UNIVERSITY OF PIRAEUS

DEPARTMENT OF INFORMATICS

Data Warehousing & Mining Techniques for Moving Object Databases

PhD Thesis

GERASIMOS D. MARKETOS

Degree in Informatics, University of Piraeus (2003)
MSc in Information Systems Engineering, UMIST (2004)

Piraeus, December 2009

UNIVERSITY OF PIRAEUS

Advisory Committee:

Supervisor:

Yannis Theodoridis
Assoc. Professor U. Piraeus

Members:

Georgios Vassilakopoulos
Professor U. Piraeus

Dimitrios Despotis
Professor U. Piraeus

Thesis
submitted for the degree of
Doctor of Philosophy
at the Department of Informatics,
University of Piraeus

GERASIMOS MARKETOS

"Data Warehousing & Mining Techniques for Moving Object Databases"

Examination Committee:

Yannis Manolopoulos
Professor Aristotle U. Thessaloniki

Themistoklis Panayiotopoulos
Professor U. Piraeus

Yannis Kotidis
Asst. Professor Athens U. of Economics & Business

Despina Polemi
Lecturer U. Piraeus

.....
GERASIMOS D. MARKETOS

Preface

Analyzing mobility data that are collected from location aware devices enables us to discover behavioral patterns that can be explored in applications like service accessibility, mobile marketing and traffic management. Online analytical processing (OLAP) and data mining (DM) techniques can be employed in order to convert this vast amount of raw data into useful knowledge. Their application on conventional data has been extensively studied during the last decade. The high volume of generated mobility data arises the challenge of applying analytical techniques on such data. In order to achieve this aim, we have to take into consideration the complex nature of spatiotemporal data and thus to extend appropriately the two aforementioned techniques to handle them in an efficient way. This thesis proposes a framework for Mobility Data Warehousing and Mining which consists of various components (actually, Knowledge Discovery & Delivery steps). More specifically, Trajectory Data Warehousing techniques are addressed focusing on modeling issues, ETL processes (trajectory reconstruction, data cube loading) and OLAP operations (aggregation etc). Moreover, we propose data mining techniques that explore mobility data and extract a) interaction patterns for spatiotemporal representation, synthesis and classification and b) traffic patterns that can provide useful insights regarding the traffic flow on a road network.

Gerasimos Marketos

December 2009

Acknowledgment

I would like to express my deepest gratitude to my supervisor Assoc. Prof. Yannis Theodoridis for his support and guidance in fulfilling this thesis. Our discussions, his ideas and suggestions have played a major role for the completion of this thesis. I would also like to thank Prof. Georgios Vassilakopoulos and Prof. Dimitrios Despotis for being members of my advisory committee. Moreover, I would like to thank professors Yannis Manolopoulos, Themistoklis Panayiotopoulos, Yannis Kotidis and Despina Polemi for accepting to be members of my examination committee.

I would like to thank my colleagues in Information Systems Laboratory for the cooperation we had all these years. Many of the ideas presented in this thesis arised from discussions with them.

Special thanks go to my parents and my brother for encouraging and supporting me all these years. Finally, I would like to thank Georgia for her endless patience during the fulfillment of this thesis.

Table of Contents

List of Tables

List of Figures

1. Introduction

This chapter highlights the background of the thesis and outlines its structure. In Section 1.1 we introduce some basic knowledge about mobility data analysis; we motivate the thesis and refer to interesting application scenarios. Section 1.2 sketches the contributions of this thesis and Section 1.3 outlines the rest of the thesis.

1.1. Analyzing Mobility Data

Our everyday actions, as expressed by the way we live and move, leave digital traces in information systems. This happens because we use mobile phones and other location aware devices that allow us to communicate and get routing instructions. Actually, through these traces we can sense the human movements in a territory and thus their potential value is really high. It's becoming even higher because of the increasing volume, pervasiveness and positioning accuracy of these traces.

The number of mobile phone users worldwide was more that 2 billion in 2005 i.e., one mobile phone every three people [Cia09]. Furthermore, location technologies, such as GSM and UMTS, currently used by wireless phone operators are capable of providing a better estimation of the device's position, while the integration of various positioning technologies proceeds [GPT08]: GPS-equipped mobile devices can transmit their location information to some service provider. Moreover, latest advances like Wi-Fi and Bluetooth devices are becoming a source of data for indoor positioning, while Wi-Max can become an alternative for outdoor positioning [GP07].

Modern communication and computing devices are ubiquitous and carried everywhere and always by people and vehicles. The consequence is that human activity in a territory may be sensed – not necessarily on purpose, but simply as a side effect of the services provided to mobile users. This fact allows considering the wireless phone network as an infrastructure to gather mobility data so as to analyze them and gain insights about human movements. Clearly, in these scenarios privacy is a concern [GP07]. In particular, how can trajectories of mobile individuals be stored and analyzed without infringing personal privacy rights and expectations? How can, out of privacy-sensitive trajectory data, patterns be extracted that are demonstrably privacy-preserving, i.e., patterns that do not disclose individuals' sensitive information?

The usage of location aware devices, such as mobile phones and GPS-enabled devices allows access to large spatiotemporal datasets. The space-time nature of this kind of data results in the generation of huge amounts of spatiotemporal data and imposes new challenges regarding the analytical tools that

can be used for transforming raw data to knowledge. In this thesis, we investigate the extension of traditional analytical techniques so as to be applicable on mobility data.

The analysis of such mobility data raises opportunities for discovering behavioral patterns that can be exploited in applications like mobile marketing, traffic management etc. Online analytical processing (OLAP) and data mining (DM) techniques can be employed in order to convert this vast amount of raw data into useful knowledge. Their application on conventional data has been extensively studied during the last decade. The high volume of generated mobility data arises the challenge of applying analytical techniques on such data. In order to achieve this aim, we have to take into consideration the complex nature of spatiotemporal data and thus to extend appropriately the two aforementioned techniques to handle them in an efficient way.

1.1.1. Motivation Scenarios

The research efforts that are presented in this thesis are motivated by some scenarios from the *human movement activity* and the *transportation management* domains and are inspired by the research work done in GeoPKDD project [Geo06]. Their different characteristics allow us to show the usefulness of our work. The former application scenario regards an educational mobile game in a city and the latter a service for managing the traffic situation in city networks in an effective way.

As for the first domain, let us consider an advertising company which is interested in analyzing mobility data in different areas of a city so as to decide upon road advertisements (placed on panels on the roads). They are interested in analyzing the demographical profiles of people visiting different urban areas of the city at different time zones of the day so as to decide about the proper sequence of advertisements that will appear on the panels at different time periods. This knowledge will enable them to execute more focused marketing campaigns and apply a more effective strategy.

Another interesting application of this domain could be the *recreational planning* scenario which can be shown through a game. A typical educational mobile game is the paper chase (also known as scavenger hunt), an old children's game in which an area is explored by means of a set of questions and hints on a sheet of paper. Each team tries to find the locations that are indicated on the questionnaire. Once a place is found, they try to best and quickly answer the questions on the sheet, note down the answer, and proceed to the next point of interest (POI). The game mainly consists of a set of geo-referenced checkpoints associated with multimedia riddles.

The player's device includes a GPS receiver which continuously tracks the current position of the device. As the checkpoints are proximity-aware, an event is raised, and the server is contacted whenever a player physically approaches one of the virtual checkpoints. The server responds by sending the information about the corresponding riddle, which is presented in a hypermedia presentation to the user.

Each riddle has associated resources like an image or other additional (media) information that are needed to solve the riddle with the respective interaction(s). The player tries to solve the riddle not only correctly but also as quick as possible, because the time needed to solve all the riddles is accumulated and added to the overall score. The answer to the riddle is communicated to the game server. It is

possible for the player to interact with the system but also with other people, e.g., by asking them for help with the riddles.

Two main application requirements have been selected based on the general lack of knowledge of recreation planners about the actual behavioral patterns of players within a recreation site, accordingly to:

- *Points of Interest* in a recreational site: The assumption is that game activities are not always performed in planned recreational zones.
- *Similar players' interactions* in a recreational site: The assumption is that the behavior of groups of players partly depends on interaction with others. For example we may identify that some groups showing different behavior might interfere with each other (e.g. teams that might start competing against each other).

In particular, the recreation planners need to:

- understand the actual trajectories followed by players, not only the starting and ending points;
- estimate the flow between points of interest;
- understand the interactions and movement of players and foreseen possible new riddles;
- identify possible players interactions and how the players respond when some unexpected event happens;
- discover alternative places for the recreational game;
- discover dangerous and suspicious behavior in the recreational game.

As for the *transportation management* scenario, trying to understand, manage and predict the traffic phenomenon in a city is both interesting and useful. For instance, city authorities, by studying the traffic flow, would be able to improve traffic conditions, react effectively in case of some traffic problems, and arrange the construction of new roads, the extension of existing ones, and the placement of traffic lights. Furthermore, studying the interactions between objects in spatiotemporal neighborhoods is essential so as to discover useful knowledge about moving behavior.

An interesting final goal would be the development of a decision support tool that aims to help in managing traffic, analyzing past movements and behaviors of people through data coming from their mobile devices. This service is addressed to urban planning departments, mobility agencies, traffic managers, but also the public administrations that are responsible for mobility both at urban and regional scale. The needs that this service aims to satisfy are:

- to identify and observe the user flow variations in geographical areas according to urban changes in different time periods;
- to know the real average time to move between different areas;
- to identify the most popular aggregated trajectories followed by people.

The input data for this kind of service can be very diverse, and according to the type of data at disposal, different functionalities can be implemented. In this case, both GSM and GPS data can be used, along with data already at disposal coming from traffic cameras and traffic sensors.

The functionality that such a tool would implement includes:

- automatic building and construction of origin destination matrix, in order to estimate the traffic flow from one region of interest to another both at urban and regional scale;
- calculation of the average travel time from one zone to another, both at regional and urban scale;
- simulation of traffic flow in presence of extraordinary events, such as football matches, strikes or big concerts.

The above motivation scenarios can be realized using novel analytical techniques that will be able to exploit the available rich mobility semantics. Indicatively, a Trajectory Data Warehouse (TDW) can help us to analyze various measures such as the number of moving objects (people, vehicles) in different urban areas, the average speed of vehicles (or people), the ups and downs of vehicles' speed as well as useful insights, like discovering popular movements. Moreover, trajectory-inspired mining techniques can be used to discover traffic related patterns. These patterns can be expressed through relationships among the road segments of the city network. In other words, we aim to discover, by using aggregated mobility data, how the traffic flows in this network, the road segments that contribute to the flow and how this happens. Finally, representing movement as a set of descriptors gives us insight about the behavior of objects as well as their interactions with the neighbors.

1.2. Thesis Contribution

This thesis proposes innovative analytical techniques aiming to extract useful patterns from spatiotemporal data. It discusses the difference of two types of spatiotemporal data: mobility and immobility data. In order to clarify this distinction, we choose seismological data as a typical case of immobile data and we present a *Seismic Data Management and Mining System* for quick and easy data collection, processing, and visualization. The proposed framework includes, among others, a seismological database for efficient and effective querying and a seismological data warehouse for OLAP analysis and data mining. We provide template schemes for these two components as well as examples of their functionality towards the support of decision making. Our results in this topic are presented in Chapter 2. Preliminary results have been already published in [MTK08].

The main part of our research focused on data warehousing and mining techniques that can be applied on Moving Object Databases. More specifically, this thesis proposes a complete framework for Mobility Data Warehousing and Mining (MDWM) which consists of various components (actually, Knowledge Discovery & Delivery steps).

Next, we discuss the contributions of this thesis, grouped by the respective issue. Here, we have to point out, that the novelty of our approach is established in each different chapter, by appropriately presenting the respective related work.

Trajectory Data Warehousing: We investigate how the traditional data cube model is adapted to trajectory warehouses in order to transform raw location data into valuable information. In particular, we focus our research on two issues that are critical to trajectory data warehousing:

- the ETL (Extract-Transform-Load) procedure that feeds a trajectory data warehouse with aggregate trajectory data. To this aim, we propose two alternative methods: a (index-based) *cell-oriented* and a (non-index-based) *trajectory-oriented* ETL process. As we will illustrate during the experimental study, the choice of a particular method is a trade-off between the selected granularity level and the number of trajectories;
- the aggregation of cube measures for OLAP purposes. The challenge is that a trajectory might span multiple base cells causing *aggregation* hindrances in OLAP operations. We extend the data cube model adding some auxiliary measures that will help us to correct the errors caused due to the duplicates when rolling-up. This is an approximate solution for this issue which turns out to perform effectively.

In both aforementioned research issues, we provide design solutions and we test their applicability and efficiency in real world settings. Our results in this topic are presented in Chapter 3. Preliminary results have been already published in [MFN+08a], [MFN+08b] and [MT09b].

Ad-hoc OLAP on trajectory data: We present a new approach in designing trajectory data cubes aiming at giving answers to ad-hoc OLAP queries taking into account different interpretations of the notion of trajectory. A flexible trajectory data cube that provides ad-hoc analysis can serve a number of applications even if they consider different definitions of trajectory data. In more details:

- We extend the OLAP data model so as to include a flexible fact table that can answer queries considering different semantic definitions of trajectories and provides the option to choose the appropriate semantic for aggregation queries over trajectory data;
- We develop an ETL technique that transforms the data appropriately in order to load them in the fact table. This technique utilizes the new OLAP data model and feeds the flexible TDW. Its performance, comparing to conventional ETL, deemed to be more efficient as we will present in the experimental section of this work;
- We enhance OLAP techniques so as to utilize the new data model. To this aim, we propose a competent algorithm that can answer ad-hoc aggregation queries. We also discuss materialization issues that improve the performance of this algorithm as they speed up the calculation process.

Preliminary experimental results illustrate the applicability and efficiency of our approach. Our results in this topic are presented in Chapter 3. Preliminary results have been submitted for evaluation [MT09a].

Mining interaction patterns: We propose a competent framework for mining interaction patterns that enables spatiotemporal representation, synthesis and classification. This work introduces two basic ideas:

- an adequate understanding of what is happening to a single object requires to look not only at its trajectory, but also at the context where it moves;
- such context is defined not only by the nature of the geographical space, but also by the presence of other objects and their interaction with the moving object under analysis.

We model interaction as a group-wise phenomenon trying to understand the behavior of an object with respect of its neighborhood. This is achieved by computing interesting interaction descriptors that can help us describe and understand the movement in various spatiotemporal windows. We compare alternative discovery processes: on the one hand, considering a dynamic neighborhood for each object and on the other hand, following a fixed grid approach and computing descriptors for static spatiotemporal windows. Preliminary experimental results illustrate the applicability and efficiency of our approach. Our results in this topic are presented in Chapter 4. Preliminary results have been submitted for evaluation [NMO09].

Mining traffic patterns: We propose a framework with mining capabilities over mobility data that are stored either in a MOD or in a TDW. We define various relationships between the edges of the network graph that allow us to combine the temporal information provided by the traffic time series with the spatial semantics inherited in the road network. We provide solutions for the detection of traffic patterns and we propose innovative mining algorithms for the detection of time-focused traffic relationships between the different road segments of a city network. We distinguish two discovery processes.

In summary, our approach discovers spatiotemporal patterns for supporting traffic management decisions. Preliminary experimental results illustrate the applicability and efficiency of our approach. Our results in this topic are presented in Chapter 4. Preliminary results have been already published in [NMM08] and [MT09b].

T-WAREHOUSE - a prototype for Visual TDW: We demonstrate a framework that transforms the traditional data cube model into a trajectory warehouse. As a proof-of-concept, we implemented T-WAREHOUSE, a system that incorporates all the required steps for Visual Trajectory Data Warehousing, from trajectory reconstruction and ETL processing to Visual OLAP analysis on mobility data. This system is based on the works thoroughly presented in Chapter 3 and is illustrated in details in Chapter 5. As for the trajectory reconstruction is concerned, we present an efficient technique that transforms sequences of raw sample points into meaningful trajectories. These reconstructed trajectories are then stored in the moving object database and are ready-available for ETL processing that feeds the TDW with aggregate mobility data.

We describe the architectural aspects of our framework and we investigate the power, flexibility and efficiency of our framework for applying OLAP analysis on real world mobility data. Preliminary

experimental results illustrate the applicability and efficiency of our approach. Preliminary results have been already published in [MFN+08a], [MFN+08b] and [LMF+10].

1.3. Thesis Outline

The outline of the thesis is as follows: In Chapter 2, we present basic concepts of spatiotemporal data distinguishing between mobile and immobile entities. Chapter 3 proposes a framework for Trajectory Data Warehousing; supporting all the steps from ETL to OLAP analysis and also introducing a variation of it that is suitably designed so as to support multiple definitions of the notion of trajectory. Chapter 4 presents two approaches for trajectory-inspired data mining focusing, the former on the discovery of interaction patterns from mobility data and the latter on the detection of traffic patterns in city networks, respectively. In Chapter 5 we present T-WAREHOUSE, a prototype tool for Mobility Data Warehousing. Finally, Chapter 6 summarizes the conclusions and results of our research and discusses interesting open issues.

2. Basic Concepts on Spatiotemporal Data

In this chapter we focus on the notion of spatiotemporal data, and we present two different kinds of them that should be treated in a different way: mobile and immobile data. The outline of the chapter is as follows: Section 2.1 introduces the issues being related to the spatiotemporal data. Section 2.2 examines the case of immobile entities, while Section 2.3 presents the notion of mobile entities. Section 2.4 discusses the need for the development of innovative decision support techniques foe either kind of spatiotemporal data. Finally, Section 2.5 provides the conclusions of the chapter.

2.1. Introduction

With advances in remote sensors, sensor networks, and the proliferation of location sensing devices in daily life activities and common business practices, the generation of disparate, dynamic, and geographically distributed spatiotemporal data has exploded in recent years. We should distinguish from the beginning of the thesis two different types of spatiotemporal data: on the one hand, those that involve the notion of mobility and on the other hand, static spatiotemporal data.

An important and interesting example of the latter category is the scientific data domain. Significant progress in ground, air- and space-borne sensor technologies has led to an unprecedented access to earth science data for scientists from different disciplines, interested in studying the complementary nature of different parameters. These developments are quickly leading towards a data-rich but information-poor environment. The rate at which geospatial data are being generated clearly exceeds our ability to organize and analyze them to extract patterns critical for understanding in a timely manner a dynamically changing world. Computer science and Geoinformatics can collaborate in order to address these scientific and computational challenges and provide innovative and effective solutions. In Section 2.2, we focus on seismological data, an exciting category of scientific data, and we present our approach on how to apply analytical techniques to help seismologists and geoscientists to gain knowledge from vast amounts of seismic data.

On the other hand, a typical category of mobility data is the time-stamped location data that can be collected by location-aware devices. The usage of such devices, such as mobile phones and GPS-enabled devices, is widely spread nowadays, allowing access to large datasets consisting of time stamped geographical locations. Appropriate handling of raw location data results in trajectory databases, a task that is usually referred as *trajectory reconstruction* [MFN+08a]. To address the emerging needs, the traditional database technology has been extended into Moving Object Databases

(MODs) that handle modeling, indexing and query processing issues for trajectories [GS05]. As it usually happens in data management world, the challenge after storing the data is the implementation of appropriate analytics that could extract useful knowledge. In Section 2.3, we introduce some basic issues regarding trajectories and MODs as well as about the *Geographic Privacy-aware KDD process*.

2.2. Immobile Entities: The Case of Seismological Data

For centuries, humans have been feeling, recording and studying earthquake phenomena. Taking into account that at least one earthquake of magnitude $M < 3$ ($M > 3$) occurs every one second (every ten minutes, respectively) worldwide, the seismic data collection is huge and rapidly increasing. Scientists record this information in order to describe and study tectonic activity, which is described by recording attributes about geographic information (epicenter location and disaster areas), time of event, magnitude, depth, etc.

On the other hand, computer engineers specialized in the area of Information & Knowledge Management find an invaluable "data treasure", which they can process and analyze helping in the discovery of knowledge from this data. A number of applications for the management and analysis of seismological or, in general, geophysical data have been proposed in the literature [AA99], [KR00], [The03], [Yu05]. In general, the collaboration between the data mining community and physical scientists has been only recently launched [BD00].

Earthquake phenomena are instantly recorded by a number of organizations (e.g. Institutes of Geodynamics and Schools of Physics) worldwide. Hence, a system that collects and analyzes the most accurate seismic data among different sources is needed. Obviously, some sources provide data about the same earthquakes though with slight differences in their details (e.g. the magnitude or the exact timestamp of the recorded earthquake). What raises from this discussion is the need for a generic architecture of a so-called *Seismic Data Management and Mining System* (SDMMS) that will be able to integrate the remote sources in a proper way by refining and homogenizing raw data [MTK08].

2.2.1. Seismic Data Warehousing and Mining

Desirable components of a SDMMS include tools for quick and easy data exploration and inspection, algorithms for generating historic profiles of specific geographic areas and time periods, techniques providing the association of seismic data with other geophysical parameters of interest, such as geological morphology, and, top line, visualization components using geographic and other thematic-oriented (e.g. topological and climatic) maps for the presentation of data to the user and supporting sophisticated user interaction.

In summary, we classify users that an SDMMS should support in three profiles:

- *Researchers of geophysical sciences*, interested in constructing and visualizing seismic profiles of certain regions during specific time periods or in discovering regions of similar seismic behavior.
- *Public administration officers*, requesting for information such as distances between epicenters and other demographical entities (schools, hospitals, heavy industries, etc.).

- *Citizens* ("web surfers"), searching for seismic activity, thus querying the system for seismic properties of general interest, e.g. for finding all epicenters of earthquakes in distance no more than 50Km from their favorite place.

The availability of systems following the proposed SDMMS architecture provides users a wealth of information about earthquakes assisting in awareness and understanding, two critical factors for decision making, either at individual or at administration level.

Collected data can be stored in a local database and/or a data warehouse (for simple querying and analysis for decision making, respectively). In general, data within the database is dynamic and detailed; while that within the data warehouse is static and summarized (this is because the modifications of the former are continuous, while the latter are subjected to periodical updates).

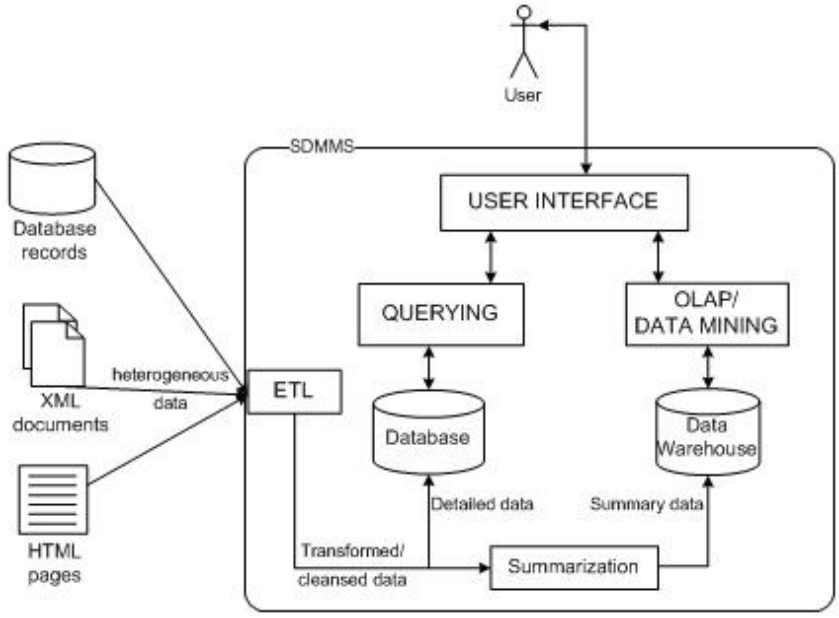


Figure 2-1: A general SDMMS architecture proposed for seismological data management.

Figure 2-1 presents the proposed abstract architecture that serves the task of collecting data from several sources around the world and storing them in a local repository (database and/or data warehouse). A mediator is responsible for the management of the process from the extraction of data from their sources until their load into the local repository, the so-called *Extract-Transform-Load* (ETL) approach. The collected data are cleansed and transformed to a common structure so as to store them in the database of the SDMMS. Furthermore, the collected data can be summarized so as to feed the Data Warehouse with aggregations.

Traditional Database Management Systems (DBMS) are known as operational database or OLTP (on-line transaction processing) systems as they support the daily storage and retrieval needs of an information system. Querying seismological databases involves spatiotemporal concepts like snapshots, changes of objects and maps, motion and phenomena [PT98], [The03]. In particular, SDMMS should provide at least the following database querying functionality:

- *Retrieval of spatial information given a temporal instance*. This concept is used, for example, when we are dealing with records including position (latitude and longitude of earthquake epicenter) and time of earthquake realization together with attributes like magnitude, depth of epicenter, and so on.
- *Retrieval of spatial information given a temporal interval*. This way, evolution of spatial objects over time is captures (assume, for example, that we are interested in recording the duration of an earthquake and how certain parameters of the phenomenon vary throughout the time interval of its duration).
- *Overlay of spatial information on layers given a temporal instance or interval*: The combination of layers and time information results into snapshots of a layer. For example, this kind of modeling is used when we are interested in magnitude thematic maps of earthquakes realized during a specific day inside a specific area (temporal instance) or modeling the whole sequence of earthquakes, including pre- and aftershocks (using the notion of layers in time intervals).

Examples of typical queries involving the spatial and the temporal dimension of seismological data are the following [The03]:

- *Find the ten epicenters of earthquakes realized during the past four months, which reside more closely to a given location.*
- *Find all epicenters of earthquakes residing in a certain region, with a magnitude M >5 and a realization time in the past four months.*
- *(Assuming multiple layers of information, e.g. corresponding to main cities' coordinates and population) find the five strongest quakes occurred in a distance of less than 100Km from cities of population over 1 million during the 20th century.*

Nevertheless, maintaining summary data in a local data warehouse can be used for data analysis purposes. Two popular techniques for analyzing data and interpreting their meaning are OLAP analysis and data mining. Summarized data and hidden knowledge acquiring from the stored data, can lead to better decisions. Similarly, summarized seismological data are of particular interest to earth scientists because they can study the phenomenon from a higher level and search for hidden, previously unknown knowledge. We illustrate the benefits obtained by such an approach with two examples of operations supported by spatial data warehouse and OLAP technologies:

- A user may ask to view part of the historical seismic profile, i.e. the ten most destructive quakes in the past twenty years, over Europe, and, moreover, he/she can easily view the same information over Greece (more detailed view, formally a *drill-down* operation) or worldwide (more summarized view, formally a *roll-up* operation).
- Given the existence of multiple thematic maps, perhaps one for quake magnitude and one for another, non-geophysical parameter such as the resulting damage, these maps could be

overlaid for the exploration of possible relationships, such as finding regions of high, though non-destructive, seismicity and vice versa.

Further to roll-up and drill-down operations described above, typical data cube operations include *slice* and *dice*, for selecting parts of a data cube by imposing conditions on a single or multiple cube dimensions, respectively (Figure 2-2), and *pivot*, which provides the user with alternative presentations of the cube.

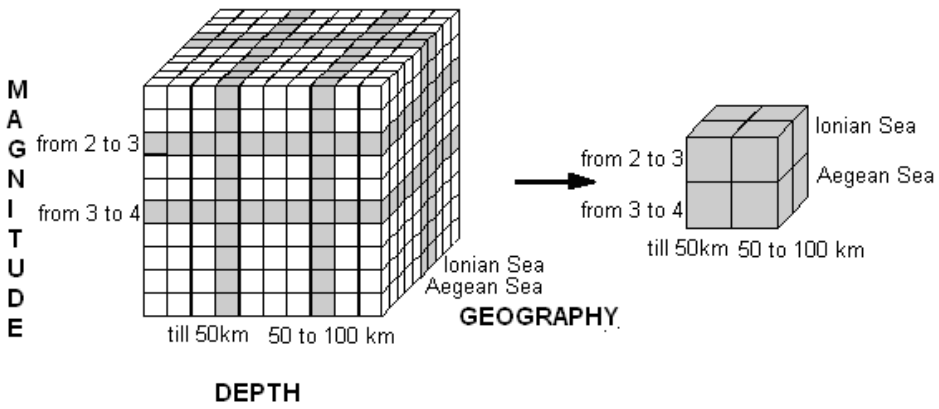


Figure 2-2: Selecting parts of a cube by filtering a single (slice) or multiple dimensions (dice).

Integrating data analysis and mining techniques into an SDMMS ultimately aims to the discovery of interesting, implicit and previously unknown knowledge. Examples of useful patterns found through Knowledge Discover & Delivery (KDD) process include clustering of information (e.g. shocks occurred closely in space and/or time), classification of phenomena with respect to area and epicenter, detecting phenomena semantics by using pattern finding techniques (e.g. characterizing the main shock and possible intensive aftershocks in shock sequences, measuring the similarity of shock sequences, according to a similarity measure specified by the domain expert, etc.).

2.3. Mobile Entities: The Case of Trajectory Data

Moving objects are geometries (i.e. points, lines, areas) changing over time and trajectory data describes the movement of these objects. Movement implies two dimensions; the spatial and the temporal. More specifically, movement can be described as continuous change of position in the geographical space and through comparison between two different temporal instants [MVO+08].

More formally, a trajectory *T* is a continuous mapping from the temporal $I \subseteq \mathrm{R}$ to the spatial domain (R^2, the 2D plane) [MVO+08]:

$$I \subseteq \mathrm{R} \rightarrow R^2 : \mathrm{t} \rightarrow \mathrm{a}(\mathrm{t}) = \left(\mathrm{a_x}(\mathrm{t}), \mathrm{a_y}(\mathrm{t})\right) \tag{2-1}$$

and,

$$T = \left\{\left(\mathrm{a_x}(\mathrm{t}), \mathrm{a_y}(\mathrm{t}), \mathrm{t}\right) \mid t \in I\right\} \subset \mathrm{R}^2 \times \mathrm{R} \tag{2-2}$$

where $\left(\mathrm{a_x}(\mathrm{t}), \mathrm{a_y}(\mathrm{t}), \mathrm{t}\right)$ are the sample points contained in the available dataset.

From an application point of view, a trajectory is the recording of an object's motion, i.e., the recording of the positions of an object at specific timestamps; while the actual trajectory consists of a curve, real-world requirements imply that the trajectory has to be built upon a set of sample points, i.e., the time-stamped positions of the object. Thus, trajectories of moving points are often defined as sequences of (x, y, t) triples [GBE+00]:

$$T = \{(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n)\}, \quad (2\text{-}3)$$

where $x_i, y_i, t_i \in \mathrm{R}$, and $t_1 < t_2 < .. < t_n$,

The space-time nature of the mobility data that are collected from modern sensing technologies and wireless telecommunication devices enables novel research fields related to the management of this new kind of data and the implementation of appropriate analytics for knowledge extraction.

Moving Object Databases is an important research field that has received a lot of attention during the last years. The main objective of this emerging area is to extend database technology so as to include appropriate techniques for the representation, querying, indexing and modeling of moving object's trajectories. Moreover, the analysis of the huge amount of collected trajectory data is a new interesting topic. This is because traditional analytical techniques cannot be applied as-is due to the spatiotemporal nature of trajectory data. What raises from this discussion is the need for a new KDD process which will be applicable on trajectory data. This thesis discusses several steps of this process; an outline of which is presented on the next subsection.

2.3.1. Moving Object Data Management, Warehousing and Mining

The research area of Moving Objects Databases has addressed the need for representing movements of objects (i.e., trajectories) in databases in order to perform ad-hoc querying and analysis on them. During the last decade, there has been a lot of research ranging from data models and query languages to implementation aspects, such as efficient indexing, query processing and optimization techniques. At least two MOD engines have been proposed in the literature, namely the SECONDO prototype [AGB06] and the HERMES engine [PFG+08].

HERMES is the core moving object database engine used for the development of the techniques proposed in this thesis and thus we briefly introduce it. HERMES is a database engine that provides spatiotemporal functionality so as to handle objects that change location, shape and size, either discretely or continuously in time. The system can be used either as a pure temporal or a pure spatial system, but its main functionality is to support the modeling and querying of continuously moving objects. Such a collection of data types and their corresponding operations are defined, developed and provided as an Oracle data cartridge, called HERMES Moving Data Cartridge (HERMES-MDC), which is the core component of the HERMES system architecture. It offers an expressive and easy to use query language for moving objects as well as a palette of moving object data types (e.g. Moving Circle, Moving Polygon, Moving Point etc). Among others, HERMES functionality includes:

- Queries on stationary reference objects; examples include distance-based or nearest neighbor queries (e.g., find nearby or closest landmarks, respectively, with respect to one's current location) and topological queries (e.g., find those crossed this area during the past hour);

- Queries on moving reference objects; examples include distance-based (e.g., find those passed close to me this evening) and similarity-based queries (e.g., find the three most similar trajectories to the one I followed yesterday);
- Queries involving unary operators, such as traveled distance or speed (e.g., find the average speed of the trajectory I have followed during weekend.

As for the analysis of mobility data, the term *Geographic Privacy-aware KDD process* emerged from the GeoPKDD project [Geo06] which proposed some solid theoretical foundations at an appropriate level of abstraction to deal with traces and trajectories of moving objects aiming at serving real world applications. This process consists of a set of techniques and methodologies that are applicable on mobility data and are organized in some well defined and individual steps that have a clear target: to extract user-consumable forms of knowledge from large amounts of raw geographic data referenced in space and in time, also taking into account privacy issues. More specifically, the main tasks (illustrated in Figure 2-3) of *Geographic Privacy-aware KDD process* are:

- reconstruction of trajectories from streams of raw data about moving objects, and construction of a privacy-aware trajectory warehouse;
- spatiotemporal privacy-preserving data mining and knowledge extraction algorithms, yielding spatiotemporal patterns;
- geographic knowledge interpretation and visualization techniques to deliver meaningful patterns to end users.

This KDD process can be applied in heterogeneous sources of mobility data. The cell phone that is illustrated in Figure 2-3 could represent various datasets coming from various devices:

- *GPS*: the fully-functional satellite navigation system that constellates more than two dozen satellites broadcasts precise timing signals by radio to GPS receivers, allowing them to accurately determine their location (longitude, latitude, and altitude) in any weather, day or night, anywhere on Earth.
- *GSM*: the most popular standard for mobile phones in the world, used by over 1.5 billion people across more than 210 countries and territories. The ubiquity of the GSM standard makes international roaming very common between mobile phone operators, enabling subscribers to use their phones in many parts of the world. GSM networks consist of a numbers of base stations each responsible for a particular geographical area (known as cell). Hence, for each GSM-enabled device we can collect from which base stations was served at different timestamps and therefore we can assume its movement.
- *Wi-Fi*: the most popular standard for wireless communication between devices. A Wi-Fi enabled device such as a laptop, mobile phone, PDA etc can connect to another device when it is within the range of a wireless network. The wireless network is defined as a set of interconnected access points — called hotspots — that can cover an area as small as a single room or as large as a whole city (WiMax). As in GSM, we can collect for each device the list of hotspots that served it at different timestamps.

In the real world, these different data types could be available and need to be stored, queried and analyzed. Towards this aim, we need appropriate *trajectory reconstruction* techniques that respect the data differences. Obviously the concept of these techniques is the same: they transform raw data into trajectories. However, the variations in the nature and the accuracy of the data demand a different approach for each data type. Moreover, trajectory reconstruction techniques may perform some basic trajectory preprocessing. This may include parameterized trajectory compression (so as to discard unnecessary details and concurrently keep informative abstractions of the portions of the trajectories transmitted so far), as well as techniques to handle missing/erroneous values.

The reconstructed trajectories are stored in the trajectory data warehouse that serves two core needs: to provide the appropriate infrastructure for advanced reporting capabilities and to facilitate the application of trajectory mining algorithms on the aggregated data. According to end users needs, they could have access either to basic reports or *OLAP-style* analysis. What-if scenarios and multidimensional analysis are typical examples of analytics that could be based on the trajectory data warehouse.

Kindly consider a relaxed definition of the trajectory warehouse that is presented in Figure 2-3. It could include a MOD that stores trajectory data in full details. The MOD can feed the TDW with aggregate data by applying an ETL process which aim to deriving qualitative information (e.g. trajectories in different granularities, aggregations, motional metadata etc.).

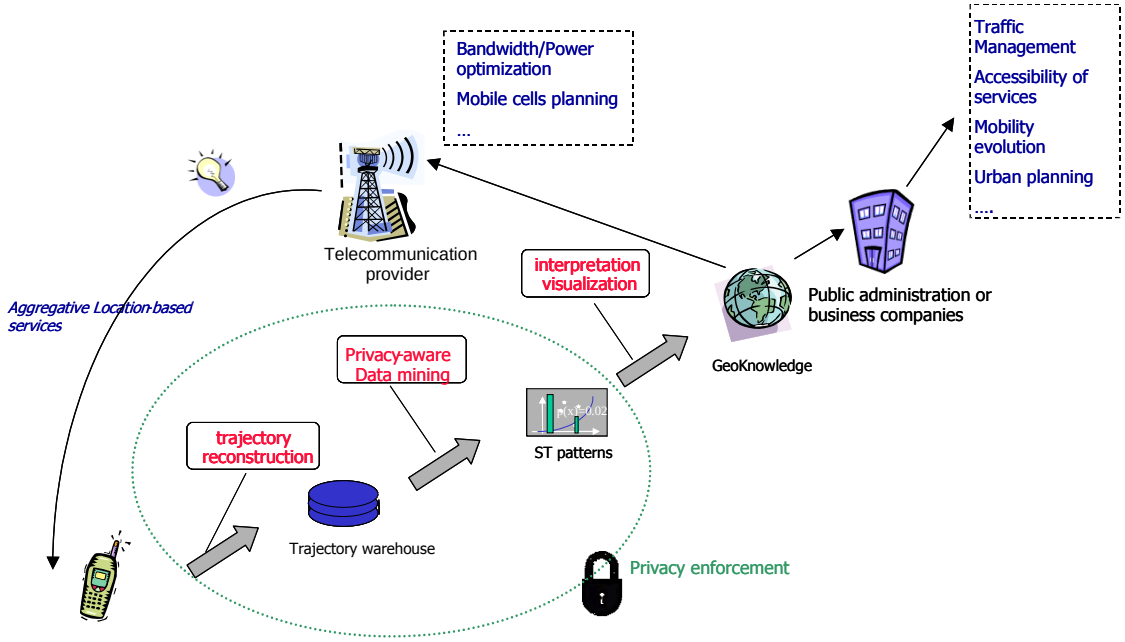


Figure 2-3: The big picture of moving object data management, warehousing and mining concepts [Geo06].

To deal with moving object applications that are restricted to some network, both the MOD and the TDW may need map matched trajectories. In other words, they may need the specific trajectory points and portions to correspond to valid network paths. This may include for example, performing pre-

processing or post-processing tasks that do not violate the validity of trajectories in terms of the real underlying network.

Additionally, Geoknowledge in Figure 2-3 can be expressed by incorporating of GIS layers that could result in a richer conceptual model providing thus more advanced analysis capabilities. Combing trajectory data with thematic layers (such as geographic, topographic and demographic layers) could enhance the analytics capabilities of potential applications.

Trajectory mining regards the application of data mining techniques on trajectory data. This task produces trajectory patterns that describe the behavior of trajectories. Sequential and frequent patterns may be discovered using traditional or ad hoc pattern extraction methods.

Concerning privacy, privacy has to be embedded into the data warehousing and mining tools (as it is exposed in Figure 2-3) so as to guarantee that trajectories of mobile individuals will stored and analyzed without violating personal privacy rights. A very simple strategy towards the protection of personal privacy is that we avoid discovering patterns that regards only a limited number of users. This has to be done so as to avoid identifying these users.

The discovered knowledge can be useful to both public administration and business companies. Especially in the case of telecommunication provider, the discovered knowledge could be really helpful in a number of domains: such as the optimization of mobile network services as well as the development of new innovative location based services that will offer useful applications.

2.4. The Need for Innovation in Decision Support Techniques

Traditional decision support techniques developed as a set of applications and technologies for gathering, storing, analyzing, and providing access to data, e.g. data warehousing, online analytical processing, data mining and visualization. These techniques are embedded in decision support systems to support business and organizational decision-making activities. Such systems help decision makers combine raw data, documents, personal knowledge, business models, etc to identify, analyze and solve problems as well as make decisions.

Decision support techniques were developed to satisfy the changeable and complicated needs of current business and technological environment. Towards this aim it is necessary to constantly extend them appropriately so as to give solutions to the new challenges that arise. It is commonly believed that there are two challenges that lead the change in the area of decision support techniques: on the one hand there is the need to ameliorate the existing tools so as to exploit real-time operational data and on the other hand the technological advances in the area of communications allow access to vast volumes of mobility datasets.

As for the former, it regards the extension of existing tools so as to support also operational beyond strategic and tactical decisions. The differences among these types are related with the time scale that every decision demands and with the nature of them as well. The top management is responsible for the strategic planning of their organizations, whereas middle managers make tactical decisions following the plans of top management. Hence, operational decisions, responsible for the daily activity of the

organization, were underestimated. Nowadays, there is the trend for integrated performance measurement and management [MT06] and this lead to new tools and techniques that can utilize operational data [Kot06], [ADH+03], [CCD+04].

This thesis focuses on the second challenge and discusses the extension of traditional techniques so as to deliver new analytics, suitable for mobility data. The aim is to server emerging applications (e.g. mobile marketing and traffic management) that need to convert raw location data into useful knowledge.

Their application of DW and OLAP techniques on conventional business data has been extensively in the literature. Applying such techniques on mobility data can provide us with useful knowledge about trajectories. A Trajectory Data Warehouse can help towards computing aggregations on trajectory data and thus studying them in a higher level of abstraction. Theoretically speaking, this approach preserves privacy as it is not focus on individual trajectories. Moreover, it leads to a data repository that collects and homogenizes data from multiple sources. This allows serving further analytical techniques that can be applied on aggregate data instead of raw location datasets.

Data mining techniques are used to discover unknown, useful patterns. The vast amount of available mobility data requires the extension of traditional mining techniques so as to be suitable for this new kind of data. Discovering spatiotemporal associations, clusters, predicting actions etc lead to mobility patterns that could help us to construct summary and useful abstractions of large volumes of raw location data and gain insights on movement behaviors.

2.5. Synopsis

In this chapter, we distinguished two different types of spatiotemporal data: those that involve the notion of mobility and the static spatiotemporal data. We also presented different architectures for the management of the data belonging in the two categories. Regarding the static spatiotemporal data, we considered as a typical example the seismological data and presented a complete framework for *Seismic Data Management and Mining*. As for mobility data, we outlined the area of Moving Object Databases and we presented the notion of *Geographic Privacy-aware KDD process*.

3. Efficient Trajectory Data Warehousing

In this chapter we focus on data warehousing techniques and we study their application on trajectory data. We present our two proposals, a framework for TDW that demands a unique and static suitable interpretation of the notion of trajectory and a framework for ad-hoc TDW which allows multiple semantic definitions regarding the notion of trajectory. The outline of the chapter is as follows: Section 3.1 introduces the issues being related to the data warehousing techniques on mobility data, whreas Section 3.2 motivates our research. Section 3.3 focuses on data warehousing issues regarding static semantic definitions of trajectories while, Section 3.4 presents a framework that considers multiple semantic definitions of trajectories. Section 3.5 examines the related work and Section 3.6 closes the chapter providing the conclusions.

3.1. Introduction

Data warehousing has received considerable attention of the database community as a technology for integrating all sorts of transactional data, dispersed within organizations whose applications utilize either legacy (non-relational) or advanced relational database systems. Data warehouses form a technological framework for supporting decision-making processes by providing informational data. A data warehouse is defined as a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management of decision making process [Inm96].

In a data warehouse, data are organized and manipulated in accordance with the concepts and operators provided by a multidimensional data model, which views data in the form of a data cube [AAD+96]. A data cube allows data to be modeled and viewed in multiple dimensions, where each dimension represents some business perspective, and is typically implemented by adopting a star (or snowflake) schema model. According to this model, the data warehouse consists of a fact table (schematically, at the centre of the star) surrounded by a set of dimensional tables related with the fact table, which contains keys to the dimensional tables and measures. A single entry in the fact table modeling the primitive analysis component is called *fact*.

