real-world movements by carrying a mobile phone that generates spatial trajectories, logging their travel routes, and posting geotagged photos or making "check-ins" in

social networks. Vehicles equipped with location acquisition devices can also generate a massive amount of trajectory data. Currently, the Global Positioning System (GPS), Europe's Galileo, and other Global Navigation Satellite Systems (GNSS) are the leading technologies to provide autonomous geospatial positioning [56].

Vehicle trajectory data offer us crucial information for applications in different research fields (e.g., urban planning, traffic analysis [44], location-based social networks, data mining, and vehicular networks [48]). For instance, the analysis of popular routes is useful for both route recommendations and the city's road development planning. Furthermore, traffic monitoring systems can predict traffic congestion regions using historical data.

A fundamental task of trajectory analysis is the comparison of trajectories. Based on similarity measures, trajectories can be clustered, classified, and retrieved [128]. Currently, there are several methods to calculate the similarity (alternatively, the distance) between two or more trajectories, each one best suited for a particular scenario. For some data analysis tasks, such as clustering, choosing a proper similarity measure plays an important role [17]. Moreover, the representation of a trajectory influences the choice of the most indicated method.

A sequence of raw GPS data is the most common vehicle trajectory representation. Trajectories adopting this representation are called positioning-based trajectories. However, receivers of satellite-based positioning systems introduce errors to the recorded locations. These errors are more common in metropolitan regions due to low satellite visibility and phenomena that affect signal quality, such as strength attenuation and interference [106]. Therefore, preprocessing techniques need to be employed to reconstruct a road-network constrained trajectory [215]. For instance, map-matching is a technique to snap each trajectory sample to the road network, and filling the gaps aims at removing the uncertainty between consecutive trajectory samples. For this representation, there is another set of methods to compare the trajectories. In the remainder of this chapter, we adopt the terms *GPS-based trajectory* and *positioning-based vehicular trajectory* interchangeable.

The trajectory similarity research field has been in evidence in the last couple of years. However, most of the proposals in this area deal with the similarity of trajectories of objects of any kind (e.g., hurricanes, people, vehicles). Therefore, there is a lack of a systematic review that provides a detailed view of the field addressing the intrinsic characteristics of terrestrial vehicular trajectories, such as the geographic restriction of this type of trajectory to the physical definitions of the road network. To this end,

this Chapter presents a survey of the methods and applications of vehicle trajectory similarity.

3.2 Literature Review

The majority of the reviews presented in the past few years on trajectory similarity focus on general GPS-based trajectories and their widely used similarity measures (i.e., ED, DTW, LCSS [190], ERP [37], and EDR [38]). They do not take into account the different set of trajectory representations and similarity measures for the case of vehicle trajectories. Besides that, these surveys do not deliberate about the applications and the suitability of the similarity measures. Table 3.1 summarizes these studies, their goals, and methods covered.

Wang et al. [191] presented an experimental study aiming at comparing popular trajectory distance functions (ED, DTW, PDTW [99], LCSS, ERP, and EDR). The focus of the study is on similarity measures for GPS-based trajectories. The purpose was to evaluate the effectiveness of the measures and demonstrate their advantages and drawbacks in different circumstances. To do this, they first created a dataset composed of two sets of trajectories. The first set is the *original trajectories*, while the second one, called *transformed trajectories*, is constructed by changing the sampling rates and adding noise or shift to the *original trajectories*. They generated the transformed trajectories in such a way that, if employing a suitable similarity measure, the similarity between both sets of trajectories should be lower when the amount of transformation is higher, and higher when the amount of transformation is lower. Wang et al. [191]x concluded that each similarity measure deals better with a specific set of transformations. For instance, although ED is very sensitive to noise, it is a suitable method for situations where the compared trajectories are similar in terms of sampling interval and amount of point shift. DTW-based methods exhibited results similar to those of ED. Lastly, LCSS is sensitive to point shift modifications and robust to changes in the sampling rates and outliers.

Magdy et al. [128] presented another comparative study between trajectory similarity measures. However, unlike Wang et al. [191], the paper focuses on the characteristics of each method instead of presenting an experimental study. First, the authors classified each method evaluating if it considers both the temporal and spatial characteristics of the trajectory or just the spatial characteristics. For the Spatio-temporal similarity methods, a further classification level informs if it takes into account the vehicle movement speeds, or if it employs specific time series analysis techniques. Finally, they classified Spatial

Similarity methods regard the use of the trajectories spatial data, geometric shape, or movement direction. Besides that, they compared the measures concerning their computational cost, whether the measure is metric or not, can handle trajectories of different lengths, and their robustness to deal with noise and local time-shifting. They compared measures such as ED, DTW, ERP, EDR, LCSS-based approaches, Spatial Assembling Distance, Hausdorff distance, Fréchet distance, and Trajectory Match Algorithm.

Yu Zheng [215] conducted a systematic survey on trajectory data mining to provide a comprehensive view of this research area. On the issue of trajectory similarity, the author addressed the main methods to estimate the similarity (alternatively, the distance) between trajectories. The article summarizes some well-known methods to compute the distance between GPS-based trajectories, such as closest-pair distance, DTW, LCSS-based, EDR, and ERP. Then it describes two measures of distance between trajectory segments, named Minimum Bounding Rectangles (MBR) distance and Trajectory-Hausdorff distance.

Toohey and Duckham [183] compared four trajectory distance functions: LCSS, Fréchet distance, DTW, and EDR. The authors coded the methods in a package using the R programming language and made the package available for free. The comparison between the measures was performed through experiments with a dataset of delivery drivers in the United Kingdom. The results showed a strong correlation between the values of the Fréchet distance and DTW, and between EDR and LCSS.

Besse et al. [23] tackled the issue of how to cluster GPS-based trajectories using the distances between them. They provided a review of four similarity measures (DTW, LCSS, EDR, and ERP). Also, they presented a novel distance function called Symmetrized Segment-Path Distance (SSPD). The authors implemented the five methods in a python package and made it available for free.

This survey differs from the studies mentioned above once our focus is to present and discuss trajectory analysis techniques regarding different trajectory representations. At the same time, the related work typically covers methods for raw GPS-based trajectories. This perspective is fundamental since any investigation in this area should start with the trajectory representation and, then, proceed in the study of a particular research issue but considering all aspects defined for that representation. Thus, for each representation, we provide a discussion regarding the state of the art of the corresponding methods. Finally, we highlight and discuss various applications that depend on similarity measures. In this way, we advance the understanding of this research area.

3.3 Methods

Several methods are available to measure the similarity of vehicle trajectories. Each method has a specific set of properties. We present the properties that trajectory similarity measures can have in Section 3.3.1. Furthermore, the most proper method depends on how we represent trajectories. In Section 3.3.2, we present a comprehensive review of the methods to compare GPS-based trajectories and, in Section 3.3.3, we review the similarity measures for road-network constrained trajectories.

3.3.1 Properties

Metricity We can classify similarity measures according to whether or not it is a metric. The advantage of being a metric is that the measure can be indexed directly by known distance-based indexing techniques. We call a distance function $D(A, B)$ (associated with a given similarity measure) as a metric if, given trajectories A, B, and C, it satisfies the conditions below.

- Uniqueness: $D(A, B) = 0 \Leftrightarrow A = B$
- Nonnegativity: $D(A, B) \geq 0$
- Symmetry: $D(A, B) = D(B, A)$
- Triangle Inequality: $D(B, C) \leq D(A, B) + D(A, C)$

Efficiency Trajectory similarity is a crucial component of data mining tasks, and in most cases, it is used to search for trajectories in massive datasets. Thus, it is primordial to have methods of low computational complexity to calculate the similarity between trajectories.

Local time shifting Vehicles are allowed to move at different speeds, and location-acquisition devices can acquire data at different sampling rates. For instance, two trajectories can follow the same path but at different timestamps. The shifts in the sample acquisition times can even be only in some parts of the trajectories. Thus, vehicle trajectory similarity measures need to be able to take into account local time shifts when computing the distance between trajectories.

Different lengths Vehicular trajectories can have different lengths. Thus, similarity measures need to be able to handle trajectories that do not have the same number of samples, if they are GPS-based trajectories, and that have a different number of road segments if they are road-network constrained trajectories.

Robustness to noise and outliers Most of the noise in GPS-based trajectories come from the fact that satellite-based positioning systems have precision errors. For instance, these errors are more common in metropolitan regions due to low satellite visibility and phenomena that affect signal quality [106]. In addition to noise, anomalies in the sensor collecting the data might introduce outliers into GPS-based trajectories. For methods based mostly on points distance, these outliers can result in similarity measures that are quite different from reality. One solution for this is to apply preprocessing techniques to remove outliers from the trajectories before comparing them. As for the noises, one drawback of most solutions that are robust to this is that they violate the triangle inequality because they do not take into account the different sub-trajectories equally [190].

Parameter-free Some methods introduce parameters to compare trajectories, such as a threshold to match similar geospatial points. The problem with approaches like the aforementioned is that the parameters need to be adjusted a priori per each scenario.

Completeness It compares trajectories as a whole instead of just segments of the trajectories.

3.3.2 Methods for GPS-based trajectories

GPS-based vehicular trajectories are a particular case of time series, and thus most of the similarity measures to compare them were initially proposed to compare time series. However, we cannot directly apply similarity measures for time series to compare GPS-based trajectories due to the unique characteristics of the later. For instance, vehicular trajectories usually have two or three dimensions, while most of the measures to compare time series focus on one-dimensional data. Besides that, GPS-based trajectories can have many outliers because of precision errors in satellite-based positioning systems. Another difference is that the different sampling rates, speeds, moving patterns, and regions of vehicle trajectories may introduce local time shifts into it. Thus, we need to adapt the similarity measures for time series before using it to compare vehicular trajectories.

Table 3.2 compiles the main notations used throughout this chapter.

Euclidean Distance (ED)

Euclidean Distance (alternatively, L_2 Norm) is a well-known function commonly used to calculate the distance between time series. In the context of vehicle trajectories, the euclidean distance between two trajectories of the same length is the average of the distances between ordered pairs of points of the two trajectories. Formally, given trajectories T_1 and T_2, the Euclidean distance between T_1 and T_2 is:

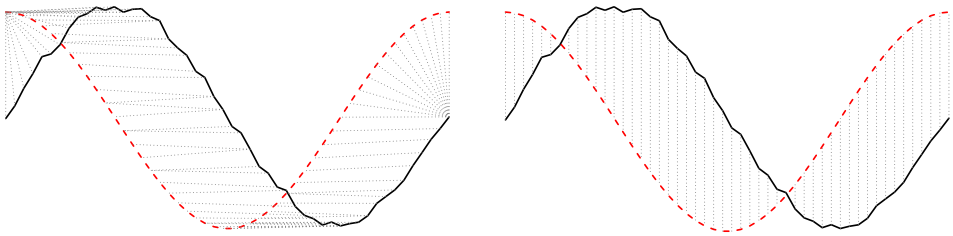
$$ED\left(T_1, T_2\right) = \frac{\sum_{i=1}^{n} dist\left(\rho_i^1, \rho_i^2\right)}{|T_1|} \tag{3.1}$$

The main drawback of ED, besides not being able to compare differently sized trajectories, is that it cannot handle local time-shifting. Instead, it compares trajectory points one by one by using their indexes. As the compared points may be far apart from each other, especially considering trajectories of different speeds, ED cannot handle trajectories of different sizes and struggles with outliers and noise. On the other hand, ED is a metric, and so is easily indexable. Furthermore, ED is parameter-free, which makes it a suitable option to compare long trajectories.

Dynamic Time Warping based Measures (DTW)

Like ED, DTW is a distance measure originally proposed to compare time series. DTW finds the match of minimal cost, where each point of the first trajectory is paired to one or more consecutive points of the second trajectory while respecting some restrictions. Moreover, DTW allows the comparison between trajectories of different lengths, and its original definition is also parameter-free. Although DTW is non-metric, many studies in the literature propose approximate and exact indexing techniques to provide a fast search of trajectories [98]. Figure 3.1 illustrates DTW in comparison with the Euclidean Distance.

Given two vehicular trajectories T_1 and T_2, one can compute its distance using Dynamic Time Warping $DTW(T_1, T_2)$ using Equation 3.2.



(a) Matching using Dynamic Time Warping.

(b) Matching using Euclidean Distance.

Figure 3.1: Unlike ED, DTW supports shrinkage or stretching of trajectories in the time axis to provides a better adjustment of the points, and thus represents a more sophisticated distance measure.

$$DTW(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = |T_2| = 0 \\ \infty, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ dist(Head(T_1), Head(T_2)) + \min \begin{cases} DTW(Rest(T_1), T_2) \\ DTW(T_1, Rest(T_2)) \\ DTW(Rest(T_1), Rest(T_2)) \end{cases}, & \text{otherwise} \end{cases} \tag{3.2}$$

Warping path is the sequence of steps that computes the value of $DTW(T_1, T_2)$. As this process has optimal substructure and overlapping sub-problems, it is implemented using dynamic programming. The computational complexity of DTW is $O(|T_1| \cdot |T_2|)$, which makes it much slower than other similarity measures, such as the Euclidean Distance. Thus, many studies proposed pruning methods to makes faster the similarity-based queries that use DTW [98, 99, 166].

Keogh and Pazzani [99] proposed the Piecewise Dynamic Time Warping (PDTW) to compute approximate values of the DTW distance efficiently. Their technique reduces the size of the time series by a piecewise aggregate approximation and, consequentially, makes the DTW faster by a constant factor that depends on input data, but typically one or two orders of magnitude. They showed through experiments that PDTW provides accurate approximations of the DTW distance.

FastDTW [166] is an approximation algorithm of DTW that uses a multilevel approach starting from a downsampled version of the time series and iteratively performs the following steps. First, it finds the warping path on the lower resolution grid. Then,

it projects the solution on a higher resolution grid and defines a first warp path band which is adjustable with a radius r parameter. The last step of the iteration is to refine the solution by evaluating the higher-level grid limited to this band. Both the time and space complexities of FastDTW are linear.

Mao et al. [129] proposed the Segment-based Dynamic Time Warping (SDTW), which is the integration of three distance measures with the traditional DTW algorithm. Given the points ρ_i^1 of trajectory T_1 and ρ_j^2 of trajectory T_2, the point-segment distance between them is their spatial distance taking into account their neighboring points and is used to reduce the sensitivity of the method to the sampling rate of the trajectories. Given that ρ_i^1 has an earlier timestamp and ρ_j^2 has a later timestamp, the prediction distance between them is the distance between ρ_i^1 and the approximate position of ρ_j^2 at the earlier timestamp. Finally, the segment-segment distance combines an adjustable parameter ω and the spatial, temporal, and angle distances. According to Mao et al. [129], the prediction distance employs the trajectory data features to optimizes the temporal distance. Also, the segment-segment distance is aware of the trajectory shape and thus improve the similarity measure accuracy. Despite its good accuracy, SDTW presents a high time complexity equal to $O(\log(|T_1| + |T_2|) \cdot |T_1| \cdot |T_2|)$.

Edit Distance-based Measures

The concept of *edit distance* was initially proposed to compare strings and then was employed to measure the similarity of time series. In the context of strings, edit distance is the smallest amount of operations (insertion, deletion, and replacement) required to make both strings equals [139]. We call *simple edit distance* (or edit distance) if all operations cost one, and *general edit distance* if the operations have different costs, or if the costs change according to the characters associated with the operation. Finally, if only insertions and deletions at cost one are allowed, we call it the Longest Common Subsequence (LCSS).

To compare trajectories using the LCSS, we usually define a threshold to match points that are close to each other. Vlachos et al. [190] proposed an LCSS-based algorithm to estimate the similarity between trajectories. It is non-metric and relies on two parameters, named δ and ϵ. δ controls how far in time a point of the first trajectory can go to match a point of the second one, and ϵ is the matching threshold. Formally, the LCSS between a pair of trajectories as presented by Vlachos et al. [190] is:

$$LCSS_{\delta,\epsilon}(T_1,T_2) = \begin{cases} 0, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ 1 + LCSS_{\delta,\epsilon}(Rest(T_1), Rest(T_2)), & \text{if } dist(Head(T_1), Head(T_2)) < \epsilon \\ & \text{and } ||T_1| - |T_2|| \leq \delta \\ \max \begin{cases} LCSS_{\delta,\epsilon}(Rest(T_1), T_2) \\ LCSS_{\delta,\epsilon}(T_1, Rest(T_2)) \end{cases}, & \text{otherwise} \end{cases} \tag{3.3}$$

Finally, they normalize the LCSS by dividing it by the number of samples of the shortest trajectory. The method proposed by Vlachos et al. is *symmetric*, that is, $LCSS_{\delta,\epsilon}(T_1,T_2)$ is equal to $LCSS_{\delta,\epsilon}(T_2,T_1)$, and has time complexity equal to $O(\delta \cdot (|T_1| + |T_2|))$ (although that of LCSS originally is $O(|T_1| \cdot |T_2|)$).

Edit Distance on Real Sequence (EDR) [38] is a similarity measure with characteristics close to the LCSS. The main difference between LCSS and EDR is that EDR assigns penalties when a point on one trajectory does not match the point on the other trajectory. Formally, EDR between two trajectories is:

$$EDR(T_1,T_2) = \begin{cases} |T_2|, & \text{if } |T_1| = 0 \\ |T_1|, & \text{if } |T_2| = 0 \\ \min \begin{cases} EDR(Rest(T_1), Rest(T_2)) + subcost \\ EDR(Rest(T_1), T_2) + 1 \\ EDR(T_1, Rest(T_2)) + 1 \end{cases}, & \text{otherwise} \end{cases} \tag{3.4}$$

where $subcost = 0$ if $dist(Head(T_1), Head(T_2)) < \epsilon$ and $subcost = 1$ otherwise.

Like other edit distance-based methods, we can apply dynamic programming to compute EDR and its time complexity is quadratic ($O(|T_1| \cdot |T_2|)$), which makes using it with sequential scan on large databases a slow procedure. Besides that, EDR uses a threshold ϵ to reduce the effects of noise. EDR is not a metric because the employment of the threshold makes it violates the triangle inequality. Therefore, we can not use distance-based indexing techniques to improve trajectory search using the EDR distance.

Chen and Ng [37] presented the Edit Distance with Real Penalty (ERP), which is metric and so can cope with some indexing problems of other methods. Besides that, ERP supports local time shifting. The idea of ERP is to use a constant reference point g instead of a threshold ϵ to compute distances. Chen and Ng [37] proved that ERP

satisfies the triangle inequality regardless of the value of g. However, they suggested using g equal to zero. Equation 3.5 defines ERP.

$$ERP(T_1, T_2) = \begin{cases} \sum_{i=1}^{|T_2|} dist(\rho_i^2, g), & \text{if } |T_1| = 0 \\ \sum_{i=1}^{|T_1|} dist(\rho_i^1, g), & \text{if } |T_2| = 0 \\ \min \begin{cases} ERP(Rest(T_1), Rest(T_2)) + dist(Head(T_1), Head(T_2)) \\ ERP(Rest(T_1), T_2) + dist(\rho_1^1, g) \\ ERP(T_1, Rest(T_2)) + dist(\rho_1^2, g) \end{cases} & , \text{ otherwise} \end{cases} \tag{3.5}$$

ERP has the same computational behavior of EDR, LCSS, and DTW, and thus its time complexity is $O(|T_1| \cdot |T_2|)$. However, unlike the other distance functions, ERP is a metric, so it can be used to prune unnecessary trajectories by applying the triangle inequality.

Time Warp Edit Distance (TWED) [130] is a distance measure presented as an alternative to ERP as it has the same features of ERP. The main difference between them is that TWED employs the samples acquisition times during the computations. TWED introduces two parameters, ν and λ. The first parameter, ν, applies higher penalties to matched points that are farther away from each other concerning their acquisition times, and λ is the penalty for the deletions. Formally, the TWED between two trajectories is:

$$TWED_{\lambda,\nu}(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = |T_2| = 0 \\ \infty, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ \min \begin{cases} TWED_{\lambda,\nu}(Rest(T_1), T_2) + \Gamma(\rho_1^1) \\ TWED_{\lambda,\nu}(T_1, Rest(T_2)) + \Gamma(\rho_1^2) \\ TWED_{\lambda,\nu}(Rest(T_1), Rest(T_2)) + \Gamma(\rho_1^1, \rho_1^2) \end{cases} & , \text{ otherwise} \end{cases} \tag{3.6}$$

where

$$\Gamma(\rho_i^j) = dist(\rho_i^j, \rho_{i+1}^j) + \nu \cdot (\rho_i^j.t - \rho_{i+1}^j.t) + \lambda \tag{3.7}$$

$$\Gamma(\rho_i^1, \rho_j^2) = dist(\rho_i^1, \rho_j^2) + dist(\rho_{i+1}^1, \rho_{j+1}^2) + \nu \cdot (|\rho_i^1.t - \rho_j^2.t| + |\rho_{i+1}^1.t - \rho_{j+1}^2.t|) \tag{3.8}$$

and $\rho_{|T_i|+1}^i$ (one index after the last point of the trajectory) represents a point with value zero.

Edit Distance with Projections (EDwP) [159] is a parameter-free distance function aimed at trajectories that have different sampling rates. The first step of EDwP is to convert the GPS-based trajectory into line segments through linear interpolation. Then, given two trajectories T_1 and T_2 that are presented as segment sequences, the EDwP between them is the least expensive set of *insert* and *replace* operations that make them identical. Like most edit distance based functions, EDwP takes into account local time shift and has a quadratic computational cost. On the other hand, EDwP robust against sampling rate variations. Equation 3.9 formally defines the EDwP.

$$EDwP(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = |T_2| = 0 \\ \infty, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ \min \begin{cases} EDwP(Rest(T_1), Rest(T_2)) + RepCost(T_1, T_2) \\ EDwP(Ins(T_1, T_2), T_2) \\ EDwP(T_1, Ins(T_2, T_1)) \end{cases}, & \text{otherwise} \end{cases} \tag{3.9}$$

where

$$RepCost(T_1, T_2) = Rep(T_1.e_1, T_2.e_1) \times Coverage(T_1.e_1, T_2.e_1) \tag{3.10}$$

$$Rep(e_1, e_2) = dist(e_1.s, e_2.s) + dist(e_1.e, e_2.e) \tag{3.11}$$

$$Coverage(e_1, e_2) = length(e_1) + length(e_2) \tag{3.12}$$

and e_i represents a line segment created by the linear interpolation between two consecutive points. $e_i.s$ and $e_i.e$ represent the first and last points, respectively, of the line segment e_i. $T_i.e_1$ represents the first line segment of the trajectory T_i. Finally, $Ins(T_1, T_2)$ is an operation to aid robust matching that returns a modified version of T_1. It works by introducing one extra point between the first two points of T_1. Thus, $Ins(T_1, T_2)$ effectively divides the first line segment of T_1 ($T_1.e_1$) into two segments. The split point is the point on $T_1.e_1$ that is spatially closest to the endpoint of the first line segment of T_2 ($T_2.e_1.e$).

The results of EDwP depends on the length of the trajectories, as it is the sum of every operation. Therefore, in certain situations, it is better to normalize the results of EDwP. To do this, we divide the EDwP by the sum of the lengths of trajectories T_1 and T_2.

Fréchet Distance

Fréchet Distance [62], illustrated in Figure 3.2, is one of the most popular similarity measures and falls within the category of geometric shape based measures. A classic example to illustrate Fréchet distance between two trajectories is that of a man walking his dog. The man is traversing a finite curved path while walking his dog on a leash, and the dog is traversing a different one. Both man and dog may have different speeds at different times, but they cannot move backward. The Fréchet distance is the length of the smallest leash sufficient for both to traverse their separate paths. The advantage of the Fréchet distance is that it can compare trajectories that have different sizes and sampling rates. However, the Fréchet distance is not robust to noise and outliers, nor it tackles local time shifting.

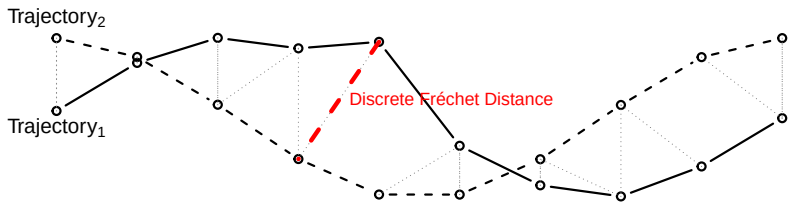


Figure 3.2: The bold dashed line represents the discrete Fréchet distance between the trajectories. In this example, the discrete Fréchet distance coincides with the Hausdorff distance.

The concept of Fréchet distance is for continuous curves, but for some applications (e.g., analysis of vehicle trajectories), it is often useful to use polygonal curves. Alt and Godau [7] presented an exact algorithm based on a parametric search that receives as input two polygonal curves and computes their Fréchet distance in $O(|T_1| \cdot |T_2| \cdot log(|T_1| \cdot |T_2|))$. To address the high computational complexity of the algorithm proposed by Ant and Godau, Eiter and Mannila [57] presented a discrete version of the Fréchet distance for polygonal curves called *coupling distance*. The idea of coupling distance is to look at all possible couplings between the endpoints of the polygonal curves line segments. Eiter and Mannila [57] showed that the coupling distance is a good approximation to the Fréchet distance and presented an algorithm that computes it in $O(|T_1| \cdot |T_2|)$.

Hausdorff Distance

Hausdorff distance, also called Pompeiu-Hausdorff distance, is a metric measure similar to Fréchet distance. In the context of GPS-based trajectories, it is the longest of all distances from a location in one trajectory to the nearest location in the other trajectory, and vice versa. The main difference between Fréchet and Hausdorff distances is that the order of points affects the former distance but not later one. In Figure 3.2, the Hausdorff distance coincides with the discrete Fréchet distance. Figure 3.3 illustrates the difference between the two distance measures.

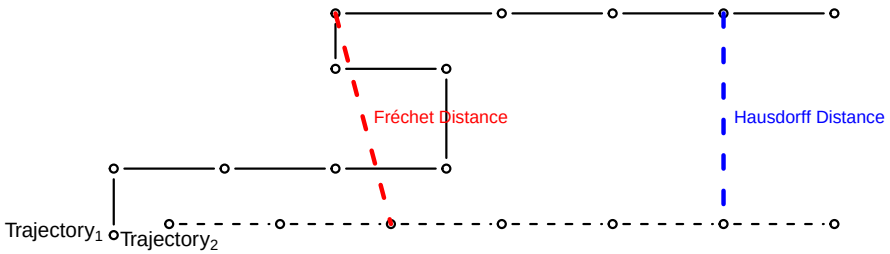


Figure 3.3: The difference between Hausdorff distance and Fréchet distance.

Equation 3.13 defines the Hausdorff distance $Dist_{Hausdorff}$ between two GPS-based trajectories T_1 and T_2. The computational complexity of the Hausdorff distance is $O(|T_1| \cdot |T_2|)$, the same as that of discrete Fréchet distance.

$$Dist_{Hausdorff}(T_1, T_2) = \max \left\{ \max_{\rho_i^1 \in T_1} \min_{\rho_j^2 \in T_2} dist(\rho_i^1, \rho_j^2), \max_{\rho_j^2 \in T_2} \min_{\rho_i^1 \in T_1} dist(\rho_i^1, \rho_j^2) \right\} \tag{3.13}$$

Other methods

Other methods are designed by considering the actual applications. Chen et al. [40] proposed a similarity function that combines the characteristics of LCSS, in the sense that it can ignore samples that do not have good match candidates in the other trajectory, and DTW, which can match a trajectory point with more than one point of the other trajectory. It works by matching each point of one trajectory to only one point of the other trajectory. Therefore, Chen et al. [40] designed their method for situations where the first trajectory represents a few query locations, and the object is to verify whether the query locations match with the trajectory instead of verifying whether the trajectory

and the query locations are similar in shape. Given two trajectories T_1 and, T_2, their similarity measure as proposed by Chen et al. [40] is:

$$Sim_o(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ \max \begin{cases} e^{-dist(Head(T_1), Head(T_2))} + Sim_o(Rest(T_1), T_2) \\ Sim_o(T_1, Rest(T_2)) \end{cases}, & \text{otherwise} \end{cases} \tag{3.14}$$

With the exponential function $e^{-dist(Head(T_1),Head(T_2))}$, points that are closer to each other receive a much higher similarity value compared to those that are far away. In other words, the contribution exponentially decreases as $dist(Head(T_1), Head(T_2))$ linearly increases.

Ta et al. [178] presented a bi-directional similarity (BDS) measure to support solutions of the trajectory similarity join problem. Trajectory similarity join is the task of finding every pair of similar trajectories from different datasets. BDS works by aligning each point of one trajectory to the nearest location on the other trajectory, which is not necessarily a sample of the trajectory, and vice versa. The overall complexity of BDS is $O(|T_1| \cdot |T_2|)$.

Other methods use the shape formed by the line strings of two trajectories to compute the distance between them. For instance, the Locality In-between Polylines (LIP) [152] sums the areas of all polygons formulated between intersection points, which in turn are generated by the overlay of the trajectories in the two-dimensional plane. They use linear interpolation between consecutive points to represent the line segments. Besides that, LIP uses the length of the segments as weights to the areas of the polygons. Equation 3.15 defines LIP.

$$LIP(T_1, T_2) = \sum_{\forall polygon_i} Area_{polygon_i} \cdot w_i \tag{3.15}$$

Pelekis et al. [152] proposed, in addition to LIP, a time-aware distance function called STLIP that takes into account the time factor. In summary, STLIP is a multiple of LIP weighted by a penalty defined by the user that adds the time factor to the distance function. The time complexity of LIP and STLIP is $O(N \cdot \log(N))$, where $N = |T_1| + |T_2|$.

NeuTraj [202] is a generic approach based on neural metric learning that accelerates approximate trajectory similarity computations for any measure discussed above. In summary, NeuTraj uses the pair-wise similarities of a sample of a given trajectory

database as guidance to a recurrent neural network (RNN) that approximates a generic distance function, such as DTW, Hausdorff, and ED. Once the neural network is constructed, NeuTraj computes the approximate distance between two trajectories in linear time.

Table 3.3 summarizes the similarity methods for GPS-based trajectories and their properties.

3.3.3 Methods for road-network constrained trajectories

The most common representation of road-network constrained trajectories is as an ordered set of connected road segments from a given road network. Therefore, some proposals use operations from set theory to measure the similarity between these trajectories [196, 200]. However, instead of using the number of segments in common, we can use the lengths of each segment. For instance, Won et al. [196] presented a dissimilarity measure called Dissimilarity with Length (DSL) to cluster road-network constrained trajectories. The DSL between two trajectories is the sum of the road lengths of their disjoint set, divided by the sum of the lengths of both trajectories. Formally:

$$DSL(T_1, T_2) = \frac{L_d(T_1, T_2)}{L_s(T_1) + L_s(T_2)} \tag{3.16}$$

where $L_d(T_i, T_j)$ is the sum of the road lengths of the disjoint set of T_i and T_j, and $L_s(T_i)$ is the total length of the roads of T_i.

A drawback of the method proposed by Won et al. [196] is that it does not identify the similarity of trajectories that are close to each other but do not have segments in common. Besides that, the DSL method does not take into account the trajectories timestamps nor their speeds.

Xia et al. [200] proposed a method based on the Jaccard similarity coefficient [179] to calculate the spatiotemporal similarity between road-network constrained trajectories. Equation 3.17 defines its spatial component, where $L_c(T_1, T_2)$ is the sum of the road lengths contained in the intersection set of trajectories T_1 and T_2, and $L_s(T_i)$ is the sum of the road lengths of trajectory T_i.

$$SSim(T_1, T_2) = \frac{L_c(T_1, T_2)}{L_s(T_1) + L_s(T_2) - L_c(T_1, T_2)} \tag{3.17}$$

They compute the temporal component $TSim(T_i, T_j)$ of the similarity measure by replacing, in Equation 3.17, the spatial lengths L_s by lifespans of the temporal dimen-

sion L_t. The authors argue that the trajectories should be considered as spatiotemporal similar when the spatial and temporal similarities are both high. Thus, the spatiotemporal similarity measure $STSim(T_i, T_j)$ between trajectories is the product of the spatial similarity measure $SSim(T_i, T_j)$ and the temporal similarity measure $TSim(T_i, T_j)$.

The similarity measures presented by Xia et al. [200] and Won et al. [196] are similar, and thus have the same drawbacks. First, their methods do not identify the similarity of trajectories that are close to each other and that do not have segments in common. Their methods also do not take into account either the timestamps of the trajectories or the speeds of the vehicles. Finally, both methods have high computational complexity.

For particular applications, such as trajectories searching, we can compute the similarity between road-network constrained trajectories according to a collection of Points of Interest (POIs) or Times of Interest (TOIs). For instance, one can be interested in trajectories that passed through some places at some time intervals. Hwang et al. [88] proposed a spatial similarity measure and a temporal similarity measure for road-network constrained trajectories, which considers the (S_{id}, d, t) coordinate system instead of the Euclidean space, where S_{id} is a road segment identifier, t is the timestamp of the sampling point, and d is the distance between the current position and the starting location of the road segment. The authors [88] applied both spatial and temporal similarity measures in a trajectory search technique that uses the spatial similarity to select some trajectories and then employs the temporal similarity measure to improve the selection. They consider two trajectories similar if they pass through the same POIs. According to the authors, the POIs can be, for instance, relevant places defined according to application criteria. Few applications can use their method because of the necessity to define in advance the points of interest. Equation 3.18 defines the spatial similarity of trajectories T_1 and T_2 using a set of POIs P.

$$Sim_{POI}(T_1, T_2, P) = \begin{cases} 1, & \text{if } \forall p \in P,\ p \in T_1 \wedge p \in T_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.18}$$

Equations 3.19 and 3.20 define the temporal component of the similarity measures. Equation 3.19 computes the distance considering one point of interest, while Equation 3.20 considers a set of points of interest.

$$Dist_T(T_1, T_2, p) = |t(T_1, p) - t(T_2, p)| \tag{3.19}$$

$$Dist_T(T_1, T_2, P) = \left(\sum_{i=1}^{k} |t(T_1, p_i) - t(T_2, p_i)|^k \right)^{\frac{1}{k}} \tag{3.20}$$

where k is the size of the set of points of interest P and $t(T_i, p)$ represents the timestamp of the moment when T_i pass through the point of interest p. If p is neither contained in T_1 nor T_2, then it is assumed a infinity temporal distance.

Hwang et al. [89] extended their previous work [88] by introducing the concept of TOI on road networks. The TOI can represent, for instance, the peak hours where there are most traffic jams. The temporal similarity between two trajectories T_1 and T_2 using a set of TOIs T is:

$$Sim_{TOI}(T_1, T_2, T) = \begin{cases} 1, & \text{if } \forall t \in T,\, t \in [T_1.s, T_1.e] \wedge t \in [T_2.s, T_2.e] \\ 0, & \text{otherwise} \end{cases} \tag{3.21}$$

where $T_i.s$ and $T_i.e$ represent the times when the trajectory started and ended, respectively.

Hwang et al. [89] also proposed Equation 3.22 to calculate the trajectories spatial distance, taking into account the set of times of interest TOI.

$$Dist_{TOI}(T_1, T_2, TOI) = \sum_{i=1}^{k} dist(p(T_1, TOI_i), p(T_2, TOI_i)) \tag{3.22}$$

where $p(T_i, TOI_j)$ denotes the point of T_i in the timestamp TOI_j and k is the size of set TOI.

The measures proposed by Hwang et al. [88, 89] have several drawbacks. First, the fact that the sets POIs and TOIs need be defined in advance by users restricts their usage to few applications and may lead to erroneous conclusions if the POIs and TOIs are not adequately defined. Besides that, the similarity space (1, if the trajectories are similar, and 0 otherwise) only informs if the trajectories are similar or not. However, in most situations, it is better to know how similar the trajectories are.

Abraham and Lal [4] presented a spatiotemporal similarity measure to overcome the lack of similarity level notion of the approach proposed by Hwang et al. [88, 89]. Their method, which compares road-network constrained trajectories that are represented by a binary encoding scheme [114], takes into account the hierarchical components of the binary-encoded location. To this, they consider each location's component separately.

Besides the binary encoding scheme, they use dimensionality reduction [3] to manage the road network data.

Tiakas et al. [180] proposed another approach to overcome the drawbacks of Hwang' measures. They defined a set of distance functions (metrics). For instance, $D_{network}$ and D_{time} compare trajectories in space and in time, respectively, and D_{total} is the total (combined) distance, which is weighted by two parameters that define the importance of space and time factors. Formally, D_{total} is defined as:

$$D_{total}(T_1, T_2) = D_{network}(T_1, T_2) \cdot W_{network} + D_{time}(T_1, T_2) \cdot W_{time} \tag{3.23}$$

Like other similarity measures for road-network constrained trajectories, the method proposed by Tiakas et al. [180] describes the road network as a directed graph. In summary, the network distance $D_{network}$ computes routing distances taking into account the graph of the road network, and the time distance D_{time} computes the time required to travel between two consecutive nodes. The distance measures $D_{network}$, D_{time}, and D_{total} can handle only equally sized trajectories. To overcome this issue, they proposed a decomposition process that splits the trajectories into sub-trajectories of equal size, and then, they index the sub-trajectories by M-trees. Although all distance measures proposed by Tiakas et al. [180] are metrics, the introduction of the $W_{network}$ and W_{time} parameters is a disadvantage of their approach.

Aiming at big trajectory data clustering, Kumar et al. [111, 110] proposed a DTW-based method, called trajDTW, that employs the Dijkstra algorithm to compute road-network constrained trajectory similarities. The trajDTW defines a window parameter w, which is equal to half the size of the shortest trajectory, to avoid overestimation of the real distance. The authors demonstrated in [111] the superiority of the trajDTW over DSL and Hausdorff distance measures. However, to be faster, trajDTW requires the pre-computation of the distances between all pairs of edges in the road network [161].

Longest Common Road Segments (LCRS) [211] similarity measure is a version of the LCSS function adapted to road segments. First, LCRS computes the longest common road segments between two road-network restricted trajectories in terms of road segment lengths instead of the number of road segments (Equation 3.24). Next, it normalizes the length of these segments using a method similar to Jaccard (Equation 3.25). We can compute the LCRS between two trajectories using a dynamic programming algorithm with quadratic time complexity. Also, LCRS shares other features of LCSS, such as being able to compute the similarity between trajectories of different sizes, and not being a metric.

$$LCRS(T_1, T_2) = \begin{cases} 0, & \text{if } |T_1| = 0 \text{ or } |T_2| = 0 \\ \text{length}(\mathit{Head}(T_1)) + & \\ \qquad LCRS(\mathit{Rest}(T_1), \mathit{Rest}(T_2)), & \text{if } \mathit{Head}(T_1) = \mathit{Head}(T_2) \\ \max \begin{cases} LCRS(\mathit{Rest}(T_1), T_2) \\ LCRS(T_1, \mathit{Rest}(T_2)) \end{cases}, & \text{otherwise} \end{cases} \tag{3.24}$$

$$LCRS_{normalized}(T_1, T_2) = \frac{LCRS(T_1, T_2)}{|T_1| + |T_2| - LCRS(T_1, T_2)} \tag{3.25}$$

Road Segments Distance

When comparing road-network constrained trajectories, one can consider the difference between the shapes of each road segment of the trajectories. To this end, we can use the concept of Minimum Bounding Rectangles (MBR) or the Trajectory-Hausdorff Distance.

Figure 3.4a illustrates the distance of two trajectory segments using their MBRs. The MBR of a line segment S_i is described by the coordinates of its lower bound (x_l, y_l) and upper bound (x_u, y_u) points. Equation 3.26 defines the MBR-based distance D_{MBR} between two trajectory segments. In the example, the MBR distance between S_1 and S_2 is 0 and between S_3 and S_4 is $y_l^3 - y_l^4$.

$$D_{MBR}(S_1, S_2) = \sqrt{(\Delta([x_l^1, x_u^1], [x_l^2, x_u^2]))^2 + (\Delta([y_l^1, y_u^1], [y_l^2, y_u^2]))^2} \tag{3.26}$$

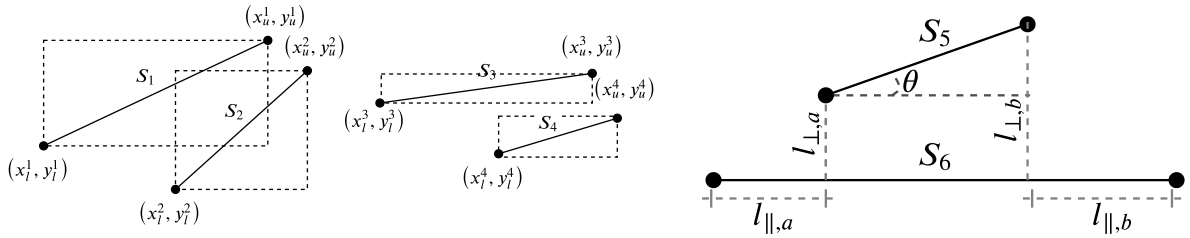
where $\Delta([x_l^1, x_u^1], [x_l^2, x_u^2])$ is the distance between two intervals, defined as:

$$\Delta([x_l^1, x_u^1], [x_l^2, x_u^2]) = \begin{cases} 0, & \text{if } [x_l^1, x_u^1] \cap [x_l^2, x_u^2] \neq \emptyset \\ x_l^1 - x_u^2, & \text{if } x_l^1 > x_u^2 \\ x_l^2 - x_u^1, & \text{if } x_l^2 > x_u^1 \end{cases} \tag{3.27}$$

The Trajectory-Hausdorff distance, as proposed by Lee et al. [113], compares trajectory segments by joining together the parallel ($d_\parallel$), angular (d_θ), and perpendicular ($d_\perp$) distances. Lee et al. [113] created their distance function by adapting techniques used in the pattern recognition research field, which in turn were inspired by the Hausdorff distance. Therefore, Zheng called the method of Lee et al. [113] as Trajectory-Hausdorff Distance (TD_{Haus}) in [215]. Formally,

$$TD_{Haus}(S_i, S_j) = \omega_\perp \cdot d_\perp + \omega_\parallel \cdot d_\parallel + \omega_\theta \cdot d_\theta \tag{3.28}$$

where $d_\perp = \frac{l_{\perp,a}^2 + l_{\perp,b}^2}{l_{\perp,a} + l_{\perp,b}}$, $d_\parallel = min(l_{\parallel,a}, l_{\parallel,b})$, $d_\theta = |S_j| \cdot \sin\theta$, and the weights $\omega_\perp$, $\omega_\parallel$, and ω_θ are defined according to the application requirements. Figure 3.4b illustrates these components.



(a) Minimum Bounding Rectangles distance. (b) Trajectory-Hausdorff distance.

Figure 3.4: The Minimum Bounding Rectangles (MBR) and Trajectory-Hausdorff (TD_{Haus}) distance measures for line segments.

