Spatio–Temporal Data Mining for Location–Based Services

Győző Gidófalvi

Ph.D. Thesis

A thesis submitted to the Faculties of Engineering, Science and Medicine at Aalborg University, Denmark, in partial fulfillment of the requirements for the Ph.D. degree in computer science.

Abstract

Largely driven by advances in communication and information technology, such as the increasing availability and accuracy of GPS technology and the miniaturization of wireless communication devices, Location–Based Services (LBS) are continuously gaining popularity. Innovative LBSes integrate knowledge about the users into the service. Such knowledge can be derived by analyzing the location data of users. Such data contain two unique dimensions, *space* and *time*, which need to be analyzed.

The objectives of this thesis are three–fold. First, to extend popular data mining methods to the spatio–temporal domain. Second, to demonstrate the usefulness of the extended methods and the derived knowledge in two promising LBS examples. Finally, to eliminate privacy concerns in connection with spatio–temporal data mining by devising systems for privacy–preserving location data collection and mining.

To this extent, Chapter 2 presents a general methodology, *pivoting*, to extend a popular data mining method, namely rule mining, to the spatio–temporal domain. By considering the characteristics of a number of real–world data sources, Chapter 2 also derives a taxonomy of spatio–temporal data, and demonstrates the usefulness of the rules that the extended spatio–temporal rule mining method can discover. In Chapter 4 the proposed spatio–temporal extension is applied to find long, sharable patterns in trajectories of moving objects. Empirical evaluations show that the extended method and its variants, using high–level SQL implementations, are effective tools for analyzing trajectories of moving objects.

Real–world trajectory data about a large population of objects moving over extended periods within a limited geographical space is difficult to obtain. To aid the development in spatio–temporal data management and data mining, Chapter 3 develops a Spatio–Temporal ACTivity Simulator (ST–ACTS). ST–ACTS uses a number of real–world geo–statistical data sources and intuitive principles to effectively generate realistic spatio–temporal activities of mobile users.

Chapter 5 proposes an LBS in the transportation domain, namely cab–sharing. To deliver an effective service, a unique spatio–temporal grouping algorithm is presented and implemented as a sequence of SQL statements. Chapter 6 identifies a scalability bottleneck in the grouping algorithm. To eliminate the bottleneck, the chapter expresses the grouping algorithm as a continuous stream query in a data stream management system, and then devises simple but effective spatio–temporal partitioning methods for streams to parallelize the computation. Experimental results show that parallelization through adaptive partitioning methods leads to speed–ups of orders of magnitude without significantly effecting the quality of the grouping. Spatio–temporal stream partitioning is expected to be an effective method to scale computation–intensive spatial queries and spatial analysis methods for streams.

Location–Based Advertising (LBA), the delivery of *relevant* commercial information to mobile consumers, is considered to be one of the most promising business opportunities amongst LBSes. To this extent, Chapter 7 describes an LBA framework and an LBA database that can be used for the management of mobile ads. Using a simulated but realistic mobile consumer population and a set of mobile ads, the LBA database is used to estimate the capacity of the mobile advertising channel. The estimates show that the channel capacity is extremely large, which is evidence for a strong business case, but it also necessitates adequate user controls.

When data about users is collected and analyzed, privacy naturally becomes a concern. To eliminate the concerns, Chapter 8 first presents a grid–based framework in which location data is anonymized through spatio–temporal generalization, and then proposes a system for collecting and mining anonymous location data. Experimental results show that the privacy–preserving data mining component discovers patterns that, while probabilistic, are accurate enough to be useful for many LBSes.

To eliminate any uncertainty in the mining results, Chapter 9 proposes a system for collecting *exact* trajectories of moving objects in a privacy–preserving manner. In the proposed system there are no trusted components and anonymization is performed by the clients in a P2P network via data cloaking and data swapping. Realistic simulations show that under reasonable conditions and privacy/anonymity settings the proposed system is effective.

Acknowledgments

This thesis is the result of an industrial Ph.D. study, which is a unique mix of theory and practice. This study would not have been possible without the support of the following organizations: Geomatic ApS, Aalborg University, in particular the Department of Computer Science, and the Danish Ministry of Science, Technology, and Innovation, which has supported the study under grant number 61480. I would like to thank these organizations for the unforgettable life experience and the priceless gift of a world–class education.

I believe that motivation plays a very significant role in a person's education. Among the numerous people that provided me motivation, I would like to gratefully acknowledge the help of Professors Sara Baase, Mahmoud Tarokh, Roman Swiniarski, and Charles Elkan. Most importantly, I would like to acknowledge my Ph.D. advisor, Torben Bach Pedersen, for all his motivation and insights throughout my graduate career. Apart from teaching me the principles of research and academic writing, he has also introduced me to several new and exciting topics in the area of spatio–temporal data management and analysis. Similarly, I would like to acknowledge Professor Tore Risch for filling a similar role during my stay–abroad study at Uppsala University.

Furthermore, I would like to thank all my colleges in Geomatic, the members of the Database and Programming Technologies Group at Aalborg University, and the members of the Uppsala Database Laboratory at Uppsala University for their help and for providing a friendly and stimulating environment. In particular, I would like to thank Martin K. Glarvig, Hans Ravnkjær Larsen, Lau Kingo Marcussen, Jesper Persson, Esben Taudorf, Kasper Rügge, Susanne Caroc and Jesper Christiansen at Geomatic, Laurynas Speicys, Nerius Tradisauskas, Xuegang Huang and Christian Thomsen in the Database and Programming Technologies Group at Aalborg University, and Erik Zeitler in the Uppsala Database Laboratory at Uppsala University.

Last, but not least, I would like to express my appreciation to my parents, Ágnes and Zoltán, and my brother Gergely. I would not have been able to do it without their continuous support and motivation.

Contents

Chapter 1

Introduction

Recent advances in communication and information technology, such as the increasing accuracy of GPS technology and the miniaturization of wireless communication devices pave the road for Location–Based Services (LBS). To achieve high quality for such services, data mining techniques are used for the analysis of the huge amount of data collected from location–aware mobile devices. The objectives of this thesis are three–fold. First, since the two most important attributes of the data collected are *location* and *time*, the thesis investigates general methods for extending existing data mining methods to the spatio–temporal domain. Second, using two promising LBSes, the thesis demonstrates the usefulness of the knowledge that can be extracted by the spatio–temporal data mining methods. Finally, since privacy is a major concern to users of LBSes, the thesis considers location privacy in connection with collection and data mining of location traces (trajectories) of users.

In the thesis the following setting and broad definitions are assumed. Mobile users carry location–enabled mobile terminals (PDA, mobile phone, etc). By location–enabled it is meant that applications running on the mobile terminal have the ability to get the current, historical, or potentially even future locations of the mobile user. Localization of the mobile terminals can be achieved through a wide variety of positioning technologies, including but not limited to, cellular network based positioning, GPS based positioning, geo–referenced sensor based positioning, or even geo–referenced user entry. Mobile users access LBSes through their mobile terminals. An LBS is a service that has one or more of the following characteristics. An LBS is either explicitly or implicitly requested by the users via the mobile terminal. An LBS delivers its service selectively based on the context of the mobile user. There are many aspects of user–context, however in this thesis the following context aspects are considered: the current, historical and future locations of the user, any possible user–patterns in the user location data, and common patterns in the location data of a group of users.

Data mining, the process of finding, e.g., associations, or in general patterns, within large amounts of data, often stored in relational databases, has been applied successfully in the past to increase business revenue. More recently, data mining has been suggested to be useful to derive context for user–friendly LBSes. A large part of this context for many LBSes can be described by general patterns in user activities. One type of user activity is *where* and *when* users were in the past. Data mining methods for such activities have to consider two additional dimensions, namely the *spatial*, the *temporal* or jointly the *spatio–temporal* dimension, and are referred to as spatio–temporal data mining methods. Due to the unique nature of the two additional dimensions, i.e., the cardinality of the dimensions are potentially extremely large, and the temporal dimension is cyclic for some types of applications, mining spatio–temporal patterns poses additional challenges. To address these challenges, Chapter 2 presents a general methodology, *pivoting*, to extend heavily researched rule mining methods, in particular association rule mining. By considering the characteristics of a number of real–world data sources, Chapter 2 also derives a taxonomy of spatio–temporal data, and demonstrates the usefulness of the rules that the extended spatio–temporal rule mining method can mine.

Chapter 4 uses pivoting to extend a projection–based frequent itemset mining method to discover spatio–temporal sequential patterns, i.e., frequent routes, in GPS traces. The extended method, through (either region–based or road network based) spatio–temporal generalization, first preprocesses the GPS traces to obtain spatio–temporal itemsets. Then, a variant of a database projection based closed frequent itemset mining method prunes the search space by making use of the minimum length and sharable requirements and avoids the generation of the exponential number of sub–routes of long routes. Considering alternative modelling options for trajectories leads to the development of two effective variants of the method. SQL–based implementations are described, and extensive experiments on both real–life- and large–scale synthetic data show the effectiveness of the method and its variants.

Simulation is widely accepted in database research as a low–cost method to provide synthetic data for designing and testing novel data types and access methods. This is even more so the case for trajectory data, where the availability of real–world data about a large population of moving objects is limited. Prior research produced a number of moving object simulators that model the physical aspects of mobility to various degrees, but fail to adequately address the important *social* and *geo–demographical* aspects of mobility. Modelling the latter aspects gives rise to unique spatio–temporal data distributions with regularities. Hence, to aid the development in spatio–temporal data management and data mining, Chapter 3 develops ST–ACTS, a Spatio–Temporal ACTivity Simulator. ST–ACTS uses a number of real–world geo–statistical data sources and intuitive principles to generate realistic spatio–temporal activities of mobile users. ST–ACTS considers that (1) objects (representing mobile users) move from one spatio–temporal location to another with the objective of performing a certain activity at the latter location; (2) not all users are equally likely to perform a given activity; (3) certain activities are performed at certain locations and times; and (4) activities exhibit regularities that can be specific to a single user or to groups of users. Experiments demonstrate that ST–ACTS is effective and that the characteristics of the generated data correspond to the input model parameters.

Clustering, the process of finding groups of similar objects among a large set of objects, is another common general data mining technique. Clustering the spatio–temporal domains have been largely neglected in the past. However, as Chapter 5 demonstrates, answering service requests in meaningful spatio–temporal groups can yield promising LBSes. The chapter presents one such LBS in the transportation domain (cab–sharing) along with a meaningful grouping method for the specific service. The grouping method uses a number of approximations and heuristics to find a near–optimal solution in the combinatorial problem space. The grouping method is expressed as a small set of standard SQL queries. Based on synthetic data derived from ST–ACTS, the chapter demonstrates that the proposed method can effectively group together cab–requests, making cab–sharing a new, promising mode of transportation.

Chapter 6 outlines how the grouping algorithm can be adapted to facilitate large–scale collective transportation systems, for example a ride–sharing system. However, Chapter 6 identifies a scalability bottleneck in the grouping algorithm. To eliminate the bottleneck, the chapter (1) expresses the grouping algorithm as a continuous stream query in a data stream management system, and (2) devises simple but effective spatio–temporal partitioning methods for streams to parallelize the computation. Extensive experimental results show that the parallel implementation using simple adaptive partitioning methods can achieve speed–ups of several orders of magnitude without significantly effecting the quality of the grouping.

Another highly promising LBS lies in the mobile advertising domain. The success of mobile advertising hinges on the ability to deliver only *relevant* information to the mobile consumer. Chapter 7 investigates models for Location–Based Advertising (LBA) where the relevance of a mobile ad depends on at least two factors: (1) the *proximity* of the mobile consumer to the product or service being advertised, and (2) the *match* between the product or service and the *interest* of the mobile consumer. While the consumer can express his/her interest *explicitly*, as demonstrated, it can also be *implicitly* inferred through data mining. To give indications for the business potential of LBA, using synthetic data from ST–ACTS, the chapter gives estimates for the capacity of the LBA channel both in the explicit and implicit case. Results show that the capacity of the Location–Based Advertising channel is rather large, which is evidence for a strong business case, but it also necessitates effective user–controls for the received mobile ads, some of which are proposed in the chapter.

To receive LBSes, users have to be willing to disclose their current, historical, or future locations. Such a disclosure naturally raises concerns among the users about potentially being tracked and followed. Hence, to assure user acceptance of LBSes, the privacy of users is of great importance. To this extent, Chapters 8 and 9 address location privacy concerns in connection with data mining of user locations. More specifically, Chapter 8 proposes a privacy–preserving location data collection and mining system. The system uses a general framework that allows user location data to be anonymized through spatio–temporal generalization, thus preserving privacy. The data mining component of the system mines anonymized location data and derives probabilistic spatio–temporal patterns. A privacy–preserving method is proposed for the core data mining task of *finding dense spatio–temporal regions*. An extensive set of experiments evaluate the method, comparing it to its non–privacy–preserving equivalent. The experiments show that the framework allows most patterns to be found, even when privacy is preserved.

The anonymization process proposed in Chapter 8 introduces some uncertainty in the patterns. To eliminate this uncertainty in patterns, Chapter 9 first adopts and combines existing privacy definitions to derive privacy definitions of various strengths for location data. Then the chapter presents a complete system for the privacy–preserving collection of *exact* trajectories. The system is composed of an untrusted server and clients communicating in a P2P network. Location data is anonymized in the system using data cloaking and data swapping techniques. Experiments on simulated but realistic movement data indicate that the proposed system is effective under reasonable conditions and privacy / anonymity settings.

The contents of this thesis are based on the contents of papers that have either been published, are to appear, or are under consideration for publication.

Since Chapters 2 to 9 are based on individual publications, they are self contained and can be read in isolation. Since some of these chapters are closely related, this entails a certain amount of overlap. In particular, some of the methods and the base data used in the data generator, ST–ACTS, in Chapter 3, are also used in the estimations in Chapter 7. Hence, there is a strong correspondence between Sections 3.4 and 3.5.4, and Sections 7.5.3 and 7.4, respectively. Furthermore, since ST–ACTS is used to generate synthetic data for experiments in most of the papers and is referenced extensively throughout the chapters, it is placed early in the thesis. Similarly, since Chapter 6 provides a scalable implementation of the algorithm presented in Chapter 5, the motivation in Section 5.1, the problem definition in Section 5.2, the service description in Section 5.3, and the description of the basic algorithm in Section 5.4 closely correspond to Sections 6.1, 6.3.1, 6.3.2, and 6.3.3, respectively. Finally, since both Chapters 8 and 9 consider location privacy in connection with data mining of trajectories, the motivations and review of related work in Section 8.1 are similar to that presented in Sections 9.1 and 9.2.

The papers that the thesis is based on are listed below. Papers 3 and 6 are extended journal versions of the conference papers 9 and 10, respectively. Chapters 2 to 9 are based on papers 1 to 8, respectively.

1. G. Gidófalvi and T. B. Pedersen. Spatio–Temporal Rule Mining: Issues and Techniques. In *Proc. of the 7th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, volume 3589 of Lecture Notes in computer Science, pp. 275–284, Springer, 2005.

2. G. Gidófalvi and T. B. Pedersen. ST–ACTS: A Spatio–Temporal Activity Simulator. In *Proc. of the 14th ACM International Symposium on Geographic Information Systems, ACM–GIS*, pp. 155–162, ACM, 2006.

3. G. Gidófalvi and T. B. Pedersen. Mining Long, Sharable Patterns in Trajectories of Moving Objects. To appear in *Geoinformatica*, 35 pages, 2008.

4. G. Gidófalvi and T. B. Pedersen. Cab–Sharing: An Effective, Door–To–Door, On–Demand Transportation Service. In *Proc. of the 6th European Congress and Exhibition on Intelligent Transport Systems and Services, ITS*, 2007.

5. G. Gidófalvi, T. B. Pedersen, T. Risch, and E. Zeitler. Highly Scalable Trip Grouping for Large–Scale Collective Transportation Systems. To appear in *Proc. of the 11th International Conference on Extending Database Technology, EDBT*, 12 pages, 2008.

6. G. Gidófalvi, H. R. Larsen, and T. B. Pedersen. Estimating the Capacity of the Location–Based Advertising Channel. To appear in *International Journal of Mobile Communications, IJMC*, 18 pages, Inderscience Publishers, 2008.

7. G. Gidófalvi, X. Huang, and T. B. Pedersen. Privacy–Preserving Data Mining on Moving Object Trajectories. In *Proc. of the 8th International Conference on Mobile Data Management, MDM*, 2007.

8. G. Gidófalvi, X. Huang, and T. B. Pedersen. Privacy–Preserving Trajectory Collection. Submitted to *the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD*, 12 pages, 2008.

9. G. Gidófalvi and T. B. Pedersen. Mining Long, Sharable Patterns in Trajectories of Moving Objects. In *Proc. of the 3rd Workshop on Spatio–Temporal Database Management, STDBM*, volume 174 of Online Proceedings of CEUR–WS, pp. 49–58, CEUR–WS, 2006.

10. G. Gidófalvi, H. R. Larsen, and T. B. Pedersen. Estimating the Capacity of the Location–Based Advertising Channel. In *Proc. of the 2007 International Conference on Mobile Business, ICMB*, pp. 2, IEEE Computer Society, 2007.

Chapter 2

Spatio–Temporal Rule Mining: Issues and Techniques

Recent advances in communication and information technology, such as the increasing accuracy of GPS technology and the miniaturization of wireless communication devices pave the road for Location–Based Services (LBS). To achieve high quality for such services, spatio–temporal data mining techniques are needed. This paper describes experiences with spatio–temporal rule mining in a Danish data mining company. First, a number of real world spatio–temporal data sets are described, leading to a taxonomy of spatio–temporal data. Second, the paper describes a general methodology that transforms the spatio–temporal rule mining task to the traditional market basket analysis task and applies it to the described data sets, enabling traditional association rule mining methods to discover spatio–temporal rules for LBS. Finally, unique issues in spatio–temporal rule mining are identified and discussed.

2.1 Introduction

Several trends in hardware technologies such as display devices and wireless communication combine to enable the deployment of mobile, Location–Based Services (LBS). Perhaps most importantly, global positioning systems (GPS) are becoming increasingly available and accurate. In the coming years, we will witness very large quantities of wirelessly Internet–worked objects that are location–enabled and capable of movement to varying degrees. These objects include consumers using GPRS and GPS enabled mobile–phone terminals and personal digital assistants, tourists carrying on–line and position–aware cameras and wrist watches, vehicles with computing and navigation equipment, etc.

These developments pave the way to a range of qualitatively new types of Internet–based services [56]. These types of services, which either make little sense or are of limited interest in the context of fixed–location, desktop computing, include: traffic coordination and management, way–finding, location–aware advertising, integrated information services, e.g., tourist services.

A single generic scenario may be envisioned for these location–based services. Moving service users disclose their positional information to services, which use this and other information to provide specific functionality. To customize the interactions between the services and users, data mining techniques can be applied to discover interesting knowledge about the behavior of users. For example, groups of users can be identified exhibiting similar behavior. These groups can be characterized based on various attributes of the group members or the requested services. Sequences of service requests can also be analyzed to discover regularities in such sequences. Later these regularities can be exploited to make intelligent predictions about user's future behavior given the requests the user made in the past. In addition, this knowledge can also be used for delayed modification of the services, and for longer–term strategic decision making [57].

An intuitively easy to understand representation of this knowledge is in terms of rules. A *rule* is an implication of the form $A \Rightarrow B$, where A and B are sets of attributes. The idea of mining association rules and the subproblem of mining frequent itemset was introduced by Agrawal et al. for the analysis of market basket data [1]. Informally, the task of mining frequent itemsets can be defined as finding all sets of items that co–occur in user purchases more than a user–defined number of times. The number of times items in an itemset co–occur in user purchases is defined to be the *support* of the itemset. Once the set of high–support, so called *frequent* itemsets have been identified, the task of mining association rules can be defined as finding disjoint subsets A and B of each frequent itemset such that the conditional probability of items in B given the items in A is higher than a user–defined threshold. The conditional probability of B given A is referred to as the *confidence* of the rule $A \Rightarrow B$. Given that coffee and cream are frequently purchased together, a high–confidence rule might be that "60% of the people who buy coffee also buy cream." Association rule mining is an active research area. For a detailed review the reader is referred to [40].

Spatio–temporal (ST) rules can be either *explicit* or *implicit*. Explicit ST rules have a pronounced ST component. Implicit ST rules encode dependencies between entities that are defined by spatial (north–of, within, close–to,. . .) and/or temporal (after, before, during,. . .) predicates. An example of an explicit ST rule is: "Businessmen drink coffee at noon in the pedestrian street district." An example of an implicit ST rule is: “Middle–aged single men often co–occur in space and time with younger women.” This paper describes experiences with ST rule mining in the Danish spatial data mining company, Geomatic.

The task of finding ST rules is challenging because of the high cardinality of the two added dimensions: space and time. Additionally, straight–forward application of association rule mining methods cannot always extract all the interesting knowledge in ST data. For example, consider the previous implicit ST rule example, which extracts knowledge about entities (people) with different attributes (gender, age) that interact in space and time. Such interaction will not be detected when association rule mining is applied in straight–forward manner. This creates a need to explore the special properties of ST data in relation to rule mining, which is the focus of this paper.

The contributions of the paper are as follows. First, a number of real world ST data sets are described, and a taxonomy for ST data is derived. Second, having the taxonomy, the described data sets, and the desirable LBSes in mind, a general methodology is devised that projects the ST rule mining task to traditional market basket analysis. The proposed method can in many cases efficiently eliminate the above mentioned explosion of the search space, and allows for the discovery of both implicit and explicit ST rules. Third, the projection method is applied to a number of different type of ST data such that traditional association rule mining methods are able to find ST rules which are useful for LBSes. Fourth, as a natural extension to the proposed method, spatio–temporally restricted mining is described, which in some cases allows for further quantitative and qualitative mining improvements. Finally, a number of issues in ST rule mining are identified, which point to possible future research directions.

Despite the abundance of ST data, the number of algorithms that mine such data is small. Since the pioneering work of [2], association rule mining methods were extended to the spatial [21, 22, 48, 63], and later to the temporal dimension [67]. Other than in [70, 95], there has been no attempts to handle the combination of the two dimensions. In [95] an efficient depth–first search style algorithm is given to discover ST sequential patterns in weather data. The method does not fully explore the spatial dimension as no spatial component is present in the rules, and no general spatial predicate defines the dependencies between the entities. In [70], a bottom–up, level–wise, and a faster top–down mining algorithm is presented to discover ST periodic patterns in ST trajectories. While the technique can naturally be applied to discover ST event sequences, the patterns found are only within a single event sequence.

The remainder of the paper is organized as follows. Section 2.2 introduces a number of real world ST data sets, along with a taxonomy of ST data. In Section 2.3, a general methodology is introduced that projects the ST rule mining task to the traditional market basket analysis or frequent itemset mining task. The proposed problem projection method is also applied to the example data sets such that traditional association rule mining methods are able to discover ST rules for LBSes. Finally, Sections 2.4 and 2.5 identify unique issues in ST rule mining, conclude, and point to future work.

2.2 Spatio–Temporal Data

Data is obtained by measuring some attributes of an entity/phenomena. When these attributes depend on the place and time the measurements are taken, the data is refer to as ST data. Hence such ST measurements not only include the measured attribute values about the entity or phenomena, but also two special attribute values: a location value, *where* the measurement was taken, and a time value, *when* the measurement was taken. Disregarding these attributes, the non–ST rule "Businessmen drink coffee" would result in annoying advertisements sent to businessmen who are in the middle of an important meeting.

2.2.1 Examples of ST Data Sets

The first ST data set comes from the "Space, Time, and Man" (STM) project [86]—a multi–disciplinary project at Aalborg University. In the STM project activities of thousands of individuals are continuously registered through GPS–enabled mobile phones, referred to as mobile terminals. These mobile terminals, integrated with various GIS services, are used to determine close–by services such as shops. Based on this information in certain time intervals the individual is prompted to select from the set of available services, which s/he currently might be using. Upon this selection, answers to subsequent questions can provide a more detailed information about the nature of the used service. Some of the attributes collected include: location and time attributes, demographic user attributes, and attributes about the services used. This data set will be referred to as STM in the following.

The second ST data set is a result of a project carried out by the Greater Copenhagen Development Council (Hovedstadens Udviklings Råd (HUR)). The HUR project involves a number of city busses each equipped with a GPS receiver, a laptop, and infrared sensors for counting the passengers getting on and off at each bus stop. While the busses are running, their GPS positions are continuously sampled to obtain detailed location information. The next big project of HUR will be to employ chip cards as payment for the travel. Each passenger must have an individual chip card that is read when getting on and off the bus. In this way an individual payment dependent on the person and the length of the travel can be obtained. The data recorded from the chip cards can provide valuable passenger information. When analyzed, the data can reveal general travel patterns that can be used for suggesting new and better bus routes. The chip cards also reveal individual travel patterns which can be used to provide a customized LBS that suggests which bus to take, taking capacities and correct delays into account. In the following, the data sets from the first and second projects of HUR will be referred to as HUR1 and HUR2, respectively.

The third ST data set is the publicly available INFATI data set [53], which comes from the intelligent speed adaptation (INtelligent FArtTIlpasning (INFATI)) project conducted by the Traffic Research Group at Aalborg University. This data set records cars moving around in the road network of Aalborg, Denmark over a period of several months. During this period, periodically the location and speeds of the cars are sampled and matched to corresponding speed limits. This data set is interesting, as it captures the movement of private cars on a day–to–day basis, i.e., the daily activity patterns of the drivers. Additional information about the project can be found in [58]. This data set will be referred to as INFATI in the following.

Finally, the last example data set comes from the Danish Meteorology Institute (DMI) and records at fixed time intervals atmospheric measurements like temperature, humidity, and pressure for Denmark for 5 km grid cells. This data set is unique in that unlike the other data sets it does not capture ST characteristics of moving objects, but nonetheless is ST. This data set will be referred to as DMI in the following.

2.2.2 A Taxonomy of ST Data

Data mining in the ST domain is yet largely unexplored. There does not even exist any generally accepted taxonomy of ST data. To analyze such data it is important to establish a taxonomy.

Perhaps the most important criterion for this categorization is whether the measured entities are *mobile* or *immobile*. The ST data in the DMI data set is immobile in the sense that the temperature or the amount of sunshine does not move from one location to the other, but rather, as a continuous phenomenon, changes its attribute value over time at a given location. On the other hand, the observed entities in the other four data sets are rather mobile.

Another important criterion for categorization is whether the attribute values of the measured entities are *static* or *dynamic*. There are many examples of static attributes values but perhaps one that all entities possess is a unique identifier. Dynamic attributes values change over time. This change can be slow and gradual, like in the case of the age of an observed entity, or swift and abrupt, like in the case of an activity performed by the observed entity, which starts at a particular time and last for a well–specified time interval only.

2.3 Spatio–Temporal Baskets

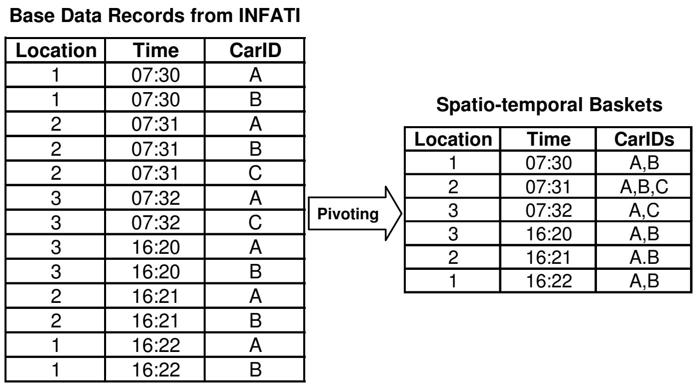
Following the methodology of market basket analysis, to extract ST rules for a given data set, one needs to define ST *items* and *baskets*. This task is important, since any possible knowledge that one can extract using association rule mining methods will be about the possible dependencies of the items within the baskets.

2.3.1 Mobile Entities with Static and Dynamic Attributes

Consider the STM data; it is mobile in nature and has several static and dynamic attributes. Base data contains the identity and some demographic attributes of the user, and the activity performed by user at a particular location and time. Further attributes of the locations where the activity is performed are also available. By applying association rule mining on this base data one can find possible dependencies between the activities of the users, the demographics of the users, the characteristics of the locations there the activities are performed, and the location and time of the activities. Since the location and time attributes are items in the baskets one may find {Strøget,noon,businessman,café} as a frequent itemset and from it the association rule {Strøget,noon,businessman} ⇒ {café}. Strøget being a famous pedestrian street district in central Copenhagen in Denmark, this rule clearly has both a spatial and temporal component and can be used to advertise special deals of a café shop on Strøget to businessmen who are in the area around noon.

In the INFATI data set, a record in the base data contains a location, a time, a driver identifier, and the current speed of the car along with the maximum allowed speed at the particular location. The possible knowledge one can discover by applying association rule mining on the base data is where and when drivers or a particular driver occur(s) and/or speed(s) frequently. However, one may in a sense pivot this table of base data records such that each new row represents an ST region and records the car identifiers that happen to be in that region. Applying association rule mining on these ST baskets one may find which cars co–occur frequently in space and time. Such knowledge can be used to aid intelligent rideshare services. It can also be valuable information for constructing traffic flow models and for discovering travel patterns. While the possible knowledge discovered may be valuable for certain applications, the extracted rules are not clearly ST, i.e.: there is no *explicit* ST component in them. In fact the same set of cars may frequently co–occur at several ST regions which may be scattered in space and time. Nonetheless, it can be argued that since the "co–occurrence" between the items in the ST baskets is actually an ST predicate in itself, the extracted rules are *implicitly* ST.

An alternative to this approach might be to restrict the mining of the ST baskets to larger ST regions. While this may seem useless at first, since the baskets themselves already define more fine–grained ST regions, it has several advantages. First,


Base Data Records from INFATI

Location	Time	CarID
1	07:30	A
1	07:30	B
2	07:31	A
2	07:31	B
2	07:31	C
3	07:32	A
3	07:32	C
3	16:20	A
3	16:20	B
2	16:21	A
2	16:21	B
1	16:22	A
1	16:22	B

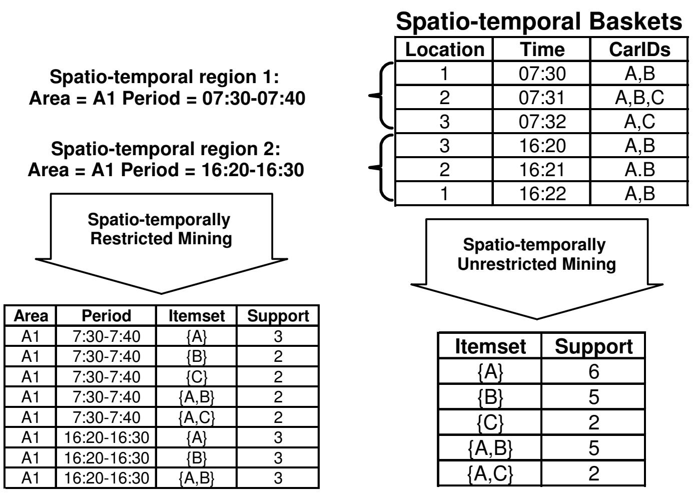
Spatio-temporal Baskets

Location	Time	CarIDs
1	07:30	A,B
2	07:31	A,B,C
3	07:32	A,C
3	16:20	A,B
2	16:21	A.B
1	16:22	A,B

Figure 2.1: Process of Pivoting to Obtain ST Baskets from INFATI Base Data.

it allows the attachment of an explicit ST component to each extracted rule. Second, it enhances the quality of the extracted rules. Finally, it significantly speeds up the mining process, as no two itemsets from different regions are combined and tried as a candidate. Figure 2.1 shows the process of pivoting of some example records abstracted from the INFATI data set. Figure 2.2 shows the process and results of spatio–temporally restricted and unrestricted mining of the ST baskets. In this example the shown frequent itemsets are based on an absolute minimum support of 2 in both cases, however in the restricted case specifying a relative minimum support would yield more meaningful results. Naturally the adjective “relative” refers to the number of baskets in each of the ST regions. Figure 2.2 also shows the above mentioned qualitative differences in the result obtained from spatio–temporally restricted vs. unrestricted mining. While the frequent co–occurrence of cars A and B, and cars A and C are detected by unrestricted mining, the information that cars A and B are approximately equally likely to co–occur in area A1 in the morning as in the afternoon, and that cars A and C only co–occur in area A1 in the morning is missed.

Similar pivoting techniques based on other attributes can also reveal interesting information. Consider the data set in HUR2 and the task of finding frequently travelled routes originating from a given ST region. In the HUR2 data set a record is generated every time a user starts and finishes using a transportation service. This record contains the identifier of the user, the transportation line used, and the location and time of the usage. For simplicity assume that a trip is defined to last at most 2 hours. As a first step of the mining, one can retrieve all the records that fall within the ST region of the origin. Following, one can retrieve all the records within 2 hours of the users that belonged to the first set. By pivoting on the user–identifiers, one can derive ST baskets that contain locations where the user generated a record by making


Spatio-temporal Baskets
Spatio-temporal region 1:
Area = A1 Period = 07:30-07:40
Spatio-temporal region 2:
Area = A1 Period = 16:20-16:30
Spatio-temporally Restricted Mining
Spatio-temporally Unrestricted Mining

Location	Time	CarIDs
1	07:30	A,B
2	07:31	A,B,C
3	07:32	A,C
3	16:20	A,B
2	16:21	A.B
1	16:22	A,B

Area	Period	Itemset	Support
A1	7:30-7:40	{A}	3
A1	7:30-7:40	{B}	2
A1	7:30-7:40	{C}	2
A1	7:30-7:40	{A,B}	2
A1	7:30-7:40	{A,C}	2
A1	16:20-16:30	{A}	3
A1	16:20-16:30	{B}	3
A1	16:20-16:30	{A,B}	3

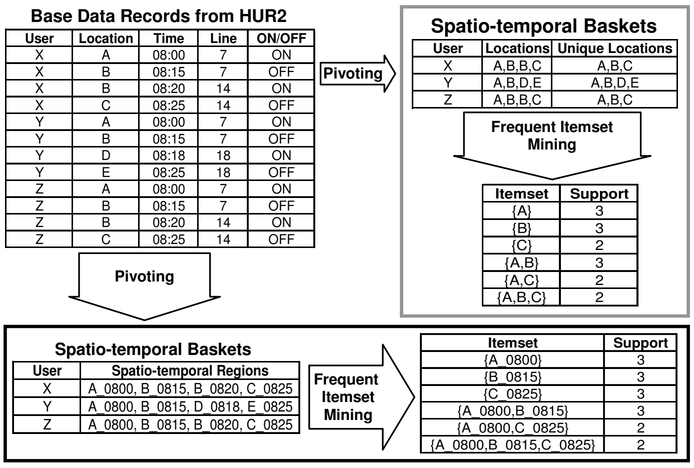
Itemset	Support
{A}	6
{B}	5
{C}	2
{A,B}	5
{A,C}	2

Figure 2.2: Process and Results of Spatio–Temporally Restricted vs. Unrestricted Mining of ST Baskets.

use of a transportation service. Applying association rule mining to the so–derived ST baskets one may find frequently travelled routes originating from a specific ST region. The pivoting process for obtaining such ST baskets and the results of mining such baskets is illustrated in a simple example in the light bordered box of Figure 2.3. Naturally, the frequent itemset mining is only applied to the ”Unique Locations” column of the ST baskets. As before the minimum support is set to 2. Considering the spatial relation between the locations one might consider altering the bus routes to better meet customer needs. For example, if locations A and C are close by on the road network, but no bus line exists with a suitable schedule between A and C, then in light of the evidence, i.e., support of A,B,C is 2, such a line can be added. Note that while the discovered frequent location sets do not encode any temporal relation between the locations, one can achieve this by simply placing ST regions into the ST baskets as items. The pivoting process and the results of mining are shown in the dark bordered box of Figure 2.3. The discovered ST itemsets can help in adjusting timetables of busses to best meet customer needs.

2.3.2 Immobile Entities with Static and Dynamic Attributes

So far the examples considered data sets that are mobile and have either static, dynamic, or both types of attribute values. Now consider an immobile ST data with mostly dynamic attribute values, as the DMI data set. The base data can be viewed as transactions in a relational table with a timestamp, a location identifier and some atmospheric measurements like temperature, humidity, and pressure. Considering the



Base Data Records from HUR2

User	Location	Time	Line	ON/OFF
X	A	08:00	7	ON
X	B	08:15	7	OFF
X	B	08:20	14	ON
X	C	08:25	14	OFF
Y	A	08:00	7	ON
Y	B	08:15	7	OFF
Y	D	08:18	18	ON
Y	E	08:25	18	OFF
Z	A	08:00	7	ON
Z	B	08:15	7	OFF
Z	B	08:20	14	ON
Z	C	08:25	14	OFF

Spatio-temporal Baskets

User	Locations	Unique Locations
X	A,B,B,C	A,B,C
Y	A,B,D,E	A,B,D,E
Z	A,B,B,C	A,B,C

Itemset	Support
{A}	3
{B}	3
{C}	2
{A,B}	3
{A,C}	2
{A,B,C}	2

Spatio-temporal Baskets

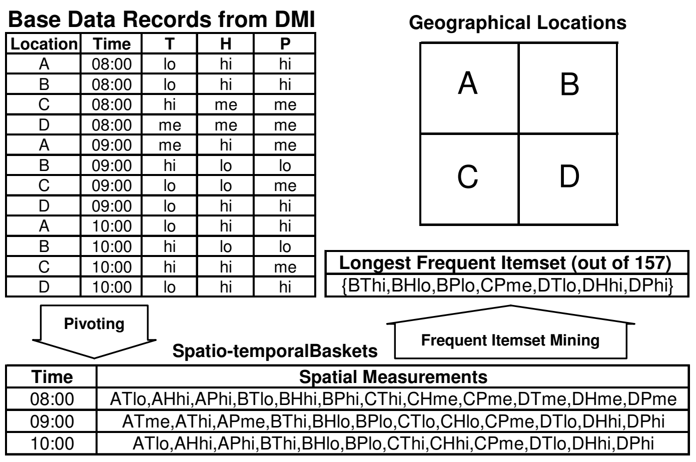
User	Spatio-temporal Regions
X	A_0800, B_0815, B_0820, C_0825
Y	A_0800, B_0815, D_0818, E_0825
Z	A_0800, B_0815, B_0820, C_0825

Itemset	Support
{A_0800}	3
{B_0815}	3
{C_0825}	3
{A_0800,B_0815}	3
{A_0800,C_0825}	2
{A_0800,B_0815,C_0825}	2

Figure 2.3: ST Baskets and Frequent Itemset Mining for HUR2.

geographical locations A, B, C, and D depicted in Figure 2.4, one might be interested in trends like, when the temperature in regions A and B is high and the pressure in regions A and C is low, then at the same time the humidity in region D is medium. By applying something similar to the pivoting techniques above, one can extract such information as follows. For each record concatenate the location identifiers with the atmospheric measurements. Then, for each distinct time interval when measurements are taken, put all concatenated values, each of which is composed of a location identifier and an atmospheric measurement, into a single, long ST basket. By performing association mining on the derived ST baskets one can obtain the desired knowledge.

As an illustrative example, depicted in Figure 2.4, consider the four neighboring cells A, B, C, and D and the corresponding measurements of temperature (T), humidity (H), and pressure (P) at three different times. Items in the ST baskets are derived by concatenating a location identifier followed by an attribute symbol and an attribute value. Hence, the item ‘ATlo‘ in the ST basket at time ‘08:00’ encodes the fact that at ‘08:00’ at location ‘A’ the temperature (‘T’) was low (‘lo’). Notice that the extracted knowledge refers to specific locations. If one is interested in obtaining knowledge about the inter–dependencies of these attributes relative (in space) to one another, for each base data record at each distinct time interval when measurements are taken, an ST basket can be constructed that encodes measurements from neighboring cells only. So, for example considering the immediate 8 neighbors of a cell and assuming three different attributes the number of items in each basket is $3 + 8 \times 3 = 27$. Considering a five–by–five relative neighborhood centered around a cell the number of items in each basket is 75, and the number of possible itemsets, given three possible attribute values for each of the attributes is $3^{75} \approx 6.1 \times 10^{34}$.



Location	Time	T	H	P
A	08:00	lo	hi	hi
B	08:00	lo	hi	hi
C	08:00	hi	me	me
D	08:00	me	me	me
A	09:00	me	hi	me
B	09:00	hi	lo	lo
C	09:00	lo	lo	me
D	09:00	lo	hi	hi
A	10:00	lo	hi	hi
B	10:00	hi	lo	lo
C	10:00	hi	hi	me
D	10:00	lo	hi	hi

Longest Frequent Itemset (out of 157)
{BThi,BHlo,BPlo,CPme,DTlo,DHhi,DPhi}

Time	Spatial Measurements
08:00	ATlo,AHhi,APhi,BTlo,BHhi,BPhi,CThi,CHme,CPme,DTme,DHme,DPme
09:00	ATme,AThi,APme,BThi,BHlo,BPlo,CTlo,CHlo,CPme,DTlo,DHhi,DPhi
10:00	ATlo,AHhi,APhi,BThi,BHlo,BPlo,CThi,CHhi,CPme,DTlo,DHhi,DPhi

Figure 2.4: ST Baskets and Frequent Itemset Mining of DMI.

To reduce complexity, top–down and bottom–up mining can occur at different spatial and temporal granularities.

While in the above examples the type of ST data that was analyzed and the type of ST knowledge that was extracted is quite different the underlying problem transformation method—referred to as *pivoting*—is the same. In general, one is given base records with two sets of attributes A and B, which are selected by a data mining expert and can contain either spatial, temporal and/or ordinary attributes. Pivoting is then performed by grouping all the base records based on the A–attribute values and assigning the B–attribute values of base records in the same group to a single basket. Bellow, attributes in A are referred to as *pivoting* attributes or *predicates*, and attributes in B are referred to as *pivoted* attributes or *items*. Depending on the type of the pivoting attributes and the type of the pivoted attributes the obtained baskets can be either *ordinary*, *spatial*, *temporal*, or *ST* baskets. Table 2.1 shows the different types of baskets as a function of the different types of predicates used to construct the baskets and the different types of items placed in the baskets. The sym-

pred/item type	s–i	t–i	st–i	ordinary–i
s–predicate	s–b	**st–b**		s–b
t–predicate	**st–b**	t–b		t–b
st–predicate	**st–b**	**st–b**	**st–b**	**st–b**
other–predicate	s–b	t–b	**st–b**	ordinary–b

Table 2.1: Types of Baskets as a Function of Predicate Type and Item Type.

basket/mining type	s–r	t–r	st–r	unr
s–basket	X			X
t–basket		X		X
st–basket	X	X	X	X
other–basket				X

Table 2.2: Possible Mining Types of Different Types of Baskets.

bols s, t, st, i, and b in the table are used to abbreviate the terms ‘spatial’, ‘temporal’, ‘spatio–temporal’, ‘items’, and ‘baskets’ respectively.

In the “co–occurrence” mining task, which was earlier illustrated on the INFATI data, the concept of restricted mining is introduced. This restriction is possible due to a side effect of the pivoting technique. When a particular basket is constructed, the basket is assigned the value of the pivoting attribute as an implicit label. When this implicit basket label contains a spatial, temporal, or ST component, restricting the mining to a particular spatial, temporal, or ST subregion becomes a natural possibility. It is clear that not all basket types can be mined using spatial, temporal, or ST restrictions. Table 2.2 shows for each basket type the type of restrictions for mining that are possible. The symbols s, t, st, r, and unr in the table are used to abbreviate the terms ‘spatial’, ‘temporal’, ‘spatio–temporal’, ‘restricted’, and ‘unrestricted’ respectively.

2.4 Issues in Spatio–Temporal Rule Mining

The proposed pivoting method naturally brings up questions about feasibility and efficiency. In cases where the pivoted attributes include spatial and/or temporal components, the number of items in the baskets is expected to be large. Thus, the number and length of frequent itemsets or rules is expected to grow. Bottom–up, level–wise algorithms are expected to suffer from excessive candidate generation, thus top–down mining methods seem more feasible. Furthermore, due to the presence of very long patterns, the extraction of all frequent patterns has limited use for analysis. In such cases closed or maximal frequent itemsets can be mined.

Useful patterns for LBSes are expected to be present only in ST subregions, hence spatio–temporally restricted rule mining will not only make the proposed method computationally more feasible, but will also increase the quality of the result. Finding and merging patterns in close–by ST subregions is also expected to improve efficiency of the proposed method and the quality of results.

Placing concatenated location and time attribute values about individual entities as items into an ST basket allows traditional association rule mining methods to extract ST rules that represent ST event sequences. ST event sequences can have numerous applications, for example an intelligent ride–sharing application, which finds common routes for a set of commuters and suggests rideshare possibilities to them. Such an application poses a new requirement on the discovered itemsets, namely, they primarily need to be "long" rather than frequent (only a few people will share a given ride, but preferably for a long distance). This has the following implications and consequences. First, all subsets of frequent and long itemsets are also frequent, but not necessarily long and of interest. Second, due to the low support requirement a traditional association rule mining algorithm, disregarding the length requirement, would explore an excessive number of itemsets, which are frequent but can never be part of a long and frequent itemset. Hence, simply filtering out "short" itemsets after the mining process is inefficient and infeasible. New mining methods are needed that efficiently use the length criterion during the mining process.

2.5 Conclusions and Future Work

Motivated by the need for ST rule mining methods, this paper established a taxonomy for ST data. A general problem transformation method was introduced, called pivoting, which when applied to ST data sets allows traditional association rule mining methods to discover ST rules. Pivoting was applied to a number of ST data sets allowing the extraction of both explicit and implicit ST rules useful for LBSes. Finally, some unique issues in ST rule mining were identified, pointing out possible research directions.

Future work will devise and empirically evaluate algorithms for both general and spatio–temporally restricted mining, and more specialized types of mining such as the ride–sharing suggestions. Especially, algorithms that take advantage of the above–mentioned "long rather than frequent" property of rideshare rules will be interesting to explore.

Chapter 3

ST–ACTS: A Spatio–Temporal Activity Simulator

Creating complex spatio–temporal simulation models is a hot issue in the area of spatio–temporal databases [80]. While existing Moving Object Simulators (MOSs) address different *physical* aspects of mobility, they neglect the important *social* and *geo–demographical* aspects of it. This paper presents ST–ACTS, a Spatio–Temporal ACTivity Simulator that, using various geo–statistical data sources and intuitive principles, models the so far neglected aspects. ST–ACTS considers that (1) objects (representing mobile users) move from one spatio–temporal location to another with the objective of performing a certain activity at the latter location; (2) not all users are equally likely to perform a given activity; (3) certain activities are performed at certain locations and times; and (4) activities exhibit regularities that can be specific to a single user or to groups of users. Experimental results show that ST–ACTS is able to effectively generate realistic spatio–temporal distributions of activities, which make it essential for the development of adequate spatio–temporal data management and data mining techniques.

3.1 Introduction

Simulation is widely accepted in database research as a low–cost method to provide synthetic data for designing and testing novel data types and access methods. Moving objects databases are a particular case of databases that represent and manage changes related to the movement of objects. To aid the development in moving object database research, a number of Moving Object Simulators (MOSs) have been developed [8,51, 77,81,83,91].

The so far developed MOSs have been using parameterizable random functions and road networks to model different physical aspects of the moving objects–such as their extent, environment and mobility–but they all neglect some important facts. When moving objects represent mobile users, most of the time the reason for movement is due to a clear objective. Namely, users move from one spatio–temporal location to another to accomplish some task, from hereon termed as perform an activity, at the latter location. For example, people do not just spend most of their nights at a particular location, they come *home* to be with their loved ones, to relax, eat and sleep. Similarly, people do not just spend most of their working days at any particular location, they go to a real–world facility, their *work place*, with the intention of working. Finally, based on their habits and likes, in their spare time, people (more or less regularly) go to other real–world facilities, which they like and are nearby.

To model the above mentioned social aspects of mobility is important for two reasons. First, the locations and times where activities can be performed and the patterns in these performed activities define a unique spatio–temporal distribution of moving objects that is essential for spatio–temporal database management. Second, the social aspects of mobility are essential when one wishes to extract spatio–temporal knowledge about the regularities in the behavior of mobile users. The field of spatio–temporal data mining is concerned with finding these regularities or patterns. To develop efficient and effective spatio–temporal data management and data mining techniques, large sets of spatio–temporal data is needed; and while location–enabled mobile terminals are increasingly available on the market, such data sets are not readily available.

Hence, to aid the development in spatio–temporal data management and data mining techniques, this paper presents ST–ACTS, a probabilistic, parameterizable, spatio–temporal activity simulator, which is based on a number of real–world data sources consisting of:

- fine–grained geo–demographic population,
- information about businesses and facilities, and
- related consumer surveys.

The importance of the use of real–world data sources in ST–ACTS lies in the fact, that they form a realistic base for simulation. Concretely, variables within any given data source are dependent, and perhaps most importantly geo–dependent. For example, there is a strong dependence between the education and the personal income of people. The variables are also geo–dependent, due to the fact that similar people or similar businesses tend to form clusters in the geographical space. Furthermore, variables are geo–dependent across the different data sources. For example, people working in bio–technology tend to try to find homes close to work places in that business branch. Using real–world data from various commercial geo–statistical databases and common sense principles, ST–ACTS captures some of the to date not modelled, yet important, characteristics of spatio–temporal activity data.

The remainder of this paper is organized as follows. Section 3.2 reviews related work. Section 3.3 defines the objectives of the simulation model. Section 3.4 describes in detail the source data that forms the basis for the simulation model. Section 3.5 describes each component of the simulator and how the source data is used in each component. Section 3.6 evaluates the simulation model in terms of its efficiency and its simulation objectives by examining the characteristics of some simulated data. Finally Section 3.7 concludes and points to future research directions.

3.2 Related Work

Due to the short history of spatio–temporal data management, scientific work on spatio–temporal simulation can be restricted to a handful of publications. The first, significant spatio–temporal simulator is GSTD (Generate Spatio–Temporal Data) [91]. Starting with a distribution of points or rectangular objects, at every time step GSTD recalculates positional and shape changes of objects based on parameterized random functions. Through the introduction of a new parameter for controlling the change of direction and the use of rectangular objects to model obstacles, GSTD is extended to simulate more realistic movements, such as *preferred movement*, *group movements* and *obstructed movement* [77]. Since most objects use a network to get from one location to the other, Brinkhoff presents a framework for network–based moving object simulation [8]. The behavior of a moving object in this framework is influenced by (1) the attributes of the object having a particular object class, (2) the combined effects of the locations of other objects and the network capacity, and (3) the location of external objects that are independent of the network. These simulators and frameworks primarily model the physical aspects of mobility. While they can all be extended to model the social aspects, i.e., the objective for movement and the regularities in these objectives, they do not pursue to do so.

Nonetheless, the importance of modelling these social aspects of mobility is pointed out in [8]. In comparison, ST–ACTS focuses on these social aspects of mobility while placing only limited constrains on the physical aspects of mobility. In effect, the problem solved by the above MOSs is orthogonal to the problem solved by ST–ACTS.

In Oporto [81]–a realistic scenario generator for moving objects motivated by a fishing application–the moving behavior of objects is influenced by other, either stationary or moving, objects of various object types. The influence between objects of different types can either be attraction or repulsion. While the repulsive and attractive influence of other objects is an objective for movement, unlike ST–ACTS, Oporto does not allow the modelling of regularities in these objectives.

The GAMMA [51] (Generating Artificial Modeless Movement by genetic– Algorithm) framework represents moving object behavior as a trajectory in the location–temporal space and proposes two generic metrics to evaluate trajectory data sets. The generation of trajectories is treated as an optimization problem and is solved by a genetic algorithm. With appropriately modified genetic operators and fitness criteria the framework is used to generate cellular network trajectories that as frequently as possible cross cell boarders, and symbolic location trajectories that (1) exhibit mobility patterns similar to those present in a set of real–life sample trajectories given as input, (2) conform to real–life constraints and heuristics. Based on sample activity trajectories, the GAMMA framework can be configured to generate activity trajectories that contain real–life activity patterns. While the generated trajectories will be similar to the input trajectories, since they are symbolic, they will, as the input trajectories implicitly assume a location–dependent context, (see third and fourth principle in Section 3.3). To simulate spatio–temporal activities of an entire population, a representative sample of context–dependent trajectories is needed, but is hard to obtain. In comparison, ST–ACTS, based on intuitive principles and a number of real–life geo–statistical data sources, is able to generate realistic, spatio–temporal activity data that takes this location–dependent context of activities into account.

Time geography [46] is a conceptual basis/paradigm for human space–time behavior which considers (1) the indivisibility or corporeality of the human condition; (2) that humans typically operate over finite intervals of space and time; (3) the natural laws and social conventions that partially constrain space–time behavior; and (4) that humans are purposive. ST–ACTS models some aspects of this paradigm in a concrete, implemented data generator.

3.3 Problem Statement

Existing MOSs capture only *physical* aspects of mobility, i.e., the *movement* of the objects, adequately. However, to aid the development of spatio–temporal data management and data mining methods, *social* aspects of mobility that arise from human behavioral patterns should be captured by a model. The most important principles that govern these social aspects of mobility are:

First Principle: People move from a given location to another location with an *objective of performing some activity* at the latter location.

Second Principle: Not all people are equally likely to perform a given activity. The *likelihood of performing an activity* depends on the interest of a given person, which in turn depends on a number of demographic variables.

Third Principle: The *activities performed by a given person are highly context dependent*. Some of the more important parts of this context are: the current location of the person, the set of possible locations where a given activity can be performed, the current time, and the recent history of activities that the person has performed.