For instance, the purpose of the data warehouse illustrated in Figure 3-1 is to store aggregated data about sales transactions taking place in various stores around the country. There is a fact table *Sales Fact Table* containing keys to the four dimension tables and three measures: *#transactions* which counts the number of sales-transactions (in other words the number of baskets), *total value* as the total amount of sales and *avg_quantity* which measures the average number of products.

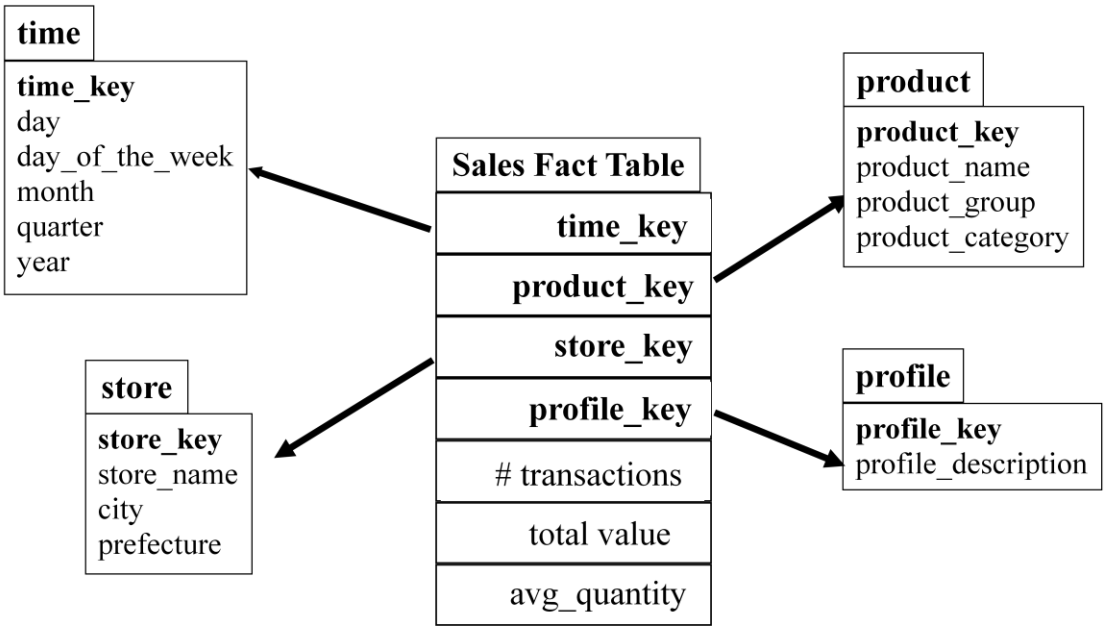


Figure 3-1: A simple (conventional) data cube schema.

Dimensions represent the analysis axes, while measures are the variables being analyzed over the different dimensions. For example, in Figure 3-1 the dimensions are *product, profile, store, time*. As for the profile dimension, it represents a specific group of people with some common characteristics (age, gender, occupation etc). So in this case, the data warehouse stores the number of transactions, the amount of sales and the average quantity for a given product bought by a specific group of people, in a given store and over a given period of time.

Each dimension is organized as a hierarchy (or even a set of hierarchies) of dimension levels, each level corresponding to a different granularity for the dimension. For example, year is one level of the time dimension, while the sequence <day, month, year> defines a simple hierarchy of increasing granularity for the time dimension. Finally, the members of a certain dimension level (e.g. the different months for the time dimension) can be aggregated to constitute the members of the next higher level (e.g. the different years). The measures are also aggregated following this hierarchy by means of an aggregation function. The same approach can be applied in the remaining dimensions.

Data Warehouses are optimized for OLAP operations. Typical OLAP operations include the aggregation or de-aggregation of information (called *roll-up* and *drill-down*, respectively) along a dimension, the selection of specific parts of a cube (*slicing* and *dicing*) and the reorientation of the multidimensional view of the data on the screen (*pivoting*) [Kim96].

Generally speaking, Data Warehousing and OLAP techniques can be employed in order to convert vast amount of raw data into useful knowledge. However, the conventional techniques were not designed for analyzing trajectory data. Hence, there is the need for extending Data Warehousing technology so as to handle mobility data. This chapter discusses all the necessary steps for building real-world Trajectory Data Warehouses. Indicatively, such a warehouse could analyze measures like the number of vehicles in specific spatial areas, the average accelerations of vehicles, the maximum and average

speed of vehicles. This analysis could be done through appropriate dimensions (e.g. a spatial and a temporal) that will allow us to explore aggregated data under different granularities.

3.2. Motivation Issues

The motivation behind a TDW is to transform raw trajectories to valuable information that can be used for decision making purposes in ubiquitous applications, such as Location-Based Services (LBS), traffic control management, etc. Intuitively, the high volume of raw data produced by sensing and positioning technologies, the complex nature of data stored in trajectory databases and the specialized query processing demands make extracting valuable information from such spatiotemporal data a hard task. For this reason, the idea is to extend traditional aggregation techniques so as to produce summarized trajectory information and provide OLAP style analysis.

Extending traditional (i.e., non-spatial), spatial or spatiotemporal models to incorporate semantics driven by the nature of trajectories introduce specific requirements as the goal is twofold: to support high level OLAP analysis and to facilitate knowledge discovery from TDWs. Having in mind that the basic analysis constituents in a TDW (i.e. facts) are the trajectories themselves, in this section, we categorize the identified requirements into modeling, analysis and management requirements. The first considers logical and conceptual level challenges introduced by TDWs, the second goes over OLAP analysis requirements, while the third focuses on more technical aspects. In each case, we also survey related work.

3.2.1. Data Cube Modeling Issues

The following paragraphs investigate the prerequisites and the constraints imposed when describing the design of a TDW from a user perspective (i.e. conceptual model), as well as when describing the final result as a system in a platform-independent tool (i.e. logical model).

Thematic, spatial, temporal measures

From a modeling point of view, a trajectory is a spatial object whose location varies in time (recall discussions on the nature of trajectories in Section 2.3). At the same time, trajectories have thematic properties that usually are space and time dependent. This implies that different characteristics of trajectories need to be described in order to be analyzed. As such, we distinguish:

- numeric characteristics, such as the average speed of the trajectory, its direction, its duration;
- spatial characteristics, such as the geometric shape of the trajectory;
- temporal characteristics, such as the timing of the movement; and
- spatiotemporal characteristics; such as a representative trajectory or a cluster of trajectories.

Additionally, as we pay particular attention to uncertainty and imprecision issues, a TDW model should include measures expressing the amount of uncertainty incorporated in the TDW due to raw data imprecision. Uncertainty should also be seen in granularities, while this implies that there are special aggregation operators propagating uncertainty to various levels. In particular, depending on the application and user requirements, several numeric measures could be considered.

- the number of trajectories found in the cell (or started/ended their path in the cell; or crossed/entered/left the cell, and so on);
    - For instance, in Figure 3-2: no one trajectory started or ended in cell C_1, two trajectories crossed, entered and left the cell.
- the {average/min/max} distance covered by trajectories in the cell;
    - For instance, in Figure 3-2: the average distance covered is the summary of the length of the portions of trajectories inside the cell C_1 (the bold lines) divided by the number of trajectories found in the cell (two trajectories).
- the {average/min/max} time required to cover this distance.
    - For instance, in Figure 3-2: the average time required lied inside cell C_1 is the summary of the durations of the portions of trajectories inside the cell C_1 (the bold lines) divided by the number of trajectories found in the cell (two trajectories).

Other measures could include motion characteristics of the trajectories, e.g. speed and change of speed (acceleration), direction and change of direction (turn), underlying spatial framework characteristics (e.g. network usage, frequency, density), and also the uncertainty associated with the locations of objects in the database. Handling uncertainty, the warehouse could even contain information regarding the quality of raw data (e.g. spatial/temporal tolerance of recordings).

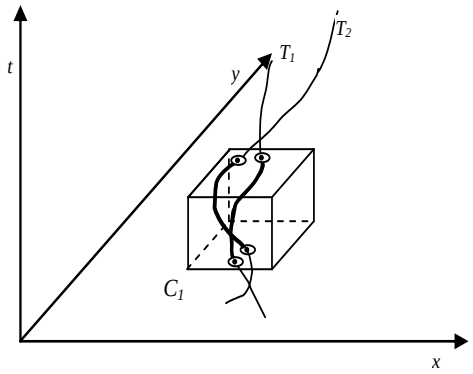


Figure 3-2: The portions of trajectories that lie within a cell.

As a final remark about measures, it is worth noticing that even restricting to numeric measures, the complexity of the computation can vary a lot. Some measures require little pre-computation and can be updated in the data warehouse while single observations of the various trajectories arrive, whereas others need a given amount of trajectory observations before updating. Braz et al. [BOO+07] propose the following classification of measures according to an increasing amount of pre-calculation effort:

a) no pre-computation: the measure can be updated in the data warehouse by directly using each single observation;

b) per trajectory local pre-computation: the measure can be updated by exploiting a simple pre-computation, which only involves a few and close observations of the same trajectory;

c) per trajectory global pre-computation: the measure update requires a pre-computation which considers all the observations of a single trajectory;

d) global pre-computation: the measure requires a pre-computation which considers all the observations of all the trajectories.

For instance:

- the number of trajectories starting/ending their path in the cell can be of type a (as explained above);
- if the first/last point of the trajectories are marked, the distance covered by trajectories in the cell, the number of trajectories that entered, left the cell are of type b (in Figure 3-2 these points have been marked for trajectories T_1 and T_2);
- the number of trajectories that covered a total distance larger than a given value *v* is of type c (e.g. in Figure 3-2, the distance covered by the portions of trajectories inside cell C_1 will be compared to the value v);
- the number of trajectories that intersect another trajectory only in the cell is of type d (e.g. in Figure 3-2 there is one intersection in cell C_1).

The amount of pre-calculation associated with each type of measure has also a strong impact on the amount of memory required to buffer incoming trajectory observations. Note that, since observations may arrive in stream at different rates, and in an unpredictable and unbounded way, low processing time and limited memory size are both important constraints.

Similar remarks can be found in [HSK98] where Han et al. present three methods to compute spatial measures in spatial data cube construction. The first one consists of simply collecting and storing the corresponding spatial data but no pre-computation of spatial measures is performed. Hence such a method may require more computation on-the-fly. The second method pre-computes and stores some rough approximation/estimation of the spatial measures in a spatial data cube. For instance, if the measure is the merge of a set of spatial objects, one can store the Minimum Bounding Rectangle (MBR) of the merge of the objects. Finally, one can selectively pre-compute some spatial measures. In this case the question is how to select a set of spatial measures for pre-computation. In [HSK98] some criteria for materialization of a cuboid are presented.

Thematic, spatial, temporal dimensions

Regarding the supported dimensions, as starting point a TDW should support the classic spatial (e.g. coordinate, roadway, district, cell, city, province, country) and temporal (e.g. second, minute, hour, day, month, year) dimensions and hierarchies, describing the underlying spatiotemporal framework wherein trajectories are moving. Additionally, it is important to allow space-time related dimensions interact with thematic dimensions describing other sorts of information regarding trajectories like technographic (e.g. mobile device used) or demographic data (e.g. age and gender of users) [MFN+08a]. This will allow an analyst not only to query the TDW for instance about the number of objects crossed an area of interest but also to be able to identify the objects in question.

This is particularly important as in the first case we usually get quantitative information, while in the second case, the information is qualitative. Consequently, a rich TDW schema should include the following dimensions:

- temporal (time);
- geographical (location);
- demographics (e.g. gender, age, occupation, marital status, home postal code, work postal code, etc.);
- technographics (e.g. mobile device, GPRS-enabled, subscriptions in special services, etc.).

Regarding the technographics and demographics dimensions, the idea behind them is to enhance the warehouse with semantic information. These dimensions allow the grouping of trajectories according to demographical characteristics or based on the technological characteristics of their devices.

An issue concerning the definition of dimensions is the considered level of detail for each. Let us consider the spatial dimension: since a trajectory is actually a set of sampled locations in time, for which the in-between positions are calculated through some kind of interpolation, the lowest level information is that of spatial coordinates. This, however, implies a huge discretization of the spatial dimension, thus more generic approaches should be followed. For example, cell positions could be used instead of point positions.

Hierarchies on dimensions

Once having defined the dimensions, their hierarchies can be explicitly specified by users or generated automatically by data clustering or data analysis techniques. A general technique used to define hierarchies includes the discretization of the values over the dimension ranges, resulting in a set-grouping hierarchy. A partial order can thus be established among these groups of values. Let us now analyze the different proposals and difficulties in creating hierarchies for the dimensions suggested in the previous subsection.

Defining hierarchies over the time dimension is straightforward, since typically there is an obvious ordering between the different levels of the hierarchy. For instance, a potential hierarchy could be: Year > Quarter > Month > Day > Hour > Minute > Second. Other hierarchies over the time dimension could concern seasons, time zones, traffic jam hours, and so on.

On the other hand, creating hierarchies over spatial data is more complicated. In fact, non explicitly defined hierarchies might exist over the spatial data. For example, in the hierarchy Country > City > District > Cell > Road, it is not always the case that an inclusion relation holds between District and Cell and between Cell and Road. A Road value, for example, might cross more than one Cell values. To solve this problem, Jensen et al. [JKP+04] proposed a conceptual model, which supports dimensions with full or partial containment relationships. Thus, when a partial containment relationship exists between the different levels of a dimension, one should specify the degree of containment, e.g. 80% of this Road is covered by this Cell.

Besides the standard relation City < Country, further hierarchies could be defined over the spatial dimension depending on the application, e.g. a set-grouping hierarchies on districts according to the pollution.

Finally, as far as the demographic and technographic dimensions are concerned, the simplest solution is to create a hierarchy for each dimension. This solution, however, might cause complexity problems especially if the number of the dimensions considered is large. Another possibility is to combine attributes of these dimensions by creating groups of dimensions values and use these groups as the levels of abstraction. As an example of such a group consider the following one: "gender = female, age = 25 - 35, marital status = single". The group definition could be performed by a domain expert or by carrying out some statistical pre-processing over the data. This approach reduces the number of dimensions, thus allowing for a simpler and more efficient data warehouse in terms of processing time and storage requirements.

Finally, some approaches [JKP+04], [MZ04b] offer the support for creating multiple hierarchies for each dimension. These works focus on the hierarchies that can be defined in the spatial dimensions. More specifically, the spatial dimension may include not explicitly defined hierarchies. Thus, multiple aggregation paths are possible and can be taken into consideration during OLAP operations.

3.2.2. OLAP Requirements

In traditional data warehouses, data analysis is performed interactively by applying a set of OLAP operators. In spatial data warehousing, particular OLAP operators have been defined to tackle the specificities of the domain [PTK+02]. Similarly, in our context, we expect an algebra of OLAP operators to be defined for trajectory data analysis. Such an algebra should include not only the traditional operators, such as roll-up, drill-down and selection properly tailored to trajectories, but also additional operators which account of the specific city of the spatiotemporal data type. Below we present these operators in more detail:

Roll-up. The roll-up operation allows us to navigate from a detailed to a more general level of abstraction either by climbing up the concept hierarchy (e.g. from the level of 'city' to the level of 'country') or by some dimension reduction (e.g. by ignoring the 'time' dimension and performing aggregation only over the 'location' dimension).

As shown in [BOO+07], depending on the kind of analyzed measures, the roll-up operation in TDWs can introduce some errors. Assuming the object or trajectory identifier is not recorded, when summing up along the spatial and/or temporal dimension, one cannot obtain the distinct number of trajectories because there is only aggregated information. This is a particular case of the distinct counting problem [TKC+04] which is presented in Section 3.5.1.2.

In Figure 3-3 let us consider the spatial projection of some cells of the cube. The TDW stores for instance the number of distinct trajectories in each cell. Hence, we have four distinct trajectories in R_4, two in R_5 and one R_6. If we try to aggregate at a higher hierarchy level, i.e. R that contains R_4, R_5 and R_6, then a traditional roll-up operation would count six distinct trajectories instead of 3 which is the correct answer.

Another open issue concerns the application of the roll-up operation when uncertain data exist, which is the case for the trajectories. Indeed, two factors of uncertainty should be taken into account during the aggregation: the uncertainty in the values, and the uncertainty in the relationships. The former refers to the uncertainty associated with the values of the dimensions and measures, which is propagated into the warehouse from the source data. The latter refers to the uncertainty imposed into the warehouse due to the non-explicitly defined concept hierarchies.

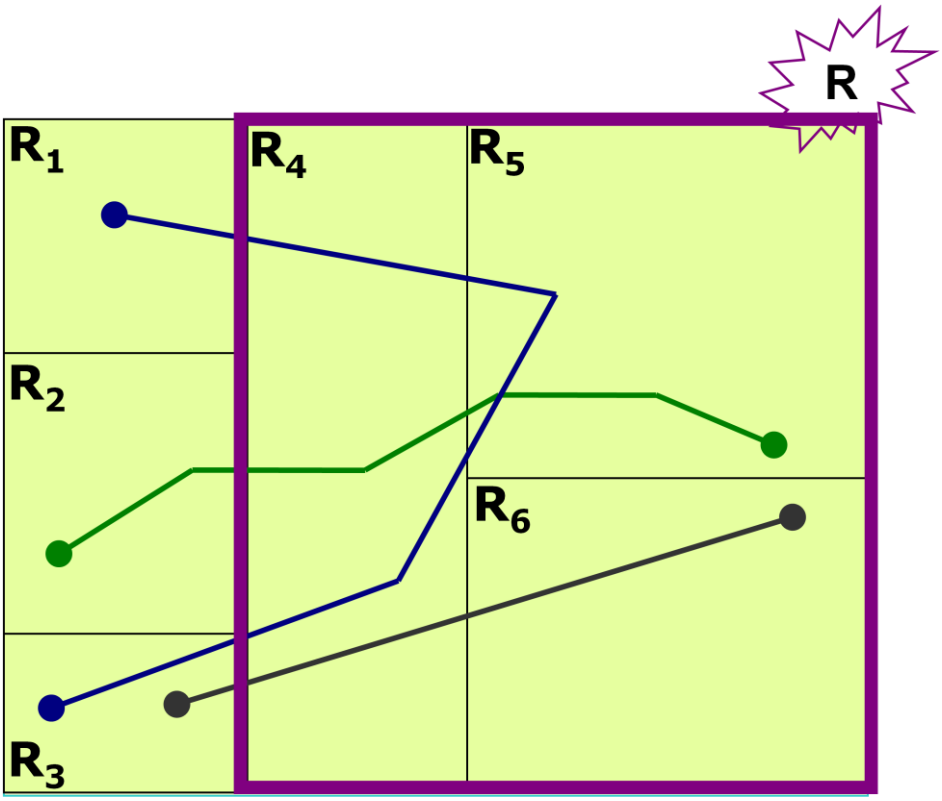


Figure 3-3: Aggregating measures in the cube.

Drill-down. The drill-down operation is the reverse of roll-up. It allows us to navigate from less detailed to more detailed information by either stepping down a concept hierarchy for a dimension (e.g. from the level of 'country' to the level of 'city') or by introducing additional dimensions (e.g. by considering not only the 'location' dimension but the 'time' dimension also). Similarly to the roll-up operation, drill-down is also 'sensitive' to the distinct counting problem and to the uncertainty associated with both values and relationships. As we already mentioned, in Figure 3-3, the number of distinct trajectories in R it is not equal to the number of distinct trajectories in the drilled-down regions R_4, R_5 and R_6.

Slice, Dice. The slice operation performs a selection over one dimension (e.g. 'city=Athens'), whereas the dice operation involves selections over two or more dimensions (e.g. 'city=Athens and year=2006'). The conditions can involve not only numeric values but also more complex criteria, like spatial and/or temporal query windows. To support these operations, the selection criteria can be transformed into a query against the TDW and processed by adequate query processing methods. In summary, traditional OLAP operations should be also supported by a TDW since they provide

meaningful information. Another motivation is the question whether other operations dedicated to trajectories might be defined. Examples include:

- operators that dynamically modify the spatiotemporal granularity of measures representing trajectories;
- medoid etc. operators which apply advanced aggregation methods, such as clustering of trajectories to extract representatives from a set of trajectories;
- operators to propagate/ aggregate uncertainty and imprecision present in the data of the TDW.

As for the first issue, we present in Section 3.4 a complete framework that allows defining the spatiotemporal granularity of trajectories in a dynamic way in the context of TDW. The second issue is discussed in Section 6.2 where the representative trajectory measure is outlined as an open issue. Although dealing with uncertainty data is beyond the scope of this thesis, we included this as a possible requirement as it is real world problem.

3.2.3. Management Requirements: ETL Issues, Support for Continuous Data Streams and Multiple Spatial Topologies

The previous sections disclosed higher level requirements for TDWs as these can be captured by extended conceptual and logical data warehouse models. In this section we investigate the management requirements of a TDW from an implementation point of view, but still without restricting the discussion under a specific physical modeling framework.

Having as main objective to build a data warehouse specialized for trajectories, and considering the complexity and the vast volumes of trajectory data, we need to differentiate our architectural design from the one in traditional DWs. The situation is made even more complicated by the streaming nature of data sources, such as logs from location-aware communication devices, which potentially come in continuous rows of unbounded size. Therefore, efficient and effective storage of the trajectories into the warehouse should be devised, capable of dealing with continuous incoming streams of raw log data, while the TDW itself must be equipped with suitable access methods to facilitate analysis and mining tasks. This poses extra challenges to be solved as the ability of incrementally processing the data stream in an efficient and accurate way, and the definition of adaptive strategies to make the hypercubes evolve with the data stream.

Also, due to the peculiarities of trajectories, some problems can arise in the loading phase of the fact table. To give an intuitive idea of these issues, consider a data cube where the facts are still the trajectories, but having only the spatial and temporal dimensions, discretized according to a regular grid, and as measure the number of distinct trajectories in the spatiotemporal cell, generated by the grid.

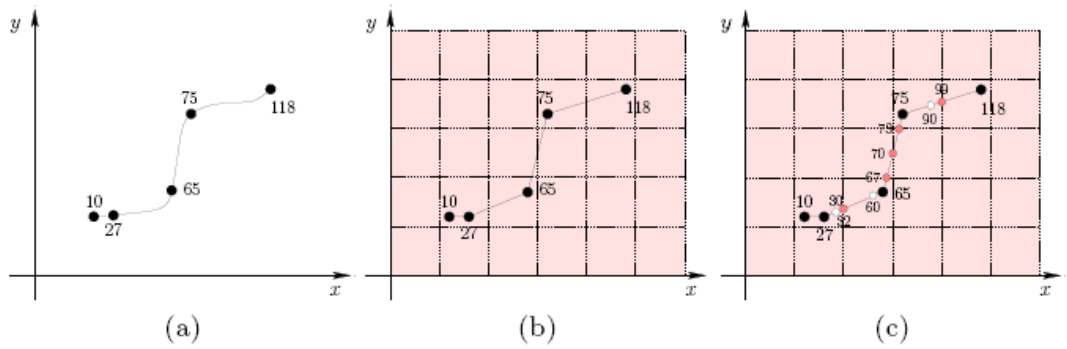


Figure 3-4: (a) A 2D trajectory, with a sampling (b) Linear interpolation of the trajectory (c) The interpolated trajectory with the points matching the spatial and temporal minimum granularity.

Moreover, assume that a trajectory is modeled as a finite set of observations, i.e. a finite subset of points taken from the actual continuous trajectory, later called sampling. For example, Figure 3-4(a) shows a sampling of a trajectory.

The main issues are the following:

- the rough observations in a sampling cannot be directly used to compute the measures of interest in a correct way, and
- these observations are not independent points; the fact that they belong to the same trajectory has to be exploited when computing some measures.

For instance, loading the fact table with the points in Figure 3-4(b) results in a very simple fact table (Figure 3-5). Notice that the first column of the table does not belong to the fact table; it is used to clarify which observations fall in the spatiotemporal cell. It is evident that other cells might be crossed by the trajectory (e.g., the cell [60; 90) x [60; 90) x [60; 90)), meaning that some information can be missing. On the other hand, the same cell can contain more than one observation, the computed measure is not correct because it does not store the number of distinct trajectories (see the cell [30; 60) x [30; 60) x [0; 30)).

Time label	X Interval	Y Interval	T Interval	N Trajs
10,27	[30,60)	[30,60)	[0,30)	2
65	[60,90)	[30,60)	[60,90)	1
75	[90,120)	[90,120)	[60,90)	1
118	[120,150)	[90,120)	[60,120)	1

Figure 3-5: A simple fact table for a trajectory warehouse.

In order to solve the first problem, Braz et al. [BOO+07] propose further intermediate points to be added by linearly interpolating the trajectory. The newly inserted points are the ones which intersect the borders of the spatiotemporal cell, considering all its three dimensions. Figure 3-4(c) shows the resulting interpolated points as white and gray circles. Note that the white interpolated points, associated with temporal labels 30, 60, and 90, have been added to match the granularity of the temporal dimension. In fact, they correspond to cross points of the temporal border of the 3D cell. On the other hand, the gray points, labeled with 32, 67, 70, 73, and 99, have been instead introduced to match the spatial dimensions. They correspond to the cross points of the spatial borders of some 3D

cell, or, equivalently, the cross points of the spatial 2D squares depicted in Figure 3-4(c). The second problem concerning duplicates is more complex and an approach to cope with it is presented in Section 3.3.2. A thorough discussion about errors in the computation of different measures related to the described issues can be found in [BOO+07].

Certainly, a factor that characterizes a TDW is the interrelationship between the development of the trajectories upon various possible spatial topologies represented by corresponding spatial dimensions. The base level partitioning of a spatial topology directly affects the multidimensional analysis of trajectories. Possible available topologies may be simple grids (e.g. artificial partitioning), complex polygonal amalgamations (e.g. suburbs of a city), real road networks and mobile cell networks. The first case is the simplest one as the space is divided in explicitly defined areas of a grid and thus it is easy to allocate trajectory points in specific areas. However, counting the number of objects that passed from an area may be proved hard for a trajectory data warehouse. This is because sampling frequency may not help in representing the actual trajectory [BOO+07]. Thus, it may be necessary to reconstruct the trajectory (as an ETL task) to add intermediate points between the sampling data (see Figure 3-4(c)).

In case of road networks, trajectories should be reconstructed so as to be network constrained, whereas managing cells is a more complex problem because the areas covered by cells may change from time to time depending on the signal strength of the base stations of the provider. Whatever the base of the spatial dimension relating with the trajectories all spatial topologies are subject to the distinct counting problem [TKC+04] which will be presented in Subsection 3.3.2.

Obviously, the reconstruction of the trajectories and the multiple counts of an object moving inside a region is straightforwardly dependent on the interpolation (e.g. linear, polynomial) used (if any) by the corresponding trajectory data model. The above discussion implies that an analyst has the ability firstly to analyze a bunch of trajectories according to a population thematic map and at a secondary level according to the road network of the most populated area.

3.2.4. Our Contributions

In order to build a TDW, several issues should be handled; we summarize these issues below accompanied with our contributions as presented in [MFN+08a], [MFN+08b], [MT09b], [MT09a]:

- The TDW is to be fed with aggregate trajectory data; to achieve it we propose two alternative solutions: a (index-based) *cell-oriented* and a (non-index-based) *trajectory-oriented* ETL process.
- Aggregation capabilities over measures should be offered for OLAP purposes (i.e., how the measures at a lower level of the cube hierarchy can be exploited in order to compute the measures at some higher level of the hierarchy). The peculiarity with trajectory data is that a trajectory might span multiple base cells (the so called *distinct count problem*). This causes *aggregation* hindrances in OLAP operations. We provide an approximation solution for this problem, which turns out to perform effectively.

- We present an alternative, innovative organization of a trajectory data cube in order to answer OLAP queries considering different interpretations of the notion of trajectory. Thus, ad-hoc analysis on trajectory data cubes can be achieved, which can be really useful for a number of applications.

3.3. Trajectory Data Warehousing

In [MFN+08a], we proposed a framework for TDW that takes into consideration the complete flow of tasks required during a TDW development. The complete lifecycle of a TDW is illustrated in Figure 3-6 and it consists of various steps. A *Trajectory Reconstruction process* is applied on the raw time-stamped location data in order to generate trajectories, which are then stored into a *MOD*. Then, an *Extract-Transform-Load* (*ETL) procedure* is activated that feeds the data cube(s) with aggregate information on trajectories. The final step of the process offers OLAP (and, eventually, DM) capabilities over the aggregated information contained in the trajectory cube model.

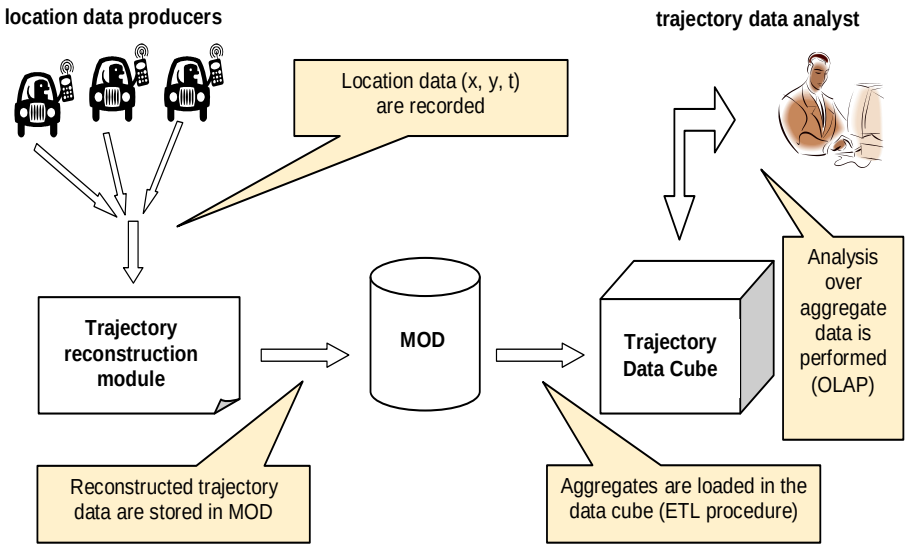


Figure 3-6: The architecture of our framework.

A MOD maintains object locations recorded at various time points in the form of trajectories. Formally, let $D = \{T_1, T_2, \ldots, T_N\}$ be a MOD consisting of the trajectories of a set of moving objects. Assuming linear interpolation between consecutive sampled locations, the trajectory $T_i = <(x_{i_1}, y_{i_1}, t_{i_1}),...,(x_{i_{n_i}}, y_{i_{n_i}}, t_{i_{n_i}})>$ consists of a sequence of n_{i-1} line segments in 3D space, where each segment represents the continuous "development" of the corresponding moving object between two consecutive locations (x_{i_j}, y_{i_j}) and $(x_{i_{j+1}}, y_{i_{j+1}})$ sampled at times t_{i_j} and $t_{i_{j+1}}$. Projecting T_i on the spatial 2D plane (temporal 1D line), we get the *route* r_i (the *lifespan* l_i, respectively) of a moving object. Additional motion parameters can be derived, including the traversed length *len* of *route* r_i, average speed, acceleration, etc.

Let us assume a MOD that stores raw locations of moving objects (e.g. humans); a typical schema, to be considered as a minimum requirement, for such a MOD is illustrated in Figure 3-7.

OBJECTS (id: *identifier*, description: *text*, gender: {M | F}, birth-date: *date*, profession: *text*, device-type: *text*)

RAW_LOCATIONS (object-id: *identifier*, timestamp: *datetime*, eastings-x: *numeric*, northings-y: *numeric*, altitude-z: *numeric*)

MOD_TRAJECTORIES (trajectory-id: *identifier*, object-id: *identifier*, trajectory: *3D geometry*)

Figure 3-7: An example of a MOD.

OBJECTS includes a unique object identifier (id), demographic information (e.g. description, gender, date of birth, profession) as well as device-related technographic information (e.g. GPS type). *RAW_LOCATIONS* stores object locations at various time stamps (i.e., samples), while *MOD_TRAJECTORIES* maintains the trajectories of the objects, after the application of the trajectory reconstruction process.

Following the multidimensional model [AAD+96], a data cube for trajectories consists of a fact table containing keys to dimension tables and a number of appropriate measures. Dimension tables might have several attributes in order to build multiple hierarchies so as to support OLAP analysis whereas measures could be trajectory-oriented (e.g., number of trajectories, number of objects, average speed, etc.). For each dimension we define a finest level of granularity which refers to the detail of the data stored in the fact table.

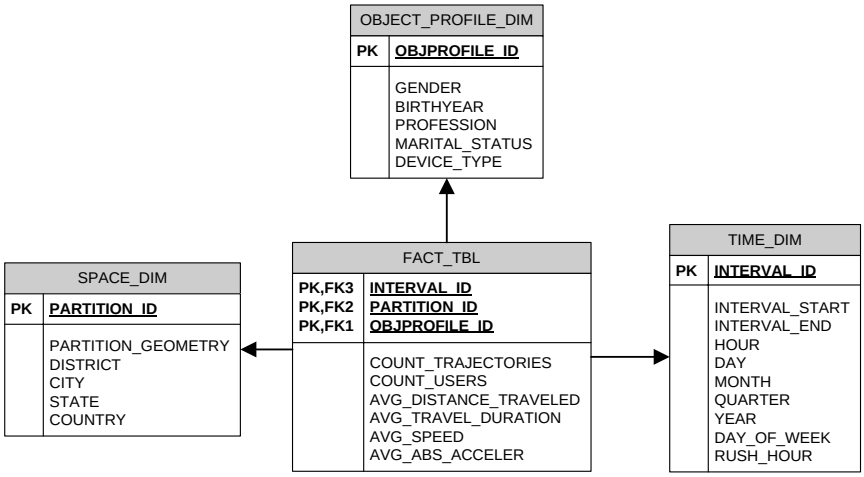


Figure 3-8: An example of a TDW.

Definitely, a TDW should include a *spatial* and a *temporal dimension* describing geography and time, respectively. Another dimension regarding *conventional* information about moving objects (including demographical information, such as gender, age, etc.) could be considered as well.

Based on the above, we consider as a minimum requirement for our framework the following dimensions (Figure 3-8):

- *Geography*: the spatial dimension (SPACE_DIM) allows us to define spatial hierarchies. Handling geography at the finest level of granularity could include (as alternative solutions) a simple grid, a road network or even coverage of the space with respect to the mobile cell network. According to the first alternative, the space is divided in explicitly defined (usually, rectangular) areas. For the purposes of this work, we assume a grid of equally sized rectangles (PARTITION_GEOMETRY in Figure 3-8), the size of which is a user-defined parameter, (e.g. 10×10 Km^2).

- *Time*: the temporal dimension (TIME_DIM) defines temporal hierarchies. Time dimension has been extensively studied in the data warehousing literature [AAD+96]. At the finest level of granularity, we assume user-defined time intervals (e.g. 1 hour periods).

- *User Profile*: the thematic dimension (OBJECT_PROFILE_DIM) refers to demographic and technographic information.

Apart from keys to dimension tables, the fact table also contains a set of measures including aggregate information. The measures considered in the TDW schema of Figure 3-8 include the *number of distinct trajectories* (COUNT_TRAJECTORIES), the *number of distinct users* (COUNT_USERS), the *average traveled distance* (AVG_DISTANCE_TRAVELED), the *average travel duration* (AVG_TRAVEL_DURATION), the *average speed* (AVG_SPEED) and the *average acceleration* in absolute values (AVG_ABS_ACCELER), for a particular group of people moving in a specific spatial area during a specific time period.

3.3.1. ETL Issues

Once trajectories have been constructed and stored in a MOD, the ETL phase is executed in order to feed the TDW. Loading data into the dimension tables is straightforward; however, this process is far more complex when addressing the fact table. In particular, recalling Figure 3-8, the main task is to fill in the measures with the appropriate numeric values for each of the base cells that are identified by the three foreign keys (PARTITION_ID, INTERVAL_ID, OBJPROFILE_ID) of the fact table.

The COUNT_TRAJECTORIES measure for a base cell *bc* is calculated by counting all the distinct trajectory ids that pass through *bc*. The COUNT_USERS measure for a base cell *bc* is calculated similarly by counting all the distinct object ids that pass through *bc*.

In order to calculate the AVG_DISTANCE_TRAVELED measure for a base cell *bc* we define an *auxiliary* measure, called SUM_DISTANCE as the summation of the length *len(TP)* of each portion *TP* of the trajectories lying within *bc*. More formally,

$$SUM\_DISTANCE(bc) = \sum_{TP_i \in bc} len(TP_i) \tag{3-1}$$

Then, the AVG_DISTANCE_TRAVELED measure is computed by dividing the SUM_DISTANCE by the COUNT_TRAJECTORIES measure:

$$AVG\_DISTANCE\_TRAVELED(bc) = \frac{SUM\_DISTANCE(bc)}{COUNT\_TRAJECTORIES(bc)} \tag{3-2}$$

Similar is the case for the AVG_TRAVEL_DURATION measure:

$$AVG\_TRAVEL\_DURATION(bc) = \frac{SUM\_DURATION(bc)}{COUNT\_TRAJECTORIES(bc)} \quad (3\text{-}3)$$

where, SUM_DURATION is also an auxiliary measure defined (3-4) as the summation of the duration *lifespan(TP)* of each portion *TP* of the trajectories inside *bc*.

$$SUM\_DURATION(bc) = \sum_{TP_i \in bc} lifespan(TP_i) \quad (3\text{-}4)$$

In the same fashion, the AVG_SPEED measure is calculated by dividing the auxiliary measure SUM_SPEED (i.e. the sum of the speeds of each portion *TP* inside *bc*) with COUNT_TRAJECTORIES:

$$AVG\_SPEED(bc) = \frac{SUM\_SPEED(bc)}{COUNT\_TRAJECTORIES(bc)} \quad (3\text{-}5)$$

$$\text{where } SUM\_SPEED(bc) = \sum_{TP_i \in bc} \frac{len(TP_i)}{lifespan(TP_i)} \quad (3\text{-}6$$

Likewise, the AVG_ABS_ACCELER is a suchlike fraction

$$AVG\_ABS\_ACCELER(bc) = \frac{SUM\_ABS\_ACCELER(bc)}{COUNT\_TRAJECTORIES(bc)} \quad (3\text{-}7)$$

where SUM_ABS_ACCELER is a supplementary measure that summates the absolute accelerations of all portions *TP* lying in *bc*

$$SUM\_ABS\_ACCELER(bc) = \sum_{TP_i \in bc} \frac{\left| speed_{fin}(TP_i) - speed_{init}(TP_i) \right|}{lifespan(TP_i)} \quad (3\text{-}8)$$

and $speed_{fin}$ ($speed_{init}$) is the final (initial, respectively) recorded speed of the trajectory portion (TP_i) in *bc*.

It is important to remark that all these measures are computed in an exact way by using the MOD. In fact our MOD Hermes [PFG+08] provides a rich palette of spatial and temporal operators for handling trajectories. Unfortunately, rolling-up these measures is not straightforward due to the count distinct problem [TKC+04] is it will be discussed in detail in the next subsection.

As already mentioned, in order to calculate the measures of the data cube, we have to extract the portions of the trajectories that fit into the base cells of the cube. We consider a MOD of *U* user profiles, *N* trajectories, *M* spatial partitions and *K* temporal intervals. We propose two alternative solutions to this problem: (i) a cell-oriented and (ii) a trajectory-oriented approach.

According to the *cell-oriented approach* (COA), we search for the trajectory portions that lie within the base cells. The ETL procedure for feeding the fact table of the TDW is described by the proposed CELL-ORIENTED-ETL algorithm (Figure 3-9). First, we search for the portions of trajectories under the concurrent constraint that they reside inside a spatiotemporal cell *C* (line 4). Then, the algorithm proceeds to the decomposition of the portions with respect to the user profiles they belong to (lines 6-9).

```
Algorithm Cell-Oriented-ETL(D MODTrajectoryTable)
1. // For each pair <Region, Interval> forming a s-t cell Cj
2. FOR EACH cell Cj DO
3.    // Find the set of sub-trajectories inside the cell
4.    S = intersects(D, Cj);
5.    // Decompose S to subsets according to object profile
6.    FOR EACH subset S' of S DO
7.     // Compute the various measures
8.     Compute_Measures(S');
9.    END-FOR
10. END-FOR
```

Figure 3-9: The CELL-ORIENTED-ETL algorithm.

The efficiency of the above described *COA* solution depends on the effective computation of the parts of the moving object trajectories that reside in the spatiotemporal cells (line 4). This step is actually a spatiotemporal range query that returns not only the identifiers but also the portions of trajectories that satisfy the range constraints. To efficiently support this trajectory-based query processing requirement, we employ the TB-tree [PJT00], a state-of-the-art index for trajectories that can efficiently support trajectory query processing.

On the other hand, the *trajectory-oriented approach* (TOA) is described by the proposed TRAJECTORY-ORIENTED-ETL algorithm (Figure 3-10). In TOA, we discover the spatiotemporal cells where each trajectory resides in (line 6). In order to avoid checking all cells, we use (line 4) a rough approximation of the trajectory, its Minimum Bounding Rectangle (MBR), and we exploit the fact that the granularity of cells is fixed in order to detect (possibly) involved cells in constant time. Then, we identify the portions of the trajectory that fits into each of those cells (lines 8-15).

```
Algorithm Trajectory-Oriented-ETL(D MODTrajectoryTable)
1. // For each Trajectory Ti
2. FOR EACH Trajectory Ti of D DO
3.    // Find the Minimum Bounding Rectangle of Ti
4.    MBRTi = Compute_MBR(Ti);
5.    // Find the set of s-t cells C that overlap with the MBR
6.    O = Overlap(C, MBRTi)
7.    // Find the portions (P) of trajectory Ti inside each cell
8.     FOR EACH O' of O DO
9.      P = singlet_intersects(Ti, O');
10.      //If the cell contains portions of the trajectory
11.      IF(P NOT NULL) THEN
12.            // Compute the various measures
13.          Compute_Measures(P);
14.      END-IF
15.     END-FOR
16. END-FOR
```

Figure 3-10: The TRAJECTORY-ORIENTED-ETL algorithm.

3.3.2. OLAP Operations: Addressing the Distinct Count problem

During the ETL process, measures can be accurately computed way by executing MOD queries based on the formulas provided in the previous section. However, once the fact table has been fed, the

trajectory and user identifiers are not maintained and only aggregate information is stored inside the TDW.

The aggregate functions computing the super-aggregates of the measures are categorized by Gray et al. [GCB+97] into three classes based on the complexity required for this computation, starting from a set of already available sub-aggregates. In our case, the aggregate functions to obtain super-aggregates for the main measures discussed in Subsection 3.3.1 are classified as holistic and as such they require the MOD data to compute super-aggregates at all levels of dimensions. This is due to the fact that COUNT_USERS, COUNT_TRAJECTORIES and, as a consequence, the other measures defined in terms of COUNT_TRAJECTORIES are subject to the distinct count problem [TKC+04]: if an object remains in the query region for several timestamps during the query interval, instead of counting this object once, it is counted multiple times in the result.

Notice that once a technique for rolling-up the COUNT_TRAJECTORIES measure is devised, it is straightforward to define a roll-up operation for the AVG measures. In fact the latter can be implemented as the sum of the corresponding auxiliary measures divided by the result of the roll-up of COUNT_TRAJECTORIES. As such, diminishing the calculations in the numerator, hereafter, we focus on the (denominator) number of distinct trajectories (COUNT_TRAJECTORIES); COUNT_USERS is handled in a similar way.

In order to implement a roll-up operation over this measure, a first solution is to define a distributive aggregate function which simply obtains the super-aggregate of a cell *C* by summing up the measures COUNT_TRAJECTORIES in the base cells composing *C*. In the literature, this is a common approach to aggregate spatiotemporal data but, as we will show in Subsection 3.3.3, it produces a very rough approximation. Following the proposal in [OOR+07], an alternative solution is to define an algebraic aggregate function. The idea is to store in the base cells a tuple of auxiliary measures that will help us to correct the errors caused due to the duplicates when rolling-up.