3.3.4 Methods for Other Trajectory Representations

For trajectory representations where the positions of the vehicle are pre-defined, and only the temporal component of the trajectories is of interest, the shrinkage or stretching of trajectories is not allowed, and thus, methods such as DTW are not appropriate. Therefore, Tiesyte and Jensen [181] considered specific distance measures (e.g., modified versions of the LCSS, L_p distance, and weighted L_p distance) that are appropriate for this scenario. For the weighted L_p distance, one can define the weights according to the application. For instance, the weights can be equal, indicating that all parts of the trajectories are evenly important. Another option is to create correlation-based weights when historical data is available. Tiesyte and Jensen [181] used Kendall's tau coefficient τ [97] for determining the weights according to the correlations. Other correlation coefficients, such as Pearson's [163], are also feasible, but the authors opted for Kendall's coefficient based on empirical results.

Deep learning-based trajectory representations, such as the Trembr framework, convert trajectories to low-dimensional feature vectors called trajectory embeddings. We can use these embeddings to calculate the similarity between two trajectories. For instance,

Fu and Lee [65] proposed to employ the Euclidean distance between the learned embeddings to measure the distance between trajectories. Their method outperformed other deep learning-based methods. However, it is unclear how much the results of the Trembr framework depend on the quality of the map-matching performed in the preprocessing step, and consequently, on the sampling rate of the raw trajectory.

3.4 Applications

The task of comparing vehicular trajectories is fundamental for trajectory analysis applications such as predictive queries [51], traffic monitoring [79, 101], and behavior mining [197]. These analysis applications are, in turn, enablers of numerous other applications. For instance, driver assistance systems can detect traffic congestion in real time and calculate optimal routes. The public administration can learn about traffic patterns to guide the policies of urban mobility improvement, or use historical data to forecast the development of traffic jams. Automobile insurers or taxi companies might benefit by applying behavior analysis or recognizing anomalous trajectories. Therefore, choosing a proper similarity measure plays a crucial role in enhancing the efficiency and precision of these applications. For example, DTW-based approaches provide accurate results for similarity-based trajectory retrieval, but they are not suitable for trajectory retrieval in large databases. In the following, we present some analysis applications of vehicular trajectory similarity measures and discuss open research problems related to these applications.

3.4.1 Similarity-based Trajectory Retrieval

Most applications of vehicular trajectory similarity measures derive from the search for related trajectories in large vehicular trajectory databases. For instance, when a vehicle is traveling on a known route, one can identify trajectories similar to the current, partial trajectory, from historical datasets to predict the travel time [181, 119, 95, 118]. As the employed similarity measure directly impacts the effectiveness and efficiency of these kinds of tasks, one needs to choose the most suitable similarity measure carefully. Given a distance function F, we can express the total time T to evaluate a query as:

$$T = \text{I/O time} + \text{number of similarity evaluations} \times \text{complexity of } F \tag{3.29}$$

However, in many applications, the cost to solve the function F is much higher than the cost of the other elements, especially now with the increasing availability of main memory [34].

Sequential Search

The most straightforward idea to search for trajectories similar to a given trajectory T is to make a linear scan over all trajectories and pick those with distance to T less than a given threshold [88]. However, using sequential scanning on today's large-scale databases is becoming impractical. Therefore, there is a demand for the development of new indexing techniques that employ fewer similarity computations [36].

Even though the sequential search is not efficient for querying similar trajectories in large databases, it is still used as a last step verification by many similarity-based retrieval techniques, which in turn employ traditional similarity measures. For instance, DTW is a ubiquitous distance measure widely used in sequential searches. However, as it presents a high computational complexity, some proposals aim to optimize the performance of DTW to allows its usage in the searching of trajectories in massive datasets. A standard technique to accelerate DTW-based queries is to apply a lower bound to prune off unlikely candidate trajectories. Rakthanmanon et al. [158] demonstrated that, with careful implementation, DTW is much more efficient than advanced ED-based techniques.

Trajectory Indexing

When the used distance function is a metric, we can apply the triangle inequality property to index trajectories and thus improve the efficiency of trajectories retrieval. One approach is the *pivot-based metric index* [39], which works as follows. First, in a preprocessing step, select the reference trajectories (pivots) $P_1, ..., P_k$ from the trajectory dataset and compute their distances to all trajectories in the dataset. Then, given a threshold distance d and a search trajectory Q, compute the distance from Q to each pivot P_i. Now we can, employing the triangle inequality, prune off all other trajectories T_i in which $|dist(Q, P_j) - dist(P_j, T_i)| > d$ guaranteeing no false dismissals. Many indexing structures follow this idea, such as AESA [164], LAESA [134], VP-tree [205], FQ-tree [13], MVP-tree [27], PM-tree [173], Omni-family [184], M-index [145], EP [165], and SPB-tree [35]. For instance, some tree-like indexes select the pivots as roots and stores the distances from each root to their children.

Another option to improve the querying of a dense set of trajectories is to store hash-

based trajectories instead of traditional spatial-indexing structures and use indexing techniques based on these hashes. For instance, trajectory fingerprinting with geodabs [33], discussed in Section 2.2.3, enables efficient queries on dense trajectory datasets. Although fingerprinting has a probabilistic nature, which makes the index unable to discriminate between true and false positives, the authors of geodabs demonstrated that it has a minimal influence on the accuracy of the results. Like other proposals, a drawback of the geodabs evaluation is that it relied on synthetic datasets due to the lack of public available large and dense trajectory datasets. Thus, an open research issue is to evaluate trajectory indexing techniques with real large-scale trajectory datasets.

Trajectory Similarity Join Problem

A variation of similarity-based retrieval is the trajectory similarity join problem, which intends at finding every pair of trajectories, from two distinct datasets, that have a similarity measure above a given threshold. Without loss of generality, techniques that address the trajectory similarity join problem can also be used to retrieve similar trajectories. Therefore, some studies employ similarity measures to address the trajectory similarity join problem [159, 168, 178, 204, 171, 211]. Besides that, self join (i.e., finding similar trajectories in only one dataset) is another way to visualize the clustering problem. We discuss trajectory clustering in Section 3.4.2.

Ta et al. [178] introduced the bi-directional similarity metric (BDS, discussed in Section 3.3.2) and signature-based methods to address the similarity join problem. They presented a set of algorithms for sorting, filtering, and indexing trajectories, which reduces the number of trajectory comparisons performed, and thus improves overall processing time. However, experimental results point to scalability issues for processing large databases. Therefore, the employment of more effective trajectory similarity measures that do not decrease the effectiveness of the results is still an open issue.

3.4.2 Clustering

Clustering is a tool that involves many disciplines for finding and defining groups of entities, called clusters, in datasets [137]. For instance, in the data mining research field, clustering is a mechanism for finding patterns in datasets. In knowledge discovery, it is a tool for updating, correcting, and extending the existing knowledge. In machine learning, it is a tool for prediction [137]. Therefore, vehicular trajectories clustering, which aims at grouping similar trajectories according to specific criteria, is a central application of

trajectory similarity measures. Recent surveys [24, 210] presented reviews of trajectory clustering algorithms. Thus, in this section, we briefly discuss similarity-based clustering techniques.

Some naive approaches rely on using trajectory similarity measures and adapting traditional clustering algorithms. For instance, Kim and Mahmassani [101] developed a framework that uses the LCSS distance and the DBSCAN algorithm [61] to cluster GPS-based trajectories. They also proposed the employment of hierarchical agglomerative clustering to identify traffic streams, and then, classify new trajectories.

K-means is a traditional and straightforward clustering algorithm that is still widely used in many areas [92]. The idea of K-means is to find K clusters and associate each data point with the nearest cluster, aiming at minimizing the total variance of all data points in each cluster. However, this task is a known NP-hard problem. Therefore, most proposals employ a standard algorithm to find a local optimum. However, Meilă [132] demonstrated that if the clusters are appropriately defined, the probability that K-means converges to the global optimum is high. K-means performs many comparisons between data points. Thus, it is crucial to use similarity measures that are fast to compute, such as FastDTW, when applied to cluster vehicular trajectories [166].

T-OPTICS [138] is a popular density-based trajectory clustering algorithm that employs the euclidean distance and extends the OPTICS algorithm [9] to work with trajectories. Besides the T-OPTICS, the OPTICS algorithm inspired many other studies. Deng et al. [52] proposed an algorithm called Tra-POPTICS aiming at improving both the scalability and computing performance to cluster big trajectory datasets. Tra-POPTICS is inspired by the POPTICS algorithm [151], which in turn is a scalable version of the OPTICS algorithm. One of its modifications is that it employs the similarity measure introduced by Frentzos et al. [63] rather than using the euclidean distance.

As vehicle trajectories can be quite long, some clustering methods aims at grouping common sub-trajectories instead of whole trajectories, which can be useful for many applications. For instance, one may be interested in recovering all vehicles that traveled close to a set of road segments. TRACLUS [113] is a framework composed of two components. In the first component, an algorithm based on the minimum description length (MDL) principle [72] is responsible for partitioning the trajectories into sub-trajectories (line segments). Then, in the second component, a density-based clustering algorithm groups similar sub-trajectories into clusters.

Hu et al. [86] applied the Dirichlet process mixtures to represent, cluster, and find trajectories. Their technique employs the discrete Fourier transform (DFT) to describe

sub-trajectories, which, in turn, are then used to train the DPMM. Then, they use the parameters of the tDPMMs to index the trajectory patterns. Unlike most other work, their method automatically estimates a suitable amount of clusters and can cluster new trajectories without retraining. However, their discussion about the method focuses on video-based trajectories, and it is unclear what would be the result of applying it to cluster low sampled GPS-based or road-network constrained trajectories.

Fast-clusiVAT [110] is another trajectory clustering algorithm that automatically estimates a suitable amount of clusters. In summary, Fast-clusiVAT is a modified and faster implementation of the clusiVAT algorithm [109] that employs the trajDTW distance measure. Experimental results showed that fast-clusiVAT outperforms some widely used clustering algorithms and provides a significant speedup over its previous version, clusiVAT, without loss of cluster quality.

Most of the clustering techniques discussed above employ a distance function but do not take into account the turn restrictions of the road networks. Han et al. [79] tackled this issue by proposing the NEAT system to cluster road-network constrained trajectories. Besides addressing road-network constrained trajectories, NEAT also considers the road-network-based proximity. NEAT is a three-phase clustering framework based on the following design guidelines. First, the road intersections represent initial partitioning points, and the indivisible sub-trajectories, called fragments, represent the trajectories. Next, the fragments that also represent road segments are clusters of objects associated with the trajectories. Finally, the last phase groups the fragments by taking into account the turn restrictions of consecutive road segments. Although the NEAT framework does not directly employ a similarity measure, it uses a function based on the Hausdorff distance in the last phase. The advantage of the NEAT framework is that as it uses the information about the road network restrictions and does not depend on computing many slow distance measures, it runs very fast. Experiments have shown that NEAT is faster than the TRACLUS framework [113] by more than three orders of magnitude. However, NEAT depends on a preprocessing stage that solves the map-matching problem to divide the trajectories into fragments.

3.4.3 Similarity-based Trajectory Prediction

We can use historical trajectory data and similarity measures to predict short- and long-term vehicle trajectories. For instance, short-term mobility prediction is crucial for advanced driver assistance systems and autonomous driving, and with long-term mobility

prediction, traffic monitoring systems can forecast traffic congestion and guide drivers to faster routes. Recently, Lefèvre et al. [115] reviewed techniques for vehicle motion prediction, most of which rely on the Markov property. Besides that, other studies have proposed the use of deep neural networks, such as Long Short Term Memory (LTSM) [94] and Convolutional Neural Network [126], for vehicle trajectory prediction.

Some proposals employ similarity measures for prediction by matching the current and partial vehicle trajectory with the motion patterns learned from historical data [146, 161, 198]. Okamoto et al. [146] proposed a technique to predict short-term vehicle-motion that, in addition to the Markov property, applies a modified DTW distance and the notion of conservation of similarity (CoS). The basic idea of the CoS is that if two vehicles behaved similarly for the previous several seconds, they would move similarly in the next several seconds, implying that the similarity between the vehicles is conserved. Their method assumes that some vehicles behave similarly due to restrictive constraints such as traffic rules. However, the high computational complexity of the modified DTW may prevent its use in large databases.

Rathore et al. [161] proposed the *Traj-clusiVAT-based TP* clustering framework. The framework is based on the Markov model and can make long-term and short-term predictions. The Traj-clusiVAT-based TP framework employs the Traj-clusiVAT algorithm, which is an extension of the clusiVAT clustering algorithm. One of the improvements of the Traj-clusiVAT algorithm is that it implements a new method to calculate, for each cluster, a *representative trajectory*, which is then used to distribute new trajectories to an existing cluster. Because of that, the Traj-clusiVAT-based TP framework can handle big trajectory datasets in dense road networks efficiently.

3.4.4 Trajectory Outlier Detection

Trajectory outlier detection aims at finding trajectories or sub-trajectories that deviate so much of other trajectories of the dataset [133]. It has a broad application base. For instance, removing outliers is a crucial preprocessing step in support of data mining tasks. Also, the occurrence of similar outliers within a brief time window can indicate abnormal events such as traffic accidents.

Recently, Meng et al. [133] make a review of trajectory outlier detection algorithms from three aspects. First, they classified the algorithms that take into account multi-attributes, such as speed, direction, position, and time. The second viewpoint is if the outlier detection algorithm employs a proper distance function to compute the difference

between trajectories efficiently and accurately. Finally, they reviewed studies that aim at enhancing prior outlier detection algorithms regarding space complexity and processing time. In this section, we are interested in trajectory outlier detection algorithms that employ distance measures.

Some of the earlier trajectory outlier detection algorithms compare whole trajectories and thus are not able to detect outliers in sub-trajectories [103]. To solve this problem, Lee et al. [112] presented a two-stage framework for trajectory outlier detection. The first stage breaks a trajectory into line segments, and the second stage identifies outliers from the set of line segments. Besides the framework, they developed an outlier detection algorithm for sub-trajectories, called TRAOD, that employs a line segment Hausdorff distance. A drawback of the TRAOD algorithm is that it only involves the spatial information of the trajectory and ignores the temporal information.

Mirge et al. [136] proposed a straightforward technique that employed the Hausdorff distance to detect outliers in GPS-based trajectory databases. In summary, they calculate the pairwise Hausdorff distance of all trajectories in the dataset and cluster them according to their distances. Then, the approach identifies all trajectories of a cluster as outliers if the cluster size is below a given outlier threshold. Due to its high computational complexity, one can use this approach only to detect outliers in small databases. Besides that, it is unclear how we can choose the value of the outlier threshold.

Wu et al. [197] presented a technique called DB-TOD for the detection of trajectory outliers that models human behavior by extending the maximum entropy inverse reinforcement learning model. The main advantage of their approach is the ability to detect potential outliers from partial trajectories. The DB-TOD learns from historical datasets the most probable route choices and then infers the likely cost of each road segment. This approach is efficient because the historical dataset is accessed only once for the training.

3.5 Future Research Directions

Even with the tremendous amount of research work in the field of vehicle trajectory data mining, the increasing availability of the so-called "big trajectory data" still presents a large set of challenges to be addressed, even for primary tasks such as trajectory similarity. Throughout this survey, we discussed some open issues that demand additional investigation related to specific methods and applications. In this section, we give an overview of potential research directions, most of which stem from the challenges of the

big data era.

Similarity-based trajectory retrieval, discussed in Section 3.4.1, despite being one of the most basic applications of a similarity measure, surprisingly is still an open issue, at least when dealing with massive datasets. The problem is two-fold. First, accurate similarity measures are time demanding. Second, as stated by Rakthanmanon et al. [158], who developed a known suite of optimizations for the DTW-based time series retrieval, "*there are no known techniques to support similarity search of arbitrary lengths once we have datasets in the billions*". Although the set of optimizations presented by Rakthanmanon et al. [158] shows the best results in the literature, the time required to process massive datasets is still high (several hours). Therefore, we still lack efficient approaches to tackle the similarity-based search of big trajectory data.

Another big challenge regards the trajectory representation. As discussed in Section 2.2, the most widely-used vehicle trajectory representations are the *GPS-based trajectories* and *road-network constrained* trajectories. The advantage of these representations is their simplicity, which facilitates the development of efficient similarity measures and applications. However, these representations lack the level of detail required by many applications. For instance, none of these representations can provide the vehicle's location/speed at any given timestamp. For a GPS-based trajectory, it is possible to apply preprocessing techniques to increase its granularity (sampling rate), but for some situations, this is not enough yet. Moreover, increasing the trajectory sizes also increases the processing time of data analysis tasks. A promising direction is to develop machine learning-based techniques to represent the trajectories better, such as the novel deep learning trajectory representation proposed by Li et al. [117].

Although this Chapter focuses on vehicle trajectory data, emerging smart mobility solutions will need to tackle trajectories from different data sources and modes of transport. Therefore, besides the development of advanced trajectory representations, there is a need for novel data fusion and visualization techniques to address multi-modal and multi-source trajectories. Furthermore, a relevant open research problem in this subject is the impact of these heterogeneous trajectories on the effectiveness of existing similarity measures.

3.6 Chapter Remarks

The increasing availability of vehicular mobility data attracted many researchers in the last couple of years due to its extensive amount of applications and the demand for

algorithms to mine big trajectory data. For some data mining tasks (e.g., clustering and retrieval), the development of efficient and accurate similarity measures plays an important role.

This Chapter surveyed the research efforts related to vehicular trajectory similarity measures. We presented a comprehensive review of methods to compare trajectories. In doing so, we classified each method according to the trajectory representation and features such as metricity, computational complexity, and robustness to noise and local time shift. Finally, we presented an overview of applications of trajectory similarity measures and briefly discussed some open research problems.

Table 3.1: Surveys on trajectory similarity measures

Reference	Methods covered	Goals
Wang et al. [191]	Methods for GPS-based trajectories: ED, DTW, PDTW, LCSS, ERP, and EDR	Evaluate the effectiveness of the measures and demonstrate their advantages and drawbacks through an experimental study
Magdy et al. [128]	Methods for GPS-based trajectories: ED, DTW, Time Warp edit distance, ERP, EDR, LCSS, Spatial Assembling Distance, Hausdorff distance, and Fréchet distance	Classifies methods concerning their computational cost, whether the measure is metric or not, the capacity to handle trajectories of different lengths, and robustness to deal with noise and local time shifting
Yu Zheng [215]	Methods for GPS-based trajectories: Closest-pair, DTW, LCSS, EDR, and ERP. Methods for road-network constrained trajectories: Minimum Bounding Rectangles distance and Trajectory-Hausdorff distance	Presents a brief discussion on the main methods to calculate the distance between trajectories, or between trajectories and GPS points
Toohey and Duckham [183]	Methods for GPS-based trajectories: LCSS, Fréchet distance, EDR, and ERP	Compares four methods through an experimental study and briefly discuss some applications
Besse et al. [23]	Methods for GPS-based trajectories: DTW, LCSS, EDR, and ERP	Reviews methods focusing on one application (trajectory clustering), and propose a new method called SSPD

Table 3.2: Main Notations

Symbol	Description
T_i	GPS Trajectory i
$\|T_i\|$	Number of sampling points of trajectory T_i
ρ_i^j	ith point of Trajectory j
$dist(\rho_i, \rho_j)$	Straight-line distance between the given points if they are in Euclidean space, or the greatest circle distance if the points are geographical
$Head(T_i)$	First point of T_i, that is, ρ_1^i
$Rest(T_i)$	Sub-trajectory of T_i without the first point ρ_1^i

Table 3.3: Trajectory Similarity Methods for GPS-based Trajectories.

Method	Complex.	Metric	$\neq$ len.	Param.-free	Loc. Ti. Shift	Noise
ED	$O(n)$	✓		✓		
DTW	$O(n^2)$		✓	✓	✓	
PDTW [99]	$O(n^2)$		✓		✓	
FastDTW [166]	$O(n)$		✓		✓	
SDTW [129]	$O(n^2 \log n)$		✓		✓	✓
ERP [37]	$O(n^2)$	✓	✓	✓	✓	
EDR [38]	$O(n^2)$		✓		✓	✓
TWED [130]	$O(n^2)$	✓	✓		✓	
LCSS [190]	$O(n^2)$		✓		✓	✓
Fréchet Dis.	$O(n^2 \log n^2)$		✓	✓		
Hausdorff Dis.	$O(n^2)$	✓	✓	✓		
EDwP [159]	$O(n^2)$		✓	✓	✓	✓

Chapter 4

The Rio Center Dataset

This Chapter presents a novel real-world and large-scale vehicle trajectory dataset, called Rio Center Dataset. The dataset is based on the central zone of Rio de Janeiro and comprises trajectories acquired from a private company located in Brazil that tracks GPS-equipped vehicles from different users (e.g., regular users, taxi/app drivers). We prepare different versions of the dataset, each version with a different set of trajectories for different purposes. The first version is a novel benchmark dataset designed to support the evaluation of trajectory reconstruction and map-matching solutions, and the second version is a large-scale dataset of reconstructed road-network constrained trajectories. This second version supports the evaluation of data mining techniques that relies on big data. Finally, we present a version containing road-network constrained trajectories for evaluating vehicular networks scenarios that relies on trajectory prediction. We use these new datasets to perform extensive experiments on the methods, algorithms, and protocols proposed in this thesis.

4.1 Introduction

Vehicular trajectory data is increasingly available due to the population growth in urban centers and the advances in location-acquisition technologies, which results in many GPS-equipped vehicles generating massive trajectory traces. This data is crucial to understanding the rapidly-changing mobility aspects of society and developing novel transportation modes. Furthermore, vehicular trajectory datasets are useful for applications in many research fields, such as location-based social networks, data-mining-based applications, traffic analysis [44, 46], urban planning, and vehicular networks.

Mobility datasets are essential for the development of applications and protocols for vehicular networks. First, we can design more efficient and effective routing protocols by analyzing vehicle traces. For instance, we can develop better delay-tolerant routing protocols by employing trajectory prediction models based on historical mobility datasets. Second, an effective and widely used way to evaluate vehicular networks is to perform simulations based on real mobility traces. However, there are few vehicle trajectory datasets available today [32]. Furthermore, most of these datasets have synthetically created traffic demand or represent only one class of vehicles (e.g., taxis), which is not enough to represent a region's traffic diversity in most cases [32].

In this Chapter, we present a new vehicular trajectory dataset based on the city of Rio de Janeiro - Brazil. The dataset contains data from more than ten thousand vehicles of different categories, tracked over one year. We generate dataset versions from different subsets of trajectories, each version with a specific purpose. To the best of our knowledge, this is the first large-scale and realistic dataset that includes various vehicle types (e.g., personal, cargo, and taxis).

We organize the remainder of this paper as follows. Section 4.2 gives an overview of the dataset, presenting the process of data acquisition and some statistics. Sections 4.3, 4.4, and 4.5 presents the three versions of the dataset. Finally, Section 4.6 concludes this Chapter.

4.2 Dataset Overview

The trajectories that comprise the Rio Center Dataset come from a private company that tracks hundreds of thousands of vehicles in Latin America. We decided to prepare a dataset based on the central zone of Rio de Janeiro city for three reasons. First, we need to constrain the road network size to allow the execution of multiple experiments that the dataset intends to (e.g., map-matching and vehicular networks simulations) in a feasible time. Second, the central zone of the city of Rio de Janeiro contains a wide variety of road network configurations, such as roundabouts, viaducts, and complex intersections, which allows the evaluation of different situations with a single scenario. Lastly, we can create scenarios with a high density of vehicles in this region, as more than ten thousand tracked vehicles traveled across it.

We retrieved the road network of the central zone of Rio de Janeiro using OpenStreetMap [148] and generated its graph representation $\mathcal{N} = (\mathcal{V}, \mathcal{E})$, as defined in Chapter 2, using the OSMnx [25]. We used the *simplify* option of OSMx, so that nodes

represent the intersection of road segments, instead of including all the interstitial OSM nodes. Each arc contains the linestring to represent the shape of the segment. Figure 4.1 shows the map of the final version of the road network $\mathcal{N}$, and Table 4.1 summarizes some statistics of the dataset.

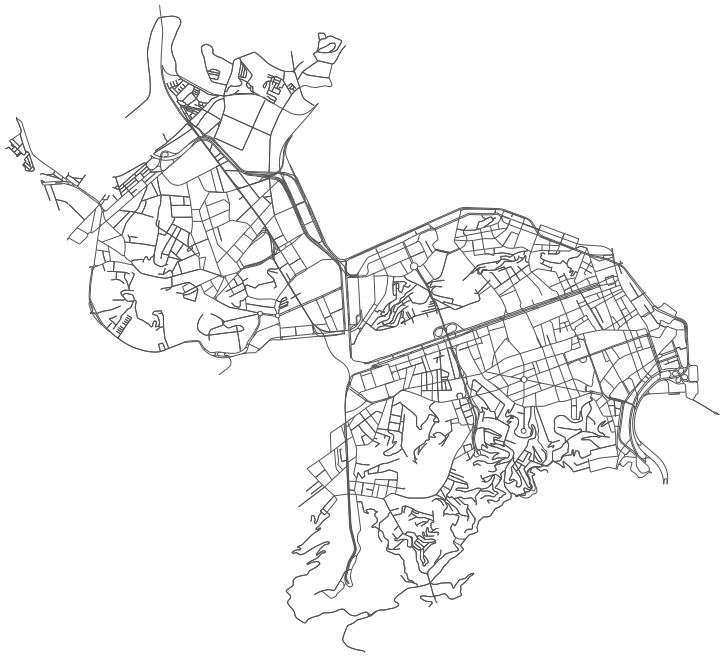


Figure 4.1: The road network of the Rio Center Dataset.

During the monitoring time, more than ten thousand vehicles traveled through the region's roads, generating millions of data points. However, we did not use all recorded data to generate the trajectory datasets. Instead, we select trajectories from different timeframes (and durations) to generate the dataset versions according to their purpose. Disregarding the timeframe, we extract the GPS-based trajectories (as defined in Chapter 2) using a basic set of information (i.e., longitude, latitude, acquisition timestamp, and vehicle ignition) from the logs as follows. First, we segment the logs from each vehicle

Table 4.1: Rio Center dataset statistics.

	Attribute	Value
Road Network	Number of Intersections	2192
	Number of Arcs	4439
	Total Arcs Size (km)	575,855.8
	Average Arcs Size (m)	129.73
Trajectories	Number of Vehicles	$\approx 10,000$
	Monitoring Time	one year

using the vehicle's ignition information. A valid GPS-based trajectory is a consecutive sequence of GPS points where ignition is on. In some situations, a GPS device can present malfunctioning and generate large consecutive sequences of online GPS points comprising more than one trajectory. We observe some of these problems in our dataset. Therefore, we preprocess the dataset to remove outliers (in terms of trajectory length) using the interquartile range (IQR) method. That is, we remove trajectories of length above the 75th or below the 25th percentile by a factor of 1.5 times the IQR. Besides, we discard sequences of size one. This process is repeated for each dataset version presented in Sections 4.3, 4.4, and 4.5.

4.3 The Rio Center Dataset for Trajectory Reconstruction

In Chapter 6, we propose a solution for the trajectory reconstruction problem. In summary, trajectory reconstruction is the process of converting GPS-based trajectories into road-network constrained trajectories. One of the significant challenges for this research area is the lack of reliable datasets with ground-truth data, which hinders the evaluation and comparison between methods to solve the trajectory reconstruction problem. Therefore, we prepared a version of the Rio Center dataset for experiments concerning this problem. The *Rio Center Dataset for Trajectory Reconstruction* is a novel benchmark dataset designed to support the evaluation of trajectory reconstruction and map-matching solutions.

We generated eighteen road-network constrained trajectories by manually analyzing

each valid raw trajectory and the underlying road network individually. A valid raw trajectory is a consecutive sequence of more than one GPS point where ignition is on. Figure 4.2 presents the trajectories of the dataset, which are from different areas in the central zone of Rio de Janeiro.

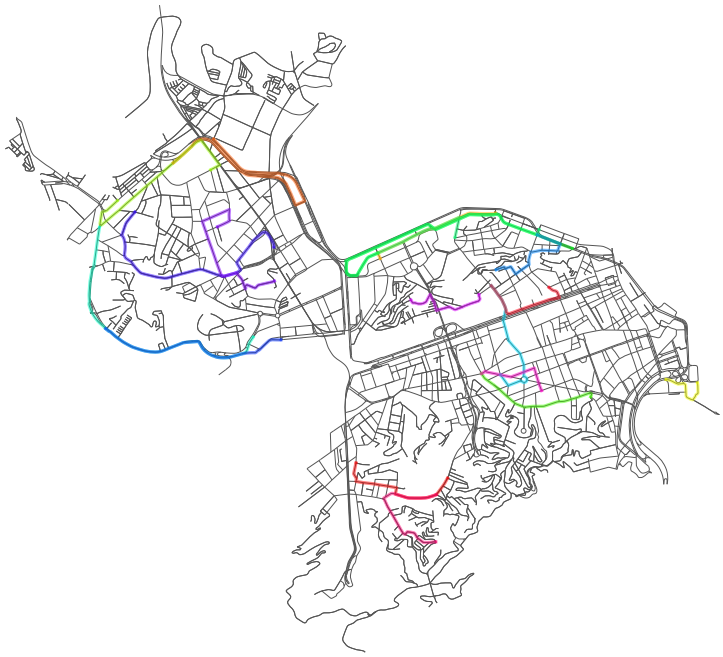


Figure 4.2: The 18 reconstructed trajectories of the Rio Center Dataset for trajectory reconstruction.

This dataset version contains a small number of trajectories compared to the other versions. This difference happens because the manual reconstruction process is extremely slow and must be done carefully. Each point is analyzed individually, and the sections of the trajectories are also analyzed together to prevent errors. Despite the small number of trajectories, they present different road-network scenarios with characteristics that hinder the trajectory reconstruction process. For instance, some trajectories cover areas

with roundabouts, complex intersections, viaducts, and tunnels. Figure 4.3 shows one of the trajectories, which crosses a region with several intersecting viaducts. Most of the map-matching algorithms that do not consider the trajectory as a whole will wrongly match the sampled points in the viaduct area.

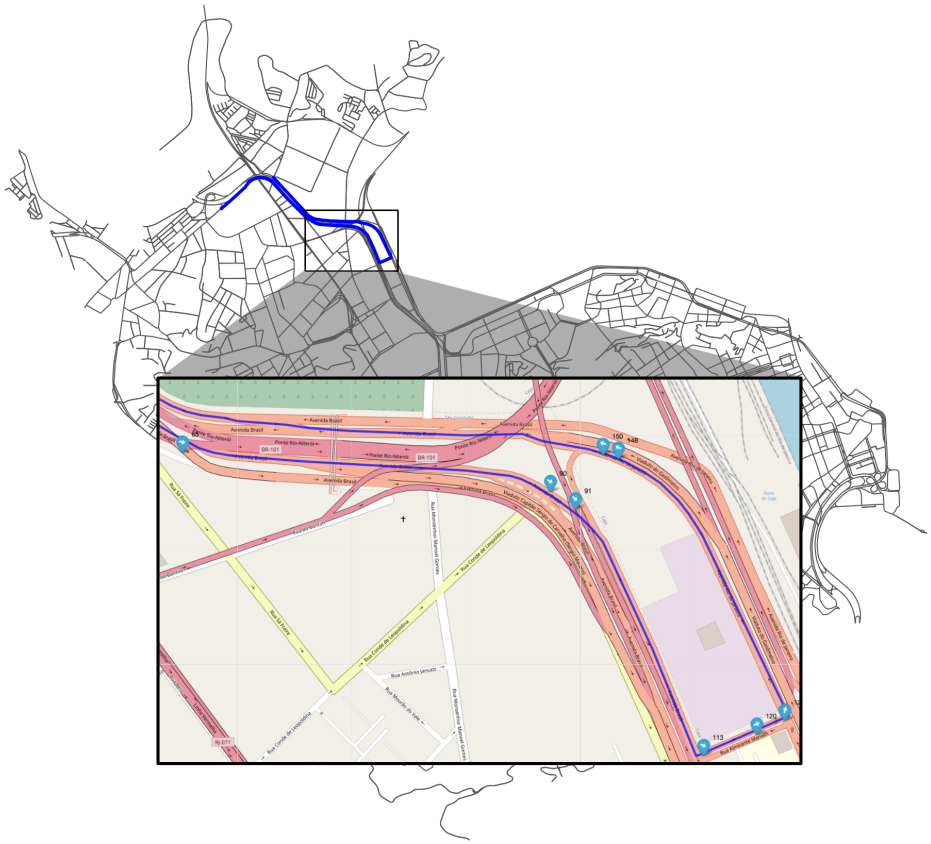


Figure 4.3: One of the trajectories from the Rio Center Dataset for Trajectory Reconstruction. The trajectory crosses a region with multiple viaducts and tunnels, which hinders the reconstruction of the trajectory.

4.4 The Rio Center Dataset for Data Mining

Data mining techniques, such as clustering and prediction, require a large volume of data. Therefore, we create a dataset version containing trajectories from a more extensive timeframe. In this thesis, we use this dataset in Chapter 6 to evaluate the proposed

trajectory prediction framework.

The Rio Center Dataset for Data Mining contains four months of recorded data, resulting in 526,467 trajectories. We prepare a set of GPS-based trajectories and another set of road-network constrained trajectories, making it helpful for methods that use these trajectory representations. We generate the set of road-network constrained trajectories using the map-matching-based framework for trajectory reconstruction proposed in Chapter 5. Figure 4.4 shows a sample of the trajectories on the road network map $\mathcal{N}$. Table 4.2 summarizes some statistics of the dataset.

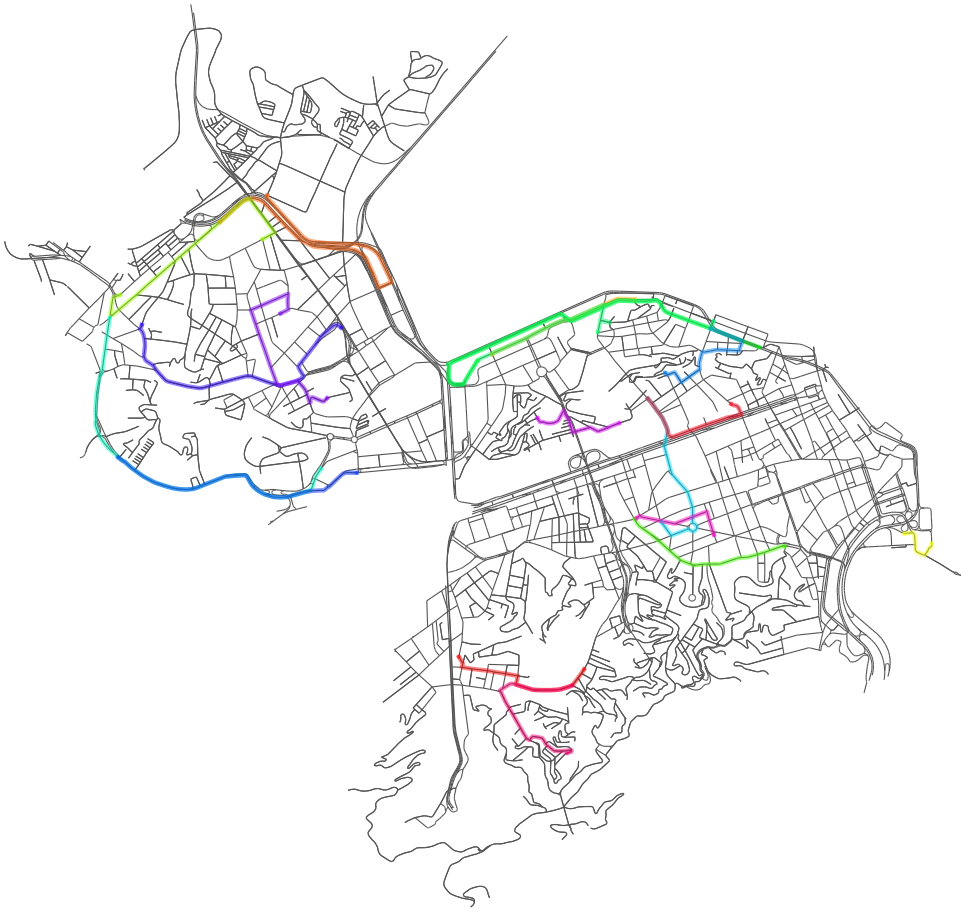


Figure 4.4: Sample of the trajectories of the Rio Center Dataset for Data Mining.

Table 4.2: Rio Center Dataset for Data Mining statistics.

Attribute	Value
Number of Trajectories	526,467
Monitoring Time	4 months
Average Size (Number of Roads)	18.36
Average Size (km)	3.34

4.5 The Rio Center Dataset for Evaluating Prediction-based Methods for Vehicular Networks

The last dataset version aims to assist in experiments of algorithms and protocols of vehicular networks based on trajectory prediction. In this thesis, we use this dataset version to evaluate the VDDTP algorithm, proposed in Chapter 8.

The dataset comprises one month of recorded trajectories, from October 17, 2021, to November 15, 2021, and has two sets of road-network constrained trajectories. We use the first set to perform vehicular network experiments and the second set to create trajectory prediction models. The first set contains all trajectories that start and end within a time window of two hours (from 6:00 a.m. to 8:00 a.m. of November 15, 2021), and the remaining ones compose the second set. As within the dataset for data mining, we generate the set of road-network constrained trajectories using the map-matching-based framework for trajectory reconstruction proposed in Chapter 5. Table 4.3 summarizes some statistics of the dataset.

Table 4.3: Rio Center Dataset for Evaluating Prediction-based Methods for Vehicular Networks.

Attribute	Value
Number of Trajectories	456,835
Number of Trajectories (training)	456,116
Number of Trajectories (evaluation)	719
Duration	30 days
Duration (evaluation)	2 hours

4.6 Chapter Remarks

This Chapter presented the Rio Center Dataset, a novel real-world and large-scale vehicle trajectory dataset based on the city of Rio de Janeiro, Brazil. Unlike most related work, the Rio Center Dataset contains data from different users (e.g., regular users, taxi/app drivers), making it more suitable for realistic experiments.

We designed various dataset versions containing different subsets of trajectories and data formats, each for specific purposes. These datasets aim to assist in carrying out experiments and simulations to evaluate the algorithms, methods, and protocols proposed throughout this thesis.

Chapter 5

A Novel Scalable Framework to Reconstruct Vehicular Trajectories from Unreliable GPS Datasets

Vehicle trajectory data is of paramount importance in many applications and research areas, such as vehicular networks and Intelligent Transportation Systems (ITS). However, data gathered from location acquisition devices generally contain positional errors that hinder its applicability, and therefore processing techniques are necessary to improve the quality of trajectory data. For instance, physical constraints of the road network that bounds the vehicles' movement can be used to represent a trajectory better. Therefore, this Chapter proposes an efficient framework to reconstruct road-network constrained trajectories from GPS-based datasets. The framework employs novel processing algorithms and models to prepare even low sampled trajectories, which naturally present gaps, for real applications. The experimental results show that the proposed framework has a better time complexity and accuracy than the other methods in all evaluated scenarios.

5.1 Introduction

The growing availability of trajectory data generated by GPS-equipped vehicles offers us crucial information for applications in different research areas (e.g., location-based social networks, vehicular networks, Intelligent Transportation Systems, route recommendation, traffic analysis, and urban planning [44, 49]). However, trajectories acquired from

satellite-based positioning systems generally contain precision errors that we need to fix to improve the applicability of these data [106]. For instance, most datasets have low sampling rate data, which results in gaps between recorded points. On the other side, high sampling rates can result in scenarios in which a vehicle records two subsequent points, and the first one appears further distant on the road segment than the second point. Some of the techniques to enhance trajectory data are noise filtering, segmentation, and map matching [215].

Map-matching is the process of mapping each trajectory point to a road of the transportation network. However, the definition of this process varies in the literature depending on the application. For instance, tracking applications use map-matching to recover the vehicle's route from a sequence of GPS points if the sampling frequency is high enough, and navigational applications only "snap" individual points to the road network [106]. These usual definitions of map-matching do not require recovering the entire path traveled by the vehicle.

In this Chapter, we investigate the trajectory reconstruction problem. Trajectory reconstruction is the problem of recovering the whole path traveled by the vehicle from a GPS-based trajectory. We call the reconstructed trajectory a *road-network constrained trajectory*. The trajectory reconstruction problem differs from map-matching as the first generates a sequence of connected road segments representing a valid trajectory, and the latter's output can be an invalid trajectory. For instance, map-matching for tracking applications can be seen as a sub-task of the trajectory reconstruction problem as it produces a sequence of unconnected road segments.

The performance evaluation of techniques for the trajectory reconstruction problem and map-matching requires reliable datasets with ground-truth data. However, this kind of dataset with ground-truth data can only be acquired through a manual process due to absence of an automated method for trajectory reconstruction that guarantees results with 100% accuracy. Because of that, there is a lack of reliable datasets with ground-truth data, representing one of the significant challenges for this research area. This lack of manually reconstructed datasets hinders the evaluation and comparison between methods to solve the trajectory reconstruction problem.

In this context, we propose a novel framework to reconstruct road-network constrained trajectories from GPS-based trajectories. The framework consists of four components to rebuild the road-network constrained trajectory from a given road network graph and GPS-based trajectory. In each component, we propose novel algorithms and models that reduce the overall time complexity of the framework and improve the ac-

curacy of the reconstructed trajectory. Compared with state of the art, we highlight the following improvements. First, we propose a preprocessing technique to reduce the overall computational overhead of map-matching based solutions for the trajectory reconstruction problem. Second, the majority of the map-matching based solutions rely on a step called *Candidate Set Selection*, which has a significant impact on the computational complexity and accuracy of the solutions. In this aspect, we introduce and evaluate a novel technique to compute the candidate set. This technique improves both the accuracy and complexity, not only of the proposed framework but also of the related proposals. Finally, different from most related work, our solution recovers the paths traveled between consecutive matched samples, and, thus, reconstructs the trajectories completely. Experiments reveal that our framework efficiently provides accurate results to reconstruct trajectories with different sampling rates and outperforms some related work regarding the accuracy and computational complexity.

It is important to note that this Chapter focuses on problems regarding data preprocessing instead of its applications. Therefore, the relation between the proposed framework and Vehicular Ad-hoc Networks (VANETs) is that VANETs can benefit from trajectories reconstructed by the framework. For instance, we can perform more accurate evaluations of vehicular networks by using reliably reconstructed trajectories instead of GPS-based data that contain errors. However, VANET (and its associated applications) is just one of the possible scenarios for our trajectory reconstruction framework.

5.2 Related Work

Most of the literature regarding the trajectory reconstruction problem proposes map-matching techniques for tracking applications [105, 140, 69, 8, 54, 66], which is a sub-task of the trajectory reconstruction problem. Therefore, this section presents a literature review of state-of-the-art solutions for the map-matching problem related to tracking applications.