Fourth Principle: The *locations of facilities,* where a given activity can be performed, are *not randomly distributed*, but are influenced by the locations of other facilities and the locations of the users those facilities serve.

The first principle can be thought of as an axiom that is in relation to Newton's first law of motion. Movement that is motivated by the sole purpose of movement and does not obey this principle–for example movement arising from outdoor exercise activities–are not modelled.

The second principle can be rectified by many examples from real life. Two of these examples are that elderly people are more likely to go to a pharmacy than younger people and younger people are more likely to go to a pop or rock concert than elderly people.

The third, perhaps most important principle, is due to several factors. First, movement is a necessary (not always pleasurable) requirement to perform some activity, and hence in most cases the amount of movement required to do so is minimized by the actor, i.e., people tend to go to a café that is near by. Second, activities are not performed with equal likelihood at different times. For example, most people tend to go to work in the morning hours as opposed to other parts of the day; consequently the likelihood of performing that activity during in the morning is higher than during other periods of the day. Furthermore, due to their nature, different activities have different durations. The duration of a given activity puts a natural constraint on the possibility of performing another activity while the previous activity lasts. For example, people tend to start to work from the morning hours for a duration of approximately 8 hours; consequently the likelihood of grocery shopping during the same period is lower than otherwise. Finally, while a person may perform an activity with a very high likelihood, the activities performed by the person are not temporally independent. For example, it is very unlikely that even a person who likes pop and rock concerts a lot, goes to several performances during the same Saturday evening.

The fourth principle is mainly a result of the supply–and–demand laws of economics. Locations of facilities are mainly influenced by competition, market cost, and market potential. For example, even though the cost of establishing a solarium salon on the outskirts of town might be low, the market potential might not even compensate this low cost. Hence it is very unlikely that one will finds several solarium salons on one city block. The spatial process that gives rise to locations of facilities is a complex, dynamic process with feed–back, which is governed by the laws of competitive markets. Hence, using a snapshot of the spatial distribution of real–world facilities as contextual information forms a reasonable basis for constructing a realistically model of spatio–temporal activities that can be performed at those facilities.

The primary *qualitative* objective of the simulation model is to capture the above described governing principles of human behavioral patterns and is referred to as the *validity* of the simulation model. In addition, the simulation model has to achieve a number of *quantitative* objectives. First, the simulation model has to be *effective*, i.e., it has to be able to generate large amounts of synthetic data within a reasonable time. Second, the simulation model has to be *parameterizable*, i.e., based on user–defined parameters it has to be able to generate synthetic data sets with different sizes and characteristics. Finally, the simulation model has to be *correct*, i.e., the synthetic data produced by model has to have the same statistical properties with respect to patterns as it is defined by the model parameters and inputs.

3.4 Source Data

The source data used in the simulation model are commercial products of Geomatic, a Danish company specializing in geo–demographic data and analysis for market segmentation, business intelligence, and direct marketing [28]. Due to the commercial nature of these data sets, the methods of their exact derivations are not to be described herein. Nonetheless, concepts and principles used in the derivation process and the resulting relevant contents of the databases are explained below.

3.4.1 conzoom® Demographic Data

conzoom® is a commercial database product that contains fine–grained, geo–demographic information about Denmark's population [28]. The variables that describe the statistical characteristics of the population can be divided into three groups: *person*, *housing unit*, and *household* variables. These variables and the number of categories for each is shown in Table 3.1.

In Table 3.1, variables that have "type" in their names are categorical variables; variables that have "count" in their name are counts of the corresponding entities within a 100–meter grid cell; and finally, the rest of the variables are continuous variables that have been categorized into categories that are meaningful for market segmentation.

Since, for example in the countryside, the number of persons, households or units could be very low in a 100–meter grid cell, grid cells are grouped together into meaningful, large enough clusters to comply with social and ethical norms and preserve the privacy of individuals. The basis for clustering is twofold: geography and the

referred entity	conzoom® variable	categories
person	person count	1
	age	9
	education type	9
	employment status type	12
	employment branch type	12
housing unit	unit count	1
	house type	6
	house ownership type	4
	house area	5
household	household count	1
	family type	5
	fortune	6
	personal income	5

Table 3.1: Variables in conzoom®.

publicly available one–to–one housing information. The intuition behind the basis is also twofold. First, people living in a given geographical region (be that a state, a county, a postal district) are similar in some sense; for example, they might have a more similar political orientation from people living in another geographical region. Second, people living in similar houses are likely to be similar in other demographic variables; for example an established family with a stable source of income is more likely to be able to buy a larger, more expensive house than a person who just started his/her career. As mentioned earlier, to preserve the privacy of individuals, the clusters are constrained to contain at least some fixed number of households. Statistics for the variables, depending on the sensitivity of the information contained in them, are obtained from Statistics Denmark [85] for clusters constructed at an appropriate level of cluster size constraint, for example 20, 50, 100, and 150 households per cluster. In case of a continuous variable, for example age, counts of the corresponding entities (in this case persons in the cluster) are obtained for the categories of the given variable.

Due to this constrained geo–clustering method, the conzoom® clusters obtained comply with the social and ethical norms and preserve the privacy of the individual, yet the statistics obtained are accurate enough for effective market segmentation. This segmentation results in grouping the Danish population into 29 conzoom® types, which are defined for each 100–meter grid cell. Cosmopolitan (type 3) is one example of the 29 conzoom® types. Comparing the demographics of type 3 to the demographics of the rest of Denmark's population gives the *demographic profile* of the type. This profile is partially shown in Figure 3.1. It roughly describes individuals that are more likely: to be middle aged (30–59 years old), to live in larger cities in larger, multi–family houses that are either owned by them or are private rentals, to

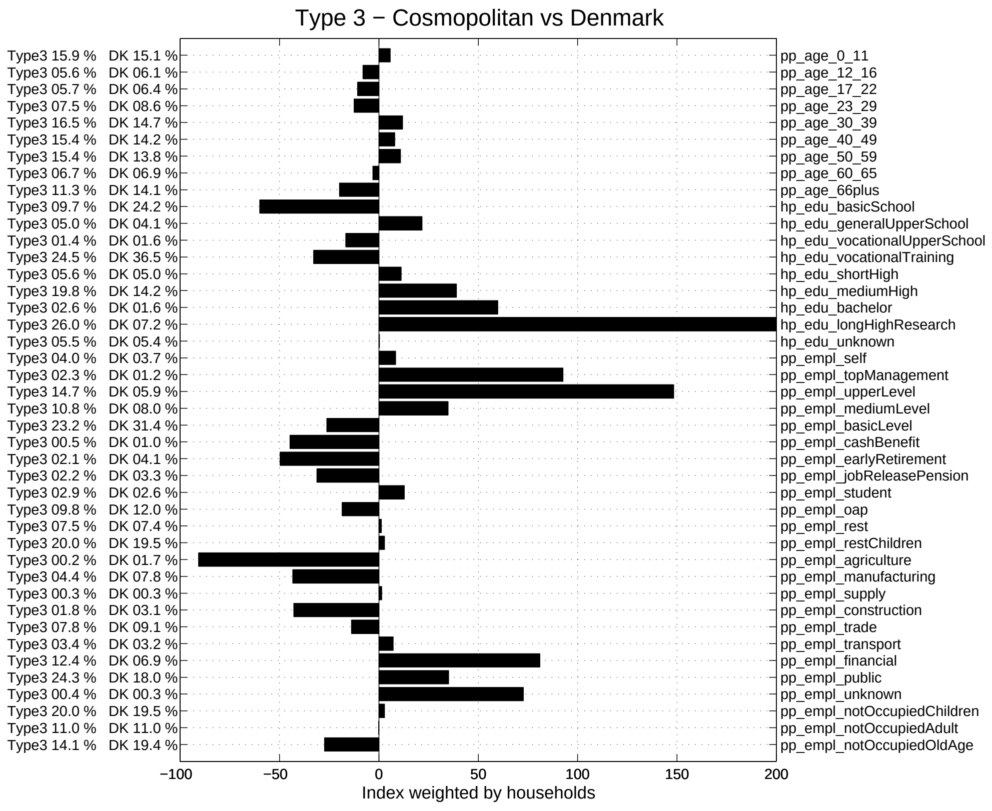


Figure 3.1: Partial Profile of conzoom® Type 3.

be mostly couples with children, to have a medium to long higher education, to hold higher level or top management positions in the financial or public sector, and to have a better household economy (both in terms of wealth and income) than the average Dane.

3.4.2 mobidk™ Daily Movement Data

mobidk™ is an upcoming, commercial database product that contains detailed information about the daily movement of the Danish population between home and work [28]. Again, to preserve the privacy of users, the movement data is aggregated to non–overlapping and connected geographical regions. It is represented in a relational database format as: $\langle from\_region, to\_region, count \rangle$, meaning that from the geographical region *from_region*, *count* number of people move on a daily basis for work to the geographical region *to_region*. In ST–ACTS, these geographical regions are parishes, which on average contain 1176 households, and 195 100–meter grid cells [1].

3.4.3 bizmark™ Business Data

bizmark™ is a commercial database product that contains detailed information about Danish businesses both in the public and the private sector [28]. Some of the one–to–one information that is available about businesses is their location, the number of employees working in them, the physical size of the business facility, and the international branch codes the businesses fall under. Detailed but aggregated information about the employees within businesses is also available for appropriate bizmark™ clusters, which are constructed taking into account geography, business branch, business size in term of number of employees and physical size of the business facility, and various other descriptive business variables.

3.4.4 GallupPC® Consumer Survey Data

GallupPC® is a commercial database product and as the name suggests, it contains detailed survey responses of consumers about their demographics; interests such as culture, hobbies, and sports; household consumptions, purchasing habits; transportation habits; views on various subjects; attitudes and exposure to various advertisement media [26]. The questions in the surveys are yes/no questions. To measure the magnitude of the response of an individual survey subject to a specific question, the original yes/no question is re–phrased with a reference to a time–frequency interval. For example the original yes/no question "Do you go to the library?" is re–phrased to 7 yes/no questions using the following time–frequency intervals: daily / almost daily; 3-4 times a week; 1-2 times a week; 1-3 times a month; 1-5 times every 6 month; seldom, and never.

3.5 ST–ACTS: Spatio–Temporal ACTivity Simulator

In this section, main components of ST–ACTS and their use of the source data is described. In the description a simulated person, who performs activities in time and space, will be abbreviated as a simperson. A MATLAB toolbox for ST–ACTS can be downloaded for research purposes from `http://www.geomatic.dk/research/ST--ACTS/`.

[1]The commercial version of mobidk™ contains the same information for smaller, neighborhood clusters that on average contain 230 households and 38 100–meter grid cells.

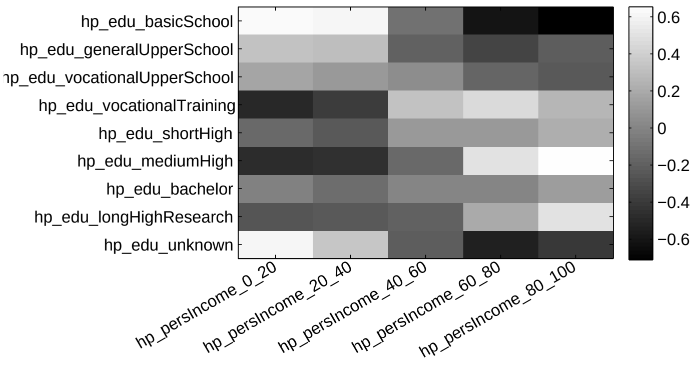


Figure 3.2: Correlation between education and income.

3.5.1 Drawing Demographic Variables for Simpersons

The conzoom® source data contains accurate, detailed demographic information about the population aggregated to a cluster level. As described in Section 3.4.1, continuous variables are discretized into categories. Clusters contain counts for all categories for all variables. Having the exact number of persons, housing units, and households at a grid cell level, and assuming the same distribution of variables in the individual grid cells as in the cluster they belong to, counts for all categories for all variables are calculated at a grid cell level. A simperson is assigned a category for a given variable proportional to the counts of the categories for the given variable in the grid cell the simperson lives in. In short, a category for the variable is assigned to the simperson according to the distribution of the variable. To draw assign categories for variables without replacement, corresponding counts in the given grid cell are decremented. Since counts of some of the variables in the grid cell refer to entities other than persons, but are variables that are part of the demographic variables that describe a person, these counts are adjusted to sum to the number of persons in the cell.

3.5.2 Skewing Distributions Based on Correlations

The above described method for assigning categories for demographic variables has one major flaw: demographic variables are not independent. For example the education type variable has a strong correlation with the personal income variable. This correlation is illustrated in Figure 3.2. Correlations are calculated between the percentages of the categorized variables, and samples are weighted by the number of persons in the cells. From the colorbar on the side one can see that darker shades mean stronger negative correlations and lighter shades mean stronger positive corre-

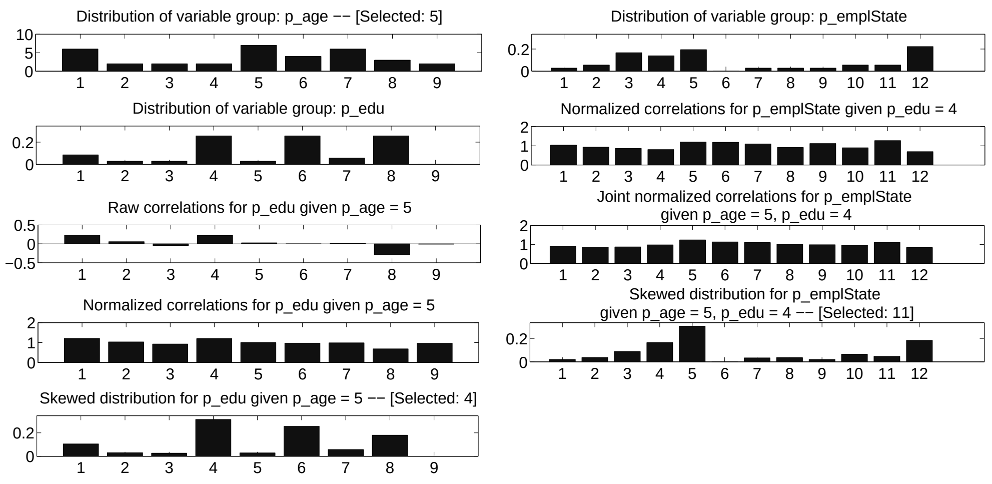


Figure 3.3: Drawing Samples Without Replacement from Correlated, Multivariate Distributions.

lations. The correlations support the common knowledge that people having higher education levels tend to have better paying jobs. Similar correlations exist between other variables.

To remedy the above described flaw, which could result in unrealistic assignment of categories for variables to simpersons, the assignment is modified by drawing categories from skewed variable distributions that try to embed the correlations between the variables as follows. For a given simperson, the category for the first variable, age, is drawn without replacement from unskewed distribution of the age variable. An example of this distribution and the result of the draw is shown in the top most left subgraph of Figure 3.3, where for the age variable the category 5 was drawn, which represents that the simperson is in the age group 30–39. The distribution of the second variable, education, is shown in the second–from–top left subgraph of Figure 3.3. Given this distribution, categories 4, 6 and 8 are most likely to be assigned to the simperson for the education variable. However, the correlations (shown in the third left subfigure of Figure 3.3) between the age category 5 and education variable reveal positive correlations for categories 1 and 4, and a negative correlation for category 8 for the education variable. After normalizing (shifting to mean 1) the correlations, the original distribution of the education variable is skewed by pair–wise multiplying the raw counts of categories of the education variable and the normalized correlations for the education variable given that the age category of the simperson is 5. This skewed distribution is shown in the bottom left subgraph of Figure 3.3 and is used for sampling the education variable, resulting in the education category 4, vocational training. Values for further variables are drawn from skewed distributions that take into account the categories for the previously drawn variables, by skewing the distribution of the current variable by the average of the normalized correlations for the so far drawn categories. This process is shown from top to bottom on the right subfigures of Figure 3.3, where given that the age category a the simperson is 5 and the education category is 4 for the third variable, employment state, the category 11 is drawn.

3.5.3 Assigning Simpersons to Work Places / Schools

Activities can be divided into two groups: *free time activities* and *mandatory activities*. While the notion of “mandatory” activity may differ from person to person, for the purposes of simulation, ST–ACTS considers going to *school* and *work* as mandatory activities. The rest of the activities in ST–ACTS are considered free time activities.

With respect to mandatory activities, simpersons can be divided into three groups: retired, worker, and student. For the retired simpersons, it can be assumed that they enjoy the fruits of a hard–working life and have no mandatory activities. Consequently, they spend the majority of the time either at home or performing free time activities. The following paragraphs describe the methods in ST–ACTS (and their usage of the base data) for assigning simpersons in the worker and student groups to their work places and schools respectively.

Assigning Worker Simpersons to Work Places: Simpersons in the worker group are assigned to *work places* in two steps. In the first step, given the *home parish* and employment branch of the simperson, the parish–to–parish commuting probabilities, and the spatial distribution of businesses in branches, a *work parish* is assigned to the simperson. In the second step, given the employment branch that the simperson works in, businesses in the same branch that are located in the work parish of the simperson are retrieved from bizmark$^{\text{TM}}$. Finally, proportional to the number of employees that work in the selected businesses, the simperson is probabilistically assigned to one of the businesses / work places.

Assigning Student Simpersons to Schools: Simpersons in the student group are assigned to schools in two steps. In the first step, depending on the age group of the simperson, he or she is assigned to either one of the four educational institution types, or is assigned to be “not in school” and hence is considered to a member of the worker group. In the second step, educational institutions of the simpersons’s educational institution type are retrieved from bizmark$^{\text{TM}}$, and the simperson is assigned to the institution that is closest to the simperson's home[2]. The following paragraph explains the first of these steps in more detail.

Simpersons in the student group can be divided into four subgroups based on which one, if any, of the four educational institution types they attend: kindergarten, primary school, secondary school, or college / university. As described above, each simperson below age 30 is assigned to one of the four age groups: $[0, 11]$, $[12, 16]$, $[17, 22]$, and $[23, 29]$. Assuming all simpersons up to age 5 or 6 go to kindergarten (or daycare centers), simpersons in the $[0, 11]$ age group are assigned with equal likelihood to either a kindergarten, or a primary school. For each of the remaining three age groups, based on information obtained from Statistics Denmark [85], the probabilities of attending one of the four education institution types are derived, which are shown in the table:

	[12-16]	[17-22]	[23-29]
primary school	0.9198	0.0235	0.0002
secondary school	0.0654	0.4639	0.0552
college / university	0.0000	0.1194	0.2365
not in school	0.0148	0.3933	0.7081

Then, given the age group of the simperson and the corresponding probabilities, the simperson is assigned to either one of the three educational institution types, or to be "not in school" and is considered to be a member of the worker group.

3.5.4 Daily Activity Probabilities

A subset of the GallupPC® consumer survey questions, described in Section 3.4.4 represent activities that require the movement of the consumer. Some of these activities are shown on the y–axis of Figure 3.4. To preserve space and clarity, the following, additional activities are included in the model, but are excluded from the figure: art exhibition, church, pop/rock concert, museum, post office, theater, solarium, hairdresser, and shopping. The shopping activity is further subdivided into 22 subtypes of shopping that are tied to a particular brand or type of store.

Using the geo–demographic parts of the surveys, each survey subject is assigned to one of the 29 conzoom® types. To derive a single indicator for how likely a given conzoom® type is to perform a given activity, the answers to the re–phrased time–frequency questions are normalized and averaged as follows. First, every time–frequency interval for an activity is normalized to represent the probability of performing the given activity on an average day. For example, a subject's positive reply to the question "Do you perform activity a n times during a period Δt?" equivalently

[2]The Danish public school system is controlled by the municipalities, which assign students to educational institutions that are nearby. Locations of these institutions are carefully planned to meet the needs of the population.

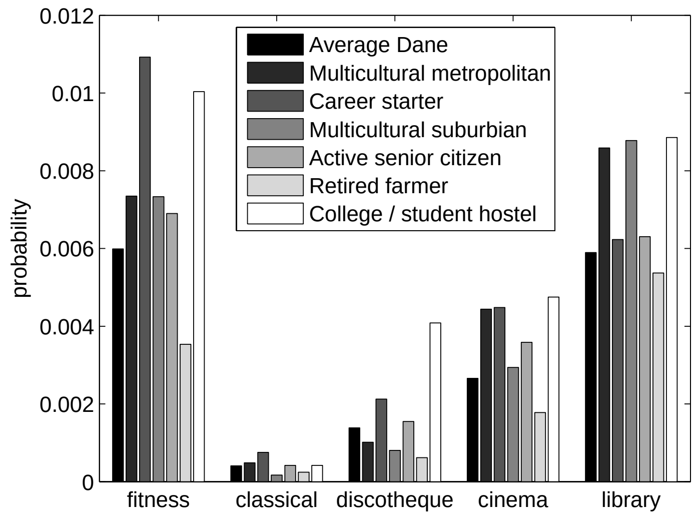


Figure 3.4: Sample Daily Activity Probabilities.

means that the probability of that subject to perform activity a on any given day is $P(a) = n/day(\Delta t)$, where day is a function that returns the number of days in period Δt. $P(a)$ is equivalently referred to as the *Daily Activity Probability* (DAP) of activity a. Second, these daily activity probabilities of individual subjects of a given conzoom® type are averaged. Figure 3.4 shows a sample of these daily activity probabilities for a subset of the conzoom® types. From the figure it can be seen, for example, that a college student is most likely to go to a library, a cinema, a discotheque, or a fitness center; while a retired farmer is the least likely to perform these activities. Since the figure has the same probability scale, it also reveals that, depending on type, going to a fitness center is about a 7 to 22 times more frequent or popular activity as going to classical concerts. As mentioned before, ST–ACTS includes the daily activity probabilities of 35 activities for 29 conzoom® types.

3.5.5 Activity Simulation with Spatio–Temporal Constraints

A simple, random, discrete event activity simulator can be constructed as follows. At every time step, a random subset of the simpersons is chosen to perform an activity. Then, for each selected simperson, given his/her conzoom® type and the associated daily activity probabilities, an activity is assigned. Then, each selected simperson is moved to the closest facility, where his/her assigned activity can be performed. This simple simulator does not model several spatio–temporal constraints on the activities. In the following, these constraints are discussed, and for each, the proposed modelling solution that ST–ACTS implements is presented.

Temporal Activity Constraint: Certain activities are more likely to be performed during specific periods than others. For example, people in the work force tend to leave their homes for work at the beginning of a workday. Consequently, the same people are less likely to go to a discotheque, which is presumably closed, during the same period. To model the *Temporal Activity Constraint* (TAC), ST–ACTS allows the user to define for each of the three population groups the probabilities for each of the activities for every hour of every day of the week. These probabilities are used to limit the ability of the simperson to perform certain activities during certain time periods. They are not to be confused with the conzoom$^{®}$ type dependent *daily activity probabilities*, which encode the activity preference of each type. Through the TACs ST–ACTS allows the modelling of opening hours, and to some degree sequential patterns. The TACs of an activity are defined by a 7 by 24 matrix, where columns represent hours of the day, and rows represent days of the week.

Activity Duration Constraint: Not all activities take the same amount of time. For example people usually work 6-10 hours, spend about 2 hours in a cinema, and 30 minutes in a grocery store. To model this, from the starting timestamp of an activity a that is assigned to a simperson s, s becomes *occupied* for $\delta_{occupied}(a)$ time steps. During this period s is not assigned any other activities. In ST–ACTS, *Activity Duration Constraint* (ADC) for each activity are probabilistically drawn from the user–defined activity duration distributions, which is normally distributed with mean $\mu_{\delta_{occupied}(a)}$ and variance $\sigma_{\delta_{occupied}}(a)$.

Minimum Elapsed Time Between Activity Repetition Constraint: While people prefer some activities over others, it is very unlikely that they would repeat the same, even if preferred, activity many times, one–after–the–other within a short period. For example, it is very unlikely, that even a very active simperson, right after finishing his workout at the fitness center, decides to go to a fitness center again. This constraint is modelled in ST–ACTS through the user–defined $\delta_{elapsed}(a)$, activity–dependent *Minimum Elapsed Time Constraint* (METC). The constraint is enforced by maintaining a recent history of activities for each simperson and validating newly drawn activities against it.

Maximum Distance Constraint: For most activities there is a *maximum distance* a person is willing to travel. This maximum distance represents a spatial constraint on the activities that a simperson s will perform, given the current location of s and the locations of facilities, where a selected activity a can be performed. Hence, during the simulation if there is no suitable facility for a within maximum distance of the current location of s, the activity is considered invalid for s, and s becomes idle. The *Maximum Distance Constraint* (MDC) is controlled by a user–defined, activity–dependent parameter in ST–ACTS.

```
(0) //geo–demographic data (conzoom®): $D$
(0) //population movement data (mobidk™): $M$
(0) //business data (bizmark™): $B$
(1) procedure ST–ACTS($T$,$\Delta T$,$DAP$,$TAC$,$ADC$,$METC$,$MDC$)
(2)     $s.dem \leftarrow$ drawDemographicVariables($D$)
(3)     $s.work \leftarrow$ simpsToWork($s$,$B$,$M$)
(4)     $s.acts \leftarrow$ initSimpsActs($s$,t=1)
(5)     for $t$ = 1...$T$
(6)         $free \leftarrow$ unoccupiedSimps($s.acts$,$t$)
(7)         $a \leftarrow$ validActsToFreeSimps($s$,$DAP$,$TAC$,$METC$,$t$)
(8)         $[f, d] \leftarrow$ facilitiesForActs($a$,$s$,$MDC$,$B$)
(9)         $\delta_{occupied} \leftarrow$ durationsOfActs($a$,$ADC$)
(10)        $\delta_{trans} \leftarrow$ transitionTimes($d$,$speed(d)$)
(11)        $s.acts \leftarrow$ updateSimps($a$,$f.loc$,$\delta_{occupied}$,$\delta_{trans}$,$t$)
```

Figure 3.5: Discrete Event Simulation in ST–ACTS.

Physical Mobility Constraint: To move from one location to another takes time. While detailed simulation of this movement is not an objective of ST–ACTS, basic physical mobility constraints are modelled. After a facility f for an activity a is selected for a simperson s, s is moved after δ_{trans} time steps to the new location. δ_{trans} is calculated based on the Euclidian distance d in km between the current location of s and the location of facility f, assuming a constant speed. This constant speed, in km/h, is probabilistically drawn from the distribution $speed(d) = \max(5, N(3d, d^2))$. $speed(d)$ assigns lower speeds to shorter, and higher speeds (with larger variance) to longer distances. It, to some extent, captures common modes of transportation, i.e., people tend to walk on shorter trips, use public transportation or bicycle on slightly longer trips, and use a car or commuting train on even longer trips.

3.5.6 Discrete Event Simulation

Using the conceptual building blocks presented so far, the discrete event simulation performed in ST–ACTS can be summarized as shown in Figure 3.5. The first three comments indicate that named data sets are used in the simulation, but are not user–defined parameters of it. Arguments to ST–ACTS, shown on line 1, are the user–defined parameters that have been described in the previous paragraphs. On line 2 demographic variables are assigned to simpersons based on skewed variable distributions. On line 3 simpersons are assigned to work places and schools. On line 4, at time step $t = 1$ (Monday, 00:00) all simpersons are initialized to be at "home"

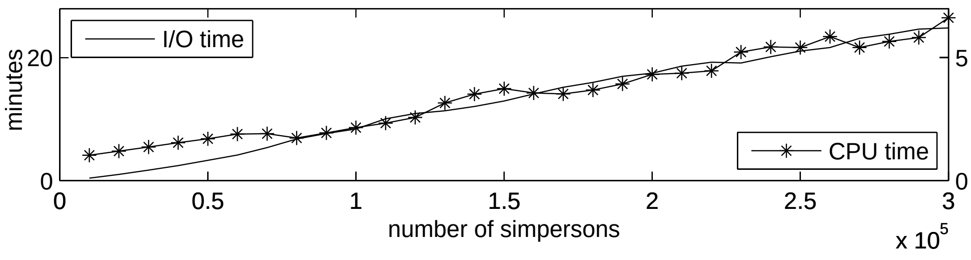


Figure 3.6: CPU and I/O Times for Simulations.

doing activity “home to stay” until the early morning hours. Following these preprocessing steps, at every time step t, on line 6, currently unoccupied (free) simpersons are found. Then, on line 7 for each free simperson a valid action is found according to the daily activity probabilities (DAP) of actions for the conzoom® type of the simperson. Actions are valid, if they both meet the temporal activity constraint (TAC) and the minimum elapsed time constraint (METC). On line 8 valid facilities are found for these valid activities. Facilities are valid if they meet the maximum distance constraint (MDC). On line 9, activity durations are drawn that meet the activity duration constraint (ADC). On line 10, according to the distances to the assigned activities, transition times are calculated. Finally, on line 11, information about the newly assigned activities are stored and the activity histories are updated for the affected simpersons.

3.6 Evaluation of the Simulation

ST–ACTS was implemented and tested in MATLAB running on Windows XP on a 3.6GHz Pentium 4 processor with 1.5 GB main memory. The geographical extent of ST–ACTS was restricted to the municipalities of Copenhagen and Frederiksberg in Denmark. In this extent, the number of simpersons is 590,050 (178,826 retired, 268,615 workers, and 142,609 students), the number of working places is 1,264,129 in 193,299 businesses, and the number of facilities is 10,544. Simulation experiments were performed for a time step length of $\Delta T = 5$ minutes. To test the performance of ST–ACTS, in all experiments “strict” TACs were set on the two most likely activities, go “home to visit” and go “home to stay”. TACs of other activities were set to model opening hours of corresponding facilities. As a result a simperson performs on average 9.6 ± 3.2 activities per day.

To evaluate the effectiveness of ST–ACTS, simulations were performed for varying sizes of randomly selected subsets of simpersons during the course of a single day (24 hours). Figure 3.6 shows both the CPU times (right y–axis) and the I/O time for logging the events (left y–axis). Both of these quantities scale approximately linearly with the number of simpersons. In short, the simulation is fast and scales well.

In a larger experiment activities of the total population for the course of a full week have been simulated. The table below shows the output of ST–ACTS for a cosmopolitan type simperson during the course of a day.

a.begin	a.loc(x)	a.loc(y)	a.end	a.name
8:35	722941	6172634	15:50	work / school
17:05	720408	6173933	17:45	Fakta
18:55	721350	6177550	20:20	home to visit
20:45	723555	6175390	21:10	solarium
21:50	723483	6175299	23:30	cinema
23:40	721350	6177550	8:25	home to stay

The simulation, without logging the individual events and only keeping statistics about activities, took 98 minutes. To evaluate the validity of ST–ACTS, the gathered statistics have been analyzed. Due to space limitation, only some results of this analysis are discussed in detail, while others are only summarized.

To evaluate ST–ACTS's ability to generate the correct distribution of activities, the input DAPs have been compared to the simulated DAPs, shown in Figure 3.7. While, due to the previously mentioned "strict" TACs, the simulated DAPs are about 4 times higher than the input DAPs, the relative simulated DAPs among activities is similar to the input DAPs. By using less "strict" TACs, i.e.: allowing simpersons to go home earlier after work, the scale of simulated DAPs match that of the input DAPs. Differences in the relative DAPs can be explained by the effects of spatio–temporal constraints on activities.

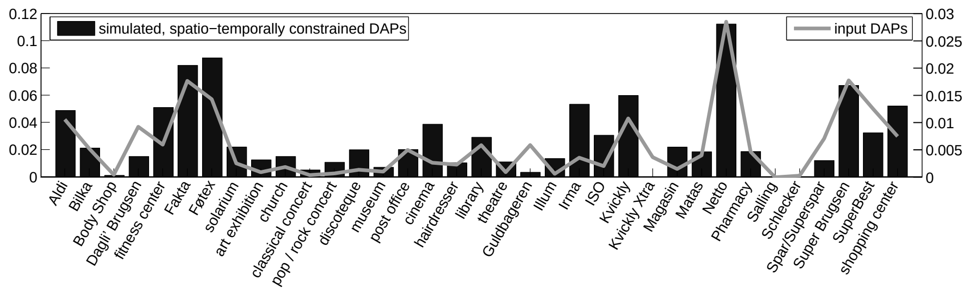


Figure 3.7: Input and Simulated DAPs.

To evaluate ST–ACTS's ability to control temporal constraints on activities, counts for each assigned activity for every hour–of–day and day–of–week were maintained. Figure 3.8 shows the average number of assigned activities for each hour–of–day averaged over the days–of–week. Due to the large variation in frequency counts for

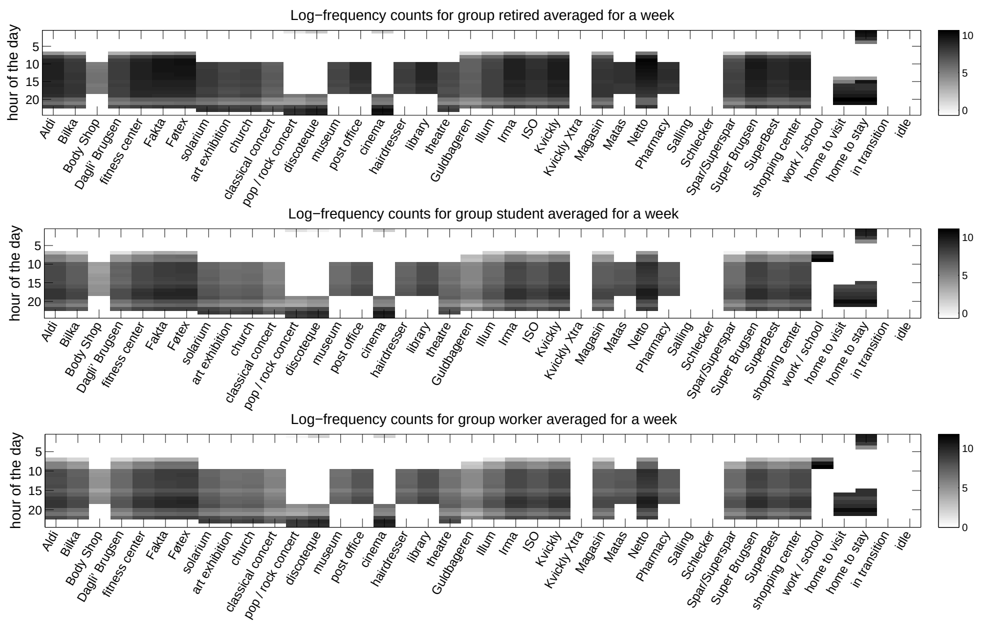


Figure 3.8: Validity of ST–ACTS in Terms of TACs.

different activities in different periods of the day, the base 2 logarithm of frequency counts are shown. From the figure it can be seen that certain groups perform certain activities at certain times of the day more frequently than other groups. For example, it can be seen that the retired group is more likely to perform activities during working hours, simply because they are free to do so. Opening and closing times of facilities is also controlled by the parameters. For example, no one goes to discotheques during the day, and no one goes to shopping centers in the middle of the night.

To evaluate ST–ACTS's ability to control spatial constraints on activities, the daily distance travelled to work by an average simperson (2.7 ± 2.3 km) was compared to the total daily distance travelled by an average simperson (8.3 ± 3.6 km). While for the same numbers no ground truth was available to evaluate against, considering the average 9.6 activities per day the numbers seem reasonable. The simulated data has also been verified that no trips violate the activity–dependent maximum distance criteria.

3.7 Conclusions and Future Work

Realistic models that simulate the spatio–temporal activities of users, and hence the distribution of moving objects, are essential to facilitate the development of adequate spatio–temporal data management and data mining techniques. In this paper, ST–ACTS, the first of such simulators is presented. Experimental results show that, using a number of real–world geo–statistical data sources and intuitive principles, ST–ACTS is able to effectively generate realistic spatio–temporal activity data. It is also demonstrated that the generated data has the same characteristics as it is defined by the user–controllable model parameters. ST–ACTS has been implemented in MATLAB and is available for research purposes.

While the correspondence between the characteristics of the generated data and the model parameters is demonstrated, the accuracy of the simulation has to be necessarily affected by the limited modelling of physical aspects of mobility. Hence in future work, integrating the output of ST–ACTS as an input to sophisticated network–based moving object simulation as in [8] is planned. Such a more complex simulator will provide synthetic data sets that can aid the development in telematics, intelligent transportation systems, and location–based services.

Chapter 4

Mining Long, Sharable Patterns in Trajectories of Moving Objects

The efficient analysis of spatio–temporal data, generated by moving objects, is an essential requirement for intelligent location–based services. Spatio–temporal rules can be found by constructing spatio–temporal baskets, from which traditional association rule mining methods can discover spatio–temporal rules. When the items in the baskets are spatio–temporal identifiers and are derived from trajectories of moving objects, the discovered rules represent frequently travelled routes. For some applications, e.g., an intelligent ride–sharing application, these frequent routes are only interesting if they are long and sharable, i.e., can potentially be shared by several users. This paper presents a database projection based method for efficiently extracting such long, sharable frequent routes. The method prunes the search space by making use of the minimum length and sharable requirements and avoids the generation of the exponential number of sub–routes of long routes. Considering alternative modelling options for trajectories, leads to the development of two effective variants of the method. SQL–based implementations are described, and extensive experiments on both real life– and large–scale synthetic data show the effectiveness of the method and its variants.

4.1 Introduction

In recent years Global Positioning Systems (GPS) have become increasingly available and accurate in mobile devices. As a result large amounts of spatio–temporal data is being generated by users of such mobile devices, referred to as *moving ob-*

jects in the following. Trajectories of moving objects, or trajectories for short, contain regularities or patterns. For example, a person tends to drive almost every weekday to work approximately at the same time using the same route. The benefits of finding such regularities or patterns is many–fold. First, such patterns can help the efficient management of trajectories. Second, they can be used to facilitate various Location–Based Services (LBS). One LBS example is an intelligent rideshare application, which finds sharable routes for a set of commuters and suggests rideshare possibilities to them, is considered. Such a rideshare application can be one possible solution to the ever increasing congestion problems of urban transportation networks.

Patterns in trajectories for an intelligent rideshare application are only interesting if those patterns are sharable by multiple commuters, are reoccurring frequently, and are worthwhile pursuing, i.e., are long enough for the savings to compensate for the coordination efforts. The discovery of Long, Sharable Patterns (LSP) in trajectories is difficult for several reasons. Patterns do not usually exist along the whole trajectory. As a example, consider two commuters A and B living in the same area of town, leaving for work approximately the same time, and working in the same part of town. Given the underlying road network and traffic conditions, for a given support threshold the middle part of the trips of the two commuters may be frequent, the initial and final parts may not. In recent work [30] a general problem transformation method, called *pivoting*, was proposed for the analysis of spatio–temporal data. Pivoting is the process of grouping a set of records based on a set of attributes and assigning the values of likely another set of attributes to groups or baskets. Pivoting applied to spatio–temporal data allows the construction of spatio–temporal baskets, which can be mined with traditional association rule mining algorithms. When the items in the baskets are spatio–temporal identifiers and are derived from trajectories, the discovered rules represent frequently travelled routes. While there exist several efficient association rule mining methods [40], the straight–forward application of these algorithms to spatio–temporal baskets representing trajectories is infeasible for two reasons. First, all sub–patterns of frequent patterns are also frequent, but not interesting, as longer patterns are preferred. Second, the support criterion used in association rule mining algorithms is inadequate for a rideshare application, i.e., a frequent itemset representing a frequent trajectory pattern, may be supported by a single commuter on many occasions and hence presents no rideshare opportunity.

In this paper, to overcome the above difficulties of finding LSPs in trajectories, a novel method is given. According to a new support criterion, the proposed method first efficiently filters the trajectories to contain only sub–trajectories that are frequent. Next, it removes trajectories that do not meet the minimum length criterion. Then it alternates two steps until there are undiscovered LSPs. The first step entails the discovery of a LSP. The second step entails the filtering of trajectories by the previously discovered pattern. An advantage of the proposed method is the ease of implementation in commercial Relational Database Management Systems (RDBMSes). To demonstrate this, a SQL–based implementation is described. Considering the global modelling of trajectories, leads to the development of two other effective variants of the proposed method. The effectiveness of the method and its variants are demonstrated on the publicly available INFATI data, which contains trajectories of cars driving on a road network, and on a number of large–scale synthetic data sets.

The herein presented work is novel in several aspects. It is the first to consider the problem of mining LSPs in trajectories. It describes a novel transformation, and the relationship between the problem of mining LSPs in trajectories and mining frequent itemsets. Finally, it describes an effective method with a simple SQL–implementation to mine such LSPs in trajectories.

The remainder of the paper is organized as follows. Section 4.2 reviews related work. Section 4.3 describes the transformation, the use of the framework in frequent itemset mining, and formally defines the task of mining LSPs in trajectories. Section 4.4 discusses a naïve method for mining LSPs and points out its shortcomings. Section 4.5 describes the proposed algorithm and a SQL–based implementation for mining LSPs. Section 4.6 presents alternative modelling of trajectories and derives variants of the proposed method based on these modelling options. Section 4.7 presents detailed experimental results. Finally, Section 4.8 concludes and points to future research.

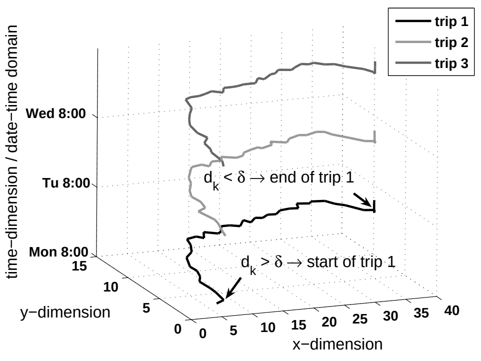
4.2 Related Work

Frequent pattern mining is a core field in data mining research. Since the first solution to the problem of frequent itemset mining [1, 2], various specialized in–memory data structures have been proposed to improve the mining efficiency, see [40] for an overview. It has been recognized that the set of all frequent itemsets is too large for analytical purposes and the information they contain is redundant. To remedy this, two modification to the task have been proposed: mining of Closed Frequent Itemsets (CFI) and mining of maximal frequent itemsets. A frequent itemset X is *closed* if no itemset Y exists with the same support as X such that $X \subset Y$. A frequent itemset X is *maximal* if no frequent itemset Y exists such that $X \subset Y$. Prominent methods that efficiently exploit these modifications to the problem are MAFIA [9], GenMax [42], CLOSET [76], CLOSET(+) [98], and CHARM [104]. Later in the paper, a relationship between the problems of mining LSPs in trajectories and mining CFIs are described. While CFI mining methods can be modified to find the desired solution that meets the *sharable* criterion, they employ complex data structures and their implementation is quite involved; hence their augmentation is difficult. In particular, a projection–based CFI mining algorithm that employs an in–memory FP–tree to represent itemsets, would need to be modified at every node to maintain a set of distinct objects at that have transactions associated with them that support the itemset that is represented by the node. In comparison, the herein presented method –building on work presented in [84]–exploits the power of commercial RDBMSs, yielding a simple, but effective solution.

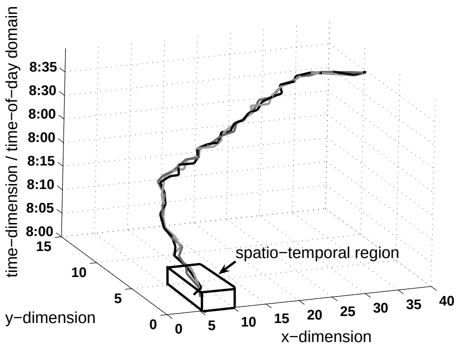
Since trajectories are temporally ordered sequences of locations, sequential pattern mining [3] naturally comes to mind. However, a straight forward interpretation of trips as transactions and application of a state–of–the–art closed frequent sequential pattern mining algorithm [103] does not yield the desired solution, since in this case sequences of frequent sub–trajectories would be found. Furthermore, since the trajectories can contain hundreds of items, closedness checking of frequent itemsets even for prominent methods would be computationally expensive. Interpreting single elements of trajectories as transactions and applying closed sequential pattern mining could find frequent sub–trajectories. However a number of problems arise. First, to meet the sharable criterion, the in–memory data structures would need similar, non–trivial augmentation as described above. Second, since patterns in trajectories could be extremely long, even state–of–the–art sequential mining methods [95, 103] would have a difficulties handling patterns of such lengths. Third, patterns in trajectories repeat themselves, which cannot be handled by traditional sequential pattern mining algorithms. The extraction of spatio–temporal periodic patterns from trajectories is studied in [70], where a bottom–up, level–wise, and a faster top–down mining algorithm is presented. Although the technique is effective, the patterns found are within the trajectory of a single moving object. In comparison, the herein presented method effectively discovers long, sharable, periodic patterns.

Moving objects databases are particular cases of spatio–temporal databases that represent and manage changes related to the movement of objects. A necessary component to such databases are specialized spatio–temporal indices such as the Spatio–Temporal R–tree (STR–tree) and Trajectory–Bundle tree (TB–tree) [55]. An STR–tree organizes line segments of a trajectory according to both their spatial properties and the trajectories they belong to, while a TB–tree only preserves trajectories. If trajectories are projected to the time–of–day domain, STR–tree index values on the projected trajectories could be used as an alternative representation of trajectories. While this approach would reduce the size of the problem of mining LSPs in trajectories, it would not solve it. In comparison, the herein presented method solves the problem of mining LSPs in trajectories, which is orthogonal, but not unrelated to indexing of trajectories.

In [97] a way to effectively retrieve trajectories in the presence of noise is presented. Similarity functions, based on the longest sharable subsequence, are defined, facilitating an intuitive notion of similarity between trajectories. While such an efficient similarity search between the trajectories will discover similar trajectories, the usefulness of this similarity in terms of length and support would not be explicit.



(a) Identification of Trips in Raw Trajectories.



(b) Time–Of–Day Projection and Spatio–Temporal Region Substitution.

Figure 4.1: From Trajectories to Transactions.

In comparison, there herein proposed method returns only patterns that meet the user–specified support and length constraints. Furthermore, the trajectory patterns returned by the proposed method are explicit, as opposed to the only implicit patterns contained in similar trajectories.

4.3 Long, Sharable Patterns in Trajectories

The following section describes a novel transformation of raw trajectories. This transformation allows (1) the formulation of the problem of mining LSPs in trajectories in a framework similar to that used in frequent itemset mining, (2) to establish a relationship between the two problems.

4.3.1 From Trajectories to Transactions

The proposed transformation of raw trajectories consists of three steps: identification of trips, projection of the temporal dimension, and spatio–temporal region substitution. It is assumed that locations of moving objects are sampled over a long history. That is, a raw trajectory is a long sequence of (x, y, t) measurements at regular time intervals.

Identification of Trips

A trip is a temporally consecutive set or sequence of measurements such that for any measurement m_i in the sequence, the sum of spatial displacement during the k measurements immediately following m_i, denoted d_k, is larger than some user–defined displacement, δ. Trips can be identified in a straight–forward manner by linearly scanning through a trajectory, and calculating d_k using a look–ahead window of k measurements. That is, scanning through the total trajectory from the beginning, the first measurement for which $d_k > \delta$, signals the beginning of the first trip. Consecutive measurements are part of this trip until a measurement is reached for which $d_k \leq \delta$, which signals the end of the first trajectory. Trips following the first trip are detected in the same fashion from the remaining part of the total trajectory. Figure 4.1(a) shows three example trips that are derived from the total trajectory of one moving object.

Projection of the Temporal Dimension

Since frequent patterns within a single object's trajectory are expected to repeat themselves daily, the temporal dimension of the so identified trips is projected down to the time–of–day domain. This projection is essential to discover the daily periodic nature of patterns in trajectories. Mining patterns with other periodicity can be facilitated by projections of the temporal domain to appropriate finer, or coarser levels of granularity. Finer levels of granularity can be used to detect patterns with shorter periodicity. For example, a delivery person might use a different route depending on the time–of–hour knowing that at the given time of the hour certain traffic conditions arise, which make an otherwise optimal delivery route sub–optimal. The detection of these patterns in delivery routes requires the projection of the temporal dimension to the time–of–hour domain. Conversely, coarser levels of granularity can be used to detect patterns with longer periodicity. For example, a person might visit his bank only at the end of pay periods. The detection of this pattern requires the projection of the temporal dimension to the day–of–month domain. Finally, to discover the pattern that the above mentioned person makes these visits to his bank Saturday mornings following the end of pay periods, requires the projection of the temporal domain to a combination of the day–of–month, the day–of–week, and the part–of–day domains. Performing different projections is part of the inherently iterative and only semi–automatic process of doing data mining when the exact format of the patterns searched for is not known beforehand. Figure 4.1(b) shows the projection of the temporal dimension to the time–of–day domain for the three trips identified in Figure 4.1(a). Since the projection of a single database record is a constant time operation, the total processing time of this transformation step is optimal and linear in the number of database records.

Spatio–Temporal Generalization and Substitution

Trajectories are noisy. One source of this noise is due to imprecise GPS measurements. From the point of view of patterns in such trajectories, slight deviation of trajectories from the patterns can be viewed as noise. Examples of such deviations could be due to a few minute delay, or to the usage of different lanes on the route. Hence, while a person might be driving from home to work at approximately the same time of day using approximately the same route, the chance of two identi-

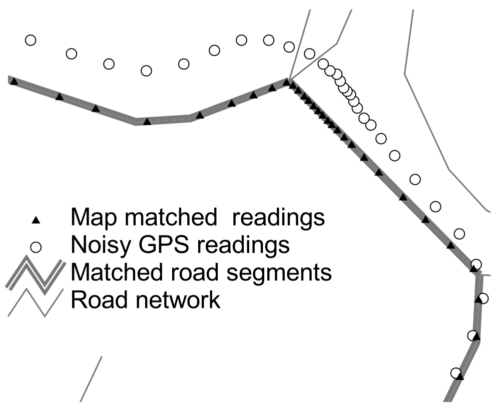


Figure 4.2: Process / Outcome of Map Matching.

cal trajectories is highly unlikely. Consequently, patterns in raw trajectories are few and certainly not long. Thus, patterns have to be mined in trajectories that are represented in a generalized way, yielding general patterns in trajectories. There are at least two different approaches to achieve this generalization of trajectories: region–based spatio–temporal generalization and road network based spatio–temporal generalization.

In the region–based spatio–temporal generalization approach individual (x, y, t) measurements of a trajectory are discretized and mapped to the spatio–temporal regions they fall into. Thus, a generalized trajectory is constructed by substituting (x, y, t) measurements with the spatio–temporal regions they map to. If within a trajectory multiple (x, y, t) measurements map to the same spatio–temporal region, they are substituted with a single instance of the corresponding spatio–temporal region. The box in Figure 4.1(b) represents such a spatio–temporal region. Since region–based spatio–temporal substitution of a single database record can be achieved using simple arithmetics from the spatial and temporal coordinates, the processing time of this transformation step is optimal and linear in the number of database records.

In the road network based spatio–temporal generalization approach, objects are assumed to be moving on a road network and coordinates of individual (x, y, t) measurements of a trajectory are matched to road segments of the underlying road network. The process of matching trajectories to road segments is called map matching and has been studied extensively in the recent past. Figure 4.2 shows the outcome of map matching, where noisy GPS readings are “snapped” to the most likely road segments the object was actually moving on. In general, two map matching approaches exist: on–line and off–line map matching. In on–line map matching, the noisy GPS readings are positioned onto the road network taking into account the past readings and the topology of the road network. In off–line map matching, the positioning of

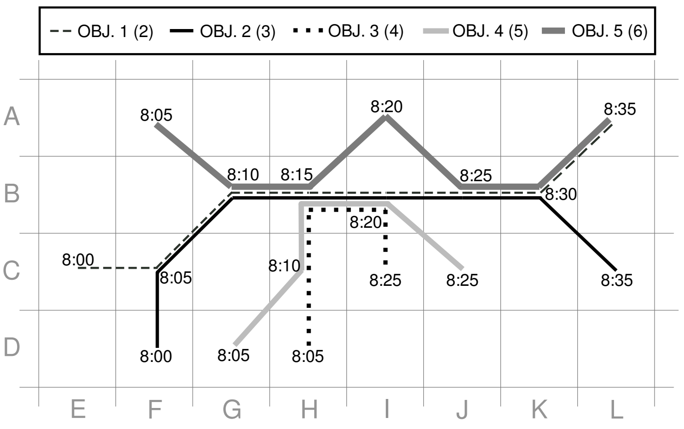


Figure 4.3: Illustration of the Sample Trajectory DB.

GPS readings onto the road network is performed with some delay, hence methods can take into consideration “future” measurements, which generally increases the matching accuracy and reduces the necessary computation. In [79] a summary of different on–line and off–line map matching algorithms is provided and disadvantages of each approach is described. Once the coordinates of trajectories are map matched, the individual (x, y, t) measurements of a trajectory are discretized and mapped to the spatio–temporal identifiers composed of a combination of road segment identifiers and temporal intervals. If within a trajectory multiple (x, y, t) measurements map to the same spatio–temporal identifier, they are substituted with a single instance of the corresponding spatio–temporal identifier. The map matching task can be performed in a distributed fashion by on–board navigation units of the moving objects. Based on the map matching results the road network based spatio–temporal substitution of a single database record can be achieved in constant time using simple arithmetics from the temporal values, hence the processing time of this transformation step is optimal and linear in the number of database records.