More formally, let $C_{(x,y),t,p}$ be a base cell, which contains, among the others, the following measures (it is worth noting that these measures are loaded without errors into the base cells, by exploiting the MOD functionalities):

- $C_{(x,y),t,p}$.COUNT_TRAJECTORIES: the number of distinct trajectories of profile *p* intersecting the cell ($C_{(x,y),t,p}$.*Traj* for short).
- $C_{(x,y),t,p}$.*cross-x*: the number of distinct trajectories of profile *p* crossing the *spatial* border between $C_{(x-1,y),t,p}$ and $C_{(x,y),t,p}$, where $C_{(x-1,y),t,p}$ is the adjacent cell (on the left) along with *x*- axis.
- $C_{(x,y),t,p}$.*cross-y*: the number of distinct trajectories of profile *p* crossing the *spatial* border between $C_{(x,y-1),t,p}$ and $C_{(x,y),t,p}$, where $C_{(x,y-1),t,p}$ is the adjacent cell (below) along with *y*- axis.
- $C_{(x,y),t,p}$.*cross-t*: the number of distinct trajectories of profile p crossing the *temporal* border between $C_{(x,y),t-1,p}$ and $C_{(x,y),t,p}$, where $C_{(x,y),t-1,p}$ is the adjacent cell (below) along with *t*- axis.

Let $C_{(x',y'),t',p'}$ be a cell consisting of the union of two adjacent cells with respect to a spatial/temporal dimension, for example $C_{(x',y'),t',p'} = C_{(x,y),t,p} \cup C_{(x+1,y),t,p}$ (when aggregating along *x*- axis). In order to compute the super-aggregate corresponding to $C_{(x',y'),t',p'}$, we proceed as follows:

$$C_{(x',y'),t',p'}.Traj = C_{(x,y),t,p}.Traj + C_{(x+1,y),t,p}.Traj - C_{(x+1,y),t,p}.cross\text{-}x \quad (3\text{-}9)$$

The other measures associated with $C_{(x',y'),t',p'}$ can be computed as follows:

$C_{(x',y'),t',p'}.cross\text{-}x = C_{(x,y),t,p}.cross\text{-}x$

$C_{(x',y'),t',p'}.cross\text{-}y = C_{(x,y),t,p}.cross\text{-}y + C_{(x+1,y),t,p}.cross\text{-}y$

$C_{(x',y'),t',p'}.cross\text{-}t = C_{(x,y),t,p}.cross\text{-}t + C_{(x+1,y),t,p}.cross\text{-}t$

The computation of $C_{(x',y'),t',p'}.Traj$ can be thought of as an application of the well-known Inclusion/Exclusion principle for sets: $|A \cup B| = |A| + |B| - |A \cap B|$. Note that in some cases $C_{(x+1,y),t,p}.cross\text{-}x$ is not equal to $|A \cap B|$, and this may introduce errors in the values returned by this algebraic function. In fact, if a trajectory is fast and agile, it can be found in both $C_{(x,y),t,p}$ and $C_{(x+1,y),t,p}$ without crossing the X border (since it can reach $C_{(x+1,y),t,p}$ by crossing the Y borders of $C_{(x,y),t,p}$ and $C_{(x+1,y),t,p}$).

It is worth noticing that the agility of a trajectory affects the error in the roll-up computation. In fact, a trajectory coming back to an already visited cell can produce an error. In the following figures we illustrate the two main kinds of error that the algebraic aggregate function can introduce.

In Figure 3-11a, if we group together the cells C_3 and C_4, we obtain that the number of distinct trajectories is $C_3.Traj + C_4.Traj - C_4.cross\text{-}x = 1+1-0 = 2$. This is an *overestimate* of the number of distinct trajectories. On the other hand, in Figure 3-11b, if we group together C_1 and C_2 we correctly obtain $C_1.Traj + C_2.Traj - C_2.cross\text{-}x = 1+1-1 = 1$, similarly by aggregating C_3 and C_4. However, if we group $C_1 \cup C_2$ with $C_3 \cup C_4$ we obtain $C_1 \cup C_2.Traj + C_3 \cup C_4.Traj - C_1 \cup C_2.cross\text{-}y = 1+1-2 = 0$. This is an *underestimate* of the number of distinct trajectories.

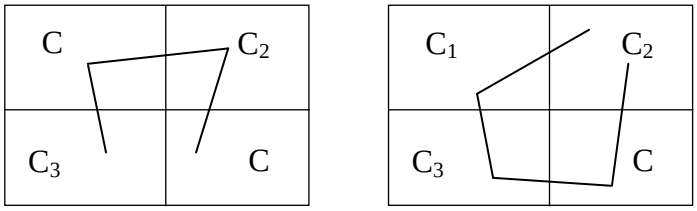


Figure 3-11: a) Overestimate of *Traj*. b) Underestimate of *Traj*.

In order to give a bound to this kind of errors (3-9), let us focus on a single trajectory. This is not a limitation because the values of the measures *Traj*, *cross-x*, *cross-y*, and *cross-t* can be computed by summing up the contributions given to such a measure by each trajectory in isolation. Since the aggregation operations are linear functions, the above property also holds for aggregated cells.

First of all, let us introduce the concept of *uni-octant sequence*. We call *uni-octant sequence* a *maximal* sequence of connected segments of a trajectory whose slopes are in the same octant. It is evident that a trajectory can be *uniquely* decomposed into uni-octant sequences.

A uni-octant sequence *us* can traverse a cell *C* only once, i.e. if *us* starts from *C* it can only exit from *C*, otherwise it can only enter once in *C*. As a consequence, if a trajectory consists of a single uni-octant sequence it does not produce any error in the roll-up computation for the measure

COUNT_TRAJECTORIES. In fact, as discussed above, errors can only arise when a trajectory visits a cell at least twice.

This can be generalized to a trajectory *T* composed by several uni-octant sequences. In this case, the computed value of the measure *Traj* in an *aggregated* cell *C* is limited by the number of uni-octant sequences of *T* intersecting *C*. This is an upper bound that can be reached, as shown in Figure 3-11a.

3.3.3. Experimental Study

In this section, we evaluate the proposed solutions by implementing the TDW architecture (Figure 3-8) for a real-world application. More specifically, we used a large real dataset: a part of the e-Courier dataset [Eco09] consisting of 6.67 millions of raw location records (a file of 504 Mb, in total), that represent the movement of 84 couriers moving in greater London (covered area 66,800 km^2) during a one month period (July 2007) with a 10 sec sample rate. Figure 3-12 illustrates some snapshots of the dataset. For all the experiments we used a PC with 1 Gb RAM and P4 3 GHz CPU.

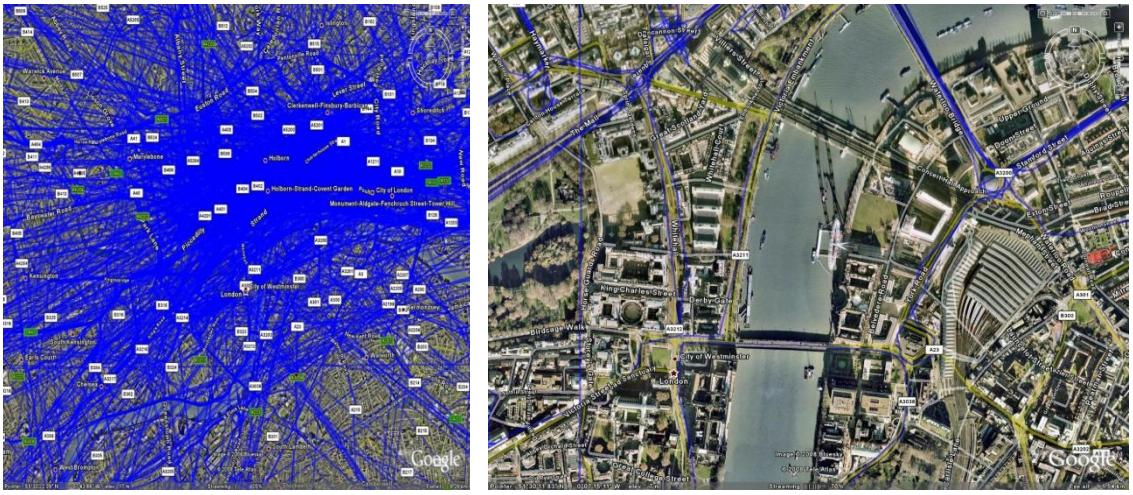


Figure 3-12: a) The complete dataset b) Zooming over Thames.

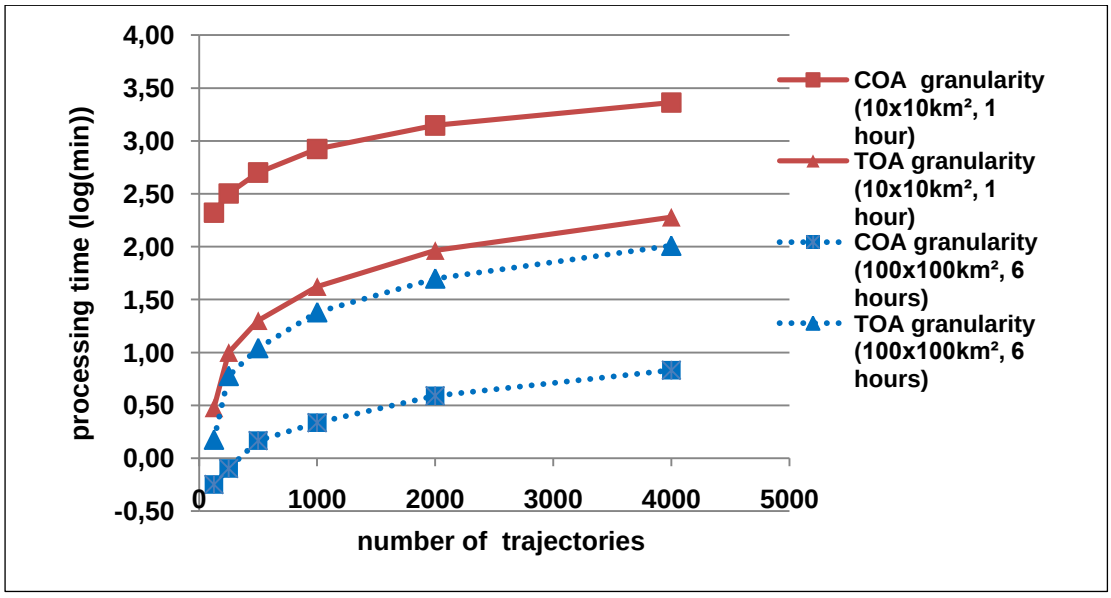


Figure 3-13: Comparison of alternative ETL processes.

For the evaluation of the ETL process we compared the performance of the TOA vs. the index-based COA approaches. Both approaches are implemented on the MOD system, Hermes, presented in [PTV+06], [PFG+08]. We used two different granularities to partition the spatial and the temporal hierarchies; a spatial grid of equally sized squares of 10×10 Km^2 (100×100 Km^2, respectively) and a time interval of one (six, respectively) hours. The results of the four cases are illustrated in Figure 3-13, where it is clear that the choice of a particular method is a trade-off between the selected granularity level and the number of trajectories.

We complete the experimental study with some results on the trajectory aggregation issue (Figure 3-14). We would like to assess the accuracy of the approximations of the measure COUNT_TRAJECTORIES computed in roll-up operations by using the distributive and the algebraic functions presented in Subsection 3.3.2. To this aim we consider the normalized absolute error proposed by Vitter et al. [VWI98]: For all the OLAP queries q in a set Q we define this error as follows:

$$Error = \frac{\sum_{q \in Q} |\overline{M_q} - M_q|}{\sum_{q \in Q} M_q} \tag{3-10}$$

where $\overline{M_q}$ is the approximate measure computed for query q, while M_q is its exact value.

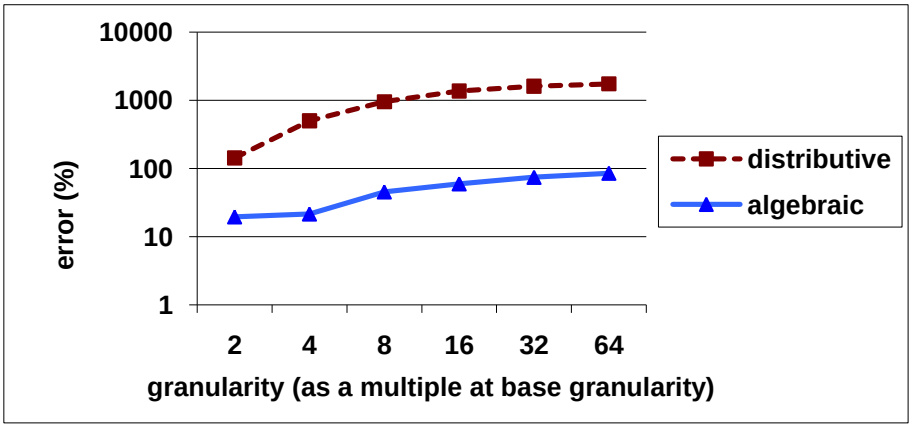


Figure 3-14: Distributive vs. algebraic aggregate functions (base granularity set to 10×10 Km^2 and 1 hour time interval).

We assume, as base granularity g, a spatial grid of equally sized squares of 10×10 Km^2 and a time interval of one hour. Then our queries compute the measure TRAJECTORIES for larger granularities $g' = n \times g$, with $n > 1$.

The distributive aggregate function has an error which always exceeds 100% and quickly grows as the roll-up granularity increases. Instead, as expected, the computations based on the algebraic function are always more precise than those based on the distributive one and they are accurate for small granularities. Still, the error grows up for large granularities but it never exceeds 100%. Although the corresponding experiments are not reported here, it is worth noting that starting from smaller base granularities g and using the algebraic function we get a better accuracy, with errors under 10% for small multiples of g.

3.4. Ad-hoc OLAP on Trajectory Data

In this section, we describe an innovative organization of a trajectory data cube in order to be able to answer OLAP queries considering different interpretations of the notion of trajectory as presented in [MT09a]. Thus, ad-hoc analysis on trajectory data cubes can be achieved, which can be really useful for a number of applications. Preliminary experimental results illustrate the applicability and efficiency of our approach.

Nevertheless, trajectory analysis is based on the specific requirements of each application. For instance, there may be a considerable difference on the semantic definition of a trajectory given by a traffic analyst and, on the other hand, a logistics manager. Let us consider a fleet of trucks moving in a city and delivering goods in various locations. For each truck, the logistic manager may consider a number of different trajectories (e.g. between the different delivery points) while the traffic analyst may consider a single trajectory for the whole day. Thus, in order to satisfy these two, quite different in semantics, needs we would have to retrieve raw location data from a common repository and, then, execute two different reconstruction tasks so as to produce trajectories that are semantically compliant to each domain. After that step, we would also have to build two different trajectory data cubes in order to allow users to apply OLAP techniques oriented to their purposes.

For instance, Figure 3-15a illustrates a raw dataset of time stamped locations. Different analytical needs may result to different set of reconstructed trajectories (Figure 3-15b-d, respectively). Recalling the previous example of the truck dataset, les us consider that Figure 3-15b and 1c illustrate the reconstructed trajectories for the logistic manager and for the traffic manager respectively. Another example of trajectory reconstruction is presented in Figure 3-15d which considers a compressed trajectory of the movement.

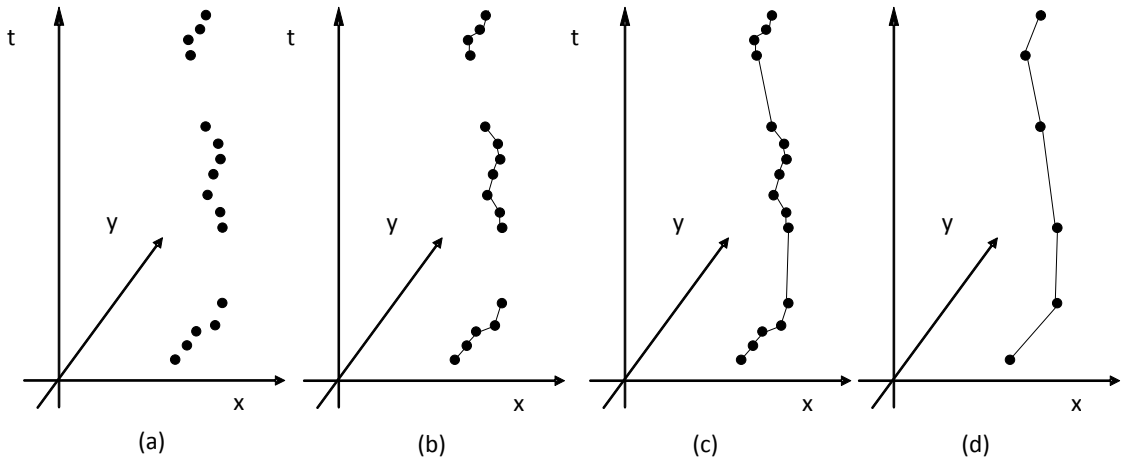


Figure 3-15: Different trajectory reconstruction approaches (b, c, d) for a raw dataset (a).

If we follow (e.g. [MFN+08a]) the conventional approach of building a TDW, for each set of reconstructed trajectories we would have to repeatedly execute an ETL process so as to build different trajectory data cubes. This is clearly presented in Figure 3-16 (traditional approach) where different

semantic definitions *(def*$_1$*, …, def*$_n$*)* demand different executions of the trajectory reconstruction process *(TR*$_1$*, …, TR*$_n$*)* resulting in different TDWs and trajectory data cubes, respectively..

In fact, this is unusual in other (conventional or not) data warehousing scenarios. For instance, the same sales data warehouse can be used by both salesmen and marketers because they both agree that a sale has some specific, though widely accepted characteristics. In scientific databases, the same seismic DW [MTK08] can be used by both a scientist and a public administration officer in order to explore seismic activity under a commonly accepted definition of earthquake. Even in latest applications of data warehousing techniques on workflow data [Kot06] the data cube is built on top of well defined facts (service provisions).

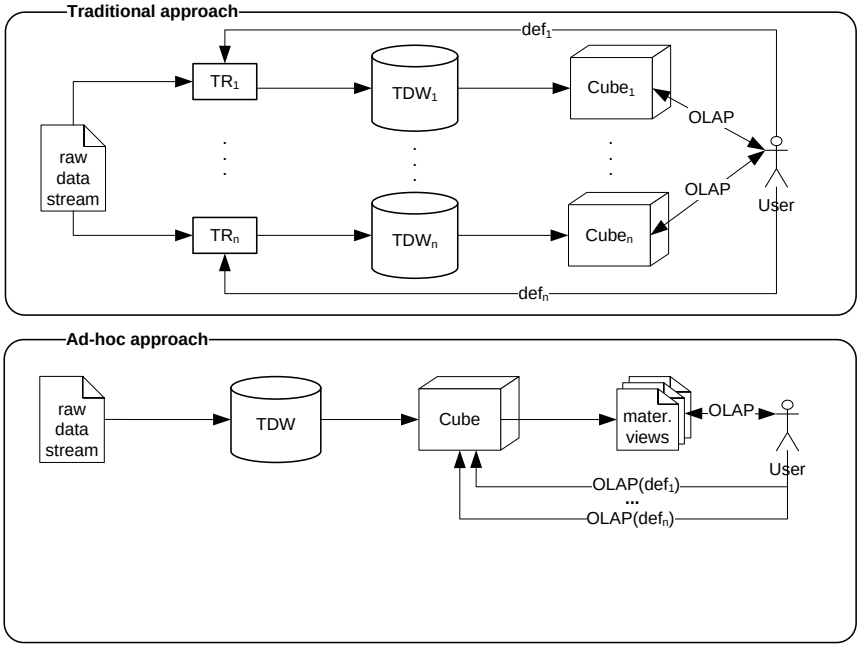


Figure 3-16: Traditional vs. ad-hoc approach.

On the other hand, the space-time nature of trajectory data allows different semantic definitions of trajectories. Intuitively, we have to revisit basic structures (fact table, dimensions, ETL, cube materialization) of DW, and build more flexible mechanisms so as to satisfy this challenging request. Furthermore, it is important to extend traditional OLAP techniques to be ad-hoc in order to handle appropriately the dynamic nature of trajectories. Our target is illustrated in Figure 3-16 (ad-hoc approach) where different semantic definitions of trajectories (*def*$_1$, …, *def*$_n$) are applied on the same cube. In case of known semantic definitions, the cube can be materialized and get the functionality described by the traditional approach.

To the best of our knowledge, this is the first work considering ad-hoc analysis in a TDW. The contribution of this section can be summarized as follows:

- We extend the OLAP data model for TDW in two ways. First, we propose a flexible fact table that will be able to answer queries considering different semantic definitions of trajectories.

Second, we introduce a parameter that supports the choice of semantics for aggregation queries over trajectory data.

- We present a suitable ETL method loading raw location data in our flexible data cube.
- We enhance OLAP techniques so as to utilize the new ad-hoc approaches. An efficient algorithm is proposed that takes advantage of our model so as to answer aggregation queries.
- To speed up the calculation process, we discuss cube materialization issues that pre-calculate results for known semantic definitions of trajectories.

3.4.1. Problem Definition

Following the relational implementation [Kim96] of the multi-dimensional model [AAD+96] for data warehousing, a TDW consists of a fact table containing keys to dimension tables and a number of appropriate measures. Dimension tables might have several attributes in order to build multiple hierarchies so as to support OLAP analysis. For each dimension, the finest level of granularity is defined, which refers to the detail of the data stored in the fact table. The multidimensional form of a data warehouse, known as data cube, can be thought of as a multidimensional array of numeric attributes on which aggregate functions are applied. More formally:

Definition 3-1 (Dimension table): A *dimension table* is a m-ary relation over $F \times A_1 \times A_2 \times \ldots \times A_v$, where:

i. F is the primary key of the dimension table;
ii. Each column A_j, $0 \leq j \leq v$ is a set of attributes values;
iii. $m = 1+v$. ■

Definition 3-2 (Fact table): A *fact table* is an n-ary relation over $K \times M_1 \times M_2 \times \ldots \times M_r$, where:

i. K is the set of attributes representing the primary key of the fact table formed by $F_1 \times F_2 \times \ldots \times F_p$, where each F_i, $1 \leq i \leq p$ is a foreign key to the dimension tables;
ii. Each column M_k, $1 \leq k \leq r$ is a set of measures that can be computed using aggregate functions on the characteristics of the raw trajectories;
iii. $n = p+r$. ■

The TDW presented in Section 3.3 follows the aforementioned definitions. However, implementing a TDW using the above approach does not take into consideration the different semantic definitions of trajectories. It assumes that trajectories have been reconstructed on an earlier phase following a specific trajectory definition and that its measures have been computed on this basis. We adopt the measures of the TDW of Section 3.3, a subset of them is also discussed in [OOR+07], as a running example in the remainder of this subsection where we discuss how the TDW can be transformed so as to consider multiple semantic definitions.

Before proceeding to the core of this new approach, let us describe the notion of different semantic definitions of trajectories. Assuming that a sequence of time-stamped locations has been recorded for a moving object (using e.g. a GPS device), the movement of this object can defined as

$$M = ((x_1, y_1, t_1), ..., (x_n, y_n, t_n)) \quad (3\text{-}11)$$

After trajectory reconstruction, this movement can result into a set of trajectories:

$$M' = (T_1, ..., T_j) \quad (3\text{-}12)$$

with each trajectory T_i defined as [GBE+00]:

$$T_i = < (x_1, y_1, t_1), ..., (x_k, y_k, t_k) > \quad (3\text{-}13)$$

where $1 \leq k,\ m \leq n$, i.e. i.e. trajectory points are points in the corresponding raw dataset or in other words, $\bigcup_i T_i \subset M$.

Two basic functions that are usually provided by MODs and can be applied on trajectories are *len*(T_i) and *lifespan*(T_i), which return the length of the (2D) spatial projection of T_i and its duration in (1D) time, respectively.

Based on the aforementioned model, a set *M* of raw time-stamped location points can result in a set *M'* of trajectories. The exact number of reconstructed trajectories depends on the different semantic definitions that can be given to a trajectory. For instance, let us consider someone who drives in the morning from her home to office, works for eight hours, and then returns home after a short stop for shopping. Different applications may consider a different number of trajectories in this case. Spaccapietra et al. [SPD+08] argue that different time granularities result in different semantic definitions of trajectories. For instance, in the previous example, there is not a single answer on the question *"How many trajectories exist?"* as it depends on the time granularity level which we are interested in. One might think of either one or two or three trajectories (e.g., setting the time granularity at the level of day, hour, minute, respectively). Authors in [SPD+08] discuss about trajectory semantics and recognize the need of enriching the underlying spatiotemporal data model with maximum flexibility. The idea behind this is the adaptation of the model to the specific semantics of trajectories in a given application context. The approach discussed in this paragraph is to model trajectories as a sequence of moves going from one stop to the next one (or as a sequence of stops separating the moves). This approach is based on the fact that trajectories may themselves be semantically segmented by defining a temporal sequence of time subintervals (moves) where alternatively the object position changes and stays fixed (stops). Based on this approach, Baglioni et al. [BMR+08] propose the enhancement of raw trajectories with semantic information by exploiting some domain knowledge encoded in an ontology.

The problem discussed in this section is that of building a flexible TDW that enables OLAP analysis on mobility data by allowing the specification of a particular semantic definition of trajectory only when an aggregation query is posed. In other words, we study the problem of discovering the set of trajectories M' during the execution of an aggregation query and computing numerical measures on this set of trajectories. This task is multi-fold as it involves several issues:

- *TDW modeling*: a new flexible model should encapsulate the necessary flexibility regarding different semantic definitions of trajectories;

- *ETL process*: an appropriate mechanism feeding the TDW should comply with the above model;
- *OLAP technology*: it should be enhanced so as utilize the new TDW model, thus allowing users to select the level of time granularity which they are interested in;
- *Cube materialization*: performance should be taken into consideration in order to increase the query response time of the TDW.

All the above issues are addressed in the section that follows, which describes in detail the proposed ad-hoc OLAP approach.

3.4.2. A Framework for ad-hoc OLAP

The challenge of building a TDW suitable for ad-hoc analysis introduces three challenges regarding the fact table: a) the adoption of a flexible structure that will allow ad-hoc analysis b) the computation of the measures during the loading phase of the cube (ETL) and c) the aggregation issues after the fact table has been loaded. In the following subsections, we present our solutions regarding the above issues, and we also present a modeling approach so as to express, in the TDW, different trajectory forms.

3.4.2.1. The model

As we mentioned in the previous section, a fact table consisting of a number of (trajectory-oriented) measures assumes that the definition given to trajectories is a-priori known. In this subsection, we propose a more flexible structure that will allow the user to decide upon the characteristics of trajectories. In particular, we focus on the measures that their computation involves the spatial and/or temporal distances between the points of each trajectory, and we propose the transformation of movement as *a sequence of spatiotemporal distances between consecutive points*. Formally, the movement of a moving object is formulated as:

$$M_i^{''} = <(0,0),(sd_2,td_2),...,(sd_n,td_n)> \quad (3\text{-}14)$$

where (sd_i, td_i) represent the (Euclidean) spatial (*sd*) and temporal distance (*td*) between the i^{th} and the $(i\text{-}1)^{th}$ point; the first pair is always set to (0, 0) as there is no previous point to compare it with. Obviously, $M_i^{''}$ is a lossy representation with regard to *M* or *M'*. However, as it happens in conventional data management world, DWs store only the necessary information so as to answer aggregation queries and they do not substitute DBs which keep the full details. Similarly, in TDWs we are allowed to store a loosy representation of movement just for the purpose of offering OLAP analysis on mobility data and *M* and *M'* can be stored in a MOD.

Based on this formulation, we replace the traditional fact table as defined in the previous section by introducing the notion of *TrajFact* table, which includes keys to the dimension tables as well as the minimum level of information that can be used for computing the measures using different interpretations of the term "trajectory". Keep in mind that the different combinations of keys in the fact table define a unique position in the multidimensional space of the cube (also known as *base cells*). In

this position, instead of storing measures, for each object that 'contributes', we can maintain a sequence of temporal and spatial distances between its points (see Figure 3-17). Formally:

Definition 3-3 (TrajFact table): the *TrajFact* table is an n-ary relation over $K \times DT$, where:

i. K is the set of attributes representing the primary key of the fact table formed by $F_1 \times F_2 \times \ldots \times F_p$, where each F_i, $1 \leq i \leq p$ is a foreign key to the dimension tables;

ii. A distance table which consists of:

- Object identification O_j, for all objects that contribute to the unique position in the multidimensional space of the cube that is defined by the primary key;
- A sequence of pairs (sd_i, td_i), where sd_i is the Euclidean distance and td_i the temporal distance between two successive time-stamped location points (x_i, y_i, t_i) and the $(x_{i-1}, y_{i-1}, t_{i-1})$ *point* of object O_j.

iii. $n = p+1$. ■

The above is graphically presented in Figure 3-17, $(F_1, \ldots, F_p)$ represent a random combination of dimension keys and for this row a distance table is defined.

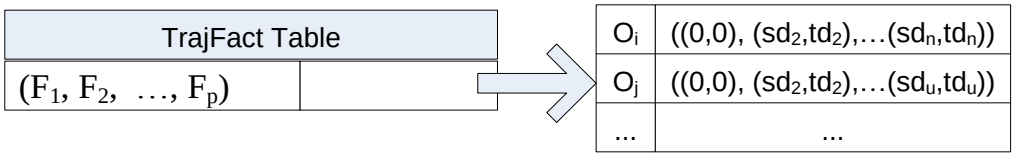


Figure 3-17: The TrajFact Table.

One could argue that spatial/temporal distance between consecutive points is derived information, thus we could store in the fact table the points (x_i, y_i, t_i) themselves. This is true but, the two main concerns when building a data warehouse are: a) the size of the DW and b) the query response time. Storing distances (sd_i, td_i) only, allows us to save 1/3 (if we stored points) or almost 2/3 (if we stored segments) of space and at the same time providing us the information we need in order to efficiently calculate measures.

Next, we need a way to model the different semantic interpretations of trajectories. For this purpose, we do not propose yet another model but we follow 0, which considers the temporal distances between consecutive points as a way to identify trajectories from a raw dataset. Hence, our model allows the user to select the maximum allowed time interval for defining a particular form of trajectory:

sem_{time}: The *maximum allowed time interval* between two consecutive time-stamped positions of the same moving object.

In other words, for each time-stamped location position a temporal period is defined so as to examine whether the next time-stamped location can be considered as part of the same trajectory or not. The value of this parameter is chosen by the user so as to specify the time granularity level, and therefore a particular interpretation of the notion of trajectory. The choice is made during aggregation querying; as such, it does not affect the overall organization of the data cube.

The value of sem_{time} is used in aggregation queries when it is needed to identify trajectories of objects that lie in each base cell. As discussed in Subsection 3.4.2.3, this can be achieved by examining the distance tables of base cells. This process results in discovering the set of trajectories in a dynamic way. Hence, the functions *len(T)* and *lifespan(T)* can be also applied in the discovered set of trajectories but our approach allows a different computation technique. In particular, the two functions summarize the spatial and the temporal distances respectively stored in the distance tables.

Our model utilizes the notion of dimension table as it was defined in the previous section. Hence, the schema of an ad-hoc TDW can be defined as:

Definition 3-4 (Schema of ad-hoc TDW): The schema of an ad-hoc TDW (*adhocTDW*) can be defined as *adhocTDW = (DT, TFT),* where *DT* is a nonempty finite set of dimension tables defined according to Definition 3-1 and *TFT* is the *TrajFact* table defined according to Definition 3-3. ■

3.4.2.2. ETL processing over trajectory data

An ETL process is executed in order to feed the ad-hoc TDW. Loading data into the dimension tables is straightforward; however, we should pay attention on loading the *TrajFact* table. This is because, instead of applying a reconstruction task during the ETL phase (or in a previous phase, during the MOD loading), we transform the data appropriately in order to load them in the fact table.

Using previous approaches [MFN+08a], [OOR+07], the computation of almost every measure (except the COUNT_USERS) assumes a specific semantic definition of trajectory. For instance, the computation of COUNT_TRAJECTORIES needs a specific definition of trajectory based on which the number of trajectories will be counted.

We propose a more flexible ETL strategy based on the model described in the previous section. In particular, we only need to calculate the temporal/ spatial distances between consecutive locations of the same object. As we already mentioned, the different combinations of keys in the fact table define a unique cell in the multidimensional space of the cube. Hence, the aim of the ETL procedure is to find which points reside in this cell and compute the distance tables. In this work, we don't consider noisy data and we assume that we work on a clean dataset, thus we compute meaningful distances.

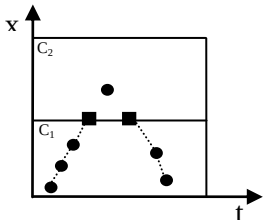


Figure 3-18: Applying linear interpolation.

Note that in order to study the movement of objects inside the cells, we have to identify the cross points of each base cell. Hence, we assume a *linear* interpolation function that considers a straight movement with constant speed [PJT00]. For instance, in cell C_1 of Figure 3-18 the square points represent the interpolated cross points of a particular object. Note that we only consider spatial/temporal distances between points that are inside C_1 and, obviously, we do not introduce distances between cross points (to be precise, we set the temporal and spatial distance between the two cross points to zero).

The above approach is incorporated into LOAD-TDW-ETL algorithm, which is illustrated in Figure 3-19. The algorithm searches for the set of points which reside into each base cell (line 4). Finally, for each list of points belonging to the same object, the spatial (using Euclidean distance) and the temporal distance are computed (line 8).

```
Algorithm Load-TDW-ETL(ListOfPoints LoP)
1. //We assume LoP is sorted by Object-id, timestamp
2. FOR EACH base cell bc_j DO
3. //Find the set of points inside the cell
4.  S = contains(LoP, bc_j);
5.    //Consider a list of points LP for
6.    //each object of S
7.    FOR EACH LP of S DO
8.     Compute_Distances(LP);
9.    END-FOR
10. END-FOR
```

Figure 3-19: The Load-TDW-ETL algorithm for loading the fact table.

3.4.2.3. Ad-hoc OLAP

The proposed TDW organization needs a new OLAP mechanism that utilizes the notion of multiple semantic definitions of trajectories. In this paragraph, we present an algorithm that incorporates this capability and performs aggregation queries in an efficient way.

A query posed towards the TDW contains a number of members of dimensions based on which the base cells are filtered and a subset is selected, and a measure which implies an aggregate function (e.g. SUM, MIN, MAX, AVG, COUNT, DISTINCT COUNT). Furthermore, it contains a value for the sem_{time} parameter, which will be used to identify different trajectories. We need it because each base cell contains no information about trajectories but includes a table containing spatial/ temporal distances between consecutive points of each object. Formally:

Definition 3-5 (Aggregation query): is a set of $\{(bc_1, bc_2, \ldots, bc_i), tm, mtd\}$, where:

i. Each bc_j, $1 \leq j \leq i$ is a base cell that was filtered based on the selected members of dimension (which result in specific rows in the *TrajFact* table)
ii. *m* is the measure
iii. *mtd* is the selected value for the parameter sem_{time} ■

In this work, we propose AD-HOC-AGGREGATION algorithm that is suitably designed to answer aggregate queries and incorporates the ad-hoc approach. There, the user chooses $sem_{time} = mtd$ and the OLAP query returns a set *SOC* of base cells .

In detail, AD-HOC-AGGREGATION algorithm examines the table containing the temporal distances in each cell and, for each object (lines 3-13), searches for temporal distances with values lower than *mtd* (line 5). This way, the algorithm identifies that the same trajectory evolves and so it updates the measures (line 7) applying the corresponding aggregate function *TM*. If it is found a temporal distance with value greater that *mtd* then a new trajectory is identified.

```
Algorithm General-Aggregation(SetOfCells SOC,
TargetMeasure TM, MaxTemporalDistance MTD)
1.  FOR EACH cell C of SOC DO
2.  //Process the distance table DT of the cell C
3.   FOR EACH object P of C DO
4.    FOR EACH TemporalDistance t of P in DT DO
5.     IF t < MTD THEN
6.      //the same trajectory evolves
7.      UpdateMeasure(TM);
8.     ELSE
9.      //a new trajectory is identified
10.      newTrajectory();
11.    END-IF
12.   END-FOR
13.  END-FOR
14. END-FOR
```

Figure 3-20: The AD-HOC-AGGREGATION algorithm.

The aggregate functions computing the super-aggregates of the measures are categorized by Gray et al. [GCB+97] into three classes (*distributive*, *algebraic* and *holistic*) based on the complexity required for this computation, starting from a set of already available sub-aggregates. The same classification is followed by our OLAP mechanism. In the remaining paragraphs of this subsection, we discuss computation issues, using our ad-hoc framework, for the measures of [MFN+08a] and [OOR+07] and we classify them to categories according to their complexity [GCB+97].

The COUNT_TRAJECTORIES measure is classified as distributive, and calculated by executing the AD-HOC-AGGREGATION algorithm on the *SOC* set of base cells. At this point, we should mention that, in this work, the COUNT_TRAJECTORIES measure refers to the *number of trajectories* and not to the *number of distinct trajectories* that is discussed in [MFN+08a], [OOR+07]. This happens because in our approach no trajectories have been reconstructed in an earlier phase, thus it does not make sense to search for distinct trajectories in base cells, which is the source of the distinct count problem [TKC+04]. Nevertheless, we discuss a different meaning of number of distinct trajectories as a piece of future work.

The COUNT_USERS measure for a base cell *bc*, refers to the *number of distinct objects*, and is calculated by counting the number of rows of the distance table of *bc* since each row corresponds to a different object. However, this measure can be classified as holistic as we cannot just count the results of each base cell. This happens because an object may appear in more than one distance tables (base cells), as such we have to avoid counting it more than one time.

The remaining measures are classified as algebraic as they are calculated on the basis of other measures. In order to calculate the AVG_DISTANCE_TRAVELED when can use the Formula (3-2). However, SUM_DISTANCE is defined as the summation of the length *len(T)* of each trajectory *T* lying within a base cell *bc*. More formally:

$$SUM\_DISTANCE(bc) = \sum_{T_i \in bc} len(T_i) \tag{3-15}$$

Therefore, the function *len(T)* is not applied on the trajectory portion that lies inside *bc* but on the trajectory T that has been discovered in a dynamic way as we have already described.

Similar is the case for the AVG_TRAVEL_DURATION measure which can be discovered using the equation (3-3) but SUM_DURATION is also an auxiliary measure defined as the summation of the duration *lifespan(T)* of each trajectory *T* inside *bc*.

$$SUM\_DURATION(bc) = \sum_{T_i \in bc} lifespan(T_i) \tag{3-16}$$

In the same fashion, the AVG_SPEED measure (equation (3-5)) is calculated by dividing the auxiliary measure SUM_SPEED (i.e. the sum of the speeds of each trajectory *T* inside *bc*) with COUNT_TRAJECTORIES:

$$SUM\_SPEED(bc) = \sum_{T_i \in bc} \frac{len(T_i)}{lifespan(T_i)} \tag{3-17}$$

Likewise, the AVG_ABS_ACCELER is a suchlike fraction (equation (3-7)) where SUM_ABS_ACCELER is a supplementary measure that summates the absolute accelerations of all trajectories *T* lying in *bc*

$$SUM\_ABS\_ACCELER(bc) = \sum_{T_i \in bc} \frac{\left| speed_{fin}(T_i) - speed_{init}(T_i) \right|}{lifespan(T_i)}$$

and $speed_{fin}$ ($speed_{init}$) is the finally (initially, respectively) recorded speed of the trajectory (T_i) in *bc*.

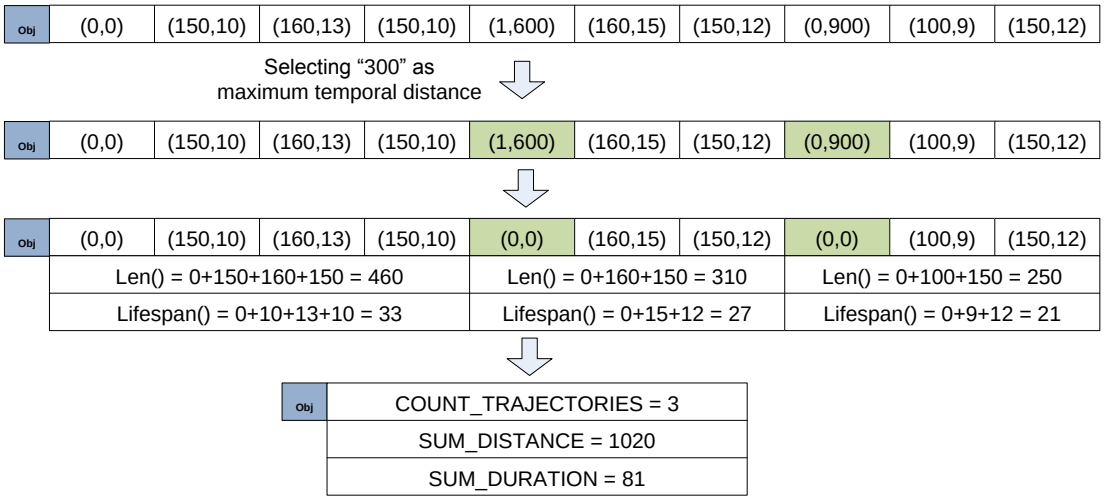


Figure 3-21: Computing COUNT_TRAJECTORIES, SUM_DISTANCE and SUM_DURATION measures.

Let us present our ad-hoc OLAP operations through an example. Let us consider that a user requests the total distance covered by the trajectories (SUM_DISTANCE), the total duration of the trajectories (SUM_DURATION) and the number of trajectories (COUNT_TRAJECTORIES) under specific spatiotemporal constraints. The user defines the maximum temporal distance as 300 secs. Figure 3-21 illustrates the complete process that is executed for the computation of the measures SUM_DISTANCE and SUM_DURATION using the distance table of a base cell. The first table contains the spatial (in meters) and temporal distances (in seconds) of each point from its previous one. As a first step, the algorithm locates the points with temporal distance greater that 300 secs. In the example of Figure 3-21 there are

exist two such points. Therefore, these points will be the start points of trajectories (so their temporal and spatial distances are set to zero). Hence, three trajectories are identified and then their measures are computed.

3.4.2.4. Discussion about data cube materialization

In the previous subsections, we presented our TDW architecture that enables ad-hoc OLAP analysis on mobility data. Obviously, there is a trade-off between flexibility and performance. A TDW based on predefined and reconstructed trajectories may perform faster than our ad-hoc TDW. However, the former does not offer flexibility regarding different semantic definitions of trajectories. In traditional DW, materialized views have been proposed to speed up query processing. These views are referred as *summary tables* [CD97] that store redundant, aggregate information. The advantage of materialized views is their small size (compared with the detailed records) that allows much faster query response.

Following the aforementioned approach, we define an appropriate summary table for our TDW with rows of the form:

Definition 3-6 (Summary table): A summary table *ST* is a quadruple *{bc, mtd, m, value}*, where:

i. *bc* is a particular base cell;

ii. *mtd* is a specific maximum temporal distance;

iii. *m* is a measure; and

iv. *value* is the pre-calculated aggregated value for *m*. ■

Obviously, this strategy performs well only in the case of distributive and algebraic (that are not based on holistic) functions. This happens because in case of a holistic measure (e.g. COUNT_USERS) the super-aggregates cannot be computed from sub-aggregates.

The most trivial way to define *mtd* is to a-priori know it. More specifically, in case that some semantic definitions of trajectories are a-priori known, we can materialize the calculations of the measures so as to speed up query response.

We provide this short discussion in order to emphasize on the fact that materializing our flexible data cube it is possible to get the functionality described by the cubes proposed in [MFN+08a] and [OOR+07]. This happens because, in this case, the possible answers are already computed as it is proposed in those two works. In other words, the data cubes described in those two papers can be considered as specific materialized data cubes using our approach.