Many algorithms to solve the map-matching problem are based on the Hidden Markov Model or closely related [105, 140, 69, 8, 54]. One of the best-known methods, and probably the most used in the industry to date, is the algorithm proposed by Newson and Krumm [140], which extends another HMM-based map-matching algorithm [105]. The original version of the algorithm restricts the map-matched trajectory to those routes whose expected travel time is congruent with the observed travel time. On the other hand, the extended version favors transitions where the distance between two consecutive

trajectory points is similar to the shortest driving distance between the matched samples.

The approach introduced by Krumm et al. also served as inspiration for many other algorithms, such as the ST-Matching [124]. ST-Matching is an algorithm designed to match points of low-sampled trajectories. It has a spatial and temporal component. The temporal component favors paths in which the average speed is similar to the speed limit of the roads. The spatial component contains an "observation probability," which is equal to the "measurement probability" proposed by Newson and Krumm [140], and a "transition probability" based on the same idea of the transition probability of Newson and Krumm. IVMM [212] is an extension of the ST-Matching that, besides the spatial and temporal information, proposes a voting-based strategy to model the weighted mutual influences between trajectory points. A shortcoming of the ST-Matching and IVMM is that they rely on road speed limits information, which is not always available, especially in publicly available datasets.

The "Path Inference Filter" (PIF) [87] is a trajectory reconstruction solution based on Conditional Random Fields (CRF). PIF employs a set of algorithms aiming at recovering trajectories and road positions from low-frequency probe data. The main reason to use CRF over HMM is to solve the "selection bias problem," which is a problem commonly found in HMM-based models. The experimental results showed in [87] are promising in terms of matching accuracy. However, the Path Inference Filter considers ten influencing factors that are difficult to obtain, and its computational complexity might be too prohibitive. Liu et al. [122] proposed the ST-CRF to improve the time complexity of the CRF-based method. To do so, the ST-CRF considers only the spatial-temporal influencing factors, and, thus, it is more suitable for most scenarios.

Yin et al. [206] proposed a feature-based algorithm that uses GPS observations and human factors to estimate the cost of candidate routes and reconstruct a vehicle trajectory. The experimental results showed that their method improves the accuracy of the reconstructed trajectory, but the processing time is higher than the two related proposals evaluated.

More recent studies propose heuristics to solve the map-matching problem for tracking applications. For instance, AntMapper [70] is an ant colony-based algorithm that uses trajectory information such as speed, bearing, and location to compute both a local heuristic and a global fitness. The search for the best path employs local and global information. The AntMapper has the shortcoming of only working if all required data (e.g., speed and bearing) is available.

The solutions reviewed in this section have one or more of the following drawbacks:

- They depend on data that may be unavailable, such as road speed limits and vehicle's speed and heading. Processing time (e.g., several minutes per trajectory point), since those solutions are inefficient in reducing the number of roads processed, making them unsuitable for large scenarios or trajectories.
- They map points to the road network individually instead of recovering the whole trajectory of the vehicle. Because of that, the recovered trajectory can be invalid considering gaps (unconnected consecutive roads, which are common in low-sampled trajectories.
- Their experiments are based on relatively small datasets. These datasets fail to include various characteristics that hamper the trajectory reconstruction problem, such as complex intersections, roundabouts, and viaducts.

5.3 Problem Formulation

We formally define the Trajectory Reconstruction problem using the definitions of the road network, GPS-based trajectory, and road-network constrained trajectory presented in Chapter 2. Given a graph $\mathcal{N}$ that represents the road network and the GPS-based trajectory $\mathrm{T}_{\mathrm{gps}}$, trajectory reconstruction is the problem of finding a path in graph $\mathcal{N}$ that corresponds to the correct route traveled by the vehicle during the generation of the $\mathrm{T}_{\mathrm{gps}}$'s points. The output of this process is represented by a road-network constrained trajectory $\mathrm{T} = (t_1, t_2, \ldots, t_n)$.

5.4 Proposed Framework

5.4.1 Preliminaries

To reconstruct a vehicle trajectory, we first need to understand how the vehicles generate the recorded points. Given some raw trajectory $\mathrm{T} = (\rho_1, \rho_2, \ldots, \rho_n)$, it is reasonable to assume that, for every point ρ_i, the vehicle must have been in one of the arcs geographically close to it. Therefore, we can represent the vehicle's exact location when it recorded the point ρ_i as:

$$l_i = (e_i, \mathit{offset}_i) \tag{5.1}$$

where $e_i \in \mathcal{E}$ is its current arc and $offset_i$ a non-negative number corresponding to its position at arc e_i (i.e., the distance, in meters, between the starting point of the arc e_i and the vehicle's location at e_i).

For each pair of consecutive locations l_i and l_{i+1}, the vehicle followed one path $p_i = (e_1, e_2, \ldots, e_{|p_i|})$ consisting of an ordered and non-empty set of connected arcs. The correct path p_i belongs to a finite set of valid paths between l_i and l_{i+1}. The first and last arcs of the correct path p_i are the arcs e_i and e_{i+1}, which correspond to the arcs traversed by the vehicle when it recorded the points ρ_i and ρ_{i+1}, respectively. Finally, the course the vehicle took is a set of locations and paths $l_1 p_1 l_2 p_2 l_3, \ldots, p_{n-1} l_n$, and the road-network constrained trajectory we are trying to reconstruct is the union of the paths p_i for all i from 1 to n. Figure 5.1 illustrates the process of trajectory reconstruction with the components discussed above.

As previously discussed, the first step to reconstruct a trajectory is to select some candidate locations close to each point of the raw trajectory. We call this step *candidate set selection*. Figure 5.1 illustrates a trajectory and some candidate locations. For instance, point ρ_1 has four candidates locations represented by the arcs e_{22}, e_{10}, e_7, and e_{21}, and the points $x_{1,1}$, $x_{1,2}$, $x_{1,3}$, and $x_{1,4}$, where point $x_{i,j}$ is the point-to-segment projection of the trajectory point ρ_i onto the candidate arc e_i. It is essential to keep the candidate set size (i.e., number of candidate locations) small because the time complexity of the entire process is directly proportional to it. Therefore, as long as we do not accidentally remove the correct arcs from the list of candidates, the most efficient solution is the one with the smaller candidate set. To the best of our knowledge, all related studies employ one of the following straightforward rules to choose the candidate arcs of each point: 1. all arcs within a given radius; 2. K-nearest neighbors; and 3. arcs within a given radius, limited to a maximum of K. We propose a custom technique for the candidate set selection in this work, which we describe in Section 5.4.4.

The next step is to find the probable paths between all pairs of consecutive candidate locations. A straightforward solution is to take the k-shortest routing paths between the candidate locations, with $k > 1$. In general, the lower the GPS sampling rate, the greater the distance between recorded locations, and the greater the number of likely paths between these locations. However, most location-acquisition devices record their positions with a sampling interval of at least 30 seconds. Accordingly, most publicly available vehicle trajectory datasets also have a high sampling rate [47]. Moreover, it is unlikely that a vehicle has traveled a path other than the shortest route between recorded locations with these sampling rates (we verified this by analyzing some publicly available

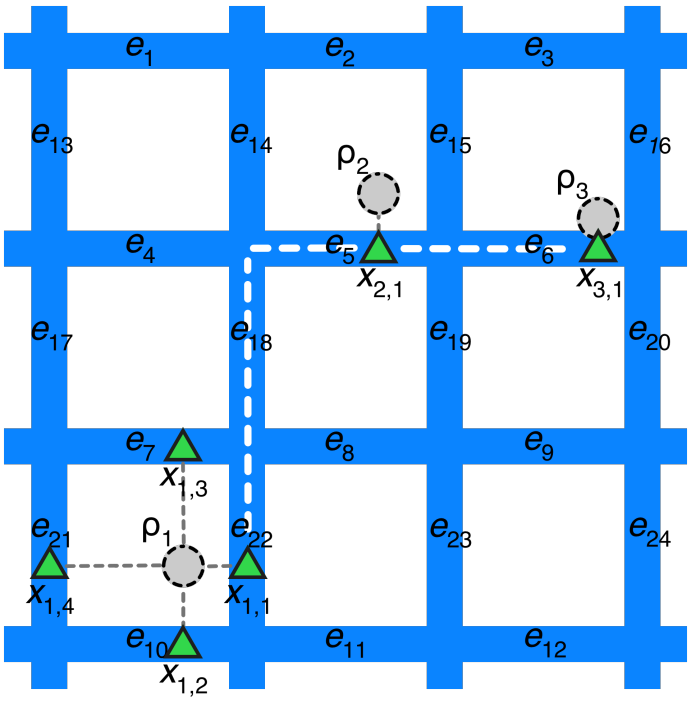


Figure 5.1: Example of a trajectory reconstruction and its components. The gray circles represent the raw GPS trajectory with three points $\mathrm{T}_{\mathrm{gps}} = (\rho_1, \rho_2, \rho_3)$. The yellow triangles $\mathrm{x}_{i,j}$ represents some of the vehicle's likely locations when it recorded the points. The white dashed line corresponds to the correct trajectory traversed by the vehicle. The correct arcs of the points ρ_1, ρ_2, and ρ_3 are e_{22}, e_5, and e_6, respectively, and the path the vehicle followed from ρ_1 to ρ_2 and from ρ_2 to ρ_3 are $p_1 = (e_{22}, e_{18}, e_5)$ and $p_2 = (e_5, e_6)$, respectively. Therefore, the reconstructed road-network constrained trajectory is $\mathrm{T} = (e_{22}, e_{18}, e_5, e_6)$.

datasets). Therefore, we ignore the alternative paths in this work. We consider only the shortest routing path between two consecutive candidate locations. Consequently, the maximum number of trajectories that we will consider in the last step of the trajectory reconstruction process is a function of the number of candidate roads only.

The last step, called *solution construction*, selects the most likely sequence of arcs and in-between paths from the candidate set and then builds the final trajectory as the concatenation of the selected paths. Given a raw trajectory T of length n, the upper bound number of candidate trajectories is k^n, where k represents the maximum number of candidate locations per point. As the number of candidate trajectories grows exponentially with the trajectory length, processing all candidate trajectories is impractical.

Therefore, we propose a solution construction algorithm based on Hidden Markov models in this work. Section 5.4.5 presents the proposed models used to extract a set of information from the candidate locations. We use this information set in the solution construction step described in Section 5.4.6.

5.4.2 Overview

Figure 5.2 presents an overview of the proposed framework. The framework's input is the road network definition and a raw trajectory, and its output is the reconstructed trajectory, as defined in Section 5.3. Besides the steps discussed in the preliminaries (i.e., candidate set selection, information modeling, and solution construction), the framework contains an initial processing step, which we present in Section 5.4.3.

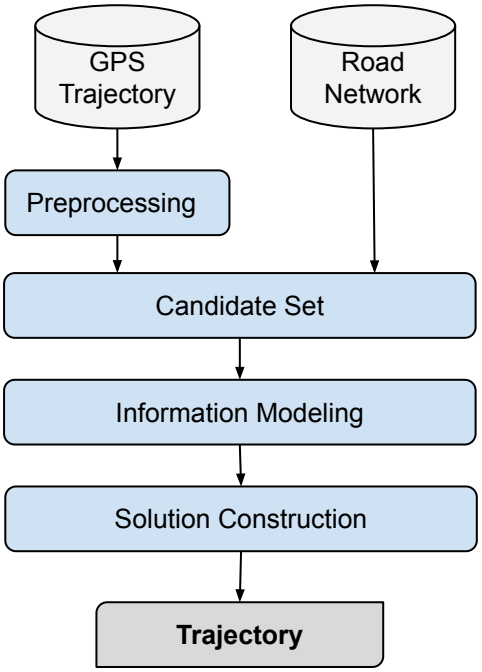


Figure 5.2: The components of the proposed framework to reconstruct vehicle trajectories.

5.4.3 Preprocessing

A preprocessing step applies techniques such as noise filtering to improve the trajectory data for the algorithms of a map-matching or trajectory reconstruction solution. For

instance, Pink and Hummel [155] used the extended Kalman filter to smooth the trajectory by combining the measured trajectory with the model's predictions. However, this approach presents a poor performance on sparse trajectories. Another preprocessing technique is the straightforward algorithm proposed by Newson and Krumm [140]. Their method ignores some trajectory points by moving sequentially through the trajectory points and removing those within a radius of two standard deviations from the last accepted one (the standard deviation is an attribute given as input to the algorithm). The reasoning behind this method is that when consecutive samples are close to each other, the difference in position is more likely to be due to noise than to actual vehicle movement.

We propose a preprocessing algorithm based on the solution presented by Newson and Krumm [140]. The difference in our technique is that, instead of ignoring points close to each other, it uses them to smooth the trajectory. Algorithm 1 shows the proposed technique. We iterate through the trajectory points sequentially, looking for clusters of points. For each cluster found, we replace all its points with a new one with coordinates (i.e., latitude and longitude), speed, and direction equal to the average of the coordinates, speeds, and directions of the points in the group (Lines 7–10). The timestamp of the new point is equal to the timestamp of the first point of the group (Line 11). The function $\|.\|_{\text{geo}}$ (Line 4) computes the great-circle distance between the given points, and the function $\text{atan2}\,(y, x)$ (Line 10) is the 2-argument arctangent function, which computes the angle between the x-axis and the vector (x, y). The algorithm assumes that the points' directions ρ.dir are in radians.

Figure 5.3 and Table 5.1 show the results of applying the proposed preprocessing algorithm to a trajectory taken from a real-world dataset compared to the technique presented by Newson and Krumm [140]. The original trajectory has seven points, and both algorithms find three groups of points. Therefore, the result of applying the preprocessing algorithms is a trajectory containing three points. However, the algorithm of Newson and Krumm [140] takes the first point of each group (Figure 5.3c), and our solution (Figure 5.3b) computes the group's average point. The original trajectory's points 4, 5, 6, and 7 indicate that the vehicle has turned left at the fork shown in the upper left corner of the map. Point 3 of the trajectory resulting from the proposed algorithm also suggests that the vehicle has turned left, but this is not evident in the trajectory resulting from the algorithm of the related work.

Algorithm 1 Procedure to reduce the trajectory length

Input: $T_{\text{gps}} = (\rho_1, , ..., \rho_n)$: trajectory, σ: standard deviation

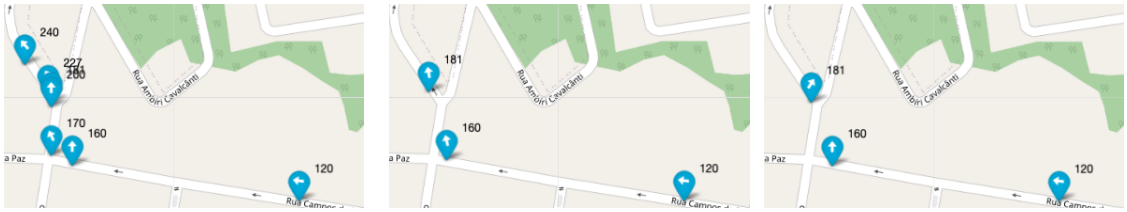
Output: The processed GPS trajectory $T_{\text{processed}}$

$T_{\text{processed}} \leftarrow \emptyset$, $i \leftarrow 1$
while $i \leq |\, T_{\text{gps}}|$ **do**
  $j \leftarrow i + 1$
  while $j \leq |\, T_{\text{gps}}| \wedge \|\rho_i, \rho_j\|_{\text{geo}} \leq 2\sigma$ **do**
    $j \leftarrow j + 1$
  end while
  $\rho_{\text{new}}.\text{lng} \leftarrow \frac{\sum_{k=i}^{j-1} \rho_k.\text{lng}}{j-i}$
  $\rho_{\text{new}}.\text{lat} \leftarrow \frac{\sum_{k=i}^{j-1} \rho_k.\text{lat}}{j-i}$
  $\rho_{\text{new}}.\text{spd} \leftarrow \frac{\sum_{k=i}^{j-1} \rho_k.\text{spd}}{j-i}$
  $\rho_{\text{new}}.\text{dir} \leftarrow \text{atan2}\left(\sum_{k=1}^{j-1} \sin(\rho_k.\text{dir}), \sum_{k=1}^{j-1} \cos(\rho_k.\text{dir})\right)$
  $\rho_{\text{new}}.\text{t} \leftarrow \rho_i.\text{t}$
  $T_{\text{processed}} \leftarrow T_{\text{processed}} \cup \rho_{\text{new}}$
  $i \leftarrow j$
end while
return $T_{\text{processed}}$

5.4.4 Candidate Set

The candidate set $C_{\text{set}} = (L_1, L_2, \ldots, L_n)$ of trajectory $\text{T}_{\text{gps}} = (\rho_1, \rho_2, \ldots, \rho_n)$ is a set of location sets, where the location set $L_{\text{i}} = (l_i^1, l_i^2, \ldots, l_i^m)$ comprises the locations of point ρ_i candidates, and each location $l_i^j = (e, \textit{offset})$ is a tuple containing the location's arc and offset.

We propose a candidate set selection algorithm based on a custom heuristic that better adapts to the different road-network scenarios than the standard solution used by related work. Algorithm 2 presents an incremental implementation of the technique, which we base on the following concepts. The *close neighborhood* is a small radius from the vehicle's position. If few candidates are in this *close neighborhood*, we need to increase the search radius until we find a *reasonable number of candidates*. We also need to define the *maximum search radius* and the *maximum number of candidates* to bound the computational complexity. Therefore, we introduce the parameters D_{soft}, K_{soft}, D_{hard}, and K_{hard} that represent the *close neighborhood*, *reasonable number of candidates*, *maximum search radius*, and *maximum number of candidates*, respectively.



(a) Trajectory before preprocessing.

(b) Proposed preprocessing algorithm.

(c) Related work preprocessing technique.

Figure 5.3: Comparison of the proposed preprocessing technique and relate work.

Special Candidates

In some scenarios, a vehicle can record two subsequent points in which the first point looks further along a road segment than the second point. This scenario, illustrated in Figure 5.4, can occur if the vehicle circles a block or due to precision errors in the recorded locations. In the case of precision errors, most trajectory reconstruction algorithms will wrongly create a trajectory that circles a block to justify the positional errors. We try to fix this problem by generating *special candidates* (Line 14 of Algorithm 2). Whenever we find two candidate locations $l_i = (e_i, \mathit{offset}_i)$ and $l_{i+1} = (e_{i+1}, \mathit{offset}_{i+1})$ of points ρ_i and ρ_{i+1}, respectively, where $e_i = e_{i+1}$ and $\mathit{offset}_i > \mathit{offset}_{i+1}$, we generate two additional candidate locations as $l_{\text{new}} = (e_i, \frac{\mathit{offset}_i + \mathit{offset}_{i+1}}{2})$ for points ρ_i and ρ_{i+1}. We call *source special candidate* the new candidate location of ρ_i and *destination special candidate* the new candidate location of ρ_{i+1}. These new candidate locations allow the proposed reconstruction algorithm (Section 5.4.6) to evaluate a trajectory without circling the block.

5.4.5 Information Modeling

The solution construction component of the framework uses two probabilistic models: the observation model and the transition model. The observation model describes the likelihood of a vehicle at location l generating point ρ, and the transition model provides the probability of a vehicle moving from location l^a of some point ρ_i to location l^b of the next point ρ_{i+1}.

Table 5.1: A trajectory of the Rio Center dataset and the results of preprocessing algorithms.

Method	#	Lon.	Lat.	T. (s)	Spd (m/s)	Dir.
Original	1	-43.2070	-22.9236	120	45.17	277.95
	2	-43.2085	-22.9234	160	0.02	0.00
	3	-43.2087	-22.9233	170	16.34	334.29
	4	-43.2087	-22.9230	181	1.46	32.86
	5	-43.2087	-22.9230	200	0.05	0.00
	6	-43.2087	-22.9229	227	10.54	328.87
	7	-43.2088	-22.9227	240	3.08	319.66
Proposed	1	-43.2070	-22.9236	120	45.17	277.95
	2	-43.2086	-22.9233	160	8.18	347.14
	3	-43.20875	-22.9229	181	3.78	349.80
[140]	1	-43.2070	-22.9236	120	45.17	277.95
	2	-43.2085	-22.9234	160	0.02	0.00
	3	-43.2087	-22.9230	181	1.46	32.86

Observation Probability

Most related studies use the *noisy generative model* to calculate the observation probability. The observation probability $o(\rho \mid l)$ is associated with the distance between point ρ and candidate location l, i.e., $o(\rho \mid l) = D(\|\rho, l\|_{geo})$, where $\|\cdot\|_{geo}$ computes the largest circle distance between the point and the location, and D is some probabilistic distribution that aims to describe the likelihood of occurrence of errors equivalent to $\|\cdot\|_{geo}$ meters while recording a GPS point.

We propose an observation model that combines the noisy generative model with a model A extracted from the difference between the vehicle bearing and the road segment angle. The proposed observation probability $o(\rho \mid l)$ has two components, one based on distance errors and another on angle errors. For the first component, the literature shows that the distribution of the errors of GPS measurements is approximately a zero-mean Gaussian. As for the second component, we analyze the Rio Center Dataset for Trajectory Reconstruction to fit the data with a proper distribution function A, and find that it fits a beta distribution approximately (angles differences are normalized by

Algorithm 2 Incremental procedure to compute the candidate set of a trajectory

Input: $\mathrm{T}_{\mathrm{gps}} = (\rho_1, \rho_2, \ldots, \rho_n)$: GPS-based trajectory, $\mathcal{N} = (\mathcal{V}, \mathcal{E})$: road network graph, K_{soft}, K_{hard}, D_{soft}, D_{hard}

Output: The ordered candidate set $C_{\mathrm{set}} = (L_1, L_2, \ldots, L_n)$

$C_{\mathrm{set}} \leftarrow \emptyset$
for each $\rho \in \mathrm{T}_{\mathrm{gps}}$ **do**
  $L \leftarrow \emptyset$
  $e \leftarrow$ edge $\in \mathcal{E}$ closest to point ρ
  $d \leftarrow$ distance from ρ to e
  while $|L| < K_{\mathrm{hard}} \wedge d \leq D_{\mathrm{hard}} \wedge (|L| < K_{\mathrm{soft}} \vee d \leq D_{\mathrm{soft}})$ **do**
    offset $\leftarrow$ distance between the starting point of e and the projection of ρ on e
    $L \leftarrow L \cup (e, \mathit{offset})$
    $e \leftarrow$ next edge $\in \mathcal{E}$ closest to point ρ
    $d \leftarrow$ distance from ρ to e
  end while
  $C_{\mathrm{set}} \leftarrow C_{\mathrm{set}} \cup L$
end for
$C_{\mathrm{set}} \leftarrow C_{\mathrm{set}} \cup$ compute *special candidates* from C_{set}
return C_{set}

dividing them by 180).

Formally, the proposed observation probability $o(\rho \mid l)$ is:

$$\begin{aligned} o(\rho \mid l) &= \mathcal{N}(\|\rho, l\|_{geo}) \times \mathcal{B}(\|\rho, e\|_{\theta}) \\ &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{\|\rho,l\|_{\mathrm{geo}}}{2\sigma}\right)^2} \times \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1} \end{aligned} \tag{5.2}$$

where B is the beta function, σ, α, and β are the parameters of the normal and beta distributions, e is the arc of location l, and $x = \|\rho, e\|_{\theta}$ is the angle difference between the vehicle angle at ρ and the angle of the arc e.

Transition Probability

The literature shows that the probability of a vehicle at the location l_i^a of point ρ_i moving to the location l_{i+1}^b of point ρ_{i+1} is proportional to the absolute difference between $\|\rho_i, \rho_{i+1}\|_{geo}$, and $\|l_i^a, l_{i+1}^b\|_{route}$, where $\|\cdot\|_{route}$ computes the shortest path distance between given locations in the road network [140] [124]. Besides that, Newson and Krumm

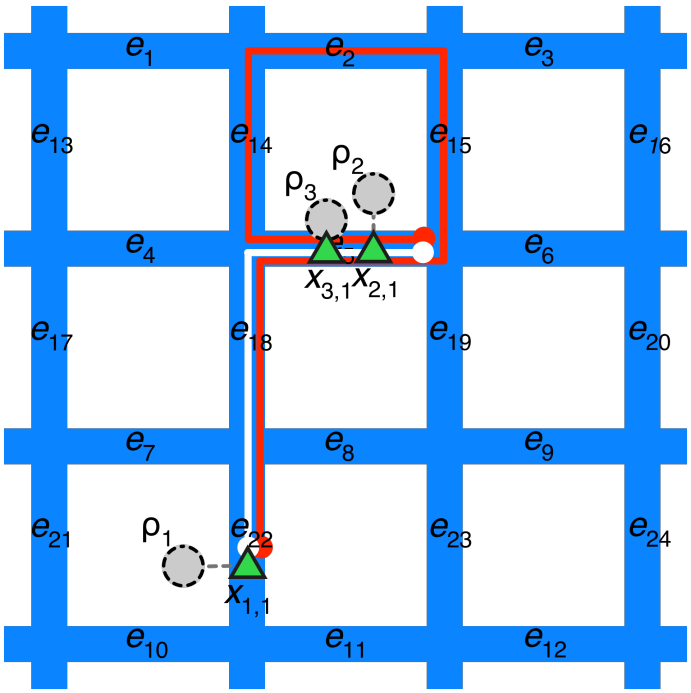


Figure 5.4: Example of a GPS trajectory with positional errors that hinder the trajectory's reconstruction. The white line corresponds to the correct path, and the red line corresponds to the wrongly matched path that most trajectory reconstruction solutions find. Our framework fix this by creating additional candidates, called *special candidates*.

[140] observed that the difference between those distances follows an exponential distribution in a real-world vehicle trajectory.

We propose a transition model based on the work of Newson and Krumm [140], but that considers the *special candidates*. The transition probability between candidate locations l_i^a of point ρ_i and l_{i+1}^b point ρ_{i+1} is zero in the following three scenarios, which we call "special candidate scenarios":

1. l_i^a is a non-special candidate and l_{i+1}^b is a special destination candidate.
2. l_i^a is a special source candidate and l_{i+1}^b is not a special destination candidate.
3. l_i^a is a special destination candidate and l_{i+1}^b is also a special destination candidate.

Another scenario in which we assign zero to the transition probability is when the speed required to travel from candidate location l_i^a to candidate location l_{i+1}^b is unrealistic. For that, we assign zero to the transition probability between locations l_i^a and l_{i+1}^b if the

average speed required to travel the shortest path between those locations is above a given threshold $v_{threshold}$.

Formally, the transition probability $T(l_i^a \rightarrow l_{i+1}^b)$ is:

$$T(l_i^a \rightarrow l_{i+1}^b) = \begin{cases} 0, & \text{if } v > v_{threshold} \text{ or is a "special candidate scenario"} \\ \frac{1}{\beta} e^{-\frac{E_r(l_i^a \rightarrow c_{i+1}^b)}{\beta}}, & \text{otherwise} \end{cases} \tag{5.3}$$

where

$$E_r(l_i^a \rightarrow l_{i+1}^b) = \left| \|\rho_i, \rho_{i+1}\|_{geo} - \|l_i^a, l_{i+1}^b\|_{route} \right| \tag{5.4}$$

and

$$v = \frac{\|l_i^a, l_{i+1}^b\|_{route}}{\rho_i.t - \rho_{i+1}.t}. \tag{5.5}$$

Our framework keeps the list of arcs representing the shortest path between the candidates' locations while computing the routing distances to reconstruct the entire trajectory later.

5.4.6 Solution Construction

The solution construction component creates a Hidden Markov Model (HMM) based on the observation and transition probabilities to reconstruct the road-network constrained trajectory. Widely used by map-matching algorithms, HMM is a statistical model that describes a system that follows a Markov process with hidden (unobservable) states. In general, the road segments and the trajectory points represents the hidden states and observations of the HMM, respectively. Then, to solve the trajectory reconstruction problem, we are interested in finding the "most likely explanation" for the specific sequence of observations (trajectory points), given efficiently by the Viterbi algorithm.

Figure 5.5 illustrates the proposed HMM-based model to reconstruct a given vehicle trajectory. The states of the HMM are the set of all pairs (ρ_i, l_i^a) of trajectory points ρ_i and candidate locations l_i^a, and the trajectory points ρ_i are the observations of the model. The emission probability $P(\rho_i \mid (\rho_i, l_i^a))$ and the transition probability $T(l_i^a \rightarrow l_{i+1}^b)$ are given by Equations 5.2 and 5.3, respectively, as described in Section 5.4.5. The initial probability π_k of the state (ρ_i, l_i^a) is computed with Equation 5.2 if i is equal to one (ρ_i is the first point of the trajectory), and zero otherwise.

The road-network constrained trajectory comes from the most likely sequence of hidden states of the HMM. Therefore, we use the Viterbi algorithm to compute the best

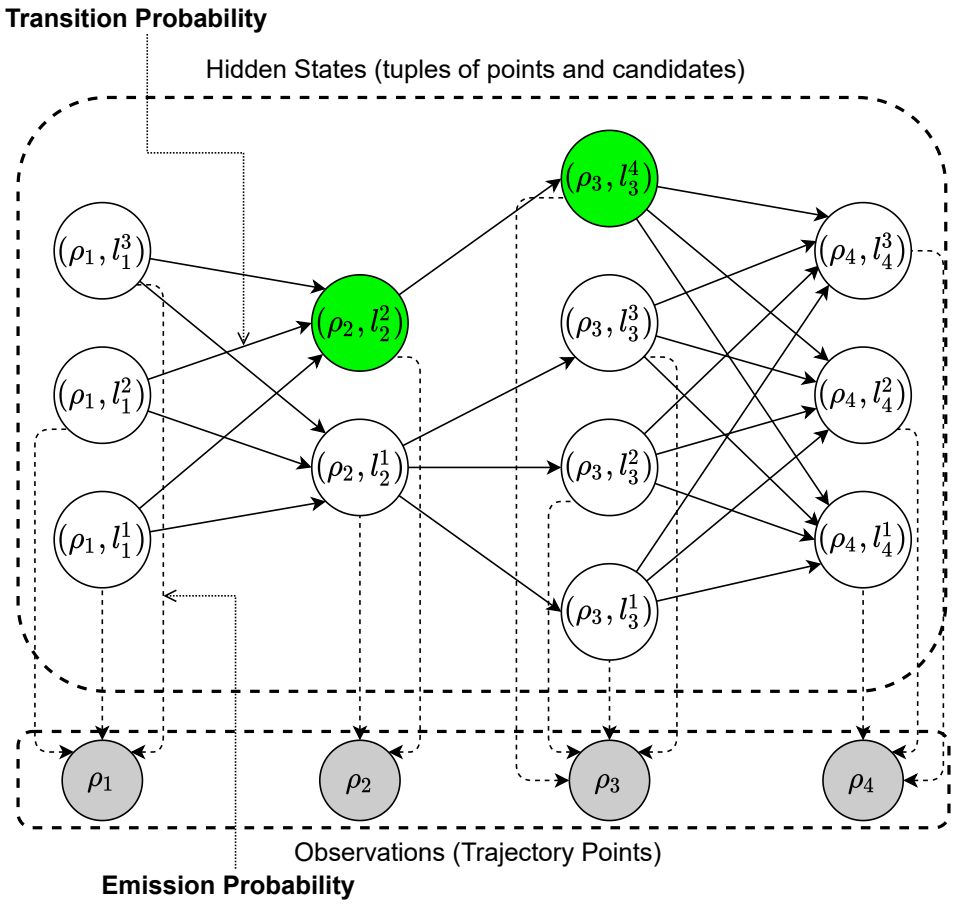


Figure 5.5: Example of the proposed HMM-based model to reconstruct a given vehicle trajectory. The states represented by green circles correspond to the *special candidates*.

path. The time complexity of the Viterbi algorithm is $O(|T| \cdot |\mathcal{E}|^2)$. Still, as only a small subset of the states (the candidates) and transitions have a probability greater than zero, the time complexity of our implementation is $O(|T| \cdot {K_{\text{hard}}}^2)$. The reconstructed trajectory is the concatenation of the arcs from the states (ρ_i, l_i^a) returned by the Viterbi algorithm, filled with the arcs of the shortest paths between these states.

5.5 Evaluation

This section evaluates the effectiveness and efficiency of the proposed framework to reconstruct vehicular trajectories and compares it to some related work. We replicated each experiment five times and plotted the graphs with a confidence interval of 95%.

Also, we implemented the proposed framework and related work methods using the C++ programming language. We performed all experiments on the same platform: a server running OS Ubuntu 18.04.4 LTS with 128 GB of memory RAM, processor Intel(R) Core(TM) i9-9900X CPU @ 3.50 GHz.

5.5.1 Dataset

We use the Rio Center Dataset of Trajectory Reconstruction (Chapter 4) and another real-world dataset [107] in the experiments. The other dataset [107] consists of GPS trajectories worldwide. Each of its trajectories is associated with a different map and has a sampling interval of one second. We use this dataset to evaluate the algorithms with trajectories of varying sampling rates. To do so, we select 16 trajectories and, for each one of them, generate others with sampling intervals ranging from 10 to 120 seconds, with increments of 10 seconds, resulting in 192 trajectories.

5.5.2 Evaluation Metrics

We assess the accuracy of the methods with two widely used metrics: Road Mismatch Fraction (RMF) [140] and F_1 score. As for the performance, we measure the algorithms' processing time. The RMF is given by Equation 5.6, where d_+ is the total length of the false-positive roads, d_- is the total length of the false-negative roads, and d_0 is the total length of ground truth road-network constrained trajectory.

$$RMF = \frac{d_+ + d_-}{d_0} \tag{5.6}$$

Equation 5.7 computes the F_1 score using the precision and recall of the data. The precision (Pre) is the total length of the true positive roads divided by the total length of the reconstructed trajectory. The recall (Rec) is the total length of the reconstructed trajectory divided by the whole length of the ground truth road-network constrained trajectory.

$$F_1 = 2 \times \frac{Pre \times Rec}{Pre + Rec} \tag{5.7}$$

5.5.3 Comparison Methods

We compare the proposed framework to three approaches from related work: 1) AntMapper [70], HMM (Newson and Krumm) [140], and ST-Matching [124]. The parameter

settings of the AntMapper, HMM (Newson and Krumm), and ST-Matching algorithms are the values recommended in their proposals [140] [124] [70]. In addition, we empowered the related work solutions with the proposed technique to fill the gaps between map-matched points (i.e., we include the arcs comprising the shortest paths between the matched locations), so they return valid trajectories, as defined in Section 5.3.

5.5.4 Parameter Estimation

We first estimate some of the parameters of the proposed framework (i.e., K_{soft}, K_{hard}, D_{soft}, D_{hard}, α, and β of the observation probability) by analyzing the trajectories of the Rio Center Dataset for Trajectory Reconstruction. Then, we replicate the parameters through all experiments, including those of the second dataset, to verify that the parameters are not biased towards the dataset.

The parameters of the candidate set selection algorithm improve the overall performance of the framework because they limit the number of arcs (road segments) that the other algorithms of the framework are going to process. However, we need to use values that include all correct arcs in the candidate sets. For this, we calculate the ten closest arcs of each point of the Rio Center dataset and order them by distance (we checked that for this dataset, the correct arcs are always one of the ten closest arcs). Figure 5.6 shows the results.

The red circles in Figure 5.6 represent the correct arcs, and the blue circles are the other arcs. The dataset comprises 395 trajectory points, so the figure contains 395 red circles and 3,555 blue circles (9 for each trajectory point). Two trajectory points have the tenth closest arc as the correct arc. These arcs are at 47.37 and 23.42 meters from their respective points. Furthermore, the farthest correct arc is at a distance of 86.49 meters from its trajectory point, and this arc is the seventh closest arc of the point. Therefore, the optimal values of D_{soft} and K_{soft} to compute the candidate sets of the Rio Center dataset with the least number of arcs without excluding any correct arc are 47.37 (we round up D_{soft} to 50) and 7, respectively. The blue region (diagonal blue lines) contains the arcs limited by the optimal K_{soft}, and the red region (diagonal red lines) contains the arcs defined by the optimal D_{soft}. The union of these two regions comprises the candidate arcs determined by the proposed algorithm. On the other side, the related work methods would at least include the arcs outside these regions and therefore are less efficient.

The parameters α and β of the beta distribution are used to calculate observation

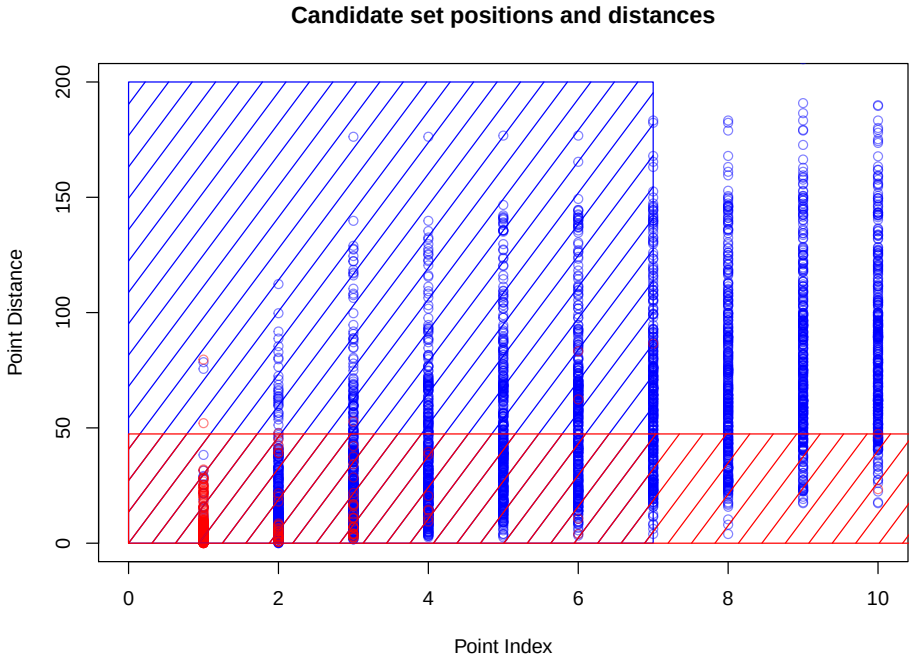


Figure 5.6: Estimation of the parameters of the candidate set algorithm. The circles represents the ten closest arc candidates for each trajectory point of the Rio Center dataset (red circles are the correct arcs. The intersection of the regions with diagonal lines comprises the arcs included in the candidate set by the proposed algorithm.

probabilities. We use the maximum likelihood method to fit the distribution of the angle differences between the points and their correct arcs with the beta distribution (the angles differences were normalized by dividing them by 180). Figure 5.7 presents the results of the distribution fitting, in which we find $\alpha = 0.3636516$ and $\beta = 2.0333713$.

Table 5.2 shows the values of all parameters. The importance of the standard deviation σ (i.e., GPS noise) and β parameter (exponential distribution in Equation 5.3) were fixed based on experimental results.

5.5.5 Results

We first compare the methods using the Rio Center dataset, which has trajectories of different characteristics, but most of them with a sampling interval of 30 seconds. Figure 5.8 shows the results.

One of the main factors that impact processing time is the approach used to limit

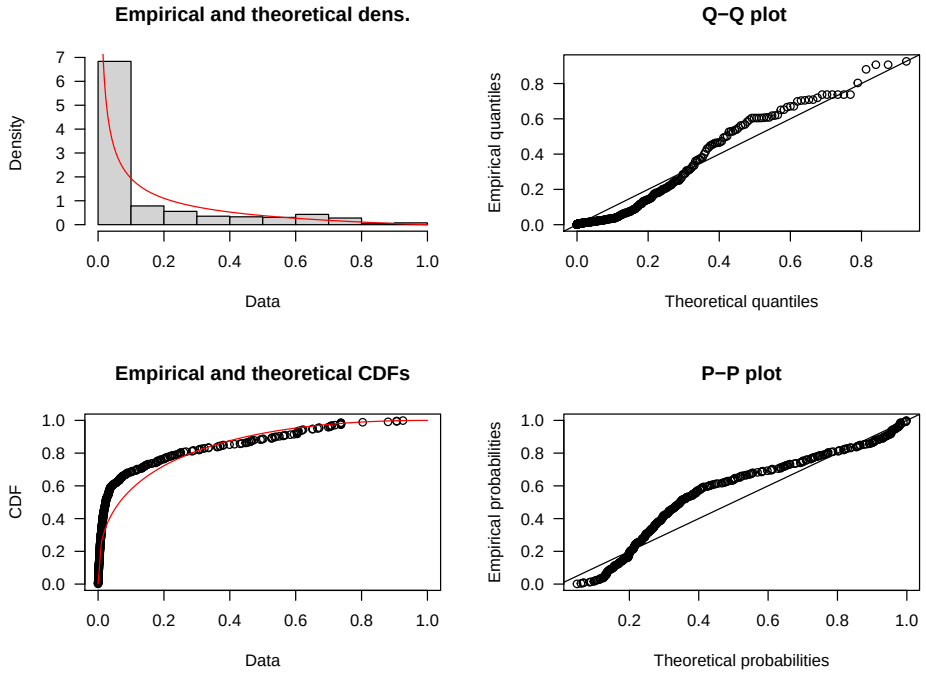


Figure 5.7: Fitting the distribution beta of the angles difference by maximum likelihood.

the number of analyzed road segments during the reconstruction process. Therefore, the proposed framework presents the lowest processing time due to the proposed candidate set selection algorithm that effectively bounds the number of road segments processed. On the other hand, Newson and Krumm's technique [140] has the worst processing time because it considers all road segments within a distance of 200 meters of the points. We verified that, if empowered with our candidate set selection algorithm, the processing time of Newson and Krumm's technique would be similar to the processing time of our framework and ST-Matching algorithm because they have the same time complexity.

Figure 5.8 shows that the proposed algorithm outperforms the related work in terms of accuracy. While the lower the road mismatch fraction (RMF), the better the results, the F1 score is the inverse, so the higher, the better. Both metrics show that the proposed framework produces more accurate road-network constrained trajectories, which determines the effectiveness of the models employed in the frameworks' algorithms. Besides that, these results show that our preprocessing algorithm, which reduces the amount of processed data, does not erroneously ignore data that may be useful for the reconstruction process.

Figure 5.9 analyzes the techniques when subject to trajectories with different sam-

Table 5.2: Framework default parameters value.

Parameter	Value
K_{soft}	7
K_{hard}	10
D_{soft}	50 m
D_{hard}	100 m
α	0.36
β (observation prob.)	2.03
σ	20
β (transition prob.)	$\frac{1}{\ln(2)} \times 20$

pling rates using the dataset proposed in [107]. Although most real-world scenarios have sampling intervals shorter than 30 seconds, we evaluate trajectories with sampling intervals of up to two minutes. Overall, the proposed framework presents the best results for all sampling rates, and Newson and Krumm's technique displays similar results for sampling intervals greater than one minute. However, the processing time of the proposed framework is significantly shorter than the related work in all scenarios. One of the main reasons for the low accuracy of related work to reconstruct trajectories with a high sampling rate (less than 20 seconds) is the situation described in Section 5.4.4. Our framework presents a higher accuracy even in these scenarios because of the proposed technique that includes "special candidates," as described in Section 5.4.4.