4.3.2 Example Trajectory Database

Figure 4.3 visualizes a sample trajectory database. It shows the trajectories of trips of 5 moving objects, which were derived using the three transformation steps described in Section 4.3.1. For clarity, the temporal dimension is projected down to the 2D–plane. Spatio–temporal regions are defined by the square cells and a five minute interval centered around time instances written inside the square. Each connected line represents specific trips of a particular object. The number of times that trip was performed by the object is represented in the width of the line, and is also written in parenthesis next to the object name in the legend. For example, the trip trajectory associated with object 3 was performed 4 times by the object. The object was in spatial regions HD, HC, HB, IB, and IC during time intervals $8{:}05 \pm 2.5$ minutes, $8{:}10 \pm 2.5$ minutes, $8{:}15 \pm 2.5$ minutes, $8{:}20 \pm 2.5$ minutes, and $8{:}25 \pm 2.5$ minutes, respectively. In the following a spatio–temporal region will be referred to by its concatenated values of the cell identifiers along the x– and y–axis, and the corresponding time instance denoting the center of the time interval of the spatio–temporal region. Hence, trips associated with object 3 will be denoted by the a sequence $\{$HD8:05, HC8:10, HB8:15, IB8:20, IC8:25$\}$. Furthermore, the trajectory database T is assumed to be in a relational format with schema $\langle oid, tid, item\rangle$, where $item$ is a single item, that is part of the transaction tid associated with object oid. Hence, each of the four trips of object 3 is represented by 5 unique rows in T.

4.3.3 Problem Statement

After performing the three above transformation steps, the data set can be represented in a database T containing tuples $\langle oid, tid, s\rangle$, where oid is an object identifier, tid is a trip identifier, and s is a sequence of spatio–temporal region identifiers. Since spatio–temporal region identifiers contain a temporal component, the sequence s can, without loss of information, be represented as a **set** of spatio–temporal region identifiers. Conforming to the naming convention used in the frequent itemset mining framework, a spatio–temporal region identifier will be equivalently referred to as an *item*, and a sequence of spatio–temporal region identifiers will be equivalently referred to as a *transaction*. Let X be a set of items, called an *itemset*. A transaction t *satisfies* an itemset X iff $X \subseteq t$. Let ST_X denote the set of transactions that satisfy X. The following definitions are emphasized to point out the differences between the frequent itemset mining framework and the one established here.

Definition 1 *The* $\mathbf{n}$***–support of an itemset*** X *in* T*, denoted as* $X.supp(n)$*, is defined as the number of transactions in* ST_X *if the number of distinct* oid*s associated with the transactions in* ST_X *is greater than or equal to* n*, and* 0 *otherwise. The* $\mathbf{n}$***–support of an item*** i *in* T*, denoted as* $i.supp(n)$*, is equal to the* n*–support of the itemset that contains only* i*.*

Definition 2 *The* ***length of an itemset*** X*, denoted as* $|X|$*, is defined as the number of items in* X*.*

Definition 3 *An itemset* X *is* $\mathbf{n}$***–frequent*** *in* T *if* $X.supp(n) \geq MinSupp$*, and* X *is* ***long*** *if* $|X| \geq MinLength$*, where* $MinLength$*,* $MinSupp$*, and* n *are user–defined values.*

Definition 4 *An itemset X is **n–closed** if there exists no itemset Y such that $X \subset Y$ and $X.supp(n) = Y.supp(n)$.*

The task of mining LSPs in trajectories can be defined as finding all long, n–closed, n–frequent itemsets. Itemsets that meet these requirements are also referred to as LSPs, or just patterns.

4.4 Naïve Approach to LSP Mining

The here presented naïve approach uses the convenience and efficiency of an RDBMS. For ease of exposure, consider the problem of finding long sub–trajectories in trajectories. Meeting the unique support requirement of the original task does not substantially change the complexity of method to be described, but eases the description and analysis of it. Finding pairs of trajectories that have long sub–trajectories can be efficiently solved using 2–way self–joins. Generally, K–way self–joins can be used to find groups of K trajectories that share parts of their trajectories. Consequently, to discover all long sub–trajectories, self–joins could be used in an iterative way, first discovering pairs, then triples, and so on, finally leading to groups of K trajectories that have long, sharable sub–trajectories. A solution based on self–joins has several drawbacks. As the number of trajectories is increasing, the maximum size of groups of trajectories that have a long sub–trajectory is expected to increase as well. Naturally, as this maximum group size is increasing, the number of self–joins that need to be performed is increasing as well. Although the sizes of the intermediate result sets of the consecutive joins that compose the K–way self–join are non–increasing with every join operation, and hence the required time to compute these joins is also non–increasing, the described K–way self–join method is inefficient. In fact its worst case running time is exponential in K, which is illustrated in the following. Consider a set of K trajectories that have a long sharable sub–trajectory in them. The iterative K–way self–join method in the first iteration, discovers all pairs of these K trajectories. Then in the next step, it discovers all groups of 3 of these trajectories, alternatively leading to the discovery of 2^K subsets of these K trajectories. This is clearly inefficient from a computational point of view not to mention the complexity it introduces in the discovered results. Since the ultimate goal of an intelligent rideshare application is the optimal coordination of possible rideshare opportunities of a set of commuters, the exponentially large number of discovered patterns is clearly a disadvantage from the user's point of view.

4.5 Projection–Based LSP Mining

Now let us turn to the description of the proposed method for mining LSPs in trajectories. This description is based on a number of observations, each of which is associated with a particular step in the method. These observations are also stated as lemmas, and their corresponding proofs show the correctness and completeness of the method. To demonstrate the simplicity of the implementation in a RDBMS, for each step a simple SQL–statement is given. The effect of each step is also illustrated on the previously introduced sample trajectory database assuming $MinLength = 4$, $MinSupp = 2$, and $n = 2$.

4.5.1 STEP 1: Filtering of Infrequent Items

Items, i.e., spatio–temporal regions that are not frequent in T cannot be part of a LSP. Hence as first step of the method, T is filtered such that it contains items with n–support larger than or equal to $MinSupp$.

Lemma 1 *An item i with $i.supp(n) < MinSupp$ cannot appear in a LSP p.*

Proof 1 *The proof trivially follows from the minimum requirement of the n–support of a pattern p. If i appears in a pattern p, then the set of transactions satisfying p must be a subset of the transactions satisfying i. Consequently, $p.supp(n) \leq i.supp(n)$. For p to be a pattern $p.supp(n) \geq MinSupp$. This is a clear contradiction, hence i cannot appear in a pattern.*

The first step can be formulated in two SQL statements. The first statement finds items that meet the unique support criterion. The second statement constructs a filtered view of T, called *TFV*, in which transactions only contain the items found by the previous statement.

```
INSERT INTO F (item, i_cnt)
SELECT item, count(*) i_cnt FROM T
GROUP BY item
HAVING COUNT(DISTINCT oid)>=n AND COUNT(*)>=MinSupp

CREATE VIEW TFV AS
SELECT T.oid, T.tid, T.item FROM T, F WHERE T.item=F.item
```

The effects of the first step are illustrated in Figure 4.4. Spatio–temporal regions, which are part of trajectories that belong to less than 2 distinct objects, are removed from trajectories. From the point of view of an intelligent rideshare application these spatio–temporal regions are uninteresting, since these parts of the trajectories cannot be shared by any objects, i.e., are not sharable.

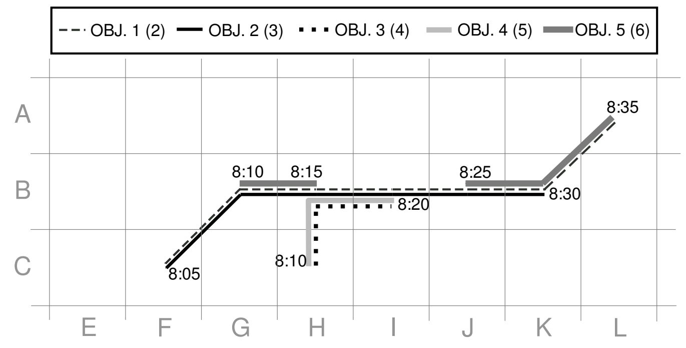


Figure 4.4: The Sample DB after STEP 1.

4.5.2 STEP 2: Filtering of Short Transactions

Transactions, i.e., trip trajectories, having less than $MinLength$ frequent items cannot satisfy a LSP. Hence, the second step of the method further filters *TFV* and constructs *TF* that only contain transactions that have at least $MinLength$ number of items.

Lemma 2 *A transaction t with $|t| < MinLength$ cannot satisfy a LSP p.*

Proof 2 *The proof trivially follows from the definition of a LSP and the definition of a transaction satisfying a pattern. p is a LSP $\iff$ $p.supp(n) \geq MinSupp$ and $|p| \geq MinLength$. For t to satisfy p, by definition all the items in p has to be present in t. Since $|t| < MinLength$ and $|p| \geq MinLength$, there must exist at least one item in p that is not in t. Hence, t cannot satisfy p.*

The second step can be formulated in one SQL statement. The sub–select is used to find trip identifiers that have at least $MinLength$ number of items. The outer part of the statement selects all records belonging to these trip identifiers and inserts them into *TF*.

```
INSERT INTO TF (tid, oid, item)
SELECT tid, oid, item FROM TFV WHERE tid IN
   (SELECT tid FROM TFV GROUP BY tid
    HAVING COUNT(item)>=MinLength)
```

The effects of the second step are illustrated in Figure 4.5. In particular, the remaining sharable parts of trips belonging to objects 3 and 5 are deleted, because the length of them is not greater than or equal to $MinLength$, which is 4 in the example. Also, note that although in this case items HB8:15 and IB8:20 did not become infrequent in *TF*, they lost n–support.

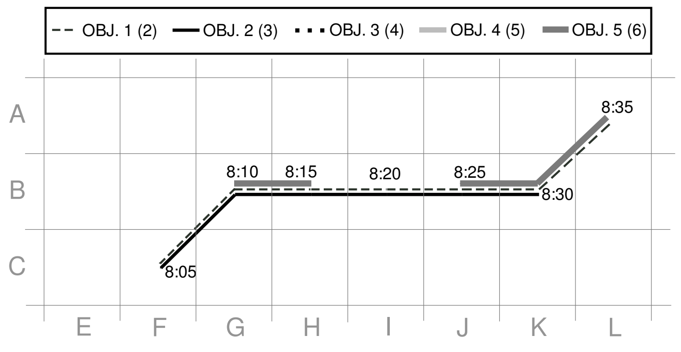


Figure 4.5: The Sample DB after STEP 2.

Before stating further observations and continuing with the development of the proposed method it is important to note the following. The set of discoverable LSPs from T is equivalent to the set of discoverable LSPs from *TF*. This is ensured by first two observations. Since further steps of the proposed method will discover LSPs from *TF*, these two observation ensure the correctness of the method so far. However, it is also important to note that not all transactions in *TF* necessarily satisfy a LSP. This is due to the sequentiality of the first two steps. After the first step all the remaining items in transactions are frequent items. Then, in the second step, some of these transactions, which are not long, are deleted. Due to this deletion a frequent item in the remaining long transactions may become non–frequent, which in turn may cause some transactions to become short again. While there is no simple solution to break this circle, note that the correctness of the first and second steps are not violated since the deleted items and transactions could not have satisfied a LSP.

4.5.3 STEP 3: Item–Conditional DB Projection

For the following discussion, adopted from [76], let an item–conditional database of transactions, equivalently referred to as an item–projected database, be defined as:

Definition 5 *Let T be a database of transactions, and i an item in T. Then, the item–conditional database of transactions, is denoted as $T_{|i}$ and contains all the items from the transactions containing i.*

The construction of an item–conditional database of transactions can be formulated in a single SQL statement as:

```
INSERT INTO T_i (oid, tid, item)
SELECT t1.oid, t1.tid, t1.item FROM TF t1, TF t2
WHERE t1.tid = t2.tid and t2.item = i
```

Given n frequent items in T, the problem of finding CFIs can be divided into n subproblems of finding the CFIs in each of the n item–projected databases [76]. Using the divide–and–conquer paradigm, each of these n subproblems can be solved by recursively mining the item–projected databases as necessary.

4.5.4 STEP 4: Discovery of the Single Most Frequent Closed Itemset

Since i is in every transaction of the item–projected database $T_{|i}$, and hence has maximum n–support, the items in $T_{|i}$ can be grouped in two: items that have the same n–support as i, and items that have n–support less than that of i. The set of items that have the same n–support in the $T_{|i}$ as i is the Single Most Frequent Closed Itemset (SMFCI) in $T_{|i}$. The fourth step of the method discovers this SMFCI.

Lemma 3 *Let i be an item and $TF_{|i}$ its corresponding item–projected database. Let A be the set of items that have the same n–support in $TF_{|i}$ as i. Then A is the SMFCI in $TF_{|i}$.*

Proof 3 *Complementary to A, let B be a set of items that have n–support less than i in $TF_{|i}$. For A to be closed there should not exist an itemset X such that $A \subset X$ and $A.supp(n) = X.supp(n)$. This implies that there should not exist an extra item i_e that is present in all transactions in which i is present. These transactions are exactly the set of transactions that make up $TF_{|i}$. The only remaining items that are not in A and are present in $TF_{|i}$ are items in B. Since items in B have n–support less than the items in A they could not be added to A to form an itemset X such that $A.supp(n) = X.supp(n)$. Hence A is a closed itemset. That A is the SMFCI trivially follows from the fact that the number of transaction in $TF_{|i}$ is $A.supp(n)$.*

The fourth step can be formulated in two SQL statements. The first statement derives the n–support of n–frequent of items in $TF_{|i}$,while the second statement selects those items from these n–frequent items that have maximum n–support.

```
INSERT INTO FT_i (item, i_cnt)
SELECT item, COUNT(*) i_cnt FROM T_i
GROUP BY item HAVING COUNT(DISTINCT oid)>=n

SELECT item FROM FT_i
WHERE i_cnt = (SELECT MAX(i_cnt) FROM FT_i)
```

Figure 4.6 shows the effects of projecting *TF* based on the item FC8:05. The numbers in parentheses show the n–support of the items in $TF_{|\mathrm{FC8:05}}$ and *TF* respectively. The SMFCI that is immediately discovered from $TF_{|\mathrm{FC8:05}}$ is {FC8:05, GB8:10, HB8:15, IB8:20, JB8:25, KB8:30}. LA8:35 is the only item that is in

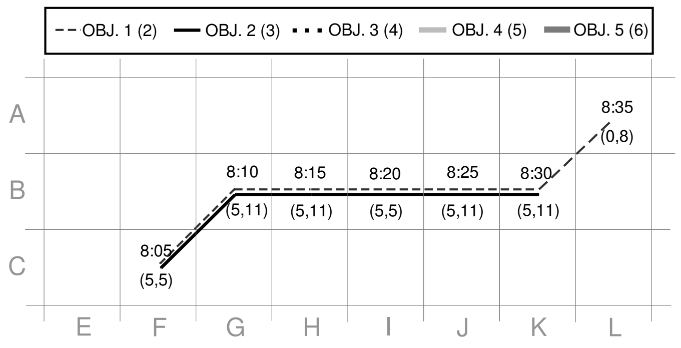


Figure 4.6: Item–Conditional Sample DB $TF_{|\mathrm{FC8:05}}$ and Pattern Discovery.

$TF_{|\mathrm{FC8:05}}$, but is not in the discovered SMFCI. Since further projecting $TF_{|\mathrm{FC8:05}}$ on LA8:35 yields a database of transactions where no item meets the minimum n–support criterion, the discovered SMFCI is the only CFI present in $TF_{|\mathrm{FC8:05}}$. Since the discovered SMFCI meets both the minimum length and an minimum n–support criteria it is a pattern.

Lemma 4 *Given an item–projected database $TF_{|i}$ and a partitioning of items in it into a set of most frequent items A, and a complementary set of items B, the recursive application of item–projection based on items in B followed by the discovery of the most frequent closed itemsets in the respective projected databases finds all CFIs in the item–projected database $TF_{|i}$.*

Proof 4 *Given any CFI X with $X.supp(n) < A.supp(n)$ in $TF_{|i}$, X contains at least 1 item $b \in B$. If not, then X contains only items in A, hence $X.supp(n) \geq A.supp(n)$, which is a clear contradiction. Then by Lemma 3, X will be found as the SMFCI in $TF_{|i|b}$, since $TF_{|i|b}$ contains all, and only those transaction from $TF_{|i}$ that satisfy b.*

4.5.5 STEP 5: Deletion of Unnecessary Items

The subproblems that are recursively solved by the method presented so far are overlapping. That is to say, viewed from a top level, a CFI that has n items is at least once discovered in each of the n corresponding item–projected databases. To eliminate this redundancy, both in the mining process and the result set, observe that an item j can be deleted from *TF* if it has the same n–support in $TF_{|i}$ as in *TF*. The intuition behind the observation is the following. If j has the same n–support in $TF_{|i}$ as in *TF*, it implies that all the transactions in *TF* that satisfy j are also present in $TF_{|i}$.

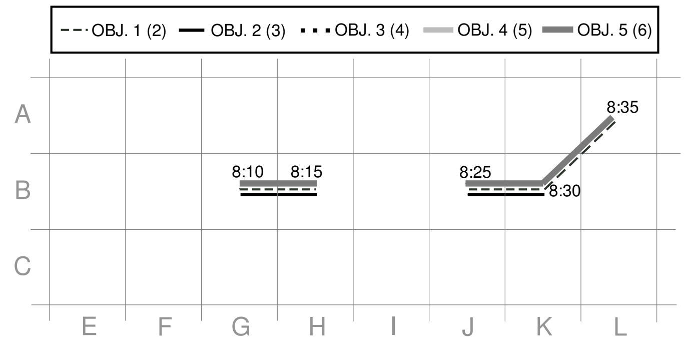


Figure 4.7: The Sample DB after STEP 5.

Thus, the set of patterns containing j, which can be discovered from *TF*, can also be discovered from $TF_{|i}$.

Lemma 5 *After the construction of $TF_{|i}$, an item j can and must be deleted from* TF *if it has the same n–support in $TF_{|i}$ as in* TF.

Proof 5 *If j has the same n–support in TF as in $TF_{|i}$, then the set of transactions that satisfy j in TF is exactly the same set of transactions that satisfy j in $TF_{|i}$. Since Lemma 4 guarantees that all closed frequent itemsets will be found in $TF_{|i}$, including those that j participates in, it is needless and incorrect to construct and mine $TF_{|j}$ at a later point to find same CFIs that j participates in again. Hence, j can and must be deleted from TF.*

The fifth step can be formulated in one SQL statement. The statement deletes all items in *TF* that have the same n–support in *TF* as in $TF_{|i}$.

```
DELETE FROM TF WHERE TF.item IN
   (SELECT F.item FROM F, FT_i
    WHERE F.item = FT_i.item
      AND F.i_cnt = FT_i.i_cnt)
```

Figure 4.7 shows the effects of deleting the unnecessary items after the mining of $TF_{|\mathrm{FC8:05}}$. Since items FC:8:05 and IB8:20 have the same n–support in $TF_{|\mathrm{FC8:05}}$ as in *TF*, shown in Figure 4.6, they are deleted from *TF*. Items remaining in *TF* are shown in Figure 4.7.

4.5.6 Item–Projection Ordered by Increasing n–Support

A LSP p in T, containing items $i_1 \ldots i_k$, can be discovered from any one of the item–projected databases $T_{|i_1}, \ldots, T_{|i_k}$. Steps 4 and 5 of the proposed method assure that

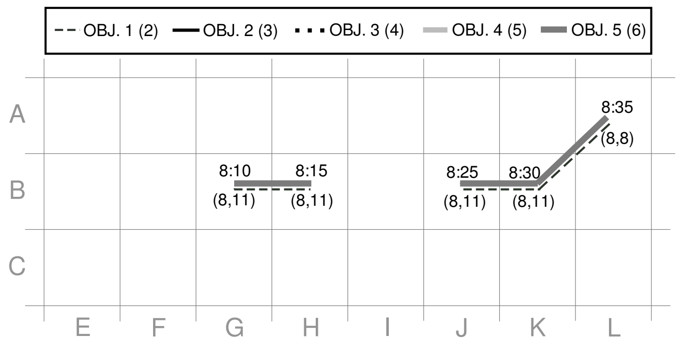


Figure 4.8: Item–Conditional DB $TF_{|\text{LA8:35}}$ and Pattern Discovery.

p will be discovered from exactly one of these item–projected databases, but the method presented so far does not specify which one. While this point is irrelevant from the point of view of correctness, it is crucial from the point of view of effectiveness.

To illustrate this, assume that $i_1.supp(n) < i_2.supp(n) < \ldots < i_k.supp(n)$. If projections are performed in decreasing order of item n–support, then, first $T_{|i_k}$ is constructed, then $T_{|i_k|i_{k-1}}$ is constructed from it, and so on, all the way to $T_{|i_k|i_{k-1}|\ldots|i_1}$, from which finally p is discovered. If on the other hand, projections are performed in increasing order of item n–support, then p is discovered from the first item–projected database that is constructed, namely $T_{|i_1}$.

Assume that p and its qualifying $(k - l + 1)$ (at least l–long) sub–patterns are the only LSPs in T. Then during the whole mining process, the total number of projections in the decreasing processing order is $P_{dec} = k$, whereas in the increasing processing order the total number of projections is only $P_{inc} = k - l + 1$. If k and l are comparable and large, then $P_{dec} \gg P_{inc}$. Similar statements can be made about the total size of the projected databases in both cases. Hence, item–projection should be performed in increasing order of item n-support.

4.5.7 Alternating Pattern Discovery and Deletion

Alternating steps 3, 4 and 5, all patterns can be discovered in a recursive fashion. The sequential application of these steps is referred to as a *Pattern Discovery and Deletion phase* (PDD). Mining terminates when all items have been deleted from *TF*.

Figures 4.6 and 4.7 shows the effects of the first of these PDD phases. Figures 4.8 and 4.9 show the effect of the next pattern PDD phase. Since after the first PDD phase LA8:35 has the lowest n–support in *TF*, namely 8, it is chosen as the next item to base the database projection on. Figure 4.8 show $TF_{|\text{LA8:35}}$ with the corresponding n–support of the items in $TF_{|\text{FC8:05}}$ and *TF* respectively. Since all the items have the same n–support in $TF_{|\text{FC8:05}}$ as LA8:35, namely 8, the closed itemset $\{$GB8:10,

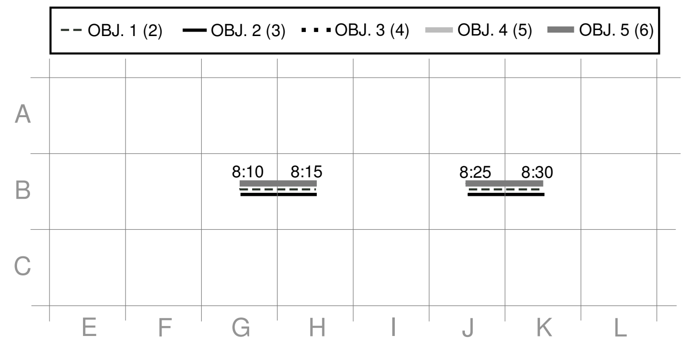


Figure 4.9: The Sample DB after the Second PDD Phase.

HB8:15, JB8:25, KB8:30, LA8:35$\}$ is discovered. Since this closed itemset both meets the minimum length and n–support requirements it is recorded as a pattern. In the deletion part of this PDD phase, item LA8:35 is deleted from *TF* as the only item that have the same n–support in $TF_{|\text{LA8:35}}$ as in *TF*. The results of this deletion are shown on Figure 4.9.

The third and final PDD phase is implicitly shown in Figure 4.9. Since after the second PDD phase all the items in *TF* have the same n–support, the next projection is performed on any one of the items, i, and the resulting item–projected database, $TF_{|i}$, is identical to the current state of *TF*, depicted on Figure 4.9. Since all the items in $TF_{|i}$ have the same n–support as i, the closed itemset $\{$GB8:10, HB8:15, JB8:25, KB8:30$\}$ is discovered. Since this closed itemset meets both the minimum length and n–support requirements, it is recorded as a pattern. Finally, items having the same n–support in $TF_{|i}$ as in *TF*, which in this case means all the items in $TF_{|i}$, are deleted from *TF*. After this deletion part of the final PDD phase, *TF* becomes empty and the

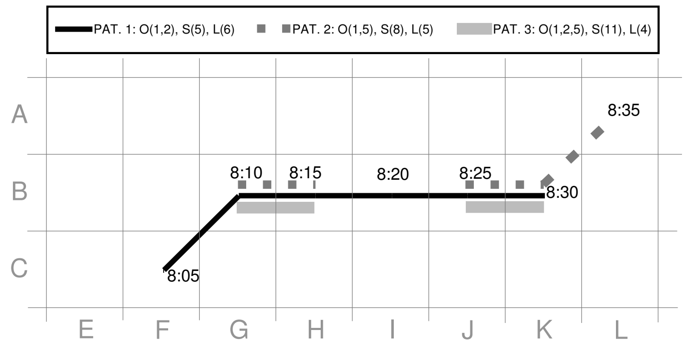

PAT. 1: O(1,2), S(5), L(6)
PAT. 2: O(1,5), S(8), L(5)
PAT. 3: O(1,2,5), S(11), L(4)
A
B
C
8:35
8:10
8:15
8:20
8:25
8:30
8:05
E
F
G
H
I
J
K
L

Figure 4.10: Three Patterns in the Sample DB.

```
(1)  procedure MineLSP ($T$, $MinSupp$, $MinLength$, $n$)
(2)      $TF^0_{freq} \leftarrow$ MinSupportFilter ($T$, $MinSupp$, $n$)
(3)      $TF^0_{long} \leftarrow$ MinLengthFilter ($TF^0_{freq}$, $MinLength$)
(4)      for each freq item $i$ in $TF^0_{long}$ ordered by asc $n$–supp
(5)          $TF^0_{|i} \leftarrow$ ConstructConditionalDB ($TF^0_{long}$, $i$)
(6)          FindLSP ($TF^0_{|i}$, 1, $MinSupp$, $MinLength$, $n$)

(1)  procedure FindLSP ($T$, $L$, $MinSupp$, $MinLength$, $n$)
(2)      $TF^L_{freq} \leftarrow$ MinSupportFilter ($T$, $MinSupp$, $n$)
(3)      $TF^L_{long} \leftarrow$ MinLengthFilter ($TF^L_{freq}$, $MinLength$)
(4)      $(P, P.supp(n)) \leftarrow$ FindSMFCI ($TF^L_{long}$)
(5)      $TF^{L-1}_{long} \leftarrow$ DeleteUnnecessaryItems ($TF^{L-1}_{long}$, $TF^L_{freq}$)
(6)      if $P.supp(n) \geq MinSupp$ and $|P| \geq MinLength$
(7)          StorePattern ($P$, $P.supp(n)$)
(8)      for each freq item $i$ in $TF^L_{long}$ ordered by asc $n$–supp
(9)          if $i$ is not in $P$
(10)             $TF^L_{|i} \leftarrow$ ConstructConditionalDB ($TF^L_{long}$, $i$)
(11)             FindLSP ($TF^L_{|i}$, $L+1$, $MinSupp$, $MinLength$, $n$)
```

Figure 4.11: The LSP Algorithm.

mining terminates. Figure 4.10 shows the three patterns that are discovered during the mining. Supporting oids, n–supports, and length for each discovered patterns are shown in the legend.

4.5.8 LSP Mining Algorithm

Using the observations and the associated steps, the complete algorithm for mining LSPs in trajectories is given in Figure 4.11. Since item–projected databases are constructed at every level of the recursion and are modified across levels when deleting unnecessary items, the level of recursion L is passed as an argument in the recursive procedure, and is used as a superscript to associate databases to the levels they were constructed in.

Lines 2 and 3 in the MineLSP procedure represent steps 1 and 2 of the method, and they construct the filtered database of transactions at the initial level, level 0. Line 4 processes frequent items in TF^0 in ascending order of n–support. Line 5 represent step 3 of the method, and for each such frequent item i, it constructs the item–conditional database of transactions $TF^0_{|i}$ at level 0. Line 6 calls procedure FindLSP to extract all LSPs from $TF^0_{|i}$ recursively.

Lines 2 and 3 in the FindLSP procedure represent steps 1 and 2 of the method, and they construct the filtered database of transactions at the current level L. Line 4 represents step 4 of the method, and it finds the SMFCI P in TF^{L}_{long}. Line 5 represents step 5 of the method, and it deletes all items from the filtered database of transactions of the previous level, TF^{L-1}_{long}, that have the same n-support in TF^{L-1}_{long} as in TF^{L}_{freq}, the current level. Lines 6 and 7 check if the single most frequent closed itemset P meets the minimum requirements and store it accordingly. Lines 8 and 9 processes frequent items in TF^{L}_{long}, which are not in P, in ascending order of n–support. Line 10 represent step 3 of the method, and for each such frequent item i it constructs the item–conditional database of transactions $TF^{L}_{|i}$ at the current level L. Finally, line 11 recursively calls procedure FindLSP to find LSPs in $TF^{L}_{|i}$ at the next level.

The structure and functionality of procedures MineLSP and FindLSP have a significant overlap. While the two functions can be merged into one, the separation of the two is used to emphasize the facts that (1) DeleteUnnecessaryItems presumes the existence of databases constructed at the previous level, and (2) FindSMFCI correctly operates only on an item–projected database, and hence it can only be applied at level 1 and above.

Several implementation details are worth mentioning. First, DeleteUnnecessaryItems deletes items from TF^{L-1}_{long} based on the n–support of items in TF^{L}_{freq}, not TF^{L}_{long}. This is important, as it was noted that MinLengthFilter decreases the n–support of items in TF^{L}_{freq}, thereby possibly making an unnecessary item appear to be necessary. Second, arguments to functions and operands in statements are logical, i.e., the functions and statements can be more efficiently implemented using previously derived tables. For example, both FindSMFCI and DeleteUnnecessaryItems are implemented using previously derived n–support count tables not the actual trajectory tables. Third, simple shortcuts can significantly improve the efficiency of the method. For example, during the derivation of TF^{L}_{freq}, if the number of unique frequent items in TF^{L}_{freq} is less than $MinLength$, no further processing is required at that level, since none of the CFIs that can be derived from TF^{L}_{freq} are long. To preserve clarity, these simple shortcuts are omitted from Figure 4.11.

4.6 Alternative Modelling of Trajectories and Mining of LSPs

The region–based and the road network based spatio–temporal generalization approaches, presented in Section 4.3, model trajectories at a local (micro) level. Consequently, the method presented in Section 4.5 analyzes the trajectories at the local level and derives local (micro) patterns. Alternatively, trajectories can also be modelled at the global (macro) level, whereby trips in trajectories are represented as origin–destination pairs. Global (macro) modelling and analysis of trajectories is a domain of considerable interest in transportation and urban analysis [41]. For example, recently a Cab–Sharing Service was proposed as an effective, door–to–door, on–demand transportation service [33]. One component of the proposed Cab–Sharing System is a Cab–Routing / Scheduling Engine. The task of this engine is to route idle cabs and assign cabs to requests or groups of requests, so called cab–shares, such that the demand for cabs is optimally served both in terms of the transportation cost of idle cabs and the service time of requests. To enable this optimization, as future work, the use of spatio–temporal patterns in cab requests for cab request demand prediction is proposed. Since cab requests are naturally represented as origin–destination pairs, the usefulness of macro analysis is apparent.

Hence, in the following two alternative options for modelling trajectories and mining of LSPs are described. Section 4.6.1 describes a simple method with an SQL implementation for mining LSPs in trajectories modelled at the global (macro) level. Section 4.6.2, using some intuitive assumptions, combines the macro and micro modelling options and LSP mining methods to derive a hybrid version.

4.6.1 Macro Modelling of Trajectories and Mining of LSPs

As briefly described above, when modelling trajectories at the global (macro) level, trips in trajectories are represented as origin–destination pairs. The preprocessing of raw trajectories can be achieved using the same three transformation steps as described in Section 4.3. This includes the possibility of using either the region–based or the road network based spatio–temporal generalization approaches as is required from the application at hand. In the so obtained transaction database, a trajectory belonging to a particular object has exactly two items.

Mining global (macro) LSPs in the so obtained trajectory database can be achieved using a single SQL statement as follows.

```
SELECT o_item, d_item, SUM(supp) AS nsupp FROM
   (SELECT oid, o_item, d_item, COUNT(*) AS supp FROM T
    WHERE dist(o_item, d_item) >= MinDist
    GROUP BY oid, o_item, d_item) a
GROUP BY o_item, d_item
HAVING COUNT(*) >= n AND SUM(supp) >= MinSupp
```

The statement, without loss of generality, assumes that the trajectory database T has the schema $\langle oid, tid, o\_item, d\_item \rangle$, where in addition to the previously used notation, o_item and d_item are generalized spatio–temporal regions, or identifiers of the origin and destination of the trajectory, respectively. Since all the trajectories in the database T have exactly two items, it does not make sense to talk about the length of a trajectory in terms of number of items it contains. Instead, a distance function

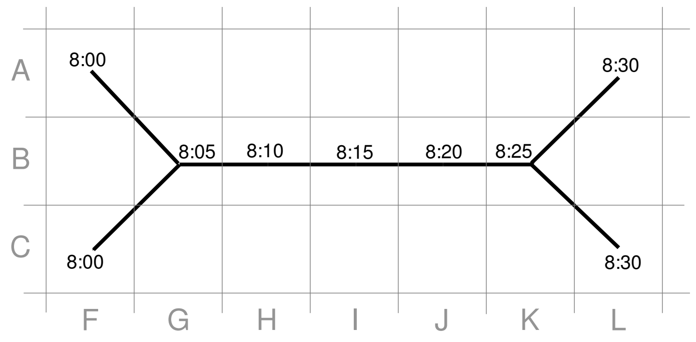


Figure 4.12: Loss of Local (Micro) LSP.

$dist()$ between two locations or items can be defined, and patterns can be evaluated against a $MinDist$ criterion. The inner select statement calculates the supports for object–specific origin–destination item combinations that satisfy the $MinDist$ criterion. The outer select statement aggregates the results of the inner select statement, and identifies origin–destination item combinations that meet the n–support criterion, i.e., global (macro) LSPs, and calculates their corresponding n–supports.

4.6.2 Hybrid Modelling of Trajectories and Mining of LSPs

The proposed global (macro) modelling approach of trajectories, the global (macro) LSP mining method and the SQL implementation given for it are likely to be very efficient due to the indexing and aggregation support provided by RDBMSs. However, the discovered global (macro) LSPs, will have very little relation to each other. For example, a set of individual global (macro) LSPs, considering the underlying road network, might give rise to local (micro) LSPs that do not exist in the macro model, but have a support that is equal to the sum of the n-supports of the individual patterns. As an illustrative example consider the road network represented by the solid black lines in Figure 4.12. Assume that for a particular setting of the parameters, the global (macro) LSP mining method finds two global (macro) LSPs, {FA8:00, LA8:30} and {FC8:00, LC8:30}, with n–supports 10 for each. Assuming that the spatial regions FA, LA, FC, and LC cover the only four cities in the area, and hence trajectories only start and finish in these regions, the global (macro) LSP mining method will not discover the local (micro) LSP {GB8:05, HB8:10, IB8:15, JB8:20, KB8:25} with n–support 20.

To overcome this deficiency of the global (macro) LSP mining method, the global (macro) and the local (micro) modelling approaches and LSP mining methods can be combined into a hybrid modelling and LSP mining method as follows. First, perform global (macro) LSP mining on the spatio–temporally generalized input trajectories.

```
(1) procedure HybridMineLSP ($Traj_I$, $MinDist$, $MinSupp$, $MinLength$, $n$)
(2)     $Traj_G \leftarrow$ STGeneralize ($Traj_I$)
(3)     $LSP_{macro} \leftarrow$ MacroMineLSP ($Traj_G$, $MinDist$, $MinSupp$, $n$)
(4)     $Traj_A \leftarrow$ ApproximateTraj ($LSP_{macro}$)
(5)     $Traj_{GA} \leftarrow$ STGeneralize ($Traj_A$)
(6)     $LSP_{micro} \leftarrow$ MineLSP ($Traj_{GA}$, $MinLength$, $MinSupp = 1$, $n = 1$)
```

Figure 4.13: The Hybrid LSP Mining Method.

Then, using the global (macro) LSPs and the underlying road network, *approximate* trajectories for the global (macro) LSPs, i.e., find shortest paths between origin–destination pairs. Then, spatio–temporally generalize the approximated trajectories and mine local (micro) LSPs in them, taking into account the global (macro) n–supports of the approximated trajectories. Taking into account n–supports of the approximated trajectories can either be achieved by slightly modifying the local (micro) LSP mining method in Section 4.5, or simply the original version of it can be called with parameters $n = 1$ and $MinSupp = 1$. The latter is necessary and sufficient to ensure that (1) the approximated trajectories belonging to the global (macro) LSPs are found as local (micro) LSPs as well, and (2) the local (micro) LSPs found meet the original n–support criterion. Note that the spatio–temporal generalization of the input and approximated trajectories can either be regions–based or road network based. Figure 4.13 gives the pseudo code of the hybrid LSP mining method, as described above.

While the hybrid LSP mining method is likely to reduce the number of input trajectories to the local (micro) LSP mining method called internally, thereby achieving a significant speed–up in running time, it does not find *all* the local (micro) LSPs. As an example consider the trajectories in Figure 4.12. If $MinSupp = 20$, then the hybrid LSP mining method will not find any patterns in the global (macro) LSP mining phase, and consequently will not find any local (micro) LSPs, even though the local (micro) LSP $\{$GB8:05, HB8:10, IB8:15, JB8:20, KB8:25$\}$ has an n–support of 20. Similarly, it can be argued that the hybrid LSP mining method will not find any LSPs in the example trajectory database in Section 4.3.2, for the parameters used in the running example in Section 4.5. However, as the spatio–temporal generalization granularity used in the mining is decreased the chances not identifying global (macro) LSPs that give rise to local (micro) LSPs is decreased. In summary, the hybrid LSP mining method is likely an effective alternative that provides lossy and approximate results when compared to the local (micro) LSP mining method.

4.7 Experimental Evaluation

The proposed LSP mining methods were implemented using MS–SQL Server 2000 running on Windows XP on a 3.6GHz Pentium 4 processor with 2GB main memory. Three groups of experiments were performed to test: (1) the parameter sensitivity of the local (micro) LSP mining method, (2) the scale–up properties of the local (micro) LSP mining method, and (3) the effectiveness of the global (macro) modelling and LSP mining method with respect to its parameters. The three groups of experiments were performed on the following three data sets respectively: (1) the publicly available INFATI data set [58], which comes from intelligent speed adaptation experiments conducted at Aalborg University, (2) the synthetic ST–ACTS trajectory data set, and (3) the ST–ACTS origin–destination data set, both of which were derived from ST–ACTS, a probabilistic, parameterizable, realistic Spatio–Temporal ACTivity Simulator [31]. Sections 4.7.1, 4.7.2 and 4.7.3 describe these data sets in detail, while Sections 4.7.4, 4.7.5 and 4.7.6 present the results of the respective groups of experiments. Finally, Section 4.7.7 visualizes some of the mining results.

4.7.1 The INFATI Data Set

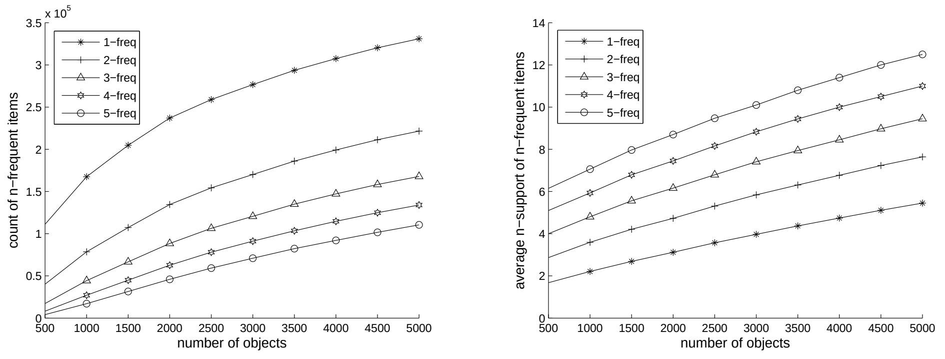
The INFATI data set records cars moving around in the road network of Aalborg, Denmark over a period of several months. 20 distinct test cars and families participated in the INFATI experiment; Team–1 consisting of 11 cars operated between December 2000 and January 2001 and Team–2 consisting of 9 cars operated between February and March 2001. Each car was equipped with a GPS receiver, from which GPS positions were sampled every second whenever the car was operated. Additional information about the experiment can be found in [58].

The method presented in Section 4.3.1 identifies trips from continuous GPS measurements, which is not the case in the INFATI data. Hence in this case, a trip was defined as sequence of GPS readings where the time difference between two consecutive readings is less than 5 minutes. Using the definition, the INFATI data contains 3,699 trips. After projecting the temporal dimension to the time–of–day domain and substituting the noisy GPS readings with 100 meter by 100 meter by 5 minutes spatio–temporal regions, the resulting trajectory database has the following characteristics. There are 200,929 unique items in the 3,699 transactions. The average number of items in a transaction is approximately 102. The average n–support of 1–, 2–, and 3–frequent items is 1.88, 4.2 and 6.4 respectively. Notice that the averages only include the n–supports of 1–, 2–, and 3–frequent items.

4.7.2 The ST–ACTS Trajectory Data Set

The ST–ACTS trajectory data set is based on the output of ST–ACTS, a probabilistic, parameterizable, realistic Spatio–Temporal ACTivity Simulator [31]. Based on a number of real world data sources and a set of principles that try to model the *social* and some of the *physical* aspects of mobility, ST–ACTS simulates realistic spatio–temporal activity sequences of approximately 600,000 individuals in the city of Copenhagen, Denmark. Since the aim of ST–ACTS is to simulate realistic spatio–temporal activities of individuals that contain patterns, rather than to simulate detailed movements of individuals, the output of the ST–ACTS for each simulated individual is a sequence of timestamped locations and activities. Two consecutive locations in such a sequence can be seen as the origin and the destination of a trip's trajectory. To obtain a realistic approximation for the missing part of the trajectories, using the underlying road network, segment–based shortest path calculations were performed between the origin–destination pairs of the trips. The so obtained trip trajectories are analogous in form and semantics to the trajectories that can be obtained using the road network based spatio–temporal generalization approach as explained in Section 4.3.1.

For the period of three working days, spatio–temporal activities of 5,000 individuals were simulated, resulting in a total of 64,144 trips. Using segment–based routing between origin–destination pairs, an average trip is 1,850 meters long with a standard deviation of 1,937 meters, and is made up of 28 road segments with a standard deviation of 27 road segments. An average road segment is 66 meters long with a standard deviation of 64 meters. After projecting the temporal dimension to the time–of–day domain and using the road segments as spatial–, and a 15–minute interval as temporal, generalization units, the resulting trajectory database has the following characteristics. There are 330,940 unique items in the 64,144 transactions. The average number of items in a transactions is approximately 28. To test the scale–up properties of the proposed method, varying sized subsets of the ST–ACTS trajectory data set were constructed. Figure 4.14 summarizes the characteristics of these subsets in terms of the number (Figure 4.14(a)) and n–support (Figure 4.14(b)) of n–frequent items. While not shown in Figure 4.14, the number of trajectories linearly scales with the number of objects in the data sets between 6,601 trajectories for 500 objects and 64,144 trajectories for 5,000 objects. Similar linear relationships exist between the number of objects and the average n–support of n–frequent items, Figure 4.14(b). The logarithmic like relationships between the number of objects and the number of n–frequent items is due to the fact that the increasing number of trajectories traverse, and make n–frequent, an increasing fraction of road segments of the total road network, see Figure 4.14(a). In other words, the number of n–frequent items naturally saturates as the density of the trajectories increases.



(a) Number of n–Frequent Items. (b) Average n–Support of n–Frequent Items.

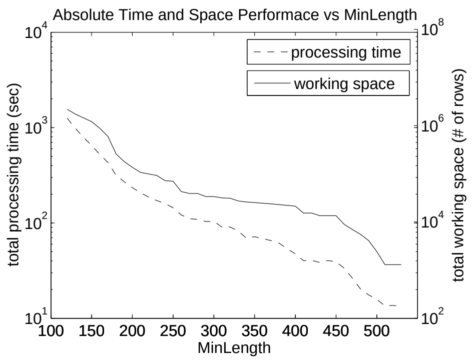
Figure 4.14: Data Characteristics of Varying Sized Subsets of the ST–ACTS Trajectory Data Set.

4.7.3 The ST–ACTS Origin–Destination Data Set

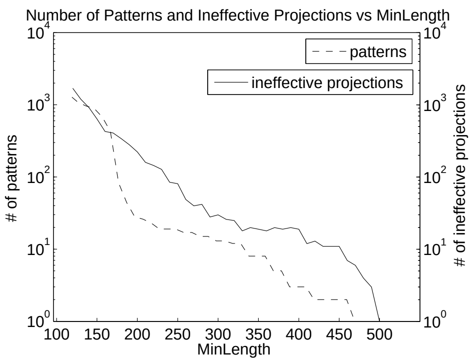
The ST–ACTS origin–destination data set, similarly to the ST–ACTS trajectory data set, is also based on the output of ST–ACTS. However, it includes the spatio–temporal activities of 50,000 individuals for a period of three working days, resulting in a total of 835,806 trips. The average Euclidean distance between the origins and destinations of the trips is 1,199 meters with a standard deviation of 1.555 meters. After projecting the temporal dimension to the time–of–day domain and substituting the origins and destinations of trips with 100 meter by 100 meter by 15–minute spatio–temporal regions, the resulting trajectory database has the following characteristics. There are 139,480 unique items in the 1,671,612 transactions. The number of items in every transactions is exactly 2, which correspond to an origin and a destination spatio–temporal region. The average n–support (and count) of 1–, 2–, 3–, and 4–frequent items is 1.12 (1,497,871), 2.14 (152,255), 3.24 (17,267), and 4.09 (3854) respectively. Notice that the averages only include the n–supports of 1–, 2–, 3–, and 4–frequent items.

4.7.4 Sensitivity Experiments for $MinSupp$ and $MinLength$ Parameters

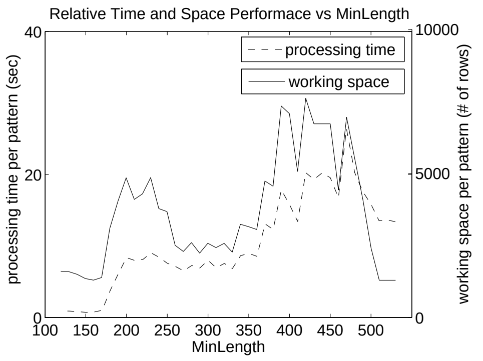
The first group of experiments was performed to test the sensitivity of local (micro) LSP mining method with respect to the $MinSupp$ and $MinLength$ parameters, and was using the INFATI data set as input. Two sets of experiments were performed, each varying one of the two parameters of the algorithm, $MinSupp$ and $MinLength$. The performance of the algorithm was measured in terms of processing time and working space required, where the working space required by the algorithm was approximated by the sum of the rows in the projected tables that were constructed by



(a) Absolute Time and Space.



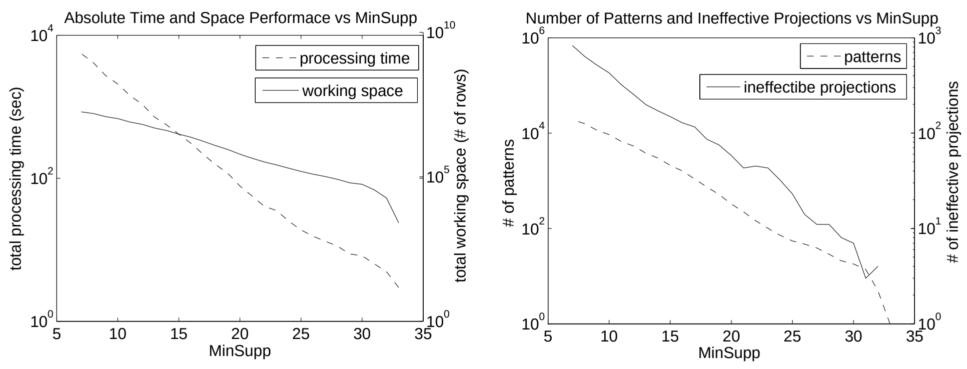
(b) Number of Patterns.



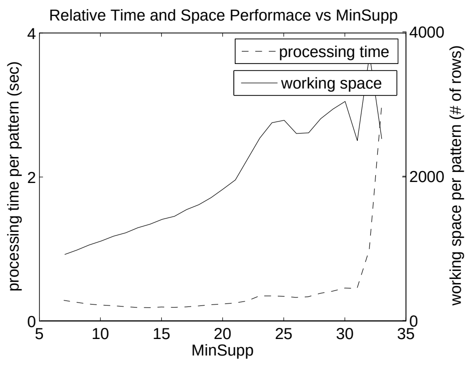
(c) Relative Time and Space.

Figure 4.15: Performance Evaluation for Various $MinLength$ Settings.

the algorithm. Both measures were evaluated in an absolute and a relative, per pattern, sense. Figures 4.15(a), 4.15(b), and 4.15(c) show the results of the first set of experiments, where $MinSupp = 2$, $n = 2$ and $MinLength$ is varied between 120 and 530. Lower settings for $MinLength$ were also tested, but due to the very low $MinSupp$ value these measurement were terminated after exceeding the 2 hour processing time limit. Noting the logarithmic scale in Figure 4.15(a) it is evident that both the running time and the working space required by the algorithm exponentially increase as the $MinLength$ parameter is decreased. Examining the same quantities in a relative, per pattern sense, Figure 4.15(c) reveals that the average running time and average working space required per pattern is approximately *linearly decreasing* as the $MinLength$ parameter is decreased. The presence of the two irregular bumps in Figure 4.15(c) can be explained in relation to the number of patterns found, and the number of ineffective projections that yield no patterns, shown in Figure 4.15(b). The sharp increases in relative processing time and working space are due to the fact that the algorithm is unable to take some of the shortcuts and it performs relatively



(a) Absolute Time and Space. (b) Number of Patterns.

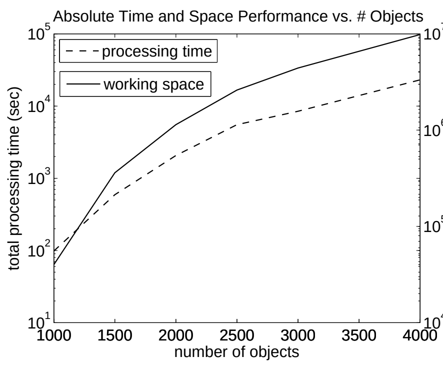


(c) Relative Time and Space.

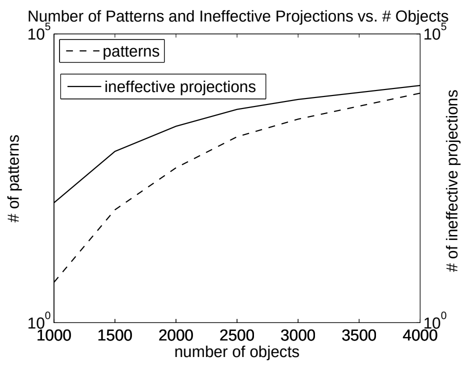
Figure 4.16: Performance Evaluation for Various $MinSupp$ Settings.

more ineffective projections yielding no pattern discovery. The sharp decreases can be explained by the presence of an increasing number of patterns that share the total pattern discovery cost.

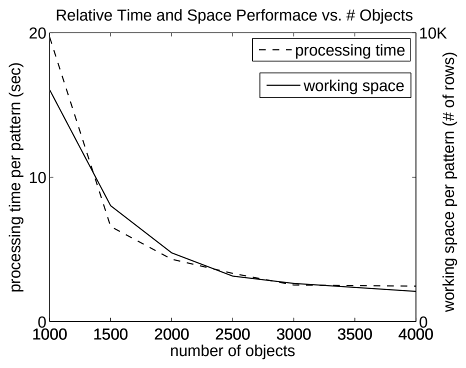
Similar observations can be made about the second set of experiments, shown in Figures 4.16(a), 4.16(b), and 4.16(c), where $MinLength = 50$, $n = 2$ and $MinSupp$ is varied between 7 and 33. For example, the sharp decrease in relative processing time in Figure 4.16(c) when going from $MinSupp = 33$ to $MinSupp = 32$ is simply due to the sudden appearance of patterns in the data for the given parameters. While there is only 1 pattern for $MinSupp = 33$, and an order of magnitude more number of patterns for $MinSupp = 32$, the projections performed and hence the absolute processing time to discover these patterns is approximately the same in both cases. Hence, the relative processing time for $MinSupp = 33$ is an order of magnitude larger than that for $MinSupp = 32$.



(a) Absolute Time and Space.



(b) Number of Patterns.

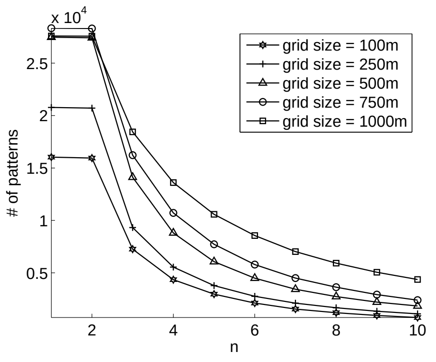


(c) Relative Time and Space.

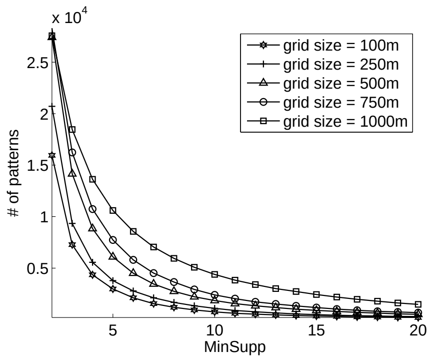
Figure 4.17: Performance Evaluation for Various Sized Data Sets, i.e., Number of Objects.