3.4.3. Experimental Study

In this section, we evaluate the proposed solutions by using the same dataset ([Eco09]) with the one in Section 3.3.3. It contains 6.67 millions of raw location records that represent the movement of 84 couriers moving in London during a one month period (July 2007) with a 10 sec sample rate. For all the experiments we used a PC with 1 Gb RAM and P4 3 GHz CPU.

In the following, we evaluate the performance of the basic components (ETL, computation methods, data cube size) of our *ad-hoc approach* that allows building of flexible trajectory data cubes. Moreover,

we provide a comparison with the respective components of TDW that are proposed in [MFN+08a], [OOR+07]. In this study, we refer to the latter approach as *static* approach. To achieve this comparison we use as TDW model the one presented in [MFN+08a] (the model of [OOR+07] can be considered as subset of it) consisting of a spatial, a temporal, and an object profile dimension, as well as the measures that have already been discussed in Subsection 3.4.2.3.

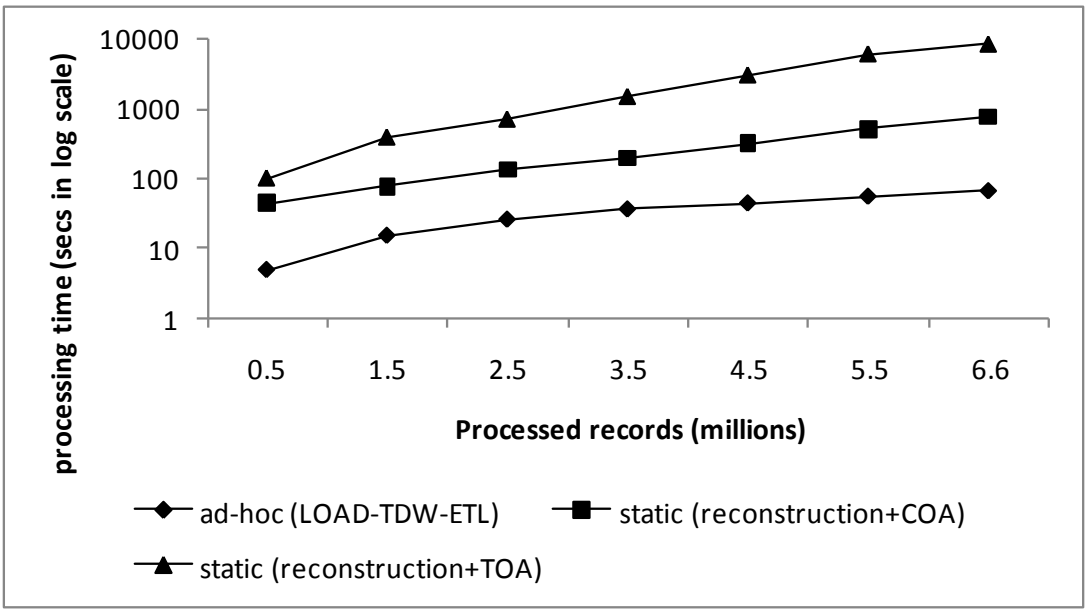


Figure 3-22: Performance of Load-TDW-ETL algorithm and comparison to the static ETL.

First, we evaluate the effectiveness of the LOAD-TDW-ETL algorithm (Figure 3-22). It is clear that it performs linear with the size of the input dataset (and allows the processing of the full dataset in about 1 min). The ETL step of the *static approach* includes a trajectory reconstruction task and the feeding of the data cube, which follows either a cell (*COA*) or a trajectory oriented approach (*TOA*). Obviously the proposed ad-hoc approach performs better than the *static* as no reconstruction work takes place and also the feeding of the data cube does not require the discovery of portions of trajectories that lie inside base cells.

Second, we evaluate the time required for the computation of different measures (using the AD-HOC-AGGREGATION algorithm). We have experimented with different sizes of the input datasets that result in a different number of base cells that have to be processed. Figure 3-23 presents computation times regarding the highest level cuboid of the lattice (left vertical axis). Furthermore, the number of processed base cells is illustrated on the right vertical axis so as to provide a view on the data cube size. The two distributive measures (SUM_ DURATION and COUNT_TRAJECTORIES) present a similar behavior. The latter has a slightly better performance as in this case the algorithm identifies only the different trajectories and does not compute anything else (as it happens in the case of SUM_DURATION where distances are summarized). AVG_DURATION is an algebraic measure, so its performance is in direct proportion to SUM_DURATION and COUNT_TRAJECTORIES. Although the measure COUNT_USERS is holistic, its performance is very good as it utilizes the structure that was proposed in Subsection 3.4.2.1. Furthermore, its performance is affected by the number of base cells and not by the number of records.

This is due to the fact the computation of this measure considers only the object ids of the distance table of each base cell.

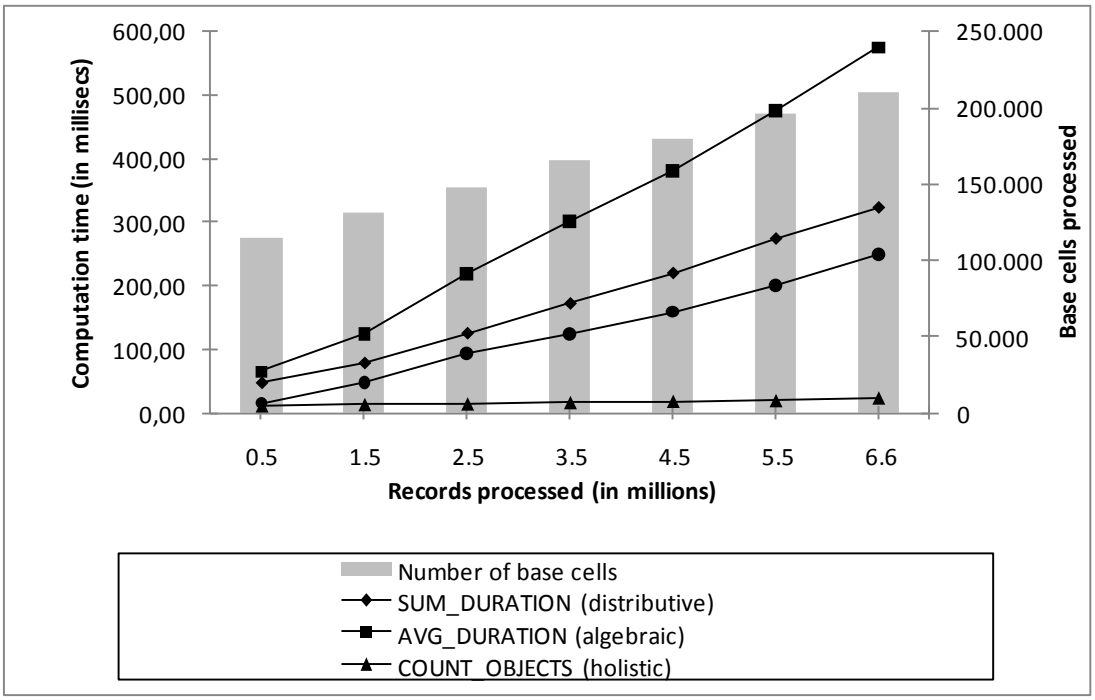


Figure 3-23: Computation time using the ad-hoc approach.

Figure 3-24 illustrates the computation of a distributive and an algebraic measure using our flexible data cube (not materialized) and a typical materialized data cube using either the summary tables as discussed in Subsection 3.4.2.4 or the *static* cubes. Obviously, the materialized cube performs better as the measure values are pre-calculated but it does not provide any flexibility regarding different definitions of trajectories (this is the trade-off between flexibility and performance). We do not provide a comparison regarding holistic measures because we cannot utilize the summary tables for their computation and also the cubes of [MFN+08a], [OOR+07] do not support such measures (they only support distributive and algebraic measures).

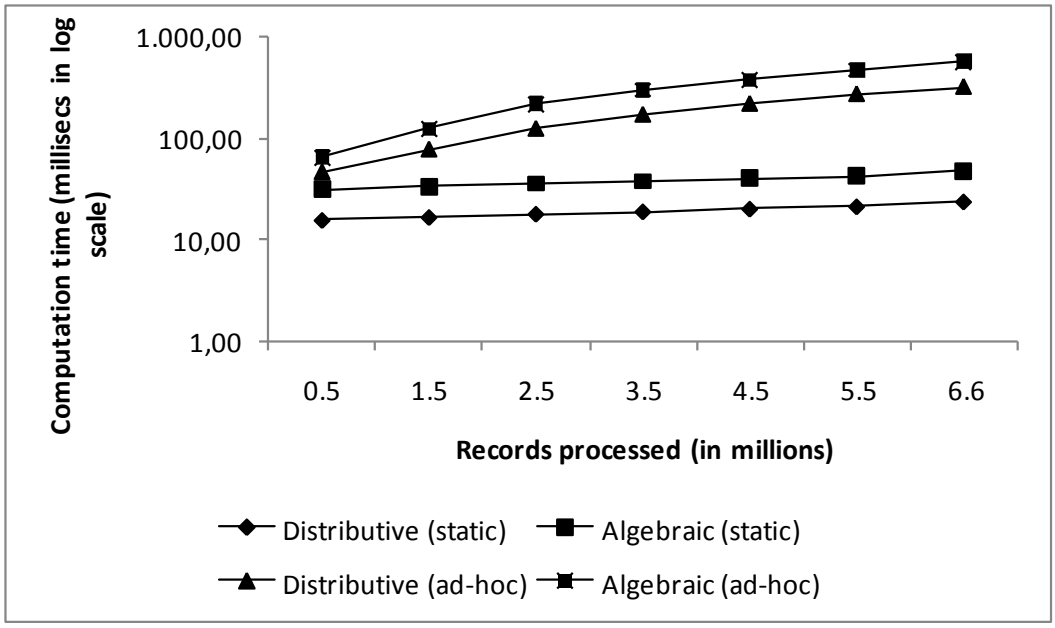


Figure 3-24: Comparing computation times: ad-hoc vs. static approach

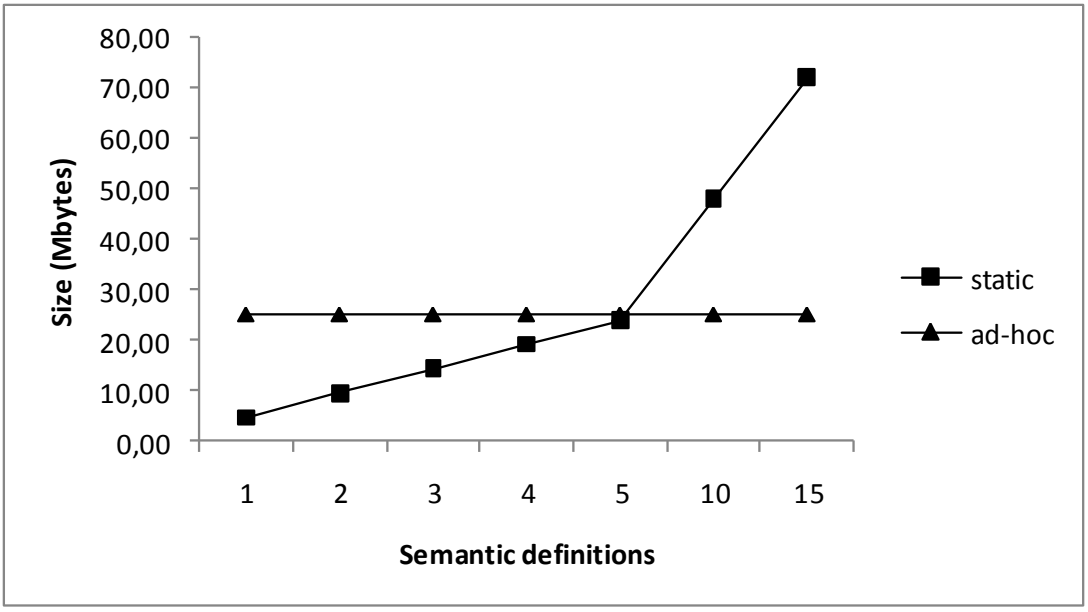


Figure 3-25: Comparing data cube sizes: ad-hoc vs. static approach.

Our fourth experiment compares the sizes of data cubes built using our ad-hoc and the static approach. We recall that our purpose is to build a data cube that will be generic enough to serve a number of applications. We assume that these applications consider some different definitions of the notion of trajectory. Both cubes consist of 210.000 base cells containing the full dataset (6.6 million records). Obviously, the size of ad-hoc data cube remains the same even if there is a large number of different definitions. Even the number of measures that are computed from this data cube does not play any role as the answers are not precomputed. Its size is in proportion to the number of location points as the ad-hoc data cube stores the distances between these points. On the other hand, the size of the *static* cubes is in proportion to the number of measures computed in this data cube and the number of base cells (for

each base cell different values have to be computed). As we see in Figure 3-25, our approach will perform well in term of data cube size if we have five or more definitions.

3.5. Related Work

In the sequel, the related work in the fields of Spatial and Spatiotemporal Warehousing are examined. These are considered as the ancestors of Trajectory Data Warehousing and we focus on dimensional modeling and measures.

3.5.1. Spatial Warehousing

The pioneering work by Han et al. [HSK98] introduces the concept of spatial data warehousing (SDW). The authors extend the idea of cube dimensions so as to include spatial and non-spatial ones, and of cube measures so as to represent space regions and/or calculate numerical data. After this work, several models have been proposed in the literature aiming at extending the classical data warehouse models with spatial concepts and the OLAP tools with spatial operators (SOLAP). However, despite the complexity of spatial data, current SDWs typically contain objects with simple geometric extents. Moreover, while a SDW model is assumed to consist of a set of representation concepts and an algebra of SOLAP operators for data navigation, aggregation and visualization, approaches proposed in literature often privilege either the concepts or the algebra; approaches that address both are rare.

Further, research in SDW modeling can be classified as addressing application requirements at either the logical data level or the conceptual data level. Mainstream solutions rely on the (logical level) relational data model [BMH01], [SHK00]. Relatively few developments focus on SDW conceptual aspects [JKP+04], [MZ04b], [BTM05], [TPG+01]. The analysis presented in [Riz03] asserts the moderate interest of the research community in conceptual multidimensional modeling. However, a significant percentage of data warehouses fail to meet their business objectives [Riz03]. A major reason for failure is poor or inappropriate design, mainly due to a lack of established DW design methods [RG00] and DW conceptual data models [RG00]. Similarly, the authors of [MZ06] state that the proposed models either provide a graphical representation based on the E-R model or UML notations with few formal definitions, or provide formal definitions without any user-oriented graphical support.

Focusing on spatial modeling, existing approaches do not rely on standard data models for the representation of the spatial aspects. The spatiality of facts is commonly represented through a geometric element, instead of an OGC (Open Geospatial Consortium) spatial feature, i.e., an object that has a semantic value in addition to its spatial characterization [Ope01].

Extending classical DW models to deal with spatial data requires allowing both dimensions and measures to hold spatial and topological characteristics. Indeed dimensions and measures should be extended with spatiality in order to enrich the query formulation and the visualization of the results. However adding spatiality to both dimensions and measures is not enough. SDWs have further specific requirements that have been studied in the state of the art, such as different kinds of spatial dimensions and measures, multiple hierarchies in dimensions, partial containment relationships between dimensions levels, non-normalized hierarchies, many to many relationships between measures and dimensions and the modeling of measures as complex entities [BTM05], [BMH01], [JKP+04].

3.5.1.1. Spatial dimensions

When adding spatiality to dimensions, most of the proposals follow the approaches by Stefanovic et al. [SHK00] and Bédard et al. [BMH01] that distinguish three types of dimension hierarchies based on the spatial references of the hierarchy members: non-geometric, geometric-to-non-geometric and fully geometric spatial dimensions. The non-geometric spatial dimension uses nominal spatial reference (e.g. name of cities and countries) and is treated as any other descriptive dimension [RBM01], [RBP+05]. The two other types denote dimensions where the members of lower or all levels have an associated geometry. In the fully geometric spatial dimension, all members of all the levels are spatially referenced while in the geometric-to-non-geometric spatial dimension, members are spatially referenced up to a certain dimension level and then become non-geometric.

More loosely, Malinowski et al. [MZ04b] extend this classification and consider that a dimension can be spatial even in the absence of several related spatial levels. In their proposal, a spatial level is defined as a level for which the application needs to keep its spatial characteristics, meaning its geometry as this is represented by standard spatial data types (e.g. points, regions). This allows them to link the spatial levels of a dimension through topological relationships that exist between the spatial components of their members (contains, equals, overlaps, etc). Based on this, they define a spatial hierarchy as a hierarchy that includes at least one spatial level. In this connection, a spatial dimension is a dimension that includes at least one spatial hierarchy. As such, a spatial dimension is a dimension that contains at least one spatial level; otherwise it is a thematic dimension. An advantage of this modeling perspective is that different spatial data types are associated with the levels of a hierarchy. For example, assuming the hierarchy *user* < *city* < *county* point type is associated to *user*, region to *city*, and set of regions to *county*.

Dimensions and their organization into hierarchies are kept very simple in traditional and operational data warehouses. Levels of traditional non-spatial dimensions are usually organized into containment hierarchies such as *district* < *city* < *county* < *country*. However when dealing with spatial data, two spatial values may not only be either disjointed or one contained into the other, rather they may overlap. For instance, if we add the dimension level *cell* before the *district* level, a cell might overlap two districts. To better address application requirements, a larger spectrum of possible hierarchies is being explored. Jensen et al. [JKP+04] propose a conceptual model that supports dimensions with full or partial containment relationships (see Figure 3-26). The dimension hierarchies can contain levels that may be linked by full or partial containment relationships. For the members of a level linked by a partial containment relationship to members of another level, the degree of containment must be specified (e.g. 80% of this cell is contained in this district).

Support for multiple hierarchies in a single dimension is also an important requirement proposed by the models of Jensen et al. [JKP+04] and Malinowski et al. [MZ06]. It means that multiple aggregation paths are possible in a dimension (e.g. cells can be aggregated in districts or directly in counties). According to these models, multiple aggregation paths enable better handling of the imprecision in queries caused by partial containment relationships. Putting this idea into our example, they argue that the result of the aggregation of cells into county may give better results than aggregating cells into

district, then into city and then into county. The models of Jensen et al. [JKP+04] and Malinowski et al. [MZ06] support non-normalized hierarchy i.e., hierarchies whose members may have more than one corresponding member at the higher level or no corresponding member (e.g. a cell may be related to two districts whereas a district may be related to no cells). Finally, in the model of Malinowski et al. [MZ06], simple hierarchies can be characterized as: symmetrical (i.e. all levels of the hierarchy are mandatory), asymmetrical, generalized (i.e. including a generalization/specialization relationship between dimension members), non-strict (same as non-normalized) and non-covering (i.e. some levels of the hierarchy can be skipped when aggregating).

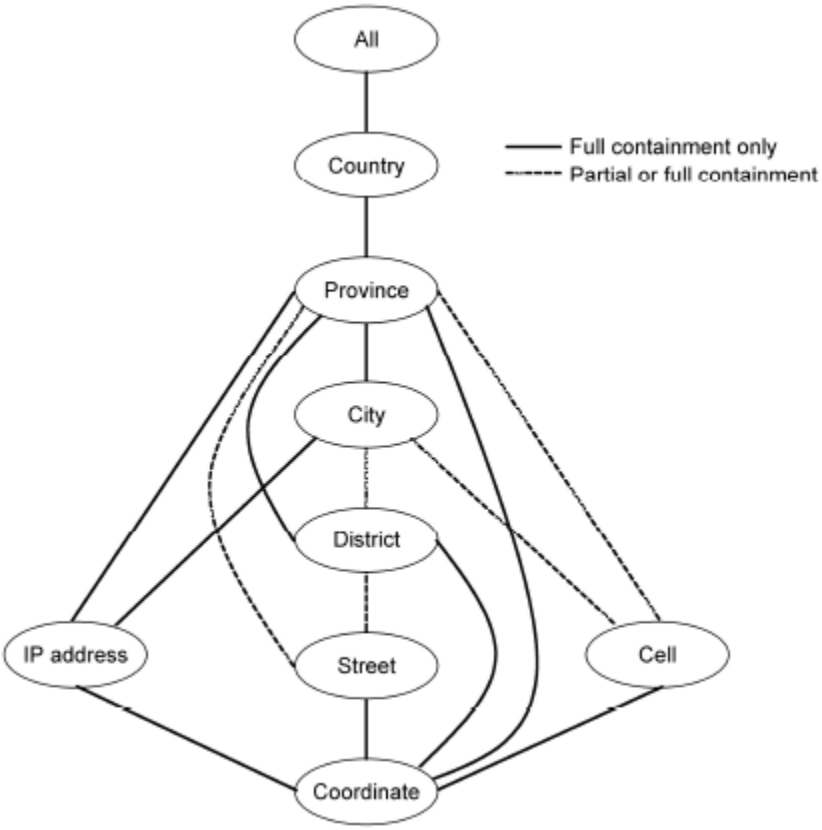


Figure 3-26: Hierarchy with full and partial containment relationship (from [JKP+04]).

3.5.1.2. Spatial measures

Similarly to spatial dimensions, when adding spatiality to measures, most of the proposals distinguish two types of spatial measures [HSK98], [RBM01], [RBP+05]: spatial measures represented by a geometry and associated with a geometric operator to aggregate it along the dimensions, a numerical value obtained using a topological or a metric operator.

When represented by a geometry, spatial measures consist of either a set of coordinates as in [BTM05], [MZ04b], [PT01], [RBM01], [RBP+05] or a set of pointers to geometric objects as in [SHK00]. Finally, Bimonte et al. [BTM05] and Malinowski et al. [MZ04b] advocate the definition of measures as complex entities. In [BTM05], a measure is an object containing several attributes (spatial or not) and several aggregation functions (eventually ad-hoc functions). In a similar way, Malinowski et al.

[MZ04b] define measures as attributes of an n-ary fact relationship between dimensions. This fact relationship can be spatial, if it links at least two spatial dimensions, and be associated with a spatial constraint such as, for instance, spatial containment.

An important issue related to spatial measures concerns the level of detail they are described with. Indeed spatial data are often available and described according to various levels of detail: for instance, the same spatial object can be defined as an area according to a precise level of detail and as a point according to a less detailed one. This is of particular importance with trajectories where the position of the objects is subject to imprecision. Damiani et al. [DS06] propose a model allowing to define spatial measures at different spatial granularities. This model, called MuSD, allows to represent spatial measures and dimensions in terms of OGC features. A spatial measure can represent the location of a fact at multiple levels of spatial granularity. Such multi-granular spatial measures can either be stored or they can be dynamically computed by applying a set of coarsening operators. An algebra of SOLAP operators including special operators that allow the scaling up of spatial measures to different granularities is proposed in [DS06].

Another requirement highlighted by Jensen et al. [JKP+04] and Bimonte et al. [BTM05] concerns relationships between measures and dimensions. Indeed while most of the models only propose to define one-to-one relationships between measures and dimensions, they advocate defining many-to-many relationships, which would allow associating the same measure with several members of a dimension.

3.5.2. Spatiotemporal Data Warehousing

Research on extracting semantically rich information from raw space-time dependent data has focused on spatial and spatiotemporal data warehouses. As we would like to treat trajectory warehouses as a branch of spatiotemporal warehousing [GKM+09], the two subsequent sections present existing approaches in the area categorizing the research efforts into, on the one hand, conceptual and logical modeling methodologies, and, on the other hand, implementation issues regarding aggregation techniques as the quintessence of the data warehousing concept.

3.5.2.1. Aggregation functions and their implementation

A related research issue that has recently gained increasing interest and is relevant for the development of comprehensive SDW data models concerns the specification and efficient implementation of the operators for spatial and spatiotemporal aggregation.

Spatial aggregation operations summarize the geometric properties of objects and as such constitute the distinguishing aspect of SDW. Nevertheless, despite the relevance of the subject, a standard set of operators (like for example the SQL operators SUM, AVG, MIN) has not been defined yet. In fact, when defining spatial, temporal and spatiotemporal aggregates some additional problems have to be faced, which do not show up for traditional data.

In particular, while for traditional databases only explicit attributes are of concern, the modeling of the spatial and temporal extent of an object makes use of interpreted attributes and the definition of aggregations is based on granularities.

A first comprehensive classification and formalization of spatiotemporal aggregate functions is presented by Lopez et al. [LT05]. The operation of aggregation is defined as a function that is applied to a collection of tuples and returns a single value. To generate the collection of tuples to which the operation is applied, the authors distinguish three kinds of methods: group composition, partition composition and sliding window composition.

Recall that a (temporal/spatial) granularity creates a *discrete* image, in terms of *granules*, of the (temporal/spatial) domain. Given a spatial granularity G^S and a temporal granularity G^T , a *spatio-temporal group composition* forms groups of tuples sharing the same spatial and temporal value at granularity G^S *x* G^T . An aggregate function can then be applied to each group. On the other hand, *spatiotemporal partition composition* is used when a finer level of aggregation is required and involves at least two granularities. The first one, which is the coarser, defines collections of tuples (the partitions). To each partition, a *sliding window composition* is performed. Instead of generating a single aggregate value for each partition, an aggregate value for every tuple in the collection at the finer granularity is computed. In order to slide through all tuples in the collection, a spatiotemporal sliding window is used.

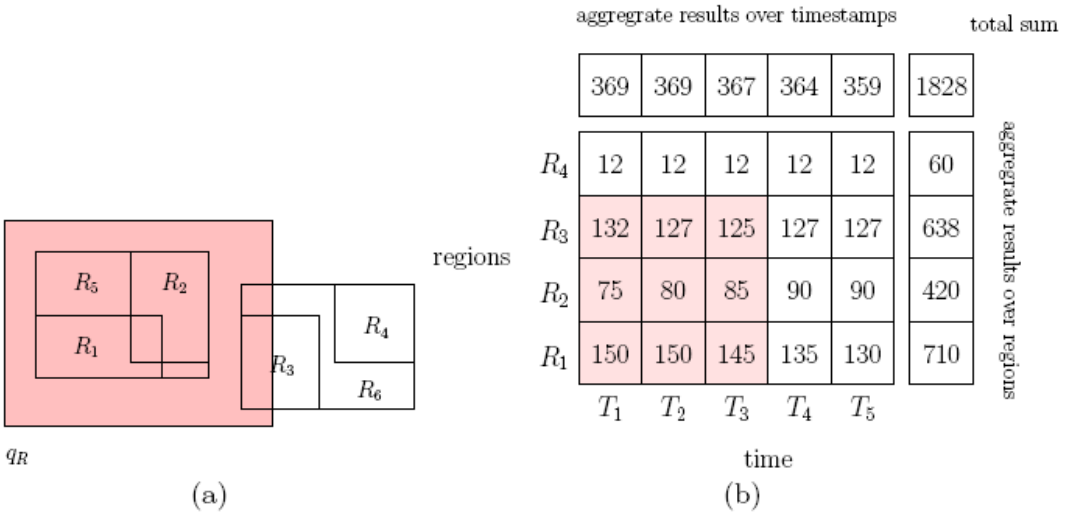


Figure 3-27: (a) Regions of interest, (b) A data cube example.

In addition to the conceptual aspects of spatio-temporal aggregation, another major issue regards the development of methods for the efficient computation of this kind of operations to manage high volumes of spatiotemporal data. In particular, techniques are developed based on the combined use of specialized indexes, materialization of aggregate measures and computational geometry algorithms, especially to support the aggregation of dynamically computed sets of spatial objects [PTK+02], [TP05], [RZY+03], [ZT05]. Papadias et al. [PTK+02], [TP05] propose an approach based on two types of indexes: a *host index*, which manages the region extents and associates to these regions an aggregate information over all the timestamps in the base relation, and some *measure indexes* (one for each entry of the host index), which are aggregate temporal structures storing the values of measures during the history. For a set of static regions, the authors define the *aggregate R-B-tree* (aRB-tree), which adopts

an R-tree with summarized information as host index, and a B-tree containing time-varying aggregate data, as measure index.

To illustrate this concept, consider the regions R_1, R_2, R_3 and R_4 in Figure 3-27(a) and suppose that the number of phone calls initiated in [T_1; T_5] inside such regions is recorded as measure in the fact table depicted in Figure 3-27(b).

Then Figure 3-28 shows the corresponding aRB-tree. This structure is well suited for the efficient processing of *window aggregate queries*, i.e., for the computation of the aggregated measure of the regions which intersect a given window. In fact, for nodes that are totally enclosed within the query window, the summarized measure is already available thus avoiding descending these nodes. As a consequence the aggregate processing is made faster.

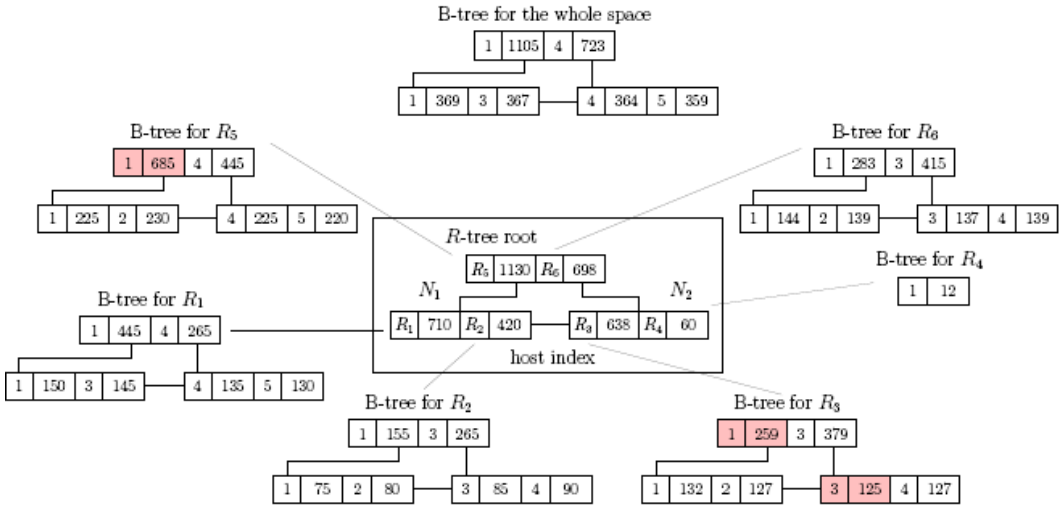


Figure 3-28: The aRB-tree.

For instance, let us compute the number of phone calls inside the shaded area in Figure 3-27(a) during the time interval [T_1; T_3]. Since R_5 is completely included in the window query there is no need to further explore R_1 and R_2 once one accesses the B-tree for R_5. The first entry of the root of this B-tree contains the measure for the interval [T_1; T_3], which is the value we are interested in. Instead, in order to obtain the sum of phone calls in the interval [T_1; T_3] for R_3 one has to visit both an entry of the root of the B-tree for R_3 and also one leaf (the colored nodes in Figure 3-28). Tao et al. [TKC+04] showed that the aRB-tree can suffer from the *distinct counting problem*, i.e., if an object remains in the query region for several timestamps during the query interval, it will be counted multiple times in the result. To cope with this problem, [TKC+04] proposed an approach which combines spatiotemporal indexes with sketches, a traditional approximate counting technique based on probabilistic counting [FM85]. The index structure is similar to the aRB-tree: an R-tree indexes the regions of interest, whereas the B-trees record the historical sketches of the corresponding region. However, this index differs from aRB-trees in the querying algorithms since one can exploit the pruning power of the sketches to define some heuristics allowing reducing query time.

Of particular interest is the analysis regarding the aggregate functions by Gray et al. [GCB+97]. More specifically, authors classify aggregation functions into three categories:

- *distributive*, whose values can be computed from the next lower dimension values,
- *algebraic*, whose values can be computed from a set of aggregates of the next lower level data, and
- *holistic*, which need the base data to compute the results in all levels of dimensions.

Finally, the work by Shekhar et al. [SLC+01] is worth to be mentioned, where the authors proposes a traffic data warehouse model for the Twin-Cities metropolitan area. Although building a warehouse for traffic management, is easier than building a warehouse for trajectories (recall here that the main difficulty is that trajectories may extend to more than one cells), several interesting issues are analyzed in this work. Moreover, for each category of [GCB+97], the authors provide representative aggregation operations inspired from the GIS domain (Table 3-1), which also seem useful in our case.

Table 3-1: Aggregate Operators (from [SLC+01])

	Aggregation Function		
Data Type	**Distributive Function**	**Algebraic Function**	**Holistic Function**
Set of numbers	Count, Min, Max, Sum	Average, MaxN, MinN, Standard, Deviation	Median, Rank, MostFrequent
Set of points, lines, polygons	Minimal Orthogonal Bounding Box, Geometric Union, Geometric Intersection	Centroid, Center of mass, Center of gravity	Equi-partition, Nearest neighbor index

3.5.2.2. Trajectory Data Warehousing

The motivation here is to transform raw trajectories to valuable information that can be utilized for decision making purposes in ubiquitous applications, such as mobile marketing, location-based services and traffic control management. Trajectory warehousing [PRD+08] is in its infancy but we can distinguish three major research directions on this field: modeling, aggregation and indexing.

From a modeling perspective, the definition of hierarchies in the spatial dimension introduces issues that should be addressed. The spatial dimension may include not explicitly defined hierarchies [JKP+04] allowing multiple aggregation paths that they should be taken into consideration during OLAP operations. Tao and Papadias [TP05] propose the integration of spatial and temporal dimensions and present appropriate data structures that integrate spatiotemporal indexing with pre-aggregation. Choi et al. [CKL06] try to overcome the limitations of multi-tree structures by introducing a new index structure that combines the benefits of Quadtrees and Grid files. However, the above frameworks focus on calculating simple measures (e.g. counts of customers).

Recently, an attempt to model and maintain a TDW is presented in [OOR+07] where a simple data cube consisting of spatial / temporal dimensions and numeric measures concerning trajectories, is defined. Authors focus on the loading and computation of the measure presence, which they define it as the number of distinct trajectories lying in a cell. An extension of [OOR+07] can be found in [LOR+09] which discusses storing and aggregation issues for frequent spatio-temporal patterns mined

from trajectories of moving objects occurring in a specific spatial zone and during a given temporal interval.

To the best of our knowledge, materialization techniques have not been studied in the context of TDW. However, there is a flurry of papers that discuss the idea of using views to accelerate computations in traditional data warehouses. A survey of such techniques can be found in [Kot02].

3.6. Synopsis

In this chapter, we studied the application of data warehousing techniques on trajectory data. Two TDW frameworks were thoroughly presented; one that allows a unique and static suitable interpretation of the notion of trajectory and a one for ad-hoc TDW which allows multiple semantic definitions regarding the notion of trajectory.
As for the former approach, we presented alternative ETL process to feed the TDW with aggregate trajectory data: a (index-based) *cell-oriented* and a (non-index-based) *trajectory-oriented*. Additionally, we provided an approximation solution for the solution of the so called *distinct count problem* which appears during OLAP operations.

As for the ad-hoc approach, we extended the OLAP data model so as to include a fact table that will be able to answer queries considering different semantic definitions of trajectories and support the choice of semantics for aggregation queries over trajectory data. Moreover, we enhanced OLAP techniques by presenting an efficient algorithm that takes advantage of our OLAP model regarding answering aggregation queries. Finally, we discussed materialization issues for pre-calculate of results for known semantic definitions of trajectories

4. Trajectory-inspired Data Mining

In the previous chapter, we presented two models for DW and OLAP operations on trajectory data. As it happens in traditional data management world, after building a DW the next step is to apply data mining algorithms either on the DW or directly on the data base. In this chapter we focus on trajectory-inspired mining issues proposing algorithms that need the full details of trajectory data and thus have to be applied on the MOD but also algorithms that need only the aggregated data that are maintained in the TDW. The outline of the chapter is as follows: Section 4.1 introduces the issues being related to pattern mining from mobility data while, Section 4.2 motivates our research. Section 4.3 proposes a framework for discovering interaction patterns that can be utilized for representation, synthesis and classification purposes and Section 4.4 discusses two approaches for mining traffic patterns. Finally, Section 4.5 examines the related work and Section 4.6 closes the chapter providing the conclusions.

4.1. Introduction

We are part of the information age and consequently we are interested in collecting data and analyzing, extracting information and knowledge from them. Companies, organizations, scientists, even simple individuals are interested not just in collecting data but also in explaining the meaning and importance of them and they all want to use this knowledge in order to make better decisions, to solve problems and in general to gain some insights about their domain and relevant phenomena.

Data mining is part of the KDD process and embraces the application of algorithms in order to discover useful patterns from raw data. There is more than twenty years research on this field and it is considered to combine techniques from three core disciplines. In classical statistics in 1960's the term data mining was first introduced. Classical statistics embrace concepts such as regression analysis, standard distribution, standard deviation, standard variance, discrimination analysis, cluster analysis, and confidence intervals. All these are techniques which are used to study data and data relationships and for this reason this field is considered as the most important until today. The second field is artificial intelligence (AI) which is built upon heuristics, contrasting to statistics, attempts to apply a closer to the human thought processing to statistical problems. The third discipline is machine learning which is considered as the union of the two above mentioned fields (statistics, AI). Machine learning introduced the idea of making computer programs intelligent enough to learn about the data they study.

Data mining can be further categorized to data clustering, association rule mining and classification techniques. The former [KR90], [JMF99] is the unsupervised process of grouping together sets of

objects into classes, also called clusters, with respect to a similarity measure. Thus, it is the behavior of groups rather than that of individual records that is detected. Segmenting a database in several partitions provides a more general view and a deeper understanding of the data.

Association rule mining aims at discovering interesting correlations among database attributes [AIS93]. Association rules are implications of the form $A \Rightarrow B$ [s, c], $A \subset J$, $B \subset J$ where A, B and J are sets of items (i.e. attributes), characterized by two measures: *support* (s) and *confidence* (c). The support of a rule $A \Rightarrow B$ expresses the probability that a database event contains both A and B, whereas the confidence of the rule expresses the conditional probability that a database event containing A also contains B.

Classification is one of the most common supervised learning techniques. The objective of classification is to first analyze a (labeled) training set and, through this procedure, build a model for labeling new data entries [HK00]. In particular, at the first step a classification model is built using a *training data set* consisting of database records that are known to belong in a certain class and a proper supervised learning method, e.g. decision trees or neural networks. In case of decision trees, for example, the model consists of a tree of "if" statements leading to a label denoting the class the record it belongs in. At the second step, the built model is used for the classification of records not included in the training set. Many methods have been developed for classification, including decision tree induction, neural networks and Bayesian networks [FPS+96].

The application of the aforementioned techniques on conventional data has been extensively studied during the last decades. Nowadays, the high volume of collected mobility data arises the challenge of applying appropriate data mining techniques on such data. Traditional techniques have to be extended so as to take into consideration the complex nature of spatiotemporal data and handle them in an efficient way. In recent years, the quick development and large diffusion of spatial localization technology led to the rise of a new field called *mobility mining* [GP07], aimed to collect and analyze mobility data for modeling and understanding the behaviors of large populations of moving objects, such as vehicles in traffic management applications or individuals in location-based services for mobile phones.

4.2. Motivation Issues

The motivation behind trajectory-inspired data mining is the development of methods that can extract useful patterns out of mobility data. Generally speaking, mobility data mining is still in its infancy, and even the most basic questions in this field are still largely unanswered: what kinds of patterns can be extracted from trajectories? Which methods and algorithms should be applied to extract them?

How can such patterns be effectively used to improve the comprehension of the application domain and to deliver better services? The following basic examples give a glimpse of the wide variety of patterns and possible applications it is expected to manage:

- *Clustering*, the discovery of groups of "similar" trajectories, together with a summary of each group. Knowing which are the main routes (represented by clusters) followed by people

during the day can represent a precious information for improving several different services to citizens. E.g., trajectory clusters may highlight the presence of important routes not adequately covered by the public transportation service.

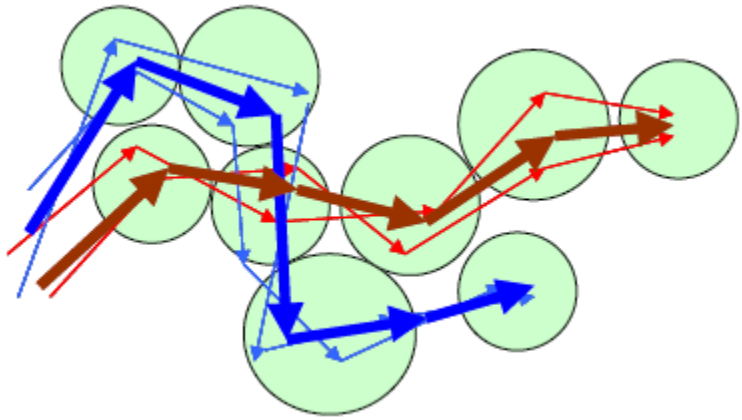


Figure 4-1: An example of trajectory clustering

- *Frequent patterns*, the discovery of frequently followed sub paths. Such information can be useful in urban planning, e.g., by spotlighting frequently followed inefficient vehicle paths, which can be the result of a mistake in the road planning.

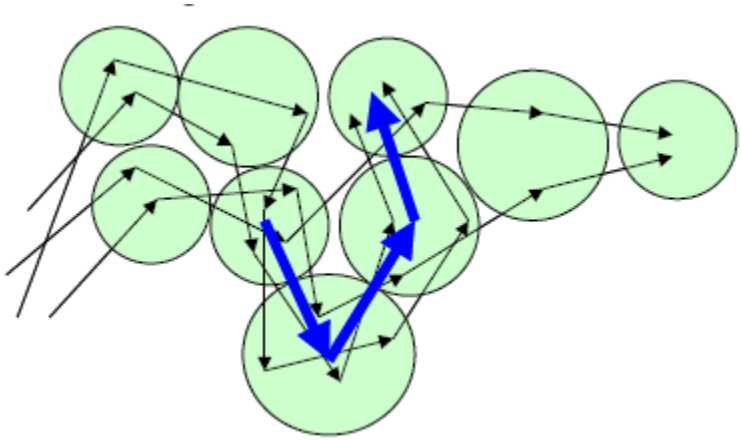


Figure 4-2: An example of frequent trajectory patterns

- *Classification*, the discovery of behavior rules, aimed at explaining the behavior of current users and predicting that of future ones. Urban traffic simulations are a straightforward example application for this kind of knowledge, since a classification model can represent a sophisticated alternative to the simple ad hoc behavior rules, provided by domain experts, on which actual simulators are based.

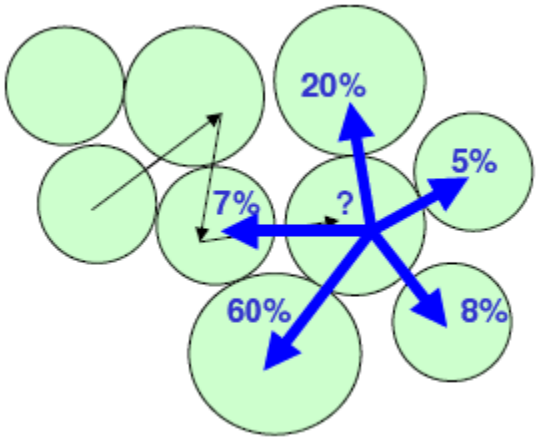


Figure 4-3: An example of trajectory classification

Spatiotemporal data introduce new possibilities and, correspondently, novel issues in performing these tasks. Clustering moving object trajectories, for example, requires finding out both a proper spatial granularity level (strongly dependent on the application) and a significant temporal subdomain (e.g., rush hours might be informative for defining a clustering structure over traffic data, while other time periods might simply add noise to the clustering process).

Whichever kind of pattern it is extracted, a special attention has to be paid to the risks of privacy violation. So far, existing work on privacy protection in data mining is essentially limited to two directions: in the first one, the aim is to allow to mine any pattern/model with the only concern of not explicitly distributing raw sensitive data, usually obtained by applying data perturbation or multi-party computation techniques; in the second one, the raw data are made public after a *sanitization* treatment aimed at masking a predefined set of sensitive patterns. The first approach is clearly insufficient in our context, since without any control procedure over the pattern generation there will always be the risk of extracting patterns that violate in some extent the privacy of an individual or a community. The second approach addresses the problem in a straightforward way.