Lastly, we analyze the impact of the candidate set selection technique on the performance of the proposed framework. Figure 5.10 shows the experimental results ranging K_{soft} from 1 to 10 and D_{soft} from 10 to 200 meters (K_{hard} and D_{hard} are fixed in 10 and 200, respectively). The results where $K_{\text{soft}} = K_{\text{hard}}$ or $D_{\text{soft}} = D_{\text{hard}}$ demonstrate the scenario in which our framework uses the candidate set selection technique commonly used by related work (i.e., the candidates are all arcs within a maximum search radius, limited to the k-closest arcs). As expected, the higher the value of K_{soft} or D_{soft}, the higher the processing time because more arcs are processed. Besides that, when K_{soft} and D_{soft} are less than their optimal values shown in Section 5.5.4 (i.e., $K_{\text{soft}} < 5$ and $D_{\text{soft}} < 50$), the accuracy of the reconstructed trajectory drops. These results show that the proposed technique reduces the processing time without compromising the framework's accuracy.

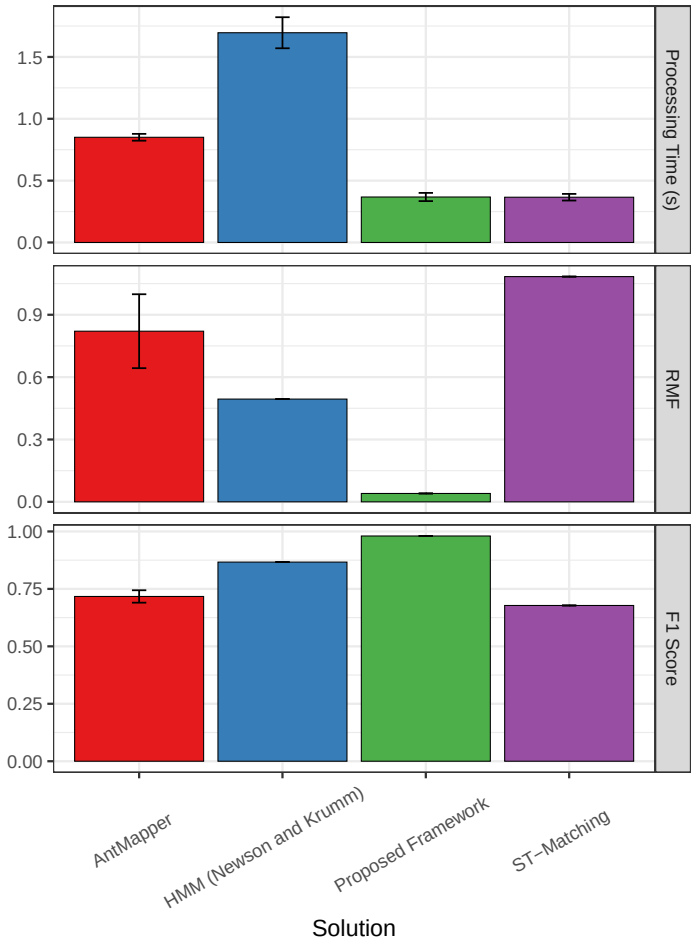


Figure 5.8: Experimental results using the *Rio Center Dataset for Trajectory Reconstruction*.

5.6 Chapter Remarks

In this chapter, we proposed a novel framework to reconstruct road-network constrained trajectories from GPS-based trajectories. The framework contains several components, such as the candidate set selection algorithm, which optimizes the number of processed road segments and, thus, reduces the processing time of the whole reconstruction process. Furthermore, the proposed models improve the accuracy of the reconstructed trajectories.

We compared the framework to some related work (i.e., AntMapper [70], Newson and Krumm [140], and ST-Matching [124]) through experiments using two real-world trajectory datasets. These datasets allowed us to evaluate the ability of the methods to reconstruct trajectories with different characteristics, including trajectories of low and

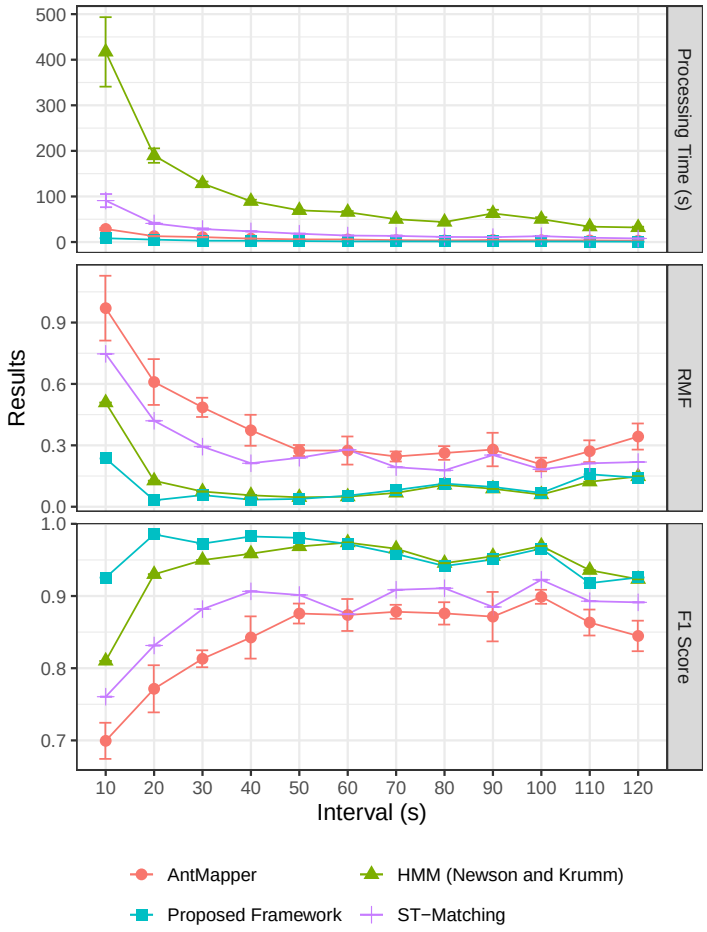


Figure 5.9: Results of the evaluation with different sampling intervals through the public available real-world dataset [107].

high sampling rates. The framework outperformed the related work in all scenarios both in accuracy and processing time.

In future work, we plan to investigate the framework's ability to perform online trajectory reconstruction. An initial step would be to assess the accuracy degradation when reconstructing "partial trajectories," as this reproduces the online reconstruction scenario.

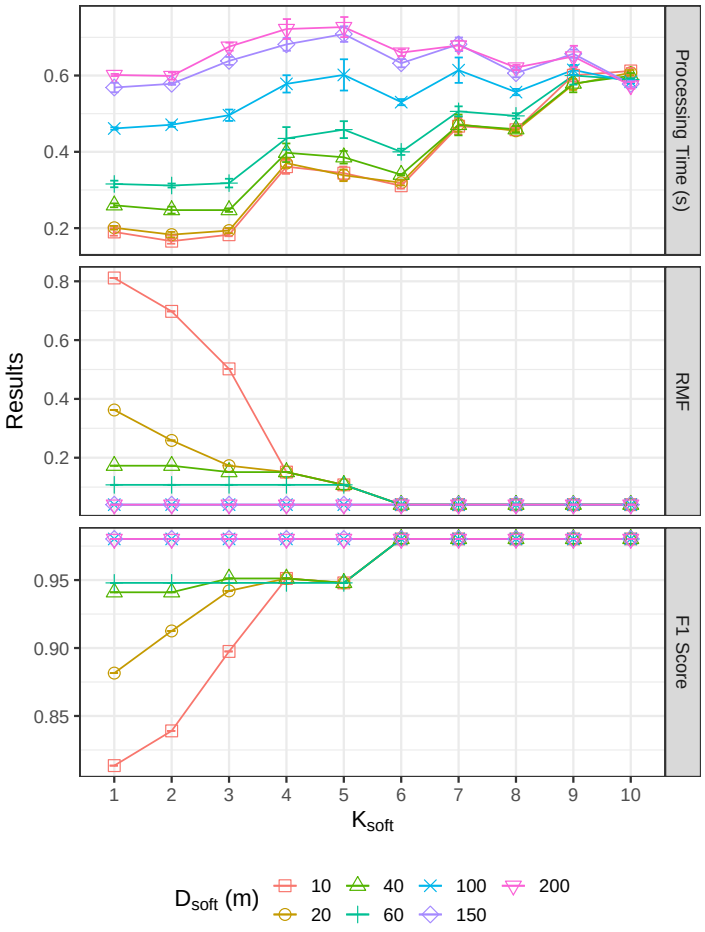


Figure 5.10: Results of the impact of the candidate set selection technique on the performance of the proposed framework using the *Rio Center Dataset for Trajectory Reconstruction*.

Chapter 6

On the Prediction of Large-Scale Road-Network Constrained Trajectories

Trajectory data mining-based applications benefit from the increasing availability of vehicle trajectory and road network datasets. For instance, the application of trajectory prediction makes it possible to design more efficient routing protocols for vehicular networks. This chapter proposes a novel cluster-based framework for the long-term prediction of road-network constrained trajectories. The framework employs a new hierarchical agglomerative clustering algorithm and trains prediction models from historical trajectory datasets. Experimental results show the framework's effectiveness and efficiency to predict trajectories with different characteristics in a new real-world, large-scale scenario. Furthermore, the framework outperformed the related work in terms of prediction accuracy and time complexity.

6.1 Introduction

The rapidly changing mobility aspects of civilization and the advances of location-acquisition technologies generate massive trajectory data. Given its applicability in different research fields, this kind of data, called big trajectory data, is a critical component for developing solutions to many problems found in modern society. For instance, trajectory data is essential for traffic analysis, location-based social networks, vehicular networks, and other data mining applications.

Mobility prediction is a pivotal application of big trajectory data. By analyzing historical trajectory datasets, we can extract knowledge regarding a city's mobility patterns and create, for instance, models of trajectory prediction, destination prediction, and travel time prediction. Next, we can apply these prediction models in applications such as the design of Intelligent Vehicular Networks.

In summary, vehicular networks are wireless communication systems that allow exchanging information between vehicles (V2V communication) and between vehicles and roadside units (V2I communication) through typically a Dedicated Short Range Communication (DSRC) protocol. This type of network has different communication challenges because of the vehicular environment's characteristics. For instance, the network topology is highly dynamic given the movement of vehicles. Therefore, we can design more efficient routing protocols and services for vehicular networks by predicting the vehicles' trajectories [48].

The most suitable trajectory prediction technique for a given application depends on several factors, such as the data representation and the prediction range. Some prediction solutions use raw trajectories as input data. In contrast, others apply preprocessing techniques to convert raw trajectories into road-network constrained trajectories, which are more suitable for describing vehicle trajectories than raw trajectories [208, 46]. The prediction methods can perform short-term or long-term predictions, i.e., maneuvers of a few seconds or many road segments ahead, respectively. Most of the proposals in the literature are for short-term trajectory prediction and, thus, are useful for applications such as safety and autonomous vehicles [170, 100, 147, 201, 127, 2]. On the other side, long-term trajectory prediction is useful for developing Intelligent Vehicular Networks [43, 6, 55] and improving urban mobility in a broader sense [172, 44, 49].

Currently, most solutions in the literature for trajectory prediction use raw trajectories as input data to perform short-term predictions. In this work, we focus on the long-term prediction of road-network constrained trajectories. When accurately constructed, road-network constrained trajectories are most suitable to represent vehicle trajectories than raw trajectories [46]. Therefore, we propose a novel and scalable cluster-based prediction framework for road-network constrained trajectories. The framework is appropriate to predict the next road segments of a partial trajectory using a dense dataset of historical trajectories.

The key contributions of this Chapter are the following:

- A novel hierarchical agglomerative clustering algorithm for road-network constrained trajectories that automatically detects the appropriate number of clusters.

- A new trajectory prediction framework that employs the proposed clustering algorithm. The framework employs multiple prediction models extracted from clustered data, and we show that the trajectory's length affects the prediction accuracy provided by models these models. Therefore, we propose a new hybrid prediction algorithm that considers the trajectory's length to dynamically select the most suitable model to predict the trajectory's next steps.

- An evaluation of the proposed framework and some related solutions using a new large-scale dataset. Experimental results show that the new framework outperforms the related work in terms of prediction accuracy and time complexity.

We organize the remainder of this Chapter as follows. Section 6.2 presents the related work. Section 6.3 introduces some basic definitions and formulates the problem. Section 6.4 compares different similarity measures for the prediction of road-network constrained trajectories. Section 6.5 describes the proposed trajectory prediction framework. Section 6.6 analyzes the frameworks' performance and compares it with some related work. Finally, Section 6.7 concludes this work.

6.2 Related Work

The road-network constrained trajectory representation is a more precise model than raw trajectories. However, most long-term vehicle trajectory prediction methods use raw trajectories as input data [26]. One reason for this is the lack of publicly available datasets of road-network constrained trajectories. Moreover, we rely on advanced algorithms and data containing accurate road network definitions to precisely reconstruct road network-constrained trajectories from raw trajectories [46]. Such algorithms and data are becoming more available in recent years. Therefore, few trajectory prediction methods use the road-network constrained trajectory representations [64, 182, 161], and most of them use cluster-based approaches.

Some of the advantages of cluster-based approaches for the prediction of road-network constrained trajectories are the following. First, prediction models can be more precise if created from clusters of similar trajectories. For instance, we show in Section 6.6, the improvements in the accuracy of predictions performed with models built from clustered trajectories compared to models built from non-clustered trajectories. Improvements are most noticeable when we try to predict several steps. Another benefit of advanced cluster-based approaches is that most proposed scalable clustering algorithms result in techniques

that we can use with large-scale datasets. Other methods, such as the widely used solution proposed by Froehlich and Krumm [64], require the computation of the distance matrix between all trajectories, which is computationally prohibitive when dealing with large-scale datasets.

We can summarize the process of cluster-based road-network trajectory prediction, found in the literature, as a sequence of the following steps. The first step forms clusters of similar trajectories by applying a clustering algorithm (e.g., DBSCAN [61], clusiVAT [108], and dendrogram clustering [64]) powered by a custom similarity measure for vehicle trajectories. The second step creates prediction models based on the trajectories of each cluster. In most cases, the model is a Markov chain or a representative trajectory constructed by merging the cluster's trajectories. Finally, the last step predicts a given partial trajectory by finding the most suitable cluster for the partial trajectory and then applies the cluster's model constructed in the second step.

Tiwari et al. [182] proposed a method that extends the solution for predicting raw trajectories proposed by Froehlich and Krumm [64]. For this, they introduced an additional step that converts raw trajectories into road-network constrained trajectories. The solution proposed by Froehlich and Krumm [64] works as follows. First, they convert trips to regular routes by clustering similar trips. They proposed a similarity measure based on the Hausdorff distance [47] that computes the mean point-to-segment distance between the compared trips. The similarity measure considers the directionality by matching each trip's point to segments of the other trip with an index equal to or greater than the last closest segment found. They use the similarity measure to create a trip similarity matrix by comparing every trip in $O(n^2)$. Next, they use the matrix to cluster similar trips by applying the *dendrogram clustering* and convert the clusters to routes by merging each cluster's trips. The merging algorithm resamples trips in the cluster at even distance intervals, and the set of average points at every interval comprises the final route. Finally, the route prediction works by continuously calculating the distance between the current trip and existing routes. Then, the predicted route can be (i) the closest route, or (ii) the closest route and a given probability based on past predictions (which depend on the knowledge about the correctness of past predictions).

Rathore et al. [161] proposed the *Traj-clusiVAT-based TP* framework based on the Markov model to make long-term predictions. That framework is based on the Markov model and makes long-term predictions. The Traj-clusiVAT-based TP framework employs a modified version of the two-stage clusiVAT clustering algorithm [108], called Traj-clusiVAT. One of the modified two-stage traj-clusiVAT algorithm's improvements

is that it implements a new method to calculate, for each cluster, a *representative trajectory*, which is used to distribute new trajectories to an existing cluster. Because of that, the Traj-clusiVAT-based TP framework can handle big trajectory datasets in dense road networks efficiently. The *Traj-clusiVAT-based TP* framework has some drawbacks. The framework represents the road network as an undirected graph, and we can not represent most of the real-world road networks as undirected graphs. Other issues are related to the dissimilarity measure *trajDTW*. First, *trajDTW* can not compute the distance between trajectories if the road network is not strongly connected. Furthermore, *trajDTW* defines the window parameter w for the DTW algorithm as $w = \frac{1}{2} \times \min\{|T_1|, |T_2|\}$, but DTW does not work if $w < ||T_1| - |T_2||$. The algorithm to compute the representative trajectory (Algorithm 1 in [161]) for each cluster presents many issues. The algorithm relies on an input parameter ($minT$) to determine the minimum number of trajectories required to cross a given road segment so that it can be part of the representative trajectory. Thus, depending on the dataset and the input value for the $minT$ parameter, the algorithm will not find a representative trajectory for some clusters. Second, if we strictly follow the algorithm, we can get an invalid representative trajectory when selecting the representative trajectory's next road segment. This situation can happen because the last road segment of the partial representative trajectory may not be connected to the next selected road segment. Finally, the algorithm can enter an infinite loop if, for two connected road segments A and B, the next step of most trajectories that cross A is B, and the next step of most trajectories that cross B is A.

In this chapter, we propose a new cluster-based prediction framework for directed road networks. The prediction framework employs a new hierarchical agglomerative clustering algorithm, which generates more meaningful clusters to calculate accurate prediction models. Furthermore, we employ the DSL distance function to compare road-network constrained trajectories, which works even if the road network is not strongly connected. Also, we propose a new algorithm to compute the cluster's representative trajectory that fixes the issues of the algorithm used in the Traj-clusiVAT-based TP framework. Finally, we introduce a new hybrid prediction algorithm that considers the trajectory's length to select the most suitable prediction model and improves the prediction stage's performance.

6.3 Preliminaries and Problem Formulation

In the following, we introduce some definitions to formulate the problem and present some concepts required in the remainder of the Chapter.

6.3.1 Problem Definition

Definition. (Partial Trajectory) A partial trajectory $\mathrm{T}^{1:P}$ of a given trajectory T is a prefix of T. That is, $\mathrm{T}^{1:P} = (t_1, t_2, \ldots, t_P)$, where $P \leq n$.

Based on the definition of the road network, GPS-based trajectory, and road-network constrained trajectory presented in Chapter 2, and the definition of Partial Trajectory presented above, we formulate the following problems:

Problem 1. (i-step Trajectory Prediction) Given a road network $\mathcal{N}$, a historical trajectory dataset with n trajectories denoted by $\Gamma = (\mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_n)$, a partial trajectory $\mathrm{T}^{1:P} = (t_1, t_2, \ldots, t_P)$, and an integer $i > 0$, the task is to predict the future road segments $t_{P+1}, t_{P+2}, \ldots, t_{P+i}$. That is, the goal is to predict the trajectory T_{pred}, where $|\mathrm{T}_{\text{pred}}| = P + i$ and $\mathrm{T}^{1:P}_{\text{pred}} = \mathrm{T}^{1:P}$.

Problem 2. (Trajectory Prediction) Given a road network $\mathcal{N}$, a historical trajectory dataset with n trajectories denoted by $\Gamma = (\mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_n)$, and a partial trajectory $\mathrm{T}^{1:P} = (t_1, t_2, \ldots, t_P)$, the task is to predict the future road segments $t_{P+1}, t_{P+2}, \ldots, t_{P+n}$, where n is unknown. That is, the trajectory prediction problem includes the problem of detecting the end of the trajectory.

Note that, from the above definitions, the vehicle is always in a road segment represented by an arc, and the predictions represent the possible subsequent road segments connected to the vehicle's current arc. Therefore, the predictions depend on how we map the road network into the graph representation. For instance, it is better to represent a small roundabout as a single node instead of many small arcs forming a circle. Thus, in this situation, the prediction will be regarding the next road segment that the vehicle will travel after the roundabout (i.e., excluding the part referring to the roundabout). Besides that, in some scenarios, the road network has many connected components, and some components are not strongly connected. Thus, by definition, the predicted arcs will always be from the same connected component in which the vehicle is.

In this Chapter, we focus on the i-step trajectory prediction, i.e., Problem 1. The trajectory prediction considering that the end of trajectory is unknown, i.e., Problem 2, is part of future work.

6.4 Similarity Measures for the Prediction of Road-Network Constrained Trajectories

A fundamental factor in any trajectory data mining task's performance is to identify a suitable distance function/similarity measure to compare trajectories. For instance, some methods for road-network constrained trajectories, such as TrajDTW, can provide a notion of distance between trajectories that are close to each other even if they do not have segments in common. These methods are useful for applications such as destination prediction. However, other applications such as trajectory prediction require exact trajectory matching (i.e., when the accuracy depends on matching the road segments' sequence). For these applications, distance functions that consider the intersection set of road segments between compared trajectories, such as the Dissimilarity with length (DSL) [196] and the Longest Common Road Segments (LCRS) [211], are more appropriate.

We assessed the clustering tendency of the Rio Center Dataset for Data Mining (described in Chapter 4) using the Hopkins statistic [84, 16] and the distance functions DSL, LCRS, and TrajDTW to demonstrate the advantage of distance functions that perform exact matching of road segments for applications such as trajectory prediction. A Hopkins statistic value close to 1 indicates that the data is highly clustered, and a value close to 0.5 suggests that the data is random. Using a sample of 1000 trajectories, the Hopkins statistic values applying the DSL, LCRS, and TrajDTW distance functions are 0.94, 0.90, and 0.68, respectively. This result indicates that the distance functions that perform exact matching of road segments (e.g., DSL and LCRS) are more appropriate to cluster the trajectories.

In this work, we employ the DSL, which is similar to the LCRS, in the trajectory prediction problem. As discussed in detail in Chapter 3, the DSL between two trajectories is the sum of the road lengths of their disjoint set divided by the sum of the lengths of both trajectories.

6.5 Proposed Prediction Framework

This section presents a novel cluster-based framework to predict long-term road-network constrained trajectories. The framework has a component responsible for training prediction models from past trajectories and another one responsible for predicting a partial trajectory based on the models. We first cluster the past trajectories using a new hierarchical agglomerative clustering algorithm and extract a representative trajectory (RT) and a first-order Markov chain model (MM) for each cluster. The cluster's representative trajectories and Markov chain models, defined in Sections 6.5.1 and 6.5.2, respectively, are used later for prediction.

6.5.1 Markov chain model

In the following, we define a cluster's transition matrix M that represents a Markov chain model. Each road segment of the road network represents a state of the Markov chain M, and for any two adjacent road segments e_i and e_j, we set the transition probability $M_{e_i \to e_j}$ from e_i to e_j as:

$$P(M_{e_i \to e_j}) = \frac{\mathrm{I}_{e_i \to e_j}}{\mathrm{R}_{e_i}} \tag{6.1}$$

where $\mathrm{I}_{e_i \to e_j}$ is the number of times the intersection $e_i \to e_j$ is crossed by the cluster's trajectories, and R_{e_i} is the number of times the cluster's trajectories traverse the road e_i. We also store in a count transition matrix C how many times the cluster's trajectories traversed each intersection $e_i \to e_j$.

We improve the framework's space and time complexities by using balanced trees instead of $N \times N$ matrices to store the M and C structures. That is because road networks are typically sparse graphs by nature, and creating quadratic matrices of large road networks can be infeasible. Besides that, some basic queries used in Markov chain-based predictors have linear time complexity.

6.5.2 Representative Trajectory (RT)

We use the idea of the representative trajectory (RT) and the transition matrices to detect the most suitable cluster for a given trajectory. This technique, described in Section 6.5.3, is applied to both the clustering algorithm and the prediction component. Most methods from the literature that compute a cluster's RT have high computational complexity [161]

or present other issues (such as those of the method used in the Traj-clusiVAT-based TP framework, discussed in Section 6.2). Therefore, we propose Algorithm 3 to compute the clusters' representative trajectories. The algorithm uses the cluster's trajectories and the densest road segments to construct the RT. The computed RT is not required to match any trajectory from the cluster.

Algorithm 3 creates one representative trajectory (RT) for each cluster's trajectories and chooses the RT with the maximum score as the cluster's RT. The process of computing the RT and the RT's score for each trajectory works as follows. We start the RT from the first road segment of the trajectory (Line 3). After, we choose the next road segment using the cluster's count transition matrix C (Line 5). We increment the score of the current RT (Line 8) by the number of trajectories that contain the intersection between the last and next road segments ($C_{\text{lastRoad}\rightarrow\text{nextRoad}}$). We repeat this process (Lines 4–13) until there is no next road segment, or the current RT already contains the next road segment.

6.5.3 Assignment of trajectories to a cluster

An essential function of the prediction framework is to assign trajectories to existing clusters. This step is used both in the clustering algorithm and in the prediction component. The straightforward idea for this task is to use the *nearest prototype rule* (NPR), which means to assign a given trajectory T to the cluster k that minimizes the distance between T and the cluster's representative trajectory. However, for some scenarios and distance functions, using NPR can result in bad assignments. For instance, Rathore et al. [161] observed poor performance of their undirected prediction framework when using NPR powered with the trajDTW distance function. Thus, they proposed a hybrid method that employs a probabilistic model to assign trajectories to a cluster and the nearest prototype rule as a fallback method.

We propose a method based on the technique presented by Rathore et al. [161]. The differences between our method and theirs are the following: (i) we use DSL instead of trajDTW, and (ii) we identify when it is not possible to find a suitable cluster for the trajectory. Algorithm 4 summarizes the proposed method. First, we select the cluster c where the trajectory T is most likely to occur, based on the transition matrices M (Lines 1–4). If we cannot find a cluster with a probability greater than zero, we use the nearest prototype rule (Line 5). However, if there is no RT_i with a non-empty intersection set with trajectory T (i.e., $\nexists i \in [1 \,..\, k] \Rightarrow DSL(T, RT_i) < 1$), the algorithm returns -1 to

Algorithm 3 Representative trajectory computation

Input: $\Gamma = \{T_1, T_2, ..., T_n\}$ – cluster's trajectories
C – cluster's count transition matrix
Output: RT $\sim (rt_1, rt_2, ..., rt_l)$ – the representative trajectory

1: RT $\leftarrow \emptyset$, bestScore $\leftarrow -1$
2: **for each** $T \sim (t_1, t_2, ..., t_n) \in \Gamma$ **do**
3: currentScore $\leftarrow 0$, lastRoad $\leftarrow t_1$, currentRT $\leftarrow$ (currentRoad)
4: **while** true **do**
5: nextRoad $\leftarrow \{e_j \mid j = \arg\min_j C_{\text{lastRoad}\rightarrow e_j}\}$
6: **if** nextRoad $\neq \varnothing \wedge$ nextRoad $\notin$ currentRT **then**
7: currentRT $\leftarrow$ currentRT $\cup$ nextRoad
8: currentScore $\leftarrow$ currentScore + $C_{\text{lastRoad}\rightarrow\text{nextRoad}}$
9: lastRoad $\leftarrow$ nextRoad
10: **else**
11: break
12: **end if**
13: **end while**
14: **if** currentScore $>$ bestScore **then**
15: bestScore $\leftarrow$ currentScore, RT $\leftarrow$ currentRT
16: **end if**
17: **end for**
18: **return** RT

indicate that the trajectory was not assigned to any cluster (Line 9).

6.5.4 Clustering algorithm for road-network constrained trajectories

Algorithm 5 presents the proposed hierarchical agglomerative clustering algorithm for road-network constrained trajectories. The algorithm, which automatically detects the number of clusters k, receives as input the trajectories dataset Γ and provides as output the clustered trajectories and the models used in the prediction component. That is, the output of the algorithm consists of the number of clusters k, the set of clusters ζ, the clusters' representative trajectories **RT**, the clusters' Markov processes transition matrices **M**, and the clusters' count transition matrices **C**.

Algorithm 4 Hybrid NPR

Input: $T \sim (t_1, t_2, ..., t_n)$ – a trajectory
$\mathbf{RT} \sim (RT^1, RT^2, ..., RT^k)$ – clusters' representative trajectories
$\mathbf{M} \sim (M^1, M^2, ..., M^k)$ – clusters' transition matrices
Output: c – the cluster label for T

1: $c \leftarrow \arg\max_i \prod_{j=1}^{n-1} M^i_{t_j \to t_{j+1}}$
2: **if** $\prod_{j=1}^{n-1} M^c_{t_j \to t_{j+1}} > 0$ **then**
3: **return** c
4: **end if**
5: $c \leftarrow \arg\min_i DSL(T, RT^i)$
6: **if** $DSL(T, RT^c) < 1$ **then**
7: **return** c
8: **end if**
9: **return** -1 ▷ Cluster not found

In the algorithm's first step, we sort the trajectories by their length (number of road segments) in non-increasing order (Line 2). In this way, we start the clusters using the longest trajectories, which are more representative than smaller ones. Next, we process each trajectory sequentially. First, we apply Algorithm 4 with T, **RT**, and **M** as input data to select the most suitable cluster for the current trajectory T (Line 4). If the algorithm does not find any cluster for the trajectory T (Line 5), we start a new cluster composed only of T (Lines 6–8). Initially, the new cluster's RT is the trajectory T, and we use the transitions found in T to calculate the cluster's transition and count matrices (Line 7). Note that Algorithm 4 will return -1 for the first trajectory as there are no clusters initially. If we find a suitable cluster c for the current trajectory T (Line 9), we use T's transitions to update the cluster c transition and count matrices M_c and C_c (Line 10), and add trajectory T to cluster c (Line 11). After processing all trajectories, we calculate each cluster's definitive RT using Algorithm 3 (Lines 14–16).

6.5.5 Prediction

We build more accurate prediction models by clustering the trajectories before calculating the models. However, the assignment of a trajectory to a cluster is less precise if the trajectory is too small. For instance, this can happen if we assign a partial trajectory

to a cluster when the vehicle just started its route (e.g., trajectory with only one road segment), resulting in less accurate predictions. We confirmed this from our experiments' results (Section 6.6). Therefore, we propose a new hybrid algorithm to predict the next steps of a partial trajectory. Besides the prediction models calculated by the clustering algorithm, we extract a Markov process transition matrix $M^{complete}$ from the entire dataset. The transition matrix $M^{complete}$ is applied instead of the cluster's transition matrices when the size of the partial trajectory is less than a given threshold controlled by the parameter *Hybrid Step Change*.

Algorithm 6 performs the i-step predictions of a given partial trajectory $T^{1:P} = (t_1, t_2, ..., t_P)$ sequentially. The algorithm receives as input the partial trajectory $T^{1:P}$, an integer $i > 0$, the parameter *Hybrid Step Change*, and the trained models (i.e., the transition matrix $M^{complete}$ and the clusters' RT and transition matrices M).

For each step prediction, we apply the complete transition matrix $M^{complete}$ until the size of the predicted trajectory is greater than or equal to *hsc*. If the size of predicted trajectory is greater than or equal to *hsc* (Line 3), we use Algorithm 4 to select the most likely cluster c for the trajectory T_{pred} (Line 5), and fallback to the first cluster if Algorithm 4 returns -1 (Lines 6–8).

After deciding which transition matrix $M^{current}$ to use (Lines 3–12), we predict the next step as follows. First, we select the most likely road segment using the transition matrix $M^{current}$ and the last road segment of the current predicted trajectory T_{pred} (Line 14). If the transition probability from the last road segment to the next road segment (predicted road segment) is greater than zero (Line 15), we append the next road segment to the predicted trajectory (Line 16). Otherwise, we finish the prediction process with less than i-steps predicted.

6.6 Performance Evaluation

This section evaluates the effectiveness and efficiency of the proposed framework and compares it to some related work. We replicated each experiment five times and plotted the graphs with a confidence interval of 95%. Also, we implemented the proposed framework and related work using the C++ programming language. We performed all experiments on the same platform: a server with 128 GB of memory RAM, processor Intel(R) Core(TM) i9-9900X CPU @ 3.50 GHz, and OS Ubuntu 18.04.4 LTS. We assess the clustering algorithm in Section 6.6.4 and the prediction framework in Section 6.6.5.

6.6.1 Dataset

We use the *Rio Center Dataset for Data Mining* (Chapter 4) in the experiments. We randomly select 70% of the trajectories to comprise the training dataset and generate the prediction dataset using the remaining trajectories. From each prediction trajectory T_i from the training dataset, we generate $|T_i| - 1$ partial trajectories $T_i^{1:1}$, $T_i^{1:2}$, ..., $T_i^{1:|T_i|-1}$ with the prefixes of T_i. Finally, from each partial trajectory $T_i^{1:j}$, we perform $|T_i| - j$ i-step trajectory predictions ($1 \leq i \leq |T_i| - j$).

6.6.2 Evaluation Metrics

For a given trajectory T, a partial trajectory $T^{1:P}$ of T ($1 \leq P < |T|$), and a predicted trajectory T_{pred} ($P < |T_{pred}|$), we defined the following metrics to assess the accuracy of the prediction methods.

Prediction Accuracy (PA)

The prediction accuracy is the ratio of correctly predicted locations to the maximum of the size differences between the predicted and actual trajectories from the partial trajectory. Formally, $PA(T, T^{1:P}, T_{pred})$ is defined as

$$PA(T, P, T_{pred}) = \frac{1}{n-P} \times \sum_{i=P+1}^{n} H(T^i, T_{pred}^i), \tag{6.2}$$

where

$$n = \max\{|T|, |T_{pred}|\}$$

$$H(T^i, T_{pred}^i) = \begin{cases} 0, & \text{if } |T| < i \text{ or } |T_{pred}| < i \text{ or } T^i \neq T_{pred}^i \\ 1, & \text{otherwise} \end{cases}$$

i-step Prediction Accuracy (iPA)

The i-step prediction accuracy is defined for a given i and trajectories T, $T^{1:P}$, and T_{pred} if and only if $0 < i \leq |T| - P$ and $|T_{pred}| \leq |T|$. Formally, the i-step prediction accuracy $iPA(i, T, T^{1:P}, T_{pred})$ is defined as

$$iPA(i, T, P, T_{pred}) = PA(T^{1:(P+i)}, T^{1:P}, T_{pred}^{1:(P+i)}) \tag{6.3}$$

where $T_{pred}^{1:i}$ is the i-sized prefix of T_{pred}.

6.6.3 Comparison Methods

We compare the proposed framework to the previous version of our framework [45], which we call *Framework (v0)*, and to two other approaches [161], described below.

Traj-clusiVAT-based-TP*

We implement an adapted version of the Traj-clusiVAT-based-TP framework [161] to work with directed road networks, which we call *Traj-clusiVAT-based-TP**. We introduced some improvements to overcome most of the limitations of the Traj-clusiVAT-based-TP framework discussed in Section 6.2. For instance, we fixed the trajDTW algorithm by modifying the definition of the window parameter w to $w = \max\{\frac{1}{2} \times \min\{|T_1|, |T_2|\}, ||T_1| - |T_2||\}$. We also defined the road network distance between arcs e_1 and e_2 of a directed road network as

$$\text{dist}(e_1, e_2) = \text{length}(e_1) + \text{length}(e_2) + \min\{\text{dist}(e1.to, e2.from), \text{dist}(e2.to, e1.from)\} \tag{6.4}$$

and removed the third step of the Traj-clusiVAT-based-TP framework as it was only necessary because of the undirected road network.

First-order Markov chain model

Grouping similar trajectories can help construct better prediction models because of the diversity of trajectories in a dataset. For instance, we can construct better models with similar trajectories in a small region rather than with longer trajectories across this small region. However, if the training dataset is small, the clustering step can have a diverse impact. Therefore, models constructed from the entire dataset can present better prediction accuracy for some scenarios. To evaluate this scenario, we implemented a single first-order Markov chain model from the entire training dataset using the approach discussed in Section 6.5.1.

6.6.4 Clustering Results

Some of the advantages of the proposed clustering algorithm are the following. Unlike the related work, the proposed clustering algorithm automatically detects the appropriate number of clusters. The algorithms used in *Framework (v0)* and *Traj-clusiVAT-based-TP**, on the other hand, rely on input parameters to control the number of clusters. Another disadvantage of the related work algorithms is that they randomly assign to

some cluster trajectories that should not be part of any cluster (e.g., outliers). These wrong assignments negatively affect the prediction models extracted from the clusters. On the other hand, our algorithm detects these outliers and creates separate clusters for them.

Figure 6.1 shows, in non-increasing order, the number of trajectories in the clusters found by each algorithm. The proposed algorithm results in more clusters than the related work. However, most of the clusters are composed of less than ten trajectories, as shown in Figure 6.1a. These small clusters contain the outliers found in the dataset. On the other hand, the other solutions assign the outliers to any of the existing clusters, resulting in clusters of similar sizes. Figure 6.1b shows the cumulative sum of cluster size. We can see that the first few clusters found by our algorithm contain almost the entire dataset. This result is consistent with the behavior observed on a city's road network, in which most vehicles move between crowded regions and follow similar paths.

Figures 6.2, 6.3, and 6.4 display the representative trajectories RT from each of the twenty largest clusters found by each solution. The new hybrid framework and its previous version *Framework (v0)* use the same algorithm to extract a cluster's representative trajectory. However, we can see that the representative trajectories from the new proposal's largest clusters cover more regions of the road network, as expected because these clusters represent a greater number of trajectories in the dataset. The *Traj-clusiVAT-based-TP** could not calculate the RT of some clusters, as discussed in Section 6.2.

Finally, Figures 6.5 and 6.6 show a sample of trajectories from the twenty largest clusters found by the New Hybrid Framework and its previous version, respectively. The figure shows the top ten trajectories from each of the twenty clusters, totaling 200 trajectories. We can see that the proposed algorithm clusters have less overlap and cover more road segments from this sample.

6.6.5 Prediction Results

Prediction Accuracy

Figure 6.7 presents the average accuracy of the i-step trajectory predictions for $1 \leq i \leq 50$. As expected, the prediction accuracy for all solutions decreases as the number of predicted steps increases. The *first-order Markov Chain*, which does not apply clustering before creating the prediction models, shows high prediction accuracy for few prediction steps ($i < 20$) and the worst prediction accuracy when trying to predict many steps ($i \geq 30$). The solutions that employ prediction models from clustered data (i.e., *Framework*

(v0) and *Traj-clusiVAT-based-TP**), on the other side, present the worst results for predicting few steps but overcome the *first-order Markov Chain* for high values of i. This result occurs because the precision of the assignment of trajectories to a cluster depends on the trajectory size. Therefore, the accuracy of prediction models extracted from clustered data is low if the partial trajectory is small.

The proposed framework outperforms all solutions for all i values. For small i values, the proposed framework's accuracy is at least as good as that of *first-order Markov Chain* since we employ the prediction model extracted from all trajectories if the partial trajectory is short (smaller than the *hybrid step change* parameter). As the predicted trajectory size increases, we begin applying the models extracted from clustered data, resulting in high prediction accuracy for larger i values. Finally, the new framework outperforms its previous version because of the hybrid prediction approach discussed above and the new hierarchical clustering algorithm results in better clusters.

Processing Time

Figure 6.8a shows the average processing time of the i-step trajectory predictions. The *Traj-clusiVAT-based-TP** presents the highest processing time since it executes the hybrid-NPR algorithm once for each step predicted, resulting in increasing computational complexity for larger i values. Figure 6.8b displays the prediction processing time excluding the *Traj-clusiVAT-based-TP** framework. The proposed framework and its previous version show a higher processing time when compared to the *first-order Markov Chain* mainly because the *first-order Markov Chain* does not have a clustering step. Finally, the new framework's processing time is better than that of its previous version since the new version applies the clustering step only when i is greater than the *hybrid step change*.

Figure 6.9 shows the average processing time of the training phase. The *first-order Markov Chain* only extracts the transition matrix from the dataset and presents the fastest training phase. As expected, the *Traj-clusiVAT-based-TP** presents the slowest training phase, primarily because of the trajDTW distance measure. The trajDTW uses the Dijkstra algorithm to compute the road network distance between arcs. In the results showed in Figure 6.9, we precomputed the distance matrix and ignored this time overhead. Even so, the overhead of performing lookups on such a large distance matrix resulted in this performance. As for the proposed solution, the new framework presents a shorter processing time than its previous version since the new clustering algorithm has a better time complexity than that of the algorithm used in our framework's previous

version.

Impact of Parameters

Lastly, we assess the impact of the parameter *hybrid step change* on the framework's prediction accuracy. We performed experiments using *hybrid step change* values from 1 to 50. Figure 6.10 shows the results. We can see that the *hybrid step change* does not influence the prediction accuracy when performing predictions of few steps. However, as we increase the number of predicted steps, the value of *hybrid step change* affects the results. Overall, values of the *hybrid step change* between 9 and 14 provide the best results, showing that the assignment of a trajectory to a cluster is more precise when the trajectory size is at least 9. Moreover, the prediction accuracy decreases for greater values of *hybrid step change*. That occurs because, with these values, the framework uses the transition matrix extracted from the entire training dataset when the models from the clustered data are a better option (i.e., for longer trajectories).

Parameter Tuning

As the *Rio Center Dataset for Data Mining* contains a large number of trajectories (526,467 trajectories recorded during four months), and its region corresponds to a full-scale metropolitan area, these experimental results can be generalized to represent the framework's behavior when used to predict trajectories from different parts of the world. Therefore, in line with the results of the experiments, we suggest a value of ten for the *hybrid step change* parameter, which should produce satisfactory results when applied in most scenarios.

6.7 Chapter Remarks

In this chapter, we proposed a novel cluster-based trajectory prediction framework. This novel solution trains prediction models from historical datasets to perform long-term predictions of road-network constrained trajectories. We highlighted the scalability of the framework, which, due to the cluster-based technique, allows the framework to train large databases efficiently.

We compared the framework to some related work through experiments using a new large-scale real-world trajectory dataset. The framework outperformed the related work in terms of i-step prediction accuracy. Moreover, the framework presented the best time

complexity in both the training and prediction phases among the solutions based on clustering.

As future work, we plan to investigate other clustering algorithms and the impact of other similarity measures on the framework's performance. Besides that, we plan to extend the framework to perform trajectory predictions without knowledge regarding the number of steps to predict, which means to detect the end of the predicted trajectory.