4.7.5 Scale–up Experiments for Various Input Data Sizes

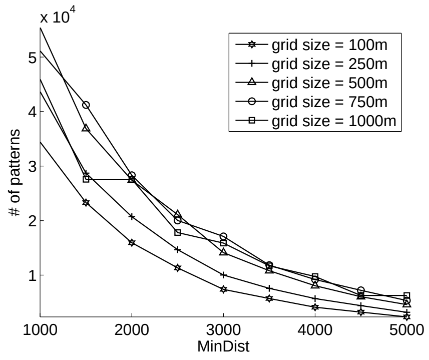
The second group of experiments was performed to test the scale–up properties of the local (micro) LSP mining method for varying size input data, and was performed using the ST–ACTS trajectory data set. For this group of experiments the algorithm's parameters were kept constant at $n = 4$, $MinSupp = 8$ and $MinLength = 25$. In other words, the patterns sought for were sub–trajectories with a minimum length of 25 road segments that were supported by at least 4 objects on average at least two times per object. The evaluation measures used in the experiments were the same as in the sensitivity experiments described in Section 4.7.4. Figure 4.17 shows the results of this group of experiments. The results can be summarized as follows. As the number of objects increases linearly, i.e. the density of the trajectories increases linearly, the number of patterns increases sub–exponentially, see Figure 4.17(b). This naturally leads to a sub–exponential increase in absolute running time (Figure 4.17(a) left) and working space (Figure 4.17(a) right). However, the examination of the same quantities in a relative, per pattern sense, see Figure 4.17(c), reveals that the average running time required per pattern gradually decreases to a close to constant value of a few seconds as the density of the trajectories increases. This is due to the fact that as the density of the trajectories is increasing, the number of ineffective projections relative to the number of patterns, i.e., the gap between the two, is decreasing, see Figure 4.17(c). Similar observations can be made about the average working space required per pattern. In summary, the scale–up experiments show that both the running time of, and working space required by the local (macro) LSP mining method scale sub–exponentially with the input data size and linearly with the output data size.



(a) Varying n.

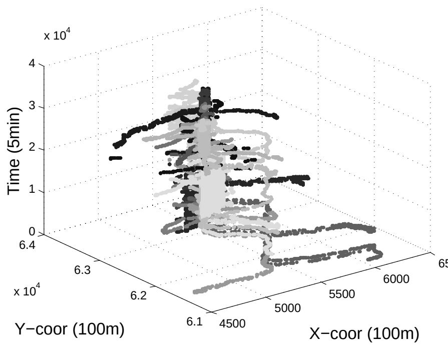


(b) Varying $MinSupp$.

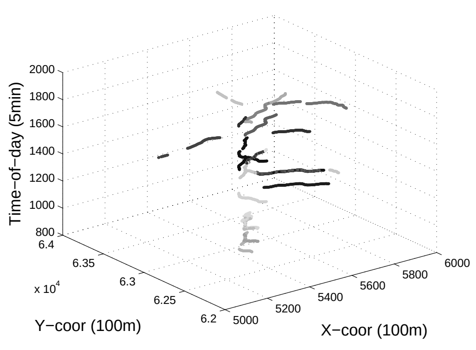


(c) Varying $MinDist$.

Figure 4.18: Number of Global (Macro) LSPs for Varying Spatio–Temporal Granularities and Parameter Settings.



(a) Moving Object Trajectories.



(b) 28 Discovered LSPs.

Figure 4.19: LSP Discovery Results in the INFATI Data Set.

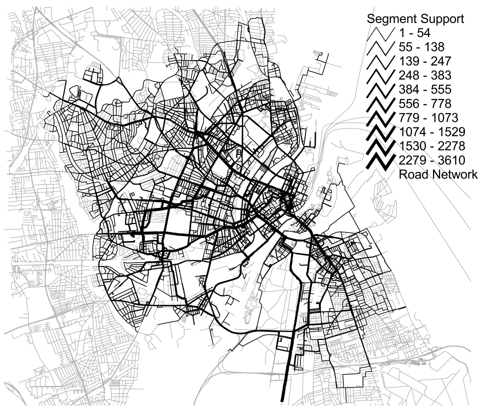
4.7.6 Global (Macro) Modelling and LSP Mining Experiments

The third group of experiments was performed to test the effectiveness of the global (macro) modelling and LSP mining method, as presented in Section 4.6.1. The experiments were performed for varying spatio–temporal generalization granularities for varying n, $MinSupp$ and $MinDist$ parameters, and were using the ST–ACTS origin–destination data set. Figure 4.18 shows the results of this group of experiments. The trends in the results are as expected. As the values for the parameters n, $MinSupp$ and $MinDist$ increases the number of global (macro) LSPs decreases, shown in Figures 4.18(a), 4.18(b), and 4.18(c), respectively. Similarly, all experiments show that as the spatio–temporal regions become coarser the number of patterns found increases. Perhaps more notable is the fact that for the rather large data set, with appropriate indexing the running time of the global (macro) LSP mining method is independent of the parameters and is under 2 seconds.

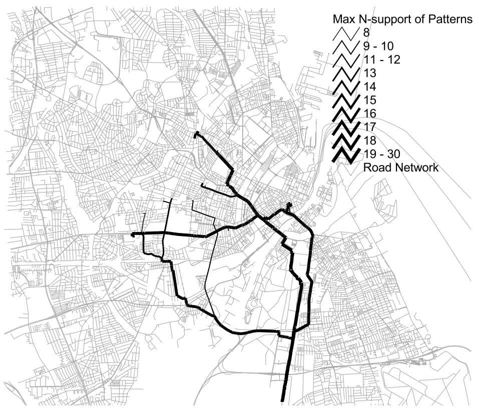
4.7.7 Visualization of Patterns

To see the benefits of mining LSPs, the mining results of two mining tasks are visualized in the following. Figure 4.19 presents the mining results of finding local (micro) patterns in the region–based spatio–temporally generalized INFATI data using the LSP algorithm from Section 4.5. Figure 4.19(a) shows a 50–fold down–sampled version of the trajectories of the 20 moving objects in the INFATI data set. While some regularities are apparent in it to the human eye, to find LSPs in it seems like a daunting task. Figure 4.19(b) shows 28 LSPs in it that are at least 200 long, sharable for at least 2 distinct objects, and have a support of at least 2.

Figure 4.20 presents the mining results of finding local (micro) patterns in the road network based spatio–temporally generalized ST–ACTS origin–destination data



(a) Overall Road Segment Support.



(b) Max. Road Segment n-Support of 298 LSPs.

Figure 4.20: LSP Discovery Results for the ST–ACTS Origin–Destination Data Set.

using the hybrid LSP mining algorithm from Section 4.6.2. The global (macro) mining of LSPs was performed for parameters $n = 4$, $MinSupp = 8$, and $MinDist = 5,000$ meters, and resulted in 109 global (macro) LSPs. After determining the value for the $MinLength$ parameter at 80, based on the minimum number of road segments in the approximated trajectories of the global (macro) LSPs, the local (micro) mining of LSPs resulted in 298 local (micro) LSPs. Figure 4.20(a) shows the overall supports (frequencies) of the road segments in the data. Figure 4.20(b) shows the maximum n–supports of the road segments induced by the collection of the 298 local (micro) LSPs. While, as described in Section 4.6.2, the hybrid LSP algorithm is not likely to find *all* local (micro) LSPs, it does find a relatively large number of additional local (micro) LSPs in a rather large data set under 144 seconds.

It is important to note that the LSPs contain additional information, which is only partially, or not presented in Figures 4.20(b) and 4.19(b), respectively. In particular, the n–supports, distances, and lengths are available for *individual* patterns, and the patterns naturally have a temporal aspect to them. With regards to the latter feature of the patterns, since the items in a pattern have a temporal component, an individual pattern refers to a particular time–of–day. Furthermore, since the spatio–temporally generalized items in a given pattern form a sequence in time, the patterns have a *direction*. While a simple temporal, frequency analysis of road segments can reveal *aggregated* information about the number of objects on the road segments at a given time–of–day, such analysis will not reveal movement patterns (including directions) of similarly moving objects. This additional information of the LSPs is likely to be of immense value in transportation and urban planning, and is necessary for the application at hand, be that intelligent ride–sharing or cab–sharing.

4.7.8 Summary of Experimental Results

In conclusion, the experimental evaluations can be summarized as follows. First, the sensitivity experiments show that the LSP mining method, presented in Section 4.5, is effective and is robust to changes in the user–defined parameter settings, $MinLength$ and $MinSupp$, and is a useful analysis tool for finding LSPs in moving object trajectories. Second, the scale–up experiments show that while the absolute running time and space required to find LSPs scales exponentially with the input data size, this is mainly due to the fact that the number of LSPs that are present in the input data, i.e., the output data size, also scales exponentially with the input data size. The scale–up experiments also demonstrate that the relative, per pattern, performance of the LSP mining method gradually decreases to a constant value as the input data size is increased. This later property of the LSP method is due to the fact that as the input size is increased, i.e., the density of the trajectories is increased, the effects of the $MinLength$ and $MinSupp$ pruning criteria become more dominant and relatively less and less ineffective projections are performed. Third, the experiments relating to global (macro) modelling and LSP mining, show that this modelling option and LSP mining method is extremely effective on large data sets, and is rather insensitive to the user–defined parameter settings, n, $MinDist$ and $MinSupp$. Finally, brief experiments show that the hybrid modelling and LSP mining method, while missing some local (micro) LSPs due to the partial global (macro) modelling, is able to find local (micro) LSPs in large data sets effectively.

4.8 Conclusions and Future Work

The herein presented work, for the first time, considers the problem of mining LSPs in trajectories and transforms it to a framework similar to that used in frequent itemset mining. The transformation allows both region–based and road network based spatio–temporal generalizations of trajectories. Two methods and their simple SQL–implementations are presented for mining either local (micro) or global (macro) LSPs in such spatio–temporally generalized trajectories. In an attempt to speed up the local (micro) LSP mining method, the two methods are combined to a hybrid LSP mining method, which is able to rapidly find most of the local (micro) LSPs. The effectiveness of the different LSP methods is demonstrated through extensive experiments on both a real world data set, and a number of large–scale, synthetic data sets.

Future work is planed along several directions. First, as discussed, the hybrid LSP method, while is able to achieve significant speed–up compared to the local (micro) LSP mining method, it does not find all the local (micro) LSPs in the trajectories. Hence future work will consider to quantify (1) the speed–up of the hybrid LSP method, and (2) the fraction of the local (micro) LSPs found by the hybrid LSP method. Second, the large number of patterns discovered are difficult to analyze. To reduce this number, future work will consider the mining of a compressed patterns in trajectories [101]. Finally, future work will consider the partitioning of trajectories using index structures designed for trajectories, like in [55], to allow the distributed / parallel mining of LSPs.

Chapter 5

Cab–Sharing: An Effective, Door–To–Door, On–Demand Transportation Service

City transportation is an increasing problem. Public transportation is costeffective, but do not provide doortodoor transportation; This makes the far more expensive cabs attractive and scarce. This paper proposes a location–based Cab–Sharing Service (CSS), which reduces cab fare costs and effectively utilizes available cabs. The CSS accepts cab requests from mobile devices in the form of origin–destination pairs. Then it automatically groups closeby requests to minimize the cost, utilize cab space, and service cab requests in a timely manner. Simulation–based experiments show that the CSS can group cab requests in a way that effectively utilizes resources and achieves significant savings, making cab–sharing a new, promising mode of transportation.

5.1 Introduction

Transportation in larger cities, including parking, is an ever increasing problem that affects the environment, the economy, and last but not least our lives. Traffic jams and the hustle of parking take up a significant portion of our daily lives and cause major headaches. Solving the problem by extending the road network is a costly and non–scalable solution. A more feasible solution to the problem is to reduce the number of cars on the existing road network. To achieve this, collective / public transportation tries to satisfy the general transportation needs of larger groups in a cost–effective manner. While being cost–effective, the services offered by public transportation often (1) do not meet the exact, door–to–door transportation needs of individuals, (2) require multiple transfers and detours that significantly lengthen travel times, and (3) are limited in off–peak hours. For these reasons, the far more expensive service offered by cabs / taxis, which meet the exact transportation needs of individuals and also eliminates the problem of parking, are in great demand. To better satisfy this demand, this paper presents an LBS that makes use of simple technologies and tools to offer a new cost– and resource–effective, door–to–door transportation means, namely *cab–sharing*.

Collective transportation is not a new concept. It is encouraged and subsidized by transportation authorities all over the world. The optimization of collective transportation has also been considered. For an extensive list of research papers in this area, the reader is referred to [92]. Two papers, however, are worth highlighting. First, in [32], where the idea of the present research originates from, in an off–line fashion, long, shareable patterns are sought in historical trajectories of moving objects to aid an intelligent ride–sharing application. Second, in [17] an almost "personalized" transportation system is proposed that alters the fixed–line buss service to include variable itineraries and timetables. In comparison, the herein proposed CSS treats the optimization of collective transportation as a truly online process, and alters the inherently routeless transportation service offered by cabs.

5.2 Problem Statement

Let $\mathbb{R}^2$ denote the 2-dimensional Euclidean space, and let $\mathbb{T} \equiv \mathbb{N}^+$ denote the totally ordered time domain. Let $R = \{r_1, \ldots, r_n\}$ be a set of *cab requests*, such that $r_i = \langle t_r, l_o, l_d, t_e \rangle$, where $t_r \in \mathbb{T}$ is the *request time* of the cab request, $l_o \in \mathbb{R}^2$ and $l_d \in \mathbb{R}^2$ are the *origin* and *destination locations* of the cab request, and $t_e \geq t_r \in \mathbb{T}$ is the *expiration time* of the cab request, i.e., the latest time by which the cab request must be serviced. A cab request $r_i =< t_r, l_o, l_d, t_e >$ is *valid* at time t if $t_r \leq t \leq t_e$. Let $\Delta t = t_e - t_r$ be called the *wait time* of the cab request. Let a *cab–share* $s \subseteq R$ be a subset of the cab requests. A cab–share is valid at time t if all cab requests in s are valid at time t. Let $|s|$ denote the number of cab request in the cab–share. Let $d(l_1, l_2)$ be a distance measure between two locations l_1 and l_2. Let $m(s, d(.,.))$ be a method that constructs a valid and optimal pick–up and drop–off sequence of requests for a cab–share s and assigns a unique distance to this sequence based on $d(.,.)$. Let the *savings* p for a cab request $r_i \in s$ be $p(r_i, s) = 1 - \frac{m(s,d(.,.))/|s|}{m(\{r_i\},d(.,.))}$. Then, the *cab–sharing problem* is to find a disjoint partitioning $S = \{s_1 \uplus s_2 \uplus \ldots\}$ of R, such that $\forall s_j \in S$, s_j is *valid*, $|s_j| \leq K$, and the expression $\sum_{s_j \in S} \sum_{r_i \in s_j} p(r_i, s_j)$ is maximized.

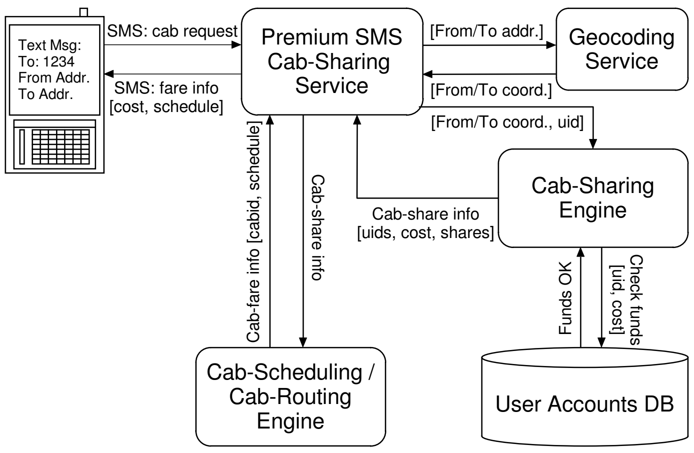


Figure 5.1: Cab–Sharing Service Components and Process.

5.3 Cab–Sharing Service

The Cab–Sharing Service (CSS) is depicted in Figure 5.1. Before using the CSS, a user signs up with the CSS and creates a prepaid user account to pay for the service. A registered user sends a cab request in the form of an origin–destination pair and an optional validity period of the request. In case no validity period is specified, the cab request is assumed to be valid from the time it is received until some default time limit has been reached. The requests are submitted via a mobile device by sending two address lines to a premium SMS service, called Premium Cab–Sharing Service. A more user–friendly specification of requests could include GPS–based localization of origins and/or the ability of users selecting origins and/or destinations from a list of frequent addresses, or even a voice–recognition–enabled, automatic call center. In either case, once received, the Premium Cab–Sharing Service sends the two address lines to a Geocoding Service, which validates them and returns the exact coordinates for them. Then these origin and destination coordinates, along with a user identifier are sent to a Cab–Sharing Engine. The Cab–Sharing Engine then, in an online fashion, automatically groups "closeby" requests into a cab–share to minimize the total transportation cost, thereby providing significant savings to the users of the service. Once a cab–share is constructed, the cost of the share is deducted from the account of every participant of the cab–share. Then, after receiving the information about the cab–share, the CSS forwards the information to the Cab–Scheduling / Cab–Routing Engine, which assigns cabs to cab–shares so that cab space is utilized and cab requests are serviced in a timely manner. Finally, the CSS sends an SMS to the user about the cab fare, such as cost and schedule of the fare. A web–demo of the CSS is available at: `http://www.cs.aau.dk/˜gyg/CSS/`.

5.4 Grouping of Cab Requests

Cab requests can be viewed as data points in spatio–temporal space. Partitioning n data points into k groups based on pairwise similarity of the data points according to a distance measure is referred to as the clustering problem, an extensively researched problem of computer science. Clustering is known to be NP–hard [49]. However, there are a number of efficient bottom–up and top–down methods that approximate the optimal solution within a constant factor in terms of a clustering criterion, which is expressed in terms of the distance measure.

Hence it is only natural to approach the cab–sharing problem as a clustering problem and adopt efficient approximations to the task at hand. For these approximation algorithms to converge to a local optima, an appropriate distance metric $d(.,.)$ between two cab requests and/or cab–shares needs to be devised. For $d(.,.)$ to be a metric for any three cab requests or cab–shares i, j, k is has to satisfy the following four conditions: $d(.,.) \geq 0$ (non–negativity), $d(i,i) = 0$ (identity), $d(i,j) = d(j,i)$ (symmetry), and $d(i,j) + d(j,k) \geq d(i,k)$ (triangular inequality).

While a clustering approach may find a near–optimal cab–sharing solution, it has several drawbacks. Since a cab request is only valid during a specific time interval, the set of valid cab request that can be considered for clustering is changing over time. While a hard time–constraint can be incorporated into a distance measure, the measure will not satisfy the triangular inequality requirement. An alternative approach could at every time step t retrieve the set of valid cab requests, and perform a partitioning–based clustering on the set according to some distance metric. However, since at any time instance t the number of valid cab requests n_t and the number of possible K-sized valid cab–shares are comparable, an iterative partitioning–based clustering approach would entail $O(n_t^2)$ distance calculations per iteration at every time instance t, making it infeasible in practice.

Since a cab request r_i has a request time t_r and an expiration time t_e it is natural to view it as a part of a data stream. When finding cab–shares in such a stream, two opposite approaches are obvious. In the first approach, referred to as the *lazy* approach, the grouping of requests is delayed as long as possible to find cab–shares with higher savings. In the second approach, referred to as the *eager* approach, request are grouped into cab–shares as early as possible to deliver a timely service. Next, the lazy version of a greedy, bottom–up grouping of cab requests is described. For ease of presentation, the described grouping method instead of maximizing savings, solves the equivalent problem of minimizing total travel cost; it is shown in Figure 5.2.

At any time t, valid cab requests can be divided into two sets: R_x, the set of valid requests that expire at time step t, and R_q, the rest of the valid requests that expire

```
(1)  procedure cabShare ($R$, $K$, min_saving)
(2)    $S \leftarrow \emptyset$
(3)    for $t = 1 \ldots T$
(4)      $\{R_x, R_q\} \leftarrow$ getValidRequests ($R$, $t$)
(5)      $E \leftarrow$ calculateE($R_x$, $\{R_x \cup R_q\}$)
(6)      while  ($|R_x| > 0$)
(7)        $\hat{e}_{min} \leftarrow \min_i \min_{k \leq K} \frac{\sum_1^k E(i,k)}{k}$
(8)        $\{i, k\} \leftarrow \mathrm{argmin}_i\ \mathrm{argmin}_{k \leq K} \frac{\sum_1^k E(i,k)}{k}$
(9)        $s \leftarrow \{r_i^1, r_i^2, \ldots r_i^k\}$
(10)       if $\hat{p}(r_i, s) <$ min_saving then break
(11)       $S \leftarrow \{S \cup s\}$
(12)       $E \leftarrow$ removeSfromE ($E$, $s$)
(13)     $S \leftarrow \{S \cup$ addRestAsSingles ($R_x$) $\}$
```

Figure 5.2: Lazy Version of a Greedy, Bottom–Up Grouping of Cab Requests.

some time after t (line 3). Given two cab requests r_i and r_j, let

$$e(r_i, r_j) = \frac{m(\{r_i, r_j\}, d(.,.)) - m(\{r_i\}, d(.,.))}{m(\{r_i\}, d(.,.))}$$

be the *fractional extra cost* of including r_j in r_i's cab fare. Using the pairwise fractional extra costs, the fractional extra cost of a cab share s w.r.t. r_i is estimated as $\hat{e}(s) = \sum_{r_j \in s} e(r_i, r_j)$, and the average savings for a cab request $r_j \in s$ is estimated as $\hat{p}(r_j, s) = 1 - \frac{1+\hat{e}(s)}{|s|}$. Furthermore, let r_i^k be the cab request that has the k-th lowest fractional extra cost w.r.t. r_i. In line 4, these fractional extra costs are calculated between cab requests in R_x and $\{R_x \cup R_q\}$ and for all $r_i \in R_x$ these fractional extra costs are stored in a 2–dimensional array E, such that $E[i,k] = e(r_i, r_i^k)$, i.e., E is sorted increasingly in row major order. Then, using only the lowest K entries for each cab request in E, in an iterative fashion the currently best (lowest amortized cost / highest savings) cab–share s is found (lines 7–9) and request in it are removed from consideration (line 12) by setting $E[r_j, .]$ and $E[., r_j]$ to some large value for all $r_j \in s$. This process continues until the currently best savings p_{max} is less than `min_saving`, at which point all the remaining cab request in R_x are assigned to their own "cab–share" resulting in no savings for them (line 13).

5.5 A Simple SQL Implementation

The grouping method for parameters `max_k` and `min_saving` can be easily implemented in a few SQL statements as described bellow. First, after geocoding, valid requests are stored in a table R_q = ⟨rid,tr,te,xo,yo,xd,yd⟩, where rid is a unique identifier for the request, tr and te are request and expiration times, and (xo,yo) and (xd,yd) are origin and destination coordinates. Then, using a temporal predicate, expiring requests are selected from R_q and stored in a table R_x with the same schema. Finally, a distance function d(x1,y1,x2,y2) is defined between two 2D coordinates. Then, the fractional extra cost function e for the origin and destination coordinates of the requests ri and rj can be defined in SQL–99 [87] as follows.

```
CREATE FUNCTION e(rixo FLOAT, riyo FLOAT,
                  rixd FLOAT, riyd FLOAT,
                  rjxo FLOAT, rjyo FLOAT,
                  rjxd FLOAT, rjyd FLOAT)
RETURNS FLOAT
BEGIN
  DECLARE ri_dist, ed FLOAT
  SET ri_dist = d(rixo, riyo, rixd, riyd)
  SET ed = d(rjxo, rjyo, rixo, riyo)
         + d(rjxd, rjyd, rixd, riyd)
  RETURN (ed / ri_dist)
END
```

5.5.1 STEP 1: Calculating Fractional Extra Costs

After creating a table E = ⟨ri,rj,e⟩ to store the fractional extra costs, the fractional extra costs between the requests ri in R_x and rj in R_q can be calculated in SQL–99 as follows.

```
INSERT INTO E (ri, rj, e)
SELECT x.rid ri, q.rid rj,
       e(x.xo,x.yo,x.xd,x.yd,q.xo,q.yo,q.xd,q.yd)
FROM R_x x, R_q q
WHERE x.rid <> q.rid
  AND e(x.xo,x.yo,x.xd,x.yd,q.xo,q.yo,q.xd,q.yd) <= 1
```

The first condition in the WHERE clause excludes the fractional extra cost of ri with itself, which is 0 by definition. The reason for doing so is to avoid falsely identifying ri on its own as the currently best (lowest amortized cost = 0 / highest savings = 1) “cab–share” in the processing steps to follow. The second condition in the WHERE clause is a pruning heuristic that excludes (ri,rj) request combinations from E where the fractional extra cost exceeds 1, in which case neither ri nor any cab–share containing ri can benefit from including rj.

5.5.2 STEP 2: Calculating Amortized Costs

Relational Database Management Systems (RDBMSs) are *set* oriented and the inherently declarative SQL language does not provide adequate support to implement operations on *sequences*, e.g., cumulative sum. Procedural language constructs that allow iteration over the elements of a sequence do exist in SQL, but are implemented less efficiently. Hence, programmers normally revert to other procedural languages to perform such operations. Nevertheless, the calculation of cumulative sum can be implemented in SQL in a declarative fashion using a self–join. Hence, after creating a table $\texttt{AE} = \langle \texttt{ri,rj,ae,k} \rangle$, the summations on line 7 and 8 of the grouping method in Figure 5.2, i.e. the amortized costs can be calculated in a single SQL–99 statement as follows.

```
INSERT INTO AE (ri, rj, ae, k)
SELECT a.ri, a.rj, (SUM(b.e)+1)/(COUNT(*)+1) ae, COUNT(*)+1 k
FROM E a, E b
WHERE a.ri = b.ri AND a.e >= b.e
GROUP BY a.ri, a.rj
HAVING COUNT(*)+1 <= max_k
```

The WHERE clause for every $(\texttt{ri},\texttt{rj})$ combination from the table a assigns a *set* of $(\texttt{ri},\texttt{rj})$ combinations from the table b, such that ri's match in the two tables and the fractional extra costs values (e) in table b are less than or equal to the values in table a. The latter condition in a sense imposes an *order* on the *set*. The aggregation for each such $(\texttt{ri},\texttt{rj})$ combination (set) is achieved through the GROUP BY clause. The corresponding aggregates ae and k are calculated by the two expressions in the SELECT statement, where ae is the amortized cost of the best cab–share of size k that contains requests ri and rj. Finally, the HAVING clause excludes cab–shares larger than size max_k from further consideration. Note that, while the calculations of sequence–oriented cumulative aggregates, for example amortized cost (ae) are simple to express in SQL, the computation performed is not optimal. While the computational complexity of sequence–oriented cumulative aggregates is $O(n)$, for a sequence of length n, the complexity of the above method based on self–joins is $O(n^2)$. Nevertheless, the self–join based simple SQL implementation can process in real–time up to 100,000 requests per day.

5.5.3 STEP 3: Selecting the Best Cab–Share

After creating a table $\texttt{CS} = \langle \texttt{sid,rid} \rangle$ to store the cab–shares, one can select the savings, b_savings, the size, b_k, and, conditioned on the min_savings parameter, store the requests of the currently best cab–share (with ID = s) in two SQL–99 statements as follows.

```
SELECT ri, (1-ae), k INTO b_rid, b_savings, b_k
FROM AE ORDER BY ae LIMIT 1

INSERT INTO CS (sid, rid)
SELECT s sid, b_rid rid FROM AE
UNION
SELECT s sid, rj rid FROM AE WHERE k <= b_k AND ri = b_rid
```

5.5.4 STEP 4: Pruning the Search Space

Since a cab request can only be part of a single cab–share, if the current best cab–share meets the minimum saving requirement, and is added to `CS`, the requests in it have to be discarded from further considerations for finding cab–shares in the future. This can be achieved by deleting tuples from the `E` table that refer to the requests in question. The SQL–99 statement for this is as follows.

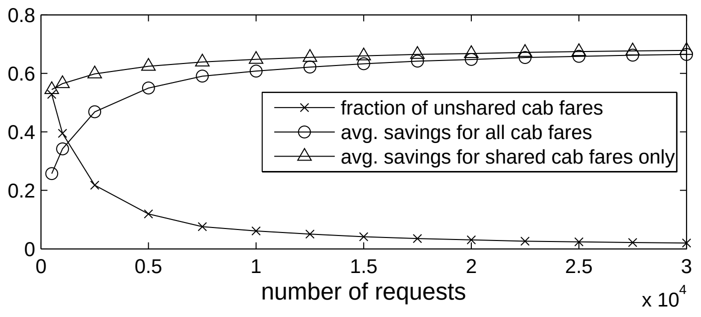
```
DELETE FROM E
WHERE ri IN (SELECT rid FROM CS WHERE sid = s)
   OR rj IN (SELECT rid FROM CS WHERE sid = s)
```

5.5.5 Periodic, Iterative Scheduling of Cab Requests

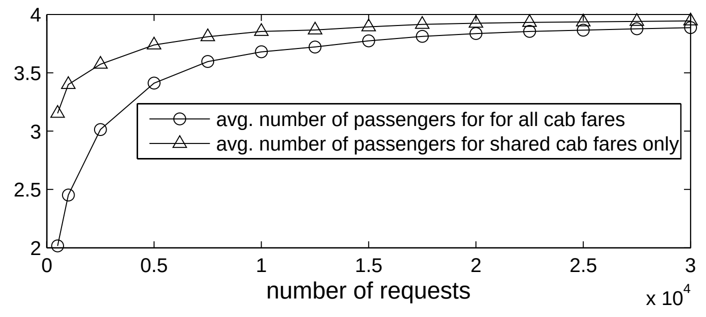
All cab requests in `R_x` are grouped in an iterative fashion by executing steps 2 through 4 until (1) there are no more cab–shares that meet the minimum savings requirement, or (2) all requests in `R_x` has been assigned to some cab–share. The loop iterating through these steps is placed in a stored procedure. Using the automatic task scheduling facilities of the operating system, `cron` in Linux or `Task Scheduler` in Windows, this stored procedure is executed periodically. Keeping the period of the executions of the stored procedure short (frequency of executions high) has several advantages. First, the shorter the period, the longer requests can be delayed until they *have to be* grouped into cab–shares, giving the requests more opportunities to end up in a good cab–share. In effect, the set of expiring requests is composed of requests that will expire before the next scheduled execution of the stored procedure. Second, smaller sets of expiring requests means smaller `E` and `AE` tables, which are cheaper to maintain during the iterations of a single execution of the stored procedure.

5.6 Experiments

To test the proposed methods, cab request data was simulated using ST–ACTS, a spatio–temporal activity simulator [31]. Based on a number of real world data sources, ST–ACTS simulates realistic trips of approximately 600,000 individuals in the city of Copenhagen, Denmark. For the course of a workday, out of the approximately 1.55



(a) Average Savings and Fraction of Unshared Cab Fares.



(b) Cab Space Utilization.

Figure 5.3: Performance Evaluations for Varying Number of Cab Requests.

million generated trips, approximately 251,000 trips of at least 3–kilometer length were selected and considered as potential cab requests. Experiments were performed for various maximum cab–share sizes $K \in [2, 8]$, wait times $\Delta t \in [1, 20]$, and cab request densities, i.e., various–sized, random subsets of the set of potential cab requests. Figures [5.3(a), 5.3(b)], in which the units on the x scale is 10,000 cab requests, show some of the results for parameter settings $K = 4$, $\texttt{min\_saving} = 0.3$, and $\Delta t = 15$ minutes (common for all cab requests).

Figure 5.3(a) shows (1) the fraction of unshared cab requests, and (2) the average savings for *all fares*, and for the *shared fares only*. As the density of cab requests increases, and hence the likelihood of two individuals wanting to travel around the same time from approximately the same origin location to approximately the same destination location increases, the number of cab–shares, meeting the required minimum savings also increases. Consequently, the fraction of unshared requests decreases to a point where only about 2% of the cab requests cannot be combined into cab–shares that meet the required minimum savings. Similarly, as the density of the cab requests increases, the average savings for fares also increases up to a point where the average savings per fares is 0.66 ± 0.11 considering all the fares, and is 0.68 ± 0.06 considering shared fares only. In other words, the CSS is able to group cab requests in a way such that the cost of 97.5% of the cab fares can be reduced by two thirds on average. Figure 5.3(b) shows how well cab space is utilized. As the density of cab requests increases, the average number of passengers per cab also increases up to a point where the average number of passengers per cab is 3.89 ± 0.49 considering all the fares, and is 3.94 ± 0.27 considering shared fares only.

Due to space limitations, the detailed results of the experiments showing the effects of parameters Δt and K are omitted, but they can be summarized as follows. The Δt experiments confirm that due to the linear relationship between Δt and the resulting spatio–temporal density of *valid* cab requests, there exists a correspondence between the above results and the omitted results, i.e., since the spatio–temporal density of valid cab requests for 15,000 requests with $\Delta t = 15$ minutes are about the same as for 30,000 requests with $\Delta t = 7.5$ minutes, the average savings and cab utilization are approximately the same in both cases. The K experiments confirm that under a fixed cab request density, both the savings and cab utilizations saturate. In the case of 30,000 requests for $K \in [2, 8]$, the average savings for shares gradually increases from 0.47 to 0.79 and the average number of passengers for shares gradually increases from 2 to 6.1.

The savings come at the expense of some delay in the CSS when meeting the end–to–end transportation needs of its users. There are three sources for this delay. First, the *grouping time*, i.e., the time that a user has to wait until his/her requests is grouped into a cab–share, which is upper bounded by the *wait time* parameter of the requests. Second, the *pickup time*, i.e., the extra time a user has to wait because some of the other members of the cab–share need to be picked up before him/her. Finally, the *additional travel time*, i.e., the extra time the cab–fare takes due to the increased length of the shared part of the cab–fare. Because no realistic simulation of the transportation phase of the CSS was performed, the delay incurred due to the latter two sources has been evaluated in terms of extra distances relative to the length of the original requests. Due to the close to constant results for various cab request densities, the measurements on the delay due to the above three sources can be (independently from the cab request density) summarized as follows. The average grouping time is 11.7 minutes with a standard deviation of 5.9 minutes. The average pickup time is equivalent to 10.7 ± 13.5% of the length of the original request. Given the average length of requests of 4.95 kilometers, and assuming an average transportation speed of 40 km/h in the city, the average pickup time is approximately 0.8 ± 1.1 minutes. The average additional travel time is equivalent to 7.9 ± 10.1% of the length of the original request, or is approximately 0.6 ± 0.9 minutes. Hence in total, the approximate additional service delay an average CSS user experiences compared to using a conventional cab service is approximately 12.1 ± 7.9 minutes, arguably a small price to pay for the savings.

5.7 Conclusions and Future Work

Motivated by the need for a novel transportation alternative that is convenient, yet affordable, this paper proposes a new LBS, namely a Cab–Sharing Service (CSS). To achieve the desired reduction in transportation cost, the paper proposes a greedy grouping algorithm, along with a simple but effective SQL implementation, that optimally groups "close by" requests into cab–shares. Experiments on simulated, but realistic cab request data show that in exchange of a short (5–15 minute) wait time, the CSS can group together requests in a way that effectively utilizes resources and provides significant savings to the user.

Future work is planned along several directions. First, since it is natural to view the incoming requests as a data stream, the CSS is being implemented using an in–memory Data Stream Management System (DSMS) [106]. Second, the cab–sharing problem is a hard optimization problem, hence investigating new heuristics for it is planned. Third, while the proposed greedy method is computationally efficient, a number of improvements to it are possible, for example to use spatial indices to prune the search space of possible cab–share candidates. Finally, while not considered here, the optimization of the Cab–Scheduling / Routing Engine through spatio–temporal cab request demand prediction is planned.

Chapter 6

Highly Scalable Trip Grouping for Large–Scale Collective Transportation Systems

Transportation–related problems, like road congestion, parking, and pollution are increasing in most cities. In order to reduce traffic, recent work has proposed methods for vehicle sharing, for example for sharing cabs by grouping "closeby" cab requests and thus minimizing transportation cost and utilizing cab space. However, the methods proposed so far do not scale to large data volumes, which is necessary to facilitate large–scale collective transportation systems, e.g., ride–sharing systems for large cities.

This paper presents highly scalable *trip grouping algorithms*, which generalize previous techniques and support input rates that can be orders of magnitude larger. The following three contributions make the grouping algorithms scalable. First, the basic grouping algorithm is expressed as a continuous stream query in a data stream management system to allow for a very large flow of requests. Second, following the divide–and–conquer paradigm, four space–partitioning policies for dividing the input data stream into sub–streams are developed and implemented using continuous stream queries. Third, using the partitioning policies, parallel implementations of the grouping algorithm in a parallel computing environment are described. Extensive experimental results show that the parallel implementation using simple adaptive partitioning methods can achieve speed–ups of several orders of magnitude without significantly effecting the quality of the grouping.

6.1 Introduction

Transportation–related problems, like congestion, parking, and pollution are increasing in most cities. Waiting in traffic jams not only degrades the quality of social life, but according to estimates, the economic loss caused by traffic jams in most countries is measured in billions of US dollars yearly. Parking is also a serious problem. According to estimates, in the inner cities of some larger cities as high as 25% of the drivers on the road are only looking for empty parking places; which again causes unnecessary congestion. Finally, the increasing number of vehicles idling on the roads results in an unprecedented carbon emission, which has unquestionably negative effects on the environment.

By reducing the number of vehicles on the roads, Collective Transportation (CT) clearly provides a solution to these problems. Public transportation, the most common form of CT, tries to meet the general transportation demands of the population at large. By generalizing the transportation needs, the individual is often inconvenienced by long wait times at off–peak hours or between connections, and a limited number of access points (bus / metro / train stops) from which the individual is forced to use other methods of transportation (walking / bicycling / using a private car). Ride–sharing, or car pooling, another form of CT is becoming widespread in metropolitan areas. In almost all cases, ride–sharing is encouraged by local transportation authorities by facilitating car pool lanes that are only accessible to multiple–occupancy vehicles and by eliminating tolls on bridges and highways for these vehicles. Despite the encouragement there is a tremendous amount of unused transportation capacity in the form of unoccupied seats in private vehicles. This fact can mainly be attributed to the lack of effective systems that facilitate large–scale ride–sharing operations. The systems that do exist [10, 52, 94] are either 1) offered from a limited number access points due to the system infrastructure constraints, 2) have inadequate methods for the positioning of trip requests and / or vehicles, or 3) have either inefficient or ineffective methods for matching or grouping trip requests and trip offers.

In a recent paper [33], yet another form of CT, namely cab–sharing was proposed to use unoccupied cab space to reduce the cost of transportation, ultimately resulting in direct savings to the individual. The described Cab–Sharing System (CSS) overcomes most of the above limitations of existing ride–sharing systems. In particular, at the heart of the system is a trip grouping algorithm that is able to find subsets of closeby trip requests, which can be grouped into collective cab fares to minimize the transportation cost, or equivalently maximize the savings to the user. Using a simple implementation in standard SQL, assuming a reasonable number (high spatio–temporal density) of trip requests, the trip grouping algorithm was demonstrated to be able to group trip requests effectively. The trip grouping algorithm can be generalized to facilitate other CT systems, e.g., a ride–sharing system. However, as it is demonstrated in the present paper, due to its algorithmic complexity, the grouping algorithm scales poorly as the volume of trip requests increases. This limits its applicability to facilitate large–scale CT systems, such as a metropolitan or nation–wide ride–sharing system.

To make the trip grouping algorithm scale to input rates several orders of magnitude larger than in a typical cab–sharing application, this paper makes the following three contributions. First, using a Data Stream Management System (DSMS), SCSQ [106], the trip grouping algorithm is expressed as a continuous stream query to allow for continuous processing of large trip request streams. Second, following the divide–and–conquer paradigm, *static* and *adaptive* versions of two space–partitioning policies (*point quad* and *KD partitioning*) for dividing the input data stream into sub–streams are developed and implemented using continuous stream queries. Finally, using the partitioning policies, the grouping algorithm is implemented using a data stream management system in a parallel computing environment. The parallelization of the implementation is facilitated by using an extension of the query language in which processes are query language objects. Extensive experimental results show that the parallel implementation using simple partitioning methods can achieve speed–ups of several orders of magnitude without significantly affecting the quality of the grouping. In particular, an adaptive partitioning method called *adaptive KD partitioning* achieves the best overall performance and grouping quality.

The remainder of this paper is organized as follows. Section 6.2 reviews related work. Section 6.3 defines the vehicle–sharing problem, reviews the operational aspects of a recently proposed Cab–Sharing System (CSS), describes and analyzes a trip grouping algorithm that solves the vehicle–sharing problem and is employed to facilitate the CSS. Furthermore, a new Ride–Sharing System (RSS) is proposed, and the trip grouping algorithm is adapted to meet the application requirements of the proposed RSS. Section 6.4, describes the main contributions of the paper in making the trip grouping algorithm highly scalable, hence applicable in large–scale CT system, such as an RSS. Section 6.6 describes and analyzes the results of the experiments that were conducted to measure the performance of the proposed highly scalable trip grouping algorithm. Finally, Section 6.7 concludes and points to future research directions.

6.2 Related Work

The optimization of CT has been studied in the scientific community for years [17, 92]. However, with the exception of the work presented in [33], on which the present paper is based, it is believed that no previous research has considered the online grouping of trip requests. The problem of grouping n objects into a number of groups is in general referred to as the clustering problem, which is an extensively researched problem in computer science. However, the unique requirements of the problem of vehicle–sharing mean that general clustering techniques have limited applicability.

Vehicle–sharing as a form of CT has been considered in industrial and commercial settings. For example, most taxi companies in larger cities have been offering the possibility of shared transportation between a limited number of frequent origins and destinations. Scientifically very little is known about the computational aspects of these vehicle–share operations. However, the computer systems supporting such operations are likely to be semi–automatic, to perform batch–grouping of requests, and to suffer from scalability problems. In comparison, the trip grouping algorithm proposed in this paper is automatic, performs online–grouping or requests, and is highly scalable.

More automatic systems that perform online optimization of vehicle–sharing also exist [10, 52, 94]. These systems however perform a computationally easier task. They either match pairs of trip requests only [52] or are offered from / between a limited set of locations [10, 52, 94]. Additionally, the high volume scalability of these systems has not been demonstrated. Nonetheless, the analysis in [93] and the existence of these systems are evidence that the problem considered by the paper is real and has industrial applications.

Parallel processing of high volume data streams has been considered by several papers [11, 54, 64, 68, 102, 105, 106]. To parallelize continuous stream queries, [54, 105, 106] decompose the computation of a single continuous stream query into a partition, a compute, and a combine phase. In [54], the distributed execution strategies are expressed as *data flow distribution templates*, and queries implementing the three phases are specified in separate scripts. In contrast, in [106], through the use of stream operators, the implementations of the three phases become part of a single parallel, continuous stream query, which is elegantly expressed in the query language. The present work utilizes the stream processing engine and query language in [106] to express and evaluate different (parallel) stream processing strategies for an RSS.

In [54], two different stream partitioning strategies are considered: *window distribute* (WD) and *window split* (WS). In WD, entire logical windows are distributed among compute nodes. In WS, an operator dependent stream split function splits logical windows into smaller ones and assigns them to particular compute nodes for processing. WS has several advantages over WD. First, in applications where the execution time of the stream query scales superlinearly with the size of the logical window, WS provides superior parallel execution performance over WD. Second, in real–time response systems, where the query scales superlinearly, WD is not applicable as it can introduce severe delays in the result stream. Third, in systems where the quality of the results that are computed in parallel are highly dependent on the tuples inside the logical windows of the compute nodes, WD provides inferior results in quality over WS, because individual tuples are not considered in the partition phase. As all of the above three conditions hold in the case of vehicle–sharing, WD is clearly not of interest. WS is similar to the spatial stream partitioning methods presented in this paper in the sense that both presented partitioning methods consider individual tuples in the partitioning process. However, in the static cases no windows are formed over the stream, but rather tuples are assigned to compute nodes based on a general partitioning table. In contrast, in the adaptive cases windows are formed over the stream, the partitioning table is periodically updated based on the tuples in a window, and then tuples are assigned to compute nodes the same way as in the static case.

Database indices support the efficient management and retrieval of data in large databases. In particular, spatial indices support efficient retrieval of spatial objects, i.e., objects that have physical properties such as location and extent. Spatial indices can be divided into two types: *data partitioning* and *space partitioning* spatial indices [82]. The partitioning mechanisms used in spatial indices have a close relation to the partitioning performed in the present paper.

Data partitioning indices usually decompose the space based on Minimum Bounding Rectangles (MBRs). A primary example is the R–tree that splits space with hierarchically nested, and possibly overlapping Minimum Bounding Rectangles (MBRs) [45]. However, for the application at hand, data partitioning schemes are not well suited for several reasons. They often use a non–disjoint decomposition of space. Consequently, a naïve partitioning based on MBRs could either assign requests to several partitions, and hence later to several shares, or could assign requests from a region where several MBRs overlap to several partitions, thereby potentially eliminating good matches. While a disjoint partitioning of space could be derived based on the MBRs, the computation to derive such a partitioning would be complex and potentially expensive, and the derived partitions will, most likely not be balanced.

On the other hand, space partitioning indices decompose the entire space into disjoint cells. These disjoint cells can be based on a regular grid, or on an adaptive grid. Regular grids can result in empty partitions because of skewed data distributions. Hence, a regular grid is not well–suited for the application at hand as it does not support load–balancing.

Quad–trees partition the space into four quadrants in a recursive fashion [23]. Quad–trees divide each region into four equally sized regions, while *point quad trees* allow the size of the regions to be dynamic. Quad–trees have been extended to higher dimensions also. One of the space partitioning methods used in this paper is quite similar to a 1–level deep, four dimensional so–called “Point Quad–tree” [82] with the exception that in the herein considered space partitioning method a split point is not necessarily a data point. The k–d–tree is a space partitioning spatial index that hierarchically divides each dimension into two along each of the k dimensions [6, 7].

The other partitioning method used in this paper corresponds to a 1–level deep, four dimensional k–d–tree.

6.3 Vehicle–Sharing

Large–scale, personalized, on–demand CT systems need efficient and effective computer support. Systems providing this support have two aspects. First, an operational aspect as to how information is communicated between the user and the service provided by the system, and how trip requests are processed. Second, a computational or algorithmic aspect as to how the optimization of CT is performed. The following subsections study an existing CT system and propose a new one. Section 6.3.1 formalizes the vehicle–sharing problem, adopted from [33]. Section 6.3.2 describes the operational aspects of a Cab–Sharing System (CSS)–an instance of a CT system in which the shared vehicles are cabs. Section 6.3.3 describes the computational or algorithmic aspects of the trip grouping algorithm employed in the CSS. Section 6.3.4 describes the problems that arise when the trip grouping algorithm is applied in larger scale CT systems. Section 6.3.5 proposes a Ride–Sharing Service (RSS) and describes its operational requirements. Finally, Section 6.3.6 describes how the trip grouping algorithm in Section 6.3.3 can be modified to meet these requirements.

6.3.1 The Vehicle–Sharing Problem

Let $\mathbb{R}^2$ denote the 2-dimensional Euclidean space, and let $\mathbb{T} \equiv \mathbb{N}^+$ denote the totally ordered time domain. Let $R = \{r_1, \ldots, r_n\}$ be a set of *trip requests* $r_i = \langle t_r, l_o, l_d, t_e \rangle$, where $t_r \in \mathbb{T}$ is the *request time*, $l_o \in \mathbb{R}^2$ and $l_d \in \mathbb{R}^2$ are the *origin* and *destination locations*, and $t_e \geq t_r \in \mathbb{T}$ is the *expiration time*, i.e., the latest time by which the trip request must be accommodated. A trip request $r_i =< t_r, l_o, l_d, t_e >$ is *valid* at time t if $t_r \leq t \leq t_e$. $\Delta t = t_e - t_r$ is called the *wait time* of the trip request. A *vehicle–share* $s \subseteq R$ is a subset of the trip requests. A vehicle–share is valid at time t if all trip requests in s are valid at time t. Let $|s|$ denote the number of trip request in the vehicle–share. Let $d(l_1, l_2)$ be a distance measure between two locations l_1 and l_2. Let $m(s, d(.,.))$ be a method that constructs a valid and optimal pick–up and drop–off sequence of requests for a vehicle–share s and assigns a unique distance to this sequence based on $d(.,.)$. Let the *savings* p for a trip request $r_i \in s$ be $p(r_i, s) = 1 - \frac{m(s,d(.,.))/|s|}{m(\{r_i\},d(.,.))}$. Then the *vehicle–sharing problem* is defined as follows.

Definition 6 *For a given* ***maximum vehicle–share size*** K, *and* ***minimum savings*** min_savings $\in [0, 1]$, *the vehicle–sharing problem is to find a disjoint partitioning* $S = \{s_1 \uplus s_2 \uplus \ldots\}$ *of* R, *such that* $\forall s_j \in S$, s_j *is* valid, $|s_j| \leq K$, *and the expression* $\sum_{s_j \in S} \sum_{r_i \in s_j} p(r_i, s_j)$ *is maximized under the condition that* $\forall r_i \in s_j$ $p(r_i, s_j) \geq$ *min_savings* or $\{r_i\} = s_j$.

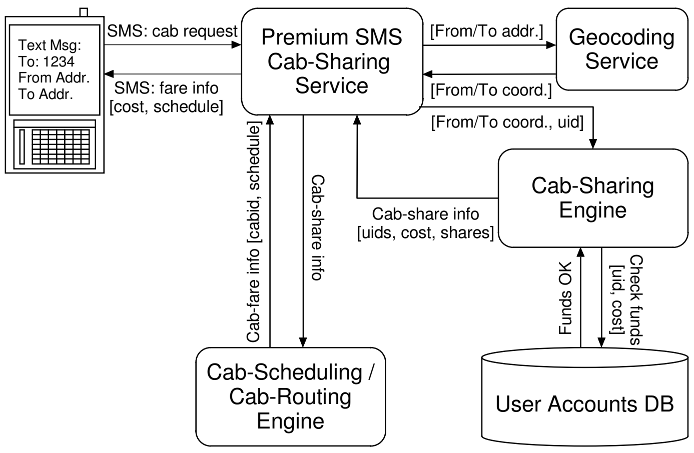


Figure 6.1: Cab–Sharing Service Components and Process.

6.3.2 Overview of the Cab–Sharing System

The Cab–Sharing System (CSS) proposed in [33] is a Location–Based Service (LBS) in the transportation domain. In its most simple form, it is accessible to the user via a mobile phone through an SMS interface. The components and operation of the CSS is depicted in Figure 6.1 and can be described as follows. The user inputs two addresses with an optional maximum time that s/he is willing to wait. The service in turn then:

1. geocodes the addresses,
2. calculates an upper bound on the cost of the fare,
3. validates the user's account for sufficient funds,
4. submits the geocoded request to a pool of pending requests,
5. within the maximum wait time period finds a nearly optimal set of "closeby" requests using a number of heuristics (described in Section 6.3.3),
6. delivers the information about the set (request end points, and suggested pickup order) to the back–end cab dispatch system,
7. delivers information about the fare (estimated time or arrival, cost, savings, etc...) to the involved users.

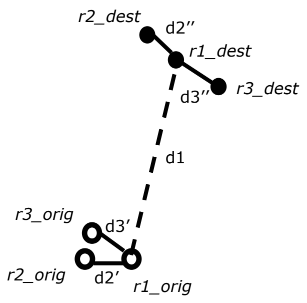
6.3.3 A Trip Grouping Algorithm

Finding the optimal solution to the vehicle–sharing problem is computationally difficult. Given n requests, the number of possible disjoint partitionings, where the size of the vehicle–shares is exactly K is:

$$\binom{n}{K} \times \binom{n-K}{K} \times \cdots \times \binom{2K}{k} \times \binom{k}{k} = \frac{n!}{\lceil n/K \rceil \times K!}.$$

In the case of n = 100 and K = 4, this expression evaluates to a number that has 155 digits. The number of possible disjoint partitions, where the size of the vehicle–shares is at most $K = 4$ is even larger. Clearly, evaluating all possible options and selecting the most optimal one is not a feasible approach. Instead, the Trip Grouping (TG) algorithm at the heart of the CSS tries to derive a nearly optimal solution by employing a number of heuristics and approximations. The steps of the TG algorithm along with the applied heuristics and approximations are described next.