However, it can be applied only to simplified contexts, and is clearly unsatisfactory for our purposes: in general, it is not feasible (or possible at all) to define a priori which patterns are sensitive and, moreover, the approach seems to be not extendible to other, not pattern-based, mining tasks, such as clustering and classification. As an example of privacy issues in the latter category of tasks, if an extremely homogeneous cluster is obtained, knowing its "summary" (e.g., a representative trajectory) can be sufficient to reconstruct information on the individuals it contains (e.g., their trajectories would coincide almost perfectly with the representative trajectory of the cluster). In the same way, an overfitted classifier could contain rules generated on the base of only one or very few individuals, thus disclosing some (potentially sensitive) information on their characteristics.

In order to properly tackle the privacy problem, the generated patterns should be precisely characterized as privacy-preserving or not, in the sense that a formal set of constraints on spatiotemporal data and patterns should be available to express privacy requirements, along with data mining algorithms that demonstrably yield only privacy-preserving patterns – e.g., patterns that do not

disclose any sensitive information of the individual trajectories they come from. Other flavors of privacy-preserving methods, such as data perturbation and multi-party secure computation should also be taken into account, whenever appropriate to deal with different privacy requirements.

4.2.1. Our contributions

We have two developed two trajectory-inspired mining techniques that are capable of extracting patters from mobility data. We summarize their functionality as presented in [NMO09], [NMM08], [MT09b]:

- The increasing pervasiveness of location-acquisition technologies (GPS, GSM networks, etc.) is leading to the collection of large spatiotemporal datasets and to the opportunity of discovering usable knowledge about movement behavior, which fosters novel applications and services. In this work, we move towards this direction and develop an extension of the supervised learning paradigm that analyzes the trajectories of moving objects. We introduce trajectory *interaction patterns* as concise descriptors of regions, in terms of both space (i.e., the regions of space visited during movements) and relations of similarity between trajectories, to extract semantics from them. We provide a general statement of the novel mining problem and then study different instantiations of different complexity.

- The flow of data coming from modern sensing devices enables the development of novel research techniques related to data management and knowledge extraction. In this work, we undertake the problem of analyzing traffic on a road network so as to help the city authorities to optimize traffic flow. A graph based modeling of the network traffic is presented which provides insights on the flow of movements within the network. We exploit this graph in order to analyze the traffic flow in the network and to discover traffic relationships like propagation, split and merge of traffic among the road segments. First experimental results illustrate the applicability and usefulness of our approach.

In traditional data management world, the data mining tasks can be applied either on the database or the data warehouse. The same applies here for our two trajectory-inspired mining techniques. The former needs the details of the raw trajectory data and hence it is applied on the MOD, whereas the latter utilizes aggregate data and so it can be applied on the TDW.

4.3. Mining Interaction Patterns for Spatiotemporal Representation, Synthesis and Classification

In the following paragraphs, we describe our work presented in [NMO09] which introduces two basic ideas: (i) an adequate understanding of what is happening to a single object requires to look not only at its trajectory, but also at the context where it moves; (ii) such context is defined not only by the nature of the geographical space, but also by the presence of other objects and their interaction with the moving object under analysis.

Therefore, we provide a framework for extracting useful features that can be used to represent the movement of an object providing useful insights on the characteristics of this movement. Furthermore,

we can explore the set of features so as to apply classification models that will help us discover possible relationships between movement characteristics and behaviors.

4.3.1. Problem Definition

The problem tackled in this work is an instantiation of the classical classification problem specialized for trajectory data. In this section we provide basic definitions for the trajectory data model adopted in this work and for the specific classification problem studied. The classification problem can be generally defined as:

Definition 4-1 (Trajectory Classification): Given a set of labels L and *a* dataset D of n labeled trajectories, $D = <(T_1; l_1),\ldots,(T_n; l_n)>$, where $\forall\ 1 \leq i \leq n : l_i \in L$ *and* $T_i \in T$, the trajectory classification problem extracts a function $C : T \rightarrow L$ that best approximates the label association induced by D. ■

Such generic definition can be instantiated into several concrete approaches, the most common of which is the feature-based approach, consisting in extracting from the input objects a set of values that describe some of their properties and can be easily represented as a fixed-size vector. That allows applying any of the several standard classification techniques developed for relational data, and will be adopted in this work. Other alternatives can require classification schema tailored around the specific data type, for instance model-based approaches, or general schema such as k-Nearest Neighbors, which can be instantiated to the specific data type by defining a similarity function between objects (e.g., similarity measures between trajectories).

In this work, therefore, we will contribute to the core step in feature-based classification, i.e., features extraction, which is formally defined below.

Definition 4-2 (Features Extraction): Given a set of labels L and a dataset D of n labeled trajectories, $D = <(T_1; l_1) ,\ldots, (T_n; l_n)>$, where $\forall\ 1 \leq i \leq n : l_i \in L$ *and* $Ti \in T$, the trajectory feature extraction defines a positive integer n and a function $F : T \rightarrow D_1\ x\ .\ x\ D_n$ that associates each trajectory to a vector of n features of proper type $(D_i, i = 1,\ldots, n)$. ■

4.3.2. Interaction patterns

Objects can be generally represented by their life span plus the subset of their (time-varying) observable properties that are relevant to the analysis context. For instance, a moving object could be represented as a time interval (when the object is monitored) plus its time-varying spatial coordinates, speed and acceleration.

Definition 4-3 (Observable Set): Any object O is represented as a pair, composed of a time interval I and a sequence of functions that associate each time instant to a real value:

$$F_O = (I, (f_O^1, \ldots, f_O^n)), \forall\ 1 \leq i \leq n\ f_O^i : I \rightarrow \mathbb{R}$$

F_O is called the observables set of object O. ■

A subset of the properties that describe an object usually define the space (physical or virtual) where the objects exist, evolve and, if close enough, possibly interact. For instance, the spatial coordinates of moving objects define where objects lie, and closer objects are more likely to interact.

Definition 4-4 (Location Attributes): Given the observable set $F_O = (I, S)$ of object O, we define the location attributes of O as a subset Loc_O of S: $Loc_O \subseteq S$ ■

Similarly, a subset of the object properties describing its relevant activities can be used to characterize the possible interaction between objects. For instance, a relevant property for a moving object can be the speed, and the relations between the speed values of different objects that interact can help to describe such interaction.

Definition 4-5 (Interaction Attributes): Given the observable set $F_O = (I, S)$ of object O, we define the interaction attributes of O as a subset $Intc_O$ of S: $Int_O \subseteq S$ ■

Location and *Interaction Attributes* can overlap, i.e., in general, $Loc_O \cap Int_O \neq \emptyset$

Interaction between objects is a relation that implies a mutual locality, therefore a notion of proximity - or, in other terms, of region of influence of each object - should be defined. Such a notion should be based on the location attributes of objects, since they describe their position in space. Moreover, such relations usually have a limited duration and can change along time; therefore they should be measured within time windows of limited width. For instance, for moving objects a set of interesting spatial regions could be defined, and interaction between objects could be measured separately on each region every hour.

In order to characterize the interaction between close objects, we look for relations between the interaction attributes of the interacting objects. This can be done in several different ways, such as looking for statistical correlations or pre-defined patterns. In this work, we choose to compare for each attribute the values of all objects within each region and time interval, then giving more emphasis to the objects that show significant divergence from the others. For instance, we could compare the speed of all objects within a region and a time interval, and locate those whose speed differs significantly from the others.

Definition 4-6 (Interaction Descriptor): Given a population of objects, such that object O has the same location and interaction attributes *Loc* and *Int* respectively, we define the *interaction descriptor* for an object $O=(I, S)$ in region r and time interval T, and for interaction attribute f , as a function $ID(O, r, T, f)$:

$$ID(O,r,T,f) = \begin{cases} AVG_{t \in T \wedge O' : Loc_{O'}(t) \in r}, & \big(f_O(t) - f_{O'}(t)\big) if\ t \in I \cap I' \\ undef, & otherwise \end{cases}$$

where I' denotes the definition interval of object O'. ■

The interaction descriptors describe how each object is related with the others in each region and each time interval, and the descriptor resulting from each different interaction attribute do that from a different perspective. Obviously, each object will traverse only a subset of the possible regions, and will do that in a precise order. Moreover, in real situations, attributes of each object are not available in a continuous fashion, but only point-wise, at a given sampling rate. The ultimate result is that each object can be characterized by a collection of time series, $ID(O; f)$, one for each interaction attribute f,

describing how, from each *viewpoint*, the interaction between object *O* and all the others changed along time. Or, more precisely, how in such interaction the object behaved differently from the others.

While accidental interactions can be neglected as spurious phenomena, recurrent behaviors can be regarded as potentially meaningful for sketching the profile of each object. For instance, if only one driver in a pool of taxi drivers always drives much faster than people around him/her (which results in a sequence of speed interaction descriptor values always very high), that can be treated as an extreme case, an outlier. But if such behavior is common, it is useful to consider a category of \fast drivers".

Definition 4-7 (Interaction Pattern): Given a population of objects *Objs*, such that each object $O \in Objs$ has the same location and interaction attributes *Loc* and *Int* respectively, we define an *interaction pattern* for interaction attribute $f \in Int$ as any sequence of values $V = (v_1, \ldots, v_n) \in R^n$ such that, given a support threshold $\sigma \in [0, 1]$ and an error tolerance $\varepsilon \in R^+$:

$$\frac{|\{O \in Objs \mid match\ (V, ID(O, f))\}|}{|Objs|} \geq \sigma$$

where $match\ (V, V') \Leftrightarrow \exists\, i_1 < \cdots < i_n\ \forall\, 1 \leq j \leq n\ |V[j] - V'[i_j]| \leq \varepsilon$ ■

An approximation of the above defined interaction patterns can be obtained by discretizing the values of each interaction descriptor, and matching two values if they fall in the same discretization interval.

4.3.3. Computing IPs and trajectory features

4.3.3.1. Selection of interacting trajectory segments

In principle, the interaction of an object with the others is highly dynamical and changes continuously during time. Therefore, it should be re-computed for each time instant *t* contained in the object's trajectory. In order to do that, an accurate solution would require to discover which objects pass close (i.e., within distance *ds*) to the reference object within a time window *[t-dt, t+dt]*, and then compare the trajectory segment of the reference object that lay in the time window against the corresponding segments of the neighboring objects. An example of this process is shown in Figure 4-4(a), where the red line segments identify the parts of trajectories that can contribute to the interaction descriptors for the central trajectory at time *t*.

A slightly less precise computation of interaction descriptors, which is followed in this work, requires to partition the temporal axis into intervals of equal width, and then to re-compute a value of the descriptors within each time interval. The segments of neighboring objects to consider are identified in a similar way as the general solution described above, expected that the spatiotemporal neighborhood adopted is simplified to a set of cylinders having centers on each point of the reference trajectory and height such that it covers the temporal gap between the point and the next one in the trajectory. This is depicted in Figure 4-4(b), where, again, red line segments identify the relevant parts of neighboring objects. Notice that the time interval of analysis is fixed, while the spatial neighborhood moves with the reference object.

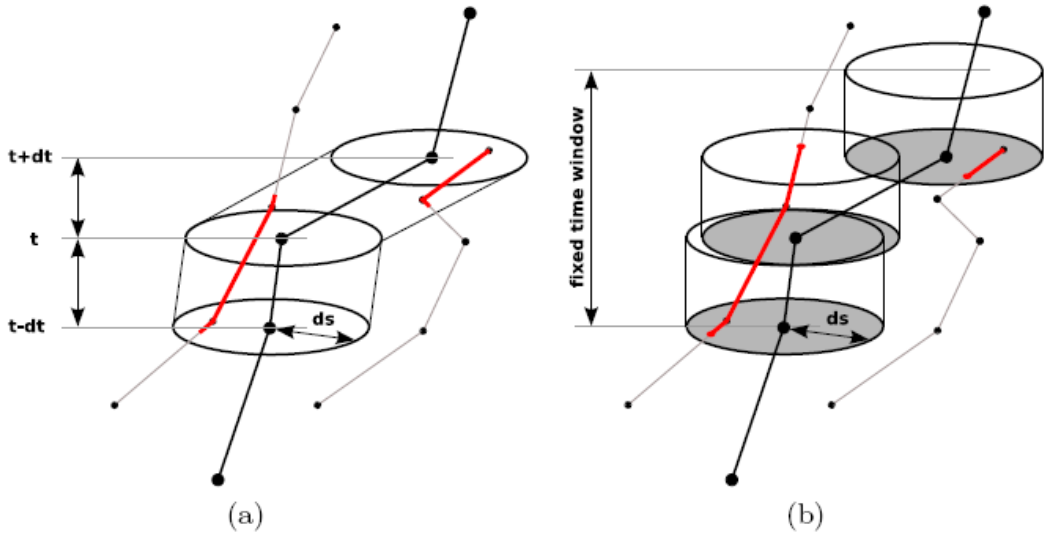


Figure 4-4: Ideal neighborhood of single point (a) and neighborhood for a time window (b)

Finally, a rougher approximate result can be obtained fixing a spatial grid, such that descriptors are computed for each trajectory segment contained in the resulting spatiotemporal cell (a spatial cell in a given time interval), taking into account all trajectory segments of neighboring objects that also fall in the cell. Figure 4-5 shows a graphical example, with similar color notation as above. Notice that this time both the time interval and the spatial neighborhood are fixed.

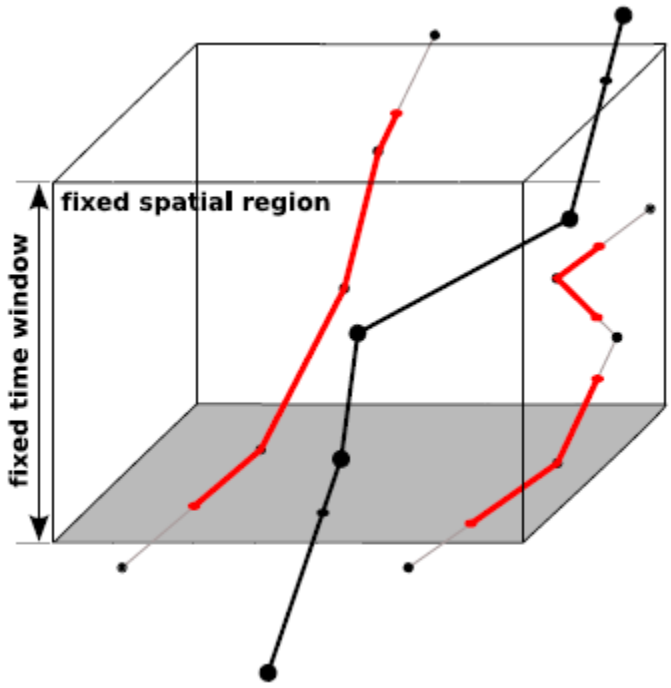


Figure 4-5: Fixed grid neighborhood

4.3.3.2. Computing Interaction Descriptors

In this work we consider two families of descriptors; depending on the computation process they require Figure 4-6, that graphically summarizes both approaches):

- *macro* interaction descriptors: all other objects are aggregated to a single value or complex object, then the reference object under analysis is compared against the result of such aggregation and yields a single value. E.g., the first aggregation might return the global average speed, whereas the comparison computes the difference between the average speed of the reference object and the global one of its neighbors.
- *micro* interaction descriptors: the reference object is compared against each of the neighboring objects, then the set of results obtained are aggregate to a single value. E.g., for each neighboring object the minimum distance from the reference object is computed, then the average of all resulting values is returned.

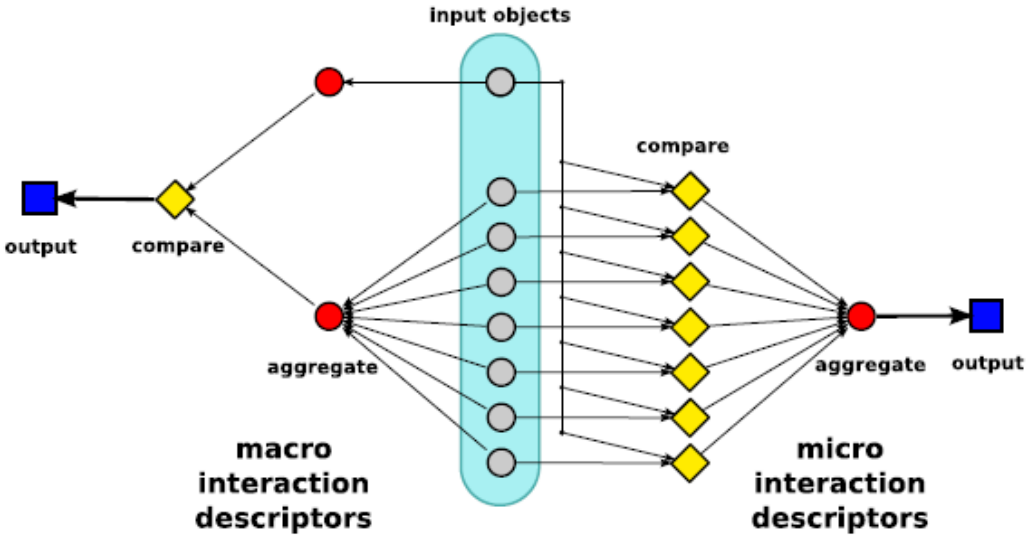


Figure 4-6: Macro and micro interaction descriptors

Let us consider a trajectory *T* and a set of trajectories *ST* that reside in a specific region *R* (which can be computed using either the dynamic neighborhood Figure 4-4(b) or the fixed grid approach Figure 4-5). Our aim is to compute the interaction between *T* and *ST* that can be quantified through various interaction descriptors.

For instance, we consider: the descriptor AVG_PERC_DISTANCE allows us to compare the length of *T* with the average length of *ST*. Similar is the case for the AVG_PERC_DURATION descriptor where the lifespan of *T* is compared with the average lifespan of *T*. In the same fashion, the AVG_PERC_SPEED descriptor is calculated by dividing the average speed of *T* with the average of the speeds of each trajectory *T'* of set *ST*. Likewise, the AVG_PERC_ABS_ACCELER is computed by dividing the average acceleration of *T* with the average of the accelerations of each trajectory *T'* of set *ST*. Finally, VAR_PERC_ACCELER is defined as the proportion of the variance of acceleration of trajectory *T* to the variance of acceleration of set of trajectories *ST*. The aforementioned set of descriptors can be computed using either the *macro* or the *micro* approach.

As it is clear from the description of the computation of the various descriptors, we need to work on trajectories instead of raw points. Collected raw data represent time-stamped geographical locations

from which we can gain valuable information about trajectories. Our main interest is to reconstruct trajectories in order to study the movement of objects. This is achieved by deciding and defining, in each specific case, which set of location points formulates a specific trajectory. In this work, we utilize the method for determining different trajectories that is proposed in [MFN+08a] and is thoroughly presented in Chapter 5.

In order to compute the different descriptors, we propose two algorithms: COMP-DESCRIPTORS-DYNAMICNEIGHBORHOOD and COMP-DESCRIPTORS-FIXEDGRID. The former is used to compute descriptors using the dynamic neighborhood approach. It takes as input a) the list of available points (lop), b) a list of predefined temporal periods (lotp) during which we wish to compute descriptors and finally c) a maximum spatial distance (msd) that is used to define the neighborhood around each point. Initially, the subset of points inside each temporal period is defined (line 4). After this, for each different object, one or more trajectories are reconstructed (line 6) and for this trajectory its neighborhood is computed (line 10).

```
Algorithm Comp-Descriptors-DynamicNeighborhood (ListOfPoints
lop, ListOfTemporalPeriods lotp, MaxSpatialDistance msd)
 1. FOR EACH period p of lotp DO
 2.  //choose the subset of points that are temporally
 3.  //restricted inside period p
 4.  lop' = GetPointsForPeriod(lop, p);
 5.   FOR EACH object o of lop' DO
 6.    lot = ReconstructTrajectory(o, lop');
 7.     FOR EACH trajectory T in lot DO
 8.      //lont is a list of trajectories that are considered
 9.      //as the neighborhood of T
10.      lont = ComputeNeighborhood(T, msd);
11.      //compare T with lont set of trajectories
12.      ComputeDescriptors(T, lont);
13.     END-FOR
14.   END-FOR
15. END-FOR
```

Figure 4-7: The algorithm COMP-DESCRIPTORS-DYNAMICNEIGHBORHOOD

COMP-DESCRIPTORS-FIXEDGRID algorithm is used to compute descriptors on a fixed spatiotemporal grid. It takes as input the list of available points (lop) and a list of predefined spatiotemporal cells (lostc). Initially, the subset of points inside each spatiotemporal cell is defined (line 4). Here there is not the notion of neighborhood and hence the set of trajectories is static and can be reconstructed once (line 5). Then, for each trajectory its descriptors are computed (line 9).

```
Algorithm Comp-Descriptors-FixedGrid
(ListOfPoints lop, ListOfSpatiotemporalCells lostc)
 1. FOR EACH spatiotemporal cell c of lostc DO
 2.  //choose the subset of points that are spatiotemporally
 3.  //restricted inside cell c
 4.  lop' = GetPointsForSTCell(lop, c);
 5.  lot = ReconstructTrajectories(lop');
 6.   FOR EACH trajectory T in lot DO
```

```
 7.    lont = lot.Exclude(T);
 8.     //compare T with lont set of trajectories
 9.     ComputeDescriptors(T,lont);
10.   END-FOR
11. END-FOR
```

Figure 4-8: The algorithm COMP-DESCRIPTORS-FIXEDGRID

4.3.3.3. Extracting Interaction Patterns

The overall feature extraction method proposed follows three main steps, summarized below:

Region extraction: we select some regions of space and time to be used later as reference locations. In this step either the dynamic neighborhood (Figure 4-4(b)) or the fixed grid approach (Figure 4-5) is chosen.

Interaction descriptors: within each selected region *R*, we consider all the segments of trajectories that lay in *R*, then we follow either the *macro* or the *micro* approach so as to compute various descriptors.

Interaction patterns: sequences of regions (e.g., those visited by each trajectory) are analyzed, searching for evolutions of the descriptors that occur frequently. Interaction patterns can be extracted using a conventional sequential patterns mining algorithm.

Interaction patterns succinctly describe some interesting aspects of the dynamics of the overall system, such as the acceleration/deceleration of groups of individuals in some region in some time interval, a change in the mutual distance between individuals, etc. In general, an interaction pattern describes a conjunction or a sequence of such events, thus potentially modeling complex, yet statistically significant, behaviors. Subsequently, each interaction pattern can be used as a descriptive feature, marked true for those trajectories that are involved at least in one of its occurrences –i.e., they pass regions where the interaction pattern occurs – and false in all other cases. In this work, in particular, such features are used as independent variables in the trajectory classification task.

4.3.4. Experimental Study

The experiment was performed over the NGSIM U.S. 101 dataset, a freely available collection of image-based reconstructed trajectories provided by the Federal Highway Administration and the Next Generation Simulation program[1]. The data describe the traffic of vehicles over a 2100-feet-long and 6-lanes-wide (5 mainlines plus an auxiliary one) segment of the U.S. Highway 101 (Hollywood Freeway) located in Los Angeles, California, covering a time interval of 45 minutes. The data contains 6101 distinct trajectories, corresponding to approx. 4 million points, each given as a quadruple (ID, t, x, y), plus some derived information, such as speed and distance from the next vehicle in the lane, also exploited in our experiments. Figure 4-9 shows a spatial projection of a sample of the input dataset. The movement direction of cars is along the Y-axis – in particular, the existence of several lanes along the Y-axis (including an auxiliary one for entering/exiting the highway) is visible. Notice the different scales used for the two axes, since the length of the monitored area is much larger than its width.

[1] http://www.ngsim.fhwa.dot.gov

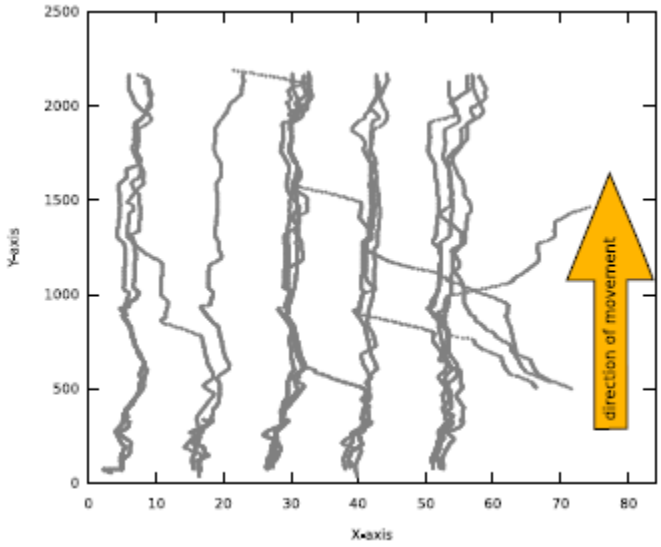


Figure 4-9: Sample of the U.S.101 dataset.

The classification task was focused on the identification of vehicles that, at the edge of the monitored segment of highway, would eventually show some dangerous behavior. Such situation has been formalized through a simple rule that labels a trajectory segment as dangerous driving profile if any of the following conditions holds at some moment:

- the vehicle changes 2 or more lanes; or
- the vehicle moves 20% (or more) of the periods above average speed; or
- the vehicle moves 10% (or more) of the periods over the speed limit.

The labeling process yielded ca. 23% of dangerous driving profiles, and 77% of non dangerous ones. The classification mechanism makes use of the discretized values of the various descriptors. Such values are used as "items" to produce a sequence of itemsets for each object. Then, frequent sub-sequences (our interaction patterns) are extracted. Each pattern is then used as a feature, set to 1 if the object followed it and 0 otherwise. Therefore, the classifiers we build try to understand whether the object is a dangerous driver (or, better, "will be a dangerous driver in the 'tail' part of its trajectory") based on which IP it followed in the first part of its motion.

The purpose of the first experiment is to prove that using our framework we can accurately predict dangerous vs. non dangerous profiles using the interaction descriptors presented in the Subsection 4.3.3.2. To this purpose, the dataset was split into two parts, respectively long the 70% and 30% (we also tried the 80%-20% pairs) of the whole dataset, the former used as a training dataset and the latter used for validation purposes. We set the temporal period inside which we look for dynamic neighborhood at 1.5 minute and the spatial tolerance at 5 meters. As for the fixed grid approach, we consider spatiotemporal grids of 10 m (width) x 64 m (height) x 2 minutes.

In Figure 4-10, we compare three classifiers: one that always classifies drivers as “not dangerous” and therefore has a constant accuracy rate (77%), and the two others explore the IP classification method using the dynamic and fixed grid approach respectively. We run the experiment both using the 70%-30% and 80%-20% of the dataset for training-validation. The difference between the IP-classification-

dynamic and the IP-classification-fixed approach can be easily explained as in the former case the interaction descriptors that are computed are between moving objects that belong in the same neighborhood and likely they interact. On the other hand, the IP-classification-fixed, allow us to compute more neighborhood-relaxed descriptors (but still the results are better from the naïve approach).

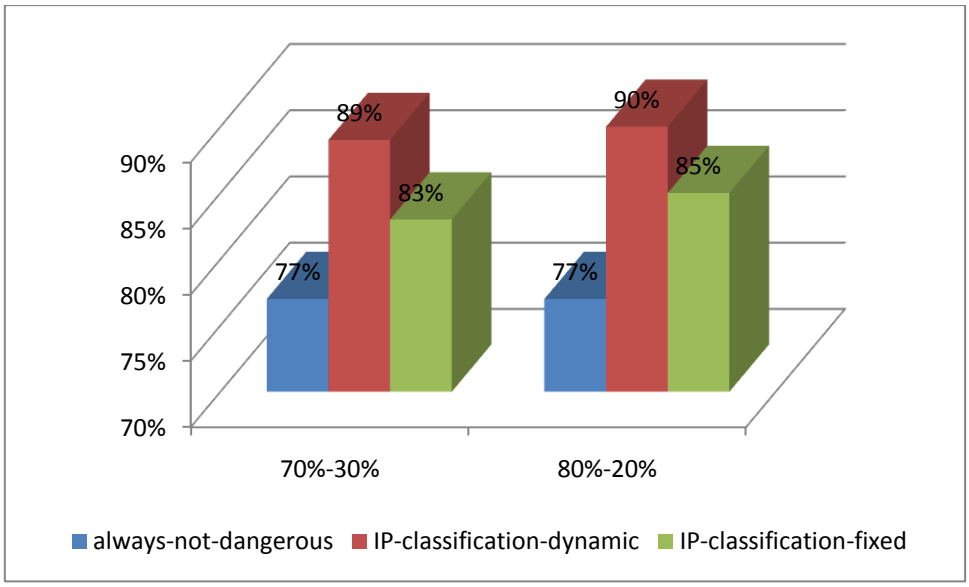


Figure 4-10: Measuring the accuracy of our framework: IP classification vs. always-not-dangerous classifier.

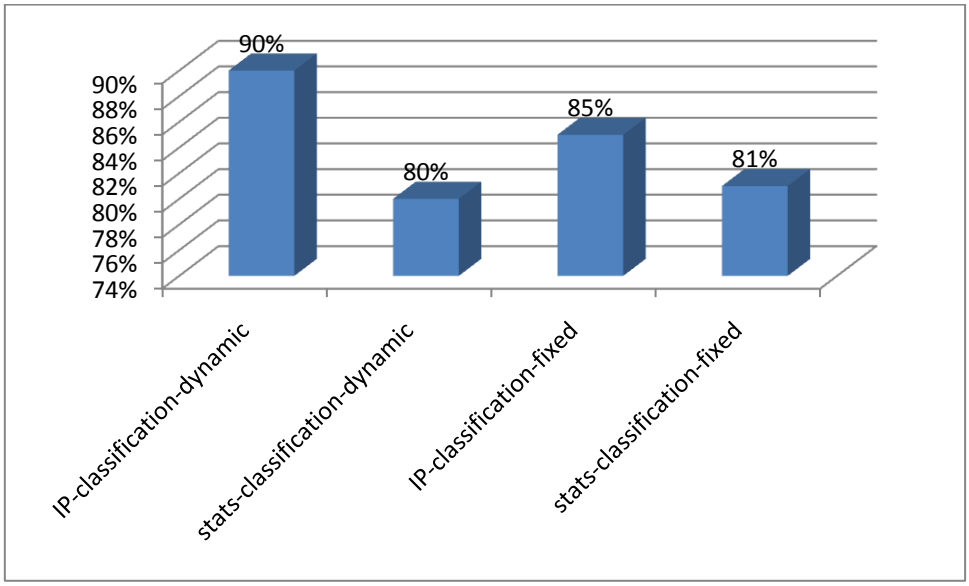


Figure 4-11: Measuring the accuracy of our framework: IP classification vs. simple statistics classifiers.

In our second experiment, we utilize simple statistics that do not take into consideration the interactions among the objects and we compare them our IP-classification approach Figure 4-11. In this case, we fix the training-validation percentages to 80%-20%. The idea here is to prove that measuring the

interactions between objects is important and can give us better insights that computing just some statistics about isolated movements. More specifically, the IP-classification-dynamic and IP-classification-fixed utilize the interaction descriptors presented in the Subsection 4.3.3.2: AVG_PERC_DISTANCE, AVG_PERC_DURATION, AVG_PERC_SPEED, AVG_PERC_ABS_ACCELER, and VAR_PERC_ACCELER. On the other hand, stats-classification-dynamic and stats-classification-fixed utilize simple statistics, i.e. AVG_SPEED_SINGLETRAJ (the average speed of the trajectory), AVG_ACCELERATION_SINGLETRAJ (the average acceleration of the trajectory), VAR_SPEED_SINGLETRAJ (the variance of the speed of the trajectory), VAR_ACCELERATION_SINGLETRAJ (the variance of the acceleration of the trajectory) and COUNT_LINESCHANGED_SINGLETRAJ (the number of lanes that the object changed). As we seen in Figure 4-11, simple statistics cannot help us classify dangerous vs. not dangerous drivers with the same accuracy as using interaction patterns.

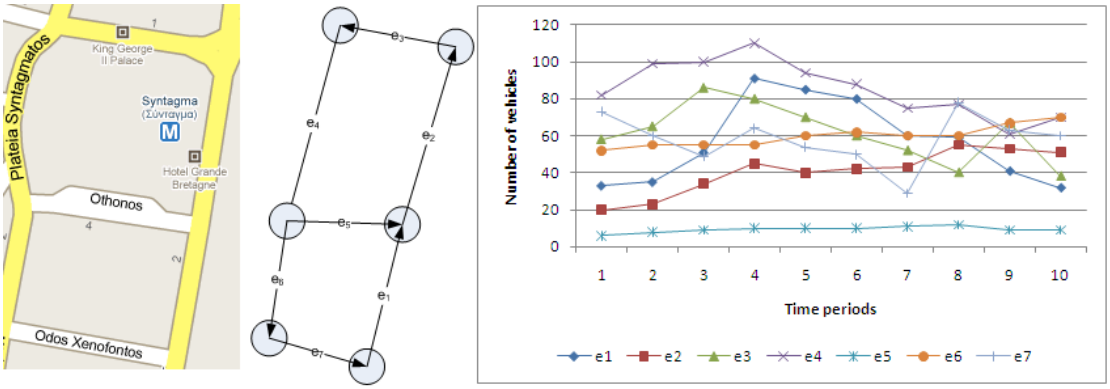
4.4. Mining Traffic Patterns

Trying to understand, manage and predict the traffic phenomenon is both interesting and useful. For instance, city authorities are looking for ways to improve traffic flow and to react effectively in case of some traffic problem, like car accident or road repairing. Moreover, studying the traffic flow, the city authorities could better arrange the construction of new roads, the extension of existing ones, and the placement of traffic lights and so on.

Intelligent Transportation Systems (ITS) have been developed allowing better monitoring and control of traffic in order to optimize traffic flow. ITS collect data utilizing various technologies such as sensors, live cameras and cellular phone data and in turn, they propose reroute of traffic to avoid congestion. Beyond transportation perspective, information technology provide a variety of applications and technologies for gathering, storing, analyzing and providing access to traffic data. Our aim is to present an intelligent traffic management decision support system that incorporate data mining techniques to extract traffic patterns.

In order to better serve the above targets we aim at analyzing traffic data so as to monitor the traffic flow and thus to discover traffic related patterns. These data can be derived either from sensors that are placed along the network and transmit traffic information at adjacent time periods or by GPS data that are map matched on edges and are aggregated at a specific temporal granularity. Traffic related patterns can be expressed through relationships among the road segments of the city network. In other words, we aim to discover how the traffic flows in this network, which road segments contribute to the flow and how this happens.

We model the road network (Figure 4-12a) as a directed graph (Figure 4-12b). We assume that traffic data is collected and we treat these data as time series (Figure 4-12c) that can be further analyzed so as to discover relationships among the different edges/road segments of the network.



(a) (b) (c)

Figure 4-12: (a) a real road network, (b) the corresponding graph network and (c) the time series of the edges of the network.

We define various relationships between the edges of the network graph: traffic propagation from one edge to some other edge, split of traffic from one edge into multiple edges and merge of traffic from multiple edges into a single edge. These relationships allow us to combine the temporal information provided by the traffic time series with the spatial semantics inherited in the road network. This way, we are able to discover spatiotemporal patterns and support traffic management decisions. In contrary to existing approaches, we do not rely on the distinct moving objects trajectories but on accumulating data provided by the sensors that monitor the traffic circulation within the network.

Figure 4-13: Example of traffic propagation in the Athens road network.

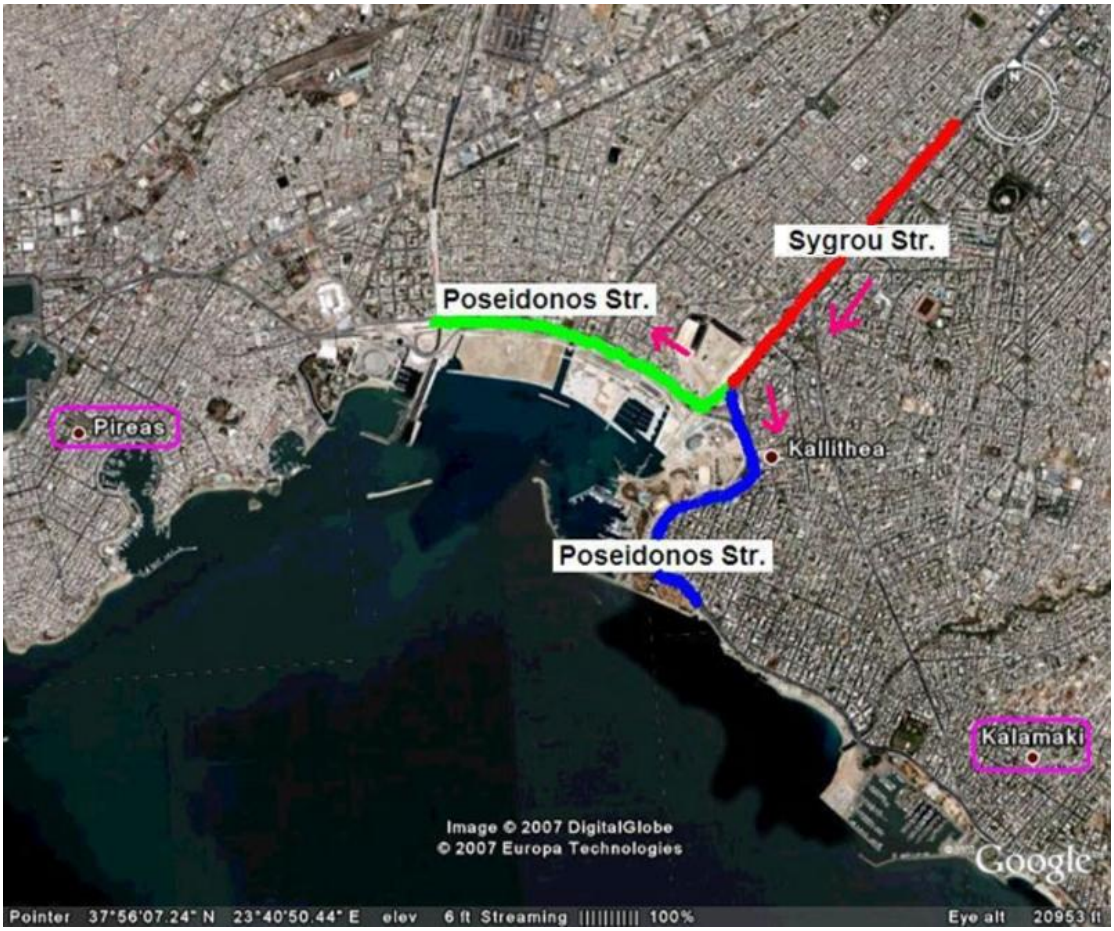


Figure 4-14: Example of traffic split in the Athens road network.

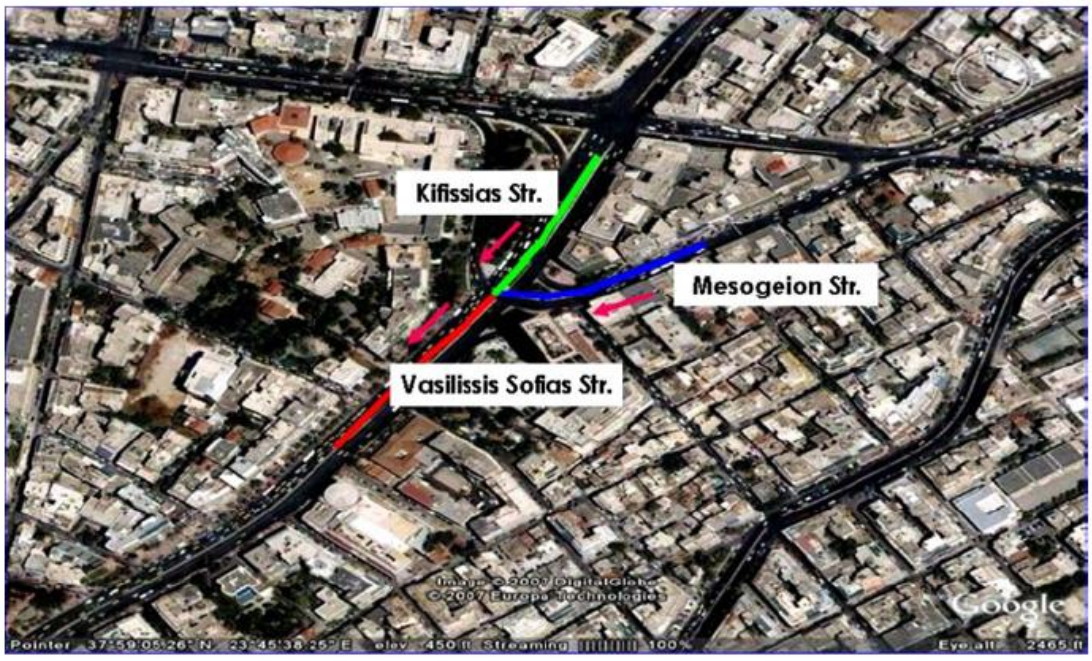


Figure 4-15: Example of traffic merge in the Athens road network.

In Figure 4-13, Figure 4-14 and Figure 4-15 we present real examples of traffic relationships in the Athens road network (the arrows in the figures show the direction of movement). In Figure 4-13, an example of traffic propagation for the Kifissias Str is depicted: objects continue to move along the Kifissias Str. In Figure 4-14, an example of traffic split is present: objects at the end of Sygrou Str have two options to enter Poseidonos Str either by turning to the left for the Kalamaki area, or by turning to the right for the Piraeus area. In Figure 4-15 an example of traffic merge is shown: objects enter Vasilissis Sofias Str from Kifissias Str and Mesogeion Str.

These examples show that detecting traffic relationships between the different road segments of a city network is quite interesting and also that the understanding of such relationships would help the city authorities to improve traffic flow and to react effectively in case of some traffic problem, like car accident or road repairing. Moreover, studying traffic relationships in the current city network, the city authorities could better arrange the construction of new roads, the extension of existing ones, the placement of traffic lights and so on.

We would like to emphasize that the TDW architecture thoroughly presented in Chapter 3 can be utilized as the repository of traffic datasets. More specifically, the two core dimensions of a TDW are the spatial and the temporal. They partition the space and the time at specific minimum granularities and several hierarchies can be built on top of these granularities. Recalling Figure 3-1, we can set the several edges of the road network and some specific time intervals as the minimum granularities of the space and the time dimension respectively. Furthermore, we can define as measure of the TDW the number of vehicles. Hence, this TDW stores aggregated data that can be used for analyzing traffic data. In other words, same as happens in the traditional KDD approach, the TDW feeds the traffic mining algorithms with appropriate data.

4.4.1. Traffic Modeling

In our model, we consider a predefined network consisting of a set of non-overlapping regions like the one depicted in Figure 4-12a. Regions might be road intersections or landmarks of interest like banks and shopping centers.

A traffic network (*TN*) consists of a set of disjoint regions, $\{r_1, r_2, \ldots, r_m\} : (r_i \cap r_j) = \emptyset, 1 \leq i, j \leq m$. ■

Objects move through these regions according to the network topology. We consider a road network modeled as a directed graph $G = (V, E)$ where the set V of vertices indicates locations (e.g. shopping centers, workplaces, crossings) and the set E of edges corresponds to direct connections (i.e., road segments) between them. More specifically, an edge e is added between two regions if exists a road segment that directly connects them. We assume that object movements take place only at the edges and also that as far as an object enters an edge it travels the whole edge. The graph model for the sample network of Figure 4-12a is depicted in Figure 4-12b.

We denote by *start*(*e*) the starting vertex of the edge *e* and by *end*(*e*) the ending vertex of *e*. We call *outgoing edges of e*, denoted as *out*(*e*), those edges that start from the node *end*(*e*). Also, we call *incoming edges of e*, denoted as *in*(*e*), those edges that end at node *start*(*e*).