Algorithm 5 Clustering road-network constrained trajectories

Input: $\Gamma \sim \{T_1, T_2, ..., T_n\}$ – trajectory dataset

Output:

k – the number of clusters

$\zeta \sim \{\Gamma_1, \Gamma_2, ..., \Gamma_k\}$ – the set of clusters

$\mathbf{RT} \sim \{RT_1, RT_2, ..., RT_k\}$ – clusters' representative trajectories

$\mathbf{M} \sim \{M_1, M_2, ..., M_k\}$ – clusters' transition matrices

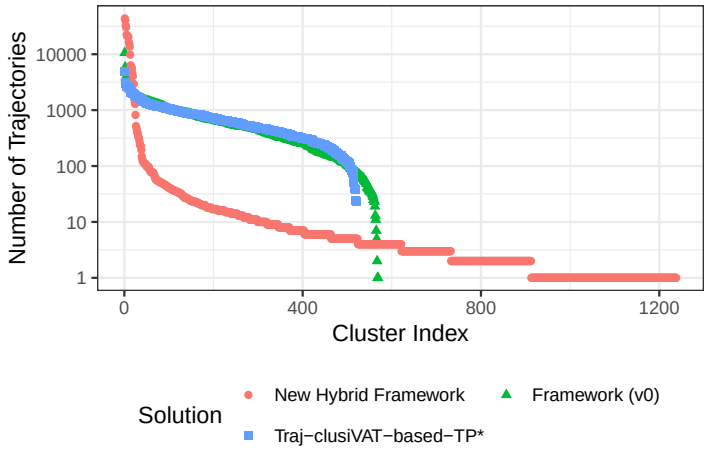
$\mathbf{C} \sim \{C_1, C_2, ..., C_k\}$ – clusters' count transition matrices

1: $k \leftarrow 0$, $\zeta \leftarrow \emptyset$, $\mathbf{RT} \leftarrow \emptyset$, $\mathbf{M} \leftarrow \emptyset$, $\mathbf{C} \leftarrow \emptyset$
2: Sort Γ in non-increasing order by the number of road segments
3: **for each** $T_i = (t_1, t_2, ..., t_n) \in \Gamma$ **do**
4: $c \leftarrow$ Execute Algorithm 4 with T, **RT**, and **M** as input data to select the most suitable cluster for trajectory T
5: **if** $c \neq -1$ **then** ▷ Cluster not found
6: $k \leftarrow k + 1$
7: M, $C \leftarrow$ compute cluster's transition and count matrices from T_i
8: $\Gamma_k \leftarrow \{T_i\}$, $RT_k \leftarrow T_i$, $M_k \leftarrow M$, $C_k \leftarrow C$
9: **else** ▷ Cluster found
10: Update M_c and C_c with T_i's road transitions
11: Append T_i to Γ_c
12: **end if**
13: **end for**
14: **for** $i \leftarrow 1$ to k **do**
15: $RT_i \leftarrow$ compute cluster's representative trajectory using Algorithm 3
16: **end for**
17: **return** k, ζ, **RT**, **M** and **C**

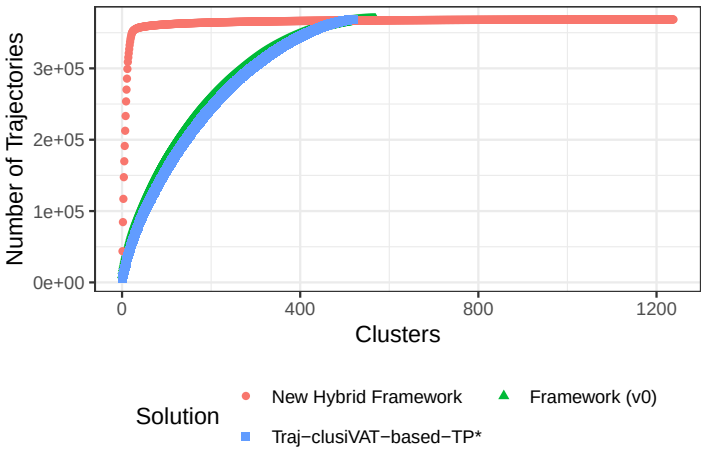
Algorithm 6 i-step Trajectory Prediction

Input: $T^{1:P} \sim (t_1, t_2, ..., t_P)$ – a partial trajectory
i – number of steps to predict
hsc – hybrid step change
$M^{complete}$ – complete transition matrix
$\mathbf{RT} \sim (RT^1, RT^2, ..., RT^k)$ – set of representative trajectories
$\mathbf{M} \sim (M^1, M^2, ..., M^k)$ – cluster's transition matrix
Output: T_{pred} – the predicted trajectory

$T_{pred} \leftarrow T^{1:P}$, lastRoad $\leftarrow t_P$, $c \leftarrow -1$
for each $j \in [0 \,..\, i-1]$ **do**
  if $|T_{pred}| \geq hsc$ **then**
    if $c = -1$ **then** ▷ Trajectory not assigned to a cluster yet
      $c \leftarrow$ Assign trajectory T_{pred} to a cluster using Algorithm 4
      if $c = -1$ **then**
        $c \leftarrow 1$ ▷ Defaults to the first cluster
      end if
    end if
    $M^{current} \leftarrow M^c$ ▷ Use the cluster's transition matrix
  else
    $M^{current} \leftarrow M^{complete}$ ▷ Use complete transition matrix
  end if
  nextRoad $\leftarrow \arg\max_r M^{current}_{\text{lastRoad}\rightarrow r}$
  if $M^{current}_{\text{lastRoad}\rightarrow r} > 0$ **then**
    $T_{pred} \leftarrow T_{pred} \cup$ nextRoad
    lastRoad $\leftarrow$ nextRoad
  else
    break
  end if
end for
return T_{pred}



(a) Clusters size in non-increasing order.



(b) Cumulative sum of the clusters size in non-increasing order.

Figure 6.1: Clusters size.

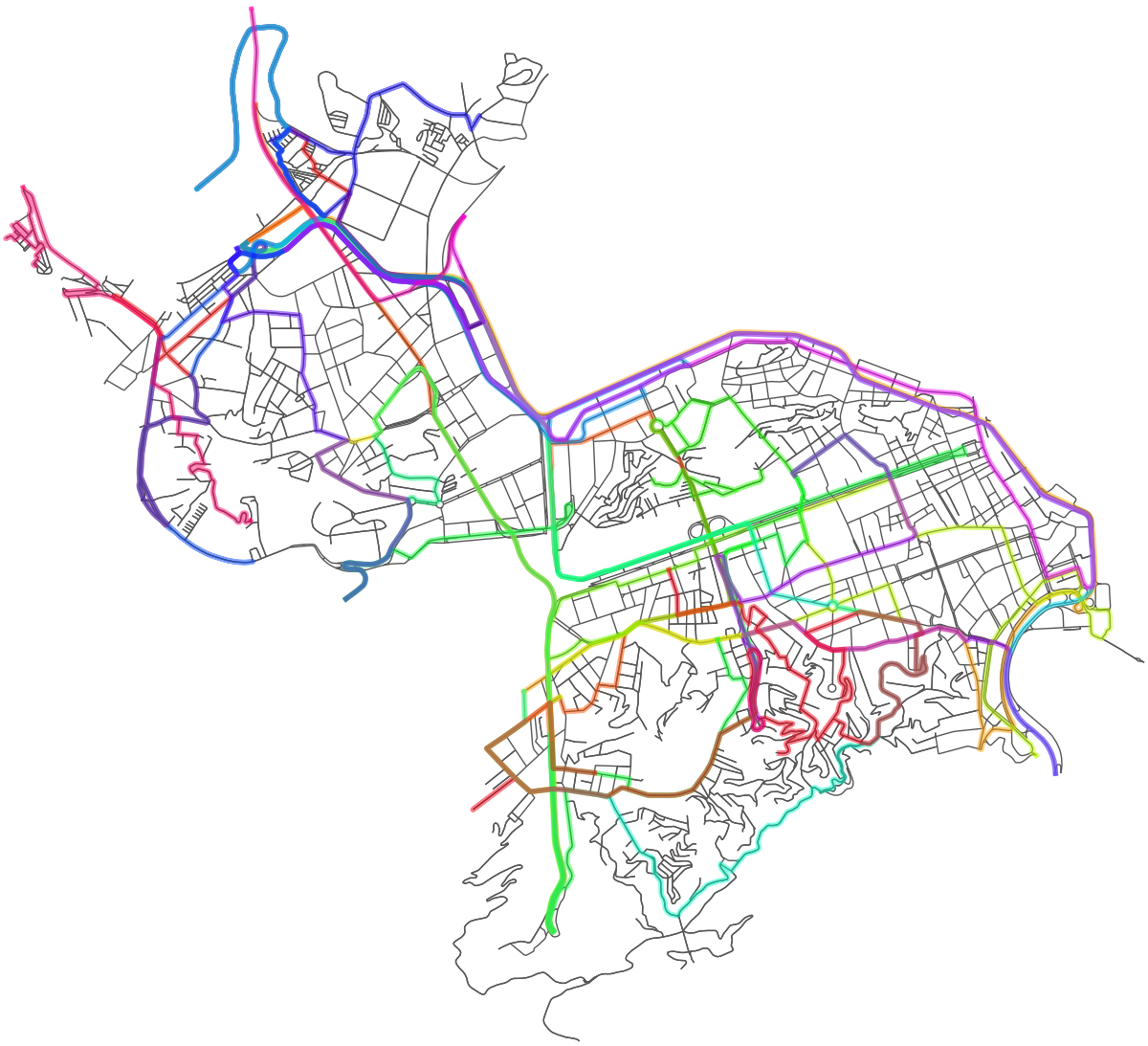


Figure 6.2: Representative trajectories from the twenty largest clusters found by the New Hybrid Framework.

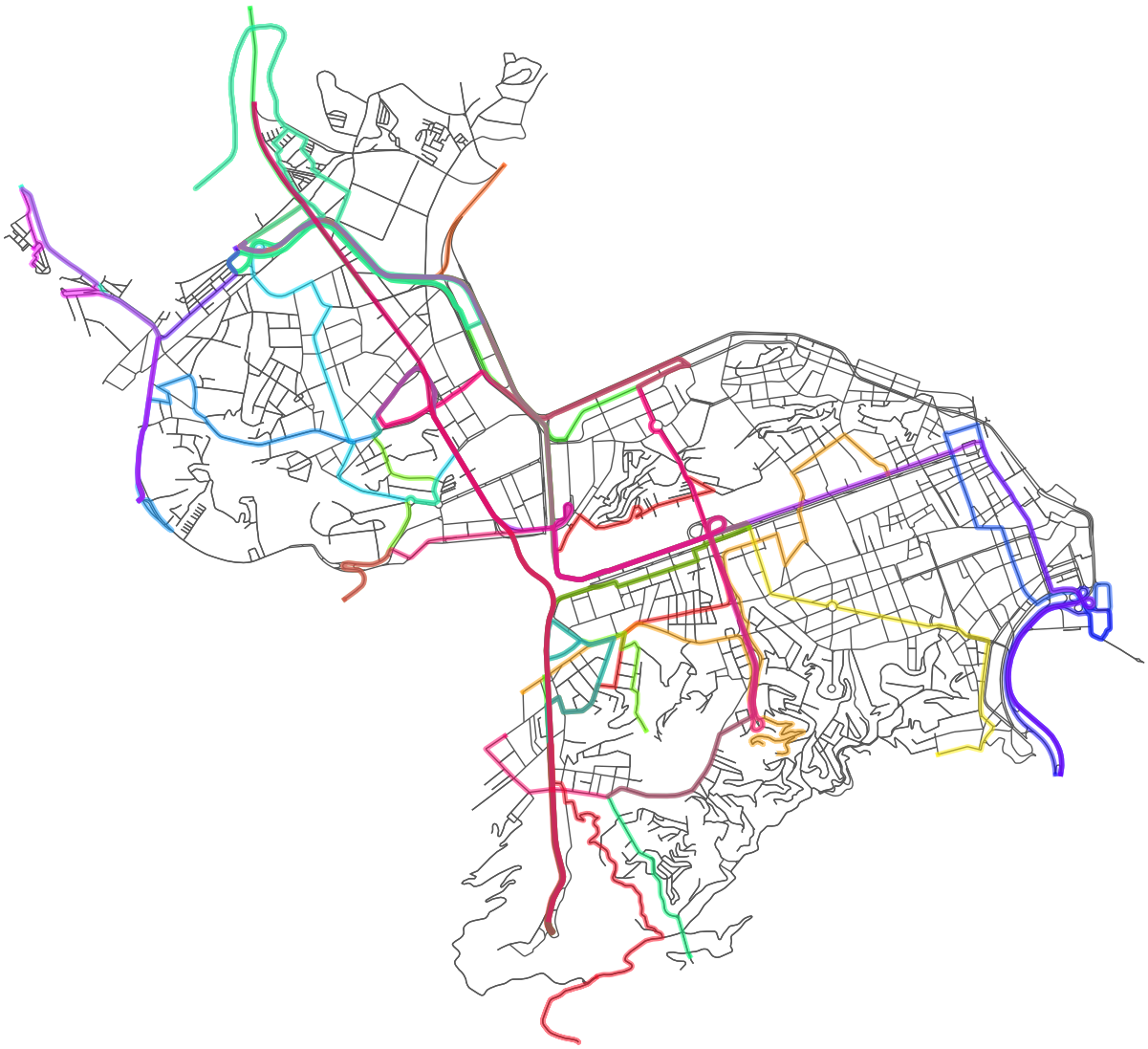


Figure 6.3: Representative trajectories from the twenty largest clusters found by the Framework (v0).

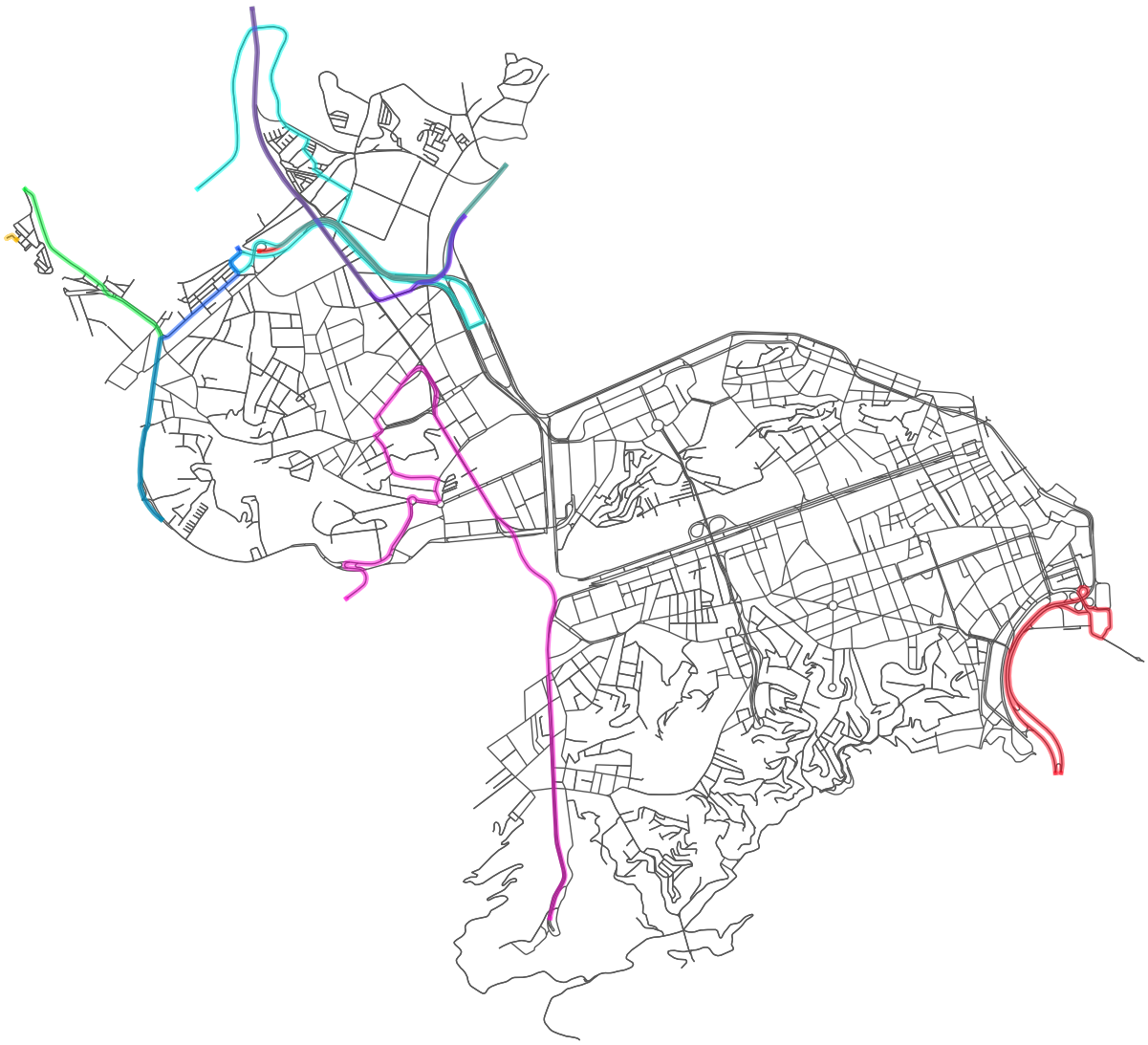


Figure 6.4: Representative trajectories from the twenty largest clusters found by the Traj-clusiVAT-based-TP* framework.

Figure 6.5: Ten longest trajectories from each of the twenty largest clusters found by the New Hybrid Framework..

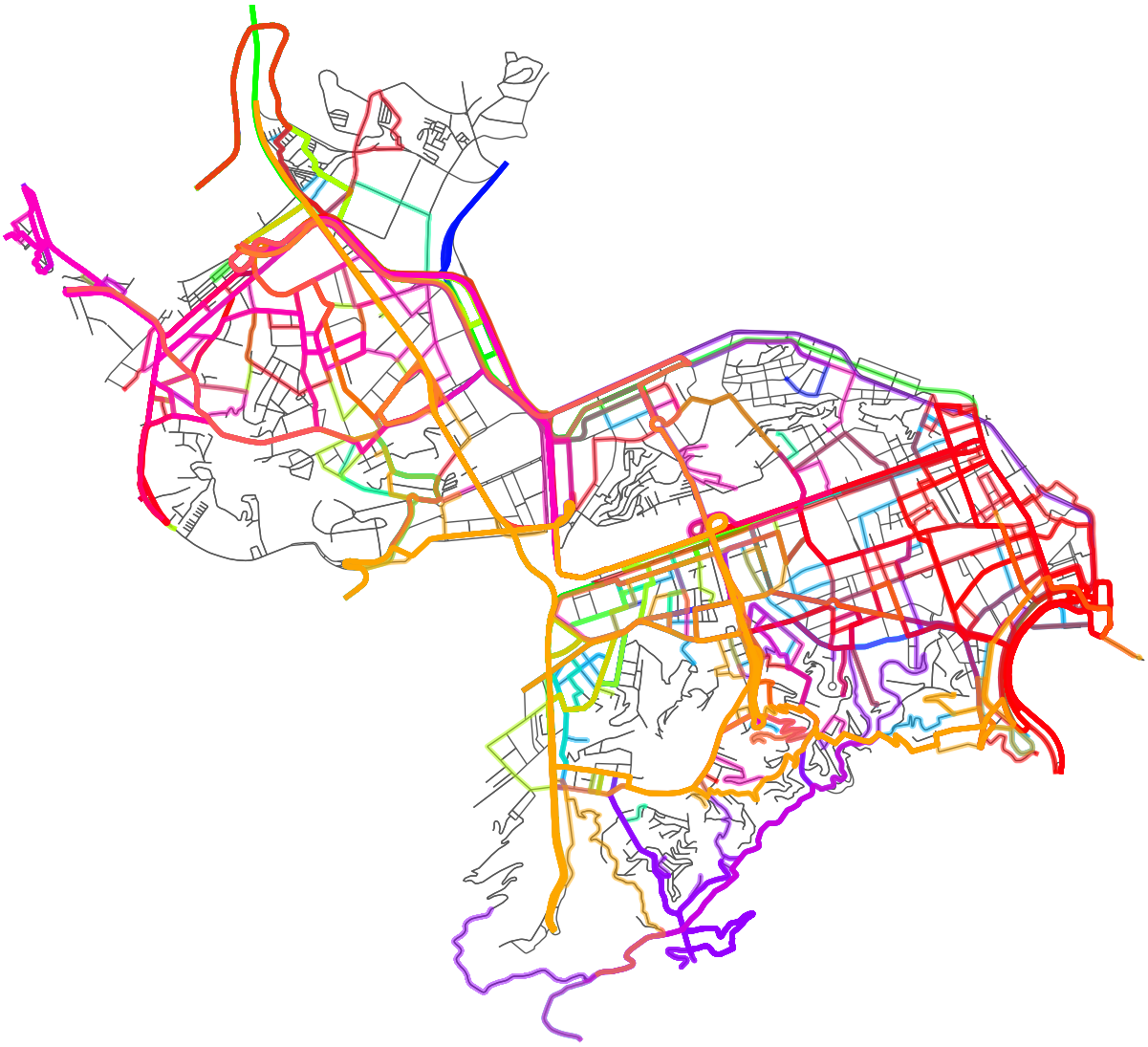


Figure 6.6: Ten longest trajectories from each of the twenty largest clusters found by the Framework (v0).

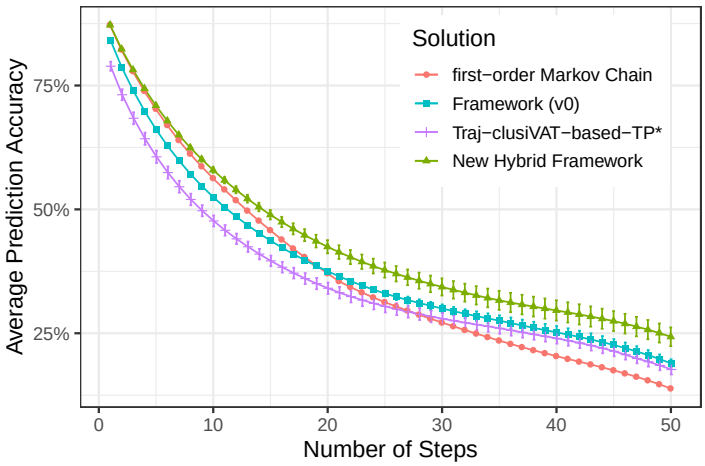
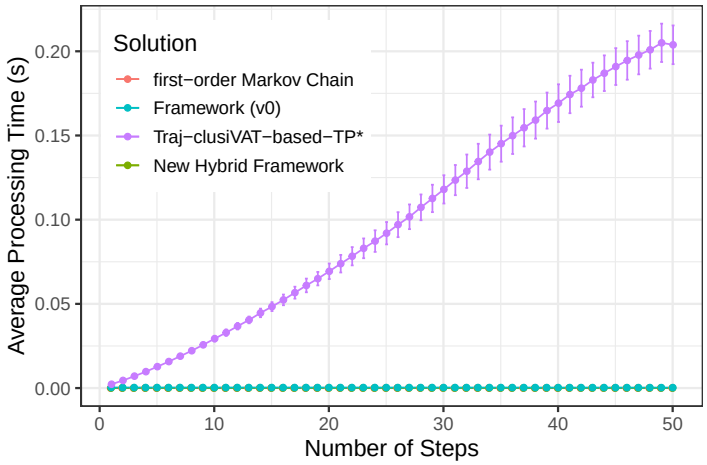
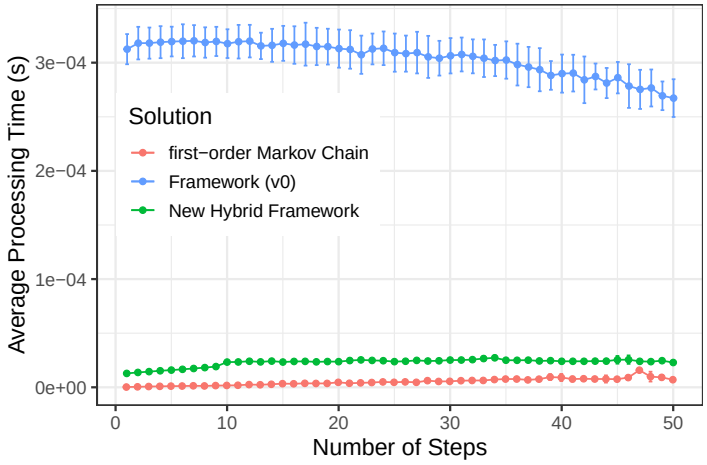


Figure 6.7: Average accuracy of the i-step trajectory predictions.



(a) All solutions.



(b) Results excluding baseline.

Figure 6.8: Average processing time of the i-step trajectory predictions.

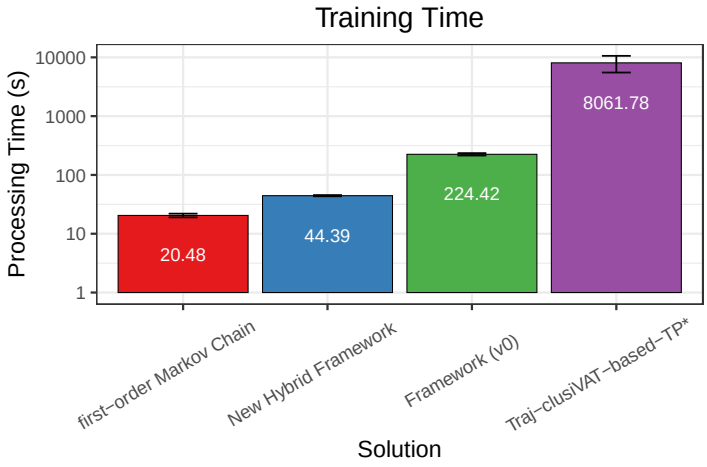
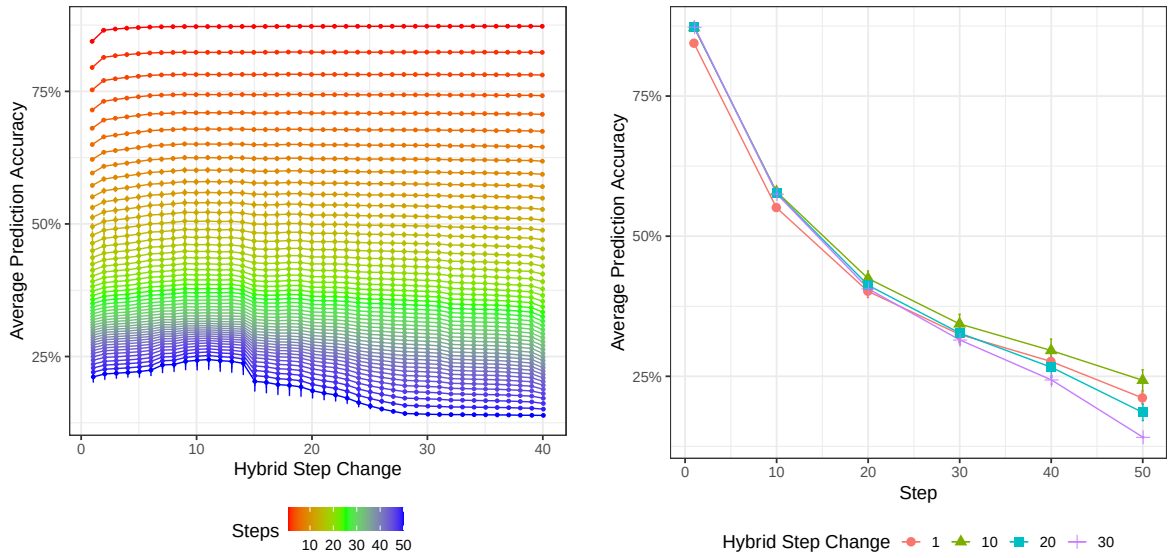


Figure 6.9: Average processing time of the training phase.



(a) *hybrid step change* vs number of predicted steps.

(b) Number of predicted steps vs *hybrid step change*.

Figure 6.10: Parameter sensitivity of *hybrid step change*.

Chapter 7

A Distributed and Low-Overhead Traffic Congestion Control Protocol for Vehicular Ad-Hoc Networks

The development of new technologies and transportation modes is a crucial component of improving urban mobility. For instance, researchers have shown that vehicular networks are a promising technology to monitor and reduce traffic jams. Nevertheless, most of these solutions rely on costly external infrastructures, such as roadside units or cellular networks. This chapter proposes DisTraC, a protocol for vehicular ad-hoc networks. DisTraC is a traffic congestion control protocol of low communication overhead that aims to estimate and reduce traffic congestion levels by employing vehicle-to-vehicle (V2V) communication. The protocol is independent of external infrastructures as it uses only V2V communication. Simulation results show that DisTraC outperforms other solutions published in the literature in terms of communication overhead and capability to reduce traffic congestion.

7.1 Introduction

Urban mobility has been a significant challenge in large cities for a long time as, for example, traffic jams directly affect the economy, the environment, and public health [102]. Moreover, the mobility aspects of society are changing rapidly, and the concentration of population in urban areas will continue to increase over the coming decades [77]. Thereby, it is necessary to develop new technologies and transportation modes to

move people and other products like goods and animals in a smart way, leading to smart transportation solutions [21].

Currently, the employment of wireless communication in the vehicular environment to detect and reduce traffic jams is of paramount importance. VANETs (Vehicular Ad-hoc Networks) are communication systems that allow the exchange of data between vehicles (V2V communication) and between vehicles and roadside units (V2I communication) through a Dedicated Short Range Communication (DSRC) protocol. However, the vehicular environment has some characteristics that can pose challenges for communication in VANETs. For instance, the network topology is highly dynamic, the number of vehicles in congested regions can lead to the broadcast storm problem [195], and areas with few vehicles make the network sparse, which hinders data transmission in these regions [213]. Besides that, proposals for vehicular networks should have low communication overhead to not adversely impact the applications that are sensitive to the transmission delay (e.g., traffic safety applications) [195]. Vehicles can also take advantage of pervasive cellular networks (e.g., 4G LTE and 5G) or multiple roadside units (RSU) to obtain a broad vision of the traffic conditions on a region. However, transmitting a large amount of data through cellular networks or deploying RSUs to cover urban regions fully incurs in prohibitive costs [41, 67]. On the other hand, applications that rely solely on V2V communication have low deployment and maintenance costs.

In this Chapter, we focus on traffic monitoring and vehicular congestion, aiming at reducing vehicle travel times through the use of V2V communication. To this end, we propose DisTraC, a **Dis**tributed and low-overhead protocol for **Tra**ffic **C**ongestion control that is independent of external infrastructures such as roadside units or cellular network towers. With DisTraC, each vehicle individually estimates the local congestion level by monitoring its speed. Then, we present two low overhead communication mechanisms to enable vehicles to validate their congestion level estimations and to get information about traffic jams of other regions. With the proposed data fusion technique, vehicles can store their congestion level estimations and the information received from others in a meaningful way. Finally, the aggregated data is employed independently by each vehicle to compute the best route to their destination.

The DisTraC protocol demonstrates how we can apply vehicular networks to improve urban mobility. However, we did not design it for any specific scenario. For instance, drivers can use it to receive route suggestions with lower travel times. On the other hand, in the case of autonomous vehicles, they can automatically change their routes using the recommendations from the protocol. The only restriction is that vehicles must

have vehicular network adapters in their onboard units.

7.2 Related Work

The advancement of vehicular communication technologies enables the emergence of novel solutions for the classic problem of urban mobility in large cities. For instance, centralized systems based on pervasive cellular data services (e.g., 4G LTE and 5G) or V2I communication can provide a comprehensive vision of the traffic conditions of a region. However, transmitting a massive amount of data through cellular networks or deploying RSUs to cover urban regions fully incurs in prohibitive costs [41, 67], while applications that rely solely on V2V communication have low deployment and maintenance costs. Therefore, in this section, we extensively review and classify the proposals that aim at improving urban traffic using vehicular networks, highlighting the used communication technology and the proposed congestion control mechanisms.

SOTIS [194] is a V2V-based traffic management system in which each vehicle performs a traffic situation analysis by recurrently receiving data packets with detailed information from other vehicles. The vehicle sends the result of the traffic situation analysis via a one-hop broadcast to all neighboring vehicles. Finally, a store-carry-forward technique is used to share the information with vehicles in other regions.

CoTEC [18] is a cooperative technique based on V2V communications to detect road traffic congestion without the need to deploy infrastructure sensors. CoTEC proposes a traffic congestion quantification system based on the fuzzy theory that takes the traffic density and vehicle speed as input parameters and provides the corresponding traffic congestion level or traffic jam intensity as an output parameter. The individual congestion information is shared among the vehicles through multi-hop communication to validate the information cooperatively and reduce the communication overhead. In this case, vehicles share data only when a traffic congestion situation is locally detected. Araújo et al. [10] presented an extension of CoTEC, called CARTIM. The main contribution of CARTIM is a heuristic that uses the congestion information to allow route modification, thus preventing vehicles from reaching a congested area. Furthermore, if there is any RSU in the area, V2I communication is used to disseminate the information to other regions.

Arbabi and Weigle [11] proposed DTMon, one of the most pioneering systems for traffic management using V2X communication. Despite the drawback of being RSU-based, DTMon is not able to measure how congested the roads are, as it only recognizes

non-congested traffic conditions.

VAM [75] is another V2V-based traffic management solution that enables vehicles to exchange data about their routes so that each vehicle can select less congested routes. With VAM, each vehicle determines an initial route and a collection of optional routes. Afterwards, each vehicle can select one of the optional routes to replace the current one based on the number of neighbors with routes similar to the current route. One of the shortcomings of this solution is that it generates a significant communication overhead.

DTraMS [68] is a system that monitors and disseminates traffic conditions data using a decentralized infrastructure and V2X communication. As other proposals that use V2I communication, DTraMS assumes that RSUs are covering all regions, which can make the solution unfeasible.

ECODE [207] is a protocol based on V2X communication to detect road segments that are suffering high traffic congestion. It uses multi-hop communication and geocast principles to gather and analyze vehicles' primary data per road segment and assumes that there is an RSU at each road intersection. Other studies also assume that there is an RSU at each road intersection to employ V2X communication [144, 5]. Noori and Valkama [144] used the gathered data and the A* search algorithm to calculate and search the best routes for the vehicles. Aissaoui et al. [5] proposed a mechanism that uses clusters to send data to the RSU in a less costly way.

Gramaglia et al. [71] proposed ABEONA, a solution based on V2V communication to estimate the flow and density of vehicles on the roads. Besides that, the solution can predict the transition from the free-flow to the synchronized traffic state. The authors do not address how predicted events are disseminated neither how to use this information to reduce traffic congestion.

Milojevic and Rakocevic [135] presented a V2V-based technique that enables each vehicle to identify and measure traffic congestion, but without a method to reduce the travel times. Each vehicle estimates the local congestion level by measuring the time during which its speed is lower or higher than a given threshold. Furthermore, all vehicles validate their local estimations with the data received from their neighbors and transmit the local estimations to other regions through a multi-hop broadcast. In order to reduce the communication overhead, a broadcast only occurs whenever there is an increase in the local congestion level. The authors do not assess the use of the collected data to reduce traffic congestion.

E-VeT [149] is an economy-based/reward-penalty system based on V2X communication to manage the vehicular traffic in road networks efficiently. The system is composed

of a scheme called R^2A and a route allocation algorithm. R^2A rewards vehicles for following system-assigned longer-time paths, and charges a fee for following system-assigned shorter-time paths. The route allocation algorithm gives lesser-time paths as a preference for vehicles that have earned higher revenue based on the R^2A scheme. The primary objective is to reduce the average travel time and fuel consumption.

Bellavista et al. [20] proposed a set of V2X-based protocols to determine the traffic characteristics in the proximity of intersections, intending to offer monitoring data to optimize traffic light management. They demonstrated through simulations that the protocols can achieve reasonable estimations of vehicular traffic even with a limited penetration rate of the vehicular network.

Garip et al. [67] proposed a distributed congestion avoidance mechanism based only on V2V communication. Vehicles change their routes based on checkpoints inserted between their initial and destination positions. When the vehicle approximates the subsequent checkpoint, it uses V2V communication to get the average speed of vehicles in the following roads and uses this information to select the best route from the current checkpoint to the next one. Although the proposed method is not able to produce traffic congestion information, it is one of the few solutions found in the literature to effectively decrease the average travel time employing only V2V communication.

Elbery et al. [59] presented a system that employs V2X communication to reduce the average vehicle fuel consumption and travel time. When a vehicle crosses a road link, it transmits its fuel consumption on the road segment to a Traffic Management Center (TMC) that, in turn, assign routes to the vehicles based on this information. One drawback of this approach is its processing overhead as all computations are centralized.

Other solutions proposed centralized systems that suffer from the problems inherent in this type of system [193, 93, 29]. Wang et al. [193] proposed a real-time path-planning algorithm based on V2X communication and heterogeneous networks where a centralized vehicle-traffic server performs all traffic planning computation. Jeong et al. [93] proposed a system where a central server gathers trajectories information from all vehicles and estimates road segment congestion for global traffic optimization. In that system, the travel delay estimation of a road segment depends on prior knowledge about the mean and variance of the link travel delay. Cárdenas-Benítez et al. [29] used V2X communication and a set of already known protocols to build a centralized system to monitor, detect, and reduce traffic congestion.

Martuscelli et al. [131] proposed and analyzed four V2V protocols to deliver traffic information updates concerning routes of interest. Although the protocols are simple

and have a limited scope, they could be used by other solutions to provide a traffic management service. The more sophisticated proposal is an implementation of a known technique to reduce overhead in ad hoc networks. Furthermore, the protocols do not address the situation where the network is disconnected.

DIVERT [150] is a privacy-aware system based on V2V communication and the cellular network to detect and reduce traffic congestion. The solution estimates the road density (number of vehicles on the road) using V2V communication. If the density is above a given threshold, it is sent with a given probability to a central server through a cellular network. The estimation algorithm is not suitable for every scenario (e.g., long roads). The central server uses the received density information to create a smoothed density information of the road. If the server detects signs of congestion, it will send messages to vehicles that reported most recently, so these vehicles will send the information to others on the road. After that, vehicles will re-route to reduce travel times.

Guo et al. [74] presented a method that uses V2X communication to estimate and share the vehicles traveling time and a real-time path planning algorithm to avoid traffic congestion. They divide the traveling time estimation into three components for each road segment: (i) the time to cross the straight road segment, (ii) the waiting time for the traffic light, and (iii) the time to bypass the road intersection. The authors demonstrated through simulations that the proposed algorithm outperforms the static path planning algorithm.

Table 7.1 summarizes the related work. The *VANET* column refers to the vehicular network communication the work uses, where the options are V2V (vehicle-to-vehicle), V2I (vehicle-to-infrastructure) and V2X (vehicle-to-vehicle and vehicle-to-infrastructure). The *Cel.* column specifies if the solution uses any of the cellular network technologies, like LTE (Long-Term Evolution). Table 7.1 also defines if the related work proposes a mechanism to quantify the road congestion level (*Quan.* column), a method to disseminate the traffic information to vehicles in other regions (*Dis.* column) and, finally, a re-routing technique to reduce traffic jams (*Rou.* column).

Most of the reviewed solutions detect traffic congestion conditions but do not offer a method to reduce traffic jams. Furthermore, the high costs to install and maintain RSUs represent a disadvantage for solutions that depend on these infrastructures. In addition to relying only on V2V communication, our proposed protocol can significantly reduce the communication overhead due to the cluster-based mechanism used to broadcast traffic data. Besides that, experiments revealed that DisTraC is more effective than solutions found in the literature in reducing average travel time.

7.3 DisTraC

7.3.1 Overview

The main goal of DisTraC is to reduce the vehicle average travel time by employing V2V communication without leading to high communication overhead. We divide its operations into four major components, as illustrated in Figure 7.1.

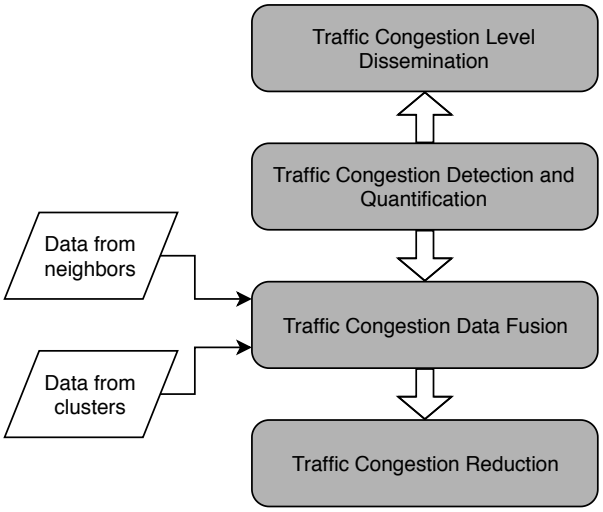


Figure 7.1: DisTraC components overview.

In the first component, each vehicle has a single view of the traffic in its current road segment. This view, called *local traffic congestion level estimation*, relies solely on sensors from the vehicle itself and, therefore, can be different from the view of other vehicles on the same road.

In the second component, the vehicles share their local traffic congestion level estimations with others. It is necessary to validate the individual estimations cooperatively and share information with vehicles that are on other roads.

As the vehicles keep estimating the traffic congestion levels and keep receiving new information from other vehicles, it is necessary to combine those data in a meaningful way. That is the purpose of the third component, in which each vehicle combines the received data and generates a traffic congestion map of the area.

Finally, in the last component, the traffic congestion map is used to compute the best route for the vehicle from its current position to its destination. As each vehicle performs these steps continuously and individually, the traffic tends to be balanced on different roads, reducing average travel time.

7.3.2 Traffic Congestion Detection and Quantification

To define traffic congestion is the first step to build a system to measure urban traffic. One of the most common definitions is found in [123]. The authors describe traffic congestion as the travel time or delay exceeding that usually incurred under light or free-flow travel conditions. On top of that, Rao and Rao [160] presented different metrics to identify and measure urban traffic congestion. They concluded that congestion is a function of a reduction in vehicle speeds and that establishing a threshold directly related to travel speeds to each road is most appropriate when measuring congestion. It is worth mentioning that in this chapter, we use the expression *road* as a road segment delimited by two crossroads.

We propose to estimate traffic congestion as follows. Each vehicle uses its instantaneous speed (v_{inst}) to measure the local traffic Congestion Level (γ) of the current road. γ admits real values between 1 and $\gamma_{\max}$. The initial value of γ is 1, which represents free flow, and values higher than one represent traffic congestion situations. We assume that each road has a reference constant $\mathrm{V_{threshold}}$ that represents the speed threshold used to quantify its congestion level. Thus, the vehicle measures how long its v_{inst} remains above or below $\mathrm{V_{threshold}}$ to calculate the local congestion level γ.