1. Distinguish between the set of expiring trip requests (R_x) and all valid requests (R_q). Wait with mandatory grouping of trip requests until expiration time. Since a request can also be grouped into a vehicle–share before its expiration time with another expiring request, this *lazy* heuristic, does not make the algorithm miss out on an early, cost–effective grouping for the request, but rather gives the requests more opportunities to be part of one.
2. Based on the distance measure $d(.,.)$, define a pair–wise *fractional extra cost* (FEC) between two requests and calculate it for every pair of expiring and valid requests. In the TG algorithm the fractional extra cost between two requests r_i and r_j (w.r.t. r_i) is defined as $FEC(r_i, r_j) = \frac{d(r_i.l_o, r_j.l_o) + d(r_i.l_d, r_j.l_d)}{d(r_i.l_o, r_i.l_d)}$. In the case when the distance measure $d(.,.)$ is the Euclidean distance, the calculations of fractional extra costs between three requests r_1, r_2, and r_3 (w.r.t. r_1) are shown in Figure 6.2. Note that the defined fractional extra cost is an upper bound on the true fractional extra cost, as there may be a shorter route than to serve the requests in the order assumed by the fractional extra cost calculation, i.e., $r_j.l_o \rightarrow r_i.l_o \rightarrow r_i.l_d \rightarrow r_j.l_d$.
3. Consider the best K–sized vehicle–share for an expiring request $r_i \in R_x$ to be composed of the first K requests with lowest FEC for r_i. This heuristic assumes that pair–wise fractional extra costs are additive.
4. Estimate the Amortized Cost (AC) of a vehicle–share s (w.r.t. r_i) as the normalized cumulative sum of FECs as: $AC(r_i, s) = \frac{1 + \sum_{r_j \in s} FEC(r_i, r_j)}{|s|}$. This heuristic assumes that there exists an optimal pick–up and drop–off sequence for requests in s, such the cost of this sequence $m(s, d(.,.)) \leq AC(r_i, s) * d(r_i.l_o, r_i.l_d)$.
5. Greedily group the “best”, maximum K-sized vehicle–share that has the minimum amortized cost over all expiring trip requests. This heuristic is greedy because it possibly assigns a not–yet–expiring request r_j to a vehicle share of



FEC(r1,r2) = (d2' + d2'') / d1

FEC(r1,r3) = (d3' + d3'') / d1

AC({r1,r2,r3})=
(1 + FEC(r1,r2) + FEC(r1,r3)) / 3

Figure 6.2: Illustration of Fractional Extra Cost (FEC) and Amortized Cost (AC) Calculations w.r.t. Request r1.

an expiring request, without considering what the current or even future best vehicle–share would be for r_j.

6. Remove requests that are part of the “best” vehicle–share from further consideration.
7. Repeat steps 2 through 7 as long as the “best” vehicle–share meets the minimum savings requirement.
8. Assign remaining trip requests to their own (single person) “vehicle–shares”.

Even though the TG algorithm is based on heuristics, estimations and assumptions, in [33], it has been found to effectively optimize the vehicle–sharing problem. Furthermore, while some assumptions about extra costs for vehicle–shares do not hold in all cases, the combination of the approximations and assumptions result in an estimated cost for the vehicle–shares that is higher than the true minimum cost if the optimal pick–up and drop–off sequence is considered.

6.3.4 Problems with Large–Scale CT Systems

Unfortunately, the TG algorithm cannot be naively applied to facilitate a large–scale CT system, such as a ride–sharing system. Since the TG algorithm needs to calculate the pairwise fractional extra costs between expiring requests and all requests in the queue, the algorithmic complexity of the TG algorithm is $\Omega(n^2)$. In [33] an simple but effective implementation of the TG algorithm was able to handle loads of up to 50,000 requests per day, during which at peak traffic hours the number of requests within 10 minutes was at most 2,500. However as input sizes increase the execution times of any serial implementation of the TG algorithm will reach a point where continuous grouping is not possible, i.e., the algorithm is not able to find nearly optimal groups for all the expiring request before they actually expire. This is demonstrated

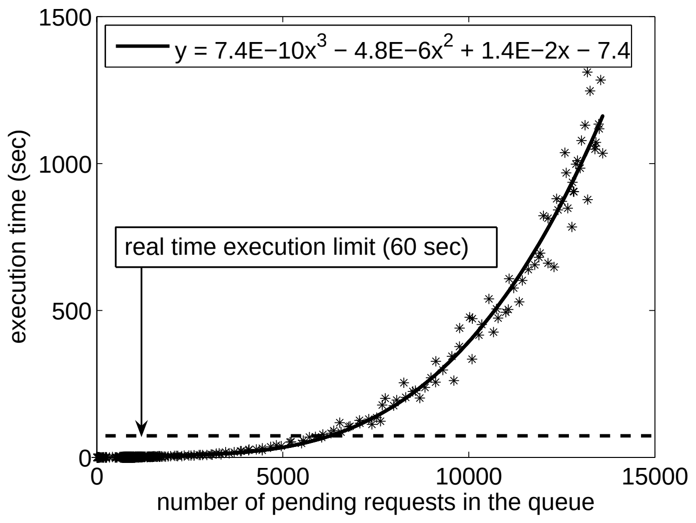


Figure 6.3: Scalability Problems of the General Trip Grouping Algorithm.

in Figure 6.3, where, using a highly efficient implementation of the TG algorithm, a load of 250,000 requests with common wait times of 10 minutes are grouped minute–by–minute. This efficient implementation of the TG algorithm is able to keep up with the request flow most of the time, but when the number of pending requests exceeds about 5,200 (during rush hour), it is not able to find groups for the expiring requests within the allowed execution time of 60 seconds. In the example the *grouping cycle time* of the TG algorithm is 60 seconds, i.e. the algorithm is responsible for grouping the request that will expire within the next 60 seconds. Altering this grouping cycle time does not eliminate the lagging of the algorithm in case of large input sizes. In fact, independent of the grouping cycle time, throughout the validity period of a request, the request is considered as an expiring request exactly once, at which point is compared to all other requests. Figure 6.3 also reveals that the computational complexity of the implementation of the TG algorithm is $O(n^3)$. This is due to the fact that, as described by the third heuristic in Section 6.3.3, the best K–sized vehicle–share is composed of the first K requests with lowest FEC for an expiring request. This necessitates a linear–time top–K selection for each expiring request, making the algorithmic complexity of the TG algorithm $\Omega(n^3)$. Consequently, the above described scalability problems severely limit the applicability of the TG algorithm in a large–scale CT system.

6.3.5 Ride–Sharing Application Requirements

Ride–sharing is a type of vehicle–sharing, where private vehicles are used as transportation. This fact represents additional requirements on solution to the general trip–sharing problem. In the context of ride–sharing there are *ride–requests* and *ride–offers*. Ride–requests are synonymous to trip requests both in form and semantics, with the exception that ride–requests do not necessarily have to be served. Ride–offers in addition to the attributes of a trip request have at least three more important attributes. The first attribute specifies whether or not the offering person is willing to leave his / her vehicle behind. A person offering a ride with willingness of leaving his / her vehicle behind is either willing to take alternate modes of transportation or relies on the efficient operation of the ride–sharing system for future trips until he / she returns to his / her vehicle. A person not willing to leave his her vehicle behind values or needs his / her independence throughout the day. The second attribute specifies a *maximum relative extra cost* the offering person is prepared to incur. Finally, the third attribute specifies the *maximum number of additional passengers* the offering person's vehicle can accommodate.

6.3.6 Application of the TG Algorithm in an RSS

It is clear that the TG algorithm cannot be applied in its current form for a ride–sharing application. However, a few simple modifications can make it applicable. First, in the context of ride–sharing the ride offering person would like to leave as soon as the "best" vehicle–share that can be constructed meets the maximum relative extra cost requirements of the ride–offer. Hence, it makes sense to prioritize the order of greedy grouping based on the time the ride–offers have been present in the system. Second, because maximum relative extra cost requirements are defined by ride–offers individually, in every grouping cycle (execution of the TG algorithm) the "best" vehicle–share for *all* ride–offers needs to be considered. Third, every vehicle–share needs to fulfill the following two conditions: 1) it can contain only one ride–offer where the offering person is not willing to leave his / her vehicle behind, and 2) it has to contain at least one ride–offer of any type. To fulfill the above conditions it is enough to distinguish between two different sets: 1) the set of ride–offers of either type $\{R_o^{\bar{o}} \cup R_o^o\}$, and 2) the joint set of ride–request and ride–offers where the offering person is willing to leave his / her vehicle behind $\{R_r \cup R_o^o\}$. Associating these sets to sets used by the TG algorithm as $R_x = \{R_o^{\bar{o}} \cup R_o^o\}$ and $R_q = \{R_r \cup R_o^o\}$, the vehicle shares constructed by the TG algorithm fulfill the above two conditions.

Obviously, the modifications to the TG algorithm that are necessary to facilitate the proposed RSS are straight–forward. However, to preserve clarity in representation, the remainder of the paper considers only the implementation of a highly scalable TG algorithm.

6.4 Highly Scalable Trip Grouping

Although the TG algorithm can be modified to meet the unique requirements of the proposed RSS, as it was demonstrated in Section 6.3.4, the algorithm in its present form does not scale with the input size and hence cannot be applied in large scale CT systems, such as the proposed RSS. This section describes a parallel implementation of the TG algorithm in the SCSQ Data Stream Management System.

Queries and procedures in SCSQ [105] (pronounced *sis–queue*) are specified in the query language SCSQL [106] (pronounced *sis-kel*). SCSQL is similar to SQL, but is extended with streams as first–class objects. SCSQ also features a main memory database. This database is used to keep the trip requests that are waiting along with statistics about the data distributions, which are used by the partitioners. The waiting requests are processed by the TG algorithm.

Details of the implementations are organized as follows: Section 6.4.1 describes how the trip grouping algorithm is implemented as a stored procedure in SCSQL. Section 6.4.2 outlines how SCSQ allows parallelization of the continuous stream query implementation of the TG algorithm. Section 6.4.3 describes four spatial partitioning methods that are used to partition the stream of trip requests into sub–streams for parallelization purposes.

6.4.1 Processing of a Request Stream

The TG algorithm is expressed as a procedure in SCSQL. The `tg` procedure takes an `input_window` of the most recently arrived trip requests, and the three algorithm parameters `K`, `min_savings`, and `wait_time` The output of `tg` is a vector of best vehicle–shares, `bcss`. `tg` executes as follows. First, on line 6, all requests in `input_window` are added to the main memory table of waiting requests `q`. Then, on line 7, based on the `wait_time` parameter and the current time `ct` (indicated by the end of the `input_window`), expiring requests, `ex`, are selected from `q`. The `for each` loop on line 10 iterates over each request `r` in `ex` as follows. On line 12, the request `r` is removed from the `q`. Then, in a compound query on lines 13–18, the best, maximum `K`–sized vehicle–share for `r` is found. The first part of the compound query, on line 16, calculates the fractional extra costs `calc_FEC(r)=<r,ri,fec>` between `r` and all other requests in `q`, and selects the tuples for the `K` requests with the lowest fractional extra costs. The remaining parts of the compound query, on lines 17–18, calculates the amortized costs `calc_AC(fec)=<r,ri,ac>` based on the top–K fractional extra costs, and selects the lowest of these costs. The best vehicle–share that corresponds to this lowest amortized cost is assigned to `s` on line 13. Finally, if the `savings` of `s` is greater than equal to `min_savings`, then the `members` of `s` are added to the best vehicle–shares, `bcss` (line 21), and are removed from `q` (line 22). Otherwise, `r` could not share its trip, and will be the only one in its vehicle–share (line 25). The implementations of the derived functions `insert_q`, `get_end`, `select_ex_q`, `remove_q`, `subvector`, `calc_FEC`, `savings`, and `members` are omitted to preserve brevity. For efficiency reasons, core functions that need to iterate over a set, such as `topk` and `calc_AC` are implemented as foreign functions in LISP. Foreign functions allow subroutines defined in C/C++, Lisp, or Java to be called from SCSQL queries. The implementation of these functions are also omitted. The following is the listing of the `tg` procedure in SCSQL:

```
(1) create function tg(vector input_window,
(2)                    integer K, real min_savings,
(3)                    integer wait_time)->vector
(4) as begin
(5)   declare vector ex, vector bcss, timeval ct;
(6)   insert_q(in(input_window));
(7)   set ct = get_end(input_window);
(8)   set ex = select_ex_q(curr_time, wait_time);
(9)   set bcss = {};
(10)  for each vector r where r = in(ex)
(11)  begin
(12)     remove_q(r);
(13)     set s = select subvector(ac,0,i)
(14)             from  vector fec, vector ac,
(15)                   integer i, integer k
(16)             where fec = topk(calc_FEC(r),2,K)
(17)               and ac = calc_AC(fec,2)
(18)               and i = min(ac,2);
(19)     if savings(s) >= min_savings
(20)     begin
(21)        set bcss = concat(bcss,members(s));
(22)        remove_q(members(s));
(23)     end;
(24)     else
(25)        set bcss = concat(bcss,r);
(26)  end;
(27)  result bcss;
(28)end;
```

6.4.2 Parallel Stream Processing in SCSQ

Apart from streams and tables, SCSQL also includes Stream Processes (SPs) as first–class objects in queries. SPs allows dynamic parallelization of continuous queries,

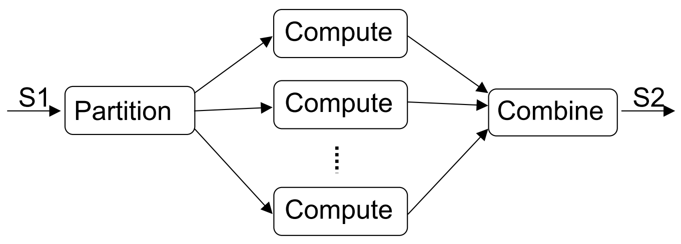


Figure 6.4: Communication Pattern of TGs Working in Parallel.

which is used in this paper to divide the incoming trip requests. The user associates subqueries with SPs. Massively parallel computations are defined in terms of sets of parallel subqueries, executing on sets of SPs.

The output of an SP is sent to one or more other SPs, which are called *subscribers* of that SP. The user can control which tuples are sent to which subscriber using a *postfilter*. The postfilter is expressed in SCSQL, and can be any function that operates on the output stream of its SP. For each output tuple from the SP, the postfilter is called once per subscriber. Hence, the postfilter can transform and filter the output of an SP to determine whether a tuple should be sent to a subscriber. Postfilters were used in the experiments to partition the input stream between the SPs that are carrying out TG.

The divide–and–conquer experiments are expressed as queries in SCSQL. All these queries have the same communication pattern between SPs, as shown in Figure 6.4. A Partition SP reads a stream of incoming trip requests (S1). That stream is partitioned into partial streams, which are sent to the Compute SPs. Each Compute SP executes the `tg` procedure on its partial stream. Also, each Compute SP evaluates the savings achieved, by comparing the total cost of all trips with the total cost of the shared trips. The results of all Compute SPs are merged together by a Combine SP. The resulting stream of cab requests (S2) is sent to the user.

6.4.3 Spatial Partitioning Methods

Section 6.3.4 showed that the TG algorithm does not scale well enough for large–scale CT systems. The key idea to overcome the scaling issue is a divide–and–conquer approach. All requests $r_i = \langle t_r, l_o, l_d, t_e \rangle$ are characterized by its origin and destination locations, $l_o \in \mathbb{R}^2$ and $l_d \in \mathbb{R}^2$. Hence, a request can be geographically characterized by a point in $l_o \times l_d$. In other words, a request is geographically characterized by a point in $\mathbb{R}^4$. The divide–and–conquer approach is to partition this space and assign each partition to one TG. Intuitively, this approach will gain in execution time since each TG has less workload, but will lose some of the vehicle–sharing opportunities since none of the partitions are able to probe all combinations that a serial implementation can do. The goal is to find a partitioner that executes efficiently and achieves maximum savings. The following partitioning strategies are implemented in SCSQL and investigated experimentally.

6.4.3.1 Baseline Queries

Two baseline queries are executed; the unpartitioned query and the round robin query. These queries form a performance baseline of the best and worst possible savings and execution speeds. All other methods should be compared to the measurements of these two queries.

The *unpartitioned* query applies a single TG on the entire request stream without any partitioning. Since all requests are going to a single TG, all possible sharing opportunities will be investigated. The unpartitioned approach will give the best savings, but it will also take the longest time to execute because all burden will be placed on a single node. The unpartitioned query is expressed in SCSQL as follows:

```
select tg(v, 4, 0.8, 600, 60)
from vector v, charstring file
where v = twinagg(streamfile(file), 60.0, 60.0)
and file in
{"L16.dat","L8.dat","L4.dat","L2.dat","L1.dat"};
```

The `streamfile(file)` function reads tuples that are stored in `file`, and streams them out. The `twinagg(inputstream, size, stride)` function is taking a stream as the first argument and emits a time window over the last `size` seconds, every `stride` seconds. Hence, if `size=stride`, `twinagg` emits tumbling (consecutive and non–overlapping) windows of the input stream. This `twinagg()` makes sure that `tg()` always will get one minute worth of requests each time. Hence, `tg()` will get called once per minute. If no requests have arrived during a certain minute, `twinagg()` will emit an empty window for that minute. `tg(input_window, K, min_savings, wait_time)` performs the trip grouping algorithm. The query is executing once per file in the collection of filenames given on the last line of the query.

The *Round–Robin* partitioner will send the first request to one working SP. The next request will be sent to another working SP, and so on. Each SP is given exactly $1/n$ of the total load, so the load balance is perfect. Since the Round–Robin partitioning scheme is perfectly load balanced, it will achieve the maximum possible execution speed. On the other hand, an SP that is operating on a Round–Robin data partition can be expected to give inferior savings since nearby requests not necessarily go to the same SP. Thus, the Round–Robin partitioner is expected to achieve the least savings. It is expressed in SCSQL as:

```
select merge(b)
from bag of sp b, sp c, integer n, charstring file
where b = spv(select streamof(tg(twinagg(stract(c),
              60.0, 60.0), 4, 0.8, 600)))
              from integer i where i=iota(1,n))
and   c = sp(winagg(streamfile(file),n,n),n,'rr')
and n in {16,8,4,2}
and file in
{"L16.dat","L8.dat","L4.dat","L2.dat","L1.dat"};
```

In this query, the output of `streamfile` is passed into `winagg(input_stream, size, stride)`, which is forming tumbling windows of size `n`, `n` being the number of subscribers to the Partition SP `c`. Each window is an ordered set of tuples, so it is represented as a vector. The round robin function `rr`, is applied once per subscriber. For subscriber *i*, `rr` picks up the *i*-th element in the vector emitted from winagg. The `SP(stream, nsubscribers, postfilter)` is assigning `stream` and `postfilter` to a new SP, which should expect `n` subscribers. Thus, a combination of a `winagg` on a stream and a vector dereference in the postfilter function results in a round robin partitioner.

`iota(m,n)` generates all integers from m to n. Hence, the query in the call to `spv(bag of stream)`, creates `n` duplicates of the query:
`streamof(tg(twinagg(stract(c),60.0, 60.0), 4, 0.8, 600))`, where `stract(sp)` is extracting the stream from stream process `sp`. Each one of these queries will be assigned to a stream process. Finally, the output of all the stream processes in b will be merged. Refer to Figure 6.4 for a graphical representation of the communication pattern: The partition is done at SP `c`, compute is performed by the SPs in `b`, and the combination is done in the `merge` at top level.

6.4.3.2 Static, Point Quad Partitioning

Static point quad partitioning (SPQ) calculates from historical data the medians of each dimension of the trip requests. Each dimension of the four–dimensional trip request data space split once along the median of each dimension. Figure 6.5(a) shows the SPQ partitions for some data points in two dimensions. By splitting each dimension once, SPQ partitions the four–dimensional trip request data space into 16 regions. One or more regions can be assigned to one SP, executing a TG for that region. This SCSQL query executes SPQ:

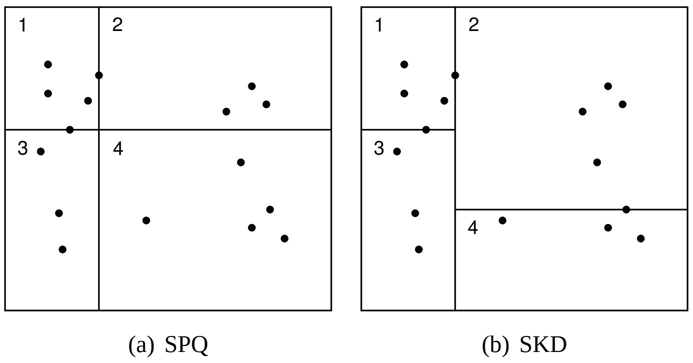


Figure 6.5: Illustrations of the Static Partitioning Methods.

```
select merge(b)
from bag of sp b, sp c, integer n, charstring file
where b = spv(select streamof(tg(twinagg(stract(c),
              60.0, 60.0), 4, 0.8, 600)))
              from integer i where i=iota(1,n))
and   c = sp(streamfile(file),n,'pq')
and n in {16,8,4,2}
and file in
{"L16.dat","L8.dat","L4.dat","L2.dat","L1.dat"};
```

The difference between this query and the Round Robin query above is only in the call to the partitioning SP `c`. Instead of applying postfilter function `rr` on a window, the SP `c` is streaming the tuples directly to the `pq` postfilter. For each tuple, `pq` decides to which subscriber it should go.

6.4.3.3 Static, KD Partitioning

Static KD partitioning (SKD) splits trip request data in a hierarchical fashion by processing dimensions one after the other as follows. For a given dimension, it first calculates the *local* median for that dimension, and then splits the local trip request data for the dimension based on the median into approximately equal sized subsets. Figure 6.5(b) shows the SKD partitions for some data points in two dimensions. The data is first split around the median of the horizontal dimension, then the data in each of the so obtained partitions is further split around the local (horizontal) median of each of the partitions. By splitting once per dimension, the KD also partitions the four–dimensional trip request data space into 16 regions. The SCSQL query that executes SKD differs from that of SPQ in that it applies another postfilter function at the partitioning SP, namely `kd` instead of `pq`. Since the difference is so small, the SCSQL query is omitted.

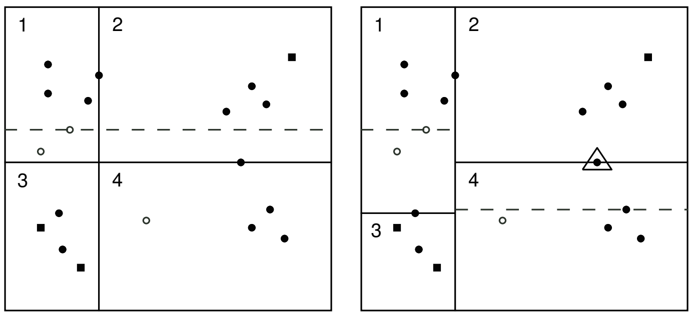
6.4.3.4 Adaptive, Point Quad Partitioning

In many applications, the distribution of the data changes over time, as it is the case for trip requests. These changes can be minor or major changes. For example, during the morning rush hours people want to move from their homes (residential district) to their work (business and industrial districts). During the evening rush hours the opposite is true. The trip requests that correspond to the morning rush hour movements are likely to fall in different partitions than the trip requests that correspond to the evening rush hour movements. Consequently, the “morning rush hour” partitions will be densely populated in the morning hours, and the “evening rush hour” partitions will be densely populated in the evening hours. Clearly, a static partitioning method does not consider these changes in data distribution and is likely to result in temporally unbalanced partitions.

The adaptive point quad partitioning (APQ) adjusts the boundaries of the partitions periodically, based on statistics obtained from a recent history buffer of the trip request stream, and distributes the newly arriving trip requests according the newly adjusted partitions. Figure 6.6(a) shows two consecutive partitionings that are constructed by the APQ partitioning for some data points in two dimensions. Hollow dots represent data points that were present when the previous partitioning was constructed, but are not present or are not relevant for the construction of the current partitioning. In contrast, solid rectangular markers represent data points that were not present when the previous partitioning was constructed, but are relevant for the construction of the current partitioning. Solid and dashed lines represent current and previous partition boundaries. The following SCSQL query executes APQ:

```
select merge(b)
from bag of sp b, sp c, integer n, charstring file
where b = spv(select streamof(tg(twinagg(stract(c),
              60.0, 60.0), 4, 0.8, 600)))
              from integer i where i=iota(1,n))
and   c = sp(pqstat(streamfile(file),
             600.0, 60.0, 10),n,'pq')
and n in {16,8,4,2}
and file in
{"L16.dat","L8.dat","L4.dat","L2.dat","L1.dat"};
```

This query differs from the SPQ query in the call to the partitioning SP `c`. The `streamfile` function is wrapped by `pqstat(inputstream, size,`



(a) APQ (b) AKD

Figure 6.6: Illustrations of the Dynamic Partitioning Methods.

`stride, samplefreq)`. This function emits the same stream as its input stream, and maintains statistics in a main memory table of SCSQ. Every `stride` $\times$ `samplefreq` seconds, `pqstat` computes medians in each dimension of $l_o \times l_d$ across the tuples seen in the last `size` seconds. These median values are then used in the `pq` postfilter. This way, the partitioning decisions are always done on recent data.

6.4.3.5 Adaptive, KD Partitioning

The adaptive KD partitioning (AKD) adjusts the boundaries of the partitions periodically, based on statistics obtained from a recent history buffer of the trip request stream. and distributes the newly arriving trip requests according the newly adjusted partitions. Figure 6.6(b) shows two consecutive partitionings that are constructed by the AKD partitioning for some data points in two dimensions. The semantics of the symbols used in the figure are the same as in the case of the APQ partitioning. However, Figure 6.6(b) depicts a situation that can happen in either one of the adaptive spatial partitioning methods. Consider the data point inside the triangle. Since it was present when the previous partition was constructed it has been assigned to compute node 2 for processing. According to the newly constructed partitions however, it should be assigned to compute node 4. To avoid communication between compute nodes, the following design choice is made: once a data point is assigned to a partition (compute node), it is never reassigned to another partition, even if the newly adjusted partitions would suggest this.

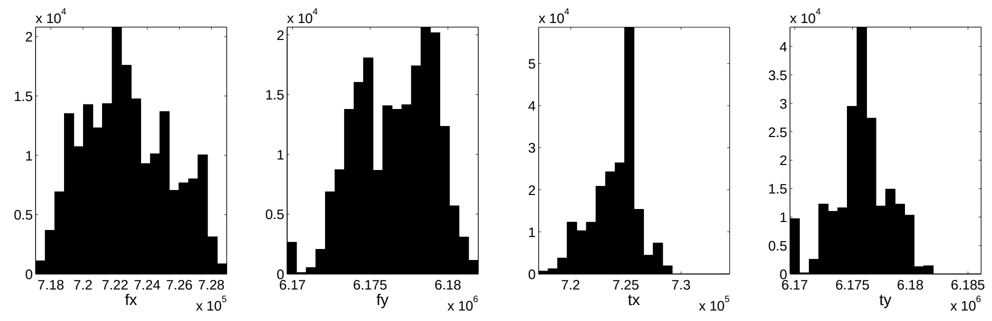
The SCSQL query that executes SKD differs from SPQ in that it applies another statistics wrapper function and another postfilter function at the partitioning SP, namely `kdstat` instead of `pqstat` and `kd` instead of `pq`. `kdstat` works analogously to `pqstat` with the difference that it maintains dynamical versions of local dimension splits of the kind that SKD has. Since the difference between this query and the APQ query is so small, the SCSQL AKD query is omitted here.

6.5 Density–Based Spatial Stream Partitioning

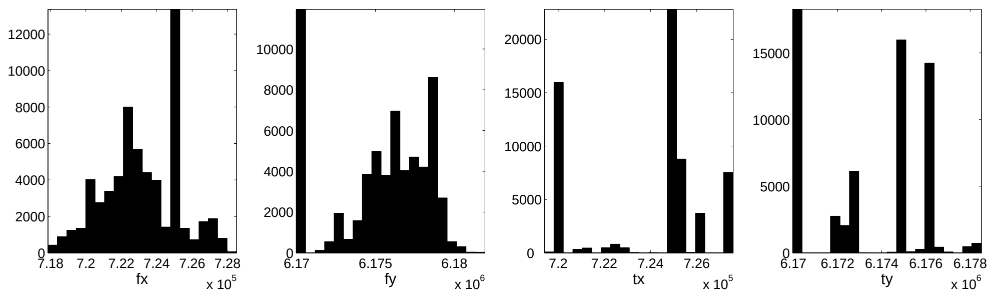
The main objective of the four proposed partitioning methods is to load balance the parallel versions of the computationally intensive TG algorithm. The planes for subdividing the space of requests is determined by the medians of request data. These splitting planes potentially eliminate the discovery of good shares, when members of the good shares are distributed to different partitions. This naturally leads to some degradation in the overall grouping. The degradation is larger when the planes are cutting through denser regions of the request space with many sharing opportunities, than when the planes are cutting through sparser regions of the request space. Since neither of the proposed partitioning methods consider other characteristics of the distribution of the requests, the degradation of grouping quality due to boundary effects is expected to be approximately the same for all four partitioning methods. However, as Section 6.6 demonstrates, this degradation is rather small.

No matter how small the degradation is, simple spatial partitioning methods that take into account the density of the data could reduce the degradation. The objective of such a density–based partitioning is to determine the positions of the splitting planes so that they pass through regions where data is sparse. To achieve this, a simple but effective clustering method [29] can be used to find local minima in the multimodal data distributions along each dimension, and place splitting planes at those locations. Figure 6.7 shows the distributions for each dimension of the request data during morning peak hours and off–peak hours. In the figure letters “f” and “t” stand for “from” and “to”, respectively. Hence, fx and fy are request origin dimensions, while tx and ty are request destination dimensions. During the morning peak hours, there does not seem to be any regions where the request data is very sparse. However, during off–peak hours, when people who are not working are most likely to be in one of the larger shopping malls, the distributions of the destination dimensions (tx, ty) are clearly multimodal. In this later situation, ensuring that splitting planes are chosen correctly at local minima would minimize the boundary effects. However, since most of the requests are during peak hours, the overall average grouping achieved by the parallel TG algorithm would not be substantially improved.

Since the local minima are likely not to be at the median values of the dimensions, there exists a trade–off between equal–sized partitions and partitions with minimal boundary effects. A dual–objective partitioning that takes this trade–off into consideration could weigh the expected degradation against the imbalance between the created partitions. Although the implementation of the density–based and the dual–



(a) Morning Peak Hours.



(b) Off–Peak Hours.

Figure 6.7: Request Data Distributions Along Each Dimension.

objective spatial stream partitioning methods is straight–forward, it is left for future research.

The proposed spatial stream partitioning methods are devised to scale the TG algorithm to very large flows of requests. However, they can be considered as a general approach to make computationally intensive spatial analysis tasks scalable through parallelization. For example, the density–based and the dual–objective spatial stream partitioning methods can be applied to speed up spatial clustering of streams, spatio–temporal rule mining [30], or the processing of high–resolution image streams.

6.6 Experiments

The parallel implementations of the TG algorithm were tested on a cluster of Intel® Pentium® 4 CPU 2.80GHz PCs. Each SP was executing on a separate PC to allow for maximum parallelism. TCP/IP over Fast Ethernet was used to carry streams between the nodes.

Trip request data was simulated using ST–ACTS, a spatio–temporal activity simulator [31]. Based on a number of real world data sources, ST–ACTS simulates realistic trips of approximately 600,000 individuals in the city of Copenhagen, Den-

load	execution time (sec)	savings
0.06125	28.8	0.325
0.125	120.1	0.388
0.25	702.9	0.445
0.5	16343.5	0.491
1	69771.6	0.530

Table 6.1: Performance of the Serial TG Algorithm.

mark. For the course of a workday, out of the approximately 1.55 million generated trips, approximately 251,000 trips of at least 3–kilometer length were selected and considered as trip requests. To test the scalability of each of the parallel implementations using the four spatial stream partitioning methods, decreasing sized subsets of the total load of 251,000 trip requests were constructed by only considering every second, fourth, eighth and sixteenth trip request in the input stream. These subsets are referred to as 1/2, 1/4, 1/8, 1/16 load, respectively.

To evaluate the effectiveness of the four spatial stream partitioning methods, for the purposes of parallelization of the TG algorithm, two measures were used: (overall) execution time and average savings achieved by the grouping (also referred to as the quality of the grouping or quality for short). The reported savings for each vehicle–share are based on amortized costs, which has been shown to overestimate the true cost of a vehicle–share that considers the optimal pick–up and drop–off sequence of requests. Hence, the reported savings underestimate the true savings. Nonetheless, the reported savings can be used as an unbiased measure for the quality of the grouping.

For each of the partitioning methods an extensive set of experiments were performed for fixed algorithm parameters ($K = 4$, $\texttt{min\_saving} = 0.2$, and $\Delta t = 10$ minutes) under varying loads using degrees of parallelization. The adaptive partitioning methods updated the partitions every 10 minutes based on the trip request that arrived in the last 10 minutes.

6.6.1 Baseline Performance

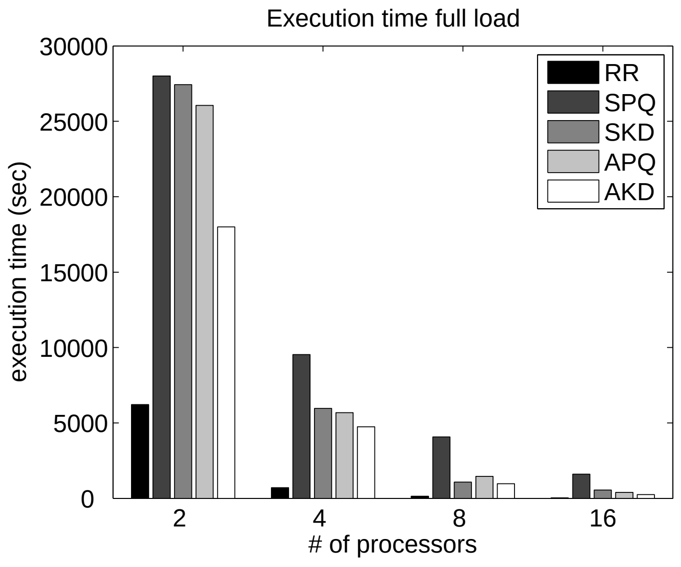
To establish a point of reference for the performance measures the baseline queries specified in Section 6.4.3.1 were executed. Table 6.1 shows the results for the unpartitioned query. Savings obtained by the unpartitioned query (serial execution) are *considered* to be optimal, while running times are *considered* to be worst case performance. Note that these measures are “optimal” and “worst case” with respect to the TG algorithm. Moreover, as it is demonstrated in Section 6.3.3, due to the computational complexity of the *vehicle–sharing problem,* the calculation of a truly optimal grouping, even in the case of a few requests, is infeasible. Due to the large difference in scale between serial and parallel execution times, serial execution times are not shown in subsequent figures. Savings achieved by the unpartitioned query (serial execution) are also not shown in subsequent figures, but are used to report relative performance of the parallel executions in terms of savings / quality.

In comparison, the round robin query was executed to obtain optimal execution times due to perfect load balancing and worst case savings due to the query independent distribution of data between query processors. The results of these experiments are shown in Figures 6.8 and 6.9 as RR, however it is emphasized that RR is *not* one of the proposed spatial stream partitioning methods, but is *only* used as a reference.

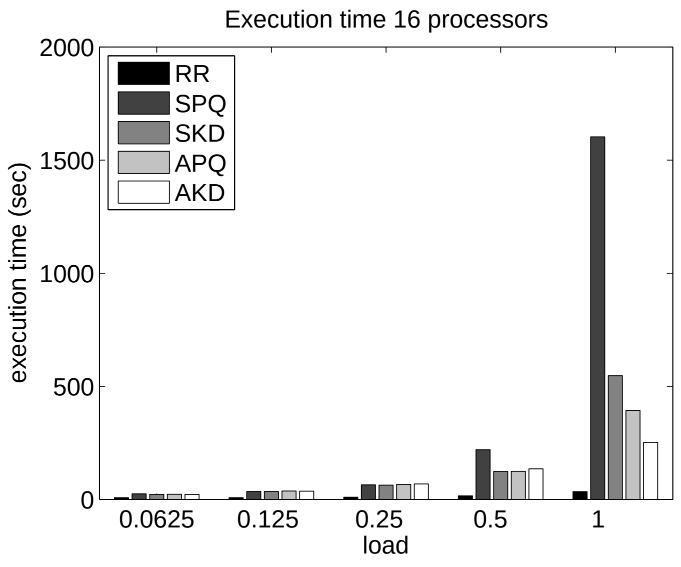
6.6.2 Absolute Performance of the Parallel TG Algorithms

Figures 6.8 and 6.9 show the absolute performance of the parallel TG algorithm for varying load and degrees of parallelization using different spatial stream partitioning methods. From Figure 6.8(a) it can be seen that the execution times of all of the methods decrease as the parallelism is increased. Figure 6.8(a) also reveals that the adaptive versions of the spatial partitioning methods adjust well to the changing spatial distribution of the requests, resulting in more balanced partitions and ultimately faster execution times when compared to their static version. The improvement in execution time due to adaptive partitioning is most evident for the SPQ partitioning. Figure 6.8(b) shows that while the execution time of the TG algorithms can be scaled, the underlying algorithmic complexity of the TG algorithm executed on the compute nodes does not change. The effect of the underlying algorithmic complexity is more observable for spatial partitioning methods that construct less balanced partitionings, in particular SPQ.

Figure 6.9(a) shows that in general the quality of the grouping decreases as the degree of parallelization is increased. However in the case of non–spatial partitioning (RR) this degradation is significant, while in the case of either one of the four spatial partitioning methods it is negligible. Figure 6.9(b) shows that as the load is increased the grouping quality increases. This is due to the simple fact that the spatio–temporal density of the trip requests increases. As a consequence, the likelihood that a request becomes part of a "good" vehicle share increases. The almost negligible differences between the qualities achieved by the four partitioning methods, as explained in Section 6.5, is due to the fact that since neither of the partitioning methods consider the data densities, but only the medians of the dimensions, the total degradation due to boundary effects is approximately the same for the four partitioning methods.

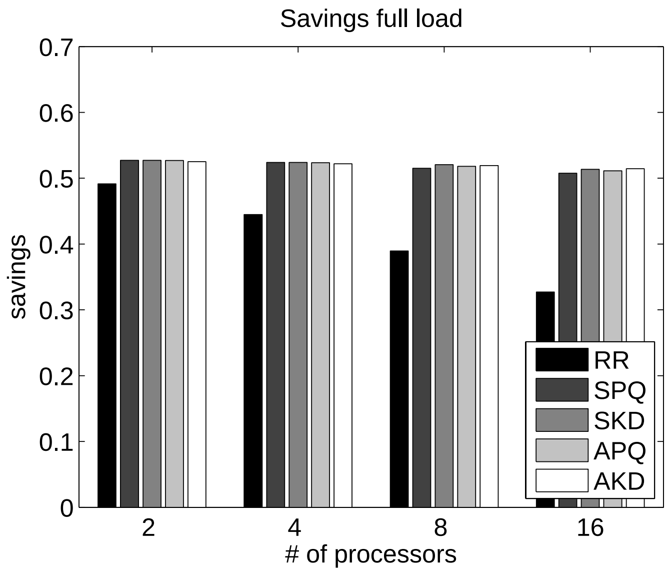


(a) Performance for Full Load.

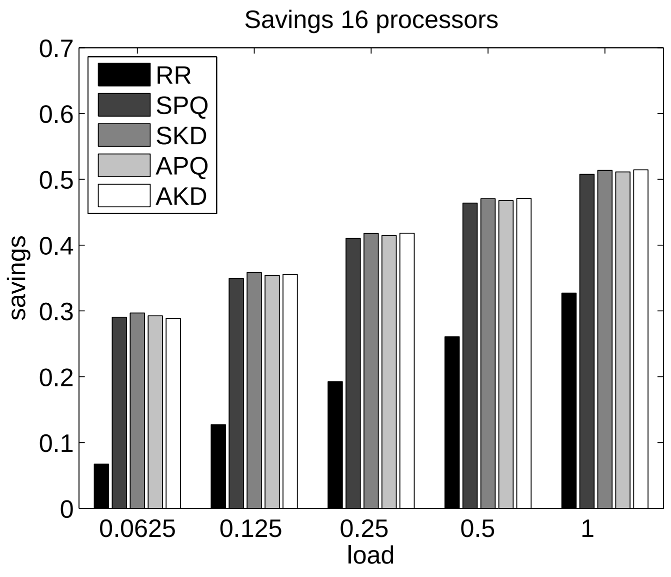


(b) Performance for 16 Processors.

Figure 6.8: Execution Times for the Parallel TG Algorithm for Different Partitioning Methods for Varying Parallelization and Load.



(a) Quality for Full Load.



(b) Quality for 16 Processors.

Figure 6.9: Savings for the Parallel TG Algorithm for Different Partitioning Methods for Varying Parallelization and Load.

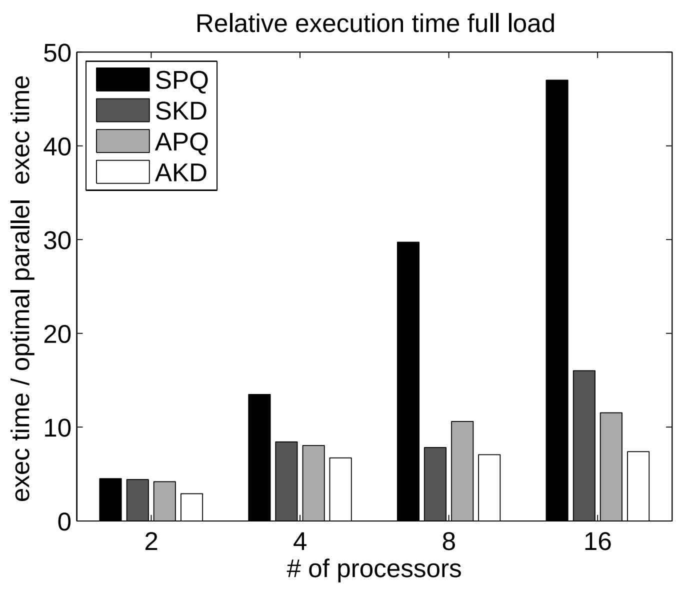


Figure 6.10: Relative Performance for the Parallel TG Algorithm (Compared to RR Partitioning) for Different Partitioning Methods for Varying Parallelization.

6.6.3 Relative Performance of the Parallel TG Algorithms

Figure 6.10 shows relative execution times of the parallel TG algorithms when compared to the optimal execution time that is achieved by RR partitioning due to perfect load balancing. With the exception of the SPQ partitioning all other partitioning methods result in parallel execution times that are within the same order of magnitude as the optimal. There are potentially two sources for this slowdown: the cost of partitioning and the extended execution times due to improper load balancing. Since adaptive partitioning methods have to maintain a limited history of the stream and periodically recompute partition boundaries based on this history, they do additional work compared to their static counterparts. Yet, in Figure 6.10 execution times resulting from adaptive partitioning are significantly lower than the execution times achieved by static partitioning. Hence, it is clear that the additional time needed to perform the spatially partitioned parallel queries can mainly be attributed to unbalanced partitions.

Finally, comparing the savings in Figure 6.9(b) to the savings in Table 6.1 reveals that the grouping quality achieved by either one of the partitioning methods is within the 95% of the optimal quality for the full load. Even if the load is decreased to 1/16 of the total load, all the spatial partitioning methods still achieve approximately 90% of the maximum possible savings.

The experiments can be summarized as follows. First, RR partitioning has perfect load balance and is a very simple partitioning method, hence it has the fastest execution time. However, RR partitions the space badly and achieves a bad grouping quality. Second, using a spatial partitioning method improves grouping quality. All spatial partitioning methods achieve at least 95% of the maximum possible savings in case of the full load. Third, the adaptive partitioning methods always execute faster than their static equivalents. That is because the adaptive methods constantly adapt the partitioning according to the last tuples observed, which will lead to better load balance. At the same time, the savings are approximately the same for both the static and dynamic partitionings. Adaptive partitioning is also preferred from an operation point of view, since it does not need any prior knowledge about the data distribution. Finally, since all partitioning methods (except RR) achieve about the same savings, the preferred method is the one with the fastest execution time of SPQ, SKD, APQ, and AKD. Thus, AKD is the best partitioning method.

6.7 Conclusions and Future Work

The paper proposed highly scalable algorithms for trip grouping to facilitate large–scale collective transportation systems. The algorithms are implemented using a parallel data stream management system, SCSQ. First, the basic trip grouping algorithm is expressed as a continuous stream query in a data stream management system to allow for a very large flow of requests. Second, following the divide–and–conquer paradigm, four spatial stream partitioning methods are developed and implemented to divide the input request stream into sub–streams. Third, using the infrastructure of SCSQ and the partitioning methods, parallel implementations of the grouping algorithm are executed in a parallel computing environment. Extensive experimental results show that the parallel implementation using simple, adaptive partitioning methods can achieve speed–ups of several orders of magnitude without significantly affecting the quality of the grouping. As discussed in Section 6.5, spatial partitioning is not only appropriate for the given application, but it is applicable to parallelize computationally expensive spatial analysis tasks. As it was demonstrated, SCSQ can easily accommodate the parallel implementations of such tasks.

Future work will be along four paths. First, for the adaptive partitioning methods, the effects of keeping a longer history versus sampling more frequently will be investigated. Second, the density–based and dual–objective spatial stream partitioning methods will be implemented and their effectiveness evaluated. Third, the proposed partitioning methods, independent of the rate of flow, always construct a fixed number of partitions. While not substantially, but as the number of partitions increases the grouping quality decreases. Hence, an adaptive partitioning approach in which the number of partitions is increased / decreased depending on the rate of flow will be devised and tested. Finally, to preserve clarity the paper presented the generic TG algorithm in its simplest form. In particular, in the presented version all vehicles are assumed to have the same passenger capacity and all requests have a common minimum savings parameter. Furthermore, in–route grouping, i.e., assigning requests to already active but not fully–occupied vehicle–shares, is not handled by the simple version of the TG algorithm. Future work will consider the implementation of a more complex version of the TG algorithm that addresses the above issues.

Chapter 7

Estimating the Capacity of the Location–Based Advertising Channel

Delivering “*relevant*” advertisements to consumers carrying mobile devices is regarded by many as one of the most promising mobile business opportunities. The relevance of a mobile ad depends on at least two factors: (1) the *proximity* of the mobile consumer to the product or service being advertised, and (2) the match between the product or service and the *interest* of the mobile consumer. The interest of the mobile consumer can be either *explicit* (expressed by the mobile consumer) or *implicit* (inferred from user characteristics). This paper tries to empirically estimate the capacity of the mobile advertising channel, i.e., the number of relevant ads that can be delivered to mobile consumers. The estimations are based on a simulated mobile consumer population and simulated mobile ads. Both of the simulated data sets are realistic and derived based on real world data sources about population geo–demographics, businesses offering products or services, and related consumer surveys. The estimations take into consideration both the proximity and interest requirements of mobile ads, i.e., ads are only delivered to mobile consumers that are close–by and are interested, where interest is either explicit or implicit. Results show that the capacity of the Location–Based Advertising channel is rather large, which is evidence for a strong business case, but it also indicates the need for user–control of the received mobile ads.

7.1 Introduction

Mobile or Location–Based Advertising (MA or LBA), i.e., sending electronic advertisements to consumers carrying mobile devices, is considered by many as one of the most promising business opportunities amongst Location–Based Services (LBS) [62]. A recent mobile marketing survey suggests that about 7% of the mobile consumers would be willing to receive promotional text messages “if they were relevant” [27]. According to other surveys, an even larger percentage of the mobile consumers are interested if they are rewarded in some way [66]. In this paper, mobile ads are regarded as a means of presenting relevant information to a recipient, be it a commercial offer on an item on sale, traffic information, or a piece of public information. To many people, the world seems to be more and more difficult to guide oneself through, thus the art of targeting information and services will prove to be of immense value. Only, efficient business cases have so far been very few, in spite of the market’s expectations.

A broad range of aspects, or variables, determine the relevance and context of a mobile ad: distance to the mobile user, explicit or implicit interest of the mobile user, uniqueness (do not send ad twice within some interval), time and place of delivery, etc. To this extent, this paper describes an LBA framework and an LBA database that can be used for the management of mobile advertisements.

In lack of comprehensive, real data on the movements and behavior of the population, estimation or simulation is extremely useful, bringing the models to life with real and well–documented consumption patterns. Using a simulated but realistic mobile consumer population and a set of mobile ads, the LBA database is used to estimate the capacity of the mobile advertising channel, i.e., the number of relevant ads that can be delivered to mobile consumers. Apart from this use, the LBA database and the estimates derived from it can also be used in mobile catchment area analysis to estimate business exposure. Results show that the capacity of the LBA channel is rather large (approx. 100 mobile ads per user within a single day), giving strong support for a business case. The same results can also be viewed as alarming, and indicate the need to incorporate user–control of the received mobile ads in the LBA framework, as suggested by the Mobile Marketing Association [73].

The remainder of this paper is organized as follows. Section 7.2 reviews related work. Section 7.3 defines the estimation problem both in case of explicit and implicit interest. Section 7.4 describes the simulated data sets and their derivations from real–world data sources. Section 7.5 describes the method and technical foundations for delivering mobile ads while taking into account both the advertisers’ and mobile users’ interests. Section 7.6 proposes a revenue model for LBA. Section 7.7 describes the experiments and discusses the estimates resulting from them. Finally, Section 7.8 concludes and points to future research directions.

7.2 Related Work

The estimations in this work are based on simulated movements of mobile users. Movements of users are influenced by physical, social and geo–demographical aspects of mobility. To aid the development in mobile data management, a number of moving object simulators have been proposed in the literature that model primarily the physical aspects of mobility to various extents. Since most objects use a network to get from one location to the other, a state–of–the–art framework for network–based moving object simulation is presented in [8]. The behavior of a moving object in this framework is influenced by (1) the attributes of the object having a particular object class, (2) the combined effects of the locations of other objects and the network capacity, and (3) the location of external objects that are independent of the network. For a review of other moving object simulators the reader is referred to [31].

Moving object simulators generally neglect the social and geo–demographical aspects of mobility. These social and geo–demographical aspects of mobility introduce patterns in the movement of users and give rise to a unique spatio–temporal (ST) distribution of users. ST–ACTS is a Spatio–Temporal ACTivity Simulator that using real–world data sources models some of these neglected aspects of mobility [31]. To make the estimations in this work as realistic as possible, movements of mobile users are obtained from ST–ACTS, which is further described in Section 7.4.4.

Database indices allow the effective management and retrieval of data in large databases. Spatial and geographical databases manage information about spatial objects, i.e., objects that have physical properties such as location and extent. An R–tree is a widely used index structure that allows the effective management and retrieval of spatial objects [45]. An R–tree splits space with hierarchically nested, and possibly overlapping Minimum Bounding Rectangles (MBRs). Search algorithms that test spatial relationships (for example; intersection, containment, nearest) between spatial objects can effectively use the MBRs to decide whether or not objects in two MBRs satisfy a specific spatial search criterion.

The location of a moving object changes over time. Thus, the path of a moving object is commonly described as a sequence of coordinate and timestamp pairs and is referred to as the trajectory of the moving object. Moving objects databases are databases that represent and manage changes related to the movement of objects. Spatio–temporal indices such as the Spatio–Temporal R–tree (STR–tree) and Trajectory–Bundle tree (TB–tree) allow the effective management and retrieval of information about moving objects [55]. An STR–tree organizes line segments of a trajectory according to both their spatial properties and the trajectories they belong to, while a TB–tree only preserves trajectories. While these spatio–temporal indices are designed to effectively manage trajectories, they are not available in commercially available Relational Database Management Systems (RDBMSs). Hence, the herein presented method uses the widely available R–trees.

Time geography [46] is a conceptual basis / paradigm for human space–time behavior which considers (1) the indivisibility or corporeality of the human condition; (2) that humans typically operate over finite intervals of space and time; (3) the natural laws and social conventions that partially constrain space–time behavior; and (4) that humans are purposive. The movements of mobile users used in the estimations are derived from ST–ACTS [31], which models some aspects of this paradigm.

Research has shown that LBSes have not yet been as widely used as expected [60]. In opposition to earlier forecasts and market expectations, technology has not been ready until now. Furthermore, and no less important, is that the user and the user needs have not been fully understood. Recent research shows that in order to succeed with location–based ads, content is imperative [61], as well as taking the consumers' permission, acceptance and responsiveness into account is crucial [5, 50]. Being aware of this, the concept proposed in the present paper takes off from a point where any user will benefit directly from the use of the system, thus leading to an increased opt–in readiness.

Ensuring full user responsiveness is by no means trivial; research indicates that behavioral intention to use mobile commerce can be greater for mobile commerce–nonusers than for users. Those most used to mobile commerce are not as prone to take action on it. However, although this documents that the triggers of mobile commerce acceptance is not yet fully understood [65], mobile users are found to be eager to make use of their phones in new ways, and methodologies are proposed to model user willingness [24, 66, 78]. In the present paper, by targeting content to the right recipients, the setup is realistic and should be attractive for the typical future user.

Market research document a significant increase in sales to customers who were exposed to mobile advertising compared to those who were not exposed [72]. In other words, it works, if only the above mentioned issues are treated seriously. It is the intention with the present paper to contribute in furthering the spread of LBSes.

7.3 Problem Statement

Let $A = \{a_1, \dots a_n\}$ be the set of ads. Each ad a has a location $adloc(a)$ and is for a certain product $prod(a)$. Let $U = \{u_1, \dots u_n\}$ be the set of (moving) users. Each user u has a location $uloc(u, t)$ depending on the time t, an explicit interest profile $expint(u)$ containing a set of products, and an implicit interest profile $impint(u)$, containing a set of demographic variable values. Also assume a scoring function $score(u, a)$ that given a user u and an ad a returns a value between 0 (no match) and 1 (perfect match) that predicts how interested user u is in product $prod(a)$ based on the values in $impint(u)$.

referred entity	conzoom® variable	categories
person	person count	1
	age	9
	education type	9
	employment status type	12
	employment branch type	12
housing unit	unit count	1
	house type	6
	house ownership type	4
	house area	5
household	household count	1
	family type	5
	fortune	6
	personal income	5

Table 7.1: Variables in conzoom®.

Given a maximum distance *maxdist* between user locations and ad locations, and a timespan $T = [t_{start} : t_{end}]$, the ***explicit location–based ad delivery estimation problem*** is to estimate how many times a user u has a location $uloc(u,t)$ within *maxdist* from $adloc(a)$ for an ad a in A and a time t in T where $prod(a) \in expint(u)$.