We assume that aggregate mobility data are available and more specifically: for each edge, the traffic volume at adjacent time periods. Hence we consider for each edge $e \in E$ a series of time stamped measurements *(v, t)*, where *v* is the number of vehicles passed through this edge during the time period [*t*, *t*+Δ*t*) and Δ*t* corresponds to a time interval for which we are interested in aggregating traffic data. The time series of the edges of Figure 4-12b are illustrated in Figure 4-12c.

The *traffic series* of a network edge *e* during a time period [t_s, t_e] is defined as the number of vehicles passed the *e* during this period, recorded at *Δt* intervals and ordered in time: $TS = (v_i, t_i)$, where v_i is the number of vehicles crossing *e* during [t_s, t_e], $t_s \leq t_i \leq t_e$ and $\Delta t = t_i - t_{i-1}$. ■

A sample traffic series looks like this: *TS={(150, 10:00), (100,10:15), (80, 10:30), … }*.

4.4.2. Clustering Traffic Edges

In this subsection, we present our approach for clustering the edges of a road network after analyzing their traffic time series and applying a number of similarity measures.

4.4.2.1. Similarity measures between traffic edges

Let e_1, e_2 be two network edges and let $TS_1=\{(v_{1i},t_i)\}$, $TS_2=\{(v_{2i},t_i)\}$ be their corresponding traffic (time) series, $t_i \in [t_s, t_e]$. In [NMM08] we proposed a distance between two traffic edges e_1, e_2 as a weighted combination of their corresponding value based, shape based and structure based distances:

$$dis(e_1, e_2) = a * dis_{shape}(e_1, e_2) + b * dis_{struct}(e_1, e_2) + c * dis_{value}(e_1, e_2) \quad (4\text{-}1)$$

where $dis_{value}(e_1, e_2)$ is the *value based distance* between e_1 and e_2 which is given by the Euclidean distance of their corresponding traffic series (TS_1, TS_2):

$$dis_{value}(e_1, e_2) = dis_{value}(TS_1, TS_2) = \sqrt{\sum (v_1[t_i] - v_2[t_i])^2}, t_s \leq t_i \leq t_e \quad (4\text{-}2)$$

The value based distance looks for traffic series of (almost) the same values.

Except for edges of similar traffic values we are also interested in detecting edges of similar traffic shape, i.e., edges whose traffic increases and decreases with the same rate. Euclidean distance is not suitable for this case since it does not allow different baselines in the traffic series (e.g., one series fluctuating around 100 and the other series fluctuating around 30), or different scales (e.g., one series having a small amplitude between 90 and 100 and the other series having a larger amplitude between 20 and 40). To detect traffic series with similar shape (but not necessary with similar values), we apply normalization transformations [DG03] over the original traffic series. Let $\mu(TS)$, $\sigma(TS)$ be the mean and standard deviation of a traffic series $TS = \{v_i, t_i\}, 1 \leq i \leq n$. The traffic series *TS* is replaced by the *normalized traffic series* $TS' = \{v_i', t_i\}, 1 \leq i \leq n$, where:

$$v_i' = \frac{v_i - \mu}{\sigma}, \text{ where } \mu = \frac{1}{n}\sum_{i=1}^{n} v_i \text{ and } \sigma = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(v_i - \mu)^2} \quad (4\text{-}3)$$

Hence the Euclidean distance of their corresponding normalized traffic series (TS_1', TS_2')

$$EuclideanDistance(TS_1', TS_2') = \sqrt{\sum (v_1'[t_i] - v_2'[t_i])^2}, t_s \leq t_i \leq t_e \quad (4\text{-}4)$$

We define $dis_{shape}(e_1, e_2)$ as the *shape based distance* between e_1 and e_2 which is given by the Euclidean distance of their corresponding normalized (to avoid differences in baselines, scales) traffic series (TS_1', TS_2'):

$$dis_{shape}(e_1, e_2) = dis_{shape}(TS_1, TS_2) = EuclideanDistance(TS_1', TS_2') \quad (4\text{-}5)$$

Finally, $dis_{struct}(e_1, e_2)$ is the *structure based distance* between two traffic edges e_1 and e_2 and equals to the minimum number of edges between *end*(e_1) and *start*(e_2).

For each application, we can instantiate the weights a, b, c according to the measure(s) on which we wish to emphasize. If we consider the three measures separately, we realize that each measure further filters the initial set of traffic edges. More specifically, the *shape* based measure returns groups of edges with similar traffic shape, the *structure* measure looks further for neighbor edges, and finally the *value* measure further restricts the result set by searching also for value based similar traffic edges.

4.4.2.2. The algorithm Traffic-Clustering

We introduced three ways to measure the similarity between two traffic edges based either on their values or on their proximity in the network graph. Each measure reveals different aspects of the traffic problem: the value based measure discovers traffic series of similar values; the shape based measure discovers traffic series of similar shape, whereas the structure based distance discovers edges that are close to each other with respect to the graph topology.

The weights *a, b, c* of equation (4-1) are instantiated according to the specific requirements of each application. This results to a rationale (e.g. *a >> b >> c*) providing a hierarchy of traffic flow organized in different levels. For instance, if the rationale *a >> b >> c* then the hierarchy is: the level of *similar traffic shape* edges (*L*1), the level of *nearby edges* in the graph network (*L*2) and finally, the level of *similar traffic value* edges (*L*3). An example of this hierarchy is depicted in Figure 4-17.

To detect such a hierarchy we employ Traffic-Clustering, a *divisive hierarchical clustering algorithm* which is illustrated in Figure 4-16. The distance measure is that of equation (4-1), which combines the three notions of distance between traffic edges. The algorithm works as follows: Initially all traffic edges are placed in one cluster. At each step of the algorithm, a cluster is further split into subclusters (lines 7-9) according to the similarity measures dis_{value}, dis_{struct}, dis_{shape}. The order in which they are applied depends on the values of weights *a*, *b* and *c*.

```
Algorithm TrafficClustering (ListOfEdgesTS loe, weight a, weight
b, weight c)
1. //similarity measures (dis_value, dis_struct, dis_shape) are applied
2. //in the order that is defined by the weights a, b and c.
3. //we define dis1, dis2 and dis3 as the first, the second and
4. //the last measure respectively
5. //each step is completed when a split is caused by the next
6. //distance measure
7.  clusters = splitClusters(loe, dis1)
8.  clusters = splitClusters(clusters, dis2)
9.  clusters = splitClusters(clusters, dis3)
```

Figure 4-16: The algorithm TRAFFIC-CLUSTERING

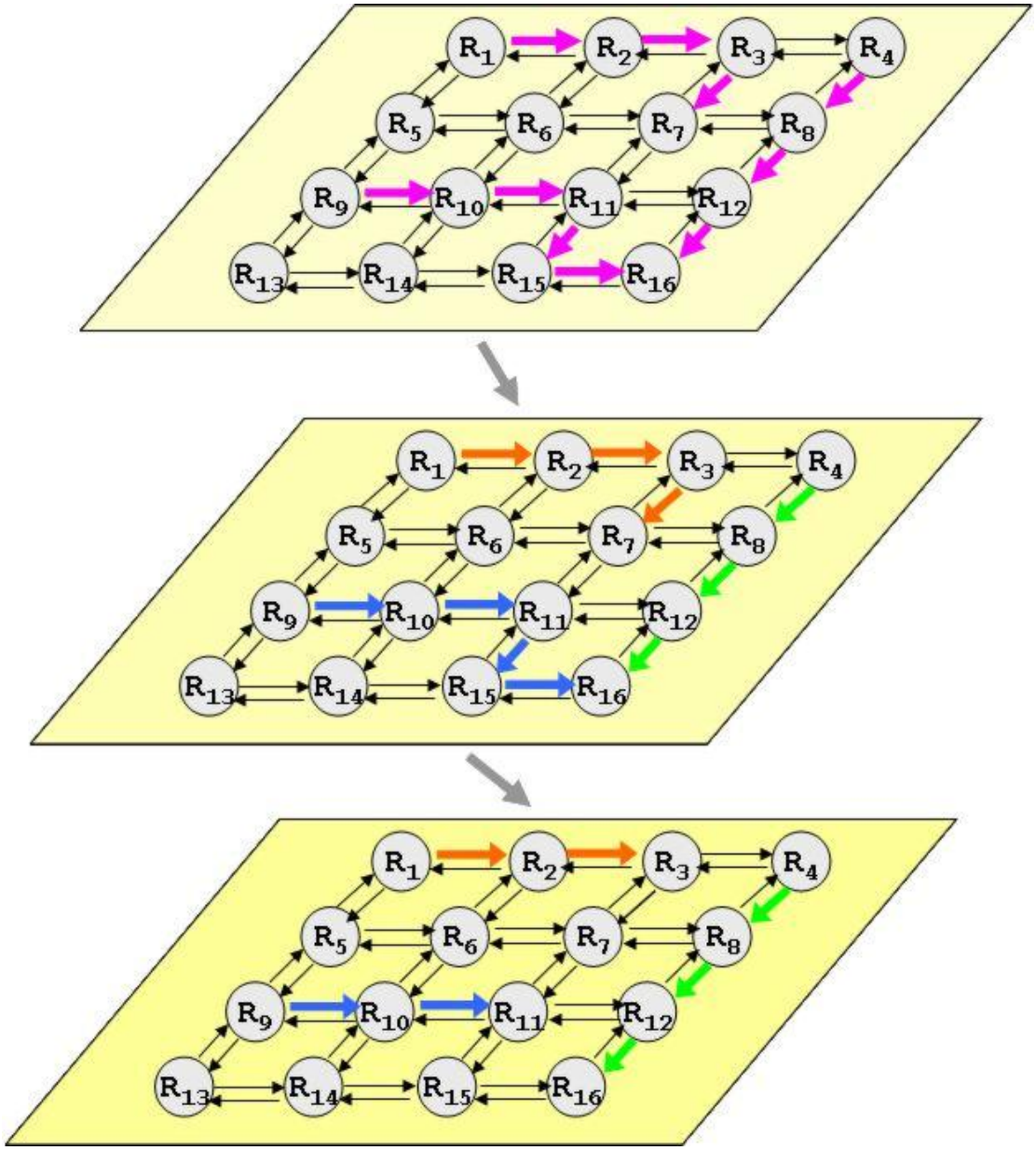


Figure 4-17: The traffic edge hierarchy–edges of the same color (mauve, green, orange, blue) belong to the same cluster.

For instance, by defining $a >> b >> c$ we indicate that we are first interested in capturing edges of similar traffic shape, then edges that lie close to each other and finally, edges that have similar values,. This way, the algorithm first favors edges of similar traffic shape (weight a), then edges that are also nearby (weight b) and finally, edges that are also similar with respect to their values (weight c). In this case, the algorithm splits the initial single cluster into subclusters according to the following three steps:

- Step 1 [Edges of similar shape]: A cluster is split into subclusters based on the shape similarity of its traffic edge members. This process is continued until a split is caused by the next distance measure, the structure based distance. At the end of this step, the clusters contain edges with similar traffic shape.

- Step 2 [Nearby edges]: The clusters generated by the previous step are further split based on the structural distance measure until a spit is caused by the traffic values distance. At this moment, the clusters contain neighboring traffic edges with similar shape.
- Step 3 [Edges of similar values]: The clusters generated by the previous step are further split based on the similar values distance. At the end of the execution, the clusters contain neighboring edges with similar values, and similar shape as well.

Discovering such a hierarchy of traffic with respect to the network edges is useful for the domain expert (e.g., city authorities), since she/ he is capable of drilling through the hierarchy from some generic group of edges with similar traffic shape, to groups of nearby edges and then to groups of edges with similar traffic values.

4.4.3. Detecting time focused Traffic Relationships

In this subsection, we present our approach for discovering traffic relationships between the edges of the road network. The rationale behind our methodology is to analyze the traffic time series and try to discover similarities between them at specific periods.

4.4.3.1. Similarity measures between traffic edges

Let e_1, e_2 be two network edges and let $TS_1=\{(v_{1i},t_i)\}$, $TS_2=\{(v_{2i},t_i)\}$ be their corresponding traffic (time) series, $t_i \in [t_s, t_e]$. We define the *value-based distance* between e_1, e_2 during periods p and p' as the absolute distance of their corresponding traffic series TS_1, TS_2 at these periods:

$$dis_{value}\left(e_1^p, e_2^{p'}\right) = dis_{value}\left(TS_1^p, TS_2^{p'}\right) \quad (4\text{-}6)$$

As we mentioned in Section 4.4.2.1, we are also interested in detecting edges of similar traffic shape, i.e., edges whose traffic increases and decreases with the same rate. To detect traffic series with similar shape (but not necessary with similar values), we utilize three well-known techniques: Euclidean distance and Dynamic Time Warping (DTW) [SC78] on normalized time series and Correlation-Coefficient. The shape similarity measure is not applied on the full dataset; we define a *shape window* $(p\text{-}w_b, p+w_a)$ in which we are looking for similarity.

As for the first two, we apply the transformation over the original traffic series that is presented in equation (4-3) and (4-5). This way, we are able to catch the changes in volumes at different time series.

We define $dis_{shapeEu}\left(e_1^p, e_2^p\right)$ as the *shape based distance* between e_1 and e_2 at period p which is defined as the *value-based distance* (4-2) of their parts of their corresponding normalized traffic series(TS_1', TS_2') at the temporal window $(p\text{-}w_b, p+w_a)$:

$$dis_{shapeEu}\left(e_1^p, e_2^p\right) = dis_{value}\left(TS_1', TS_2'\right), \text{ where for } TS_1', TS_2',\ t_s = p - w_b \text{ and } t_e = p + w_a \quad (4\text{-}7)$$

We also define $dis_{shapeDTW}\left(e_1^p, e_2^p\right)$ as the *shape based distance* between e_1 and e_2 at period p which is defined as the *distance* between the parts of their corresponding normalized traffic series (TS_1', TS_2') inside the temporal window $(p\text{-}w_b, p+w_a)$ so as to minimize the distance warping path:

$$dis_{shapeDTW}\left(e_1^p, e_2^p\right) = \sqrt{\sum\left(v_1[t_i] - v_2[t_j]\right)^2}, \text{where } p - w_b \le t_i, t_j \le p + w_a, \quad \text{and } \sum\left|v_1[t_i] - v_2[t_j]\right| = min_w\left[\sum_{k=1}^{K} d(w_k)\right] \tag{4-8}$$

where $d(w_k)$is the distance between *K* elements of the time series.

In other words, we decide not to compute the distances between the aligned values using Euclidean distance but to compute the distances between these pairs so as to minimize the distance warping path (Figure 4-18).

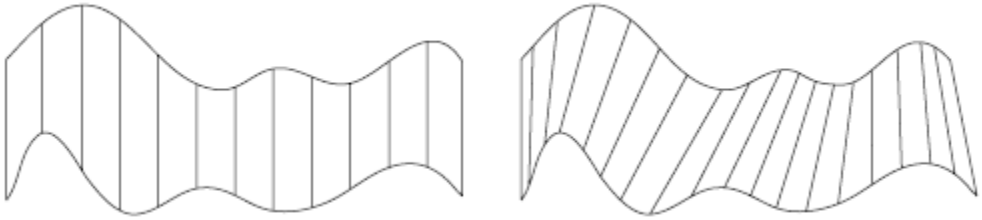


Figure 4-18: Lines between time series show value alignment used by Euclidean distance (left) and Dynamic Time Warping similarity measure (right).

Finally, we define $dis_{shapeCC}\left(e_1^p, e_2^p\right)$ as the *shape based distance* between e_1 and e_2 at period p which is defined as the Correlation-Coefficient factor (r) between the parts of traffic series inside the window $(p\text{-}w_b, p\text{+}w_a)$:

$$dis_{shapeCC}\left(e_1^p, e_2^p\right) = r\left(TS_1^{(p-w_b, p+w_a)}, TS_2^{(p-w_b, p+w_a)}\right) \tag{4-9}$$

This measure is independent from the scale of the time series and can give us a measure of similarity between them.

Note that the Euclidean and the DTW distance are not normalized and hence they cannot be used as is in order to compare two time series. To tackle this problem, we can define a maximum traffic value for each edge that shows the maximum number of vehicles that can pass from this edge at each period. This way we can define a maximum distance between two traffic time series and normalize their distances.

4.4.3.2. Traffic relationships

Detecting traffic relationships between the different road segments is an interesting problem. Within a specific time period, the traffic of a network edge might be: i) propagated "as it is" into some outgoing edge (indicating e.g., objects that continue to move in a highway), ii) split into multiple outgoing edges (indicating objects that leave a highway and follow different directions to their destination). Moreover, the traffic of an edge might be iii) the result of merge from some of its incoming edges (indicating e.g. objects that enter a highway through different directions). Also, an edge might act as iv) a sink edge (indicating e.g. a working place across the edge, where people park in the morning) or as v) a source edge (indicating e.g. the same working place in the afternoon, where people leave their offices and return home).

Below we define these relationships in a formal way. Let $p = [p_s, p_e]$ be the time period during which we want to find how the network edges are related and let TS_e^p be the part of the timeseries of edge e during that period.

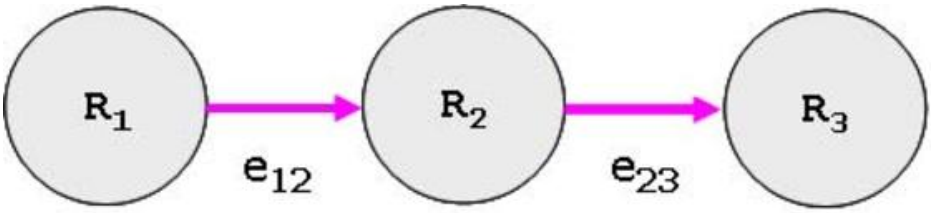


Figure 4-19: Edge e_{12} propagates its traffic into edge e_{23}.

We consider two distance thresholds: τ_v as the minimum allowed value similarity and τ_s as the minimum allowed shape similarity. Furthermore, we consider two time intervals w_b and w_a so as to define a time window inside which we look for shape similarity between two time series.

In the following definitions, dis_{shape} represents one of the shape similarity measures presented in Subsection 4.4.3.1 ($dis_{shapeEu}$, $dis_{shapeDTW}$, $dis_{shapeCC}$).

Definition 4-8 (Traffic propagate): A network edge e_1 propagates its traffic during p into an edge e_2 at p' iff:

i) $end(e_1) = start(e_2)$,

ii) $dis_{value}\left(TS_{e_1}^{p}, TS_{e_2}^{p'}\right) \geq t_v$ and

iii) $dis_{shape}\left(e_1^p, e_2^p\right) \geq t_s$. ■

Intuitively, the traffic of edge e_1 during p propagates (Figure 4-19) into e_2 during p' if most of the traffic of e_1 is directed to e_2 during this period (the required transferred amount is expressed by the threshold t_v) as well as the shape of their time series, inside the shape window, is similar (expressed by the threshold t_s).

Definition 4-9 (Traffic split): The traffic of an edge e during p is split into edges $e_i (i = 1 : k)$ at p' iff:

i) $end(e) = start(e_i)$;

ii) $dis_{value}\left(TS_e^p, \sum_i (TS_{e_i}^{p'})\right) \geq t_v$ and

i) $\forall\, 1 \leq i \leq k : dis_{shape}\left(TS_e^p, TS_{e_i}^p\right) \geq t_s$. ■

i.e., the traffic of edges $e_1, \ldots, e_k$ during p' as a total "forms" the traffic of edge e during p and the shape of their time series, inside the shape window, is similar (Figure 4-20).

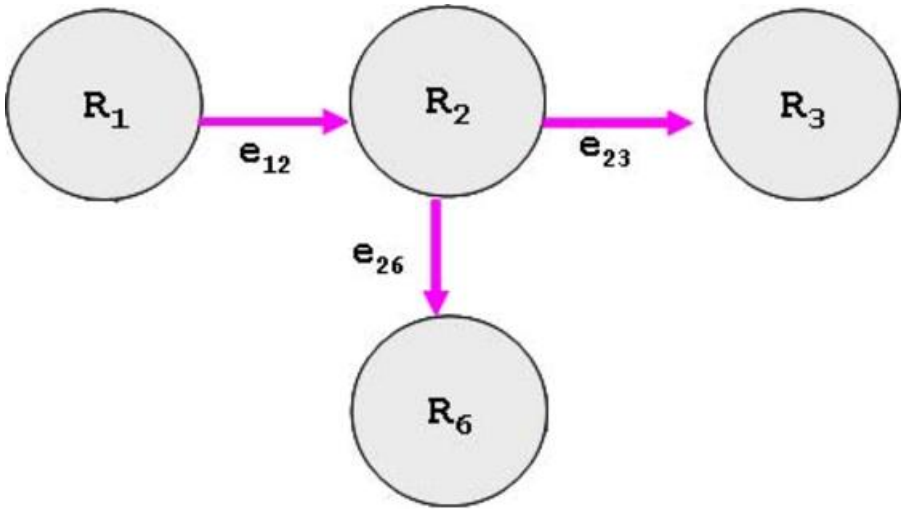


Figure 4-20: The traffic of edge e_{12} is split into edges e_{23} and e_{26}.

Definition 4-10 (Traffic merge): The traffic of an edge *e* during period *p* is the result of traffic merge from edges $e_i (i = 1 : k)$ at *p'* iff:

- $end(e_i) = start(e)$;

iii) $dis_{value}\left(\sum_i (TS_{e_i}^{p}), TS_e^{p'}\right) \geq t_v$ and

- $\forall\, 1 \leq i \leq k : dis_{shape}\left(TS_{e_i}^{p}, TS_e^{p}\right) \geq t_s$. ■

i.e., during *p* edges $e_1, \ldots, e_k$ together "form" the traffic of edge *e*.

Alternatively, we can say that the traffic of e_i during *p* has been merged (Figure 4-21) towards the traffic of *e* at *p'* and the shape of their time series, inside the shape window, is similar. The merge relationship is the opposite of the split relationship.

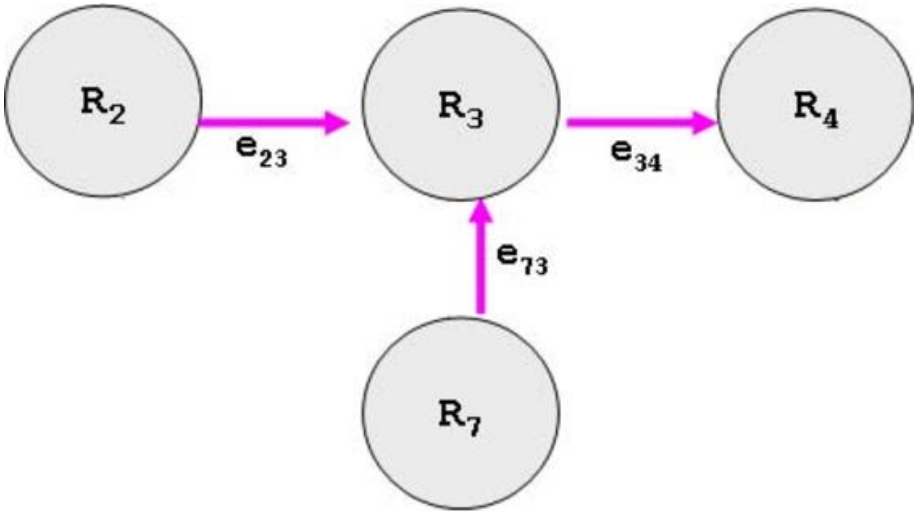


Figure 4-21: The traffic of edge e_{34} is the result of incoming traffic from edges e_{23} and e_{73}.

Definition 4-11 (Traffic sink): An edge *e* is characterized as traffic sink during *p* iff it is not related through some propagation or split relationship with its outgoing edges during that period. That is, an edge is a sink during *p* if its values are not "transferred" to some of its outgoing edges during that period. ■

Definition 4-12 (Traffic source): An edge *e* is characterized as traffic source during *p* iff during that period it is not related through some propagation or merge relationship with its incoming edges. ■

That is, an edge is a source during *p* if its traffic is not "justified" by its incoming edges during that period.

In the above definitions, we defined a relationship between two edges e_1, e_2 within a specific period *p*, i.e. how e_1 during *p* is related to e_2 during *p*. Note, however, that due to the delay of traveling from e_1 to e_2 such a relationship might hold for different periods in e_1, e_2. The most generic definition is that the traffic of e_1 during p_i is related to the traffic of e_2 during p_j , where p_i, p_j should be nearby time periods such that $j - i \leq w$, where *w* is the allowed period gap. We consider the case where $w = 0$ (i.e. $p_j = p_i$, corresponding to the same time period) or $w = 1$ (i.e. $p_j = p_i+1$, corresponding to adjacent time periods).

From the above definitions, it's clear that the traffic of an edge might be assigned one of the following characterizations: propagated, split into, merged towards, sink, source. However, there are complex cases where we can't really decide about the exact relationship between two edges.

4.4.3.3. Detecting relationships

In this paragraph, we describe our approach for discovering relationships between the edges of the network graph. Our algorithm TRAFFIC-RELATIONSHIP-DETECTOR is depicted in Figure 4-22 and as input it needs the traffic network $G = (V, E)$ and the thresholds for value and shape similarity measures as well as the size of the shape window.

The algorithm starts with a random vertice of the network and finds out how its incoming, outgoing edges are related in terms of traffic within an observation period. In the beginning, it looks for cases where only either propagation, split or merge relationship can found (lines 4-15). Functions defineOutSetEdges (line 9) and defineInSetEdges (line 14) selects the subset of outgoing and incoming set of edges that will be checked for split and merge relationships respectively. We use a heuristic approach which selects a subset of outgoing (incoming) edges so as to guarantee at least value similarity with the incoming (outgoing) edge. As for the functions CHECKFORPROPAGATE, CHECKFORSPLIT and CHECKFORMERGE they are based on the definitions given in the previous subsection. Note that each of them searches for relationships either on the same period or the consecutive one.

In more complex cases (lines 16-27), the algorithm creates possible pairs of incoming and outgoing edges and looks for all types of relationships. If more than one is found then it is possible to decide and characterizes this pair as complex. Finally, the algorithm searches for sinks and sources following the rationale that described in the previous subsection: if for a specific period an edge participates as right part in a relationship (i.e. the traffic is propagated, merged, split into it) but not in the left part of

another relationship then we consider it as a sink (lines 31-33). If the opposite one happens then it is considered as a source (lines 34-37).

```
Algorithm Traffic-Relationship-Detector(ListOfVertices lov,
ValueSimThreshold t_v, ShapeSimThreshold t_s, TimeInterval w_b, TimeInterval
w_a)
1. FOR EACH Vertice v IN lov
2.  FOR EACH Period p
3.    //check incoming and outgoing edges for v
4.    IF incomingEdges(v) = 1 AND outgoingEdges(v) = 1 THEN
5.     checkForPropagate(v.e_in, v.e_out, p, t_v, t_s, w)
6.    ELSE IF incomingEdges(v) = 1 AND outgoingEdges(v) > 1 THEN
7.     //it is defined the set of outgoing edges to look for a
8.     //relationship between the set and the incoming edge
9.     le = defineOutEdges(v)
10.    checkForSplit(v.e_in, le, p, t_v, t_s, w)
11.   ELSE IF incomingEdges(v) > 1 AND outgoingEdges(v) = 1 THEN
12.    //it is defined the set of incoming edges to look for a
13.    //relationship between the set and the outgoing edge
14.    le = defineInSetEdges(v)
15.    checkForMerge(le, v.e_out, p, t_v, t_s, w)
16.   ELSE
17.    FOR EACH pair pr IN v
18.     rels = checkForPropagate(pr.e, pr.e', p, t_v, t_s, wb)
19.     rels += checkForSplit(pr.e, pr.e_i, p, t_v, t_s, w)
20.     rels += checkForMerge(pr.e_i, pr.e, p, t_v, t_s, w)
21.      //rels stores the number of total relationships found
22.      IF rels >1 THEN
23.      //a complex relationship is found
24.      complexRel(pr, p)
25.     END IF
26.    END FOR
27.   END IF
28.  END FOR
29.   //consider as r the set of discovered relationships
30.   FOR EACH Edge e IN v
31.     IF v.e IN rightPart(r) AND NOT IN leftPart(r) THEN
32.      //the edge keeps its traffic at period p
33.      sinkEdge(e, p)
34.     ELSE IF v.e in leftPart(r) AND NOT IN rightPart(r) THEN
35.      //the edge keeps its traffic at period p
36.      sourceEdge(e, p)
37.     END IF
38.    END FOR
39. END FOR

Function checkForPropagate (Edge e, Edge e', Period p,
ValueSimThreshold t_v, ShapeSimThreshold t_s, TimeInterval w_b, TimeInterval
w_a)
```

```
1.  //let TS_e^p be the part of the timeseries of edge e during p
2.  IF dis_shape(TS_e^p, TS_e'^p) ≥ t_s THEN
3.   IF dis_value(TS_e^p, TS_e'^p) ≥ t_v THEN
4.    //a propagation relationship is found
5.    propagateRel(e, e', p)
6.   ELSE IF dis_value(TS_e^p, TS_e'^{p+1}) ≥ t_v THEN
7.    //a propagation relationship is found
8.    propagateRel(e, e', p+1)
9.   END IF
10. END IF

Function checkForSplit (Edge e, ListOutEdges e_i, Period p,
ValueSimThreshold t_v, ShapeSimThreshold t_s, TimeInterval w_b, TimeInterval
w_a)
1.  //let TS_e^p be the part of the timeseries of edge e during p
2.  // e_i(i = 1 : k) where k is the number of outgoing edges
3.  IF ∀ 1 ≤ i ≤ k : dis_shape(TS_e^p, TS_{e_i}^p) ≥ t_s THEN
4.   IF dis_value(TS_e^p, ∑_i(TS_{e_i}^p)) ≥ t_v AND THEN
5.    //a split relationship is found
6.    splitRel(e, e_i, p)
7.   ELSE IF dis_value(TS_e^p, ∑_i(TS_{e_i}^{p+1})) ≥ t_v THEN
8.    //a split relationship is found
9.    splitRel (e, e_i, p+1)
10.  END IF
11. END IF

Function checkForMerge (ListInEdges e_i, Edge e, Period p,
ValueSimThreshold t_v, ShapeSimThreshold t_s, TimeInterval w_b, TimeInterval
w_a)
1.  //let TS_e^p be the part of the timeseries of edge e during p
2.  //e_i(i = 1 : k) where k is the number of incoming edges
3.  IF ∀ 1 ≤ i ≤ k : dis_shape(TS_{e_i}^p, TS_e^p) ≥ t_s THEN
4.   IF dis_value(∑_i(TS_{e_i}^p), TS_e^p) ≥ t_v AND  THEN
5.    //a merge relationship is found
6.    mergeRel (e_i, e, p)
7.   ELSE IF dis_value(∑_i(TS_{e_i}^{p+1}), TS_e^p) ≥ t_v THEN
8.    //a merge relationship is found
9.    mergeRel (e_i, e, p+1)
10.  END IF
11. END IF
```

Figure 4-22: The algorithm TRAFFIC-RELATIONSHIP-DETECTOR

4.4.4. Experimental Study

The experiments presented in this section aim at demonstrating the applicability and usefulness of our approaches. More specifically, we experiment with Traffic-Clustering and Traffic-Relationship-Detector.

4.4.4.1. Clustering traffic edges

We used synthetic data generated by the network-based data generator developed by Brinkhoff generator [Bri02]. In particular, we generated 2000 moving objects with 400 sampled positions per each object. The network used in our experiments is depicted in Figure 4-23.

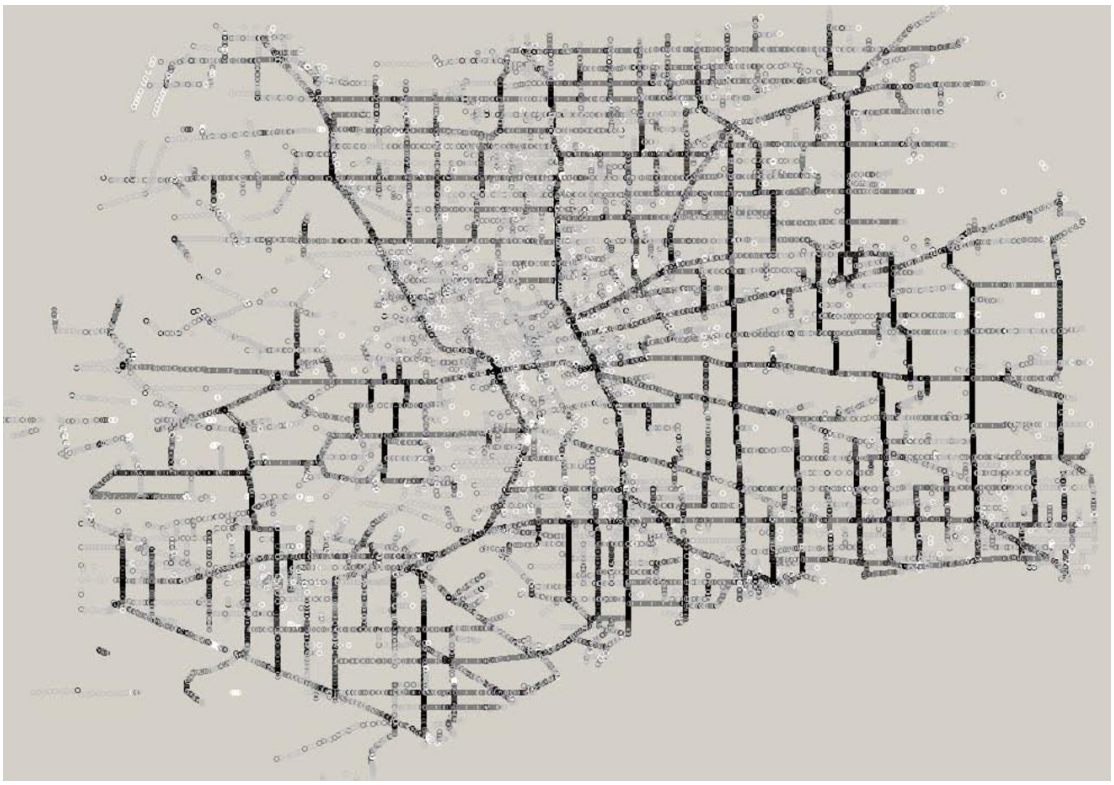


Figure 4-23: The original traffic network.

Before we proceed with the experimental results, we should note at this point that we do not consider the delay of travelling from one edge *e* to some of its outgoing edges. Rather, we assume that the time required to cross the edge *e* is smaller than the transmission rate of the sensor that monitors this edge. Also, we applied a threshold value so as to exclude from the clustering process the edges that were rarely visited by moving objects during the observation period.

In Figure 4-24, we present the clusters that were collected at the first step of the clustering algorithm, where the grouping of edges is based on their traffic shape similarity. Edges with the same color indicate network areas that share similar traffic shape. As we can observe in this figure, there are areas in the network that share the same traffic shape, even if they are non-connected. For example, the cluster depicted with the blue color is composed of three sub-areas located in different network places. This means that there exist three non-connected areas in the network that share the same traffic shape and thus, all together, they form a cluster (the blue one).

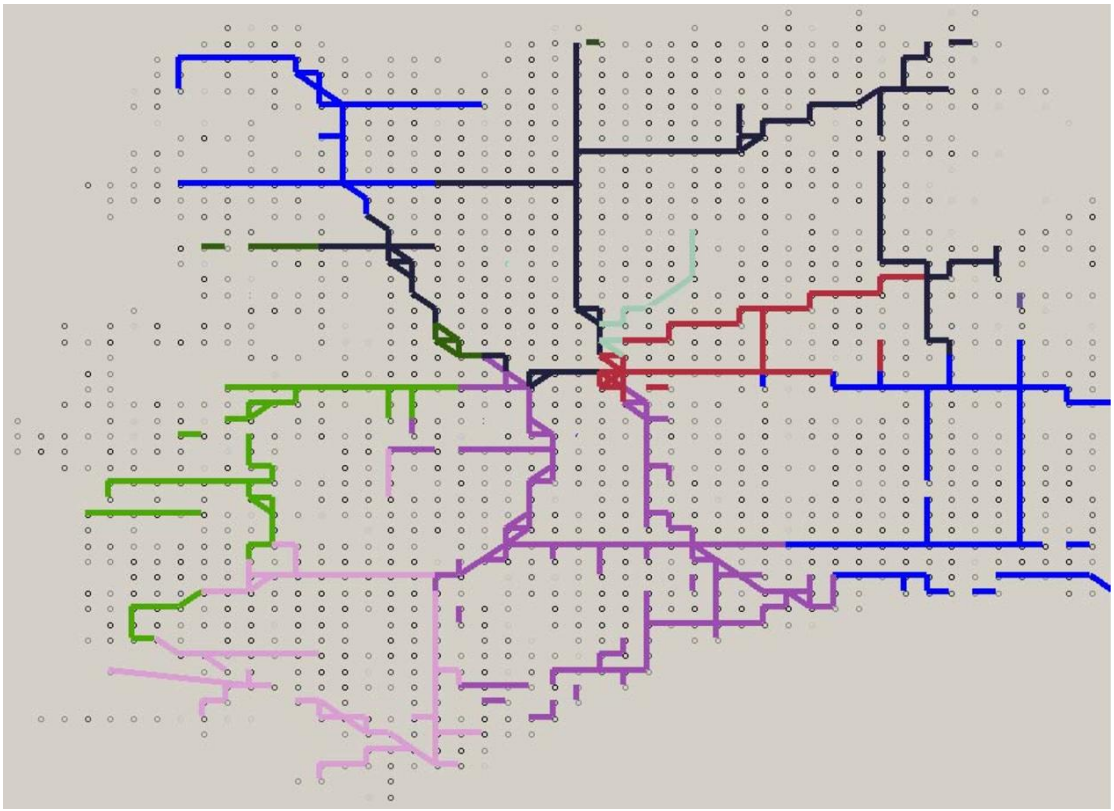


Figure 4-24: Results of Layer 1 based on grouping of edges with similar traffic shape (clusters are depicted in different colors).

In Figure 4-25, we present the clusters that were generated by the second step of the clustering algorithm, where the grouping of edges also considers their proximity in the network graph structure. Comparing to the first step of the algorithm (cf. Figure 10), we can observe that old clusters where split into new clusters, because of the proximity based distance measure applied in this step.

For example, the blue cluster described in the first step, was replaced by three new clusters (a yellow one, a blue one and a turquoise blue one). Obviously, this split took place because, as explained earlier, the initial cluster contained disjoint areas, which were placed by the distance measure applied in this step into different clusters.

Finally, in Figure 4-26, we present the result of the third step of the clustering algorithm, which looks further for traffic edges of similar values. We can see that many new clusters have emerged. The new clusters indicate areas that share the same traffic values.

As shown from the experiments, our approach can provide useful insights on the traffic problem, whereas the three level architecture facilitates the end user and enables him to monitor the traffic at different levels of abstraction.

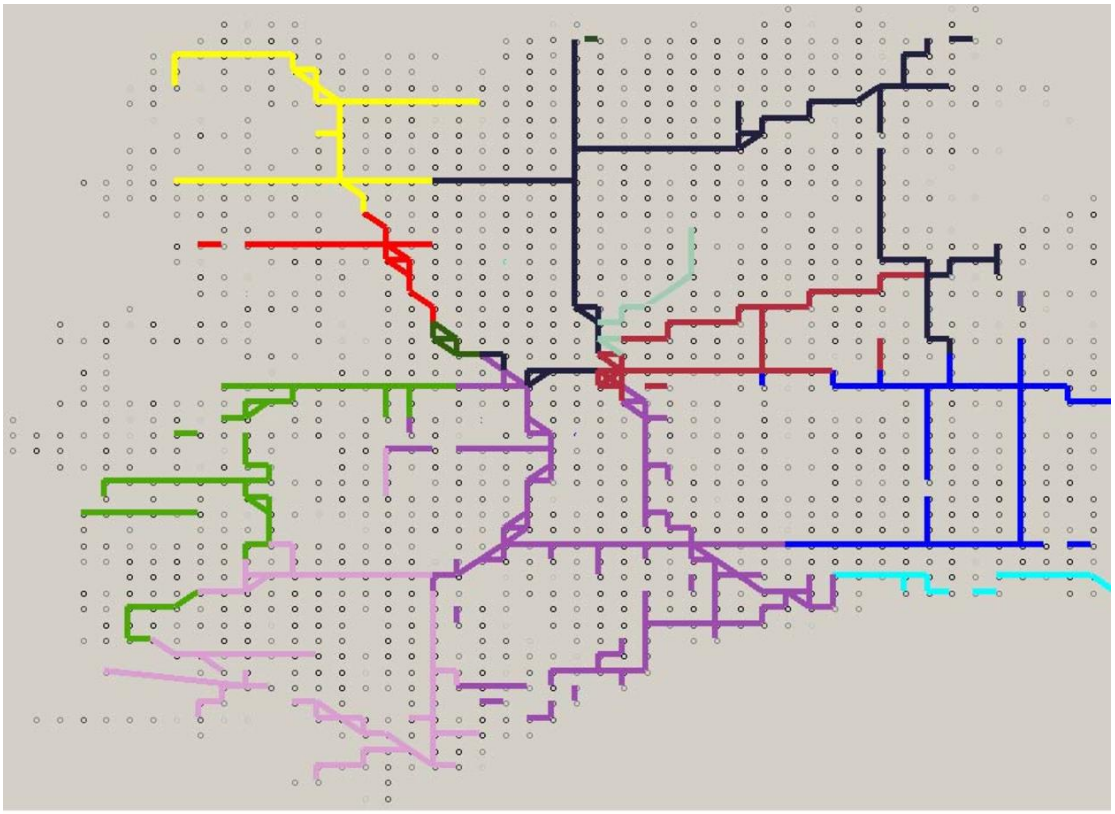


Figure 4-25: Results of Layer 2 based on graph closeness grouping (clusters are depicted in different colors).

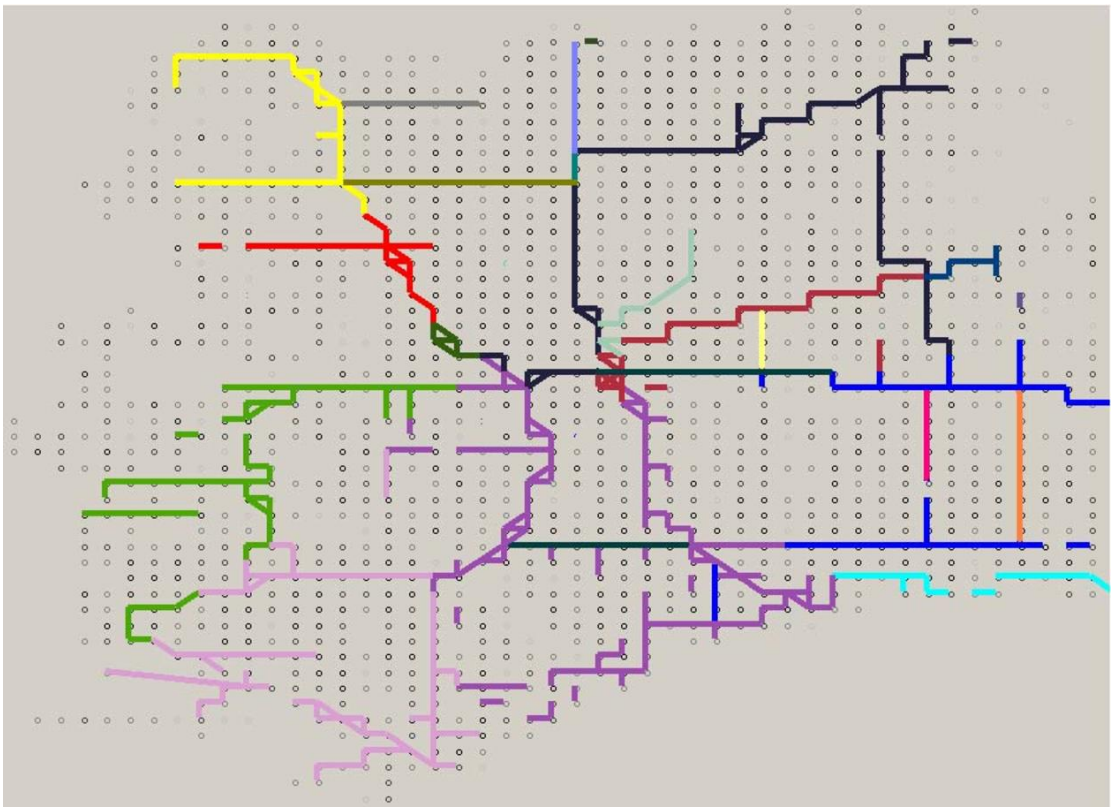


Figure 4-26: Results of Layer 3 based on grouping of edges with similar traffic values (clusters are depicted in different colors).