Each vehicle executes Algorithm 7 once every 100 ms to update its local congestion level estimation γ. The *MIN* function is responsible for limiting the maximum value of the new congestion level to $\gamma_{\max}$. For each vehicle, the global variables *congested*, *lastChangeAt*, and *lastChangeCongAt* are initialized with the values *false*, 0, and 0, respectively, and updated by Algorithm 7. Initially, we assume that the vehicle's current road segment is not congested (*congested = false*). If the current state is *congested = false* and v_{inst} stays below $\mathrm{V_{threshold}}$ for more than $\mathrm{T_{cong}}$ seconds, then the current state is set to *congested = false*, and γ is updated accordingly. Next, every time the algorithm is executed, the congestion level γ is increased. If v_{inst} stays above $\mathrm{V_{threshold}}$ for more than $\mathrm{T_{free}}$ seconds when *congested = true*, then γ will be 1, regardless of its previous value, which means that the vehicle has left a congested region. Whenever a vehicle enters another road segment, the congestion level estimation and the timers are not reset. The idea behind this is that there is a high probability that the next road segments have a traffic congestion state similar to the current road.

Figure 7.2 illustrates a vehicle γ measurement. The data was obtained through a simulation using $\mathrm{V_{threshold}}$ =5.63 m/s, $\mathrm{T_{cong}}$ =10 seconds, and $\mathrm{T_{free}}$ =20 seconds. We observe that during the initial 15 seconds, v_{inst} did not exceed ten continuously seconds

Algorithm 7 Procedure to update the congestion level γ

procedure UPDATECONGESTIONLEVEL
  if congested **then**
    if $v_{inst} \leq \mathrm{V}_{\mathrm{threshold}}$ **then** ▷ γ keeps increasing
      lastChangeAt $\leftarrow 0$
      $\gamma \leftarrow \min\left(\gamma_{\max}, 1 + \frac{\text{currentTimestamp - lastChangeCongAt}}{T_{cong}}\right)$
    else if lastChangeAt $= 0$ **then**
      lastChangeAt $\leftarrow$ currentTimestamp
    else if currentTimestamp - lastChangeAt $\geq T_{free}$ **then**
      congested $\leftarrow$ false ▷ Free-flow state
      $\gamma \leftarrow 1$
      lastChangeAt $\leftarrow 0$
    end if
  else if $v_{inst} \leq \mathrm{V}_{\mathrm{threshold}}$ **then**
    if lastChangeAt $= 0$ **then**
      lastChangeAt $\leftarrow$ currentTimestamp
    else if currentTimestamp - lastChangeAt $\geq T_{cong}$ **then**
      congested $\leftarrow$ true ▷ Congested road
      lastChangeCongAt $\leftarrow$ lastChangeAt
      lastChangeAt $\leftarrow 0$
      $\gamma \leftarrow \min\left(\gamma_{\max}, 1 + \frac{\text{currentTimestamp - lastChangeCongAt}}{T_{cong}}\right)$
    end if
  else
    lastChangeAt $\leftarrow 0$ ▷ Congestion level not changed
  end if
end procedure

above $V_{threshold}$. In this case, the estimate is that the road is in a free-flow condition ($\gamma = 1$). From 15 seconds on, v_{inst} starts decreasing and, after 20 seconds γ starts increasing linearly. After this, in three different moments of the simulation, the v_{inst} was above the threshold for more than ten consecutive seconds. Thus, in those moments, the congestion level was set to one indicating that the vehicle was crossing a non-congested road.

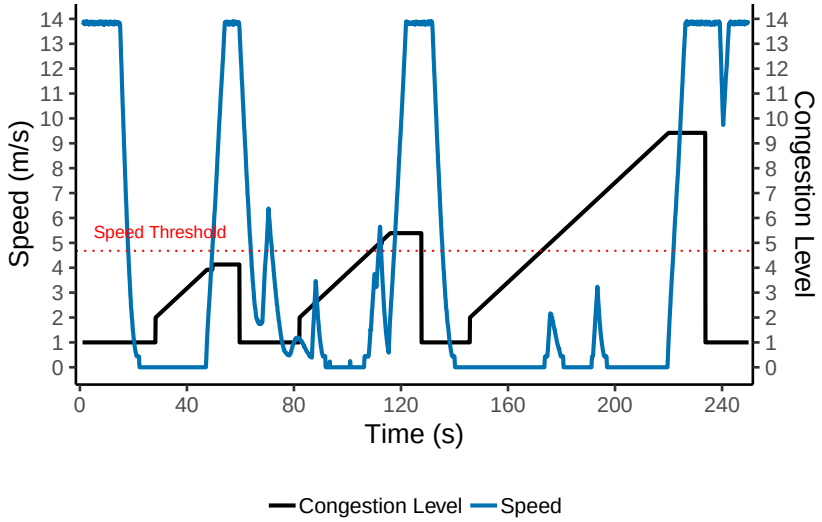


Figure 7.2: Example of a local traffic congestion level quantification.

7.3.3 Traffic Congestion Level Dissemination

Vehicles need to share their traffic congestion estimations for two reasons. First, it is necessary to obtain traffic data from regions that are out of communication range to be able to compute the best routes to the vehicle's destination. Second, individual estimations may experience outliers temporarily (e.g., when a driver makes a short stop at the side of the road while waiting for a passenger). Thus, the DisTraC protocol has two data-sharing mechanisms.

The first data sharing mechanism aims at sharing data between vehicles that are close to each other. Therefore, each vehicle includes a dataset called Local Congestion Estimate (LCE) in its periodical beacon. LCE has the congestion measurements of the last η roads traveled by the vehicle. For each road, there is a pair $\langle road_{id}, \gamma \rangle$, where $road_{id}$ is the road identifier, and γ is the last congestion level of that road measured by the vehicle. We use $\eta = 2$ in this work. That is, the vehicle shares the congestion data

of the current road and the road previously traveled. Note that we can include this data into the Cooperative Awareness Messages (CAMs) [177], or its equivalent Basic Safety Messages (BSMs) [91]. Thus, the LCE transmission does not lead to an increase in the transmission overhead.

The second data-sharing mechanism aims at disseminating the congestion information to regions that are out of the one-hop range communication. We propose a cluster-based scheme to reduce the communication overhead. Each cluster, as illustrated in Figure 7.3, is a fixed road segment of maximum length equal to the vehicle communication range. In summary, for each cluster, at most every $T_{broadcast}$ seconds, one vehicle will perform a multi-hop broadcast containing congestion data generated by the vehicles at the same cluster.

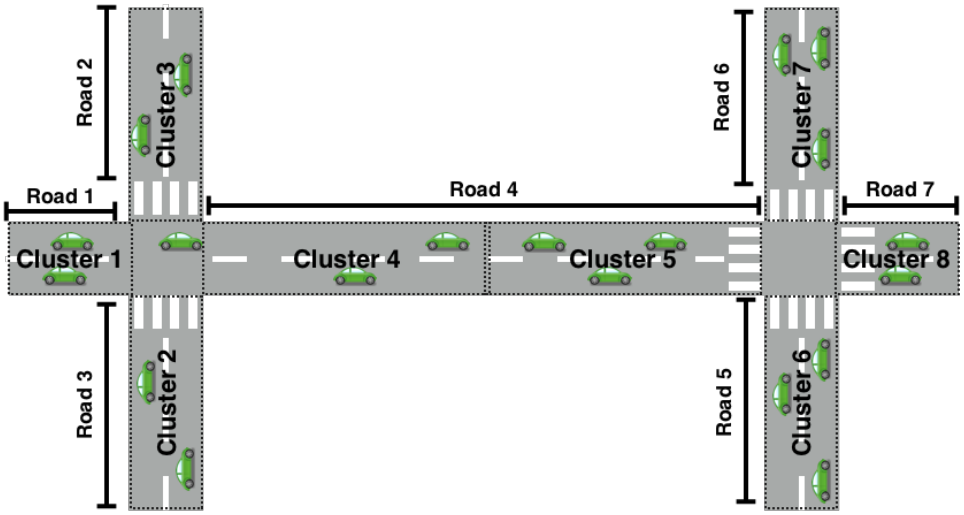


Figure 7.3: Definition of clusters and roads.

Each vehicle decides, independently from the others, if and when it will perform the broadcast by executing Algorithm 8 every second. The first step is to update the current congestion level (Algorithm 7). Next, the vehicle must check the following three conditions before performing the broadcast:

1. The new congestion level is different from the previous value;
2. The vehicle did not receive a broadcast from the current cluster in the last $T_{broadcast}$ seconds; and
3. The vehicle is the cluster-head of its current cluster.

Algorithm 8 Procedure to perform the multi-hop broadcast

```
1: procedure verifyAndPerformBroadcast
2:     γ_before ← γ
3:     γ ← updateCongestionLevel
4:     if γ ≠ γ_before then                          ▷ Change in the congestion level
5:         t ← getClusterLastBroadcastTime(currentClusterID)
6:         if (currentTimestamp − t ≥ T_min) ∧ isClusterHead(vehicle) then
7:             congestionData ← {LCE_currentVehicle}
8:             for LCE_neighbor ∈ LCE_neighbors do
9:                 congestionData ← congestionData ∪ {LCE_neighbor}
10:            end for
11:            broadcast(congestionData)              ▷ Multi-hop broadcast
12:        end if
13:    end if
14: end procedure
```

The cluster-based scheme works as follows. We assume that each vehicle has a digital map with the center position and the radius of all clusters. Besides that, the vehicles keep track of their neighbors' positions through the received beacons. First, the vehicle checks to which cluster it belongs by selecting from the list of clusters of its current road and choosing the one with a center position closer to the current vehicle's position. Next, the vehicle knows it is the cluster-head when there is no neighbor on the same road that is closer to the cluster's center than itself. This whole procedure, detailed in Algorithm 9, happens in such a way that vehicles do not have to send any additional data across the network.

In this work, we implemented a simple flooding protocol to perform the multi-hop broadcast. Nevertheless, we can use other protocols to reduce communication overhead without decreasing the delivery rate. This investigation is part of future work.

Whenever a vehicle receives a broadcast containing congestion data from a cluster, it updates a local database that stores the moment of the last broadcast received from each cluster. Besides, the DisTraC protocol component called Traffic Congestion Data Fusion aggregates the received data, as explained in the next section.

Algorithm 9 Procedure to check if the vehicle is the cluster-head

Output: If the vehicle is the cluster-head

```
1: procedure isClusterHead(vehicle)
2:     clusters ← {⟨center, radius⟩}                    ▷ clusters of the current road
3:     currentCluster ← null
4:     minDist ← ∞
5:     vehPos ← vehicle.position
6:     for cluster ∈ clusters do
7:         clusterDist ← Dist(cluster.center, vehPos)
8:         if clusterDist < minDist then
9:             currentCluster ← cluster
10:            minDist ← clusterDist
11:        end if
12:    end for
13:    clusterPos ← currentCluster.center
14:    vehDis ← Dist(clusterPos, vehPos)
15:    for neighbor ∈ neighbors do
16:        dist ← Dist(neighbor.center, clusterPos)
17:        if dist < vehDist then
18:            return false          ▷ Found another vehicle closer to the cluster center
19:        end if
20:    end for
21:    return true                                      ▷ The vehicle is the cluster-head
22: end procedure
```

7.3.4 Traffic Congestion Data Fusion

Vehicles receive new traffic congestion estimations from three sources: (i) neighbors that send periodical beacons, (ii) the cluster-head from different areas that send regular multi-hop broadcasts, and (iii) the vehicle itself that keeps estimating the current road congestion level.

Each vehicle maintains two databases Γ and $\Gamma_{\text{aggregated}}$ with congestion information of roads. Γ entries are of the form $\langle \text{roadID} \rangle \rightarrow (\langle \text{vehicleID} \rangle \rightarrow \{\gamma, \text{t}\})$, where γ is the congestion level and t is the timestamp when γ was estimated. We use Γ_{limit} to limit the cardinality of each road entry in Γ. The vehicle adds its own congestion level estimation to the database Γ with a frequency equal to the beaconing frequency (1 Hz in this work). Whenever a vehicle adds a new value to the database Γ (from itself or another source), it will aggregate this information into the database $\Gamma_{\text{aggregated}}$ by employing Algorithm 10. In summary, for a given road, we use all of its congestion level estimations stored in Γ to compute a single congestion level $\gamma_{\text{aggregated}}$, assigning higher weights for the most recent estimations.

Figure 7.4 presents a comparison between the congestion level estimated by the vehicle (Own Congestion Level) for its current road and the congestion level stored in $\Gamma_{\text{aggregated}}$ (Database Congestion Level) for the same road. The data received from other vehicles incur in the changes observed in the database congestion level that does not happen along with a change in the own congestion level. For example, in some situations, the database congestion level begins to change even before the vehicle slows down, which happens because the vehicle starts receiving traffic data even before it enters the congested road. It is important to note that the estimations made by the vehicle and the database values remain compatible throughout the simulation.

7.3.5 Traffic Congestion Reduction

The traffic congestion data fusion module invokes the traffic congestion reduction module whenever there is a change in the database $\gamma_{\text{aggregated}}$. Next, the vehicle uses the updated database $\gamma_{\text{aggregated}}$ to estimate the best routes individually. To this, we assume that each vehicle has an initial position and a destination position. The vehicle also has a digital map initialized with a weight assigned to each road equal to the time needed to cross the road at the maximum allowed speed. Whenever an update occurs in $\gamma_{\text{aggregated}}$, we update the weight for the given road using Equation 7.1, where ω is the updated road weight, L is the road length, V_{max} is the road maximum allowed speed and γ_{db} is the road

Algorithm 10 Procedure to aggregate the new congestion level estimation.

procedure AGGREGATEDATA(roadID, γ_{received}, timestamp, vehicleID)
  $V \leftarrow \Gamma(\text{roadID})$
  $V(\text{vehicleID}) \leftarrow \{\gamma_{\text{received}}, \text{timestamp}\}$
  while $|V| > \Gamma_{\text{limit}}$ **do**
    remove the oldest entry from V based on its timestamp
  end while
  $\Gamma(\text{roadID}) \leftarrow V$ ▷ update Γ
  oldestEntry $\leftarrow$ get the oldest entry from Γ(roadID)
  oldestEntryTimestamp $\leftarrow$ oldestEntry.timestamp
  $\text{weight}_{\text{sum}} \leftarrow 0$
  $\gamma_{\text{sum}} \leftarrow 0$
  for each $e \in \Gamma(\text{roadID})$ **do**
    $\text{weight} \leftarrow e.\text{timestamp} - \text{oldestEntryTimestamp} + 1$
    $\text{weight}_{\text{sum}} \leftarrow \text{weight}_{\text{sum}} + \text{weight}$
    $\gamma_{\text{sum}} \leftarrow e.\gamma \times \text{weight}$
  end for
  $\gamma_{\text{aggregated}} \leftarrow \frac{\gamma_{\text{sum}}}{\text{weight}_{\text{sum}}}$
  $\Gamma_{\text{aggregated}}(\text{roadID}) \leftarrow \gamma_{\text{aggregated}}$ ▷ update $\Gamma_{\text{aggregated}}$
end procedure

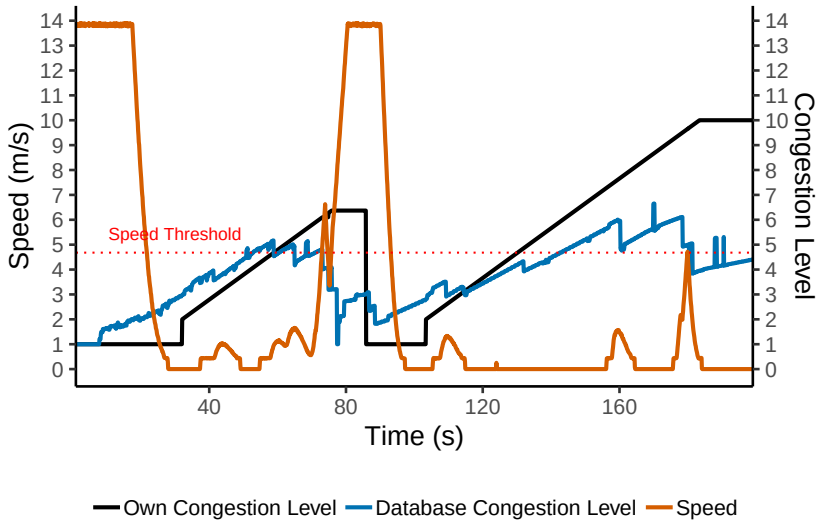


Figure 7.4: Example of a comparison between the road congestion level measurement and the congestion level value in the γ.

congestion level got from the database $\gamma_{\text{aggregated}}$. The vehicle uses the Dijkstra Shortest Path Algorithm [53] to calculate the new route after updating the road weights.

$$\omega = \frac{L}{V_{max}} \times \gamma_{db} \tag{7.1}$$

One can wonder if the protocol does not incur in the formation of bottleneck roads, as all vehicles use the same routing algorithm. DisTraC protocol avoids this mainly for two reasons. First, each vehicle has a distinct $\gamma_{\text{aggregated}}$, and so the weights a vehicle assigns to the roads are different from weights given by other vehicles. Second, the vehicles are always recomputing their best routes. Therefore, when a road congestion level starts increasing, other vehicles will be less likely to choose this road soon, thus balancing the traffic across different roads.

7.4 Performance Evaluation

We evaluate the performance of DisTraC through simulations aiming at verifying the protocol's ability to reduce traffic jams as well as the amount of communication overhead caused in the vehicular network. First, in Section 7.4.3, we compare DisTraC with other solutions. Next, in Section 7.4.4, we evaluate the main parameters of the protocol. In Section 7.4.5, we evaluate DisTraC in a real-world, large-scale simulation scenario.

7.4.1 Limitations of Simulation-based Evaluation

We usually divide the methods for computer performance evaluations into three areas: performance measurement, analytic performance modeling, and simulation. As expected, each one has its advantages, disadvantages, and applicability [82]. Simulation-based evaluation is the most common technique in vehicular network research because of its versatility, which enables the evaluation of a wide range of scenarios. That is because the cost of such experiments would be prohibitive for performance measurements, and, on the other hand, analytical modeling is typically too hard for complex scenarios. However, it is essential to note that performance evaluation based on simulations has its limitations. For instance, even if they are carefully designed, simulations may produce results different from that observed through measurements. Besides that, the best approach to evaluate vehicular networks employs real-world traces of vehicles, but there is a lack of comprehensive datasets that allow extensive simulations.

7.4.2 Simulation Setup

We implement and simulate the DisTraC protocol and related work using the discrete event simulator OMNeT++ 5.1 [189], the traffic and urban mobility simulator SUMO 0.25.0 [104], and the vehicular network simulator Veins 4.7.1 [174]. The simulation scenario, generated with the SUMO simulator, is a 5×5 Manhattan Grid under a region of 1 km^2 with road segments that are 250 meters long, two lanes in each direction and speed limit of 13.9 m/s. We randomly generated routes of at least 1 km long, and consider different traffic demands by varying the total number of vehicles in 150, 200, 250 and 300. The duration of the simulations was 10 minutes. We replicate each scenario five times and plotted graphs with a confidence interval of 95%. The vehicle's transmission power was 7.1 milliwatts (mW) for V2V communication, which gives a communication radius of 300 meters, approximately.

7.4.3 Comparison with Related Work

We compare DisTraC with its initial version [44] and the solutions found in [67] and [135], as these solutions have the characteristics discussed in Section 7.2. In the Milojevic et al. [135] algorithm simulations, we add our mechanism to route the vehicles but using their method to generate traffic congestion data. This improvement is necessary because their solution does not include a traffic congestion reduction mechanism. In this way, we can

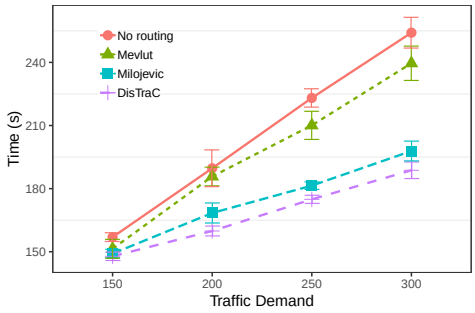
evaluate the quality of the traffic congestion data generated by their technique. Besides that, we increased the transmission power on the simulations of the protocol of Garip et al. [67] to 20 mW, which results in a transmission range of around 500 meters. This modification was required to improve their results as they use only one-hop broadcast. We set the DisTraC parameters with the values shown in Table 7.2.

First, we evaluate the ability of the protocols to reduce traffic congestion. Figure 7.5 shows the average travel times, travel lengths, completed routes, and CO_2 emission. Through Figure 7.5a, we verify that DisTraC is the most effective in reducing the travel time of vehicles, with an improvement of approximately 26% in congested scenarios when compared with *no routing* (which represents vehicles using the shortest path that do not re-route).

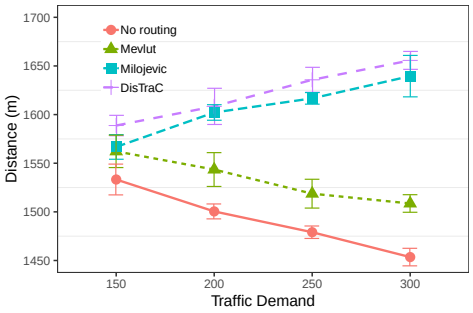
The results presented in Figures 7.5b and 7.5c confirm that DisTraC is more efficient than other protocols in reducing traffic jams. Figure 7.5b shows the average travel distance. When employing a re-routing strategy, there are two reasons for which the vehicles can travel longer distances. First, as the initial routes are calculated using the shortest path with only the road length, the updated routes always are greater than or equal to the initial routes. Second, when the vehicles do not re-route, they get stuck in traffic jams more often, and many of them do not complete their routes before the end of the simulation. Figure 7.5c shows the percentage of vehicles that conclude their routes. We can see that more vehicles complete their routes when running the DisTraC protocol. Besides that, there is a correlation between the improvement of travel time and the percentage of vehicles that reach their destination. The more efficient the solution in reducing travel time is, the higher the number of vehicles that can arrive at the destination before the conclusion of the simulation.

Finally, Figure 7.5d presents the CO_2 emission rate, which we determine using a statistical transmission model called EMIT [28]. Sommer et al. [175] include more information regarding the implementation of the transmission model in the Veins simulator. Two factors significantly influence the CO_2 emission rate. First, it is the duration and size of the trip, as fuel consumption is directly related to these variables. Second, congested situations where most vehicles are in *stop-and-go* traffic increases emissions per kilometer yet further [73]. Thus, the CO_2 emission rate is lower when using the solutions that can effectively decrease the vehicle travel times and so DisTraC presents the best results.

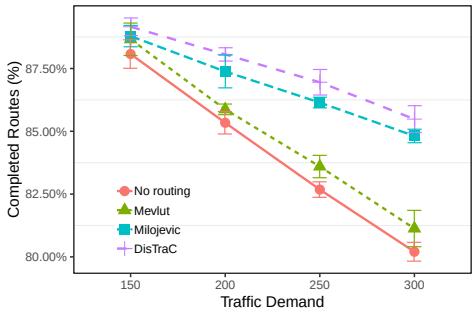
Figure 7.6 shows the results of the transmission overhead evaluation. As expected, every protocol's overhead increases in scenarios with more traffic congestion because of



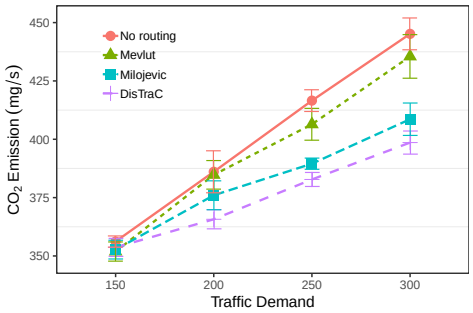
(a) The average time each vehicle takes to complete the route.



(b) The average distance traveled by each vehicle.



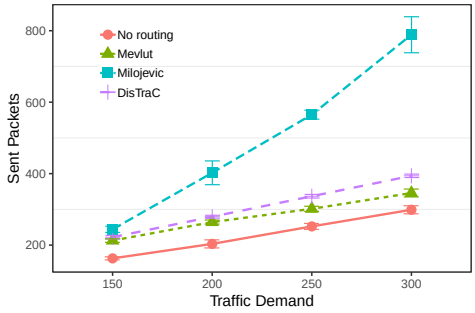
(c) The percentage of vehicles that complete their routes.



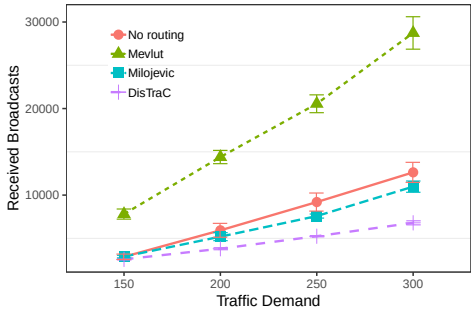
(d) The average CO_2 emission rate.

Figure 7.5: Comparison with related work of the protocol's ability to reduce traffic congestion.

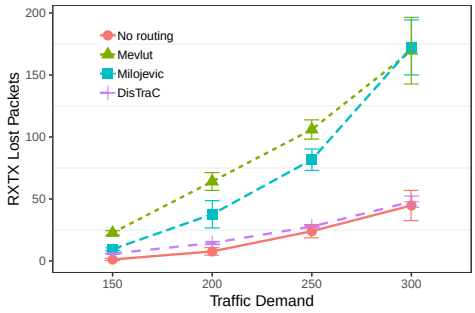
packets retransmissions. These retransmissions also result in higher packet loss rates, as shown in Figures 7.6c and 7.6d. The higher number of sent and received packets of DisTraC when compared with its old version is due to the new traffic congestion dissemination module. However, as shown in Section 7.4.4, we can decrease DisTraC's overhead without affecting traffic congestion reduction by adjusting the value of $T_{broadcast}$.



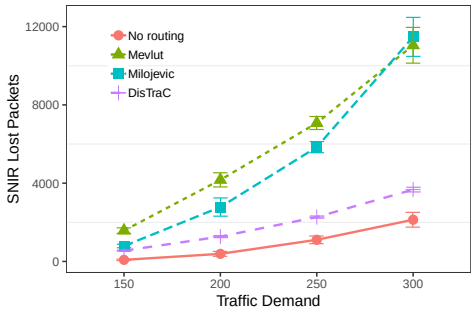
(a) The average number of messages transmitted by vehicles.



(b) The average number of messages received by each vehicle.



(c) The average number of packets lost because the vehicle was sending while receiving.



(d) The average number of packets lost due to bit errors.

Figure 7.6: Comparison with related work of the protocol's communication overhead.

7.4.4 Evaluation of Parameters

The values we chose for the parameters of DisTraC directly affect the performance of the protocol. For instance, the higher the value of $T_{broadcast}$, the higher the frequency of cluster data broadcasts, which increases the communication overhead in exchange for keeping the vehicles more up to date about traffic congestion in remote areas. Thus, we evaluate the most critical parameters by varying one at a time while keeping the others fixed with the values in Table 7.2.

Minimum Broadcast Time Interval

We prevent the broadcast storm problem by imposing a minimum broadcast time interval on the clusters. In summary, the cluster-head performs the multi-hop broadcast only if it did not receive a broadcast from its cluster in the last 2 seconds. We simulate different values of $T_{broadcast}$ from 5 seconds to 2 minutes. Figure 7.7 shows the results.

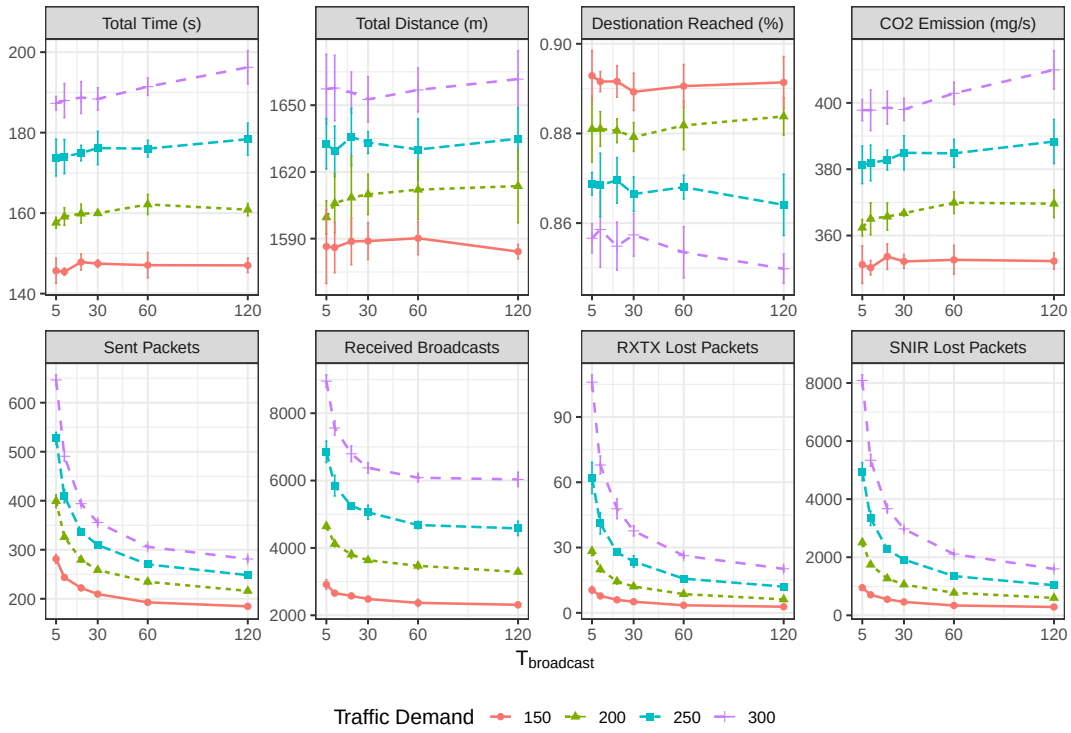


Figure 7.7: Evaluation results of the parameter $T_{broadcast}$.

As expected, the transmission overhead is higher when $T_{broadcast}$ is lower. Besides that, the growth rate of the number of packets sent is higher in scenarios with more vehicles, which happens because in scenarios with lower traffic demand, it is more common for clusters to be empty for periods longer than $T_{broadcast}$ seconds. On the other side, varying $T_{broadcast}$ from 5 to 30 seconds did not result in significant changes in the average travel time of vehicles. This result demonstrates that it is better to use values for $T_{broadcast}$ of at least 20 seconds, thus reducing both the communication overhead and traffic jams.

Maximum Number of Entries per Road in Γ

We use the parameter Γ_{limit}, which bounds the maximum number of entries per road in the database Γ of each vehicle to limit the amount of memory used by the protocol. However, low values of Γ_{limit} may decrease the quality of traffic congestion estimates as vehicles are required to discard older estimates of other vehicles. Figure 7.8 shows the results of varying Γ_{limit} in 5, 10, 20, 30, 120.

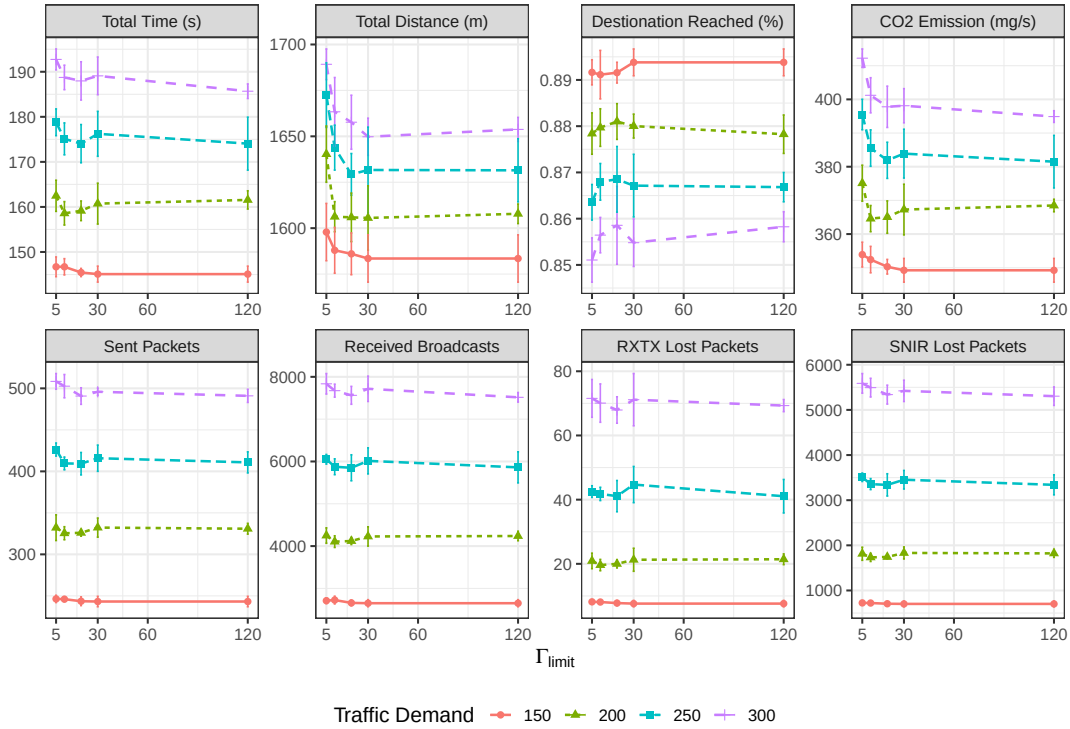


Figure 7.8: Evaluation results of the parameter Γ_{limit}.

We can verify that the best improvements in traffic congestion were using higher values of Γ_{limit}. Despite that, the road lengths physically bound the number of vehicles on the same cluster. Therefore, it is rarely necessary to remove old entries when Γ_{limit} is greater than or equal to 20, which makes the results of these simulations similar.

Finally, the results confirm that parameter Γ_{limit} does not cause significant changes in communication overhead. The reason behind this is that communication overload is mainly affected by the number of vehicles and the broadcast rate. Therefore, $T_{\text{broadcast}}$ is the only protocol parameter that significantly impacts the number of sent packets.

Congestion Level Increase Time Interval

The parameter T_{cong} determines how fast we should increment the congestion level when the vehicle is at low speed. Besides that, it defines how long the vehicle should be idling before a free-to-congested traffic change is detected. Figure 7.9 shows the results for different values of T_{cong}.

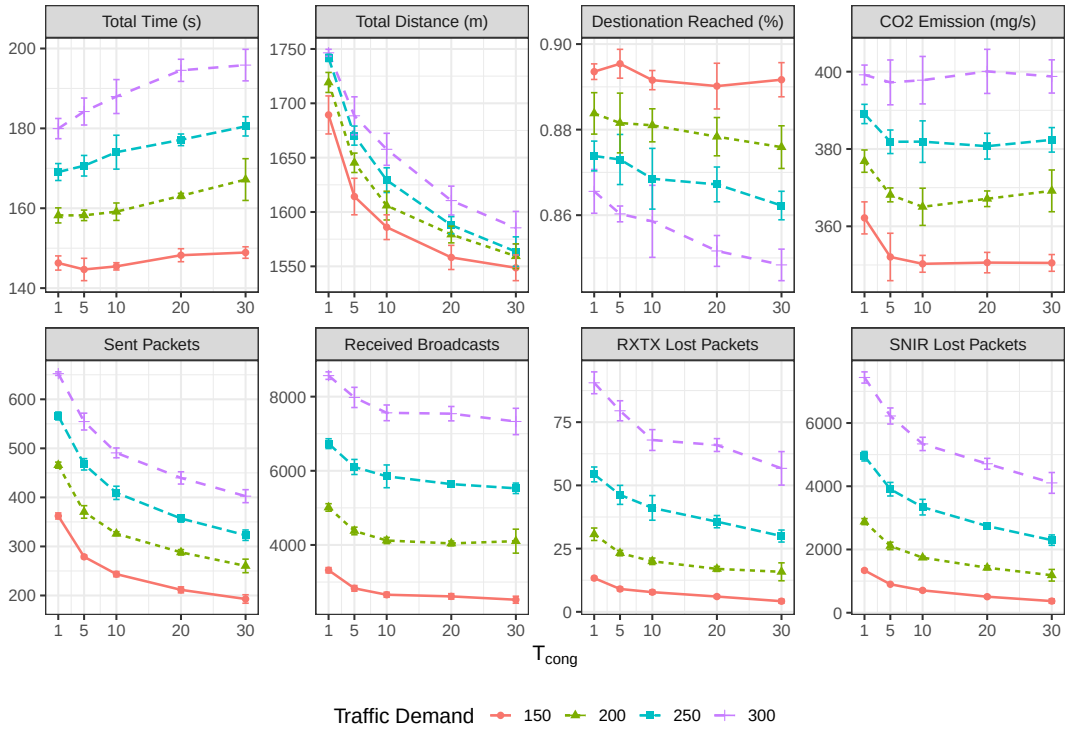


Figure 7.9: Evaluation results of the parameter T_{cong}.

Lower values of T_{cong} make the protocol react faster when the vehicle enters congested regions and, thus, leads to better travel times. However, the cluster-head performs multi-hop broadcast only when the congestion level changes. Therefore, lower values of T_{cong} result in a higher number of packets sent.

Congestion Level Reset Time Interval

The congestion level reset time interval, defined by the parameter T_{free}, represents how long the vehicle should remain above the threshold speed before detecting a traffic state change from congested to free-flow. We simulate different values of T_{free} from 1 second to 60 seconds. Figure 7.10 shows the results. Similar to what occurs with T_{cong}, T_{free}

makes the vehicle to react faster to traffic changes. However, unlike T_{cong}, when the value of the parameter is too low, the number of packets sent is lower, and the average travel time is higher, which happens because with DisTraC vehicles perform multi-hop broadcasts more often when traffic state is congested.

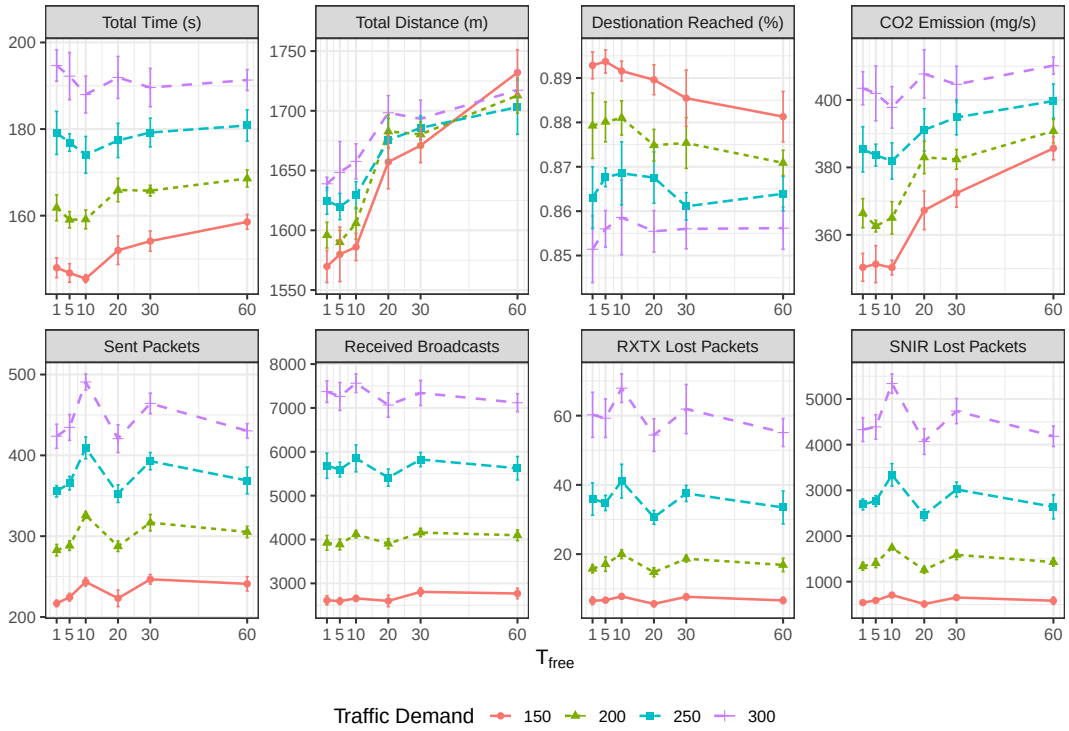


Figure 7.10: Evaluation results of the parameter T_{free}.

7.4.5 LuST Scenario

Perform simulations of vehicular network protocols in large-scale realistic scenarios is challenging, especially when the objective is to assess traffic congestion control. First, although it is easy to extract publicly available road network data using tools such as the *OpenStreetMap* [78], we still need to perform many fixings before using its road network. For instance, we have to fix the turn restrictions in every road segment to represent the traffic correctly. Second, there are few data available on actual traffic demand. Thus, in most cases, we need to generate synthetic traffic demand, which can be unrealistic for evaluating traffic congestion. Last, simulating large-scale scenarios requires a lot of computing processing time. For instance, in the previous version of this work, we took

almost one month to conclude twenty minutes of a single simulation of the protocol in a large-scale scenario. Therefore, in this section, we present preliminary results of the evaluation of DisTraC in a real-world, large-scale scenario. To this end, we used the Luxembourg SUMO Traffic (LuST) Scenario [42], illustrated in Figure 7.11, which is one of the few publicly available datasets that have the features we need to evaluate traffic congestion.

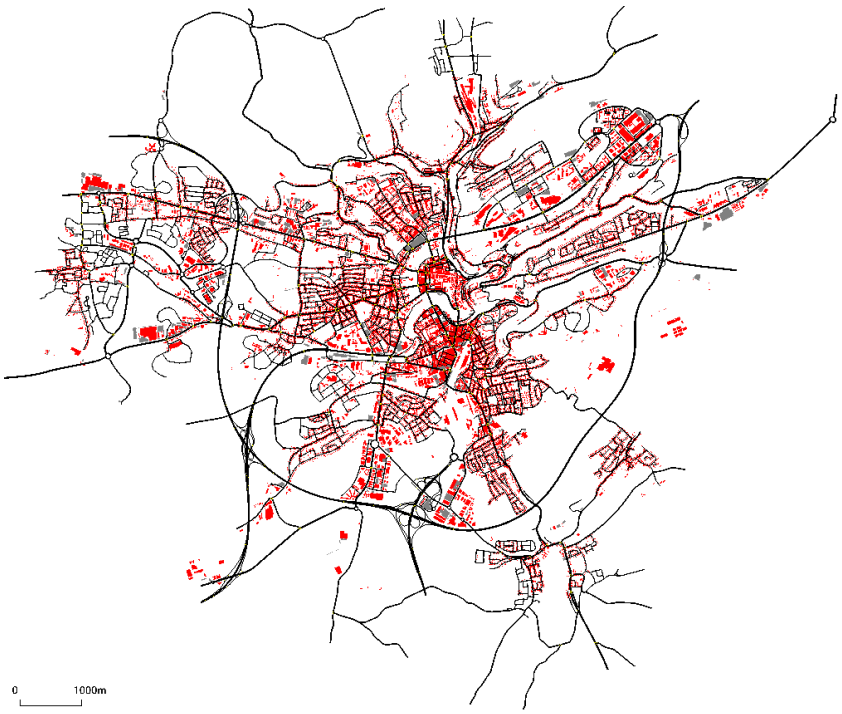


Figure 7.11: Map of Luxembourg city used in the simulations.