Given a *maxdist* between user locations and ad locations, a *minscore*, and a timespan $T = [t_{start} : t_{end}]$, the ***implicit location–based ad delivery estimation problem*** is to estimate how many times a user u has a location $uloc(u,t)$ within *maxdist* from $adloc(a)$ for an ad a in A and a time t in T where $score(u,a) \geq minscore$.

7.4 Data

The estimations stated in Section 7.3 are based on a number of real–world data sources. The use of real–world data sources is important to derive realistic estimates. While the data sources refer to the Danish market and population, similar data sources are available for other major markets [18, 19]. The following subsections describe in detail the data sources used to derive the estimates.

7.4.1 conzoom® Demographic Data

conzoom® is a commercial database product that contains fine–grained, geo–demographic information about Denmark's population [28]. The variables that describe the statistical characteristics of the population can be divided into three groups: person, housing unit, and household variables. These variables and the number of categories for each are shown in Table 7.1.

In Table 7.1, variables that have “type” in their names are categorical variables; variables that have “count” in their name are counts of the corresponding entities within a 100–meter grid cell; and finally, the rest of the variables are continuous variables that have been categorized into categories that are meaningful for market segmentation. Since, for example in the countryside, the number of persons, households or units could be very low in a 100–meter grid cell, grid cells are grouped together into meaningful, large enough clusters to comply with social and ethical norms and preserve the privacy of individuals. The basis for clustering is twofold: geography and the publicly available one–to–one housing information. The intuition behind the basis is also twofold. First, people living in a given geographical region (be that a state, a county, or a postal district) are similar in some sense; for example, they might be more likely to have a certain political orientation than people living in another geographical region. Second, people living in similar houses are likely to be similar in other demographic variables; for example an established family with a stable source of income is more likely to be able to buy a larger, more expensive house than a person who just started his/her career.

As mentioned earlier, to preserve the privacy of individuals, the clusters are constrained to contain at least some fixed number of households. Statistics for the variables, depending on the sensitivity of the information contained in them, are obtained from Statistics Denmark [85] for clusters constructed at an appropriate level of cluster size constraint, for example 20, 50, 100, and 150 households per cluster. In case of a continuous variable, for example age, counts of the corresponding entities (in this case persons in the cluster) are obtained for the categories of the given variable. Due to this constrained geo–clustering method, the conzoom® clusters obtained comply with the social and ethical norms and preserve the privacy of the individual, yet the statistics obtained are accurate enough for effective market segmentation. This segmentation results in a grouping of the Danish population into 29 conzoom® types, one of which is defined for each 100–meter grid cell. Cosmopolitan (type 3) is one example of the 29 conzoom® types. Comparing the demographics of type 3 to the demographics of the rest of Denmark’s population gives the demographic profile of the type. This profile is partially shown in Figure 7.1. It roughly describes individuals that are more likely: to be middle aged (30–59 years old), to live in larger cities in larger, multi–family houses that are either owned by them or are private rentals, to be mostly couples with children, to have a medium to long higher education, to hold higher level or top management positions in the financial or public sector, and to have a better household economy (in terms of wealth and income) than the average Dane.

7.4.2 GallupPC® Consumer Survey Data

GallupPC® is a commercial database product and as the name suggests, it contains detailed survey responses of consumers about their demographics; interests such as

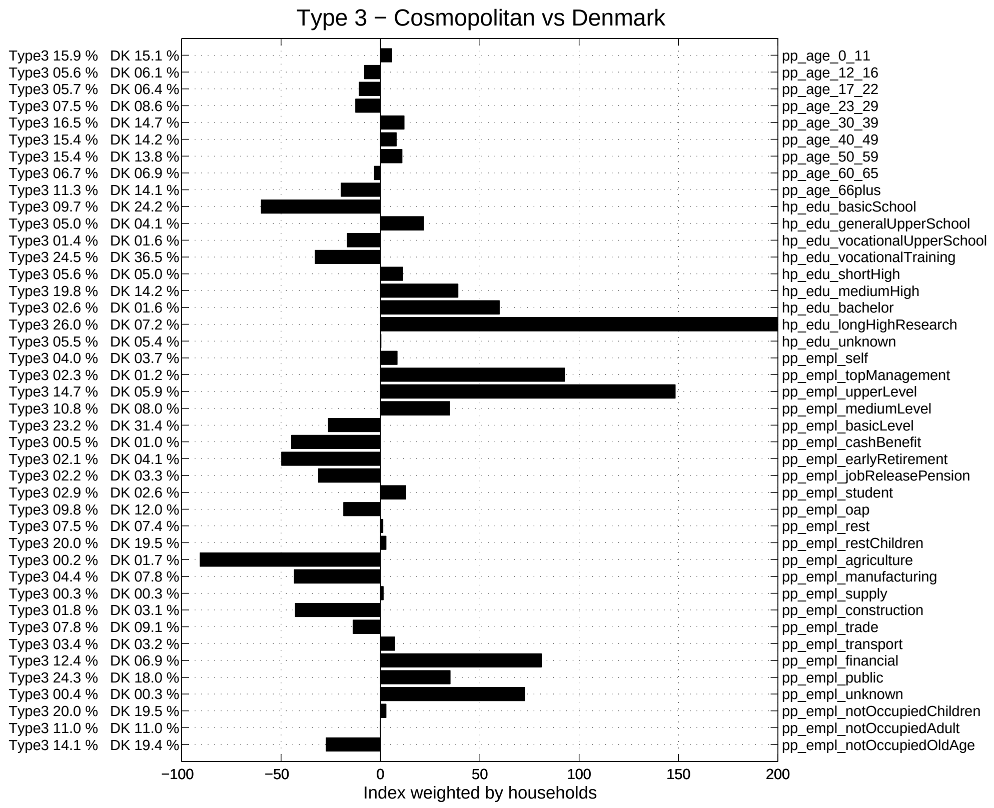


Figure 7.1: Partial Profile of conzoom® Type 3.

culture, hobbies, and sports household consumptions, purchasing habits; transportation habits; views on various subjects; attitudes and exposure to various advertisement media [26]. The questions in the surveys are yes/no questions. To measure the magnitude of the consumer's interest in a specific area, the original yes/no question is re–phrased as categorical questions. For example the original yes/no question "Are you interested in fashion?" is re–phrased to 5 yes/no questions using the following answer possibilities: very, rather, somewhat, not very, or not interested.

7.4.3 bizmark™ Products and Services

bizmark™ is a commercial database product that contains detailed information about Danish businesses both in the public and the private sector [28]. Some of the one–to–one information that is available about businesses is their location, the number of employees working in them, the physical size of the business facility, and the international branch codes the businesses fall under. Using the hierarchy of international branch codes 40 product and service categories were identified for which related consumer surveys were also available. The product and service categories are as follows: classical concert; pop/rock concert; discothèque; art exhibition; museum; cinema; theater; pharmacy; bicycle / moped; car, stereo/HI–FI; CDs/DVDs; computer/internet; new technologies/telecommunication; do–it–yourself; fashion; cosmetics/skincare; glasses/contacts; hairdresser; jeweler/watches; interior design; travel; pets, fast–food; and 14 brand specific supermarkets. Based on the international branch codes a one–to–many relationship has been established between a subset of the businesses in bizmark™ and the 40 product / service categories.

7.4.4 Simulating Mobile Users with ST–ACTS

ST–ACTS is a probabilistic, parameterizable, realistic Spatio–Temporal ACTivity Simulator [31]. ST–ACTS is realistic in the sense that it is based on a number of real–world data sources (among others, the data sources described above) and a set of principals that try to model the *social* and some of the *physical* aspects of mobility. The modelled principles that govern the social aspects of mobility are: (1) People move from a given location to another location with an *objective* of performing some activity at the latter location; (2) Not all are equally likely to perform a given activity. The likelihood of performing an activity depends on the *interest* of a given person, which in turn depends on a number of demographic variables; (3) The activities performed by a given person are highly *context dependent*. Some important parts of context are: the current person location, the set of locations where a given activity can be performed, the current time, and the recent history of activities of the person; (4) The locations of facilities where a given activity can be performed, are not randomly distributed, but are *influenced* by the *locations* of other facilities and the *locations* of the users of those facilities.

The output of ST–ACTS is a population of simulated persons, each described by a set of demographic variables and associated with a trajectory. The trajectories are sequences of time–stamped activities performed at particular physical locations, i.e., coordinates. In addition to the four principles above, the simulated activities also obey the following constraints. First, the *temporal activity constraint*, which states that certain activities are more likely to be performed during some periods than others. Second, the *activity duration constraint*, which states that not all activities take the same amount of time. Third, the *maximum distance constraint*, which states that for most activities there is a maximum distance a person is willing to travel. Finally, the trajectories assume linear movement between two consecutive activities, i.e. locations, but obey some *physical mobility constraints*, namely, that it takes time to move from one location to another. The time it takes to move from one location to another is calculated based on the distance between the two locations and

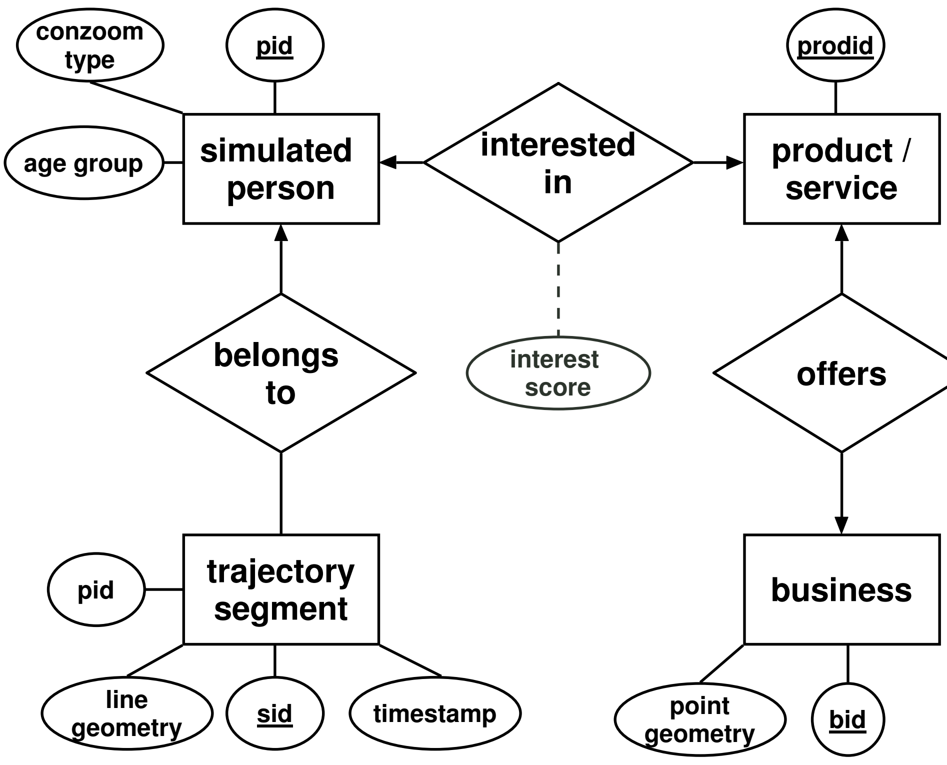


Figure 7.2: Simplified, Extended ER Diagram of the LBA Database.

a realistic speed model that assigns lower speeds to shorter, and higher speeds (with larger variance) to longer distances.

7.5 Method

The method presented here uses the Oracle RDBMS, and one of its extensions, Oracle Spatial, which provides advanced spatial features to support high–end GIS and LBS solutions [75].

7.5.1 LBA Relational Database

The objects or entities in the database are: simulated persons (or equivalently referred to as mobile users), trajectory segments, businesses, products and services. A simplified extended Entity–Relation (ER) diagram of the database is shown in Figure 7.2. In the extended ER diagram, square boxes represent entities, oval represent properties of entities, and diamonds represent relationships between entities. Underlined properties represent primary constraints. The arrows between entities encode connectivity of relationships, i.e., and arrow represents “one” and no arrow represents “many”. For example, the “belong to” relationship is a many–to–one relationship between trajectory segments and simulated persons, i.e. one trajectory segment belongs to exactly one simulated person, but many trajectory segments can belong to one simulated person. Mobile ads are indirectly modelled by in the relational database the many–to–many “offers” relationship between businesses and products or services. Through this relationship, a mobile ad can be thought of as an entity having a unique identifier composed of a unique combination of `bid` and `prodid`, and having a *location* specified by the *point geometry* of the business offering the advertisement.

As it was introduced earlier, in the implicit case, the interest of a mobile user u and a mobile ad a about a product or service $prod(a)$ is not a Boolean function or binary relation. Rather, it is a continuous function that given the demographic characteristic $impint(u)$ of u, assigns a real valued interest score $score(u, a)$, usually from 0 (not interested) to 1 (very interested), for $prod(a)$. In direct marketing this function is termed a scoring function, which encodes a particular scoring model. This real valued scoring function is untraditionally represented as a property of the “interested in” relationship in the ER diagram.

7.5.2 Proximity Requirements on Mobile Ads

A mobile ad a is likely to be considered relevant to a mobile user u only if at the time of delivery t, u is (or at some foreseeable future time point will be) within a maximum distance, *maxdist*, to the origin of the mobile ad $adloc(a)$, i.e. the location of the business. Using the spatial features of Oracle Spatial, this proximity criterion between mobile ads and mobile users is tested as follows. The geometries of the businesses, equivalently mobile ads, are buffered to a maximum distance, and tested for any spatial interaction with the geometries of the trajectory segments, by performing a spatial join operation in the database. To make the join operation as fast as possible, geometries are indexed using R–trees.

7.5.3 Interests Based on Demography

The relevance of a mobile ad for a particular product or service is naturally influenced by the interest of the user for the given product or service. As described above, a subset of the GallupPC® consumer survey questions are related to products or services that can be directly linked to businesses in bizmark™, and measure the interests of the consumer in the products or services.

Using the geo–demographic parts of the surveys, each survey subject is assigned to one of the 29 conzoom® types. To derive a single indicator, an interest score, for how interested a given conzoom® type is in a given product or service, the answers

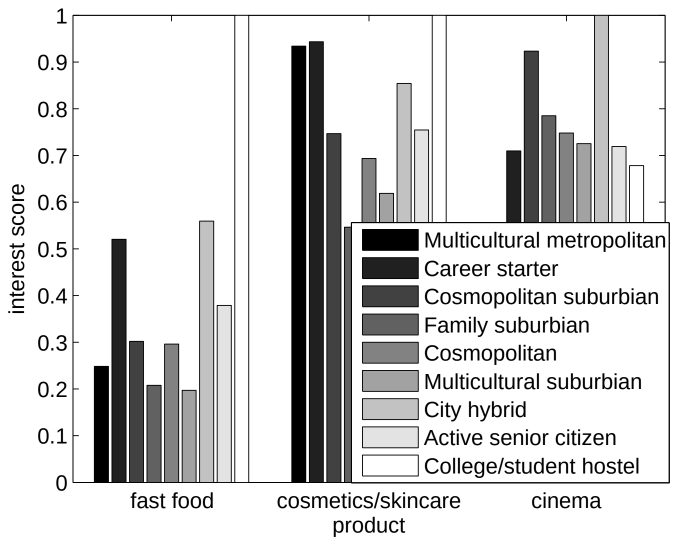


Figure 7.3: Some Interest Scores for Products or Services for Different conzoom® Types.

to the questions processed as follows. First, the five possible answer choices are associated with the following interest scores: very interested (1), rather interested (0.75), somewhat interested (0.5), not very interested (0.25), and not interested (0). Second, for a given conzoom® type and product or service the interest scores assigned to individual answers are averaged. Finally, the mean interest scores for a given product or service are scaled to the [0, 1]–interval amongst the 29 conzoom® types. Figure 7.3 shows a sample of these interest scores for a subset of the conzoom® types. In Figure 7.3 it can be seen, that college students are most interested in fast food and cosmetics / skincare products, and among the conzoom® types listed, suburban families are least likely to be interested in the same.

7.5.4 LBA – Implicit Interest Case

The interest score of a mobile user u in a particular product or service, which is advertised by mobile ad a, is implicitly encoded in the demographic characteristic, $impint(u, a)$, or historical behavior (reaction to previously received mobile ads) of u. The latter encoding is commonly referred to as relevance feedback in the scoring task in direct marketing, and while not considered in the current mobile advertising database, it can be naturally incorporated. In direct marketing, the model for this interest score is usually derived for one or many product(s) or service(s) of a particular company through the process of data mining or machine learning. This model can be represented as a *company–specific interest score function* in the mobile advertising database that for given user–demographics and historical user–behavior, see Section 7.5.8, assigns an interest score to the user. In the estimations however, these interest score functions are not company– but rather only product– and service–specific. Furthermore, due to their simplicity, they are implemented as table with the following schema: `interest_score` = ⟨`conzoom_type, prodid, score`⟩.

7.5.5 LBA – Explicit Interest Case

Mobile users can also explicitly state their interest in certain products and services. In this case the "interested in" relationship is a binary relationship in the mobile advertising database. To provide a realistic estimates, the explicit interests of users are probabilistically simulated by randomly drawing a fixed number of products for every mobile user according to the distribution of interest scores given the conzoom® type of the user.

7.5.6 Uniqueness and User–Defined Quantitative Constraints on Mobile Ads

Receiving the same ad multiple times naturally decreases the relevance of the ad as the therein presented information is not new. Primary key constrains in RDBMSs are an effective mechanism for guaranteeing that only unique combination of mobile users and mobile ads are considered for delivery. In the mobile marketing database the delivered ads are stored in a `mobile_ad_delivery` table with the following schema: ⟨`pid, bid, prodid, delivery_time`⟩. Placing a primary key constrain on the first three columns guarantees that a mobile ad is delivered at most once to a mobile user. Recording the *delivery time* allows the control of the re–delivery of mobile ads after a certain period of time has passed. For clarity, the `mobile_ad_delivery` table is omitted from the ER diagram in Figure 7.2.

As the number of mobile ads increases, or the other constraints on the delivery of mobile ads weaken, the number of mobile ads delivered to a mobile user will naturally increase. After a certain number of ads have been delivered to the user, any additionally delivered ad, while maybe relevant, will likely be perceived as annoying. Hence, the mobile user's ability to limit the number of delivered ads is important. This user–control can be effectively facilitated by the top–k query mechanism which is provided in most RDBMSs.

7.5.7 User–Defined ST Constraints on Mobile Ads

Time and location are important aspects of the context of mobile ads. Most users would consider receiving a mobile ad as intrusive or disturbing when receiving it during work hours or after a certain time in the evening in their homes. Hence, the mobile user's ability to prevent the delivery of mobile ads in certain regions of space and time are important. While the user–control of spatio–temporal constraints on mobile ads is not present in the mobile advertising database, the database can be easily extended to accommodate for this feature as follows. Users can specify *mobile ad profiles* by restricting certain spatial and / or temporal regions for mobile ad delivery. Then, spatio–temporal joins between the mobile ad profiles and mobile ads can be performed to further control the delivery of mobile ads.

7.5.8 Inferring Personal Interests and Relevance Based on Historical User–Behavior

Geo–demographic variables can be used to predict the general interests of an individual user, as explained in Section 7.5.3. However, since an individual user cannot be perfectly characterized by a few geo–demographic variables, it is likely that the predicted general interests differ slightly from the true, *personal interests* of the individual user. In the following, two methods are proposed to infer the personal interests of individual users.

The first method uses the locations that an individual user visits to infer the user's personal interests. A subset of the locations a user visits are commercial in nature, i.e., are businesses that offer products or services. During the lifetime of a user, the frequencies of how often the user visits particular businesses or types of businesses that offer particular products or services, can be recorded. Furthermore, periodical patterns can also be easily detected in the sequence of visits. An example of such a simple periodical pattern would be that a user visits a hairdresser approximately every second month. Basing the delivery of mobile ads on the personal frequencies of visited locations and periodic patterns ensures a closer match between the mobile ad and the true personal interest of the mobile user. The storage, maintenance, and derivation of the frequencies of visited locations and periodic patterns in those visits can either be done on the server or the client side. For the server side management, information about the frequencies of visits to particular types of businesses, offering specific products or services, are stored in a table with the following schema: `visit_frequency` = $\langle$`pid, prodid, num_visits`$\rangle$. Simple periodic patterns are stored in a table with the following schema: `pattern_period` = $\langle$`pid, prodid, last_visit_time, period`$\rangle$. To preserve clarity, these two tables are omitted from the ER diagram in Figure 7.2. The same information about personal interests can also be managed on the client side by a client application. Such a client side application, based on the current location of the mobile user, would have to be able to infer the type of business (product or service) that the user is currently visiting. This inference can either be aided by the server, or smart transmitters located at the businesses could communicate the required information to the client application.

The second method uses the information about how an individual user has reacted to mobile ads received in the past. More specifically, assume that a mobile user u received mobile ad a at time t. If at any time t' within a time period δt after time t the mobile user u visits the business that offered mobile ad a, i.e., $uloc(u, t') = adloc(a)$ and $t' - t \leq \delta t$, it is considered as a strong indication of interest of the user towards the mobile ad. Similarly, if the user does not visit the offering business within the δt period, it is considered as a weak indication of the user's indifference towards the mobile ad. The two events mentioned above can thus be considered as positive and negative feedback in an active control loop, respectively. The positive and negative feedback values can be quantified and summed over the lifetime of the user for specific businesses or types of businesses, i.e., products or services. After the control loop sum for a particular business or a product or service goes below a certain (user–defined) threshold, the business or product or service is "black–listed" for the user, i.e., no more ads are delivered from the business or for the product and service to the user. The information needed to manage the control loops is already stored in the LBA database. In particular, the locations of users at time t are stored in the `trajectory segment` table, and the received ads are stored in the `mobile_ad_delivery` table. Positive feedback conditions in the control loops can be checked by joining the recently changed location of mobile user to the locations of the locations of mobile ads that have been delivered within the last δt period. Negative feedback conditions in the control loops are indicated by the ageing of delivered mobile ads, i.e. a tuple ⟨`pid, bid, prodid, delivery_time`⟩ in the `mobile_ad_delivery` table indicates a negative feedback condition for mobile user with ID `pid`, for a particular business with ID `bid`, or product or service with ID `prodid` if `delivery_time` + δt is less than or equal to the current time. The same information can also be stored and managed in a similar fashion on the client side.

For both methods, both the server side and the client side approaches have advantages and disadvantages. The server side approach requires that personal behavioral data is stored on the server. This raises questions about scalability and privacy–related issues. In comparison, the client side approach requires a client application that manages the personal behavioral data on the client device. The client side approach seems to be more scalable and privacy–protecting, but in the case of loss or theft of the device, issues regarding the misuse of the sensitive personal information can arise.

7.5.9 An Operational LBA Database

The so far presented LBA database was developed for simulation and estimation purposes. As it is presented it can be commercially used as a tool to forecast LBA exposure and penetration. However, the LBA database can easily be altered to support the online management of location–based advertising. In an online, operational setting it is assumed that the mobile units of the mobile users periodically, but not necessarily at regular time intervals, communicate their position to the server. In such an operational setting the only necessary alteration to the LBA database is that instead of storing the historical trajectories of mobile users, the current locations of mobile users are stored. These locations can be represented as point geometries in the LBA database. Spatial queries to determine proximity between locations of mobile user and mobile ads can be implemented much in the same way using spatial joins. To manage mobile ads, periodically, relevant mobile ads are selected and delivered to mobile users who recently changed their location.

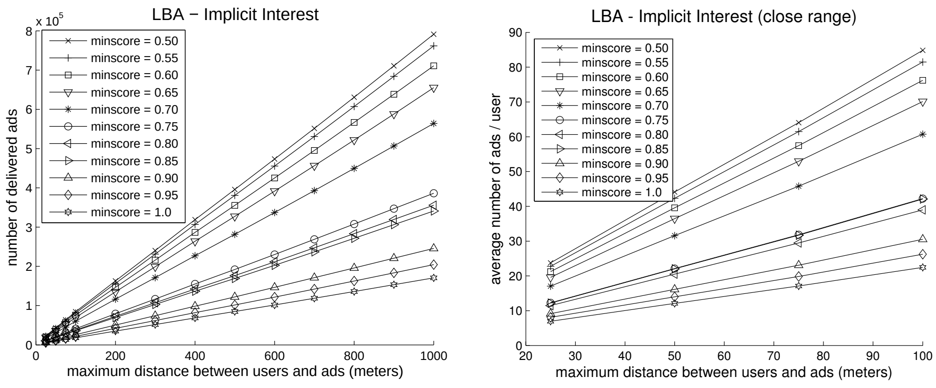
7.6 Proposal for a Revenue Model for LBA

A viable revenue model is a necessary prerequisite for successful, commercial LBA. There are essentially three parties involved in LBA: 1) the advertiser, 2) the consumer (mobile user), and 3) the LBA service provider or operator.

As in most other advertising media, and in LBA too, the advertiser pays for the majority of cost of advertising. These costs are for paying the other two parties for the participation in (consumer) and the facilitation of (operator) LBA. The incentive in doing so is clear: to increase the revenue of the business doing the advertising.

Most people do not like advertising. Some advertisements, such as advertisements on billboards, they cannot escape. Some they are willing to endure in return of other services, for example newspapers and commercial TV. Finally, some, such as direct mail or commercial fliers, they may choose to opt–out from. Since, according to EU law, conducting LBA requires the informed consent of the consumer[1] [15], the need for a clear consumer incentive is eminent. One way to motivate the consumer is to provide her/him with value–added services. One example of such a value added service could be a recently proposed Location–Aware Mobile Messenger that facilitates user–friendly communication and coordination between users [15]. Another possible value–added service can be the free delivery of non–commercial information, such as for example information about traffic or weather. The consumer can also be financially motivated through electronic coupons or reward programs.

[1]Directive on privacy and electronic communications (2002/58/EC, article 13(1)) involves asking the users' permission to send unsolicited advertising messages via all electronic communications for marketing purposes. Most countries outside of the EU also enforce similar legislative regulations.



(a) Larger Values for *maxdist*. (b) Smaller Values for *maxdist*.

Figure 7.4: Number of Delivered Ads (Implicit Interest) to a Population of 1000 Mobile Users for Various *minscore* and *maxdist*.

The operator charges the advertiser for the services provided. The cost for these services can be determined based on 1) a flat rate per LBA campaign, 2) the number of delivered mobile ads, 3) the weighted number of delivered mobile ads taking into account the interests scores, or 4) the weighted number of delivered mobile ads taking into account the reactions of the mobile users to the received mobile ads, i.e., strong interest or indifference. An accounting module for either one of the service cost schemes can easily be facilitated by the so far presented LBA database.

7.7 Experiments and Results

Two sets of experiments (implicit and explicit interest case) were performed to measure the capacity of the mobile advertising channel under various *maxdist* and *minscore* settings. The estimation are based on (1) 4,314 businesses in Copenhagen, Denmark offering one or many of the 40 hand–selected products or services, (2) the simulated movements of 1000 randomly selected simulated mobile users during the course of seven days (on average 3,800 trajectory segments per day). Scores for implicit interests were modelled as described above. To simulate explicit interests, 1 product or service of interest was assigned to every simulated mobile user, as described above.

Figure 7.4(a) shows the number of delivered ads during the course of the first day in the implicit case. As expected, the number of delivered mobile ads increases as the *minscore* is decreased or the *maxdist* is increased. The rather surprising, close to linear relationship between the number of deliverable ads and the maximum distance criteria is due to the following facts. Simulated mobile users move from one location

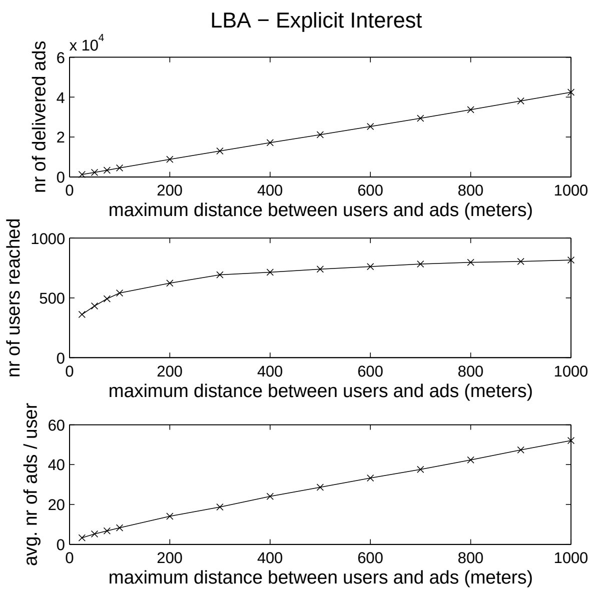
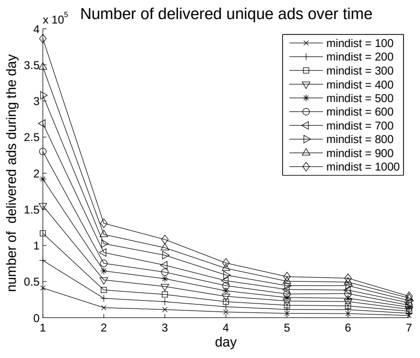


Figure 7.5: Statistics About the Delivered Ads (Explicit Interest) for Various *maxdist*.

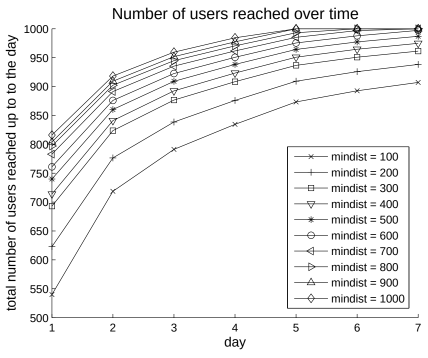
to another with the objective to perform an activity. These activities are tied to a subset of the businesses that advertise. Hence, the businesses that advertise the products or services often lie on the actual streets that the trajectories follow. If businesses are assumed to be uniformly distributed on those streets, then the relationship is indeed expected to be linear, as the number of businesses "reachable" along a street grows linearly with *maxdist*. Another, rather interesting result is the sheer number of mobile ads that can be delivered to a small set of 1000 users within a course of a day. Even for $maxdist = 500$ meters (arguably a worthwhile detour for the mobile user) and *minscore* = 0.9 (quite high match in direct marketing) the average number of delivered ads to a user is about 100. This represents a huge marketing potential.

The same numbers are likely to be viewed as alarming by many mobile users. As it is shown in Figure 7.4(b), even for rather high minimum interest scores for very low *maxdist* ranges the average number of ads delivered to a mobile user during the course of the first day is in the range of 6 to 40. This is a rather large number of ads to be received on a small, by many considered as extremely personal, mobile device. Hence, to avoid bad reputation, businesses interested in employing or facilitating mobile advertising should make great efforts to provide simple yet effective user–controls on the number of received mobile ads, as suggested in Sections 7.5.6 and 7.5.7.

Figure 7.5 shows some statistics about the number of delivered mobile ads in the explicit interest case. Similar observations can be made about the relationship



(a) Number of Deliverable Ads by the Day.



(b) Number of Reached Mobile Users Over Time.

Figure 7.6: Effects of the Uniqueness Constraint on the Number of Delivered Ads to a Population of 1000 Mobile Users Over a Period of Seven Days for Various *maxdist*.

between the *maxdist* and *minscore* parameters and the number of delivered mobile ads as in the implicit case. However, it is surprising that even though every mobile user is only interested in exactly 1 of the 40 products or services, due to the presence of a large number of businesses offering those products and services, the number of deliverable ads is rather high even for small values of *maxdist*. It is also important to note, that for $maxdist < 100$, over half of the mobile users do not receive any mobile ads (middle graph). Hence the mobile users who are interested in getting good deals on products or services of their interest, have to set their *maxdist* values appropriately high.

Since most mobile users need to perform mandatory activities at specific locations, such as going to work or coming home daily, and since they tend to perform their other activities either around, or along the way in–between, these specific locations, they tend to move around in approximately the same space from day to day. Hence, because of the uniqueness constraints on mobile ads, for a fixed set of static mobile ads, the number of deliverable mobile ads per day is expected to decrease over time. This decrease is shown in Figure 7.6(a) for the implicit case for an interest score of 0.75 for various $mindist$ for a period of seven days. As it can be seen in Figure 7.6(a), the rate of decrease is also decreasing with time. This is due to the fact that the number of infrequent and less regular (non–daily) destinations of an individual mobile user are limited, and over time are eventually visited by the mobile user, at which time all the relevant mobile ads are delivered. Since the number of such irregular destination are limited, and the number of unvisited ones are decreasing as time progresses, the chance of a mobile user visiting an unvisited irregular destination is also decreasing with time. Consequently, the number of deliverable mobile ads per day also decreases with time. At the same time, because of the slight variation in day–to–day movements of mobile users, the penetration rates of LBA increase. More specifically, in the explicit interest case, Figure 7.6(b) shows that the number of reached users increases over time.

User–defined ST constraints on mobile ads are used to disallow the delivery of mobile ads at specific locations and/or times. To test the reduction in deliverable mobile ads, in a set of experiments the delivery of mobile ads was not permitted when mobile users were either at home or at work[2]. Somewhat counter intuitively, no reduction in deliverable mobile ads was observed. This result can be explained by the following two facts. First, in the LBA database movements of mobile users are represented as continuous trajectories. Second, the set of mobile ads used in the estimation were constant during the estimation period, i.e., all mobile ads were effective during the course of the whole simulation. Hence, after mobile users left their home for the *first* time, and travelled a short distance away from home, they received all the relevant mobile ads, and similarly they received all the relevant mobile ads as they for the first time approached their work place. The effects of user–defined ST constraints on mobile ads would be more observable in LBA environments where mobile ads are dynamic and have short lifetimes. For example a cinema, after realizing that over 90% of the seats are empty 30 minutes prior to the movie, might want to run a "50% off" LBA campaign for 30 minutes only. Such dynamic mobile ads with short lifetimes will be filtered out by user–defined ST constraints, if applicable. Extensions to the LBA database to handle such dynamic LBA conditions are trivial and are left for future work.

In summary, the experiments show that the capacity of the location–based advertising channel is very high indeed, even for relatively small settings for minimum distance, and relatively specific interest settings. This is good news for LBA advertisers, as they can expect to reach a large set of potential customers.

7.8 Conclusions and Future Work

The aim of this paper was to investigate the capacity of the Location–Based Advertising (LBA) channel. The paper proposed two types of LBA models (implicit vs. explicit interest) and described a relational database for the effective management of both types of LBA. Using a number of real–world data sources and simulated but realistic movement data of mobile users, the paper gave estimates on the number of deliverable mobile ads in both the implicit and the explicit interest cases. Experimental results show that the capacity of the LBA channel is rather large implying a huge marketing potential. At the same time, the potentially large number of mobile ads could be alarming to mobile users, hence the paper warns businesses interested

[2]Home and work places for mobile users have been identified by slightly altering the output of ST–ACTS.

in LBA to provide the mobile users with adequate means to control the number of delivered ads and the time and place of delivery.

Future work is planned along two paths. First, while the presented LBA framework considers LBA both from the mobile users' and advertisers' perspective, the provided estimates are valid only if mobile users are willing to accept unlimited mobile ads at all times and places. Incorporating user–defined constraints on mobile ads, as described in Sections 7.5.6 and 7.5.7, will provide better estimates on the true audience size of LBA. Second, in the implicit interest case, the relevance of a mobile ad is estimated using a simple scoring model which is based on a consumer segmentation that divides users into 29 different consumer groups. However, as it is pointed out in Section 7.5.8, in real life, no two users' interests are *exactly* the same, hence a given mobile ad does not have the same relevance to them. Hence, altering the scoring model to include the personal interests of the individual mobile user, which are derived from historical behavior of the mobile user – such as the type businesses the user has previously visited or the user's reactions to previously received mobile ads – will allow targeting the individual mobile user with even more *relevant* and *personalized* mobile ads. Since, in the current simulation of mobile user movements, the possible influence of mobile ads on the future movements of mobile users is not accounted for, the evaluation of the effects of personal interest scoring are left for future research.

Chapter 8

Privacy–Preserving Data Mining on Moving Object Trajectories

The popularity of embedded positioning technologies in mobile devices and the development of mobile communication technology have paved the way for powerful Location–Based Services (LBS). To make LBSes useful and user–friendly, heavy use is made of context information, including patterns in user location data which are extracted by data mining methods. However, there is a potential conflict of interest: the data mining methods want as precise data as possible, while the users want to protect their privacy by not disclosing their exact movements. This paper aims to resolve this conflict by proposing a general framework that allows user location data to be anonymized, thus preserving privacy, while still allowing interesting patterns to be discovered. The framework allows users to specify individual desired levels of privacy that the data collection and mining system will then meet. A privacy–preserving method is proposed for a core data mining task of *finding dense spatio–temporal regions*. An extensive set of experiments evaluate the method, comparing it to its non–privacy–preserving equivalent. The experiments show that the framework still allows most patterns to be found, even when privacy is preserved.

8.1 Introduction

The efficient management of moving object databases has gained much interest in recent years due to the development of mobile communication and positioning technologies. A typical way of representing moving objects is to use the trajectories. Much work has focused on the topics of indexing, query processing and data mining of moving object trajectories, but little attention has been paid to the preservation of privacy in this setting. In many applications such as intelligent transport systems (ITS) and fleet management, floating car data (FCD), i.e., tracked vehicle locations, are collected, and used for mining traffic patterns. For instance, mining vehicle trajectories in urban transportation networks over time can easily identify dense areas (roads, junctions, etc.), and use this for predicting traffic congestion. By data mining the periodic movement patterns (objects follow similar routes at similar times) for individual drivers, personalized, context–aware services can be delivered. However, exposing location/trajectory data of moving objects to application servers can cause threats to the *location privacy* of individual users. For example, a service provider with access to trajectory data can study a user's personal habits. It is not enough to keep the user ID secret, since common locations such as the home and office address can be found by correlating historical trajectories, followed by cross–referencing these locations with, e.g., Yellow Pages, to reveal user identity. Privacy–preserving data mining of moving object trajectories has not been addressed in the literature. The challenge of obtaining detailed, accurate patterns from anonymized location and trajectory data is the motivation for this paper.

This paper makes a number of novel contributions that together constitutes an effective method for trajectory data collection and mining that preserves user location privacy. First, the paper proposes a novel *anonymization model* for preservation of location privacy on moving object trajectories. Here, the users specify their requirements of location privacy, based on the notions of *anonymization rectangles* and *location probabilities*, intuitively saying how precisely they want to be located in which areas. Second, the paper shows a *common problem* with existing methods based on the notion of *k–anonymity*. This problem allows an adversary to infer a commonly occurring location of a user, e.g., the home address, by correlating several observations. Third, the paper presents an effective *grid–based framework* for data collection and mining over the anonymized trajectory data. The framework is based on the notions of *anonymization grids* and *anonymization partitionings* which allow effective management of both the user–specified location privacy requirements and the anonymized trajectory data. Along with the framework, three *policies* for constructing *anonymization rectangles*, called *common regular partitioning*, *individual regular partitioning*, and *individual irregular partitioning* are presented. These policies avoid the problems in existing methods. Fourth, the paper presents a *client–server architecture* for an efficient implementation of the system. A distinguishing feature of the architecture is that anonymization is performed solely on the client, thus removing the need for trusted middleware. Fifth, the paper presents techniques for solving a basic trajectory data mining operation, namely *finding dense spatio–temporal areas*. In an extended technical report [36], the same framework and techniques are also evaluated on a more complex data mining operation, namely *finding frequent routes*. The techniques are based on probabilistic counting. Finally, *extensive experiments* with a prototype implementation show the effectiveness of the approach, by comparing the presented solutions to their non–privacy–preserving equivalents. The experiments show that the framework still allows most patterns to be found, even when privacy is preserved. In summary, this paper it believed to be the first to consider the topic of data mining on anonymized trajectory data.

Privacy protection in databases has been a core area in the database research community and many related topics have appeared in the literature, such as access control, inference control and statistical databases. To protect the privacy of LBSes users, three existing solutions [43, 44, 74] propose to use a trusted middleware (an anonymizer) that maintains location updates and queries between the LBS users and LBS server. Each time a query request is sent from a LBS user, the anonymizer, in the spirit of *k–anonymity* [89], encloses the query location in a “cloaking” rectangle that includes both the query location and the locations of $k-1$ other users, and sends the query to the LBS server with the cloaking rectangle. The LBS server returns a superset of the results and the final results are filtered by the anonymizer and sent back to each LBS user.

This method for anonymizing locations and trajectories has several problems. First, it requires trusted middleware. Second, while [74] provides an effective solution for finding locations of the other $k-1$ users in the presence of such trusted middleware, a solution to the same task in an environment that contains only untrusted components is unknown and likely to be computationally prohibitive. Third, the notion of location privacy that is guaranteed by *k–anonymity* may not be satisfactory in the case where a large number of moving objects stay in a small area where users do not want to be observed (such as a red light district). This problem can be eliminated by requiring cloaking rectangles to have a minimum area [74]. Fourth, the cloaking rectangles calculated for the same user for the same location at different times depends on locations of the other $k-1$ users, and hence may vary in extent and location. This, in a sense *non–deterministic* or *probabilistic* nature of cloaking rectangles sacrifices location privacy, as demonstrated later. Finally, traditional mining methods cannot be easily and effectively adapted to the anonymized location or trajectory.

As a result, the present paper does not consider *k–anonymity* and does *not* assume the existence of trusted middleware for providing the *k–anonymity* rectangles. Instead, it focuses on novel ways to conceal the actual moving object trajectories while still allow the data mining algorithms on the LBS server to extract detailed, accurate traffic patterns and rules from the anonymized trajectory data. Note that the proposed solution does *not even aim* to provide k–anonymity. The reason is that for some applications, e.g., traffic services in remote areas, even a rather small k will cause the reported rectangles to become extremely large, and thus worthless for the purpose of mining. Instead, the proposed solution performs a *spatial anonymization* that meets the user's requirements for location privacy.

Spatio–temporal data mining is an on–going topic in the database community. Approaches have appeared for finding dense areas of moving objects [47,59,90] and extracting spatio–temporal rules and patterns [30,95]. The present paper is focused on discovering areas with potential traffic jams and roads that are frequently used by drivers. Two very related papers [47,59] study the querying of spatio–temporal regions with a high concentration of moving objects. The first paper [47] divides the data space into a uniform grid so that the density query is simplified as reporting cells that satisfy the density conditions. This solution provides fast answers, but can lead to *answer loss* (as termed in the second paper [59]), such as regions that cover boundaries of several cells with a high density of objects (but each individual cell does not contain enough number of objects to be dense). The second paper [59] provides a new definition of density query that eliminates answer loss and proposes a two–phase filter–and–refinement algorithm for computing the density queries. A method to provide approximate answers to *distinct* spatio–temporal aggregation is proposed in [90], where aggregation is grid–based, and the distinct criterion is time– and space–effectively solved by combining a spatio–temporal index (aRB–tree) and sketches.

A lot of recent research work has focused on techniques for privacy–preserving data mining [4]. This topic has appeared due to the advances in data collection and dissemination technologies which force existing data mining algorithms to be reconsidered from the point of view of privacy preservation. Various papers have recently addressed privacy–preserving data mining. Important techniques include perturbation, condensation, and data hiding with conceptual reconstruction. Paper [96] presents a good review of these techniques. The techniques proposed in this paper follow the spirit of a common strategy used for privacy–preserving data mining, namely *generalization*.

The rest of this paper is organized as follows. Section 8.2 discusses anonymization models of trajectory data. Section 8.3 presents the grid–based framework, while Section 8.4 presents an empirical evaluation. Finally, Section 8.5 concludes and points out future directions for research.

8.2 Spatio–Temporal Anonymization

For the simplicity of the discussion, assume that the time domain $\mathbb{T}$ is totally ordered and use the non–negative numbers as the time domain. Let the trajectory of a moving object in 2–dimensional (2D) space be described by a sequence of tuples $S = \langle (loc_1, t_1), \ldots, (loc_n, t_n) \rangle$ where $loc_i \in \mathbb{R}^2$ ($i = 1, \ldots, n$) describe locations, and $t_1 < t_2 < \ldots < t_n \in \mathbb{T}$ are irregularly spaced but temporally ordered time instances, i.e., gaps are allowed.

The trajectory is anonymized by reducing the spatio–temporal resolution of the 2D space. One basic method is to enclose the trajectory into one or more space–time rectangles, denoted as an *anonymization rectangles*. A formal definition is as follows:

Definition 7 Given an area size $areasize \in \mathbb{R}^+$ and a probability threshold $maxLocProb \in [0; 1]$, an **anonymization rectangle** satisfying ($areasize$, $maxLocProb$) for a moving object o is a three–tuple (R, t_s, t_e), where $t_s < t_e \in \mathbb{T}$ are two time instances, and R is a 2D rectangle such that the maximum probability that can be *inferred* about o being in any subregion A of size $areasize$ in R during the period $[t_s, t_e]$ is at most `maxLocProb`.

Definition 8 Given an area size $areasize \in \mathbb{R}^+$, this maximum probability that can be *inferred* about the whereabouts of object o inside R is called as the **location probability** of R and is denoted as $R.LocProb$.

Privacy preservation in spatio–temporal data sets is challenging because spatio–temporal data sets are so rich in correlations, allowing many "privacy attack" strategies that are difficult to counteract and sometimes even to anticipate. The proposed method is believed to protect against a few obvious threats, namely, 1) detection of frequent private/personal/individual locations due to self–correlations in historical spatio–temporal (trajectory) data sets, 2) detection of the current position due to physical mobility constraints on objects (maximum speed, road network, spatio–temporal restrictions in general).

In the definitions *inferred* is emphasized, because the straight–forward, uniform spatio–temporal probability distribution for the location of an object o does not hold for any rectangle $R \in \mathbb{R}^+$. By relating external spatial and/or temporal data sources, which put limitations on the possible locations of o, more specific distributions can be derived that sacrifice the privacy of o. This is illustrated in Figure 8.1, where anonymization rectangle R of o is composed of 4 unit–area cells (c_1, c_2, c_4, c_5). Not combining any external data sources, $R.LocProb = 1/4$. Knowing that cells c_1 and c_4 are covered by water, $R.LocProb = 1/2$. Finally, knowing about the location and opening hours of the Nature Resort Park in cell c_2 and the current time (8am), $R.LocProb = 1$. Clearly, relating more and more spatio–temporal, external data sources to R raises the location probability of it, and guarantees less privacy for o. One natural way to guarantee a location probability of at most `maxLocProb`, is to spatially, or temporally, extend R to $R_{extended}$, such that $R_{extended}.LocProb \leq$ `maxLocProb`. Section 8.3.2 describes how to do this in practice.

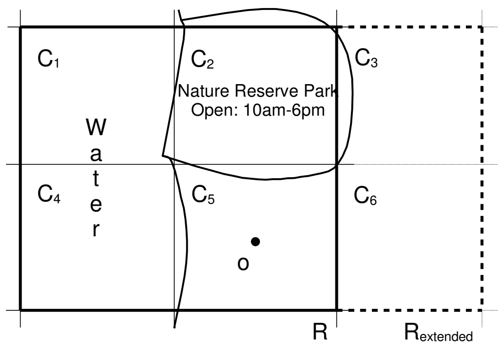


Figure 8.1: Location Privacy.

If the currently known spatio–temporal probability distribution for the location of an object o is denoted as PD_o, then any kind of "extra" external spatio–temporal information can be modelled as a function $F(PD_o)$ that returns a new spatio–temporal probability distribution PD'_o. If the location probability of o at certain locations is then over the threshold `maxLocProb` with the new distribution, there is a problem that needs to be handled somehow, most often by enlarging the area partitions.

Intuitively, the whole trajectory of a moving object can be enclosed into a single rectangle so that the anonymity of the trajectory is preserved. However, as the trajectories are often very long, the rectangles can be very big so that it becomes impossible for the data mining algorithms to return any useful results. The proposed method provides an **anonymized format** of the trajectory by cutting a long trajectory into pieces and enclosing each piece in an anonymization rectangle. This format can give opportunities for doing data mining without sacrificing location privacy.

8.2.1 Practical "Cut–Enclose" Implementation

The "cut–enclose" procedure splits the whole trajectory of a moving object o into a set of polylines which correspond to a set of time periods $\{[t_1, t_2], [t_2, t_3], [t_3, t_4], \ldots, [t_{k-1}, t_k]\}$, such that at any time instance $t_i \in \{t_2, t_3, \ldots, t_{k-1}\}$ o's trajectory crosses an edge between two neighboring anonymization rectangles R_i and R_{i+1}. Since around this instance t_i, o is more likely to be close to the edge between R_i and R_{i+1}, $R_{i+1}.LocProb$ will temporarily be higher, which might sacrifice the location privacy of o. More specifically, from the times spent in the previous anonymization rectangles, their sizes, and relative locations to each other, a malicious server can easily maintain a linear movement model of o. Using this movement model, when o sends the anonymization rectangle R_{i+1}, the malicious server can *deduce* a *possible location range* R^* of o, such that $R^*.LocProb >$ `maxLocProb`. To avoid this situation and preserve the location privacy of o, a **time delay factor** $\delta_{[i,i+1]}$ for delaying the

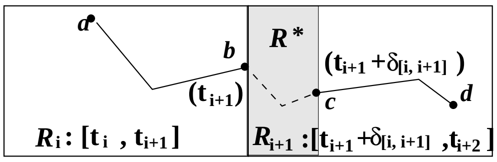

a
b
R*
$(t_{i+1} + \delta_{[i,i+1]})$
(t_{i+1})
c
d
$R_i : [t_i , t_{i+1}]$
$R_{i+1} : [t_{i+1} + \delta_{[i,i+1]} , t_{i+2}]$

Figure 8.2: Time Delay Factor.

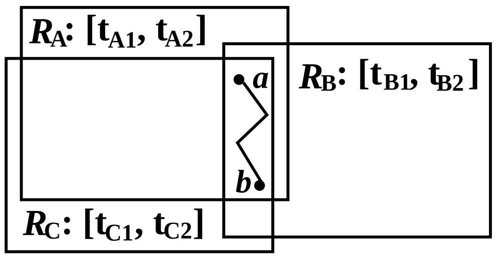
sending of the anonymous rectangle R_{i+1} after leaving R_i is introduced. The factor $\delta_{[i,i+1]}$ can be calculated as follows. Object o can maintain the same linear movement model about its own movement as the malicious server can. Hence, at any time instance $t^* > t_i$, having entered R_{i+1}, o can calculate R^* and $R^*.LocProb$. As time progresses, the size of R^* is monotonically increasing and $R^*.LocProb$ is monotonically decreasing. Hence, at some time point $t^s > t_i$, when the associated $R^*.LocProb \leq$ `maxLocProb` it is *safe* for o to send R_{i+1} to the server. The time delay factor is then $\delta_{[i,i+1]} = t^s - t_i$.

Most moving objects are confined to road networks. In the presence of road networks, more sophisticated movement models are possible. Actual values for the time delay factor have been investigated for a number of network–based movement models on real–world data sets in [14], but this work had a different aim, namely to aid tracking.

8.2.2 Problems with Existing Methods

To construct an anonymization rectangle for a given piece of trajectory, one naive method is to randomly choose a location in the vicinity of the trajectory and use this location as the center to build the anonymization rectangle based on a pre–defined size. Another method, motivated from the discussion of *location k–anonymity* in the literature [43, 44, 74], is to build the anonymization rectangle that enclose this piece with trajectory pieces of $k - 1$ other moving objects.

However, these two methods can lead to an undesired *loss of location privacy*. Sensitive locations that need to be kept private, or trajectory pieces that lead to these, are often re–visited by the objects many times, at a similar time of day. For example, objects (users), in the evening hours return to their *home* using the same path (trajectory piece). If on different occasions the anonymization rectangles for this trajectory piece are constructed in a *non–deterministic* way, the location of the trajectory piece can be narrowed down to the intersection of these anonymization rectangles. This leads to an undesirable loss of privacy. In the example on Figure 8.3, object o returns to its *home* `b` using the same trajectory piece $[a, b]$ on three different occasion at the same time of the day. On the three occasions, three anonymization rectangles R_A, R_B, and R_C are constructed, such that they contain both the trajectory piece $[a, b]$



R_A: [t_{A1}, t_{A2}]
R_B: [t_{B1}, t_{B2}]
R_C: [t_{C1}, t_{C2}]
a
b

Figure 8.3: Overlapping Area.

and the location b. Based on the multiple visits, the location of $[a, b]$ can be narrowed down to the small overlapping area of the anonymization rectangles.

The next section presents a grid–based solution and several methods for constructing anonymization rectangles in a *deterministic* way on this grid, thereby avoiding the privacy loss described above. The grid–based framework also allows for an efficient implementation of the "cut–enclose" procedure described in Section 8.2.1.

8.3 A Grid–Based Solution

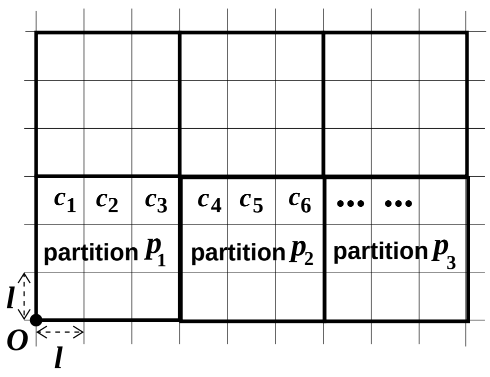
A basic method to anonymize location is to reduce the spatial resolution. Thus, instead of randomly constructing the anonymization rectangles or building the rectangles based on trajectories of other moving objects, the anonymization rectangles for all moving objects is built based on a single, pre–defined 2D grid. The following subsections discuss the solution in detail.