4.4.4.2. *Detecting time focused traffic relationships*

We used a real data set consisting of 59263 location points, belonging to 990 objects moving in the greater Milano area, map matched on 9256 edges. We experimented with various similarities measures and we compute the precision/recall:

- Precision = true positives / (true positives + false positives)
- Recall = true positives / (true positives + false negatives)

In Figure 4-27, we test different values for the value similarity threshold and we compute precision/recall for each of the possible relationships (propagate, merge, split, source, sink). We observe that the precision is high even if we apply only the value similarity filter.

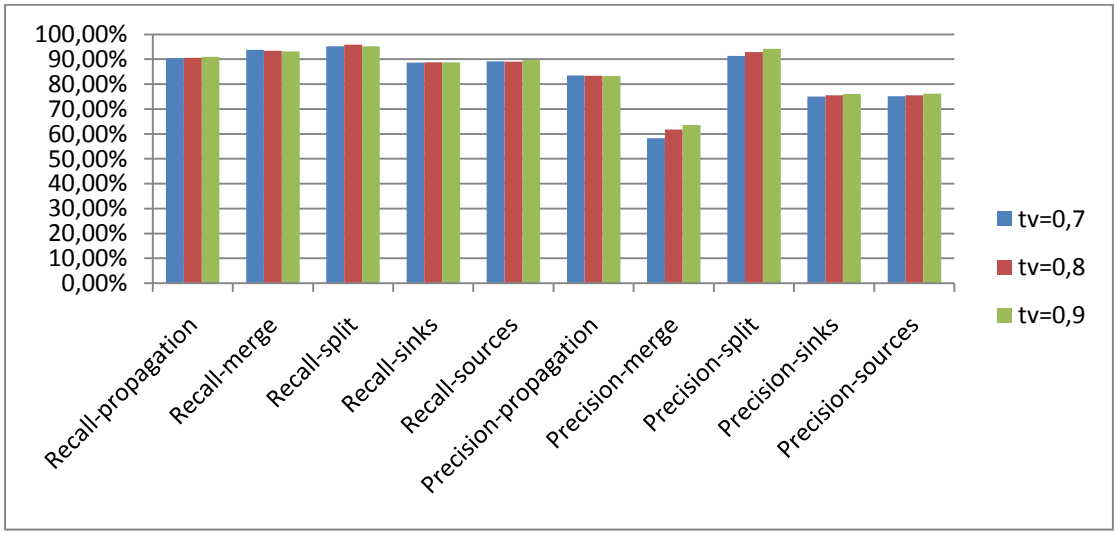


Figure 4-27: Precision/recall applying only the value similarity filter.

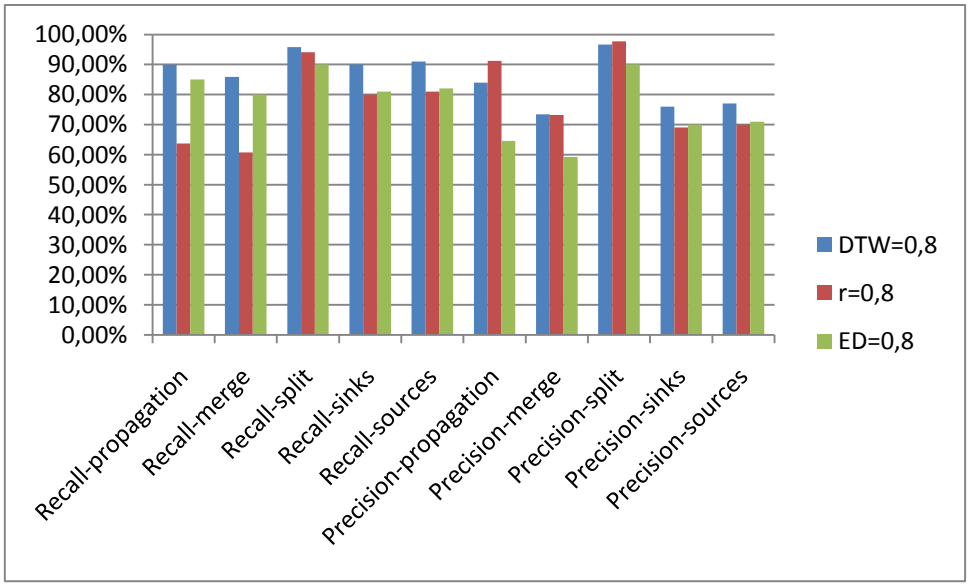


Figure 4-28: Precision/recall applying value/shape similarity for the temporal window *(p-5,p+5)*.

In Figure 4-28, we apply different shape similarity measures inside the temporal window *(p-5,p+5)*. What raises from this experiment is that using the DTW filter we can have better results comparing to

the Correlation-Coefficient filter and of course much better than using the simple Euclidean distance filter. We observe a small improvement regarding the precision of merge relationship which was not high when we applied only the value similarity filters.

4.5. Related Work

Recently, there has been considerable research on extending traditional data mining techniques to the context of trajectories (see [KNK+08] for a comprehensive survey).

4.5.1. Trajectory Clustering

A line of research relevant to our work is that of *spatiotemporal clustering* or *trajectory clustering* that aims at grouping trajectories of moving objects into groups of similar trajectories.

Relative to our approach is the work [LHL+07] for the discovery of hot routes on a road network. Hot routes correspond to sequences of road segments with heavy traffic. The road segments that participate in the formation of a hot route should share some common traffic and should also be nearby. To discover hot routes authors propose a density-based algorithm, called FlowScan, which cluster road segments based on the density of the common traffic they share. The algorithm, however, requires the trajectories of the objects that move within the network, thus cannot be applied in our problem settings.

Lee et al. [LHW07] propose a partition-and-group framework for trajectory clustering that does not consider clustering of the whole trajectories, but rather it groups together common subtrajectories. In the partition step, a trajectory is split into a set of line segments using the Minimum Description Length (MDL) principle. In the group step, similar line segments are grouped into a cluster using a density based clustering method. For each cluster, the representative trajectory is discovered which is defined as the trajectory that describes the overall movement of the trajectory partitions that belong to the same cluster. In this work, however, the trajectories of the moving objects are required, thus their solution cannot be transferred to our problem settings. Furthermore, this work concerns free movement and no some predefined network like, in our case, the road network.

Something similar is proposed in Giannotti et al. [GNP+07] where the notion of trajectory patterns (T-patterns) is introduced and appropriate trajectory mining algorithms for their discovery have been proposed. Trajectory patterns do not represent real trajectories but sequences of spatial areas of interest that are temporally related. Such areas of interest can be predefined by the user or they can be discovered in a dynamic way using some density-based algorithm.

Kalnis et al. [KMB05] introduce the notion of moving clusters for discovering groups of objects that move close to each other for a long time interval. At each timepoint, objects are grouped using a traditional clustering algorithm and then the moving clusters are detected by tracing the common data records between clusters of consecutive timepoints. However their method requires the IDs of the objects, and consequently it does not fit in our case where only the number of objects that passes through some edge/road segment is available. Furthermore, their method considers unconstrained environments, whereas we consider objects moving in a pre-defined road network. Moreover, the

authors only consider the discovery of a cluster of objects at the next timepoint whereas we consider further relationships like split and merge.

Also relevant to our work is the work on *change detection*. For example, Spiliopoulou et al. [SNT+06] propose the MONIC framework for modeling and detecting transitions between clusters discovered at consequent time points. MONIC includes both external transitions (e.g., survival, split, absorption) and internal transitions (e.g., change in size, change in location). However, their method relies on cluster members, thus cannot be directly applied to our problem settings, where only the number of objects that passes through some network edge is available and not the IDs of these objects.

Li et al. [LHK06] deal with the problem of anomaly detection in movements of objects. Instead of focusing on trajectories, they propose some methods to generate motif expressions (in order to smooth out the temporal and spatial granularities) and combine this information with other regular or spatiotemporal features in order to enhance the data mining task. However authors rely on information extracted from the trajectory data, thus the considered problem is different from our problem settings.

4.5.2. Trajectory Classification

Lee et al. [LHL+08] proposed *TraClass* for trajectory classification showing that it is necessary and important to mine interesting knowledge on partial trajectories rather than on the whole. For this reason they employ trajectory and region-based algorithms so as to partition trajectories.

Most of the approaches (e.g., [Doc06]) apply variants of Markov models to describe the movement of trajectories, and trajectories to classify are compared against each component of the model to assign it to one of the classes. Keogh and Pazzani [KP98] use a piecewise linear representation of time series and weight each segment according to its importance. This representation is used for classification, clustering and relevance feedback. In [Geu01] time series are classified by the application of patterns as test criteria in decision trees. Each pattern thereby corresponds to a temporally confined, constant model of the signal which can, for example, represent the velocity of an object.

In general, trajectories can be classified using nearest neighbor algorithms provided that an appropriate distance function is given. However, the definition of a distance function depends on the classification task and is not easily determined as various scaling, translation and noise effects have to be taken into account.

All the approaches addressing this problem are using elegant and sophisticated techniques, but without including factors related to the more profound nature of the dynamic of such systems such as the cognitive aspects and the constraints and interdependencies between entities and their interaction.

The tight relations between local patterns and global models have been the subject of some recent work. For instance, frequent local patterns have been applied to classification problems in the context of graphs by [DKW+05], where frequent subgraph discovery algorithms are used to find all topological and geometric substructures present in the data set, which are then used as features for the subsequent classification task. The utility of using frequent patterns as basic features for classification has been recently investigated in [CYH+07] for the case of frequent itemsets, showing that frequency is an effective means for ensuring a good discriminative power for pattern-based features.

4.5.3. Traffic Analysis

Shekhar et al. [SLC+01] discuss the application of traditional data warehousing and data mining techniques on sensor network measurements collected from a freeway system. Furthermore, they propose some research directions on detecting traffic flow outliers, discovering spatiotemporal association rules and sequential patterns. However, this work does not thoroughly examine the traffic problem. Authors just present a case study on how to analyze traffic data using traditional techniques.

Closer to our research is the work [LCZ+06], where a distributed traffic stream mining system is proposed: the central server performs the mining tasks and ships the discovered patterns back to the sensors, whereas the sensors monitor whether the incoming traffic violates the patterns extracted from the historical data. All the sensors that have a common pattern are grouped into the same cluster. If some violation occurs at some sensor, the sensor sends an alarm to the server, which in turn notifies all the members of the same cluster. This framework has been instantiated for frequent episode patterns. However, this work emphasizes on the description of the distributed traffic stream system, rather on the discovery of traffic related patterns.

Nakata and Takeuchi [NT04] employ probe-car data for collecting traffic information concerning much larger areas than by traditional fixed sensors. They model traffic time as time series and they apply the Auto Regression Model after removing periodic patterns. However, in this work spatial information is not taken into consideration.

4.6. Synopsis

In this chapter, we introduced trajectory *interaction patterns* aiming to extracting semantics as well as *traffic patterns* as relationships between the road segments of a city network.

Interaction patterns are concise descriptors of regions, in terms of both space and relations of similarity between trajectories that can help us understand movement behaviors. Towards this purpose, we proposed an approach that can discover what is happening to a single object by looking not only at its trajectory, but also at the context (to the presence of other objects and their interaction with the moving object under analysis) where it moves.

We also studied the problem of analyzing traffic on a road network by considering the traffic time series of each road segment and utilizing similarity measures. We exploit various techniques so as to discover traffic relationships like propagation, split and merge of traffic among these road segments as well as traffic sink and source.

5. A Framework for Trajectory Data Warehousing

In the previous chapters we studied data warehousing & mining techniques on trajectory data. In this chapter we focus on the design and development of a system that incorporates the techniques studied in the previous chapters. More specifically, we implemented T-WAREHOUSE, a system that incorporates all the required steps for Visual Trajectory Data Warehousing, from trajectory reconstruction and ETL processing to Visual OLAP analysis on mobility data. We also present our approach for reconstructing trajectories. The outline of the chapter is as follows: Section 5.1 introduces the issues being related to the data warehousing techniques on mobility data, whereas Section 5.2 motivates our research. Section 5.3 presents the architecture of T-WAREHOUSE and describes its various components whereas Section 5.4 illustrates its specifications. Finally, Section 5.5 presents the experimental study and Section 5.6 closes the chapter providing the conclusions.

5.1. Introduction

As it has already been mentioned, the motivation behind a TDW is to transform raw trajectories to valuable information that can be used for decision making purposes in ubiquitous applications, such as Location-Based Services (LBS), traffic control management, etc. Intuitively, the high volume of raw data produced by sensing and positioning technologies, the complex nature of data stored in trajectory databases and the specialized query processing demands make extracting valuable information from such spatiotemporal data a hard task. For this reason, the idea is to extend traditional aggregation techniques so as to produce summarized trajectory information and provide OLAP style analyses.

This analysis can be efficiently offered by a TDW. However, various issues have to be considered:

- the presence of a preprocessing phase dealing with the explicit construction of the trajectories, which are then stored into a Moving Object Database (MOD) that offers powerful and efficient operations for their manipulation;
- the implementation of an efficient trajectory-oriented Extract-Transform-Load (ETL) process;
- the incorporation of appropriate aggregation mechanisms that will follow the trajectory oriented cube model;
- the design of a Visual OLAP interface that allows for multidimensional and interactive analysis.

5.2. Motivation Issues

One could mention an abundance of applications that would benefit from the aforementioned approach. Let us consider an advertising company which is interested in analyzing mobility data in different areas of a city so as to decide upon road advertisements (placed on panels on the roads). They are interested in analyzing the demographical profiles of the people visiting different urban areas of the city at different time intervals of the day so as to decide about the proper sequence of advertisements that will be shown on the panels at different time periods. This knowledge will enable them to execute more focused marketing campaigns and apply a more effective strategy.

To further motivate the demonstration scenario of T-WAREHOUSE, below we appose some interesting questions that an analyst could interactively try to answer via the functionalities offered by T-WAREHOUSE:

- Where does the highest traffic appear? At what hour?
- What happens exactly at the road network level?
- How does the movement propagate from place to place?

5.2.1. Our contributions

Based on our recent results in the field [MFN+08a], which tackles the problem of Trajectory Data Warehousing in all its aspects, as a proof-of-concept, we developed T-WAREHOUSE, a system for Visual Trajectory Data Warehousing. We summarize these aspects below accompanied with our contributions as presented in [MFN+08a], [MFN+08b], [MT09b], [LMF+10]:

- Sampled positions received by GPS-enabled devices need to be converted into trajectory data and to be stored in a MOD; to this end, we propose a *trajectory reconstruction* technique that transforms sequences of raw sample points into meaningful trajectories.
- We describe the architectural aspects of our framework as well as various research challenges that are tackled.
- We investigate the power, flexibility and efficiency of our framework for applying OLAP analysis on real world mobility data.

5.3. System Architecture

The overall architecture of T-WAREHOUSE is illustrated in Figure 5-1. More specifically, mobile devices are transmitting periodically the latest part of their trajectory. This vast amount of data collected by all subscribed users is forwarded to trajectory reconstruction software, whose purpose is to perform some basic trajectory preprocessing. This may include trajectory reconstruction (so as to decide which points are part of which trajectories), as well as techniques to handle noisy values. These trajectories are stored to a Moving Object Database (MOD) wherein appropriate querying and Extract-Transform-Load (ETL) processes are applied (possibly taking into account various types of

infrastructural geodata) so as to derive information about trajectories (e.g. trajectory content in different granularities, aggregations, motional metadata etc.) to feed in the TDW. Storage of raw trajectories into the MOD and processed trajectories into the TDW are materialized through mapping constructs of the corresponding models.

A TDW serves two core needs: to provide the appropriate infrastructure for advanced reporting capabilities and to facilitate the application of trajectory mining algorithms on the aggregated data. According to end users needs, they could have access either to basic reports or OLAP-style analysis. What-if scenarios and multidimensional analysis are typical examples of analytics that could served by a TDW.

Additionally, incorporating GIS layers could result in a richer conceptual model providing thus more advanced analysis capabilities. Combining trajectory data with thematic layers (such as geographic, topographic and demographic layers) could enhance the analytics capabilities of potential applications. Finally, on top of such functionality we have further developed an advanced module for performing OLAP operations in a visual way.

In the following paragraphs, we present the main components and we discuss their functionality accompanied by our contributions.

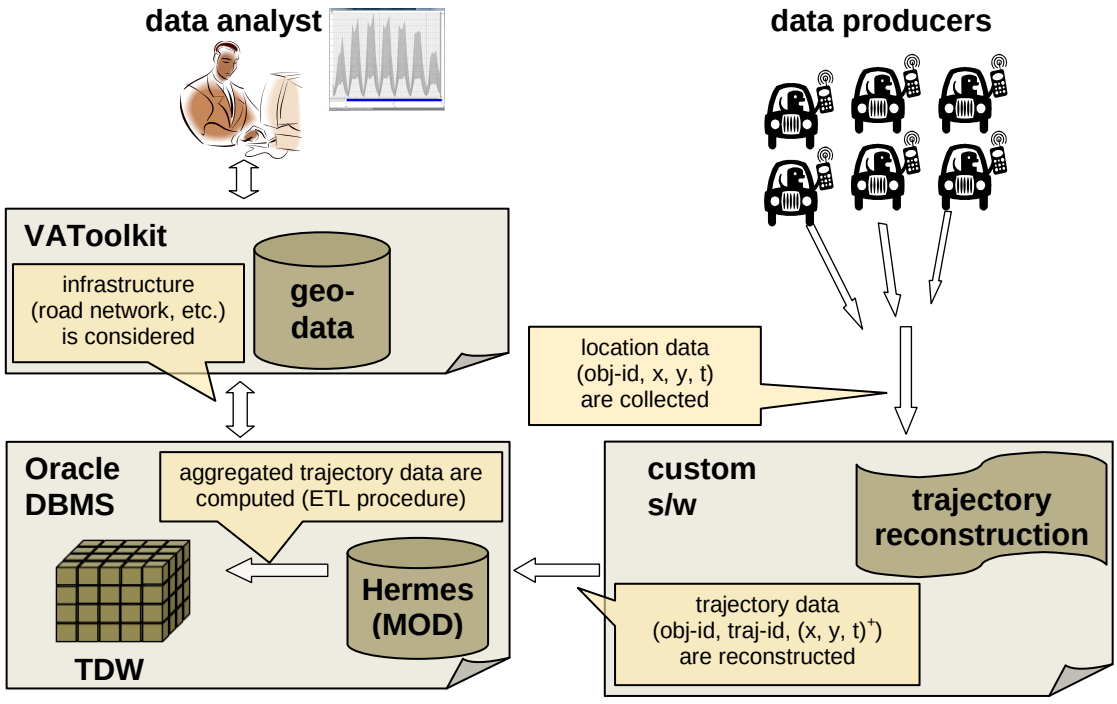


Figure 5-1: T-Warehouse architecture.

5.3.1. Trajectory Reconstruction

As already discussed, collected raw data represent time-stamped geographical locations (Figure 5-2a). Apart from storing these raw data in the MOD, we are also interested in reconstructing trajectories

(Figure 5-2b). The so-called *trajectory reconstruction* task is not a straightforward procedure. Having in mind that raw points arrive in bulk sets, we need a filter that decides if the new series of data is to be *appended* to an existing trajectory or not. In [MFN+08a], we presented an algorithm for trajectory reconstruction.

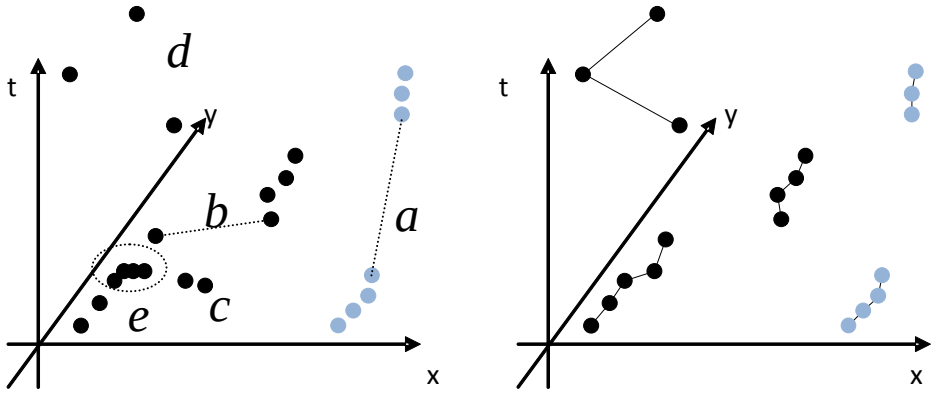


Figure 5-2: a) raw locations, b) reconstructed trajectories.

In this work, we assume this filter to be part of a trajectory reconstruction manager, along with a simple method for determining different trajectories, which applies it on raw positions. Taking into consideration that the notion of trajectory cannot be the same in every application due to the fact that different requirements and semantics arise, we define the following generic trajectory reconstruction parameters:

- *Temporal gap between trajectories* gap_{time}: the maximum allowed time interval between two consecutive time-stamped positions of the same trajectory for a single moving object. As such, any time-stamped position of object o_i, received after more than gap_{time} time units from its last recorded position, will cause a new trajectory of the same object to be created (case *a* in Figure 5-2a) .
- *Spatial gap between trajectories* gap_{space}: the maximum allowed distance in 2D plane between two consecutive time-stamped positions of the same trajectory. As such, any time-stamped position of object o_i, with distance from the last recorded position of this object greater than gap_{space}, will cause a new trajectory to be created for o_i (case *b* in Figure 5-2a).
- *Maximum speed* V_{max}: the maximum allowed speed of a moving object. It is used in order to determine whether a reported time-stamped position must be considered as noise and consequently discarded from the output trajectory. When a new time-stamped location of object o_i is received, it is checked with respect to the last known position of that object, and the corresponding instant speed is calculated. If it exceeds V_{max}, this location is considered as noise and (temporarily) it is not considered in the trajectory reconstruction process (however, it is kept separately as it may turn out to be useful again – see the parameter that follows) (case *c* in Figure 5-2a).
- *Maximum noise duration* $noise_{max}$: the maximum duration of a noisy part of a trajectory. Any sequence of noisy time-stamped positions of the same object will result in a new trajectory

given that its duration exceeds $noise_{max}$. For example, consider an application recording positions of pedestrians where the maximum speed set for a pedestrian is V_{max} = 3 m/sec. When he/she picks up a transportation mean (e.g., a bus), the recorded instant speed will exceed V_{max}, flagging the positions on the bus as noise. The maximum noise length parameter stands for supporting this scenario: when the duration of this sequence of 'noise' exceeds $noise_{max}$, a new trajectory containing all these positions is created (case *d* in Figure 5-2a).

- *Tolerance distance* D_{tol}: the tolerance of the transmitted time-stamped positions. In other words, it is the maximum distance between two consecutive time-stamped positions of the same object in order for the object to be considered as stationary. When a new time-stamped location of object o_i is received, it is checked with respect to the last known position of that object, and if the distance of the two locations is smaller than D_{tol}, it is considered redundant and consequently discarded (case *e* in Figure 5-2a).

The proposed TRAJECTORY-RECONSTRUCTION algorithm that considers the above parameters is illustrated in Figure 5-3. The input of the algorithm includes raw data points (i.e., time-stamped positions) along with object-id, and a list containing the partial trajectories processed so far by the *trajectory reconstruction manager*; these partial trajectories are composed by several of the most recent trajectory points, depending on the values of the algorithm parameters.

As a first step (lines 1-6), the algorithm checks whether the object has been processed so far, and, if so, retrieves its partial trajectory from the corresponding list, while, in the opposite case, creates a new trajectory and adds it to the list. Then (lines 7-31), it compares the incoming point P with the tail of the partial trajectory (LastPoint) by applying the above mentioned trajectory reconstruction parameters:

- it rejects P if it is closer than D_{tol} to LastPoint (lines 7-12) or
- it rejects P when a speed greater than V_{max} is calculated, unless the $noise_{max}$ case is triggered (lines 13-18) or
- it creates a new trajectory if the temporal duration between P and LastPoint is longer than gap_{time} (lines 8-12 and 24-27) or their spatial distance is greater than gap_{space} (lines 19-22);

in any other case, it reports LastPoint in the partial trajectory and replaces it with P.

The above procedure supports the reasonable requirement for detecting one trajectory per trip: Let us consider the case where a tracked user is traveling from home to work in the morning and from work to home in the evening, keeping his/her tracking device (e.g., GPS) always active. In this case, during the time the car is parked there are no spatial gaps, and no maximum speed problems, which might cause a new trajectory creation. Moreover, the GPS outputs a position every second, so no temporal gaps initially exist; however, since the car is not moving, the algorithm eliminates all points reported during the non-moving interval, and, consequently an artificial temporal gap is created (i.e., only the first point after the car parking and the last before starting moving again exist in the trajectory reconstruction algorithm). As a consequence, the algorithm detects the temporal gap and creates new trajectories, as needed, based only on the information that the tracked object stopped moving for a sufficiently large temporal period (i.e., greater than gap_{time}).

```
Algorithm Trajectory-Reconstruction (PartialTrajectories List, P
Point, OId ObjectId)
 1.   IF NOT PartialTrajectories.Contains(OId) THEN
 2.     CTrajectory=New Trajectory;
 3.     CTrajectory.AddPoint(P);
 4.     PartialTrajectories.Add(CTrajectory);
 5.   ELSE
 6.     CTrajectory=PartialTrajectories(OId);
 7.     IF Distance(CTrajectory.LastPoint,P)<= D_TOL THEN
 8.       IF P.T – CTrajectory.LastPoint.T > gap_Time THEN
 9.         Report Ctrajectory.LastPoint;
10.         CTrajectory.Id=CTrajectory.Id+1;
11.         CTrajectory.AddPoint(P);
12.       ENDIF
13.     ELSEIF Speed(CTrajectory.LastPoint,P)> V_max THEN
14.       IF P.T – CTrajectory.LastPoint.T > noise_max THEN
15.         Report CTrajectory.Noise;
16.       ELSE
17.         CTrajectory.AddNoise(P);
18.       ENDIF
19.     ELSEIF Distance(CTrajectory.LastPoint,P)> gap_space THEN
20.       Report Ctrajectory.LastPoint;
21.       CTrajectory.Id=CTrajectory.Id+1;
22.       CTrajectory.AddPoint(P);
23.     ELSE
24.       IF P.T – CTrajectory.LastPoint.T > gap_Time THEN
25.         Report Ctrajectory.LastPoint;
26.         CTrajectory.Id=CTrajectory.Id+1;
27.         CTrajectory.AddPoint(P);
28.       ELSE
29.         CTrajectory.AddPoint(P);
30.       ENDIF
31.     ENDIF
32.   ENDIF
```

Figure 5-3: The TRAJECTORY-RECONSTRUCTION algorithm.

5.3.2. MOD Engine

HERMES [PTV+06] is a robust framework capable of aiding a spatiotemporal database developer in modeling, constructing and querying a database with dynamic objects that change location, shape and size, either discretely or continuously in time. HERMES provides spatiotemporal functionality to state-of-the-art Object-Relational DBMS (ORDBMS). The whole functionality is packaged and provided as a data cartridge using the extensibility interface of Oracle.

HERMES is used as the repository of trajectories that are utilized by the tool T-WAREHOUSE. Additionally, it provides the functionality that is necessary during the TDW feeding phase. Some typical forms of queries that may be employed during the ETL are presented below:

- "*View RELtrajectories*": It retrieves trajectories in relational form specified by an object and a trajectory identifier.

- "*View MODtrajectories*": It retrieves trajectories in object-relational form specified by a trajectory identifier. We should note that whenever the result of a query is a trajectory, namely a complex object, the corresponding retrieved column is transformed in a string format as for the browser to be able to visualize it.
- "*(Spatial) Intersection operator*": It restricts a trajectory inside a given geometry, which is specified as an Oracle's Spatial sdo_geometry (e.g. a rectangle, a polygonal region etc).
- "*(Temporal) Intersection operator*": It restricts a trajectory inside a given time period.
- "*(Multi-Temporal) Intersection operator"*: It restricts a trajectory inside a set of (possibly disjoint) time periods.
- "*Topological operator*": It checks (relate operator) whether a sdo_geometry object type, satisfies a specific topological relationship (identified by a string mask; e.g. 'ANYINTERACT') with the projection of a trajectory at a given timepoint. The result of the query is a boolean value.
- "*within_distance operator*": It checks whether a trajectory at a given timepoint is within some given distance from a sdo_geometry object type.
- "*distance operator*": It returns the distance of a trajectory at a given timepoint from another trajectory projected at another given timepoint.
- "*enter operator*": This query demonstrates the construction of a trajectory at query time and for this trajectory it returns the timepoint when this trajectory entered inside a given area (sdo_geometry).
- "*leave operator*": This query demonstrates the construction of a trajectory at query time and for this trajectory it returns the timepoint when this trajectory left from an area (sdo_geometry).
- "*directional operator*": It returns the direction of the linestring formulated between two points. The first point is the projection of a trajectory at a given timepoint and the second point is given as a sdo_geometry point.
- "*speed operator*": It returns the speed of a trajectory at a given timepoint.

5.3.3. TDW Feeding

Let us consider as a running example a sample TDW schema, illustrated in Figure 5-4, which includes a *spatial* (SPACE_DIM) and a *temporal* (TIME_DIM) *dimension* describing geography and time, respectively. Non-spatiotemporal dimensions can be also considered. For instance, the schema in Figure 5-4 contains the dimension OBJECT_PROFILE_DIM which collects demographical information, such as gender, age, job, of moving objects.

Apart from the keys to dimension tables, the fact table also contains a set of measures representing aggregate information. The measures considered in the TDW schema of Figure 5-4 include the *number of distinct trajectories* (PRES), the *average traveled distance* (DISTANCE), the *average travel duration* (TIME), the *average velocity* (VELOCITY) and some auxiliary measures (i.e. CROSSX, CROSSY, CROSST), for a particular group of people (having a certain profile) moving in a specific spatial area during a

specific time period. Actually the schema is the same as the one presented in Figure 3-8, there is a slight differentiation in the names (but not in the notion) of the measures.

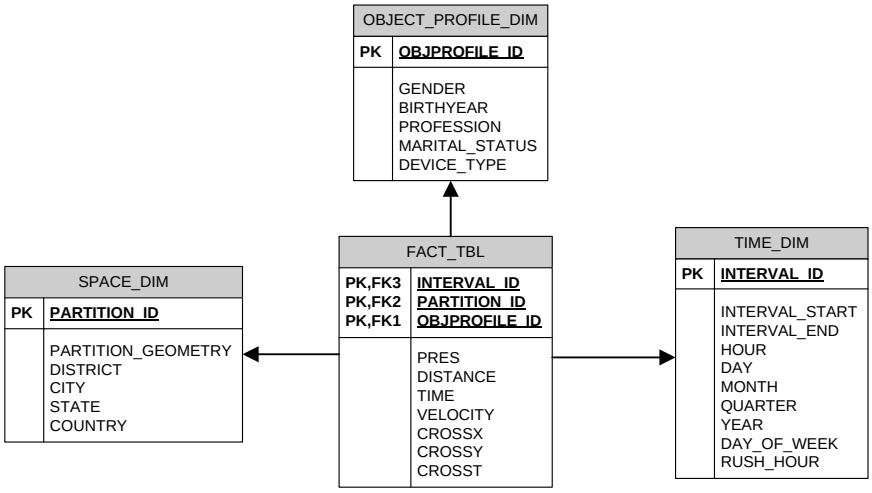


Figure 5-4: A sample TDW schema used by T-WAREHOUSE.

The TDW is to be fed with aggregate trajectory data; this is achieved by employing an efficient ETL process so as to fill in the measures of the TDW with the appropriate numeric values for each base cell. Our ETL process, thoroughly presented in 3.3.1, detects the trajectory portions that lie within the base cells. This step actually corresponds to spatiotemporal range queries that return not only the identifiers but also the portions of trajectories that satisfy the range constraints. To efficiently support the above described storage as well as trajectory-based query processing requirements, we use the HERMES MOD engine (as we presented in paragraph 5.3.2).

5.3.4. Aggregation

Aggregation capabilities over measures are offered for OLAP purposes (i.e., how the measures at a lower level of the cube hierarchy can be exploited in order to compute the measures at some higher level of the hierarchy). A peculiarity with trajectory data is that a trajectory might span multiple base cells. Hence in the aggregation phase we have to cope with the so called *distinct count problem* [TKC+04]. This is problematic since, once loaded in the TDW, the identifiers of the trajectories are lost. This problem causes *aggregation* hindrances in OLAP operations, for example in the computation of the measure PRES that should return the number of *distinct* trajectories of a certain profile crossing a spatiotemporal cell. This affects also the other measures (*average*) defined on top of PRES. In order to face this problem, we use an approximate solution, presented in paragraph 3.3.2, which turns out to perform effectively. Assuming that the PARTITION_GEOMETRY in Figure 5-4 is a regular grid, in the auxiliary measures CROSSX, CROSSY and CROSST we store respectively the number of *distinct* trajectories of a certain profile crossing the spatial/temporal border of two adjacent cells along the x/y/t axis. Knowing the number of trajectories crossing the border between cells is helpful in correcting the errors due to the duplicates when aggregating such cells (see paragraph 3.3.2 for more details).

5.3.5. OLAP Operations and Visualization.

In order to overcome these limits we developed visual OLAP operations, by using the Visual Analytics Toolkit (VAToolkit) [AAW07], an interactive Java-based geographical information system. This toolkit permits a user to see geo-referenced data over a map and, it also offers functionalities to handle temporal data, by using graphs or animations, according to the type of data to analyze.

The advantages of our system are manifold. First the user can visualize the partition of the spatial domain over the map which the spatial data refer to. Additionally, the user can graphically select an area and apply roll-up and drill-down operations in order to obtain, respectively, a more abstract or detailed view of such an area. To these views the user can apply a variety of specialized visualization techniques, which provide insightful understanding of the measures contained in the TDW.

In summary, the visual interface we implemented allows the user to easily navigate the data stored inside the TDW at different levels of the hierarchies, to have an overall vision of the data in time and in space or to focus on some particular measures, spatial areas or temporal intervals.

5.4. System Demonstration

In this paragraph, we demonstrate the functionality of the system using a large real dataset of a fleet of cars moving in the metropolitan area of Milan (Italy). The dataset consists of two millions of raw location records that represent the movement of 17,000 objects (i.e. about 200,000 trajectories) moving during a week period from Sunday to Saturday.

Before showing the T-WAREHOUSE at work we describe the specific features of the TDW prototype used in this scenario.

The user can choose the base granularity both for the spatial and temporal dimension and the corresponding hierarchies. We set a grid of rectangles (PARTITION_GEOMETRY in Figure 5-4), the size of which is 300×400 m^2, and time intervals of 1 hour, as base granularity. The spatial hierarchy consists of a set of grids aggregating groups of spatially adjacent base cells, whereas the temporal hierarchy is hour - 3-hours interval – day – week.

GUI and Visual analytics. We next present the functionalities provided to the analyst by the T-WAREHOUSE. By using our system, it is simple to handle and visualize the spatiotemporal grids of the TDW at various levels of granularities. If the roll-up operation involves the spatial dimension, visually this affects the granularity of the grid which becomes larger. The inverse operation is the drill-down which increases the level of detail of data; it allows the user to descend the hierarchies. In Figure 5-5, an example of drill down is presented so as to focus on the centre of Milan.

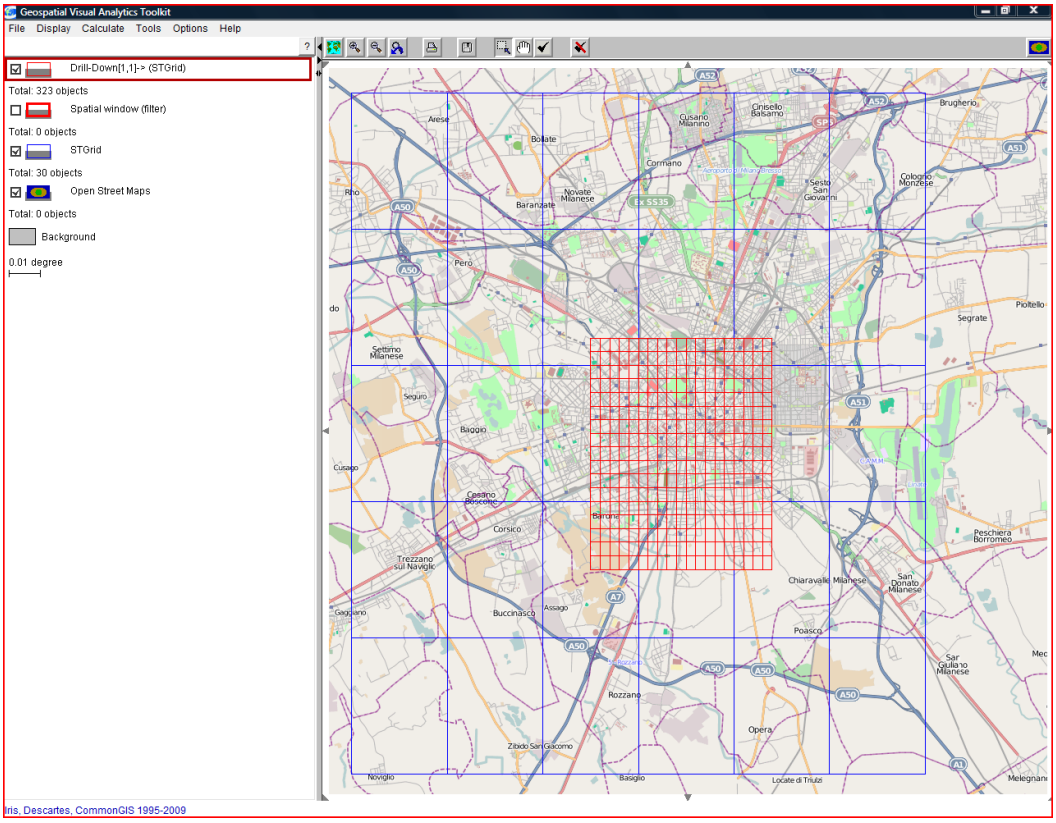


Figure 5-5: Drill down operation in T-WAREHOUSE.

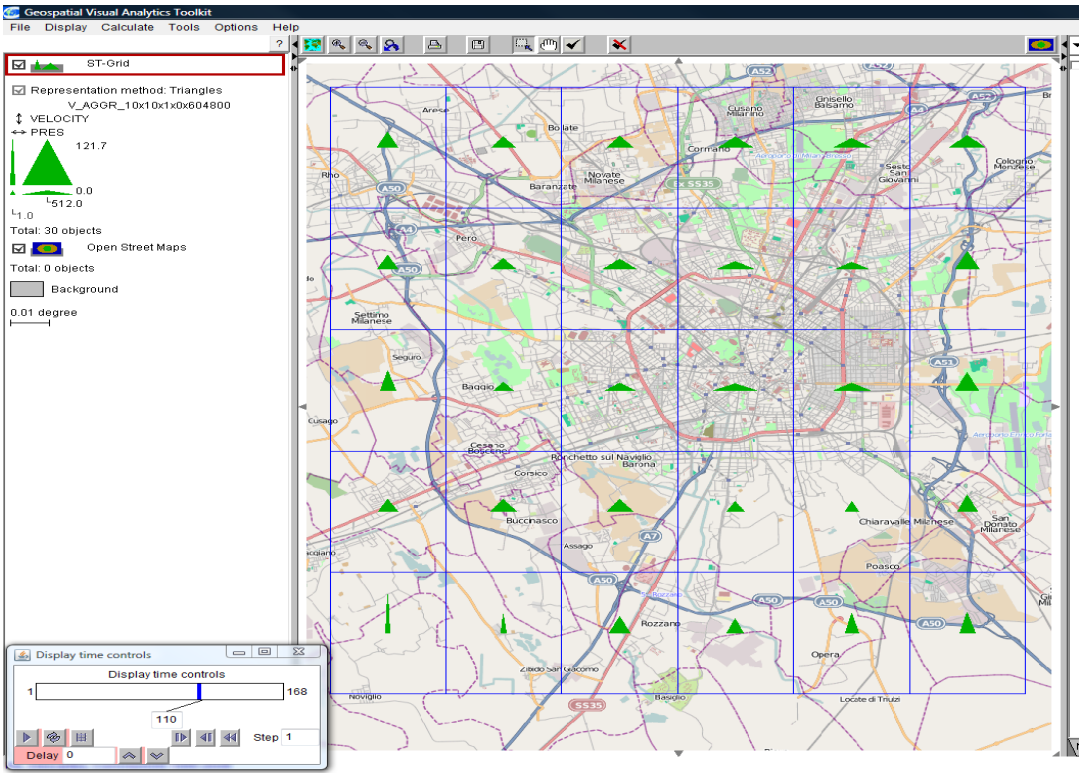


Figure 5-6: Relationship between Presence and Velocity.

Starting from the visualization of the space, one can then decide to highlight some measures, which can be visualized according to several methods.

In the *Triangle* visualization style, a triangle is drawn in each grid cell at a given level of the TDW hierarchy. The base and the height of such a triangle correspond to the value of two selected measures that the user wants to analyze.

As an example, Figure 5-6 shows a screenshot of an animation that illustrates the variation of the speed and the presence along each hour of the whole week, using the triangle style. The height of the triangle represents the Velocity whereas the base represents the Presence. Note the underlying map of Milan, which allows us to better understand the traffic phenomenon. The presence is higher in the centre and this has a strong impact on the speed of cars that is very low. On the other hand, along the ring roads the speed is higher except in the north-east zone, where the larger number of cars slows down the traffic.

The *unclassified choropleth map* is a visualization style, in which all the grid cells are filled with a color shade, according to a classification of the value of a selected TDW measure. This style is illustrated in Figure 5-7 which reports 6 screenshots respectively taken at 0-3 am, 3-6 am, 6-9 am, 12am-3pm, 3-6pm, 9-12pm of Tuesday (a working day). The images give us a qualitative view of the measure PRES: the denser is the traffic in a cell, the darker is its color. Compared to the grid of Figure 5-6, this finer level of spatial granularity highlights the road network: several road rings around the centre, and some radial streets used to enter/exit to/from the centre. During the rush hours the traffic increases in the centre of the city, as well as in the main roads. From 0am to 3am there are few cars moving around since there are no dense areas; then the outer road ring of the town becomes denser, and after, the inner rings and the radial roads.