LuST is a large-scale scenario that contains one day of traffic data on the city of Luxembourg. However, due to its high computational complexity, we were able to simulate only the DisTraC protocol during a few minutes of traffic using this scenario. With these simulations, we could observe an improvement in vehicle travel time when compared to the situation where vehicles use the shortest route. We plan to investigate the performance of related work as future work.

We choose two different moments to simulate the DisTraC protocol, one with moderate traffic demand the other with high traffic demand. Figure 7.12 shows the number of vehicles running over the day and highlights the moments we simulate. The simulation of moderate traffic demand refers to the period between 1:00 p.m. and 1:20 p.m., while the simulation of high traffic demand corresponds to the period between 7:30 a.m. and

7:50 a.m.

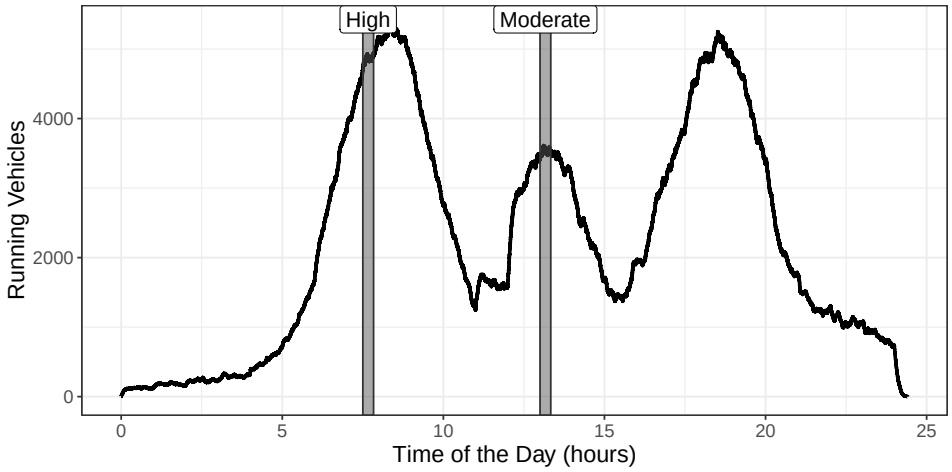


Figure 7.12: Traffic demand of the LuST scenario over the day and the moments chosen for simulation.

Figure 7.13 presents the results of the LuST scenario. In the scenario with moderate traffic demand, the improvement of the average travel time of the vehicles was only 2.2%. In the scenario with high traffic demand, the average reduction of vehicle travel time was 7.6%. The results show that the DisTraC protocol's ability to reduce vehicle travel time is greater in scenarios with greater traffic congestion. We believe that the improvements would be better if we had carried out longer simulations. Figure 7.14, which shows the average travel time of vehicles by departure time, reinforces our hypothesis, as we can verify that the reduction of travel time increases as the departure time increases.

Figure 7.13 also shows the impact of the protocol on the travel distance and CO_2 emission rate. As in the Manhattan Grid scenario, the protocol makes vehicles travel longer distances when compared to vehicles traveling on the shortest paths. However, because of the reduced simulation time, it was not enough to reduce the CO_2 emission rate. Besides that, the traveling distance is smaller in the scenario with higher traffic demand rather than the scenario with moderate traffic demand for two reasons. First, because in the scenario of high traffic demand, the vehicles get stuck in traffic jams more often. Second, because we computed the results from vehicles that concluded their routes, which, especially in the scenario with the highest traffic demand, are those with smaller routes.

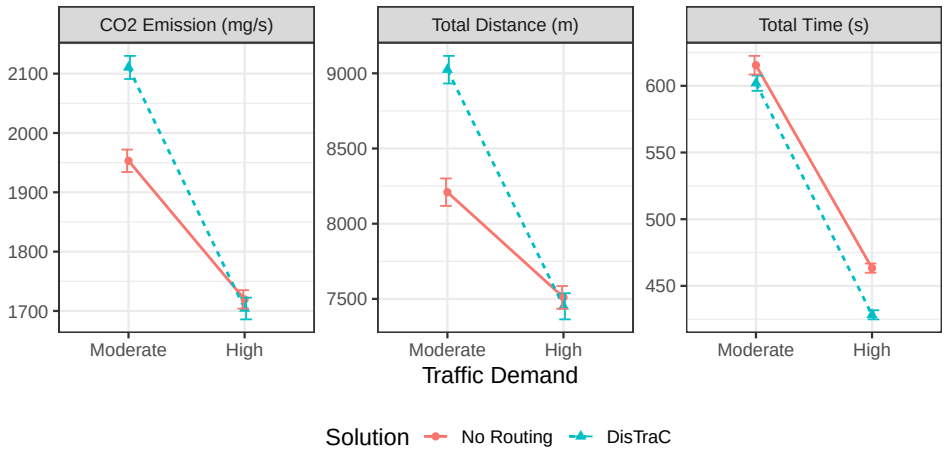


Figure 7.13: LuST scenario results.

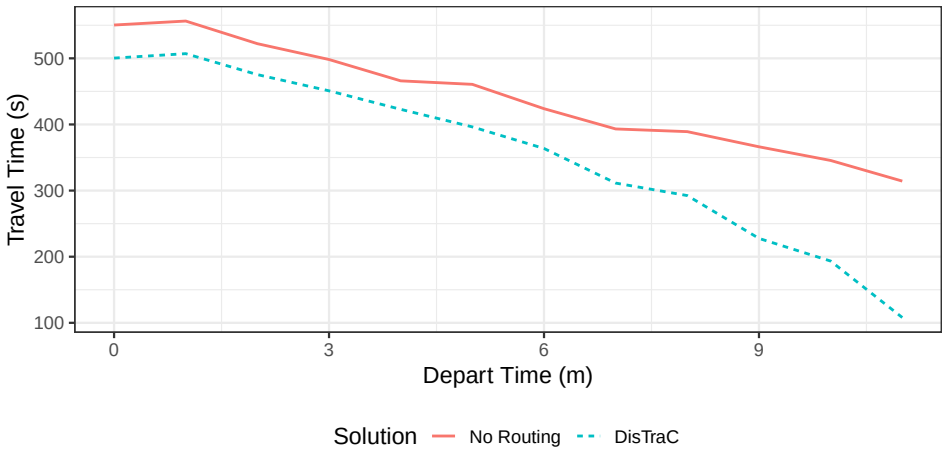


Figure 7.14: Average travel time of vehicles by departure time.

7.5 Chapter Remarks

In this Chapter, we proposed the DisTraC protocol for vehicular networks. DisTraC aims to reduce the average travel time of vehicles by employing V2V communication. The protocol is independent of the existence of roadside units or other external infrastructures as it uses only V2V communication.

We compared DisTraC with other solutions published in the literature through simulations. The results showed that DisTraC is the most effective in reducing the average travel time of vehicles. Moreover, DisTraC presented a low communication overhead. The evaluation of the protocol parameters showed the impact of each parameter on the protocol performance, and how we can configure them to balance the protocol's capability in reducing traffic congestion and the communication overhead it produces.

As future work, we plan to carry out an extensive study in a real-world scenario, including other solutions. For this, it will be necessary to implement modules in the simulator that allow more efficient use of the available computational resources. Besides that, we plan to investigate the employment of smart dissemination algorithms, such as prediction-based delay-tolerant broadcast algorithms, to reduce the protocol's communication overhead without decreasing its capacity to reduce the average travel time of vehicles.

Table 7.1: Reference list of Traffic management using vehicular networks

Reference	Communication		Traffic Management		
	VANET	Cel.	Quan.	Dis.	Rou.
DisTraC [44]	**V2V**	**No**	**Yes**	**Yes**	**Yes**
SOTIS [194]	V2V	No	Yes	Yes	No
CoTEC [18]	V2V	No	Yes	Yes	No
DTMon [11]	V2X	No	No	Yes	No
VAM [75]	V2V	No	No	No	Yes
DTraMS [68]	V2X	No	Yes	Yes	No
ECODE [207]	V2X	No	Yes	Yes	Yes
H. Noori et al. [144]	V2X	No	Yes	Yes	Yes
ABEONA [71]	V2V	No	Yes	No	No
R. Aissaoui et al. [5]	V2X	No	Yes	No	No
M. Milojevic et al. [135]	V2V	No	Yes	Yes	No
E-VeT [149]	V2X	No	No	No	Yes
P. Bellavista et al. [20]	V2X	No	Yes	No	No
CARTIM [10]	V2X	No	Yes	Yes	Yes
M. T. Garip et al. [67]	V2V	No	No	No	Yes
A. Elvery et al. [59]	V2X	No	No	No	No
M. Wang et al. [193]	V2X	Yes	Yes	Yes	Yes
J. Jeong et al. [93]	V2X	Yes	Yes	Yes	Yes
N. Cárdenas-Benítez et al. [29]	V2X	No	Yes	Yes	Yes
G. Martuscelli et al. [131]	V2V	No	No	Yes	No
DIVERT [150]	V2V	Yes	Yes	Yes	Yes
C. Guo et al. [74]	V2X	No	Yes	Yes	Yes

Table 7.2: DisTraC default parameters value.

Parameter	Value
$T_{broadcast}$	10 s
Γ_{limit}	20
T_{cong}	10 s
T_{free}	10 s
γ_{max}	10 s
$V_{threshold}$	4.63 m/s

Chapter 8

Vehicle-assisted Data Delivery based on Trajectory Prediction

This chapter proposes a novel vehicle-assisted data delivery algorithm called VDDTP. VDDTP creates an extended trajectory model and uses predicted road-network constrained trajectories to calculate packet delivery probabilities. Then, it applies the predicted trajectories and some proposed heuristics in a data forwarding strategy, aiming to improve the vehicular network's global metrics (i.e., delivery ratio, communication overhead, and delivery delay). We perform extensive experiments using the Rio Center dataset, a novel real-world and large-scale trajectory dataset for evaluating vehicular networks applications. The results demonstrate the algorithm's ability to improve the global metrics compared to some related work.

8.1 Introduction

Vehicular networks (VANETs) are communication systems that enable vehicles to exchange data with each other (V2V communication), and with infrastructures located along the edge of the roads (V2I communication), called Road Side Units (RSUs) [81]. VANETs are an essential component to construct smart cities, as they are enablers of new applications that aim at improving the quality of life in urban areas [58]. For instance, some smart mobility applications use vehicular networks to detect and reduce traffic congestion cooperatively [44, 49].

The vehicular network's environment has some characteristics that pose communication challenges generally not found in traditional networks. For instance, the network

topology quickly changes because vehicles are constantly moving. Besides that, routing protocols need to consider traffic congestion scenarios to avoid the broadcast storm problem. At the same time, regions with few vehicles result in a sparse network, which hinders ad hoc communication [116]. The challenges mentioned above make it necessary to develop alternative techniques to improve vehicle-assisted data delivery [48].

Recent studies demonstrate the importance of vehicle mobility for improving data delivery in vehicular networks [96, 121]. Furthermore, the advances and popularization of location-acquisition technologies lead to the availability of the so-called "Big Trajectory Data". In addition to the importance in several research areas, such as location-based social networks and traffic analysis, this massive trajectory data has a pivotal role in learning mobility patterns and, therefore, is essential in designing Intelligent Vehicular Networks [46].

Most studies in the literature that attempt to apply mobility information to improve vehicular networks ignore common issues found in vehicle trajectory data. For instance, GPS-based trajectories usually have a low sampling rate and positional errors, making their applicability difficult. Therefore, preprocessing techniques, such as trajectory reconstruction that convert raw GPS-based trajectories into road-network constrained trajectories, are necessary to improve data accuracy and applications [47].

In this Chapter, we propose the VDDTP algorithm, a novel **V**ehicle-assisted **D**ata **D**elivery algorithm based on **T**rajectory **P**rediction. The VDDTP algorithm first enhances data accuracy by applying processing techniques to create an extended trajectory model. Besides that, we propose some heuristics to calculate packet delivery probabilities from predicted road-network constrained trajectories. Finally, we present data forwarding strategies to improve the delivery ratio and reduce delivery overhead and delay. We perform extensive experiments from a real-world, large-scale dataset to demonstrate the algorithm's ability to upgrade the global metrics of the vehicular network when compared to related work.

We organize the remainder of this Chapter as follows. Section 8.2 reviews the related work. Section 8.3 presents the reference model and formulates the problem. Section 8.4 describes the proposed VDDTP algorithm, and Section 8.5 analyzes the algorithm's performance and compares it with related work. Finally, Section 8.6 concludes this work.

8.2 Related Work

Most of the literature regarding data delivery in vehicular networks derives from delay-tolerant networks and employs straightforward strategies such as flooding. In flooding or epidemic routing techniques, vehicles exchange data packets with newly discovered neighbor nodes that do not possess a copy of the packet. These techniques can reach a high packet delivery ratio but present a high communication overhead, leading to the broadcast storm problem [188].

Many studies propose and evaluate alternative approaches to overcome this issue [214, 199, 96, 121, 31], and more recently, advanced techniques extract vehicle mobility information from historical trajectory data to improve data delivery [1]. For instance, Jiang et al. [96] developed message forwarding metrics extracted from the future trajectory of vehicles to calculate the probability of a vehicle delivering a message to a region. However, their metrics have some drawbacks. First, the "encounter cases" calculation is flawed because it does not consider many encounters. For instance, they do not consider the encounter of vehicles in parallel roads. Another significant drawback of their method is that it assumes the availability of future trajectories from on-board navigation systems, which is unrealistic. Besides that, they performed experiments with a small transmission range, which favors their method because it cannot detect most of the encounters of vehicles a little further apart.

Wu et al. [199] mined an extensive dataset and demonstrated through a conditional entropy analysis that the future trajectory of a vehicle has a high correlation with its previous trajectory. Based on their findings, they developed multiple order Markov chains for predicting the future trajectory of vehicles. With the future trajectory of vehicles, they proposed an analytical model to derive a packet's delivery probability theoretically. Next, they proposed centralized and distributed algorithms for vehicle-assisted data delivery. Their technique divides the region into fixed cells, which reduces the applicability of the method. Besides that, their solution uses an algorithm with exponential computational complexity, making it impractical.

Liu et al. [121] proposed a deep learning algorithm called DeepVDD to facilitate vehicle-assisted data delivery in a scenario with roadside units (RSU). They use deep learning to predict a vehicle's trajectory and propose a data forwarding strategy that verifies if the vehicle's future trajectory will contact the RSU directly. A disadvantage of their algorithm is that the training phase relies on the previous information of the RSU positions, restricting its applicability to just that scenario.

The proposals discussed above have at least one of the following drawbacks:

- They assume the availability of future trajectories (e.g., from on-board navigation systems).
- They use GPS-based trajectories but ignore positional errors.
- They have high computational complexity.

Because of these shortcomings, those proposals have limited applicability, and some of them are impractical for any scenario (e.g., algorithms with exponential time complexity). Therefore, we proposed in this Chapter a novel vehicle-assisted data delivery algorithm called VDDTP that overcomes these drawbacks. VDDTP uses road-network constrained trajectories as the base model, reducing problems with positional GPS data errors. Furthermore, the VDDTP employs an advanced prediction algorithm instead of assuming the availability of future trajectory from on-board navigation systems.

8.3 Reference Model and Problem Formulation

This section introduces some basic notation used through this Chapter and formulates the vehicle-assisted data delivery problem.

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ be a directed graph that represents the road-network of a given region, as defined in Chapter 2. That is, the set of edges $\mathcal{E}$ represents the roads, and the set of vertices $\mathcal{V}$ represents the road intersections. Each edge $e \in \mathcal{E}$ connects a start vertex $s(e) \in \mathcal{V}$ to an end vertex $e(e) \in \mathcal{V}$, and each vertex $v \in \mathcal{V}$ is endowed with its location $l(v)$ (i.e., longitude and latitude), representing its coordinates on Earth. Moreover, each edge $e \in \mathcal{E}$ stores a *linestring* (i.e., an ordered set of coordinates) representing its shape, where the first and last coordinates are equal to the positions of $s(e)$ and $e(e)$, respectively.

The set of vehicles V represents the nodes of the vehicular network. Each vehicle $v \in V$ has a trajectory with start and end time equals $t_s(v)$ and $t_e(v)$, respectively (the time is slotted). The trajectory T_v of vehicle v is an ordered set of locations, denoted as

$$\mathrm{T}_v = \{l_v(1), l_v(2), \ldots, l_v(n)\}$$

where $l_v(i)$ represents the vehicle's exact position at time slot $t_s(v) + i - 1$, and each location $l_v(i) = (e, \text{offset})$ has the vehicle's current edge $e \in \mathcal{E}$ and a non-negative number offset corresponding to its position at edge e (i.e., the distance, in meters, between $s(e)$

and the vehicle's location at e). Section 8.4.3 describes how we convert GPS-based trajectories, which contain positional errors, into the trajectory model described above. This model extends a vehicle trajectory representation called road-network constrained trajectory. A road-network constrained trajectory $\mathrm{T} = (e_1, e_2, \ldots, e_n)$ is an ordered set of connected road segments $e_i \in \mathcal{E}$ that represents a path traveled by a vehicle.

Two vehicles can communicate when their distance is less than a given threshold (e.g., 300 meters). Let $\mathcal{P}$ denote a set of packets to be delivered, in which each packet $p \in \mathcal{P}$ has a source $s(p) \in V$, destination $d(p)$, creation time slot $t(p)$, expiration time $e(p)$, and content size $s(p)$. The content size $s(p)$ represents the number of time slots required to transfer the packet p. The destination $d(p)$ can be another vehicle, a roadside unit (RSU), or a region. In this chapter, we assume that the destination is a fixed region and consider a packet delivered when a vehicle containing a copy of the packet enters the region before it expires (i.e., the vehicle reaches the region before $t(p) + e(p)$).

The objective of the vehicular network is to improve some routing metrics, such as maximizing the number of packets delivered, minimizing the average delivery delay, and minimizing the communication overhead. The data forwarding strategy, which defines how vehicles exchange data during contacts, is crucial for improving these metrics. However, the problem of finding optimal routing paths is NP-hard even with prior knowledge of future trajectories and network demand [14]. Therefore, in this chapter, we design a data forwarding strategy based on trajectory prediction and routing heuristics to maximize these routing metrics.

8.4 The VDDTP Algorithm

8.4.1 Preliminaries

The VDDTP algorithm uses predicted road-network constrained trajectories to perform routing decisions to improve data delivery ratio and reduce delivery overhead and delay. For that, vehicles exchange their last n traveled road segments with neighbors through beacons. A vehicle computes its last traveled road segments using a trajectory reconstruction framework to convert its recorded GPS points into a road-network constrained trajectory. In this work, we use our trajectory reconstruction framework presented in [46], and n equals 5.

When a vehicle needs to make routing decisions (i.e., when detecting a new neighbor through a beacon or receiving a new data packet), it predicts the future trajectory of

the neighbors and its own, as described in Section 8.4.2. Then, the vehicle converts the predicted road-network constrained trajectories into an extended trajectory model (Section 8.4.3) and uses it to compute packets' delivery probabilities (Section 8.4.4) and perform routing decisions, as described in Section 8.4.5.

8.4.2 Predicting Road-Network Constrained Trajectories

We use our trajectory prediction framework, presented in [45, 50], to predict the future m road segments of a vehicle's trajectory from its last n traveled road segments. The trajectory prediction framework extracts a set of prediction models from clustered historical trajectories. We assume that the vehicles can download these models from a data center using a cellular network or roadside units. We use m equals 20 and the framework's default parameter values in this chapter.

8.4.3 From GPS-based trajectories to extended road-network constrained trajectories

A road-network constrained trajectory contains only an ordered set of road segments, which is enough to compute the likelihood that a vehicle reaches the destination region. However, the VDDTP algorithm also needs information regarding when the vehicle can reach the destination region. Therefore, we calculate a discrete trajectory (the model defined in Section 8.3) from the basic road-network constrained trajectory. In summary, this trajectory contains the coordinates of the vehicle's path through the road segments, extracted at one-second time intervals.

Given a GPS-based trajectory $\mathrm{T_{gps}} = (\rho_1, \rho_2, \ldots, \rho_n)$, we use the map-matched points $m_1, m_2, \ldots, m_n$ and the sequence of road segments $e_1, e_2, \ldots, e_m$ acquired during the trajectory reconstruction process to calculate the discrete trajectory as:

$$T = (m_1, d_1^1, d_2^1, \ldots, d_{x1}^1, m_2, d_1^2, d_2^2, \ldots, d_{x2}^2, \ldots, m_n)$$

where d_i^j are discrete points extracted from the road segments between the map-matched points m_j and m_{j+1}. To extract the locations d_i^j, we assume that the vehicle traveled with constant speed between m_j and m_{j+1}. We can improve the precision of this calculation using live traffic information. For instance, we can use the traffic congestion level from our traffic congestion control protocol [44, 49] in each road segment between m_j and m_{j+1} as a weight to define the vehicle's positions. In this case, the current implementation,

which considers constant speed, is equivalent to an implementation that uses the traffic congestion level equals one (i.e., free flow) for all road segments. This extension is part of future work. Figure 8.1 illustrates the extraction of an extended road-network constrained trajectory from a GPS-based trajectory.

Figure 8.1: The extraction of an extended road-network constrained trajectory from a GPS-based trajectory.

8.4.4 Computing Packets Delivery Probability

A vehicle can deliver a packet p directly or indirectly. Direct delivery is possible if the vehicle's future trajectory intersects the packet's destination region. To verify if a vehicle can deliver a packet directly, we must check if its future trajectory intersects the destination region. On the other side, an indirect delivery is possible if the vehicle's future trajectory intersects another vehicle's future trajectory that can directly or indirectly deliver the packet. As discussed earlier, finding optimal routing paths is an NP-hard problem. Therefore, we propose a heuristic to calculate packet delivery probability.

We divide the packet delivery probability into two metrics: time-based delivery probability and distance-based delivery probability. The time-based delivery probability dp_t is inversely proportional to the time it will take the vehicle to reach the destination region. Likewise, the distance-based delivery probability dp_d is inversely proportional to the minimum distance between the vehicle's future trajectory and the destination region. We use the distance-based delivery probability as a tiebreaker when two vehicles have dp_t equal zero (i.e., both future trajectories do not intersect the destination region). Equations 8.1 and 8.2 calculate the time-based and distance-based delivery probabilities, respectively, where *ttr* is the time it will take to the vehicle v reaches the destination region, and *md* is the minimum distance between vehicle's v future trajectory and the

destination region.

$$\mathrm{dp_t}(v,p) = \begin{cases} 0, & \text{if } ttr > e(p) \vee \\ & ttr = \infty \\ \frac{1}{ttr}, & \text{otherwise} \end{cases} \tag{8.1}$$

$$\mathrm{dp_d}(v,p) = \begin{cases} 1, & \text{if } md \leq 1 \\ \frac{1}{md}, & \text{otherwise} \end{cases} \tag{8.2}$$

8.4.5 Data Forwarding Strategy

The VDDTP's data forwarding strategy defines routing decisions in two situations to optimize the global metrics (i.e., delivery rate, delivery delay, and communication overhead). First, it defines the packets from the vehicle's buffer to transfer during contacts. Second, it defines if a vehicle will rebroadcast a packet when it receives it. Besides that, the vehicle checks its buffer every second to remove expired packets.

Algorithm 11 presents the data forwarding strategy for when vehicle v_a detects a new neighbor v_b through a beacon. In summary, vehicle v_a will try to send (through one-hop broadcast) to vehicle v_b all packets that vehicle v_b is more likely to deliver than vehicle v_a. For that, vehicle v_a iterates through the packets in its buffer (Lines 2–11). For each packet p, it calculates the delivery probabilities $\mathrm{dp_t}(v_a,p)$, $\mathrm{dp_d}(v_a,p)$, $\mathrm{dp_t}(v_b,p)$, and $\mathrm{dp_d}(v_b,p)$ (Line 3). If both v_a and v_b have time-based delivery probabilities equal to zero (i.e., none of the vehicles will reach the packet's destination region $d(p)$), then the vehicle uses distance-based delivery probabilities to make the routing decision (Lines 4–7). Otherwise, vehicle v_a will broadcast packet p if its time-based delivery probability $\mathrm{dp_t}(v_a,p)$ is greater then $\mathrm{dp_t}(v_b,p)$ (Lines 8 and 9).

Algorithm 12 presents the data forwarding strategy for when vehicle v_a receives a new data packet p. It returns true if the vehicle should immediately rebroadcast packet p, and false otherwise. Vehicle v_a will rebroadcast packet p if at least one of its neighbors (besides the vehicle that sent packet p) has a greater delivery probability. First, vehicle v_a returns false if it has already received packet p (Line 1 and 2). Otherwise, it will add the packet in its buffer (Line 4) and iterates through its neighbor (Lines 5–17) to check if any neighbor is more likely to deliver packet p than itself (Lines 6–16). For that, it uses the same strategy of Algorithm 11. Finally, the algorithm returns false if no neighbor is more likely to deliver the packet than v_a (Line 18).

Algorithm 11 VDDTP: Vehicle v_a detects new neighbor v_b

Input: $B(v_a)$: current buffer of vehicle v_a

Output: $\mathcal{P}_{\text{send}}$: resulting set of packets to be broadcast by vehicle v_a

1: $\mathcal{P}_{\text{send}} \leftarrow \emptyset$
2: **for each** $p \in B(v_a)$ **do**
3: $\text{dp}_\text{t}(v_a, p)$, $\text{dp}_\text{d}(v_a, p)$, $\text{dp}_\text{t}(v_b, p)$, $\text{dp}_\text{d}(v_b, p) \leftarrow$ Calculate the time-based and distance-based delivery probabilities by using Equations 8.1 and 8.2
4: **if** $\text{dp}_\text{t}(v_a, p) = 0 \wedge \text{dp}_\text{t}(v_b, p) = 0$ **then**
5: **if** $\text{dp}_\text{d}(v_b, p) \geq \text{dp}_\text{d}(v_a, p)$ **then**
6: $\mathcal{P}_{\text{send}} \leftarrow \mathcal{P}_d \cup p$
7: **end if**
8: **else if** $\text{dp}_\text{t}(v_b, p) \geq \text{dp}_\text{t}(v_a, p)$ **then**
9: $\mathcal{P}_{\text{send}} \leftarrow \mathcal{P}_d \cup p$
10: **end if**
11: **end for**
12: **return** $\mathcal{P}_{\text{send}}$

8.5 Evaluation

This section evaluates the effectiveness and efficiency of the VDDTP algorithm and compares it to some related work. We implemented the proposed algorithm and related work using the C++ programming language and conducted the experiments using the ns-3 network simulator with an 802.11p-compliant MAC layer implementation [162]. Each vehicle from the dataset (Section 8.5.1) generates a data packet containing a payload of 1,000 bytes every 10 seconds. The destination of the data packets is a circular region with a radius varying from 100 to 2,500 meters and a center position chosen randomly among the starting positions of the road network segments. The vehicles transmit beacons through the control channel and data packets through a service channel. Table 8.1 summarizes the default parameters of the experiments.

8.5.1 Dataset

We use the *The Rio Center Dataset for Evaluating Prediction-based Methods for Vehicular Networks* (Chapter 4) in the experiments. As discussed in Section 8.4.3, we assume that vehicles move at a constant speed through the road network to generate the vehicle's

Algorithm 12 VDDTP: Vehicle v_a receives packet p

Input: p: received packet, $N(v_a)$: neighbors of v_a, $B(v_a)$: current buffer of vehicle v_a

Output: Returns true if vehicle v_a should immediately broadcast packet p, false otherwise

1: **if** v_a has previously received packet p **then**
2: **return** false
3: **end if**
4: $B(v_a) \leftarrow B(v_a) \cup p$ ▷ Add the new packet in the vehicle's buffer
5: **for each** $v_i \in N(v_a)$ **do**
6: **if** v_i is the vehicle that transmitted packet p **then**
7: **continue**
8: **end if**
9: $\mathrm{dp_t}(v_a, p)$, $\mathrm{dp_d}(v_a, p)$, $\mathrm{dp_t}(v_i, p)$, $\mathrm{dp_d}(v_i, p)$ $\leftarrow$ Calculate the time-based and distance-based delivery probabilities by using Equations 8.1 and 8.2
10: **if** $\mathrm{dp_t}(v_a, p) = 0 \wedge \mathrm{dp_t}(v_i, p) = 0$ **then**
11: **if** $\mathrm{dp_d}(v_i, p) \geq \mathrm{dp_d}(v_a, p)$ **then**
12: **return** true
13: **end if**
14: **else if** $\mathrm{dp_t}(v_i, p) \geq \mathrm{dp_t}(v_a, p)$ **then**
15: **return** true
16: **end if**
17: **end for**
18: **return** false ▷ If no neighbor is more likely to deliver the packet, the vehicle will not retransmit it

Table 8.1: Default evaluation parameters.

Parameter	Value
Experiment Duration	2 hours
Interval of data packet generation	10 seconds
Data packet size	1,000 bytes
Communication throughput	6 Mbps
Transmission Power	16.0206 dBm
Communication Range	$\approx$ 300 meters

locations between consecutive map-matched points.

8.5.2 Comparison Methods

We compare the efficiency and effectiveness of the VDDTP algorithm to four other solutions: an epidemic algorithm, MOP-RG [31], and two optimum algorithms, VDDTP Optimum and MOP-RG Optimum, based on the VDDTP and MOP-RG, respectively. The epidemic algorithm employs a best-effort policy in which whenever a vehicle detects a new neighbor for the first time, it broadcasts all unexpired data packets in its buffer. Besides that, when a vehicle receives a new data packet for the first time, it stores the packet in its buffer and rebroadcasts it if the packet has more than one neighbor (i.e., there are other vehicles in its vicinity besides the vehicle that transmitted the packet).

The VDDTP Optimum is a modified version of the VDDTP algorithm that uses the vehicle's actual future trajectory instead of predicted trajectories to compute delivery probabilities. This approach is "optimum" because it uses the information of the exact moment when the vehicle will reach the destination region of the packets. However, the VDDTP Optimum algorithm is unrealistic because we do not know the future trajectories of all vehicles. Therefore, we use the VDDTP Optimum algorithm as a baseline algorithm to evaluate the impact of trajectory prediction errors on the efficiency of the VDDTP algorithm.

Besides that, we implemented two versions of the MOP-RG protocol. The original version of the MOP-RG protocol uses GPS points of the previous trajectories of a given vehicle to extract the *Radius of Gyration* (RG) metric that characterizes its mobility. As we do not have previous information regarding individual vehicles, in this evaluation we use the vehicle's current trajectory to calculate the RG. In the first version, called MOP-

RG, we use the vehicle's previous points and the trajectory predicted from our trajectory prediction framework to extract the vehicle's RG. In the second version, called MOP-RG Optimum, we use the vehicle's actual future trajectory instead of the predicted trajectory to evaluate the impact of trajectory prediction errors on the efficiency of the protocol.

We compare the efficiency and effectiveness of these three algorithms using three metrics:

- **Delivery ratio:** the percentage of generated packets that are successfully delivered to their destination area before the expiration time.
- **Overhead ratio:** the number of packets transmitted by vehicles per successfully delivered packet. A lower overhead ratio is essential so that there is no overload in the wireless medium of the vehicular network.
- **Delivery delay:** the time it takes to deliver packets since its generation.

8.5.3 Results

Figure 8.2 shows the delivery ratio of the five algorithms using destination areas of different sizes. The larger the destination area, the more likely the delivery of the packet, as more vehicles will travel close to the area. Therefore, we can see that the delivery ratios of the five algorithms increase as the destination area radius increases. The two versions of our algorithm (i.e., VDDTP and VDDTP Optimum), which use mobility data to assist data delivery, present the best delivery ratios. The delivery ratio of our algorithm is up to 6.5%, 5.5%, and 6.0% higher than the epidemic, MOP-RG, and MOP-RG Optimum, respectively. With the VDDTP Optimum, these differences increase to 8.5%, 7.5%, and 6.7%. This result confirms the observations from previous studies showing that mobility data can help improve data delivery in vehicular networks. The VDDTP Optimum algorithm presents a slightly higher delivery ratio than the VDDTP algorithm (1.6%–2%). This result shows that the VDDTP Optimum computes more accurate delivery probabilities than the VDDTP algorithm, which occurs for two reasons. First, the VDDTP Optimum algorithm has the exact information if the vehicle will reach the destination area and when this will occur. Second, the VDDTP algorithm uses predicted trajectories, which can contain prediction errors, to compute the time the vehicle will reach the destination area. The epidemic algorithm, which uses a best-effort policy to deliver data, has a lower delivery ratio because the vehicles fail to send all the packets in their buffers during contact with other vehicles.

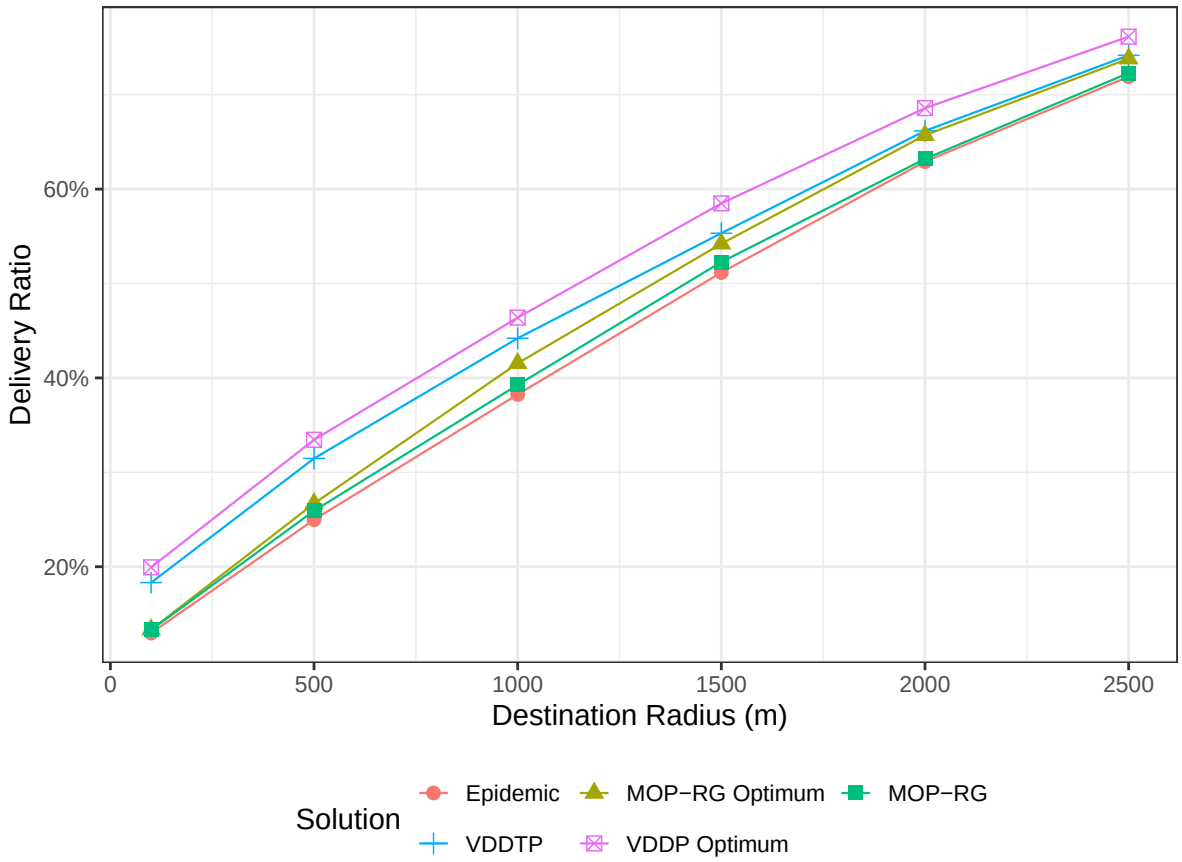


Figure 8.2: The packets delivery ratio.

Figure 8.3 analyzes the communication overhead of the five algorithms. With the epidemic algorithm, vehicles broadcast all packets in their buffers during the first contacts, resulting in a much higher communication overhead than the mobility-aware algorithms. The VDDTP Optimum and the VDDTP algorithm use the same data forwarding strategy, where the only difference is in the computation of packet delivery probabilities. Besides that, the MOP-RG and the MOP-RG Optimum use similar data forwarding strategies. Therefore, these mobility-aware algorithms present similar communication overheads. The most significant differences occur in more complex scenarios (i.e., smaller destination area radius), where the VDDTP Optimum shows an overhead ratio 12% smaller than the VDDTP algorithm, thanks to its more accurate computation of delivery probabilities.

Lastly, we assess the delivery delay of the compared algorithms. Figure 8.4 presents the results. Contrary to the results observed in the other metrics, the epidemic algorithm

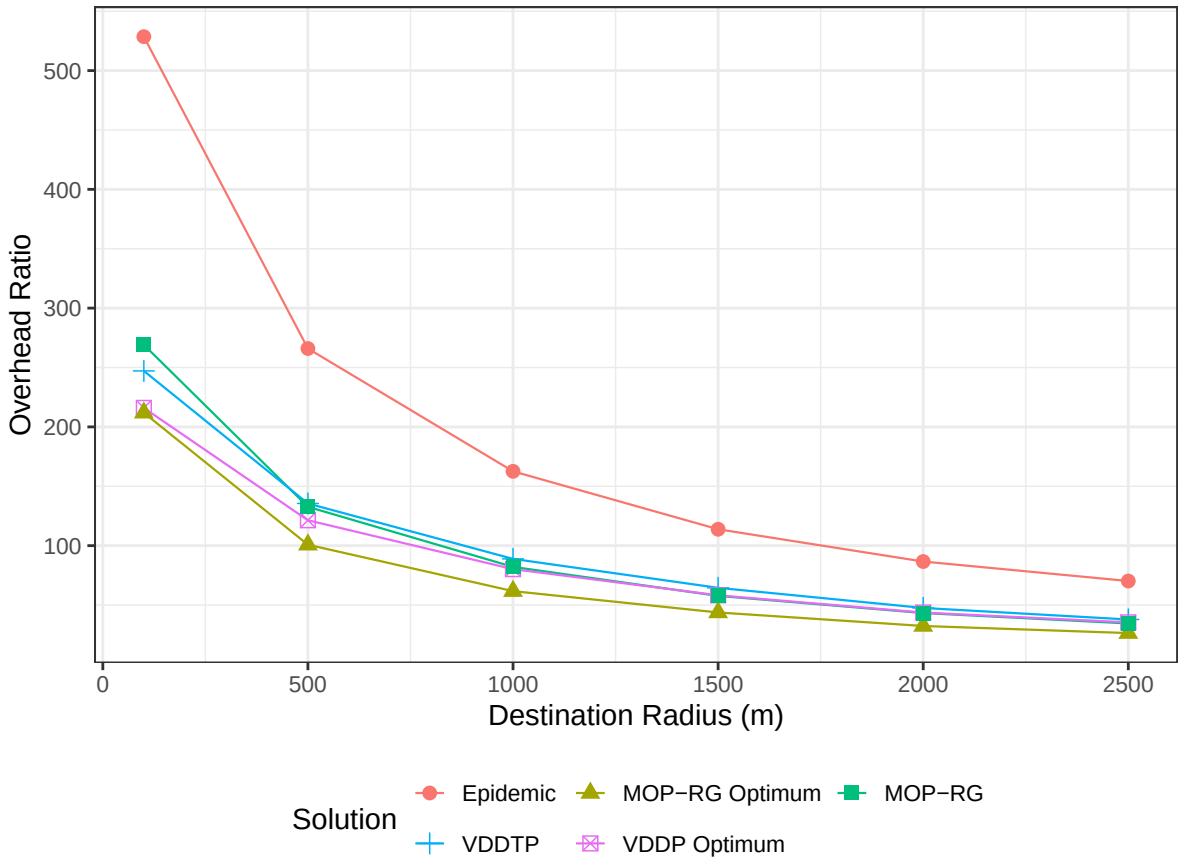


Figure 8.3: The overhead ratio of delivered packets.

shows a better delivery delay than the mobility-aware solutions in most scenarios. This result happens because the main goal of the mobility-aware algorithms is to improve the delivery ratio, while the epidemic algorithm tries to deliver packets as soon as possible by sending all packets during all contacts. Therefore, the mobility-aware algorithms can deliver data packets that are harder to deliver but with a higher delay in some cases, while the epidemic algorithm cannot.

8.6 Chapter Remarks

In this Chapter, we proposed the VDDTP algorithm, a novel Vehicle-assisted Data Delivery algorithm based on Trajectory Prediction. The algorithm employs heuristics based on predicted road-network constrained trajectories to improve the data delivery ratio and reduce communication overhead and delay.

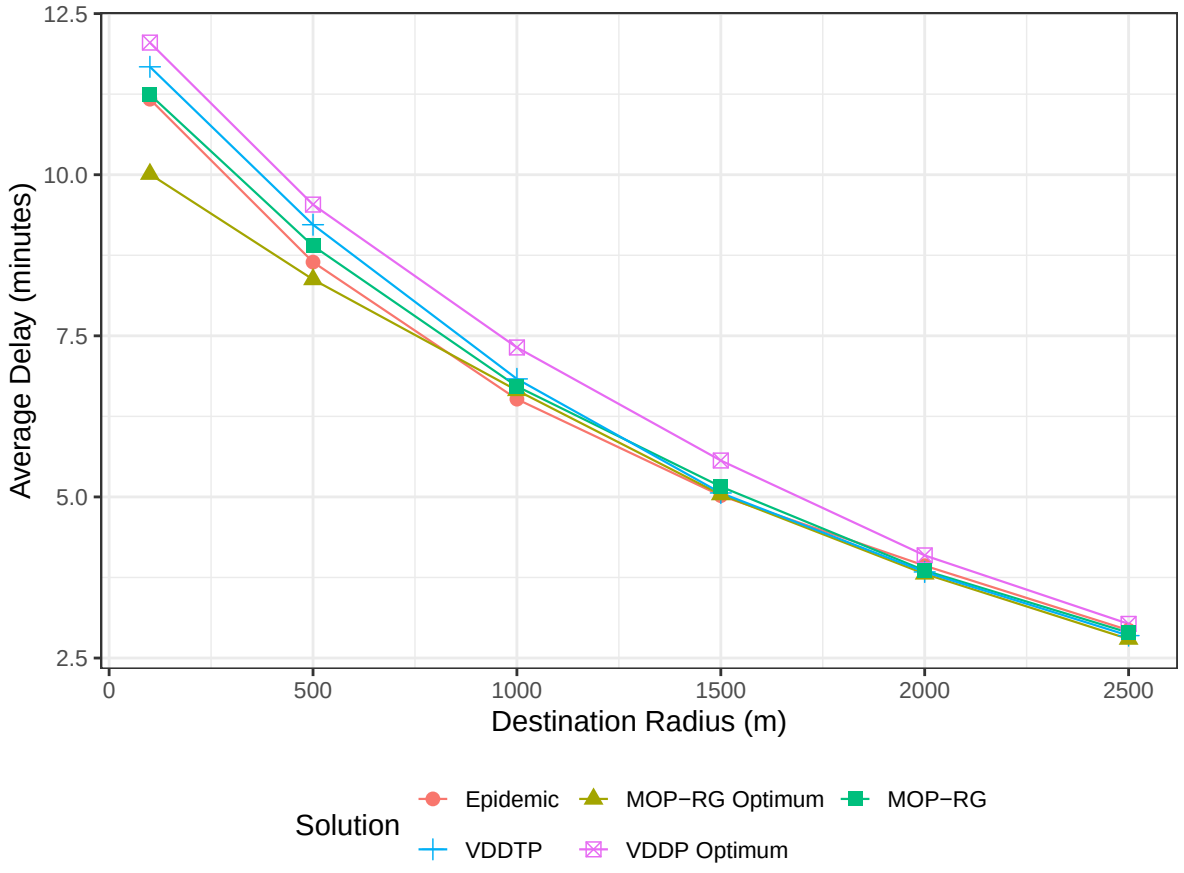


Figure 8.4: The delay in the delivery of packets.