8.3.1 Grid–Based Anonymization

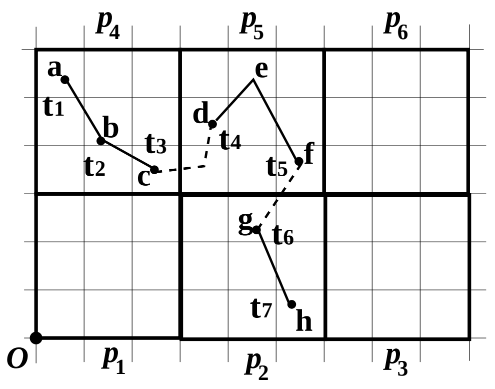
Denote the whole 2D Euclidean space as $\mathbb{R}^2$ and proceed to define an anonymization grid and anonymization partitioning as follows.

Definition 8.3.1 An **anonymization grid** (briefly, a grid) G is a uniform grid of $\mathbb{R}^2$ with a pre–defined $O \in \mathbb{R}^2$ as the starting point and a side length l. An **anonymization partitioning** (briefly, a partitioning) is a set of pairwise disjoint sets of grid cells covering all of G.

As illustrated in Figure 8.4(a), given a starting point $O \in \mathbb{R}^2$, the anonymization grid (briefly, the grid) G uniformly divides the whole space into square–shaped **grid cells**, each of which has side length l. Each grid cell has an ID value, such as $c_1, c_2, \cdots$ in Figure 8.4(a). A *partition* of a partitioning that is defined on the grid is a set of grid cells.



(a) Anonymization Grid and Partitioning.



(b) Anonymized Trajectory.

Figure 8.4: Grid–Based Anonymization of Trajectories.

Next, several methods for constructing **anonymization partitionings** based on the anonymization grid are developed. All of the partitionings are constructed *deterministically,* thereby avoiding the privacy loss due to overlapping partitions.

Common Regular Partitioning (CRP): The simplest method is to define a single, regular partitioning that is used by all the objects. A partitioning is called **regular** if all the partitions are rectangles with side lengths $i_x \times l$ and $i_y \times l$, where i_x and i_y are integers.

Such a regular partitioning can be seen as a coarser grid on the 2D space. As illustrated in Figure 8.4(a), given the grid (the grid of thin lines), the partitioning (the grid of thick lines) is defined by an origin O and $i_x = 3, i_y = 3$. In the example the grid cells c_1, c_2, c_3 belong to the partition p_1. With the grid and partitioning, a moving object trajectory can be transformed to a set of non–overlapping anonymization rectangles to preserve anonymity. For instance, given the trajectory $\langle(a, t_1), (b, t_2), \cdots, (h, t_7)\rangle$ in Figure 8.4(b), a grid on the 2D space is built and the partitioning on the grid is made. The partitions are denoted as $p_1, \cdots, p_6$ in the figure and they are non–overlapping rectangles. As described in Section 8.2.1, given the time delay factor δ, the whole trajectory is cut into several pieces with $t_4 - t_3 = \delta$ and $t_6 - t_5 = \delta$. Then, the whole trajectory is transformed into a list of anonymization rectangles $\langle(p_4, t_1, t_3), (p_5, t_4, t_5), (p_2, t_6, t_7)\rangle$.

The above described partitioning guarantees the same minimal level of privacy for all users in any region of the space. This method of partitioning is termed Common Regular Partitioning (CRP).

Fundamental spatio–temporal data mining tasks, like finding dense spatio–temporal regions, are based on simple counts or identities of the users that are present in a given spatio–temporal region. Since in the CRP model all users report the same set of grid cells for the same location, the spatio–temporal granularity of any pattern found is lower bounded by the size of a partition. In the example in Figure 8.4(b), the size of the common partition is 9 grid cells, hence the smallest dense ST–region that can be found will be 9 grid cells.

Individual Regular Partitioning (IRP): Not all objects require the same level of location privacy. This requirement of individual objects can easily be accommodated in the anonymization grid–based framework. Objects requiring higher levels of privacy construct and use a regular partitioning with larger partitions, while objects requiring lower levels of privacy define and use a regular partitioning with smaller partitions. This method of partitioning is termed Individual Regular Partitioning (IRP).

Besides being more flexible in terms of the objects' privacy requirements, the IRP method allows the discovery of patterns of spatio–temporal granularity that is equal to the size of a single grid cell (if enough data is present).

Individual Irregular Partitioning (IIP): Objects may have different location privacy requirements in different regions of space. For example, most objects (users) desire a higher level of location privacy when being at *home* or the *work place* than when being in transition or when being in other general areas of the city. This requirement of individual objects can again be easily accommodated in the proposed anonymization–grid–based framework. Objects can be allowed to individually define privacy levels for regions in space that reflect their needs. The definition of these regions can be either manual, or can be aided by discovering frequent (presumably sensitive) locations of individual objects. Since the selection or discovery of these sensitive locations can be accomplished on the client side, it can be kept private. This method of partitioning is termed as Individual Irregular Partitioning (IIP).

The IIP method also allows the discovery of patterns of spatio–temporal granularity that is equal to the size of a single grid cell. The additional ability to define spatially varying privacy levels not only adds more privacy control, but it is also expected to allow the discovery of more patterns with finer spatio–temporal granularity. This is due to the fact that most objects are expected to require higher levels of location privacy in relatively small subregions. The more detailed patterns are expected to be more useful for ITS applications.

With the proposed grid–based framework, the knowledge one can infer about the whereabouts of a user does not depend on the number of samples collected. The certainty of the inference only depends on the amount of external spatio–temporal information available for the anonymous rectangle.

8.3.2 System Architecture

The grid–based solution is implemented based on a client/server architecture. As illustrated in Figure 8.5, the server side has three components, the *anonymity com-*

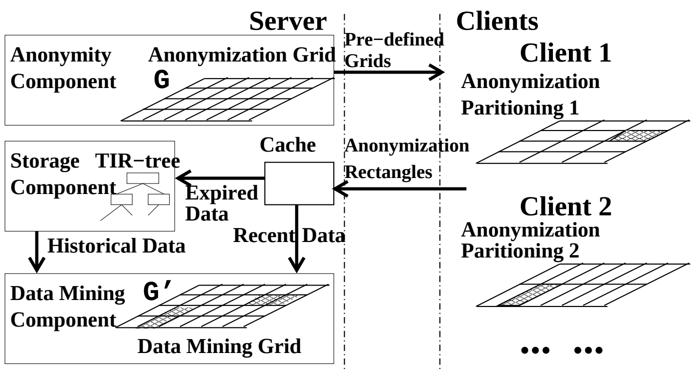


Figure 8.5: System Architecture.

ponent which defines one or more grids and communicates them to the client, the *storage component* which collects the anonymization rectangles sent from the clients and stores the data on disk, and the *data mining component* which discovers certain patterns and rules either directly from the incoming data stream or from the historical data retrieved from the storage component.

The clients are responsible for accepting an anonymization grid and developing a partitioning based on the grid. In practice, the partitioning will be made in one of two ways: a) the user selects among a small number of pre–computed partitionings to find one that meets their privacy requirements, or b) the partitioning is computed by a dedicated program on the

client, based on user input about privacy requirements. Both a) and b) take available background knowledge into account. The framework can also handle the presence of road networks. If road networks are dense compared to the partition size, the framework can be used without modification. If not, the partitions have to be enlarged so that each partition contains enough road to get a location probability that is comparable to those of the other partitions.

These client– and grid–specific partitionings are stored on the clients and only anonymization rectangles (in the form of sets of grid cells), which are computed *at the clients*, are transmitted to the server. It is assumed that the client has a fair amount of storage and CPU power, but not more than what can be found in most currently available smartphones or PDAs.

Saving a partitioning at the client side does not take much space. For a regular partitioning, where partitions form a regular grid, it is enough to store the starting point and the side length of the partitioning. Finding the partition that corresponds to a location is a matter of simple arithmetic. For a non–regular partitioning, where partitions do not form a regular grid, i.e., are of different size and/or shape, partitions can be kept in an R–tree. Finding the partition that corresponds to a location can be done by issuing a stabbing query on the R–tree for the location. The communication cost between the clients and the server is very low since the grids can be described with the starting point and side length l of the grid, and the anonymization rectangles only involves a few data fields (i.e., coordinates of the client's current partition and the time instances).

The clients always send their current anonymization rectangle, i.e., partition, to the server. When the anonymized data is transmitted to the server, it is stored it in two places. To be able to perform data mining on historical data, the data is first stored in a time–interval R–tree (TIR–tree in Figure 8.5) on disk. The TIR–tree is a 1–dimensional R–tree that indexes the data on the time intervals. To be able to perform online data mining on the current data, the data is also stored in a *cache*, with a FIFO replacement policy as follows. According to the size of the cache, when a new anonymization rectangle of a moving object arrives, either the previous anonymization rectangle of this moving object (if in the cache) or the oldest data in the cache is deleted.

The system architecture in Figure 8.5 supports data mining on both historical trajectories and recent data. Each anonymization grid in the anonymity component corresponds to an in–memory instance of the same grid in the data mining component. For instance, the anonymization grid `G` in Figure 8.5 corresponds to the data mining grid `G'` (it is assumed that the data mining component has enough memory to store `G'`). Based on this architecture, in the following, algorithms for discovering *dense ST–areas* on the anonymized trajectory data is presented.

8.3.3 Finding Dense Spatio–Temporal Areas

Discovering dense areas is one of the most common topics for spatial and spatio–temporal data mining. Existing research work has explored density clustering [20], spatio–temporal dense area discovery [95], and density queries [59]. For dense area discovery on the anonymized trajectory data, the most basic operation is to find those grid cells that contain a large amount of moving objects during specified time intervals. In the anonymized format, objects are present in a grid cell with some *probability* only. Hence, a **time interval *probabilistically* dense spatio–temporal area query**, or *dense ST–area query* for short is proposed, which can be seen as a basic, atomic operation for advanced dense area mining algorithms over the anonymization grid. Such advanced and complex data mining algorithms can be made by assembling this operations with other basic query types.

Specifically, suppose a moving object o corresponds to a partition P on a given anonymization grid `G`, a partition cell $p \in P$ contains o's trajectory during time interval $[t_s, t_e]$, and p includes grid cells $c_1, c_2, \ldots, c_k$. p is used as the anonymization rectangle for o's trajectory and each grid cell $c_i \in p$ has the location probability $c_i^o.LocProb = 1/k$ for o at any time instance during $[t_s, t_e]$. Let O^{c_i} be the set of moving objects whose anonymization rectangles include the grid cell c_i in at least one time instance during the time interval $[t_s, t_e]$. Then $c_i.count = |O^{c_i}|$ and $c_i.prob = \sum_{o \in O^{c_i}} c_i^o.LocProb/|O^{c_i}|$. Intuitively, $c_i.count$ is the *maximum* number of objects that *can* be inside c_i during $[t_s, t_e]$, while $c_i.prob$ is the *average* location probability of the objects that can be inside c_i during $[t_s, t_e]$. Consequently, $c_i.prob \times c_i.count$ is the *expected* number of objects inside c_i during $[t_s, t_e]$. Furthermore, the *pattern certainty* $c_i.cert = \prod_{o \in O^{c_i}} c_i^o.LocProb$ is defined as the probability of *actually* having $c_i.count$ number of moving objects inside of c_i during $[t_s, t_e]$.

A grid cell c_i is said to be **probabilistically dense** during $[t_s, t_e]$ if $c_i.count \geq$ `min_count` and $c_i.prob \geq$ `min_prob`, for some given threshold values `min_count` and `min_prob`. Thus, the **dense ST–area query** is formulated as follows:

Definition 8.3.2 A **dense ST–area query** $Q = ([t_s, t_e], \texttt{min\_count}, \texttt{min\_prob})$ retrieves all the grid cells whose corresponding $count$ and $prob$ values during $[t_s, t_e]$ are greater than or equal to `min_count` and `min_prob`, respectively.

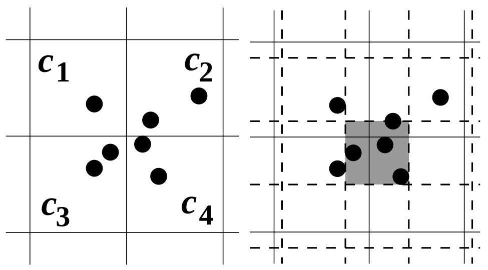
To process a dense ST–area query, the first step is to compute the $count$ and $prob$ values for each grid cell c_i for the specified time interval $[t_s, t_e]$. Based on the system architecture in Figure 8.5, a range query over the TIR–tree needs to be issued to find all the anonymization rectangles whose time periods have intersections with $[t_s, t_e]$. Results of the range query are used to fill in the $count$ and $prob$ values for each cell c_i of the data mining grid `G'`. Then the set of dense ST–grid cells is:

$$D = \{c_i : c_i.count \geq \texttt{min\_count} \wedge c_i.prob \geq \texttt{min\_prob}\}$$

During the query time interval $[t_s, t_e]$ a moving object can leave and later reenter a given grid cell c_i. To avoid counting such an object multiple times for c_i, a hash array of object IDs is maintained and values for $c_i.count$ and $c_i.prob$ are only updated when an object ID is encountered for c_i for the first time. If only approximate counts are considered, these can be more effectively obtained using the methods from [90].

As it will be seen in Section 8.4, the cut–off criteria for dense areas presented above is in some cases not strict enough, thus generating too many dense areas (false positives). To remedy this, the alternative *steepest slope* cut–off criteria is introduced, which is calculated by first sorting the expected counts for dense areas passing the first criteria in descending order, finding the deltas between any two consecutive values, and making the cut–off where the (negative) delta is the smallest, i.e., where the “slope” is steepest.

The above method is simple to implement but can not discover all the dense areas that have the size of a single grid cell. As illustrated in Figure 8.6(a), grid cells c_1, c_2, c_3, c_4 have several moving objects (big dots in the figure) and the threshold `min_count` = 4. None of the four cells are reported as dense since the *count* value of each is less than 4.



(a) Loss of Result. (b) Using the Second Grid.

Figure 8.6: Multi–Grid to Overcome Answer Loss.

To overcome such loss of results, the server provide several anonymization grids with different starting points (and perhaps side length values) and distribute these grids to the moving objects so that there are in equal amount of moving objects that build their partitioning based on each of the anonymization grids. This *Multi–Grid* approach can capture the answer loss that occurs with a single grid. As shown in Figure 8.6(b), the dense junction area of the four cells can be captured by the dark cell belonging to another grid. This *Multi–Grid* extension of the anonymization framework is left for future work.

The time interval dense ST–area query can be seen as an atomic operation over the anonymized trajectory data. Advanced and complex data mining functions can be made by assembling this operations with other basic query types.

8.4 Evaluation

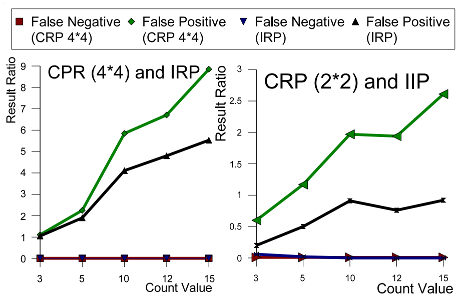
To evaluate dense ST–area query algorithms, Brinkhoff's network–based generator of moving objects [8] is used to generate trajectories on the Oldenburg network. 600 to 3000 trajectories are generated for the time period from 0 to 100 and sampled at every time unit. To capture the real world time span between two consecutive time instances, for all the trajectories, the average distance between every two subsequent reported locations is calculated. The average distance is $234.96m,$ which is about 14 seconds travel time for a $60km/hour$ moving object. Thus, the actual time span between two consecutive time instances is about 14 seconds. The default time span for all the queries are 50 time instances.

To implement the grid–based solution, the anonymization grid is generated based on the minimum bounding rectangle (MBR) of the Oldenburg network. A randomly–chosen anonymization partitioning based on the grid is assigned to each generated trajectory .

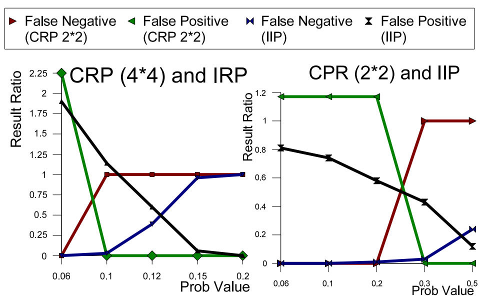
The default grid is a 40×40 partitioning on the MBR of the Oldenburg network. Based on the Oldenburg network data, the size of each grid cell is $589m \times 672.9m$. In the experiments, the grid partitioning is also tuned from 20×20 to 50×50 to observe the performance. The three policies are applied on the trajectories with the anonymization grid. To implement the CRP policy, two fixed partitionings are made, where each user has 2×2 or 4×4 grid cells. In the IRP policy, every user partition contains at most 4×4 grid cells. In the IIP policy, each moving object is set to use the anonymization partition (each partition contains at most 4×4 grid cells) that covers the start location of the moving object. After the object is out of this partition, it uses the lowest level of privacy so that each partition equals to a grid cell.

The experiments focus on evaluating the accuracy of the algorithms, i.e., the amount of false positives and false negatives. A false negative, is the error of not finding a pattern that does exist in the data. A false positive, is the error of finding a "pattern" that does not exist in the data. To compare the algorithms, the algorithms are also applied on an ideal case, where the partitioning of every user equals the anonymization grid, and use the results of this case as the evaluation target. Suppose the actual amount of dense grid cells is D, the number of false positives P and false negatives N are collected for every algorithm and the ratio between these values and D are reported, called the *false positive rate* (FPR) and *false negative rate* (FNR), respectively. The choice of these measures over the precision and recall measures used in information retrieval is because of conceptual simplicity. In the present case different kinds of errors are related to the same reference set (D), whereas in information retrieval the same set of correctly retrieved patterns are related to the set of all true patterns (recall) and to the set of retrieved patterns (precision). Hence, more accurate results are characterized by lower error rates rather than by higher recall and precision. However, it holds that Recall=1-FNR and Precision=(1-FNR)/(1-FNR+FPR).

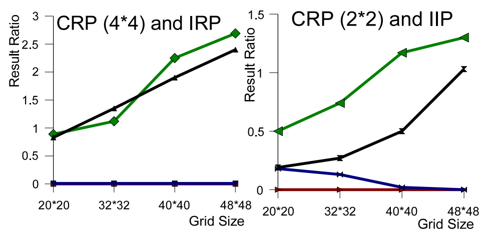
In the experiments, the *count* and *prob* values are tuned to observe the amount of false positives and false negatives. Experiments have also been conducted to test the effect of grid size, time span and amount of trajectories on the accuracy. As seen in Figure 8.7(a) to Figure 8.7(e), there are very few false negatives and the amount of false positives grows in certain cases. In particular, based on Figure 8.7(a), with the growth of *count* values, more false positives appear. With the experiment on the *prob* value (Figure 8.7(b)), it is possible to reach an optimal situation by tuning this *prob* value for each policy. For instance, the IRP policy has fewer false positives when *prob*$= 0.1$ and so has IIP when *prob*$= 0.3$. An observation from Figure 8.7(c) is that the amount of false positives grows with the grid size. The explanation for this is as follows. As the grid becomes denser, there are fewer really dense grid cells, but the amount of dense cells found through the three policies does not decrease very much, so the reported ratio value becomes larger. Figure 8.7(d) and Figure 8.7(e) show that



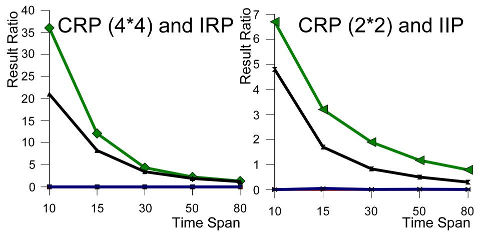
(a) Effect of Count.



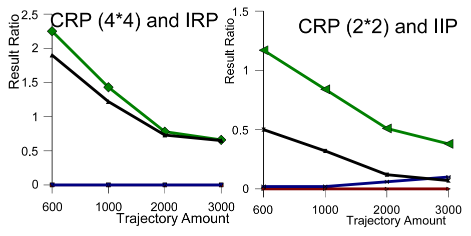
(b) Effect of Prob.



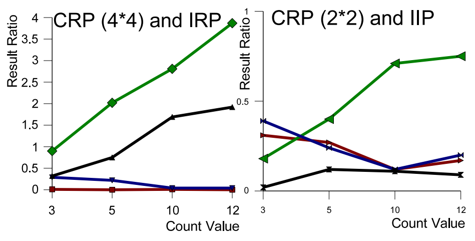
(c) Effect of Grid Size.



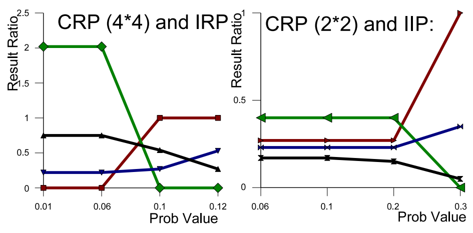
(d) Effect of Time Span.



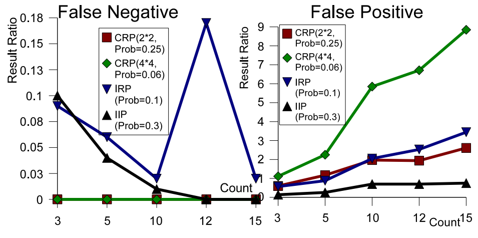
(e) Effect of Trajectory Amount.



(f) Effect of Count with Steepest Cut–Off.



(g) Effect of Prob with Steepest Slope Cut–Off Criteria.



(h) Optimized Results of Different Policies.

Figure 8.7: Experiments on Dense ST–area Query.

the increase of the time span and the amount of trajectories reduces the amount of false positives for all the policies.

To test how the steepest slope cut–off criteria influences the algorithms, the *count* and *prob* values are tuned to observe the amount of false positives and false negatives on the different policies. As illustrated in Figure 8.7(f) and Figure 8.7(g), the cut–off criteria decreases the amount of false positives but, compared to the same settings in Figure 8.7(a) and Figure 8.7(b), brings more false negatives. Thus, considering all the parameters for the three policies, the following are the **recommendation** settings for the dense ST–area query:

The IIP is the most effective policy for doing dense ST–area query with privacy protection. The second and third best choice is the CRP policy with 2×2 *partitioning and the IRP policy. For all the policies, certain optimal situation on the amount of false positives and false negatives can be reached by tuning the prob value. To increase the time span and amount of trajectories will improve the performance of all approaches.*

Based on the **recommendation**, an experiment is conducted to compare the different policies with their optimal settings. In this experiment, the amount of trajectories is increased to 1000 and optimal *prob* values are used for each policy. Figure 8.7(h) presents the results. The CRP policy with each partition containing 2×2 cells and the IIP policy shows the most promising performance. These two policies guarantee a precision level that makes them useful for most applications.

8.5 Conclusions and Future Work

Motivated by the possible loss of location privacy for LBS users, this paper proposed a general grid–based framework that allowed user location data to be anonymized. Thus, privacy is preserved, but interesting patterns could still be discovered. The framework allowed users to specify individual desired levels of privacy and developed three policies for implementing that. Privacy–preserving methods were proposed for a core data mining task, namely *finding dense spatio–temporal regions*. An extensive set of experiments evaluated the methods and showed that the framework still allowed most patterns to be found, even when privacy was preserved.

Future work will be along three paths. First, the *Multi–Grid* approach will be further investigated as it offers a direction for getting more detailed data mining results without violating the privacy. Second, in addition to the CRP, IRP and IIP policies, it is possible to develop more policies for creating anonymization rectangles suitable for different real world situations. Third, since the grid–based solution can be seen as a simple and general framework for privacy preserving data mining on moving object trajectories, hance the framework is planned to be extended to support more kinds of spatio–temporal data mining algorithms.

Chapter 9

Privacy–Preserving Trajectory Collection

Context awareness is one of the most important features of user–friendly Location–Based Services (LBS). To support context awareness in LBSes, real location data of mobile users has to be collected so that spatio–temporal patterns can be extracted by data mining methods. This brings a new conflict of interest: the data mining methods want precise location data, while the mobile users want to protect their privacy by not disclosing their exact movements (trajectories). To resolve the conflict, the paper first formally defines novel location privacy requirements. Then, it presents a system for privacy–preserving trajectory collection that meet these requirements. The system is composed of an untrusted server and clients communicating in a P2P network. Location data is anonymized in the system using data cloaking and data swapping techniques. The proposed system is empirically evaluated on realistic simulated movement data and is found to be effective under reasonable conditions and privacy/anonymity settings.

9.1 Introduction

The convergence among mobile services and positioning technologies paves the way for a range of new, innovative services, which are commonly referred to as Location–Based Services (LBSes). Emergence of LBSes creates a demand for novel data management technologies to efficiently support new data types, operations, and workloads. Context awareness, as one of the most important features of LBSes, is integrated in mobile devices so that these devices have information about the circumstances under which they operate and can react accordingly. To support context awareness in LBSes, real location data of mobile users has to be collected so that spatio–temporal patterns can be extracted by data mining methods. This brings a new conflict of interest: the data mining methods want precise location data, while the mobile users want to protect their privacy by not disclosing their exact movements. The following example describes the problem scenario.

A large shopping center wants to do data mining on the activities of customers in business hours. One type of customer activity is where and when customers were in the shopping center. To collect the activity information, it is assumed that a positioning system such as GPS, radiolocationing, or RFID detects the locations of the customers' mobile phones. Customers locations are recorded in the phones and sent to a central server. The central server then collects and analyzes the customers' spatio–temporal location data. However, it is a violation of privacy for *anyone*, including the server, to know exactly when and where an individual customer was. Thus, a solution that can collect the activity information from mobile phones without exposing the privacy of users is desirable.

Movements of mobile users are often modelled as trajectories in 2D space. Data mining on the trajectory data has many applications, not only in LBSes, but also in telematics and Intelligent Transportation System (ITS). Methods have been proposed to extract patterns, such as dense regions [47, 59] and frequent routes [32] from the trajectory data. An existing solution for protecting the location and trajectory data in LBSes is to anonymize the users' location data by decreasing the spatio–temporal resolution of the locations. After this "anonymization" step, the exact location data becomes a set of spatio–temporal rectangles. Although such ambiguity protects the mobile users' privacy, it also reduces the accuracy of the data mining results.

Motivated by these observations, this paper defines new privacy requirements for location/trajectory data and presents a system for collecting mobile users' trajectory data in a privacy–preserving manner. Compared to existing solutions, the proposed system does not require trusted components, yet it protects the privacy of mobile users and preserves the accuracy of data mining by keeping the spatio–temporal data intact. The proposed solution assumes that clients on mobile phones can communicate through a wireless P2P network. The process of trajectory collection is divided in five stages as follows. First, in the *client registration* stage, a group of k clients obtains permission and parameters from the server for executing the *trajectory sampling and anonymization–*, the *trajectory exchange–*, and the *data reporting* stages in a multi-threaded fashion. In the *trajectory sampling and anonymization* stage, clients record their private trajectories and generate a set of k "cloaking" trajectories to anonymize their actual trajectory. In the *trajectory exchange* stage, clients exchange sets of k partial trajectory pieces with other clients in the P2P network. Finally, in the *data reporting* stage, clients send anonymous partial trajectory pieces to the server. In the meanwhile, in the *data summarization* stage, the server continuously listens to data reports from clients, assembles the trajectory pieces and filters out the "cloaked" trajectories. An extensive experimental evaluation, based on realistic simulated trajectory data, shows that the proposed approach is effective under reasonable conditions and privacy/anonymity settings.

The contributions of this paper are believed to be as follows. First, the paper adapts and combines previous general privacy definitions to derive privacy definitions of various strengths for location data. More specifically, the paper defines *k–anonymity*, *α–diversity*, requiring spatial diversity, and *k–α–anonymity* for location data. Second, the paper proposes a complete system, including algorithms and implementation details, for the *lossless* collection of *exact* trajectories of moving objects. The proposed system does not require trusted components, yet it guarantees at least *k–anonymity* in all parts of the collection process. The proposed system is *scalable* as the process of anonymization is performed in a distributed fashion by clients interacting via a P2P network. The security of the proposed system is evaluated and the system is found to be robust against a wide variety of attacks by malicious clients or a malicious server (if it is hacked). In particular, the system is shown to guarantee the anonymity of the client data on both the client and the server side and any time during transmission in the air. Finally, the paper demonstrates through an extensive empirical evaluation that the proposed system is effective under reasonable conditions and privacy/anonymity settings.

The rest of this paper is organized as follows. Section 9.2 explores related work. Section 9.3 discusses the basic concepts relating to location privacy. Section 9.4 describes the proposed solution in detail, while Section 9.5 analyzes the privacy guarantee of the solution. Section 9.6 presents the empirical evaluation. Finally, Section 9.7 concludes and points out future directions for research.

9.2 Related Work

The topic of privacy preserving data mining [4] has appeared due to the concern of protecting privacy in the data collection and dissemination steps of data mining process. The database and data mining research communities have been very active in the discussion of privacy–preserving data mining. Important techniques include perturbation, condensation, and data hiding with conceptual reconstruction. The paper [96] presents a good review of these techniques. The techniques behind the proposed solution follow the spirit of two common strategies used for privacy–preserving data mining, namely data *cloaking* and data *swapping*.

Several papers [12, 43, 44, 74] have addressed the topic of privacy–preserving location–based services. These papers assume an architecture where a piece of trusted middleware, often termed an *anonymizer*, exists between mobile users and LBS servers. Based on the query requests from mobile users, the anonymizer constructs anonymity rectangles that include the locations of k users and sends a query to the LBS server with this rectangle as the location. The LBS server returns a superset of the results and final results are filtered by the anonymizer and sent back to each LBS user. Compared to these papers, the architecture proposed in this paper does not require a piece of trusted middleware. In fact, the anonymization process is done at the client side through the P2P network.

The proposed solution is focused on the data modification of the collected user data. In the existing papers, the data modification techniques used in the anonymizer are based on the ideas of aggregation and perturbation. In particular, papers [43, 44] extend the k–anonymity model [89] in relational databases to aggregate the locations of k mobile users so that the location of an individual user is not distinguishable. The adaptive location anonymizer discussed in [74] uses a grid–based pyramid structure to put the exact location of mobile users into cloaking rectangles made of grid cells. Paper [12] suggests a data model to augment uncertainty to location data, and propose imprecise queries that hide the location of the query issuer and yield probabilistic results. Compared to these existing techniques, the architecture in this paper does not use an anonymizer but distributes the anonymization step to client–side computation and communication. In previous work [35], a grid–based framework for privacy–preserving data collection is proposed. This framework protects the privacy of mobile users but also blurs the exact collected spatio–temporal locations which influences the accuracy of data mining results. To keep the quality of the collected data without violating the privacy, this paper proposes a novel data modification approach based on the idea of *swapping* [96]. Similar to the architecture described in [13], the present solution assumes the existence of a P2P network. As it will be shown later, the technique behind the present solution, compared to the aggregation and perturbation techniques, does not introduce any ambiguity into the location data, which ultimately guarantees the precision and quality of the data mining results.

9.3 Preliminaries

For the simplicity of the discussion, assume that the time domain $\mathbb{T}$ is totally ordered and use the non–negative numbers as the time domain. Denote each moving object as a *data item*. A data item is modelled as a two tuple $di = (id, S)$ where id is the identity attribute value of the data item and S is the trajectory of di. It is modelled as a sequence of tuples $S = \langle (loc_1, t_1), \ldots, (loc_n, t_n) \rangle$ where $loc_i \in \mathbb{R}^2$ ($i = 1, \ldots, n$) are locations, and $t_1 < t_2 < \ldots < t_n \in \mathbb{T}$ are (possibly irregularly spaced) temporally ordered time instances. Next, a *trajectory piece* is defined as a subset of a trajectory ordered on the time domain. For instance, $S_1 = \langle (loc_1, t_1), (loc_2, t_2), \ldots, (loc_k, t_k) \rangle$ and $S_2 = \langle (loc_{k+1}, t_{k+1}), (loc_{k+2}, t_{k+2}), \ldots, (loc_n, t_n) \rangle$ are two trajectory pieces of $S = \langle (loc_1, t_1), \ldots, (loc_n, t_n) \rangle$. For a given data item $di = (id, S)$, both $di_1 = (id, S_1)$ and $di_2 = (id, S_2)$ (with $di_1.id = di_2.id = di.id$) can represent part of the trajectory of di. di_1 and di_2 are called *partial data items*, or *items* for short.

A simple way of protecting trajectories of mobile users is to hide the identity of users in the data items. Specifically, given a data item (id, S), by hiding the identity (e.g., encoding the id value), it is impossible to deduce the mobile user from the id value. However, an adversary can still study if a given data item corresponds to a specific user by cross–referencing from other public information about the user (e.g., work address, home address, etc.). If a trajectory S covers many relevant addresses of a user, it is possible that the data item (id, S) describes the trajectory of this user.

As described in related works, a dominant technique for protecting the spatial and spatio–temporal locations of users is to keep each user's locations anonymous among a set of other users. This so–called *k–anonymity* technique is also applicable to protect the trajectories (and, trajectory pieces) of mobile users. The following defines the *k–anonymity of data items*.

Definition 9 (*k–anonymity of data items*) For a set of moving objects $\mathcal{MO} = \{o_1, \ldots, o_m\}$ and a set of data items $\mathcal{DI} = \{(id_1, S_1), \ldots, (id_n, S_n)\}, m \leq n$ where $id_1, \ldots, id_n$ are encoded identity values and $S_1, \ldots, S_n$ are trajectory pieces, the moving objects and the data items are said to preserve *k–anonymity* if both of the following conditions are true:

1. For any object o, there are at least k data items $(id'_1, S'_1), \ldots, (id'_k, S'_k)$ that correspond to this object with equal probability.
2. For any data item (id, S), there are at least k objects $o'_1, \ldots, o'_k$ that correspond to the data item with equal probability.

The *k–anonymity of data items* preserves the anonymity when an adversary matches trajectories to specific moving objects. However, in an extreme case, when the trajectories of moving objects are almost identical (e.g., cars moving on the same roads in the road network), the spatial and spatio–temporal privacy of these objects is still exposed as the possible locations of each object can be narrowed down to the common parts of the trajectories. To improve the protection of the trajectories of moving objects, as in the general privacy protection framework [69], it is necessary to require that these trajectories possess a certain spatial diversity. Hence, the definition of *α–diversity* is introduced in the following.

Definition 10 (*α–diversity of data items*) For a set of data items $\mathcal{DI} = \{(id_1, S_1), \ldots, (id_n, S_n)\}$ where $id_1 \neq id_2 \neq \ldots \neq id_n$, a given threshold α, these data items preserve *α–diversity* if $AREA(MBR(\{S_1, \ldots, S_n\})) \geq \alpha$, where *MBR* is the minimum bounding rectangle of $S_1, \ldots, S_n$ and *AREA* returns the area size of an MBR.

Next, combining the definitions of *k–anonymity* and *α–diversity,* a more strict setting to protect the privacy the data items is presented, namely the *k–α–anonymity*.

Definition 11 (*k–α–anonymity of data items*) For a set of moving objects $\mathcal{MO} = \{o_1, \ldots, o_m\}$ and a set of data items $\mathcal{DI} = \{(id_1, S_1), \ldots, (id_n, S_n)\}, k \leq m \leq n$ that preserve *k–anonymity,* and a threshold value α, the moving objects and data items satisfy *k–α–anonymity* if the following condition is true. For any k data items $(id'_1, S'_1), \ldots, (id'_k, S'_k) \in \mathcal{DI}$, these data items preserve *α–diversity*.

The *k–α–anonymity* is a more restrictive requirement to preserve the privacy of moving object trajectories. In particular, given an α value, it is not guaranteed that the *k–α–anonymity* can be achieved by the sets of *k–anonymous* moving objects and data items. For example, consider a group of k drivers living in the same suburb area and working in the city center. The set of k drivers and their corresponding set of trajectories is *k–anonymous*. However, since it is very likely that the trajectories of these drivers are quite alike, the MBR of these trajectories is likely to have an area that is smaller than any reasonable value for α. Hence, the set of drivers and their corresponding set of trajectories do not satisfy *k–α–anonymity* for any reasonable value of α. The diversity of spatial and spatio–temporal locations is decided by the mobile users.

The aim of the herein proposed data collection solution is to preserve anonymity of data items (or partial data items) in any set of data being stored, transmitted or collected in the system. As described later, the solution preserves *k–anonymity* in all sections of the system and achieves *α–diversity* or *k–α–anonymity* in the parts that are more vulnerable to privacy threats. In the following details of the proposed solution are described.

9.4 Solution

The proposed solution is for a scenario in which users have mobile phones equipped with a positioning device and these mobile phones can communicate with each other through a P2P network. Such a scenario is very reasonable in real–world applications that are fuelled by technical advances such as the development of smart phones with embedded GPS devices and the widespread availability of WiFi and Bluetooth, and the scalability and robustness of the P2P application design paradigm. Assuming that the solution aims to preserve *k–anonymity, α–diversity* and *k–α–anonymity* (with k and α as concrete values) in all aspects of trajectory data collection, the following subsections describe the architecture of the proposed system and its detailed privacy preserving procedures.

9.4.1 System Architecture

The proposed system involves a server side application installed at a central server and a client side application that is installed at each mobile phone. Each client has a profile that keeps its settings on the client side application. The client side application can connect to the central server through a mobile connection protocol such as SMS, MMS or GPRS. Each client side application is also able to communicate with other client applications through a wireless P2P network such as WiFi or Bluetooth. Each client side application is called a *client* and the server side application is called the *server*. The *server* includes two processes, *server_reg* and *server_sum*, which are kept running during the whole lifetime of the *server*. The *client*, once activated, starts a single process, namely *client_proc*. The top–level algorithm of *client_proc* is listed below.

```
(1)  procedure client_proc $(server, phone\_no, k, \alpha, \lambda, L)$
(2)    $(T_s, \tau, \tau_{max}) \leftarrow register(server, phone\_no, k)$
(3)    $id_1, id_2, \ldots, id_k \leftarrow$ gen_hashID$(phone\_no, k)$
(4)    while $now() \in [T_s, T_s + \tau)$
(5)      $DB \leftarrow$ sample_anonymize$(\{id_1, \ldots, id_k\}, k, \alpha, \lambda)$
(6)      exchange$(k, DB, TRUE)$
(7)      $t_{old} \leftarrow past\_time(DB)$
(8)      if $(size\_of(DB) \geq L) \cup (now() - t_{old} \geq \tau_{max})$
(9)        report$(server, DB)$
(10)   if $size\_of(DB) \leq k \times num\_of\_peers(k)$
(11)     exchange$(k, DB, FALSE)$
(12)   else : report$(server, DB)$
(13)   return $Q_{dp}$
```

Among the parameters of *client_proc*, $server$ and $phone\_no$ specify the server address and the client's phone number, and k, α, λ, L are kept in the client's profile. The values k, α specify the user's requirement on *k–α–anonymity*. λ and L are described in the following. In the pseudo code of *client_proc*, the variable *DB* is a database of all trajectory pieces in the current instance of *client_proc*.

With the three processes, *server_reg*, *server_sum* and *client_proc*, the whole system can be summarized into five stages, namely: 1) *client registration*, 2) *trajectory sampling and anonymization*, 3) *trajectory exchange*, 4) *data reporting*, and 5) *data summarization*. As depicted in Figure 9.1, at stage 1, a client A sends a registration request to the server (line 2 of *client_proc*). The server process (*server_reg*) accepts the request and assigns the time and period that the client should start and continue other steps of *client_proc*. Upon receiving the approval from the server, client A starts stage 2. At this stage, client A generates a set of k different ID values (line 3 of *client_proc*). One of these ID values is selected as the ID of the actual trajectory and

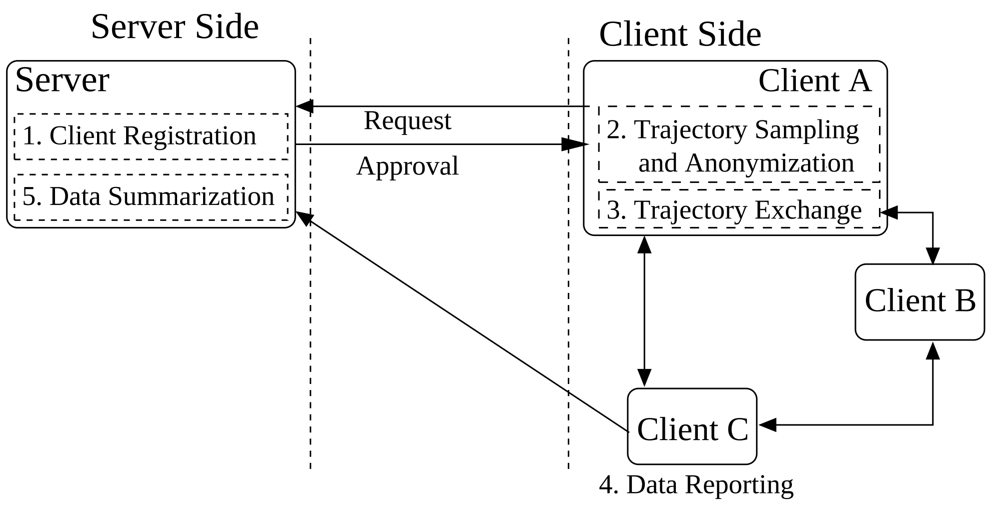


Figure 9.1: System Architecture.

the others are used as the ID values of synthetic trajectories (explained later). The synthetic trajectories with these ID values are called the *cloaking trajectory data*. Client A begins to record its trajectory data and generate the cloaking trajectory data (line 5 of *client_proc*). The parameter λ of *client_proc* specifies the time interval for cutting the collected trajectory and the cloaking data into trajectory pieces. The parameter C is used in the *trajectory sampling and anonymization* stage to specify how the synthetic trajectories should be generated. Then, at stage 3, through the P2P network, client A communicates with clients B and C to exchange the anonymized data (line 6 of *client_proc*). The *trajectory sampling and anonymization* and *trajectory exchange* steps are kept running until the period specified by the server is reached (the while–loop of *client_proc*). At stage 4, when a *client* has reached its storage limit (specified by the parameter L of the *client_proc*) or if it has a very "old" piece of trajectory data, this client reports its data to the server (line 9 of *client_proc*). The stage of *data reporting* also happens when clients are at the end of the activation period (lines 10–12 of *client_proc*). In Figure 9.1, clients A, B choose to transmit their collected data to other peers in the P2P network (line 11 of *client_proc*) and client C sends its the data to the server (line 12 of *client_proc*). Finally, at stage 5, the server process *server_sum* summarizes the data from the clients, filters out the synthetic trajectory data and computes the real trajectory data. The next subsections describes details of these stages.

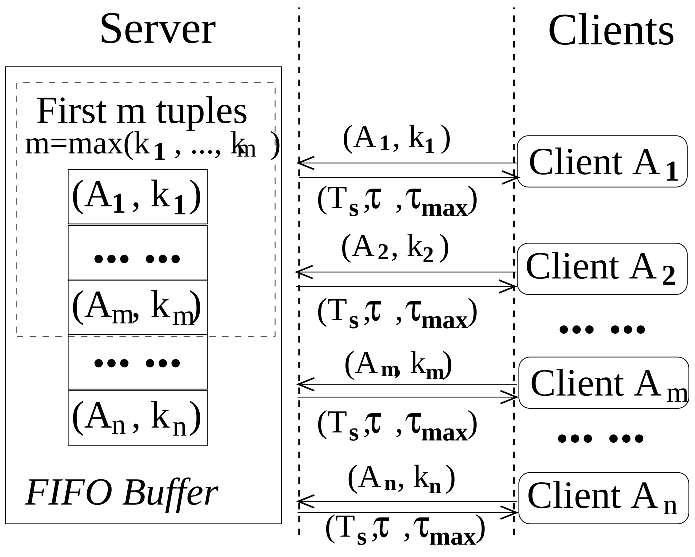


Figure 9.2: Client Registration.

9.4.2 Client Registration

Each client must start with the *client registration* step. At this step, the function *register*$(server, phone\_no, k)$ sends a registration request message to the *server*, together with its phone number and the k value.

At the server side, when the process *server_reg* receives a request message, it saves the request in a FIFO buffer. This process also maintains the maximal k value of all these requests, denoted as k_{max}. When the FIFO buffer has k_{max} elements, the *server_reg* process sends out messages to each of the first k_{max} registration requests, removes all elements from the buffer, and sets $k_{max} = 0$. As illustrated in Figure 9.2, each of the clients $A_1, \ldots, A_n$ sends a message with $phone\_no, k$. When the amount of tuples in the buffer is equal to k_{max}, i.e., $m = max(k_1, k_2, \ldots, k_m)$, the server sends approval messages to these m mobile phones. An example approval message can be T_s, τ, τ_{max} where T_s denotes the time that the client should continue with other steps of *client_proc*, τ describes how long each client should keep running these steps, and τ_{max} specifies a time period that will be used in the *client_proc* to start the *data reporting* stage. In reality, the value of τ reflects the average time period that a client is active, e.g., the average time span that customers spend at a shopping center. Without loss of generality τ is a static threshold value stored at the server. However, it is possible to implement τ as a dynamic value. For instance, each client can send the registration request message with an anticipated active time period. Then the server can organize k clients into a group if they have similar active time periods and use the maximal of these time periods as τ for clients of this group.

9.4.3 Trajectory Sampling and Anonymization

At this stage, a set of k different ID values are generated at the client. To ensure that the k values are unique among the other ID values generated at different instances of the clients, in the function *gen_hashID* (line 3 of *client_proc*), the Secure Hash Algorithm (SHA) [88] is used to generate the ID values. The input to the SHA function is a combination of the phone number $phone\_no$ and the current timestamp. If the input values are unique, based on the properties of the SHA functions, the generated hash values are also unique among all instances of the clients. Although recent research [99] has found an attack in which two separate messages could be found that deliver the same SHA–1 hash using 2^{69} operations, stronger versions of SHA, such as SHA–256 and SHA–512, can be used to further improve the security of the hash function.

Take client A in Figure 9.1 as an example. Suppose these k ID values generated at client A are $id_1, id_2, \ldots, id_k$. Client A randomly decides one of these k values as the ID value of the collected trajectory. In the *sample_anonymize* function started at the *trajectory sampling and anonymization* stage (line 5 of *client_proc*), without loss of generality, id_1 is used for the collected trajectory, denoted by S_1, and the other ID values are used for the synthetic trajectories $S_2, \ldots, S_k$. This function collects the trajectory of the client and adds synthetic trajectories for anonymizing the collected data. The pseudo code is listed in the following.

```
(1) procedure sample_anonymize($\{id_1, \ldots, id_k\}, k, \alpha, \lambda, C$)
(2)   $S_1 \leftarrow$sample_trajectory()
(3)   $S_2, \ldots, S_k \leftarrow$ gen_trajectory($k, \alpha, C$)
(4)   at every $\lambda$ interval:
(5)     for each $S_i, i = 1, \ldots, k$
(6)       $S'_i \leftarrow$get_piece($S_i, now() - \lambda, now()$)
(7)     $n_1 \leftarrow 2z; n_2, \ldots, n_k \leftarrow 2z + 1 : z \in \mathbb{Z}^+$
(8)     add $(id_1, S'_1), \ldots, (id_k, S'_k)$ to DB
(9)   return DB
```

In this algorithm, while the trajectory data S_1 is being sampled by the *sample_trajectory* function (line 2), client A also generates $n-1$ synthetic trajectories $S_2, \ldots, S_k$ (function *gen_trajectory* in line 3). The *sample_trajectory* function collects the actual locations at every unit time instance. Then, at every time interval λ, client A applies the *get_piece* function on the trajectories $S_1, \ldots, S_k$ and gets the trajectories pieces for time $[now() - \lambda, now())$. The interval λ is one or more time units and function $now()$ gives the current unit time instance.

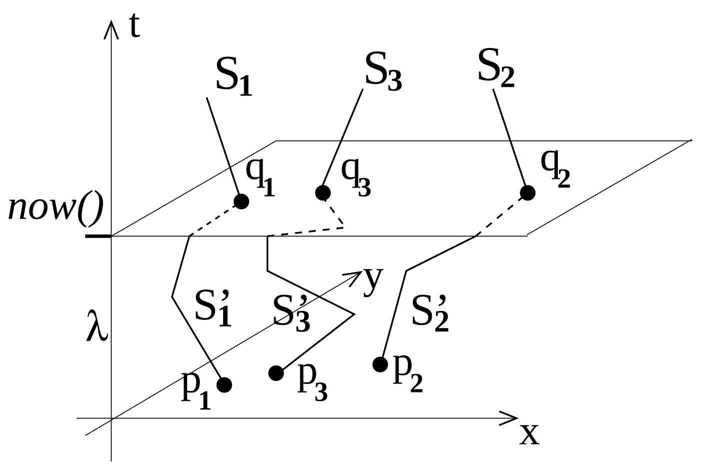
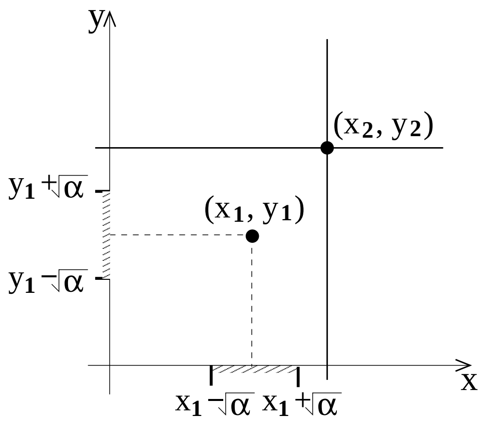


Figure 9.3: The *get_piece* Function.

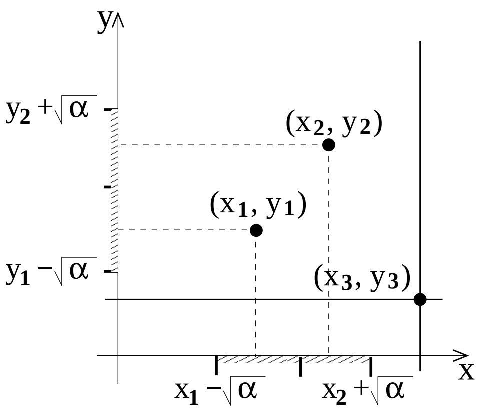
As illustrated in Figure 9.3, the *get_piece* function cuts the pieces during time interval $[now() - \lambda, now())$ on each of the three trajectories S_1, S_2, S_3. The trajectory pieces between (p_1, q_1), (p_2, q_2), (p_3, q_3) are denoted by S'_1, S'_2, S'_3. Tuples (id_1, S'_1), (id_2, S'_2), (id_3, S'_3) are added to *DB* (line 8). The tuple (id_1, S'_1), where id_1 is the ID value of the actual trajectory S_1 and S'_1 is obtained from S_1, is called the *real data item*. The other data items $(id_2, S'_2), \ldots, (id_k, S'_k)$, are called *cloaking data items*.

To distinguish the real data item among all the data items, the client generates an even number of copies of the real data item and an odd numbers of copies of the cloaking items. The idea of having copy amounts of the data items with different parity is to hide the real data items in exchanges between clients, but to be able to identify them at the server. Specifically, given an instance of k data items, one with an even and $k - 1$ with an odd number of copies, it is impossible for a client holding some, but not necessarily all, copies of a data item to decide whether the data item is real or cloaked. Accordingly, on line 7, one even (n_1) and $k - 1$ odd $(n_2, \ldots, n_k)$ copy amounts are calculated; each value n_i determines the number of copies of a data item (id_i, S'_i). Finally, copies of the data items $(id_1, S'_1), \ldots, (id_k, S'_k)$ are added to the trajectory database *DB* (line 8).

In the *gen_trajectory* function, synthetic trajectories $S_2, \ldots, S_k$ are generated in two steps, i.e., generating locations at the current time instance and generating locations at the next time instance. Suppose the collected location in S_1 at the first time instance is (x_1, y_1) in the 2D space. As illustrated in Figure 9.4(a), two line–sweeping processes are running at both the X and Y axes. Each process randomly chooses an X or Y value outside the interval $[x_1 - \sqrt{\alpha}, x_1 + \sqrt{\alpha}]$ on the X axis or $[y_1 - \sqrt{\alpha}, y_1 + \sqrt{\alpha}]$ on the Y axis. The intersection of the two sweeping lines,



(a) First Synthetic Location.



(b) Other First Locations.

Figure 9.4: Generating the First Locations.

(x_2, y_2) is the location of the first synthetic trajectory S_2 at the first time instance. First, locations of the other synthetic trajectories are generated in the same way. As illustrated in Figure 9.4(b), (x_3, y_3) is the first location of S_3 where x_3 is outside the interval $[x_1 - \sqrt{\alpha}, x_1 + \sqrt{\alpha}] \cup [x_2 - \sqrt{\alpha}, x_2 + \sqrt{\alpha}]$ on X axis and y_3 is outside the interval $[y_1 - \sqrt{\alpha}, y_1 + \sqrt{\alpha}] \cup [y_2 - \sqrt{\alpha}, y_2 + \sqrt{\alpha}]$ on y axis. Based on the description, the following Lemma 1 holds for the first set of generated locations.

Lemma 1 If (x_1, y_1) is the actual location and $(x_2, y_2), \ldots, (x_k, y_k)$ are generated in the described way then: $\forall (x_i, y_i), (x_j, y_j),\ i, j = 1, \ldots, k, i \neq j$: AREA$(\mathit{MBR}((x_i, y_i), (x_j, y_j))) > \alpha$.