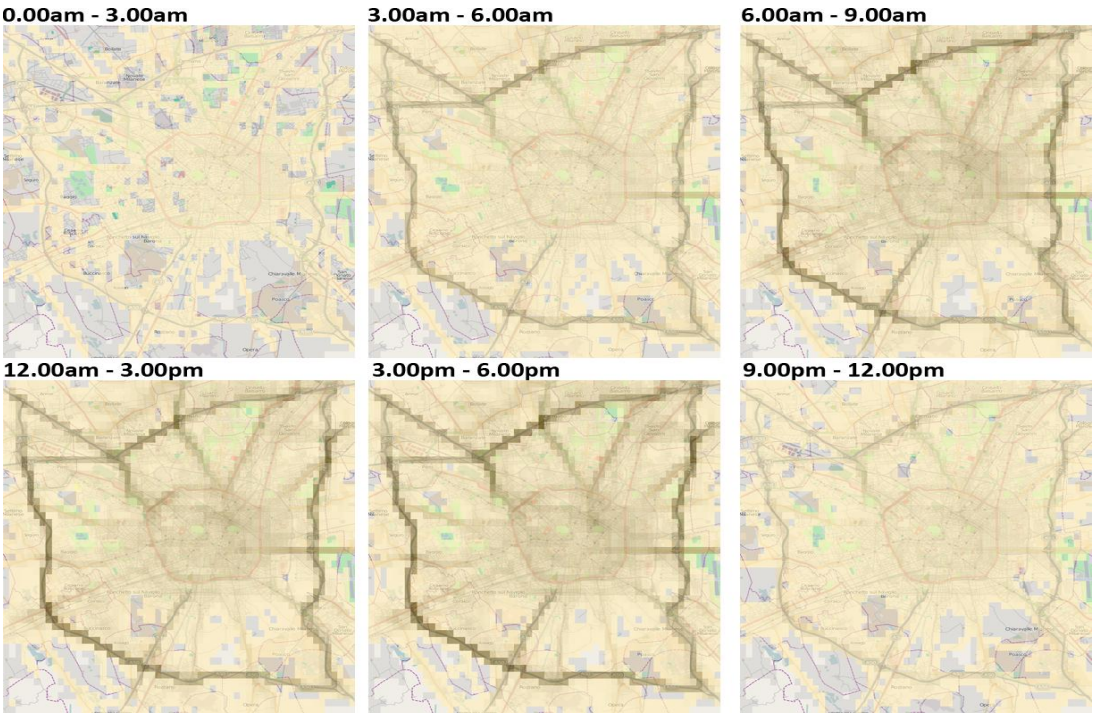


Figure 5-7: Presence on Tuesday at base granularity.

The *Line thickness* visualization style, instead, permits us to draw lines whose thickness is proportional to the value of a given TDW measure. In the screenshot of Figure 5-8 these lines are used to visualize

the *cross* measures. The measure CROSSX (crossing of X border) is represented by the horizontal lines, whereas the measure CROSSY (crossing of Y border) by the vertical lines.

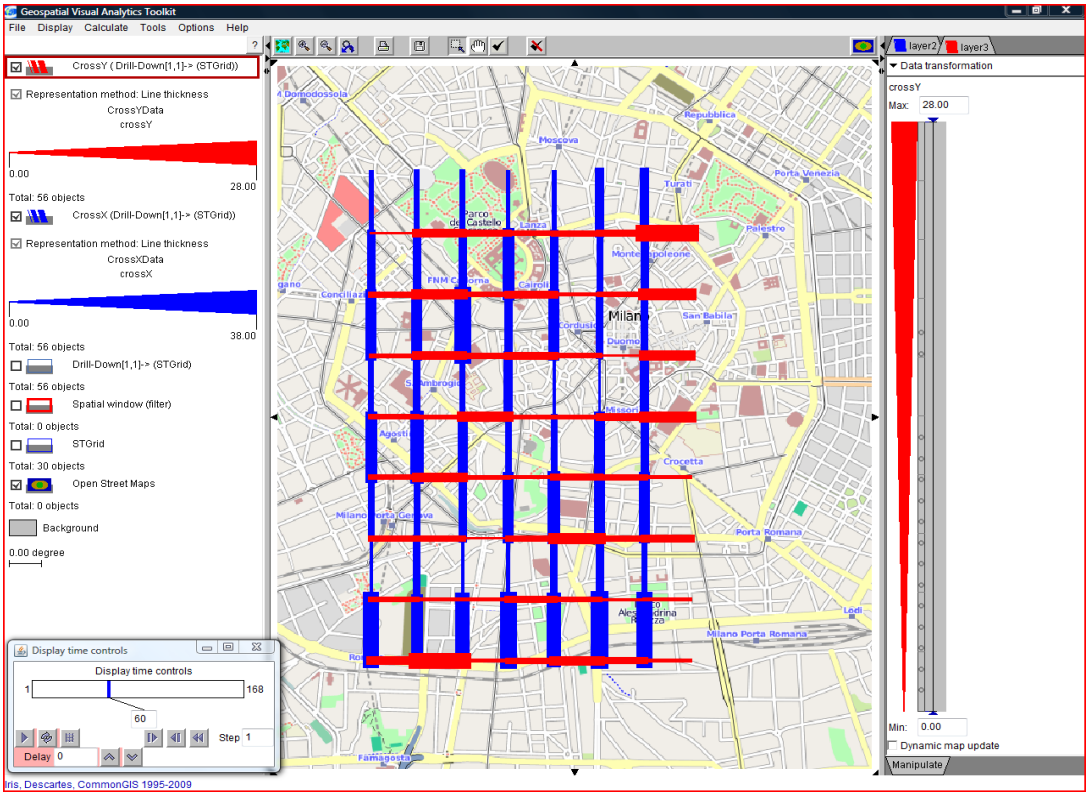


Figure 5-8: Visualization of CROSSX and CROSSY.

The described visualization methods can produce also animations, in which each frame represents the selected measure(s) in a time interval of the period of interest. In this way the user can obtain a visual representation of the variations of such measure(s) in different zones of the target space, and during different time intervals.

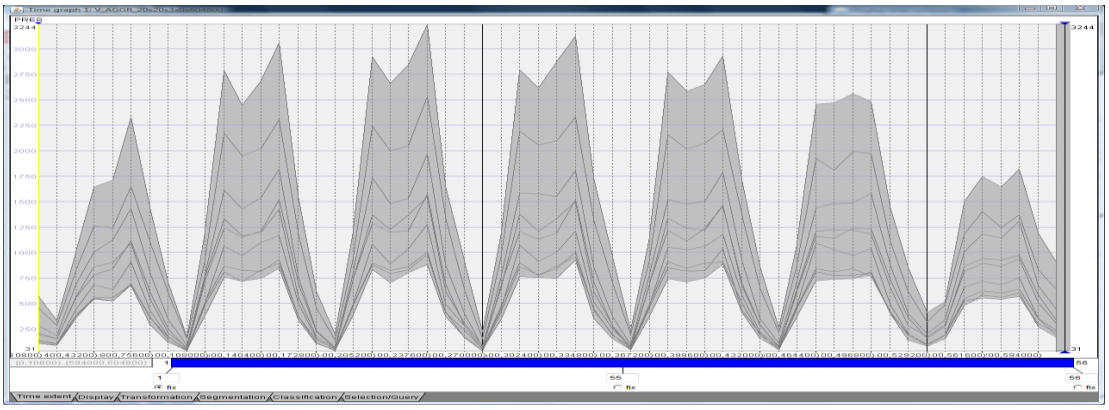


Figure 5-9: The evolution of Presence during the week.

Another type of visualization is the *Time Graph* which generates a graph showing the temporal evolution of a selected measure.

As an example, Figure 5-9 reports the time graph representing the evolution of the measure PRES,

along the week starting from Sunday up to Saturday at a granularity of 6 x 8 km^2 for the spatial dimension and of 3-hour interval for the temporal dimension. We can clearly recognize the week-days: the traffic grows during a day and decreases in the late hours of the same day. Moreover, during the week-end the presence is definitely lower than in the working days. Remark that each curve of the graph is associated with a cell of our grid and this correspondence is highlighted by clicking on the curve.

5.5. Experimental Study

In this section, we evaluate our trajectory reconstruction algorithm. As in Chapter 3, we used the e-Courier dataset [Eco09] consisting of 6.67 millions of raw location records (a file of 504 Mb, in total), that represent the movement of 84 couriers moving in greater London (covered area 66,800 km^2) during a one month period (July 2007) with a 10 sec sample rate. For all the experiments we used a PC with 1 Gb RAM and P4 3 GHz CPU.

Default values of the trajectory reconstruction parameters were set as follows: gap_{time} = 900 sec, gap_{space} = 5 Km, V_{max} = 50 m/s, $noise_{max}$ = 600 sec and D_{tol} = 20 m, and the volume of the raw locations dataset varied from 0.5 millions of records up to the full size available. This setting resulted in a maximum of 4263 trajectories (which corresponds to an average 1.64 trajectories per courier per day).

We concluded in the above values of parameters after analyzing the behavior of different candidate values and assessing the number of produced trajectories. As a first consideration, the selected value for maximum speed of vehicles is reasonable. Secondly, different values of distance accuracy parameter have not an effect on the number of produced trajectories. Below, we illustrate the results of analysis for the parameters: temporal gap (Figure 5-10), maximum noise duration (Figure 5-11) and the spatial gap parameters (Figure 5-12). As it is depicted for each parameter, the number of produced trajectories seems to stay stable from the point selected on.

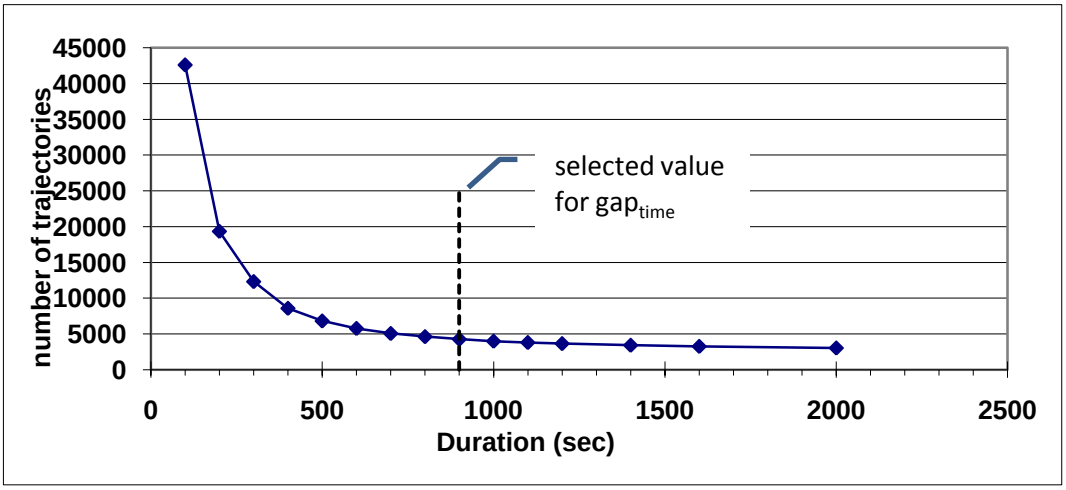


Figure 5-10: The effect of temporal gap parameter.

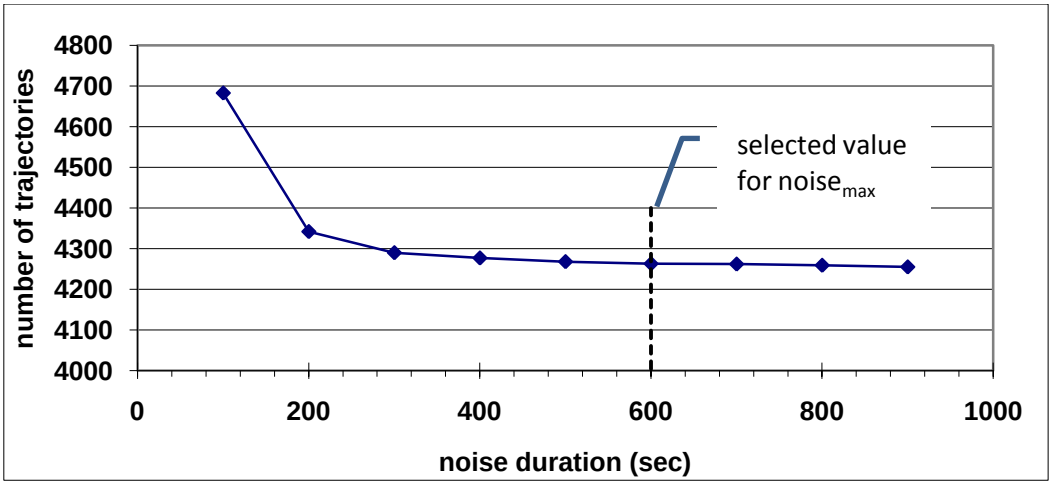


Figure 5-11: The effect of noise duration parameter.

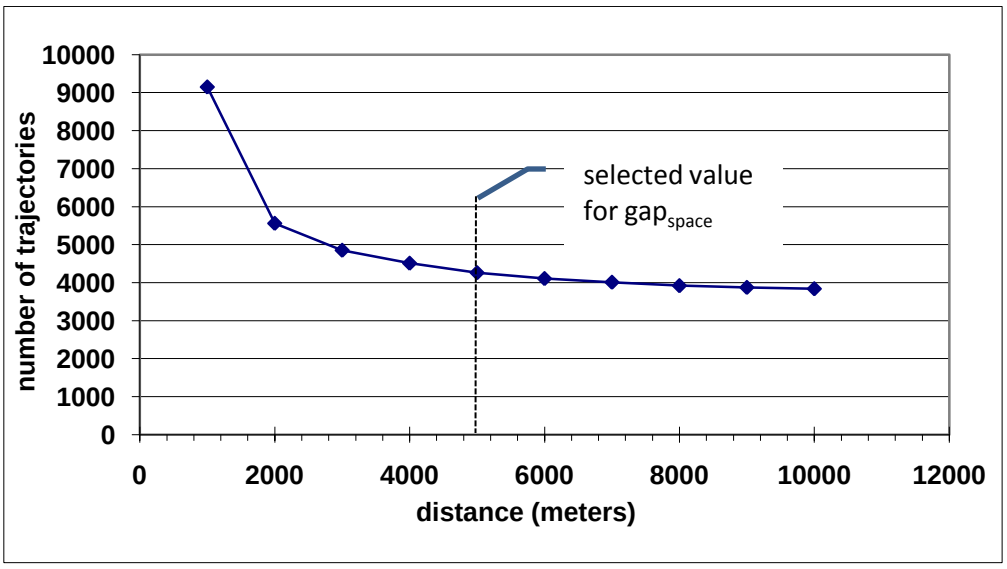


Figure 5-12: The effect of spatial gap parameter.

The next experiment, illustrated in Figure 5-13, is about the efficiency of the TRAJECTORY-RECONSTRUCTION algorithm, proposed in Subsection 5.3. It is clear that the TRAJECTORY-RECONSTRUCTION algorithm performs linear with the size of the input dataset (and allows the processing of the full dataset in about 2 min). Furthermore, the average processing rate is almost stable (~ 50K records/sec).

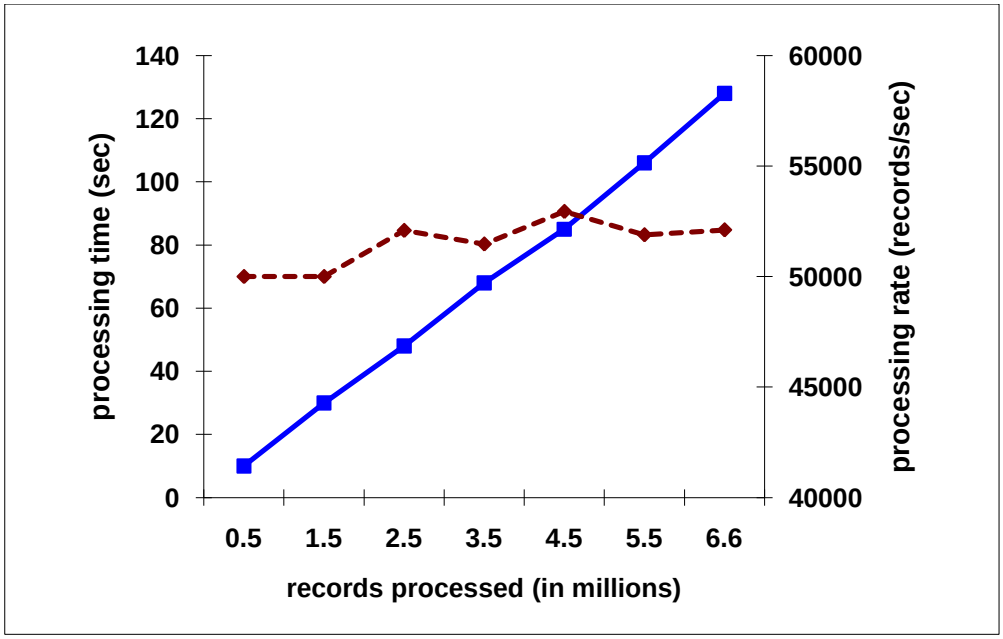


Figure 5-13: Performance of trajectory reconstruction
(solid line: processing time; dotted line: processing rate).

In the next set of experiments, we evaluate the effectiveness of the TRAJECTORY-RECONSTRUCTION algorithm in the case of a large number of users and towards the goal of processing input in real-time. In particular, we measure its processing time for various sample rates, as illustrated in Figure 5-14. According to this experiment, by choosing e.g. a 20 sec sample rate the algorithm can handle in real time up to 1000K objects, which is a really large number for real-world applications (at least nowadays). The conclusion from this experiment is that the proposed TRAJECTORY-RECONSTRUCTION algorithm is effective for real-time processing by keeping a trade-off between the number of users to be supported and the sample rate set for transmitting their location.

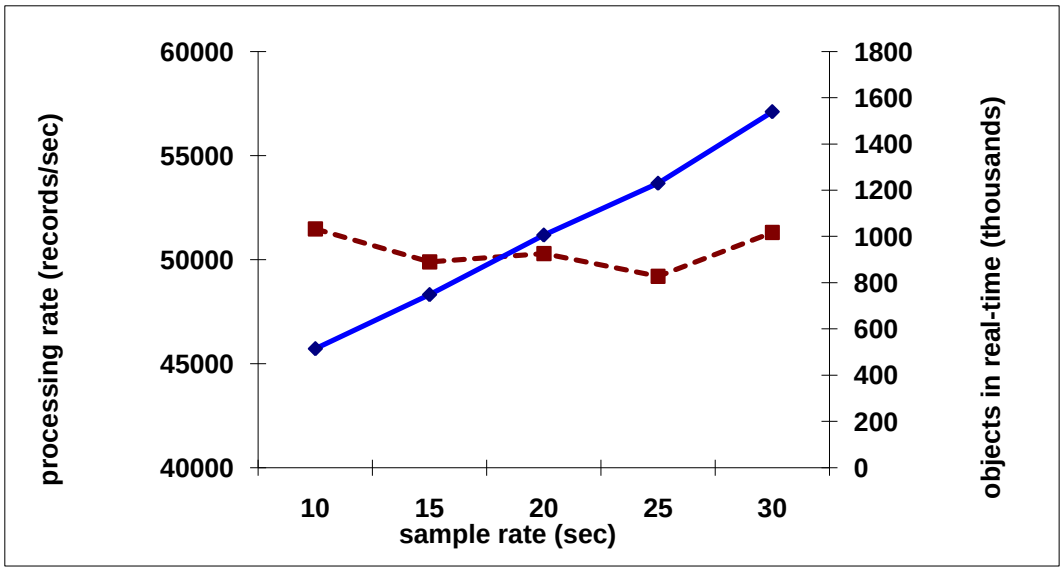


Figure 5-14: The effect of sample rate in real-time processing (solid line: objects in real-time; dotted line: processing rate).

5.6. Synopsis

In this chapter, we presented T-WAREHOUSE, a system for Visual Trajectory Data Warehousing, which incorporates all the necessary steps, from trajectory reconstruction and ETL processing to Visual OLAP analysis. Particularly for the former step, we proposed a *trajectory reconstruction* technique that transforms sequences of raw location points into trajectories. We describe the architectural issues of our framework and examined the power, flexibility and efficiency of our framework for applying OLAP analysis on real world mobility data.

6. Epilogue

6.1. Conclusions

This thesis discussed various works for efficient Mobility Data Warehousing & Mining techniques on Moving Object Databases. Our aim was to contribute to the realization of the *Geographic Privacy-aware KDD process* as a sequence of individual steps: from *trajectory reconstruction* to OLAP analysis and patterns mining. In the remaining paragraphs we outline the main contributions of this thesis.

In Chapter 2, we clarified the difference among the two categories of spatiotemporal data: immobility and mobility data. We discussed about seismological data, as an example of the former category, and we proposed a *Seismic Data Management and Mining System* that supports all the required steps: from data collection to patterns mining. As for the mobility data, we presented some basic concepts about trajectory data and set the specification of a Mobility Data Warehousing and Mining framework.

In Chapter 3, we proposed solutions for the efficient and effective development of trajectory data warehouses. More specifically, we proposed techniques for supporting *ETL* of trajectory data in order to feed the trajectory data cubes, and for addressing the problem of measure aggregation, focusing particularly to the *distinct count problem*. As for the ETL procedure, we presented two alternative methods: a (index-based) *cell-oriented* and a (non-index-based) *trajectory-oriented*. As for the *distinct count problem*, we proposed to add some auxiliary measures in the data cube model that will help us to correct the errors caused due to the duplicates when rolling-up. Our approaches have been experimentally tested in a large real dataset and have been shown to be efficient. As we will illustrated during the experimental study, the choice of a particular ETL method is a trade-off between the selected granularity level and the number of trajectories. Regarding our approximate solution for the distinct *count problem*, it turns out to perform effectively comparing to a distributive aggregate function.

Moreover, in Chapter 3, we proposed an innovative way to organize a trajectory data cube that considers different interpretations of the notion of trajectory and therefore can serve a number of applications. More specifically, we extended the OLAP data model so as to encapsulate the necessary flexibility regarding different semantic definitions of trajectories, we proposed appropriate ETL and OLAP mechanisms that utilize the new model and we discussed materialization issues. We used the same dataset with the one we tested our Trajectory Data Warehousing framework and therefore we

provided comparisons between the two methods. From this experimental study, it is clear that our ad-hoc approach is useful in cases where a trajectory data cube has to serve a number of applications.

Chapter 4 illustrated our framework for mining interaction patterns that enables spatiotemporal representation, synthesis and classification allowing an adequate understanding of what is happening to moving objects with regards to the context (i.e. the geographical space, the interaction with other objects etc) where they move. Our framework supports the computation of various interaction descriptors considering either a dynamic neighborhood or a fixed grid approach. These descriptors provide insights on the movement trends in various spatiotemporal windows giving insights on the interactions between objects. Preliminary experimental results showed the applicability and efficiency of our approach and include comparisons between the two methods (dynamic and static), precision analysis using different descriptors etc.

Additionally, in Chapter 4, we discussed the problem of traffic management on a road network considering a traffic time series for each edge of the network. We defined different traffic relationships to capture the way that traffic is circulated within the network and proposed a clustering based approach in order to capture these relationships. Our approach provides a hierarchy of groups of similar edges, starting from groups of edges with similar traffic shape, proceeding to groups of nearby edges and finally, resulting into groups of edges with similar traffic values. The data that are required for this analysis can be collected either using sensing devices that monitor the traffic of each edge or by aggregating positioning data into traffic time series. Although it seems that this framework does not consider trajectories and requires a different infrastructure, please note that the TDW architecture thoroughly presented in Chapter 3 can be utilized as the repository of traffic datasets. The city network can be modeled as a spatial dimension, the temporal periods as the time dimensions and the measures can store aggregate data for analyzing traffic data. The experimental results showed that our method can provide useful insights regarding the traffic flow on a road network, e.g., which edges share similar traffic shapes and how these group of edges are modulated when the network proximity or/and the value based similarity is also considered.

Finally, Chapter 5 demonstrated T-WAREHOUSE, a system that integrates all the required steps for Visual TDW, from *trajectory reconstruction* and ETL processing to Visual OLAP analysis on mobility data. Chapter 5 illustrates the architectural aspects of our framework and investigates its power, flexibility and efficiency for applying OLAP analysis on real world mobility data. We also presented an efficient algorithm for transforming sequences of raw positioning points into trajectories and tested ts performance using a large real dataset. Our experimental study proved that our framework is effective even for real-time processing.

6.2. Open Issues

Several research fields are remained open in the field of Mobility Data Warehousing & Mining. In the next paragraphs we describe the future research work directly fountain by the advances of this thesis.

Regarding Trajectory Data Warehousing, a line of research includes the examination of new measures for the trajectory warehouse, specifically suited for trajectories. An example of such a measure is the

so-called *typical trajectory* (e.g. [GNP+07], [LHW07]) that describes the trend of movement within a cell. This is a rather challenging problem as it is not straightforward to derive the representative trajectory of a cuboid based on the representative trajectories of its sub-cuboids. The representative trajectory of a cuboid can be considered as a buffer-trajectory representing the pattern of movement within the cell. Let $D = \{T_1, T_2, \ldots, T_n\}$ be a set of trajectories, all defined between time points t_1 and t_m. Let τ_s be the *tolerance* in the *spatial* domain. At a specific time point t_i, two trajectories T_1, T_2 are τ_s tolerant if their corresponding space coordinates differ less than τ_s. If this stands for all time points t_1 and t_m thenT_1, T_2 are included in the same trajectory buffer. If this stands for every pair of trajectories in D, we state that D trajectories form a trajectory buffer *TB.* The number of trajectories included in the buffer is called the *support* of the buffer. The discovery of such representative trajectories as a twofold problem: discovering representative trajectories from raw data or other representative trajectories. Actually, the first one is a clustering problem whereas the second one is an aggregation problem in the TDW context because the representative trajectory of a cuboid should be calculated from its lower level cuboids (in order to avoid execute again the clustering task for the aggregate cuboid).

Other interesting measures could be investigated like the *average direction* measure of the trajectories within a cell. Also, materialization issues for Trajectory Data Warehouses have to be considered. More specifically, to develop techniques so as to decide which cuboids should be materialized and how these can be used so as to answer OLAP queries. In traditional DWs, a common approach is to use the smallest parent so as to answer a query. However, in the case of TDW maybe it's wiser to select a cuboid which reduces the error rate (due to the distinct count problem).

As for the proposed ad-hoc TDW approach, new measures can be examined, like cross cell measures that will exploit the proposed trajectory warehouse model. More specifically, it is challenging to study holistic measures like the number of distinct trajectories in a set of cells. In this case we need a technique that allows the identification of trajectories that evolves in consecutive base cells. As a second line of research on this topic, sophisticated methods for materializing the data cube can be developed. Such techniques may scan a dataset so as to suggest some possible maximum temporal distances based on which the materialization is performed.

Regarding the interaction mining approach, the interaction descriptors that are computed by our framework can be used to develop OLAP operations. Instead of applying e.g. a roll-up operation in order to aggregate a subset of cells of a data cube, maybe it is interesting to aggregate only the spatiotemporal cells that contain specific interaction patterns (i.e. "dangerous driving behavior") extracted from our classification method. In this case, a sophisticated computational strategy that will speed up the aggregation phase is needed.

As for the traffic patterns mining approach, our approach can be enhanced by discovering more complex relationships between graph edges. So far we considered propagation, split and merge of traffic. However, more complex cases might appear, e.g., combinations of these primitive relationships like a traffic propagation accompanied by a traffic split and then by a traffic merge, that results after that to the initial traffic series. Moreover, one more interesting extension is the continuous monitoring of the traffic network. So far, we analyze the network traffic for a specific period of interest. However,

traffic data are dynamic by their nature, so it is also important to online monitor traffic flow in order to detect abnormalities with respect to the expected behavior (where expected comes from the historical analysis of traffic).

Regarding the topic of *trajectory* reconstruction, future work includes the exploration of intelligent ways to automatically extract proper values of trajectory reconstruction parameters according to a number of characteristics of datasets as well as the extension of this technique so as to be able to identify different movement types (pedestrian, bicycle, motorbike, car, truck etc) and hence to apply customized trajectory reconstruction.

7. References

[ADH+03] van der Aalst, W. M., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., and Weijters, A. J. Workflow mining: a survey of issues and approaches. *DKE* 47(2):237-267, 2003.

[AAD+96] Agarwal, S., Agrawal, R., Deshpande, P., Gupta, A., Naughton, J., Ramakrishnan, R., and Sarawagi. S. On the computation of multidimensional aggregates. *Proceedings of VLDB*, 1996.

[AIS93] Agrawal, R., Imielinski, T., and Swami, A. Mining Association Rules between Sets of Items in Large Databases. *Proceedings of ACM SIGMOD*, 2003.

[AGB06] Almeida, V. T., Güting, R. H., and Behr, T. Querying moving objects in secondo. *Proceedings of MDM*, 2006.

[AA99] Andrienko, G., Andrienko N., Knowledge-based visualization to support spatial data mining. *Proceedings of IDA*, 1999.

[AAW07] Andrienko, G., Andrienko N., and Wrobel, S. Visual Analytics Tools for Analysis of Movement Data. *ACM SIGKDD Explorations* 9(2): 28-46, 2007.

[BMR+08] Baglioni, M., Macedo, J., Renso, C., and Wachowicz, M. An Ontology-Based Approach for the Semantic Modeling and Reasoning on Trajectories. *Proceedings ER Workshops*, 2008.

[BMH01] Bédard, Y., Merrett, T., and Han, J. Fundamentals of Spatial Data Warehousing for Geographic Knowledge Discovery. *Geographic Data Mining and Knowledge Discovery*, pp. 53-73. Taylor & Francis, 2001.

[BD00] Behnke, J., Dobinson, E., NASA Workshop on Issues in the Application of Data Mining to Scientific Data. *ACM SIGKDD Explorations* 2(1):70-79, 2000.

[BTM05] Bimonte, S., Tchounikine, A., and Miquel, M. Towards a spatial multidimensional model. *Proceedings* of *DOLAP*, 2005.

[BOO+07] Braz, F., Orlando, S., Orsini, R., Raffaetta, A., Roncato, A., and Silvestri, C. Approximate Aggregations in Trajectory Data Warehouses. *Proceedings of ICDE Workshop on Spatio-Temporal Data Mining*, 2007.

[Bri02] Brinkhoff, T. A Framework for Generating Network-Based Moving Objects. *GeoInformatica* 6(2):153-180, 2002.

[CCD+04] Castellanos, M., Casati, F., Dayal, U., Shan, M.C. A Comprehensive and Automated Approach to Intelligent Business Processes Execution Analysis. *Int. J. Distributed and Parallel Databases* 16:239-273, 2004.

[CD97] Chaudhuri, S., and Dayal, U. An overview of Data Warehousing and OLAP Technology. *SIGMOD Record* 26(1):65-74, 1997.

[Cia09] CIA. The World Factbook. 2009

[CYH+07] Cheng, H., Yan, X., Han, J. and Hsu, C.W. Discriminative Frequent Pattern Analysis for Effective Classification. *Proceedings of ICDE*, 2007.

[CKL06] Choi, W., Kwon, D., and Lee, S. Spatio-temporal data warehouses using an adaptive cell-based approach. *DKE* 59(1):189-207, 2006.

[DG03] Das, G. and Gunopulos, D. Time series similarity and indexing. *Chapter in Ye, N.(Ed.): The Handbook of Data Mining*, pp. 279–302, Lawrence Erlbaum Associates, 2003.

[DS06] Damiani, M.-L. and Spaccapietra, S. Spatial Data Warehouse Modelling. *Chapter in Processing and Managing Complex Data for Decision Support*, pp. 12-27, Idea Group Publishing, 2006.

[DKW+05] Deshpande, M., Kuramochi, M., Wale, N. and Karypis, G. Frequent Substructure-Based Approaches for Classifying Chemical Compounds. *TKDE* 17(8):1036-105, 2005.

[Doc06] Dockstader, S.L. Motion Trajectory Classification for Visual Surveillance and Tracking. *Proceedings of AVSS*, 2006.

[Eco09] eCourier.co.uk dataset, http://api.ecourier.co.uk/. (URL valid on December 14, 2009).

[FPS+96] Fayad, U., Piatetsky-Shapiro, G., Smith, P., and Uthurusami, R. *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996.

[FM85] Flajolet, P. and Martin, G. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.* 31(2):182-209, 1985.

[Geo06] GeoPKDD project, http://www.geopkdd.eu/. (URL valid on December 14, 2009).

[Geu01] Geurts, P. Pattern Extraction for Time Series Classification. *Proceedings of PKDD*, 2001.

[GNP+07] Giannotti, F., Nanni, M., Pinelli, F., and Pedreschi, D. Trajectory pattern mining. *Proceedings of KDD*, 2007.

[GP07] Giannotti, F., and Pedreschi, D. (Eds). Mobility, Data Mining and Privacy. Springer, 2007.

[GPT08] Giannotti, F., Pedreschi, D., and Turini, F. Mobility, Data Mining and Privacy the Experience of the GeoPKDD Project. *ACM SIGKDD PinKDD*, 2008.

[GKM+09] Gómez, L., Kuijpers, B., Moelans, B., Vaisman, A. A Survey of Spatio-Temporal Data Warehousing. *IJDWM* 5(3):28-55, 2009.

[GCB+97] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. Data cube: A relational aggregation operator generalizing groub-by, cross-tab and sub-totals. *DMKD* 1(1):29-54, 1997.

[GBE+00] Güting, R. H., Böhlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., and Vazirgiannis, M. A foundation for representing and quering moving objects. *ACM Transactions on Database System* 25(1):1-42, 2000.

[GS05] Güting, R.H., and Schneider, M. Moving Object Databases, Morgan Kaufman Publishers. 2005.

[HK00] Han, J., and Kamber, M. Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000.

[HSK98] Han, J., Stefanovic, N., and Koperski, K. Selective Materialization: An Efficient Method for Spatial Data Cube Construction. *Proceedings of PAKDD*, 1998.

[Inm96] Inmon, W. Building the Data Warehouse. John Wiley & Sons, 1996.

[JMF99] Jain, A., Murty, M., and Flynn, P. Data Clustering: A Review. *ACM Computing Surveys* 31(3):264–323, 1999.

[JKP+04] Jensen, C.S., Kligys, A., Pedersen, T.B., Dyreson, C.E., and Timko, I. Multidimensional data modeling for location-based services. *The VLDB Journal* 13(1):1–21, 2004.

[KMB05] Kalnis, P., Mamoulis, N. and Bakiras, S. On discovering moving clusters in spatio-temporal data. *Proceedings of SSTD*, 2005.

[KR90] Kaufman, L., and Rousseeuw, P.J. Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 1990.

[KP98] Keogh, E. and Pazzani, M. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. *Proceedings of KDD*, 1998.

[Kim96] Kimball, R. The Data Warehouse Toolkit. John Wiley & Sons, 1996.

[Kot02] Kotidis, Y. Aggregate View Management in Data Warehouses. *Chapter in Handbook of Massive Data Sets*, pp. 711-742, Kluwer Academic Publishers, 2002.

[Kot06] Kotidis, Y. Extending the data warehouse for service provisioning data. *DKE* 59(3):700-724, 2006.

[KR00] Kretschmer, U., Roccatagliata, E., CommonGIS: A European Project for an Easy Access to Geo-data. *Proceedings of EUGISES*, 2000.

[KNK+08] Kuijpers, B., Nanni, M., Korner, C., May, M. and Pedreschi, D. Spatiotemporal data mining. *Chapter in Giannotti, F. and Pedreschi, D. (Eds.): Mobility, Data Mining and Privacy: Geographic Knowledge Discovery*, pp.275–300, Springer, 2008.

[LHL+08] Lee, J., Han, J., Li, X., and Gonzalez, H. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *VLDB Endowment* 1(1), pp. 1081-1094, 2008.

[LHW07] Lee, J., Han, J., and Whang, K. Trajectory Clustering: A Partition-and-Group Framework. *Proceedings of ACM SIGMOD*, 2007.

[LMF+10] Leonardi, L., Marketos, G., Frentzos, E., Giatrakos, N., Orlando, S., Pelekis, N., Raffaetà, A., Roncato, A., Silvestri, C., Theodoridis, Y. T-Warehouse: Visual OLAP Analysis on Trajectory Data. *Proceedings of ICDE*, 2010.

[LOR+09] Leonardi, L., Orlando, R., Raffaetà, A., Roncato, A., and Silvestri, C Frequent spatio-temporal patterns in trajectory data warehouses. *Proceedings of SAC*, 2009.

[LHK06] Li, X., Han, J. and Kim, S. Motion-alert: automatic anomaly detection in massive moving objects. *Proceedings of IEEE International Conference on Intelligence and Security Informatics*, 2006.

[LHL+07] Li, X., Han, J., Lee, J-G. and Gonzalez, H. Traffic density-based discovery of hot routes in road networks. *Proceedings of SSTD*, 2007.

[LCZ+06] Liu, Y., Choudhary, A.N., Zhou, J. and Khokhar, A.A. A scalable distributed stream mining system for highway traffic data. *Proceedings of ECML/PKDD*, 2006.

[LT05] Lopez, I., Snodgrass, R., and Moon, B. Spatiotemporal Aggregate Computation: A Survey. *TKDE* 2(17):271-286, 2005.

[MVO+08] Macedo, J., Vangenot, C., Othman, W., Pelekis, N., Frentzos, E., Kuijpers, B., Ntoutsi, I., Spaccapietra, S. and Theodoridis, Y. Trajectory data models. *Chapter in F. Giannotti and D. Pedreschi (eds), Mobility, Data Mining and Privacy: Geographic Knowledge Discovery*. Springer, 2008.

[MZ04a] Malinowski, E. and Zimányi, E. OLAP Hierarchies: A Conceptual Perspective. *Proceedings of CAiSE*, 2004.

[MZ04b] Malinowski, E. and Zimányi, E. Representing Spatiality in a Conceptual Multi-dimensional Model. *Proceedings of ACM-GIS*, 2004.

[MZ06] Malinowski, E. and Zimányi, E. Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *DKE* 59(2):348-377, 2006.

[MFN+08a] Marketos, G., Frentzos, E., Ntoutsi, I., Pelekis, N., Raffaeta, A., and Theodoridis, Y., Building Real-World Trajectory Warehouses. *Proceedings of MobiDE*, 2008.

[MFN+08b] Marketos, G., Frentzos, E., Giatrakos, N., Ntoutsi, I., Pelekis, N., Raffaeta, A., and Theodoridis, Y., A Framework for Trajectory Data Warehousing. *Proceedings of HDMS*, 2008.

[MT09a] Marketos, G., Theodoridis, Y., Ad-hoc OLAP on Trajectory Data. *Submitted*.

[MT06] Marketos, G., Theodoridis, Y., Measuring Performance in the retail industry. *Proceedings of BPI*, 2006.

[MT09b] Marketos, G., Theodoridis, Y., Mobility Data Warehousing & Mining. *Proceedings of VLDB PhD Workshop*, 2009.

[MTK08] Marketos, G., Theodoridis, Y., and Kalogeras, I., Seismological Data Warehousing and Mining – A Survey. *IJDWM* 4(1):1-16, 2008.

[NT04] Nakata, T. and Takeuchi, J. Mining traffic data from probe-car system for travel time prediction. *Proceedings of ACM SIGKDD*, 2004.

[NMO09] Nanni, M., Marketos, G., Quattrociocchi. W. Mining Interaction Patterns for Spatiotemporal Representation, Synthesis and Classification. *Manuscript prepared*.

[NMM08] Ntoutsi, I., Mitsou, N., Marketos, G., Traffic mining in a road-network: How does the traffic flow?. *IJBIDM* 3(1), 2008.

[Ope01] OpenGIS Consortium. Abstract Specification, Topic 1: Feature Geometry (ISO 19107 Spatial Schema), 2001. http://www.opengeospatial.org. (URL valid on December 14, 2009).

[OOR+07] Orlando, S., Orsini, R., Raffaetà, A., Roncato, A., and Silvestri, C. Trajectory Data Warehouses: Design and Implementation Issues. *JCSE* 1(2):240-261, 2007.

[PKZ+01] Papadias, D., Kalnis, P., Zhang, J., and Tao, Y. Efficient OLAP Operations in Spatial Data Warehouses. *Proceedings of SSTD*, 2001.

[PTK+02] Papadias, D., Tao, Y., Kalnis, P., and Zhang, J. Indexing Spatio-Temporal Data Warehouses. *Proceedings of ICDE*, 2002.

[PT01] Pedersen, T. and Tryfona, N. Pre-aggregation in Spatial Data Warehouses. *Proceedings of SSTD*, 2001.

[PFG+08] Pelekis, N., Frentzos, E., Giatrakos, N. and Theodoridis, Y. HERMES: Aggregative LBS via a Trajectory DB Engine. *Proceedings of ACM SIGMOD*, 2008.

[PKM+07] Pelekis, N., Kopanakis, I., Marketos, G., Ntoutsi, I., Andrienko, G., and Theodoridis, Y., Similarity Search in Trajectory Databases. *Proceedings of TIME*, 2007

[PRD+08] Pelekis, N., Raffaetà, A., Damiani, M.-L., Vangenot, C., Marketos, G., Frentzos, E., Ntoutsi, I., and Theodoridis, Y. Towards Trajectory Data Warehouses. *Chapter in Mobility, Data Mining and Privacy: Geographic Knowledge Discovery*. Springer-Verlag. 2008.

[PTV+06] Pelekis, N., Theodoridis, Y., Vosinakis, S. and Panayiotopoulos, T. Hermes - A Framework for Location-Based Data Management. *Proceedings of EDBT*, 2006.

[PJT00] Pfoser, D., Jensen, C.S., and Theodoridis, Y. Novel Approaches to the Indexing of Moving Object Trajectories, *Proceedings of VLDB*, 2000.

[PT98] Pfoser, D., and Tryfona, N. Requirements, Definitions and Notations for Spatiotemporal Application Environments. *Proceedings of ACM-GIS*, 1998.

[RZY+03] Rao, F., Zhang, L., Yu, X., Li, Y., and Chen, Y. Spatial Hierarchy and OLAP-Favored Search in Spatial Data Warehouse. *Proceedings of DOLAP*, 2003.

[RBM01] Rivest, S., Bédard, Y., and Marchand, P. Towards Better Support for Spatial Decision Making: Defining the Characteristics of Spatial On-Line Analytical processing (SOLAP). *Geoinformatica* 55(4):539-555, 2001.

[RBP+05] Rivest, S., Bédard, Y., Proulx, M., Nadeau, M., Hubert, F., and Pastor, J. SOLAP: Merging Business Intelligence with Geospatial Technology for Interactive Spatio-Temporal Exploration and Analysis of Data. *Journal of International Society for Photogrammetry & Remote Sensing* 60(1):17-33, 2005.

[Riz03] Rizzi, S. Open problems in data warehousing: eight years later. *Proceedings of Design and Management of Data Warehouses*, 2003.

[RG00] Rizzi, S.and Golfarelli, M. Data warehouse design. *Proceedings of Enterprise Information Systems*, 2000.

[SC78] Sakoe, H., and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing* 26(1):43–49, 1978.

[SLC+01] Shekhar, S., Lu, C-T., Chawla, S. and Zhang, P. Data Mining and Visualization of Twin-cities Traffic Data, *Technical Report* (TR 01-015), 2001.

[SPD+08] Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., and Vangenot, C. A conceptual view on trajectories. *DKE* 65(1):126-146, 2008.

[SNT+06] Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y. and Schult, R. Monic: modeling and monitoring cluster transitions. *Proceedings of ACM SIGKDD*, 2006.

[SHK00] Stefanovic, N., Han, J., and Koperski, K. Object-based selective materialization for efficient implementation of spatial data cubes. *TKDE* 12(6):938-958, 2000.

[TKC+04] Tao, Y., Kollios, G., Considine, J., Li, F., and Papadias, D. Spatio-Temporal Aggregation Using Sketches. *Proceedings of ICDE*, 2004.

[TP05] Tao, T., and Papadias, D. Historical Spatio-Temporal Aggregation. *Proceedings of ACM TODS* 23(1):61-102, 2005.

[The03] Theodoridis, Y. Seismo-Surfer: A Prototype for Collecting, Querying and Mining Seismic Data. *Proceedings of PCI*, 2003.

[TPG+01] Trujillo, J., Palomar, M., Gomez, J. and Song, I. Designing Data Warehouses with OO Conceptual Models. *IEEE Computer* 34(12):66-75, 2001.

[VS99] Vassiliadis, P. and Sellis, T. A survey of logical models for OLAP databases. *SIGMOD Record* 28(4):64-69, 1999.

[VWI98] Vitter, J.S., Wang, M., and Iyer, B. Data Cube Approximation and Histograms via Wavelets. *Proceedings of CIKM*, 1998.

[Yu05] Yu, B., Mining Earth Science Data for Geophysical Structure: A Case Study in Cloud Detection. *Proceedings of SIAM*, 2005.

[ZT05] Zhang D. Tsotras, V. Optimizing spatial Min/Max aggregations. *The VLDB Journal* 14(2):170-181, 2005.