We compared the framework to some related work through experiments using the Rio Center dataset, a novel large-scale, real-world benchmark dataset that supports evaluating applications based on road-network constrained trajectories. The framework outperformed the related work and performed similarly to an "optimum" algorithm that employs the vehicle's actual future trajectory instead of predicted trajectories to compute delivery probabilities.

In future work, we plan to extend the algorithm using live traffic congestion data to improve the computation of packet delivery probabilities. Besides that, we intend to perform experiments using other large-scale datasets and compare the algorithm with techniques that employ different trajectory prediction algorithms.

Chapter 9

Conclusion and Future Work

This Chapter presents the summary of this thesis (Section 9.1) and future research directions (Section 9.2).

9.1 Thesis Summary

This thesis researched vehicle trajectory data mining to design intelligent vehicular networks. Overall, the contribution of this work was twofold. First, we developed data mining techniques to extract mobility knowledge from big trajectory data. Second, we employed mobility knowledge to design algorithms and protocols to improve the vehicular networks' global metrics.

In the first part, we surveyed the fundamental characteristics of vehicle trajectory data, such as data representation and similarity measures. From this, we acknowledged that solving the trajectory reconstruction problem through different preprocessing steps is fundamental to enabling the full potential of big trajectory data. Therefore, we developed a new trajectory reconstruction framework to convert trajectories from large-scale GPS datasets into a more precise data representation, called road-network constrained trajectory.

We designed a novel trajectory prediction framework that uses data enhanced by the reconstruction framework. One of the prediction framework's components is a new hierarchical agglomerative clustering algorithm for road-network constrained trajectories that automatically detects the most appropriate number of clusters. The framework extracts multiple prediction models from the clustered historical data to efficiently predict the following road segments of a vehicle's partial trajectory.

In the second part of this thesis, we demonstrated the application of different data mining techniques to improve data delivery in vehicular networks. For that, we employed the enhanced trajectory data and the prediction framework to design a novel data delivery algorithm for vehicular networks called VDDTP. VDDTP is a vehicle-assisted data delivery algorithm based on trajectory prediction. With VDDTP, we created an extended trajectory model and used predicted road-network constrained trajectories to calculate packet delivery probabilities. Then, we designed a data forwarding strategy to improve the delivery ratio and reduce communication overhead and delivery delay. Furthermore, we also demonstrated the usage of vehicular networks to improve urban mobility in the second part of this thesis by proposing the DisTraC protocol for vehicular networks. DisTraC is a protocol that uses V2V communication to detect, estimate, and reduce traffic congestion.

9.2 Future Research Directions

The future research work also consists of two parts, the first regarding trajectory data mining techniques and the second on applying these techniques to design intelligent vehicular networks.

Data representation is the first challenge for extracting knowledge from big trajectory data. Although a more precise model, road-network constrained trajectories and the approximated extended model used in VDDTP still lack detailed information required in some applications. Another open problem regarding representation is data fusion methods to merge trajectories from different formats, sources, and transportation modes. A promising direction is to design machine learning-based techniques to merge and represent vehicle trajectory data. Consequentially, we will also need to adapt or design new clustering and prediction algorithms for these data representations. Besides that, we plan to investigate other similarity measures for clustering trajectory data, and the impact of it on the prediction framework.

Regarding intelligent vehicular networks, we plan to design and evaluate different heuristics that use predicted trajectories to improve data delivery in vehicular networks. Besides that, we intend to evaluate centralized solutions that take advantage of potential heterogeneous vehicular networks. Another important step we are planning is to modify the VDDTP algorithm to use live traffic data, such as the congestion level provided by the DisTraC protocol, to improve the accuracy of packet delivery probabilities.

Bibliography

[1] Islam Tharwat Abdel-Halim and Hossam Mahmoud Ahmed Fahmy. Prediction-based protocols for vehicular ad hoc networks: Survey and taxonomy. *Computer Networks*, 130:34–50, 2018.

[2] Hatem Abou-zeid, Hossam S. Hassanein, and Ramy Atawia. Towards mobility-aware predictive radio access: Modeling; simulation; and evaluation in lte networks. In *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '14, page 109–116, New York, NY, USA, 2014. Association for Computing Machinery.

[3] Sajimon Abraham and P. Sojan Lal. Trigger based security alarming scheme for moving objects on road networks. In Christopher C. Yang, Hsinchun Chen, Michael Chau, Kuiyu Chang, Sheau-Dong Lang, Patrick S. Chen, Raymond Hsieh, Daniel Zeng, Fei-Yue Wang, Kathleen Carley, Wenji Mao, and Justin Zhan, editors, *Intelligence and Security Informatics*, pages 92–101, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[4] Sajimon Abraham and P. Sojan Lal. Spatio-temporal similarity of network-constrained moving object trajectories using sequence alignment of travel locations. *Transportation Research Part C: Emerging Technologies*, 23:109 – 123, 2012. Data Management in Vehicular Networks.

[5] R. Aissaoui, H. Menouar, A. Dhraief, F. Filali, A. Belghith, and A. Abu-Dayya. Advanced real-time traffic monitoring system based on V2X communications. In *IEEE International Conference on Communications (ICC)*, pages 2713–2718, June 2014.

[6] Noura Aljeri and Azzedine Boukerche. Mobility and handoff management in connected vehicular networks. In *Proceedings of the 16th ACM International Sym-*

posium on Mobility Management and Wireless Access, MobiWac'18, page 82–88, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.

[8] Heba Aly and Moustafa Youssef. semMatch: Road semantics-based accurate map matching for challenging positioning data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, pages 5:1–5:10, New York, NY, USA, 2015. ACM.

[9] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60, June 1999.

[10] G. B. Araújo, M. M. Queiroz, F. d. L. P. Duarte-Figueiredo, A. I. J. Tostes, and A. A. F. Loureiro. CARTIM: A proposal toward identification and minimization of vehicular traffic congestion for vanet. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, June 2014.

[11] H. Arbabi and M. C. Weigle. Monitoring free flow traffic using vehicular networks. In *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 272–2760, Jan 2011.

[12] Kendall Atkinson and R. Modelling a road using spline interpolation, 2002.

[13] Ricardo A. Baeza-Yates, Walter Cunto, Udi Manber, and Sun Wu. Proximity matching using fixed-queries trees. In *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, CPM '94, pages 198–212, Berlin, Heidelberg, 1994. Springer-Verlag.

[14] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. Dtn routing as a resource allocation problem. *SIGCOMM Comput. Commun. Rev.*, 37(4):373–384, aug 2007.

[15] Ya Ban, Rui Wang, Hongjing Liu, Jing Yuan, Hao Luo, Fuli Yang, Ling Yu, and Xinping Xu. A moving objects index method integrating geohash and quadtree. In *2018 International Conference on Computer Modeling, Simulation and Algorithm (CMSA 2018)*, Paris, France, 2018. Atlantis Press, Atlantis Press.

[16] A. Banerjee and R. N. Dave. Validating clusters using the hopkins statistic. In *2004 IEEE International Conference on Fuzzy Systems (IEEE Cat. No.04CH37542)*, volume 1, pages 149–153 vol.1, 2004.

[17] Gustavo E. Batista, Eamonn J. Keogh, Oben Moses Tataw, and Vinícius M. Souza. Cid: An efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28(3):634–669, May 2014.

[18] R. Bauza, J. Gozalvez, and J. Sanchez-Soriano. Road traffic congestion detection through cooperative vehicle-to-vehicle communications. In *IEEE Local Computer Network Conference*, pages 606–612, Oct 2010.

[19] L. Bedogni, M. Fiore, and C. Glacet. Temporal reachability in vehicular networks. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 81–89, Washington, DC, USA, 2018. IEEE Computer Society.

[20] P. Bellavista, L. Foschini, and E. Zamagni. V2X protocols for low-penetration-rate and cooperative traffic estimations. In *IEEE 80th Vehicular Technology Conference (VTC Fall)*, pages 1–6, Sept 2014.

[21] Clara Benevolo, Renata Paola Dameri, and Beatrice D'Auria. Smart mobility in smart city. In Teresina Torre, Alessio Maria Braccini, and Riccardo Spinelli, editors, *Empowering Organizations*, pages 13–28, Cham, 2016. Springer International Publishing.

[22] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013.

[23] P. C. Besse, B. Guillouet, J. Loubes, and F. Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3306–3317, Nov 2016.

[24] Jiang Bian, Dayong Tian, Yuanyan Tang, and Dacheng Tao. A survey on trajectory clustering analysis. *arXiv preprint arXiv:1802.06971*, 1, 2018.

[25] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017.

[26] Ali Bohlooli and Kamal Jamshidi. A gps-free method for vehicle future movement directions prediction using som for vanet. *Applied Intelligence*, 36(3):685–697, Apr 2012.

[27] Tolga Bozkaya and Meral Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. *SIGMOD Rec.*, 26(2):357–368, June 1997.

[28] A. Cappiello, I. Chabini, E. K. Nam, A. Lue, and M. Abou Zeid. A statistical model of vehicle emissions and fuel consumption. In *The IEEE 5th International Conference on Intelligent Transportation Systems*, pages 801–809, 2002.

[29] Néstor Cárdenas-Benítez, Raúl Aquino-Santos, Pedro Magaña-Espinoza, José Aguilar-Velazco, Arthur Edwards-Block, and Aldo Medina Cass. Traffic congestion detection system through connected vehicles and big data. *Sensors*, 16(5), 2016.

[30] C. Celes, F. A. Silva, A. Boukerche, R. M. d. C. Andrade, and A. A. F. Loureiro. Improving vanet simulation with calibrated vehicular mobility traces. *IEEE Transactions on Mobile Computing*, 16(12):3376–3389, Dec 2017.

[31] Clayson Celes, Azzedine Boukerche, and Antonio A. F. Loureiro. Mop: A novel mobility-aware opportunistic routing protocol for connected vehicles. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, 2020.

[32] Clayson Celes, Azzedine Boukerche, and Antonio A. F. Loureiro. Mobility trace analysis for intelligent vehicular networks: Methods, models, and applications. *ACM Comput. Surv.*, 54(3), apr 2021.

[33] B. Chapuis and B. Garbinato. Geodabs: Trajectory indexing meets fingerprinting at scale. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1086–1095, Washington, DC, USA, July 2018. IEEE Computer Society.

[34] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, September 2001.

[35] L. Chen, Y. Gao, X. Li, C. S. Jensen, and G. Chen. Efficient metric indexing for similarity search. In *2015 IEEE 31st International Conference on Data Engineering*, pages 591–602, Washington, DC, USA, April 2015. IEEE Computer Society.

[36] Lei Chen. *Similarity Search over Time Series and Trajectory Data*. PhD thesis, University of Waterloo, Waterloo, Ont., Canada, Canada, 2005. AAINR03008.

[37] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 792–803. VLDB Endowment, 2004.

[38] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 491–502, New York, NY, USA, 2005. ACM.

[39] Lu Chen, Yunjun Gao, Baihua Zheng, Christian S. Jensen, Hanyu Yang, and Keyu Yang. Pivot-based metric indexing. *Proc. VLDB Endow.*, 10(10):1058–1069, June 2017.

[40] Zaiben Chen, Heng Tao Shen, Xiaofang Zhou, Yu Zheng, and Xing Xie. Searching trajectories by locations: An efficiency study. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 255–266, New York, NY, USA, 2010. ACM.

[41] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, and X. S. Shen. Big data driven vehicular networks. *IEEE Network*, 32(6):160–167, November 2018.

[42] Lara Codecá, Raphaël Frank, Sébastien Faye, and Thomas Engel. Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63, 2017.

[43] Felipe D. Cunha, Davidysson A. Alvarenga, Guilherme Maia, Aline C. Viana, Raquel A.F. Mini, and Antonio A.F. Loureiro. Exploring interactions in vehicular networks. In *Proceedings of the 14th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac '16, page 131–138, New York, NY, USA, 2016. Association for Computing Machinery.

[44] R. S. de Sousa, A. Boukerche, and A. A. F. Loureiro. Distrac: A distributed and low-overhead protocol for traffic congestion control using vehicular networks. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6, Washington, DC, USA, July 2019. IEEE Computer Society.

[45] R. S. de Sousa, A. Boukerche, and A. A. F. Loureiro. A cluster-based framework for predicting large scale road-network constrained trajectories. In *2020 ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Network (PE-WASUN)*, pages 1–8, New York, NY, USA, November 2020. Association for Computing Machinery.

[46] R. S. de Sousa, A. Boukerche, and A. A. F. Loureiro. A map matching based framework to reconstruct vehicular trajectories from gps datasets. In *2020 IEEE International Conference on Communications (ICC)*, pages 1–6, Washington, DC, USA, June 2020. IEEE Computer Society.

[47] R. S. de Sousa, A. Boukerche, and A. A. F. Loureiro. Vehicle trajectory similarity: Models, methods, and applications. *ACM Comput. Surv.*, pages 1–33, June 2020.

[48] R. S. de Sousa, F. S. da Costa, A. C. B. Soares, L. F. M. Vieira, and A. A. F. Loureiro. Geo-sdvn: A geocast protocol for software defined vehicular networks. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, Washington, DC, USA, May 2018. IEEE Computer Society.

[49] Roniel S. de Sousa, Azzedine Boukerche, and Antonio A.F. Loureiro. A distributed and low-overhead traffic congestion control protocol for vehicular ad hoc networks. *Computer Communications*, 159:258 – 270, 2020.

[50] Roniel S. de Sousa, Azzedine Boukerche, and Antonio A.F. Loureiro. On the prediction of large-scale road-network constrained trajectories. *Computer Networks*, 206:108337, 2022.

[51] Ke Deng, Kexin Xie, Kevin Zheng, and Xiaofang Zhou. *Trajectory Indexing and Retrieval*, pages 35–60. Springer New York, New York, NY, 2011.

[52] Ze Deng, Yangyang Hu, Mao Zhu, Xiaohui Huang, and Bo Du. A scalable and fast optics for clustering trajectory big data. *Cluster Computing*, 18(2):549–562, June 2015.

[53] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[54] Marko Dogramadzi and Aftab Khan. Accelerated map matching for gps trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 23(5):4593–4602, 2022.

[55] Augusto C. S. A. Domingues, Henrique de Souza Santana, Fabrício A. Silva, Pedro O. S. Vaz de Melo, and Antonio A. F. Loureiro. Are we still friends? evaluating tie persistence in mobility traces. In *Proceedings of the 17th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac '19, page 1–8, New York, NY, USA, 2019. Association for Computing Machinery.

[56] John M. Dow, R. E. Neilan, and C. Rizos. The international gnss service in a changing landscape of global navigation satellite systems. *Journal of Geodesy*, 83(3):191–198, Mar 2009.

[57] Thomas Eiter and Heikki Mannila. Computing discrete fréchet distance. Technical report, Citeseer, 1994.

[58] Mahmoud Hashem Eiza, Yue Cao, and Lexi Xu. *Toward sustainable and economic smart mobility: shaping the future of smart cities*. World Scientific, 2020.

[59] A. Elbery, H. Rakha, M. Elnainay, W. Drira, and F. Filali. Eco-routing using v2i communication: System evaluation. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 71–76, Sept 2015.

[60] T. Emrich, H. Kriegel, N. Mamoulis, M. Renz, and A. Zufle. Querying uncertain spatio-temporal data. In *2012 IEEE 28th International Conference on Data Engineering*, pages 354–365, Washington, DC, USA, April 2012. IEEE Computer Society.

[61] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[62] M. Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22(1):1–72, Dec 1906.

[63] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 816–825, Washington, DC, USA, April 2007. IEEE Computer Society.

[64] Jon Froehlich and John Krumm. Route prediction from trip observations. In *Society of Automotive Engineers (SAE) 2008 World Congress, April 2008*, pages 1–15, April 2008.

[65] Tao-Yang Fu and Wang-Chien Lee. Trembr: Exploring road networks for trajectory representation learning. *ACM Trans. Intell. Syst. Technol.*, 11(1), February 2020.

[66] Xiao Fu, Jiaxu Zhang, and Yue Zhang. An online map matching algorithm based on second-order hidden markov model. *Journal of Advanced Transportation*, 2021, 2021.

[67] M. T. Garip, M. E. Gursoy, P. Reiher, and M. Gerla. Scalable reactive vehicle-to-vehicle congestion avoidance mechanism. In *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 943–948, Jan 2015.

[68] J.R. Geraldo, M.E.M. Campista, and L.H.M.K. Costa. A decentralized traffic monitoring system based on vehicle-to-infrastructure communications. In *IFIP Wireless Days (WD)*, pages 1–6, Nov 2013.

[69] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 776–781, Sep. 2012.

[70] Y. Gong, E. Chen, X. Zhang, L. M. Ni, and J. Zhang. AntMapper: An ant colony-based map matching approach for trajectory-based applications. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):390–401, Feb 2018.

[71] M. Gramaglia, M. Calderon, and C.J. Bernardos. Abeona monitored traffic: Vanet-assisted cooperative traffic congestion forecasting. *IEEE Vehicular Technology Magazine*, 9(2):50–57, June 2014.

[72] Peter D. Grnwald, In Jae Myung, and Mark A. Pitt. *Advances in Minimum Description Length: Theory and Applications (Neural Information Processing)*. The MIT Press, Cambridge, MA, US, 2005.

[73] Matt Grote, Ian Williams, John Preston, and Simon Kemp. Including congestion effects in urban road traffic co2 emissions modelling: Do local government authorities have the right options? *Transportation Research Part D: Transport and Environment*, 43:95 – 106, 2016.

[74] C. Guo, D. Li, G. Zhang, and M. Zhai. Real-time path planning in urban area via vanet-assisted traffic information sharing. *IEEE Transactions on Vehicular Technology*, 67(7):5635–5649, July 2018.

[75] S. Gupte and M. Younis. Vehicular networking for intelligent and autonomous traffic management. In *IEEE International Conference on Communications (ICC)*, pages 5306–5310, June 2012.

[76] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. *SIGMOD Record*, 14(2):47–57, June 1984.

[77] UN Habitat. *State of the world's cities 2012/2013: Prosperity of cities*. Routledge, 2013.

[78] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, Oct 2008.

[79] B. Han, L. Liu, and E. Omiecinski. Road-network aware trajectory clustering: Integrating locality, flow, and density. *IEEE Transactions on Mobile Computing*, 14(2):416–429, Feb 2015.

[80] P. Hao, K. Boriboonsomsin, G. Wu, and M. J. Barth. Modal activity-based stochastic model for estimating vehicle trajectories from sparse mobile sensor data. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):701–711, March 2017.

[81] H. Hartenstein and L. P. Laberteaux. A tutorial survey on vehicular ad hoc networks. *IEEE Communications Magazine*, 46(6):164–171, June 2008.

[82] Heidelberger and Lavenberg. Computer performance evaluation methodology. *IEEE Transactions on Computers*, C-33(12):1195–1220, 1984.

[83] David Hilbert. *Über die stetige Abbildung einer Linie auf ein Flächenstück*, pages 1–2. Springer Berlin Heidelberg, Berlin, Heidelberg, 1935.

[84] Brian Hopkins and J. G. Skellam. A new method for determining the type of distribution of plant individuals. *Annals of Botany*, 18(70):213–227, 1954.

[85] Kathleen Hornsby and Max J. Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1):177–194, Sep 2002.

[86] Weiming Hu, Xi Li, Guodong Tian, Stephen Maybank, and Zhongfei Zhang. An incremental dpmm-based method for trajectory clustering, modeling, and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1051–1065, May 2013.

[87] T. Hunter, P. Abbeel, and A. Bayen. The path inference filter: Model-based low-latency map matching of probe vehicle data. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):507–529, April 2014.

[88] Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. Spatio-temporal similarity analysis between trajectories on road networks. In *Proceedings of the 24th International Conference on Perspectives in Conceptual Modeling*, ER'05, pages 280–289, Berlin, Heidelberg, 2005. Springer-Verlag.

[89] Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. Searching for similar trajectories on road networks using spatio-temporal similarity. In *Proceedings of the 10th East European Conference on Advances in Databases and Information Systems*, ADBIS'06, pages 282–295, Berlin, Heidelberg, 2006. Springer-Verlag.

[90] INRIX. 2018 global traffic scorecard. Technical report, INRIX, Inc., 2018.

[91] SAE International. SAE J2735D: Dedicated Short Range Communications (DSRC) Message Set Dictionary. Standard, Society of Automotive Engineers, March 2011.

[92] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[93] J. Jeong, H. Jeong, E. Lee, T. Oh, and D. H. C. Du. SAINT: Self-adaptive interactive navigation tool for cloud-based vehicular traffic optimization. *IEEE Transactions on Vehicular Technology*, 65(6):4053–4067, June 2016.

[94] H. Jiang, L. Chang, Q. Li, and D. Chen. Trajectory prediction of vehicles based on deep learning. In *2019 4th International Conference on Intelligent Transportation Engineering (ICITE)*, pages 190–195, Washington, DC, USA, Sep. 2019. IEEE Computer Society.

[95] M. Jiang and T. Zhao. Vehicle travel time estimation by sparse trajectories. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMP-*

SAC), volume 1, pages 433–442, Washington, DC, USA, Jul 2019. IEEE Computer Society.

[96] Ruobing Jiang, Yanmin Zhu, Tian He, Yunhuai Liu, and Lionel M. Ni. Exploiting trajectory-based coverage for geocast in vehicular networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(12):3177–3189, 2014.

[97] Maurice G. Kendall. *Rank correlation methods*. Griffin, London, 1948.

[98] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, March 2005.

[99] Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 285–289, New York, NY, USA, 2000. ACM.

[100] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 399–404, Oct 2017.

[101] Jiwon Kim and Hani S. Mahmassani. Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories. *Transportation Research Procedia*, 9:164 – 184, 2015. Papers selected for Poster Sessions at The 21st International Symposium on Transportation and Traffic Theory Kobe, Japan, 5-7 August, 2015.

[102] Christopher R Knittel, Douglas L Miller, and Nicholas J Sanders. Caution, drivers! children present: Traffic, pollution, and infant health. *Review of Economics and Statistics*, 98(2):350–366, 2016.

[103] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3):237–253, Feb 2000.

[104] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[105] John Krumm, Eric Horvitz, and Julie Letchner. Map matching with travel time constraints. Technical report, SAE Technical Paper, 2007.

[106] M. Kubicka, A. Cela, H. Mounier, and S. Niculescu. Comparative study and application-oriented classification of vehicular map-matching methods. *IEEE Intelligent Transportation Systems Magazine*, 10(2):150–166, Summer 2018.

[107] M. Kubička, A. Cela, P. Moulin, H. Mounier, and S. I. Niculescu. Dataset for testing and training of map-matching algorithms. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1088–1093, June 2015.

[108] D. Kumar, J. C. Bezdek, M. Palaniswami, S. Rajasegarar, C. Leckie, and T. C. Havens. A hybrid approach to clustering in big data. *IEEE Transactions on Cybernetics*, 46(10):2372–2385, 2016.

[109] D. Kumar, M. Palaniswami, S. Rajasegarar, C. Leckie, J. C. Bezdek, and T. C. Havens. clusivat: A mixed visual/numerical clustering algorithm for big data. In *2013 IEEE International Conference on Big Data*, pages 112–117, Washington, DC, USA, Oct 2013. IEEE Computer Society.

[110] D. Kumar, H. Wu, S. Rajasegarar, C. Leckie, S. Krishnaswamy, and M. Palaniswami. Fast and scalable big data trajectory clustering for understanding urban mobility. *IEEE Transactions on Intelligent Transportation Systems*, 19(11):3709–3722, Nov 2018.

[111] Dheeraj Kumar, Sutharshan Rajasegarar, Marimuthu Palaniswami, X Wang, and C Leckie. A scalable framework for clustering vehicle trajectories in a dense road network. In *Proc. ACM SIGKDD Int. Workshop Urban Comput.*, New York, NY, USA, 08 2015. ACM.

[112] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 140–149, Washington, DC, USA, 2008. IEEE Computer Society.

[113] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 593–604, New York, NY, USA, 2007. ACM.

[114] SangYoon Lee, Sanghyun Park, Woo-Cheol Kim, and Dongwon Lee. An efficient location encoding method for moving objects using hierarchical administrative district and road network. *Inf. Sci.*, 177(3):832–843, February 2007.

[115] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH Journal*, 1(1):1, 2014.

[116] F. Li and Y. Wang. Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular Technology Magazine*, 2(2):12–22, June 2007.

[117] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 617–628, Washington, DC, USA, April 2018. IEEE Computer Society.

[118] Xiucheng Li, Gao Cong, Aixin Sun, and Yun Cheng. Learning travel time distributions with deep generative model. In *The World Wide Web Conference*, WWW '19, page 1017–1027, New York, NY, USA, 2019. Association for Computing Machinery.

[119] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18, page 1695–1704, New York, NY, USA, 2018. Association for Computing Machinery.

[120] Siyuan Liu, Ce Liu, Qiong Luo, Lionel M. Ni, and Ramayya Krishnan. Calibrating large scale vehicle trajectory data. In *Proceedings of the 2012 IEEE 13th International Conference on Mobile Data Management (Mdm 2012)*, MDM '12, pages 222–231, Washington, DC, USA, 2012. IEEE Computer Society.

[121] Wei Liu, Yoshito Watanabe, and Yozo Shoji. Vehicle-assisted data delivery in smart city: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 69(11):13849–13860, 2020.

[122] X. Liu, K. Liu, M. Li, and F. Lu. A ST-CRF map-matching method for low-frequency floating car data. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1241–1254, May 2017.

[123] T.J. Lomax, Texas Transportation Institute, and National Research Council (U.S.). Transportation Research Board. *Quantifying Congestion*. Number Nº 398;Nº 1465 in Construction Research. National Academy Press, 1997.

[124] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 352–361, New York, NY, USA, 2009. ACM.

[125] Qiang Lu, Rencai Wang, Bin Yang, and Zhiguang Wang. Trajectory Splicing. *Knowledge and Information Systems*, pages 1–34, 2019.

[126] J. Lv, Q. Li, Q. Sun, and X. Wang. T-conv: A convolutional neural network for multi-scale taxi trajectory prediction. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 82–89, Washington, DC, USA, Jan 2018. IEEE Computer Society.

[127] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6120–6127, 07 2019.

[128] N. Magdy, M. A. Sakr, T. Mostafa, and K. El-Bahnasy. Review on trajectory similarity measures. In *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 613–619, Cairo, Egypt, Dec 2015. IEEE.

[129] Yingchi Mao, Haishi Zhong, Xianjian Xiao, and Xiaofang Li. A segment-based trajectory similarity measure in the urban transportation systems. *Sensors*, 17(3), 2017.

[130] Pierre-François Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):306–318, February 2009.

[131] Giuseppe Martuscelli, Azzedine Boukerche, Luca Foschini, and Paolo Bellavista. V2V protocols for traffic congestion discovery along routes of interest in vanets: a quantitative study. *Wireless Communications and Mobile Computing*, 16(17):2907–2923, 2016.

[132] Marina Meilă. The uniqueness of a good optimum for k-means. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 625–632, New York, NY, USA, 2006. ACM.

[133] Fanrong Meng, Guan Yuan, Shaoqian Lv, Zhixiao Wang, and Shixiong Xia. An overview on trajectory outlier detection. *Artificial Intelligence Review*, Feb 2018.

[134] Luisa Micó, Jose Oncina, and Rafael C. Carrasco. A fast branch & bound nearest neighbour classifier in metric spaces. *Pattern Recogn. Lett.*, 17(7):731–739, June 1996.

[135] M. Milojevic and V. Rakocevic. Distributed road traffic congestion quantification using cooperative VANETs. In *13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pages 203–210, June 2014.

[136] Vaishali Mirge, Kesari Verma, and Shubhrata Gupta. Outlier detection in vehicle trajectories. *International Journal of Computer Applications*, 171:1–6, 08 2017.

[137] Boris Mirkin. *Clustering: a data recovery approach*. Chapman and Hall/CRC, Boca Raton, FL, USA, 2016.

[138] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.*, 27(3):267–289, November 2006.

[139] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, March 2001.

[140] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 336–343, New York, NY, USA, 2009. ACM.

[141] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, 1999.

[142] Johannes Niedermayer, Andreas Züfle, Tobias Emrich, Matthias Renz, Nikos Mamoulis, Lei Chen, and Hans-Peter Kriegel. Probabilistic nearest neighbor

queries on uncertain moving object trajectories. *Proceedings of the VLDB Endowment*, 7(3):205–216, November 2013.

[143] Gustavo Niemeyer. Geohash. *Retrieved June*, 6:2018, 2008.

[144] H. Noori and M. Valkama. Impact of VANET-based V2X communication using ieee 802.11p on reducing vehicles traveling time in realistic large scale urban area. In *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 654–661, Dec 2013.

[145] David Novak, Michal Batko, and Pavel Zezula. Metric index: An efficient and scalable solution for precise and approximate similarity search. *Information Systems*, 36(4):721 – 733, 2011. Selected Papers from the 2nd International Workshop on Similarity Search and Applications SISAP 2009.

[146] K. Okamoto, K. Berntorp, and S. Di Cairano. Similarity-based vehicle-motion prediction. In *2017 American Control Conference (ACC)*, pages 303–308, Washington, DC, USA, May 2017. IEEE Computer Society.

[147] K. Okamoto, K. Berntorp, and S. Di Cairano. Similarity-based vehicle-motion prediction. In *2017 American Control Conference (ACC)*, pages 303–308, May 2017.

[148] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017.

[149] N. Padhariya, O. Wolfson, A. Mondal, V. Gandhi, and S. K. Madria. E-vet: Economic reward/penalty-based system for vehicular traffic management. In *2014 IEEE 15th International Conference on Mobile Data Management*, volume 1, pages 99–102, July 2014.

[150] J. (. Pan, I. S. Popa, and C. Borcea. DIVERT: A distributed vehicular traffic rerouting system for congestion avoidance. *IEEE Transactions on Mobile Computing*, 16(1):58–72, Jan 2017.

[151] Mostofa Ali Patwary, Diana Palsetia, Ankit Agrawal, Wei-keng Liao, Fredrik Manne, and Alok Choudhary. Scalable parallel optics data clustering using graph algorithmic techniques. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 49:1–49:12, New York, NY, USA, 2013. ACM.

[152] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady Andrienko, and Yannis Theodoridis. Similarity search in trajectory databases. In *Proceedings of the 14th International Symposium on Temporal Representation and Reasoning*, TIME '07, pages 129–140, Washington, DC, USA, 2007. IEEE Computer Society.

[153] Dieter Pfoser and Christian S. Jensen. Capturing the uncertainty of moving-object representations. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, SSD '99, pages 111–132, London, UK, UK, 1999. Springer-Verlag.

[154] Dieter Pfoser and Christian S. Jensen. Indexing of network constrained moving objects. In *Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems*, GIS '03, pages 25–32, New York, NY, USA, 2003. ACM.

[155] O. Pink and B. Hummel. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 862–867, Oct 2008.

[156] M Purss, R Gibb, F Samavati, P Peterson, J Rogers, J Ben, and C Dow. Topic 21: Discrete global grid systems abstract specification. Technical report, Open Geospatial Consortium, Wayland, MA, USA, 2017.

[157] Shaojie Qiao, Changjie Tang, Huidong Jin, Teng Long, Shucheng Dai, Yungchang Ku, and Michael Chau. Putmode: prediction of uncertain trajectories in moving objects databases. *Applied Intelligence*, 33(3):370–386, Dec 2010.

[158] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 262–270, New York, NY, USA, 2012. ACM.

[159] S. Ranu, Deepak P, A. D. Telang, P. Deshpande, and S. Raghavan. Indexing and matching trajectories under inconsistent sampling rates. In *2015 IEEE 31st*

International Conference on Data Engineering, pages 999–1010, Washington, DC, USA, April 2015. IEEE Computer Society.

[160] Mohan Rao and K Ramachandra Rao. Measuring urban traffic congestion – a review. *International Journal for Traffic and Transport Engineering*, 2:286–305, 12 2012.

[161] Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, Marimuthu Palaniswami, and James C. Bezdek. A scalable framework for trajectory prediction. *CoRR*, abs/1806.03582, 2018.

[162] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[163] Joseph Lee Rodgers and W. Alan Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[164] E V Ruiz. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recogn. Lett.*, 4(3):145–157, July 1986.

[165] Guillermo Ruiz, Francisco Santoyo, Edgar Chávez, Karina Figueroa, and Eric Sadit Tellez. Extreme pivots for faster metric indexes. In Nieves Brisaboa, Oscar Pedreira, and Pavel Zezula, editors, *Similarity Search and Applications*, pages 115–126, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[166] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, October 2007.

[167] I. Sanchez, Z. M. M. Aye, B. I. P. Rubinstein, and K. Ramamohanarao. Fast trajectory clustering using hashing methods. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3689–3696, Washington, DC, USA, July 2016. IEEE Computer Society.

[168] Swaminathan Sankararaman, Pankaj K. Agarwal, Thomas Mølhave, Jiangwei Pan, and Arnold P. Boedihardjo. Model-driven matching and segmentation of trajectories. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 234–243, New York, NY, USA, 2013. ACM.

[169] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 76–85, New York, NY, USA, 2003. ACM.

[170] M. Schreier, V. Willert, and J. Adamy. Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 334–341, Oct 2014.

[171] Shuo Shang, Lisi Chen, Zhewei Wei, Christian S. Jensen, Kai Zheng, and Panos Kalnis. Parallel trajectory similarity joins in spatial networks. *The VLDB Journal*, 27(3):395–420, June 2018.

[172] Fabrício A. Silva, Clayson Celes, Azzedine Boukerche, Linnyer B. Ruiz, and Antonio A.F. Loureiro. Filling the gaps of vehicular mobility traces. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '15, page 47–54, New York, NY, USA, 2015. Association for Computing Machinery.

[173] Tomás Skopal, Jaroslav Pokornỳ, and Vaclav Snasel. Pm-tree: Pivoting metric tree for similarity search in multimedia databases. In *ADBIS (Local Proceedings)*, Berlin, Heidelberg, 2004. Springer-Verlag.

[174] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011.

[175] Christoph Sommer, Robert Krul, Reinhard German, and Falko Dressler. Emissions vs. Travel Time: Simulative Evaluation of the Environmental Impact of ITS. In *71st IEEE Vehicular Technology Conference (VTC2010-Spring)*, pages 1–5, Taipei, Taiwan, May 2010. IEEE.

[176] Han Su, Kai Zheng, Jiamin Huang, Haozhou Wang, and Xiaofang Zhou. Calibrating trajectory data for spatio-temporal similarity analysis. *The VLDB Journal*, 24(1):93–116, February 2015.

[177] ETSI Technical Committee Intelligent Transport Systems. ETSI TS 102 637-2: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Ap-

plications; Part 2: Specification of Cooperative Awareness Basic Service. Standard, European Telecommunications Standards Institute, March 2011.

[178] Na Ta, Guoliang Li, Yongqing Xie, Changqi Li, Shuang Hao, and Jianhua Feng. Signature-based trajectory similarity join. *IEEE Transactions on Knowledge and Data Engineering*, 29(4):870–883, April 2017.

[179] Pang-Ning Tan. *Introduction to data mining*. Pearson Education India, 2018.

[180] E. Tiakas, A. N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. Searching for similar trajectories in spatial networks. *Journal of Systems and Software*, 82(5):772–788, May 2009.

[181] Dalia Tiesyte and Christian S. Jensen. Similarity-based prediction of travel times for vehicles traveling on known routes. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, pages 14:1–14:10, New York, NY, USA, 2008. ACM.

[182] V. S. Tiwari, S. Chaturvedi, and A. Arya. Route prediction using trip observations and map matching. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 583–587, Feb 2013.

[183] Kevin Toohey and Matt Duckham. Trajectory similarity measures. *SIGSPATIAL Special*, 7(1):43–50, May 2015.

[184] Caetano Traina, Jr., Roberto F. Filho, Agma J. Traina, Marcos R. Vieira, Christos Faloutsos, and Christos Faloutsos. The omni-family of all-purpose access methods: A simple and effective way to make similarity search more efficient. *The VLDB Journal*, 16(4):483–505, October 2007.

[185] Goce Trajcevski. *Uncertainty in Spatial Trajectories*, pages 63–107. Springer New York, New York, NY, 2011.

[186] Goce Trajcevski, Alok Choudhary, Ouri Wolfson, Li Ye, and Gang Li. Uncertain range queries for necklaces. In *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, MDM '10, pages 199–208, Washington, DC, USA, 2010. IEEE Computer Society.

[187] Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29(3):463–507, September 2004.

[188] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2):153–167, 2002.

[189] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[190] Michail Vlachos, Dimitrios Gunopoulos, and George Kollios. Discovering similar multidimensional trajectories. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 673–, Washington, DC, USA, 2002. IEEE Computer Society.

[191] Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. An effectiveness study on trajectory similarity measures. In *Proceedings of the Twenty-Fourth Australasian Database Conference - Volume 137*, ADC '13, pages 13–22, Darlinghurst, Australia, Australia, 2013. Australian Computer Society, Inc.

[192] Hua Wang, Changlong Gu, and Washington Yotto Ochieng. Vehicle trajectory reconstruction for signalized intersections with low-frequency floating car data. *Journal of Advanced Transportation*, 2019, 2019.

[193] M. Wang, H. Shan, R. Lu, R. Zhang, X. Shen, and F. Bai. Real-time path planning based on hybrid-vanet-enhanced transportation system. *IEEE Transactions on Vehicular Technology*, 64(5):1664–1678, May 2015.

[194] L. Wischoff, A. Ebner, H. Rohling, and R. Halfmann. SOTIS - a self-organizing traffic information system. In *The 57th IEEE Semiannual Vehicular Technology Conference (VTC 2003-Spring)*, volume 1, pages 2442 – 2446, 2003.

[195] N. Wisitpongphan, O. K. Tonguz, J. S. Parikh, P. Mudalige, F. Bai, and V. Sadekar. Broadcast storm mitigation techniques in vehicular ad hoc networks. *IEEE Wireless Communications*, 14(6):84–94, December 2007.

[196] J. Won, S. Kim, J. Baek, and J. Lee. Trajectory clustering in road network environment. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 299–305, Nashville, TN, USA, March 2009. IEEE Computer Society.

[197] Hao Wu, Weiwei Sun, and Baihua Zheng. A fast trajectory outlier detection approach via driving behavior modeling. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 837–846, New York, NY, USA, 2017. ACM.

[198] M. Wu, X. Zeng, Y. Lin, and Y. Wen. Vehicle trajectory prediction models by combining communication data. In *2019 Eleventh International Conference on Advanced Computational Intelligence (ICACI)*, pages 113–117, Washington, DC, USA, June 2019. IEEE Computer Society.

[199] Yuchen Wu, Yanmin Zhu, and Bo Li. Trajectory improves data delivery in vehicular networks. In *2011 Proceedings IEEE INFOCOM*, pages 2183–2191, 2011.

[200] Ying Xia, Guoyin Wang, Xu Zhang, Gyoung Bae Kim, and Hae-Young Bae. Spatio-temporal similarity measure for network constrained trajectory data. *Int. J. Comput. Intell. Syst.*, 4:1070–1079, 2011.

[201] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang. Vehicle trajectory prediction by integrating physics- and maneuver-based approaches using interactive multiple models. *IEEE Transactions on Industrial Electronics*, 65(7):5999–6008, July 2018.

[202] D. Yao, G. Cong, C. Zhang, and J. Bi. Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1358–1369, Washington, DC, USA, April 2019. IEEE Computer Society.

[203] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi. Trajectory clustering via deep representation learning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3880–3887, Washington, DC, USA, May 2017. IEEE Computer Society.

[204] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Time series joins, motifs, discords and shapelets: A unifying view

that exploits the matrix profile. *Data Min. Knowl. Discov.*, 32(1):83–123, January 2018.

[205] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 311–321, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[206] Yifang Yin, Rajiv Ratn Shah, Guanfeng Wang, and Roger Zimmermann. Feature-based map matching for low-sampling-rate GPS trajectories. *ACM Trans. Spatial Algorithms Syst.*, 4(2):4:1–4:24, August 2018.

[207] M. B. Younes and A. Boukerche. Efficient traffic congestion detection protocol for next generation vanets. In *2013 IEEE International Conference on Communications (ICC)*, pages 3764–3768, June 2013.

[208] Maram Bani Younes and Azzedine Boukerche. Traffic efficiency applications over downtown roads: A new challenge for intelligent connected vehicles. *ACM Comput. Surv.*, 53(5):1–30, Sep 2020.

[209] Ge Yu, Yu Gu, Jianzhong Qiao, Lei Chen, and Chuanfei Xu. Interval reverse nearest neighbor queries on uncertain data with markov correlations. In *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013)*, ICDE '13, pages 170–181, Washington, DC, USA, 2013. IEEE Computer Society.

[210] Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. A review of moving object trajectory clustering algorithms. *Artif. Intell. Rev.*, 47(1):123–144, January 2017.

[211] H. Yuan and G. Li. Distributed in-memory trajectory similarity search and join on road network. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1262–1273, Washington, DC, USA, April 2019. IEEE Computer Society.

[212] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G. Sun. An interactive-voting based map matching algorithm. In *2010 Eleventh International Conference on Mobile Data Management*, pages 43–52, May 2010.

[213] Zhenxia Zhang, Azzedine Boukerche, and Richard Pazzi. A novel multi-hop clustering scheme for vehicular ad-hoc networks. In *Proceedings of the 9th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac '11, pages 19–26, New York, NY, USA, 2011. ACM.

[214] Jing Zhao and Guohong Cao. VADD: Vehicle-assisted data delivery in vehicular ¡emphasis emphasistype="italic"¿ad hoc¡/emphasis¿ networks. *IEEE Transactions on Vehicular Technology*, 57(3):1910–1922, 2008.

[215] Yu Zheng. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.*, 6(3):29:1–29:41, May 2015.