The next locations of $S_2, \ldots, S_k$ are generated in the following way. When the next location (x'_1, y'_1) of the actual trajectory S_1 is sampled, two line sweeping processes on the X and Y axes are started to generate the next locations of $S_2, \ldots, S_k$. As illustrated in Figure 9.5, to generate the next location of S_2, the sweeping lines start from x_2 and y_2 on the X and Y axes and find values x'_2, y'_2 on the two axes that are outside the intervals $[x'_1 - \sqrt{\alpha}, x'_1 + \sqrt{\alpha}]$ and $[y'_1 - \sqrt{\alpha}, y'_1 + \sqrt{\alpha}]$. The selected x'_2 and y'_2 values are kept as close to x_2 and y_2 as possible, i.e., $|x'_2 - x_2| \leq \xi_x \times |x'_1 - x_1|, |y'_2 - y_2| \leq \xi_y \times |y'_1 - y_1|$. The values ξ_x, ξ_y are tuned incrementally to decide the x'_2 and y'_2 values. The rationale of this incremental process is to keep the speed vector of the generated trajectory as similar to the actual speed as possible. The next locations $(x'_3, y'_3), \ldots, (x'_k, y'_k)$ of $S_3, \ldots, S_k$ are generated in the same way. Similar to the first locations of the trajectories, the synthetic next locations hold for the following Lemma 2.

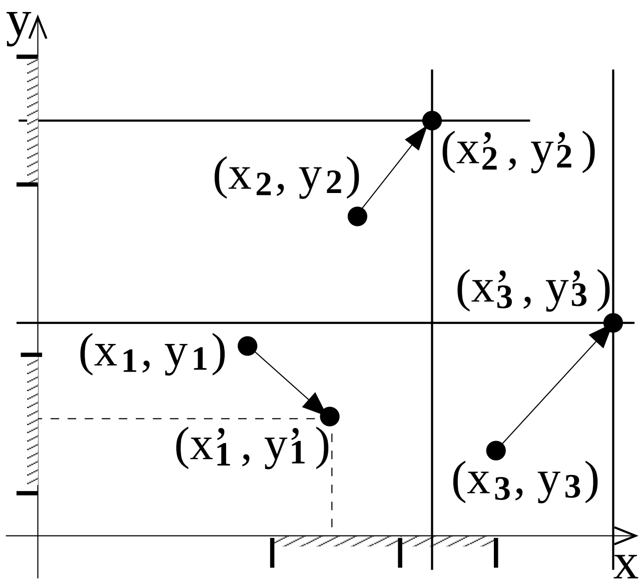


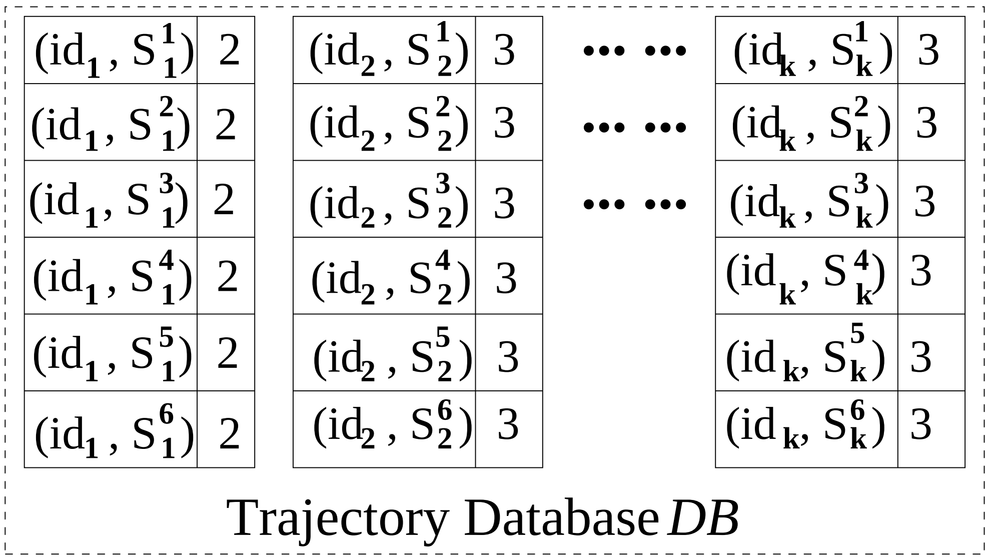
Figure 9.5: Generating the Next Locations of Synthetic Trajectories.

Lemma 2 Assume the second locations $(x'_1, y'_1), \ldots, (x'_k, y'_k)$ of each trajectory $S_1, \ldots, S_k$ where (x'_1, y'_1) is the actual location and $(x'_2, y'_2), \ldots, (x'_k, y'_k)$ are generated in the described way, then: $\forall (x'_i, y'_i), (x'_j, y'_j), i, j = 1, \ldots, k, i \neq j$ such that: AREA$(MBR((x'_i, y'_i), (x'_j, y'_j))) > \alpha$.

Following the same way, subsequent locations of the synthetic trajectories are generated. Based on Lemma 2, the area size of the MBR on every two locations of trajectories $S_1, \ldots, S_k$ at the same time instance is greater than α. With Lemmas 1 and 2, the area size of the MBR on all the trajectories $S_1, \ldots, S_k$ is greater than α. When the function *get_piece* executes on the trajectories $S_1, \ldots, S_k$, the collected trajectories pieces $S'_1, \ldots, S'_2$ satisfy the following lemma.

Lemma 3 Assume the data items $(id_1, S'_1), \ldots, (id_k, S'_k)$ are generated by function get_piece, then the area sizes for all of the MBRs of the trajectory pieces $S'_1, \ldots, S'_k$ is greater than α and for any two trajectory pieces $S'_i, S'_j, i, j = 1, \ldots, k, i \neq j$, the area size AREA$(MBR(S'_i, S'_j)) > \alpha$.

Based on Lemma 3, when the data items are added to the trajectory database *DB*, the MBR of all the trajectory pieces has an area size bigger than α. Since there are at least k data items in the *DB* and these data items have k different ID values, all the data items at *DB* preserve *k–anonymity* as well as *α–diversity* when a new group of data items are added to *DB*.



(id_1, S_1^1)	2	(id_2, S_2^1)	3	••• •••	(id_k, S_k^1)	3
(id_1, S_1^2)	2	(id_2, S_2^2)	3	••• •••	(id_k, S_k^2)	3
(id_1, S_1^3)	2	(id_2, S_2^3)	3	••• •••	(id_k, S_k^3)	3
(id_1, S_1^4)	2	(id_2, S_2^4)	3		(id_k, S_k^4)	3
(id_1, S_1^5)	2	(id_2, S_2^5)	3		(id_k, S_k^5)	3
(id_1, S_1^6)	2	(id_2, S_2^6)	3		(id_k, S_k^6)	3

Trajectory Database *DB*

Figure 9.6: Trajectory Database *DB* with Collected / Generated Items.

An example trajectory database *DB* is illustrated in Figure 9.6. Each tuple in *DB* records the ID value, the partial data item (trajectory piece), and the amount of copies of this item. Figure 9.6 shows the state of *DB* for a client that since the *client registration* step has, in six *trajectory sampling and anonymization* steps, added six sets of k data items (with 2 and 3 copies) to its *DB*. The most recently added data items are $(id_1, S_1^6), (id_2, S_2^6), \ldots, (id_k, S_k^6)$. Figure 9.6 shows only a logical view of *DB*, in reality, the items in *DB* are stored in two priority queues to support efficient trajectory exchange, (described in detail in Section 9.4.4).

9.4.4 Trajectory Exchange

When the trajectory sampling and anonymization step has been started and the trajectory database *DB* has more than k data items, the *client_proc* process begins the *trajectory exchange* step to exchange its trajectory data in *DB* with other peers in the P2P network. The *exchange* function is activated in two cases. The first is when a client is in the stage of exchanging data with other peers (line 6 of *client_proc*). In this case, the exchange is mutual, i.e., both ends send and receive data. The second is when a client has reached the end of the period τ specified by the server (line 2 of *client_proc*). As seen in lines 10–11 of *client_proc*, when at least k peers are available for accepting trajectory data (found through the function *num_of_peers*(k)), the client pushes all the trajectory data to these peers and does not accept any data sent from other peers. During the execution of *exchange*, the current client finds other clients running the same function and exchanges the trajectory data from its database *DB*. The parameter *state* of function *exchange* specifies whether the current client wants mutual data exchange ($state = TRUE$) or to only push data to other peers ($state = FALSE$). The pseudo code of *exchange* is listed bellow.

```
(1)  procedure exchange($k, DB, state$)
(2)    $p_1, \ldots, p_m \leftarrow$find_peers($k$)
(3)    for each $p_i, i = 1, \ldots, m$
(4)      $DB_{in}, DB_{out} \leftarrow \emptyset$
(5)      $t \leftarrow$get_state($p_i$)
(6)      if $t = TRUE$
(7)        $DB_{out} \leftarrow$pick_items($k, DB$)
(8)        DB$\leftarrow$DB$\setminus DB_{out}$
(9)        send($p_i, DB_{out}$)
(10)       if $state = TRUE$
(11)         get($p_i, DB_{in}$) // $DB_{in} : \{di'_1, \ldots, di'_{k_i}\}$
(12)     else: get($p_i, DB_{in}$)
(13)     DB$\leftarrow DB \cup DB_{in}$
(14)   return
```

The algorithm first searches for peers from the P2P network (line 2). Then, the client starts a while–loop to communicate with each of the peers (lines 3–14). When a peer p_i is connected, the algorithm first reads p_i's state (line 5), i.e., whether p_i wants mutual exchange or only wants to push its data to other peers. If p_i wants mutual exchange, the current client selects k items from *DB*, puts the items into a set DB_{out}, and uses function $send$ to send the data to peer p_i (lines 7–9). If the current client wants mutual exchange, the function get is called to accept data from p_i (sent by the $send$ function at p_i) to a local set DB_{in} (lines 10–12). If p_i only wants to push its data to the current client, the function get is also called to accept the data to DB_{in} (line 13). Finally, the data items in DB_{out} are removed from *DB* (line 8) and the fresh items from p_i are read into *DB* (line 14).

The function *find_peers*(k) finds nearby mobile clients in the P2P network that have at least k clients in their respective vicinities. To protect a client from sending too much data to a single and perhaps malicious peer, the function uses a small FIFO buffer that keeps a list of the recently–contacted peers and avoids sending too much data to these peers.

The function $pick\_items$ is used for picking k partial data items from the trajectory database. It works as follows. First, it picks two collected/generated items. Then, it picks $k - 2$ items that were received in previous exchanges. To ensure that there are always at least two collected/generated items to pick from, when the number of such items is very small, extra copies of them are generated without altering the

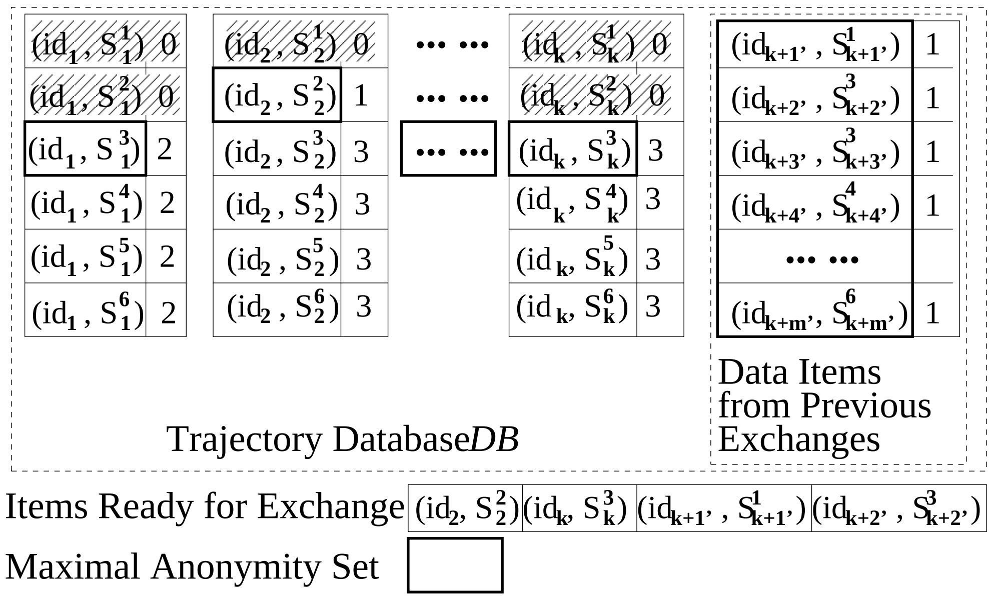


Figure 9.7: State of *DB* After Several Exchanges.

parities, i.e., always an even number of extra copies are generated. If there are not enough previously exchanged items to pick from, the function picks the remaining items from the collected/generated items. All of the items are picked in order of age and number of copies, i.e. the oldest item with the most number of copies first; ties are broken at random. All of the picked items have different ID values. When an item is picked, the corresponding copy amount of the item is reduced in the trajectory database *DB*. For each item in *DB*, if the copy amount is reduced to 0, the item is erased from *DB*.

A concrete example of the item selection is shown in Figure 9.7. Since the six initial *trajectory sampling and anonymization* steps, the client has performed several exchange steps in which it received m data items with IDs $id_{k+1'}, \ldots, id_{k+m'}$, and sent all copies of some of its collected/generated items, f.ex., (id_1, S_1^1), shown in shaded cells. According to "the oldest with the most copies" selection criteria, the $pick\_items$ function picks two collected/generated items $(id_2, S_2^2), (id_k, S_k^3)$ and two previously exchanged items $(id_{k+1'}, S_{k+1'}^1), (id_{k+2'}, S_{k+2'}^3)$ for exchange. The copy amounts of the selected items are only updated after the items have been successfully exchanged.

The above selection can be easily accommodated by storing the collected/generated and previously exchanged items in two priority queues PQ_{cg} and PQ_{ex}, respectively. Items are prioritized according to their age and copy amount (oldest with most copies first) in their respective queues. The queues support the following four operations: *insert*, *delete*, *increase*/*decrease* the priority of an element, and *peek* at the top k elements. Using a Fibonacci heap implementation, most of the operation can be supported in amortized constant time (deletions are still O(log n)) [16]. Using the queue operations, the $pick\_items$ function can be implemented as follows. *Peek* at the top 2 elements of PQ_{cg}, and the top $k-2$ elements of PQ_{ex}; consider the items in these elements for exchange. After a successful exchange of the items, depending on the copy amount of the exchanged items prior to the exchange, either *decrease* the priority of an element based on the updated copy amount, or *delete* the item from the queue if the last copy has been exchanged.

A number of aspects of the selection process performed by the $pick\_items$ function is important to emphasize. First, the items picked by the $pick\_items$ function have k different ID values and the MBR of at least two items has an area size larger than α. Hence, the items picked for exchange satisfy *k–α–anonymity.* Second, the selected collected/generated items are the oldest amongst all such items and correspond to trajectory pieces from the past. Hence, the trajectory pieces described by these items are spatially disconnected from the location where the exchange is taking place, which prevents a malicious client from making inferences based on such a connection. Third, the selection process ensures that the oldest items are exchanged first and that the distribution of the number of remaining items for different IDs is as close to uniform as possible. These latter aspects are important for reaching / maintaining anonymity state, which is defined next.

When the *exchange* process has been started for a while, and the client has received data items for $m \gg k$ ID values from other clients, identification of the actual trajectory of the client becomes less trivial. Nonetheless, if the entire *DB* was reported to the server, the server, based on the number of data items for each ID, could with high probability associate a set of k IDs with the client. Later, during the *data summarization* stage, the server could identify the actual trajectory of the client by filtering out synthetic trajectories. To study when a part of *DB* is *k–anonymous* towards the server, the *anonymity state* of a subset of data items is defined as follows.

Definition 12 (*Anonymity state*) A given subset S of the data at a client c is in *anonymity state* if there are at least $k-1$ IDs from other clients and the distribution of the number of partial data items for the different IDs is statistically equal to the uniform distribution according to the Kolmogorov–Smirnov One–Sample test for a given critical value α_{KS}[1].

[1]The Kolmogorov–Smirnov One–Sample (KS–1) test is a goodness–of–fit test between an empirical distribution e and a known distribution u [71]. According to the KS–1 test the null hypothesis, e not being significantly different from u, can be rejected at a confidence level $p = 0.05$ if the maximum absolute difference between the cumulative distribution of e and u is larger than a critical value $\alpha_{KS} = 1.36/\sqrt{n}$ for a sample size of $n > 35$.

The subset S in practice contains collected/generated items ranging from the oldest until the most recent collected/generated item that can be included while still maintaining anonymity state. The exchanged items in the subset S span a time period that includes the time period that is spanned by the collected/generated items in the subset. The largest possible subset of the data in anonymity state is called the *maximal anonymity set*. In Figure 9.7, items in cells with think borders comprise the maximal anonymity set .

Given the way items are picked for exchange, the way FIFO buffers are employed at each client to avoid sending data too many times to the same client, and the likelihood of re–encountering the same client, the likelihood of receiving several data items for the same ID is very low. Hence, for practical purposes, the entire trajectory database *DB* of client c is in anonymity state if it contains at least $k - 1$ items from other clients, and it contains on average at most one data item for each of c's IDs. To achieve anonymity state, a client has to exchange most of its collected/generated data items. There are $n_i \in \{2z, 2z + 1\} : z \in \mathbb{Z}^+$ copies of k data items collected/generated per sampling (λ) period. Since in one exchange process, with the exception of the initial conditions, two of these items are selected for exchange, $\lceil k \times n_i/2 \rceil$ exchanges are necessary to send all of the client's own data to peers. Assume that the client's entire *DB* was in anonymity state at time t_a, and no data exchange occurred up to time $t_{now} = t_a + n * \lambda$. Then, the client needs to participate in at least $\lceil n \times k \times n_i/2 \rceil$ exchanges during the period $[t_a + n \times \lambda, t_a + (n+1) \times \lambda)$ for its entire *DB* to be in anonymity state again. Assuming the number of possible exchanges a client can perform is uniform over time, a client is said to *maintain* anonymity state if it participates, on average, in at least $\lceil k \times n_i/2 \rceil$ exchanges per sampling period. In a concrete example, if the client collects/generates 2 and 3 copies of $k = 5$ items per sampling period, the client maintains anonymity state if it participates, on average, in at least $\lceil 5 \times 3/2 \rceil = 8$ exchanges per sampling period.

Based on the above discussion, anonymity state may not be achieved if the client does not have a good amount of data exchange with peers. The empirical study in Section 9.6 investigates under what conditions and privacy settings the amount of data exchanges performed by clients is sufficient for clients to reach and maintain anonymity state.

9.4.5 Data Reporting

The *data reporting* step is activated in two cases. First, when the *client_proc* process is in the period τ, and the size of the trajectory database *DB* is close to the storage limit or the *DB* has partial data items that are too old (line 8 of *client_proc*). Second, when a client has reached the end of the period τ, in which case the client can either send out its data via exchange processes to other peers in the P2P network or continue with the *data reporting* step.

The *report* function is started at the *data reporting* step. If the *report* function is invoked in the first case, it sends all of the data items obtained from other peers to the server and keeps the other data items. If the *report* function is invoked in the second case, it sends the maximal anonymity set to the server and erases rest of the (collected/generated) data items at the client.

By erasing some of the most recently collected/generated data items, the recent pieces of a trajectory are lost. To reduce the extent of this loss before it occurs, the client can dynamically, depending on the current time and the end of the period τ, in the *trajectory exchange* step increase the number of collected/generated data items in the set of k items that are selected for exchange. The empirical evaluations in Section 9.6 show that, even without clients performing the above outlined loss reduction strategy, the loss is negligible. Ultimately, the loss can also be entirely eliminated, if the client at the end of the period τ saves the remaining collected/generated items and restarts the $client\_proc$ at the *clients registration* stage. Since the saved items are the oldest items amongst the newly collected/generated data items, they will be selected first by the $pick\_items$ function in future exchanges. Hence, the probability that the client re–reports items for one or more of its previous IDs in the next *data reporting* stage, thereby sacrificing its privacy towards the server, is *very* low. To completely eliminate the above probability the client can optionally flag the saved items and ensure that no flagged items are reported to the server. This flagging technique effectively trades a slim probability for the loss of privacy against a negligible loss in the trajectory.

In the very unlikely event when a client c is not able to perform a single exchange within the τ period, and hence has an empty maximal anonymity set, the entire trajectory of a client is erased and not reported to the server. Even in this case the *k–anonymity* of the other $k-1$ clients started at the same time as c will not be changed. Specifically, if there is only one client that reports data and the other $k-1$ clients erase their collected/generated data, the server still does not know who this trajectory refers to among the k clients.

9.4.6 Data Summarization

The *data summarization* step starts the *server_sum* process. This process keeps running to collect data from clients and summarizes the data into trajectories. Specifically, when the server receives data from clients, the *server_sum* process maintains a *temporary table* that arranges the received partial data items into groups based on the ID values of the items. As illustrated in Figure 9.8, the *temporary table* puts all partial data items with the same ID values under the same row .

The *server_sum* process also records the timestamps when the first data item of each column is received ($t_1, t_2, t_3, \ldots$ in Figure 9.8. These timestamps are used to determine when the items under each ID value have been fully received. Specifically,

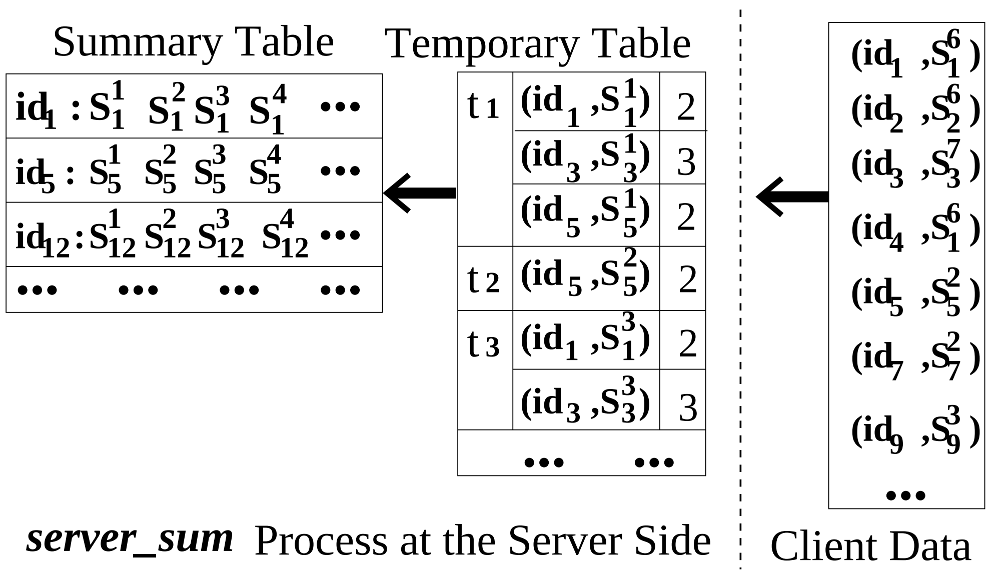


Figure 9.8: Data Summarization.

suppose the first data item of a column in the summarization table is received at time t_1. Then the *server_sum* processes waits for a $2\tau_{max}$ time period to wait for the rest of the data items of the same ID value to be sent from the clients. Since data items older than τ_{max} at every client are required to be sent to the server, all the data items of the column should have been received after a $2\tau_{max}$ time period since t_1. After this period, the *server_sum* process checks the amounts of copies of the data items in each group. If the parities of the amounts in a group are odd, the group is removed as it represents cloaking data items. If the parities in the group are even, the trajectory pieces in this group are merged into a whole trajectory based on the timestamps of the trajectory pieces. Finally, as illustrated in Figure 9.8, the actual trajectories are recorded in a *summary table* and saved at the server database.

It is possible that not all copies of a data item are reported to the server. For example the missing copies could be part of the loss described in Section 9.4.5. The copies could also be missing because a malicious client deliberately deleted or altered the copies. In these cases, checking the parity of the amount of copies of a single data item alone would be misleading. Hence, instead of performing parity checks of copy amounts of individual data items, a check for the majority parity of copy amounts within a group is performed.

9.4.7 Neighborhood Detection in P2P Networks

The function *find_peers*(k), on line 2 of the *exchange* function, finds nearby mobile clients in the P2P network that have at least k clients in their respective vicinities. It is executed in order to ensure that exchanges satisfy the second condition of the *k–anonymity* definition (Definition 9). In a trusted P2P environment, where clients can be trusted, the *find_peers*(k)–query can be answered in a straight forward fashion, i.e., the client asks its neighbors about their neighbors and decides based on the answers. In an untrusted P2P environment, where potentially malicious clients might try to lie about their neighbors, answers from individual peers can still be verified based on the totality of the answers as follows. First, the acquiring client c_i asks its neighbors $C_n = \{c_l, \ldots, c_m\}$ for their neighbors' IDs $I_n = \{I_l, \ldots, I_m\}$. Then, client c_i summarizes the ID sets, I_n, and retains the IDs, I^v, that are present in more than min_verif_cnt number of ID sets. Assuming that there are at most $min\_verif\_cnt - 1$ malicious and *cooperating* clients among the clients C_n, the *verifiable* IDs of an ID set $I_j \in I_n$ are $I_j \cap I^v$. Consequently, a client $c_j \in C_n$ is part of the answer set to the *verified find_peers*(k)–query if $|I_j \cap I^v| \geq k$. Naturally, if there is very little overlap between the *true* ID sets, i.e., the size of I^v is very small, then this verification method will reject answers from non–malicious clients and thereby limit the number of possible exchanges. However, the fraction of the *true* answers that are rejected by the proposed verification method turns out to be very small in practice, as is shown by the experiments in Section 9.6.

9.5 Discussion

The proposed solution aims to collect trajectory data while keeping each mobile user's trajectory anonymous among other trajectories. The anonymity settings are kept in the privacy profile of each client. Due to the big variety of privacy threats, the possible privacy risks are generalized into three categories: privacy risk at the server side, the client side, and in the air. In the following each of these risks is discussed.

9.5.1 Privacy at the Server

In the server side application, in accordance with each set of k_{max} mobile users (k_{max} is the maximal k value sent by all these users) that start data collection at the same time, there will be the same or less amount of trajectories returned by the *data summarization* step. Since the ID values are generated at the client side, it is impossible to link a moving object with a specific trajectory. Thus, the requirement of *k–anonymity* of the clients is preserved at the server side.

9.5.2 Privacy at Clients

Each client, after starting the *trajectory sampling and anonymization* and the *trajectory exchange* steps, has at least k (k is defined by its own privacy profile) groups of data items with different ID values. After a client has achieved *anonymity state*, the data items at this client satisfy *k–anonymity*. Based on Lemma 3, the area size of the MBR of all the data items in *DB* can always be kept over α. Thus, the data items at the client also satisfy *α–diversity*.

9.5.3 Privacy in the Air

Data is transmitted in the air in two cases. The first is when the data is exchanged between peers in the P2P network. As described in the *trajectory exchange* step, each client sends k data items with different ID values and the area size is kept over α. Thus, the data transmitted between peers preserves *k–α–anonymity*. The second case happens when a client reports the data to the server. Based on the description of Section 9.4.5, this client sends either only the exchanged data from other peers or the maximal anonymity set to the server. Thus, the data in the air satisfies *k–anonymity* and *α–diversity* in this case.

The proposed solution for trajectory collection preserves the *k–anonymity* in all aspects of the data collection process. The solution does not change the actual trajectory data, but uses *cloaking data items* to preserve the anonymity, the accuracy of the original data is guaranteed. Since the server can have more safety protections such as firewalls and many other softwares to safeguard the data, the clients and the data being transmitted are more fragile under various forms of privacy threats. The proposed solution have further anonymity protection, such as *α–diversity* and *k–α–anonymity*, at the clients and the data in the air. To summarize, the proposed solution preserves the accuracy of the collected data and achieves anonymity requirements by taking the advantage of the P2P network to swap and anonymize the data at the clients.

9.6 Empirical Evaluation

The effectiveness of the proposed privacy preserving trajectory collection method was empirically evaluated for a simulated shopping mall environment. A grid–based shopping movement simulator was developed to generate realistic movements of clients in a 177,200m^2 large shopping mall with 16 shops. In the simulation, clients move from a 5$\times$5–meter grid cell to a neighboring grid cell at a speed of 1 m/sec. A simulation step is 5 seconds long, i.e., a client can only move at most one grid cell in one simulation step. The movements of clients are random, but obey the following rules:

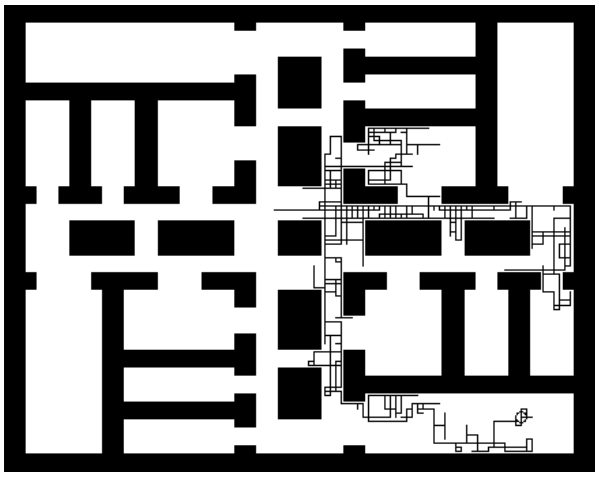


Figure 9.9: Layout of the Shopping Mall with a Sample Shopping Trajectory.

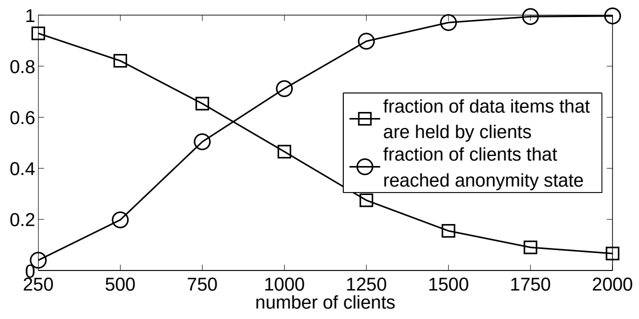
1. A client never crosses a wall.
2. A client at each step picks a random direction obeying rule 1) and preferring a direction that was the same as its previous direction of movement. This preference is defined by a $heading\_factor$ parameter which for higher values makes clients move more in a straight line. This parameter is different for the corridor environment and the shop environment, in order to simulate transition like movement between shops and browsing like movement within shops.
3. A client never visits a shop twice.
4. A client becomes idle with probability $browsing\_prob$ inside shops, this essentially slows down the average speed of the client inside a shop.
5. A client, obeying rule 3), is drawn with a certain strength into a shop as it passes the door of the shop, the strength of this drawing force is controlled by a parameter $entrance\_factor$.

The layout of the shopping mall used in the simulations and a two hour long sample trajectory of a simulated client is shown in Figure 9.9. In the example shown and in all of the simulations the $heading_factor$ was twice as large in the corridor environment than in the shop environment, and the $browsing_prob$ parameter was set to 0.25, resulting in an average speed during browsing that is $1/4$ of the normal walking speed.

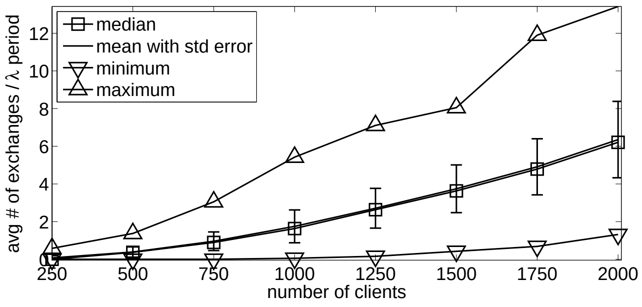
The effectiveness of the proposed method was evaluated using four measures: 1) the fraction of clients that reached anonymity state at some point during the simulation period, 2) the fraction of collected/generated data items that are held by clients responsible for collection/generating them, 3) the number of exchanges performed by clients per sampling (λ) period, and 4) the ages of the oldest collected/generated data items that are held by clients responsible for collection/generating them. Three groups of 20–minute long simulations were performed varying the number of clients m, the anonymity parameter k, and the communication range r in the P2P network. The sampling period in all of the experiments was fixed at $\lambda = 60$ seconds, during which each client collected/generated $n \in \{2, 3\}$ copies of k data items. During an exchange a client, if possible, sent 2 of its oldest collected/generated data items and $k - 2$ of its previously received data items. In all the experiments, each client's FIFO buffer for recording recently–contacted peers was 10 elements long. The results of the three groups of experiments are shown in Figures 9.10, 9.11, and 9.12, respectively.

Figures 9.10(a)–9.10(c) show the four effectiveness measures for anonymity parameter $k = 5$ and communication range $r = 10$ meters, for varying number of clients. As the number of clients increases, i.e., the spatial density of clients increases, the average number of exchanges an average client can perform during a sampling (λ) period also increases, see Figure 9.10(b). Given the particular settings for k and n, the average number of exchanges a client needs to perform per sampling period to reach (maintain) anonymity state is about 7.5 (5), which is achieved by most clients in the case of 2,000 clients. Under these conditions, most of the clients successfully have managed to exchange most of their old collected/generated data items and only hold own data items that have been collected/generated in the last few sampling periods, see Figure 9.10(c). The same results, viewed at a system level, are shown in Figure 9.10(a), where it can be seen that as the number of clients, i.e., the spatial density of the clients increases, the fraction of clients that reach, and with a sufficiently high likelihood, maintain anonymity state during the simulation period also increases. Similarly, due to the increasing number of exchanges, an increasing fraction of the total amount of colected/generated data items have been passed onto other clients and consequently a decreasing fraction of the data items is present in the system in a less anonymized state, held by the producing client.

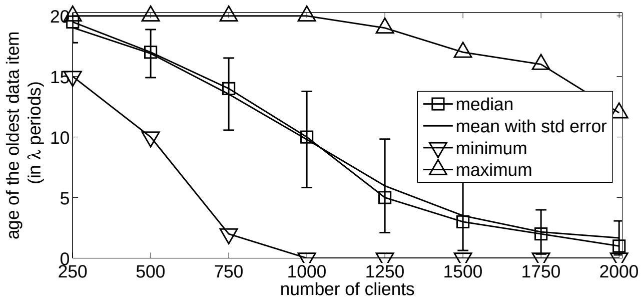
Figures 9.11(a)–9.11(c) show the four effectiveness measures for anonymity parameter $k = 5$ and $m = 1,000$ clients for varying communication range r. The results are similar to the previous results. That is, as the communication range is increased, the number of exchanges increases (Figure 9.11(b)), the ages of the oldest data items held by the producing clients decreases (Figure 9.11(c)), the fraction of the clients that reach (maintain) anonymity state increases, and the fraction of the total data items that is in a less anonymized state decreases (Figure 9.11(a)). It is important to note that the effects of the communication range parameter are more pronounced due to the quadratic relationship between the communication range and the number of exchanges (Figure 9.11(b)).



(a) Anonymity State and Data Dissemination.

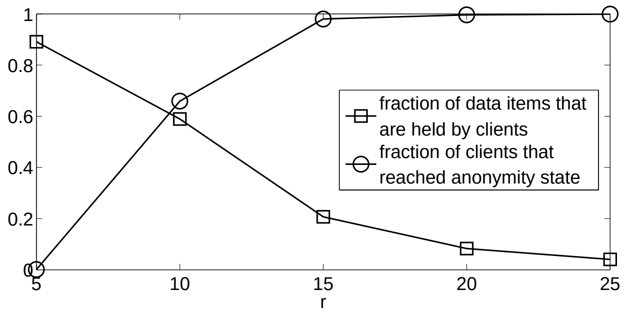


(b) Number of Exchanges per Sampling Period.

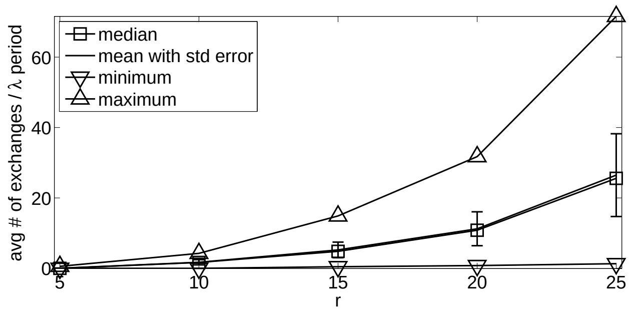


(c) Age of Oldest Data Item.

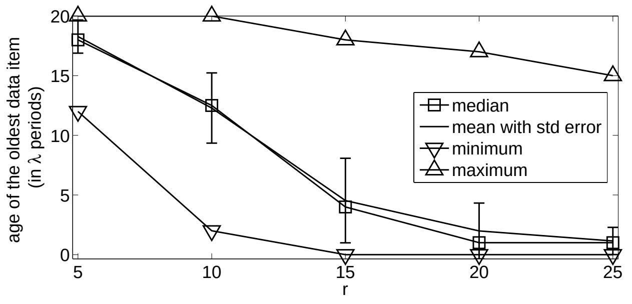
Figure 9.10: Efficiency Evaluation for Varying Number of Clients.



(a) Anonymity State and Data Dissemination.

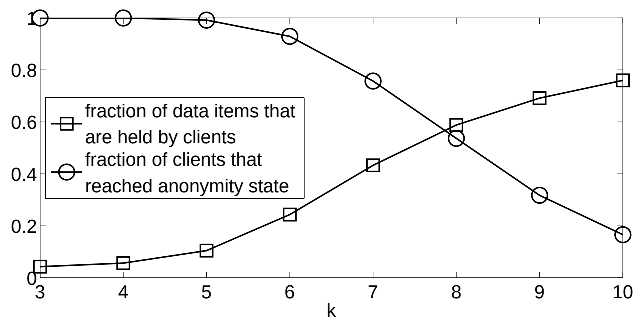


(b) Number of Exchanges per Sampling Period.

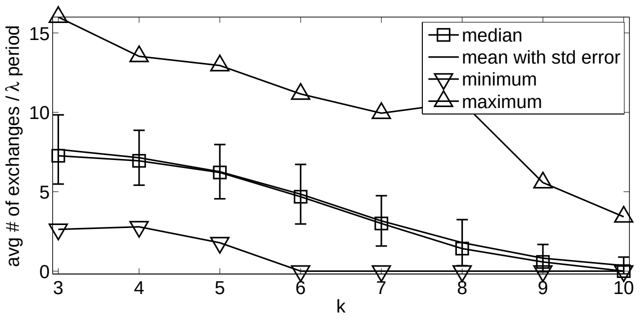


(c) Age of Oldest Data Item.

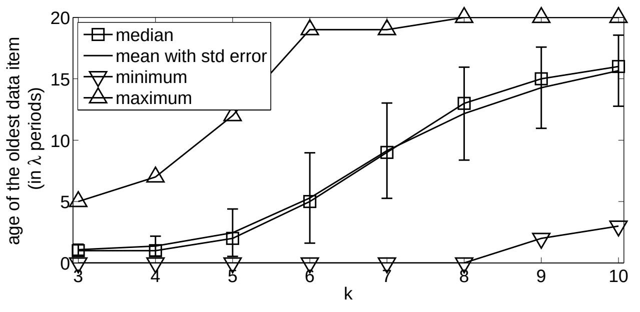
Figure 9.11: Efficiency Evaluation for Varying Communication Range r.



(a) Anonymity State and Data Dissemination.



(b) Number of Exchanges per Sampling Period.



(c) Age of Oldest Data Item.

Figure 9.12: Efficiency Evaluation for Varying k–Anonymity.

Figures 9.12(a)–9.12(c) show the four effectiveness measures for communication range $r = 5$ meters and $m = 2,000$ clients for varying anonymity parameter values r. Increasing values for anonymity parameter value k results in decreasing number of possibilities for exchange and increasing amount of colected/generated data items to be exchanged. Hence the total effect of these changes affect the effectiveness measures in an inverse manner to the effects of the parameters m and r.

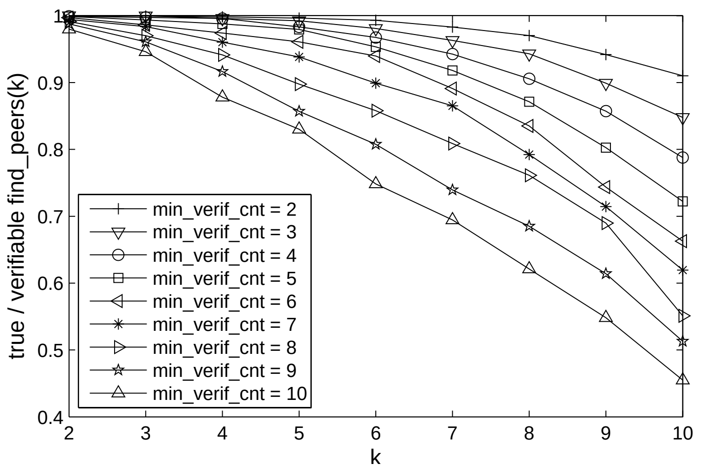


Figure 9.13: Fraction of $\mathit{find\_peers}(k)$ Answers that are Verifiable for Varying Values of k and $\mathit{min\_verif\_cnt}$.

In all of the so far presented experimental results, the $\mathit{find\_peers}(k)$–queries were executed assuming that there are no malicious clients. As proposed in Section 9.4.7, with a rather simple method the answers to the $\mathit{find\_peers}(k)$–queries can easily be verified by the clients even in case of multiple and potentially cooperating malicious clients. To determine the effectiveness of the proposed verification method, 500 uniform random points were generated in the unit square, representing locations of clients, and the true versus verifiable answer sets to $\mathit{find\_peers}(k)$–queries have been compared for all the points for varying k and $\mathit{min\_verif\_cnt}$ parameters and a fixed communication range of $r = 0.1$. To eliminate the effects of randomness, the measurements are based on 100 independent test runs. The results of these experiments are shown in Figure 9.13. The trends in the results are as expected, i.e., as the values of the k and $\mathit{min\_verif\_cnt}$ parameters increase the fraction of answers that are verifiable using the proposed method decreases. However, it is interesting to see that for rather high values of k and $\mathit{min\_verif\_cnt}$ the fraction is relatively high. In particular, assuming there is at most one malicious client near the client issuing the $\mathit{find\_peers}(k)$–query, the client can verify over 90% of the answers even in the case of large k parameter values. Consequently, using the proposed verification method, even in an untrusted environment clients can perform most of the exchanges and hence reach and maintain anonymity state.

In summary, the experimental results show that the proposed privacy preserving trajectory collection method is effective, i.e., under reasonable conditions (m, r) and for anonymity parameter values (k), clients are able to perform exchanges frequently enough, so that within a short period of time (1 - 5 minutes) most clients are in anonymity state and only a small fraction of the data items is in a less anonymized state, held by the producing client.

9.7 Conclusions and Future Work

The paper considered the problem of collecting trajectories of moving objects in a privacy–preserving manner. As a premiss for studying the problem, the paper first adapted and combined previously proposed general data privacy definitions to derive definitions for location privacy. The derived location privacy definitions in increasing strength are: *k–anonymity*, *α–diversity* (requiring spatial diversity), and *k–α–anonymity* of location data. To solve the problem, the paper proposed a complete system for the *lossless*, privacy–preserving collection of *exact* trajectories of moving objects. The proposed system is based on a client–server architecture and the collection process can be summarized into five stages, namely: 1) *client registration*, 2) *trajectory sampling and anonymization*, 3) *trajectory exchange*, 4) *data reporting*, and 5) *data summarization*. The proposed system does not require trusted components, yet it guarantees at least *k–anonymity* in all stages of the collection process. The proposed system is *scalable* as the process of anonymization is performed in a distributed fashion by clients interacting via an assumed wireless P2P network. Finally, through an extensive empirical evaluation, the paper demonstrated that the proposed system is effective under reasonable conditions and privacy/anonymity settings.

Future work will be along four paths. First, different kinds of system architectures will be considered. For example, as is common in wireless P2P network research [100], the addition of hotspots to the P2P network will be considered. These hotspots will further reduce the chance of isolated clients and could potentially be used to distribute the *client registration* and *data summarization* stages. Second, future work will consider the implementation of a full system and the large–scale real–world deployment of the system. Third, satisfying the *k–anonymity* of exchanges even in the presence of malicious clients is essential to the proposed system. Initial experiments indicate that the proposed method to verify $find\_peers(k)$–query results in a P2P network is effective. Nonetheless, it is only one possible solution to answer the query in a P2P network with malicious clients. Future work will consider other methods that guarantee even higher accuracy or security. Finally, a theoretical line of work will consider 1) proving the robustness of the system, and 2) finding a theoretical model of the system performance in relation to privacy protection.

Chapter 10

Summary

Several hardware trends in mobile technology, in particular the increasing availability and accuracy of mobile position technologies, pave the road for LBSes. Innovative LBSes integrate knowledge about the mobile user into the service. Much knowledge can be gained about users by analyzing the location data of users. To this extent, this thesis 1) devised effective spatio–temporal data mining methods for the desired analysis, 2) demonstrated the usefulness of the spatio–temporal data mining methods in promising LBS examples, and 3) devised privacy–preserving systems for trajectory collection and analysis.

10.1 Summary of Conclusions

The conclusions from Chapters 2 to 9 are the following. Chapter 2 proposed pivoting as a general methodology to extend a popular data mining method, namely rule mining. By considering a number of different types of data sources, the chapter derived a taxonomy of spatio–temporal data and investigated the types of knowledge that can be extracted using the extended spatio–temporal rule mining method.

Chapter 4 used pivoting to extend a frequent itemset mining method to find long sharable routes in trajectories. Considering different modelling options for trajectories led to the development of two variants of the method that can analyze large amounts of trajectories. High–level SQL–based implementations are described, and extensive experiments on both real–life- and large–scale synthetic data show the effectiveness of the method and its variants. The knowledge that the method and its variants can discover are believed to be useful for traffic planning and optimization and LBSes in the transportation domain.

Since real–world data sets about large populations of moving objects are difficult to obtain, to aid the development in spatio–temporal data management and analysis, Chapter 3 developed ST–ACTS, a Spatio–Temporal ACTivity Simulator. By using a number of real–world data sources and intuitive principles, ST–ACTS models some of the so far neglected social and geo–demographical aspects of mobility. Experimental results showed that ST–ACTS is able to effectively generate realistic spatio–temporal activities of a large population.

Some LBSes only make sense or are most effective if spatio–temporally closeby service requests are served in a group. Chapter 5 presented one example of such an LBS, namely a cab–sharing service. To provide an effective service the chapter proposed an algorithm for grouping closeby cab requests into a cab–share to minimize the total transportation cost, or equivalently maximize the savings. The algorithm was expressed as a sequence of simple SQL statements. Experiments based on simulated request data demonstrated that the proposed algorithm can effectively group together requests and thereby achieve significant savings.

In Chapter 6, the grouping algorithm was suggested as a generic building block to optimize large–scale collective transportation systems. To scale the algorithm to a large request stream, it was expressed as a continuous query in a Data Stream Management System (DSMS), the problem was sub–divided through static and adaptive spatial stream partitioning methods, and the computation was parallelized using the partitioning methods and the facilities of the DSMS. Experimental results showed that using the adaptive partitioning methods, the parallel implementations execute several orders of magnitude faster, yet achieve almost the same quality of grouping as their serial equivalent.

Location–Based Advertising (LBA), i.e., delivering *relevant* commercial information to mobile consumers, is regarded by many as one of the most lucrative business opportunities in LBSes. In order to give an indicator for the magnitude of this opportunity, in Chapter 7 an LBA framework and database was developed and used to estimate the capacity of the LBA channel. The proposed framework models *relevance* as a function of the *proximity* of the consumer to the service/product and the *interest* of the consumer. Two interest cases were considered: *explicit* and *implicit*. The chapter outlined several data mining techniques to infer the latter implicit interest. Experimental results showed that the channel capacity is indeed extremely large, which not only supports a business case, but also indicates the necessity of adequate user controls.

Whenever data about users is collected and analyzed, privacy naturally becomes a concern. To eliminate this concern, Chapter 8 proposed a grid–based framework to anonymize, collect, and analyze location data. Since the proposed anonymization is through spatio–temporal generalization, i.e., the locations of users can be narrowed down to sub–regions with a certain probability only, the analysis results are also probabilistic. To demonstrate the analysis component of the framework, the core data mining task of *finding dense spatio–temporal regions* was implemented. Experimental evaluations compared the results of the privacy–preserving mining method to its non–privacy–preserving equivalent, and found that the privacy–preserving mining method can find most of the patterns.

To entirely eliminate the uncertainty in the mining results, Chapter 9 presented a system for the privacy–preserving collection of *exact* trajectories. The system is composed of an untrusted server and clients communication via a P2P network. Location data is anonymized by the clients through data cloaking and data swapping techniques. Experimental results based on simulated movements of mobile users demonstrated that the proposed system is effective under reasonable conditions and anonymity/privacy settings.

In summary, Chapters 2 through 9 demonstrated that common data mining methods can be effectively extended to the spatio–temporal domain. The usefulness of the knowledge that the extended data mining methods can extract was demonstrated in two promising LBSes: cab/ride–sharing service and location–based advertising. Finally, to eliminate privacy concerns, systems were proposed for the privacy–preserving collection and analysis of location data. Thus, the thesis has shown that common data mining methods can be effectively extended to the spatio–temporal domain to discover useful knowledge for LBSes in a privacy–preserving manner.

10.2 Summary of Research Directions

Several directions for future work remain. As Chapter 2 demonstrated, pivoting is a general methodology that can extend rule mining methods to discover useful spatio–temporal rules. Spatio–temporally restricted rule mining was proposed to speed up the mining process by processing meaningful spatio–temporal subregions in isolation. Devising an automatic or semi–automatic system to determine these subregions is believed to be an interesting research direction.

As Chapter 4 demonstrated, spatio–temporal generalization is an effective method to discretize the spatio–temporal domain. Frequent itemset mining methods can effectively discover patterns in sets of generalized spatio–temporal items. Such a methodology however does have some shortcomings. First, closeby location measurements may fall in neighboring, but *different* spatio–temporal regions. This reduces the support of some patterns and potentially eliminate their discovery. As suggested in Chapter 4, road network based spatio–temporal generalizations is one approach to overcome this problem. In situations where the movements of objects is not confined to a road network, future work could consider the multi–grid approach, outlined in Chapter 8, as a way of reducing or eliminating the pattern support loss. Second, frequent itemset mining methods treat two spatio–temporally generalized items irrespective of their spatio–temporal proximity. As a consequence, mining results will likely contain several spatio–temporally very similar patterns, which could be represented by a single pattern. While post–analysis can aggregate the similar patterns, it is not an optimal approach as considering spatio–temporal item–proximity during the mining can speed up the mining process and eliminate some of the pattern support loss. Hence, devising effective frequent itemset mining methods that consider spatio–temporal item–proximity is believed to be an interesting future research direction. Mining based on dimensional hierarchies could be helpful to determine spatio–temporal item–proximity. The hybrid method to discover long, sharable routes in trajectories was demonstrated to be an effective trajectory analysis tool. However, as it was demonstrated on simple examples, due to the hybrid modelling of trajectories the method might not discover all patterns, or due to the approximated trajectories might discover false patterns. Evaluating the accuracy of the hybrid method is considered to be a valuable extension to the results of Chapter 4.

The simulator in Chapter 3 models only some of the physical aspects of mobility. Integrating the output of ST–ACTS as an input to a sophisticated network–based moving object simulator is believed to yield synthetic data sets that could aid the development in telematics, intelligent transportation systems, and location–based services. Furthermore, the modelling capabilities of ST–ACTS can be extended to generate even more realistic activities of users. Some possible extensions are as follows. First, given the available real–world data sets, ST–ACTS models activities in terms of daily activity probabilities and spatio–temporal activity constraints. While this modelling approach allows ST–ACTS to generate data with spatio–temporal activity distributions that correspond to the probabilities and obey the constraints, the generated activity sequences only exhibit limited temporal regularities. Extending ST–ACTS to model such temporal *sequential* regularities, for example through hidden Markov models or sequential patterns, is believed to be a useful extension to ST–ACTS. Second, since daily activity probabilities are derived for conzoom® types, i.e., groups of simulated persons, personal preferences are not modelled. Since data mining can be applied to personalize LBSes, modelling personal preferences is believed to be a useful extension to ST–ACTS.

Chapters 5 and 6 considered the vehicle–sharing problem and proposed a generic trip grouping algorithm and implementations that can be applied to optimize collective, door–to–door transportation systems. The proposed trip grouping algorithm uses a number of heuristic and approximation to derive near–optimal solutions to the vehicle–sharing problem. Devising new heuristic–based algorithms or applying common optimization methods (clustering, genetic algorithms) to derive even closer to optimal solutions is believed to be an interesting direction for research. Mainly to preserve clarity, the proposed trip grouping algorithm was presented in its simplest form, however the following improvements could be considered in a more complex version. First, the current equal–share cost model could be altered to take into account the costs of the shared trip parts versus the costs of "detours". Second, the basic algorithm could be altered to accommodate for individually defined passenger capacities of vehicles and minimum savings of requests. Third, the basic algorithm could be altered to handle in–route grouping, i.e., assigning requests to already active but not fully–occupied vehicle–shares. Finally, the grouping algorithm when applied to facilitate a real–world application should perform the optimizations based on road network distances.

In Chapter 6, parallelization through spatial partitioning of the request stream was an effective method to scale the computationally intensive trip grouping algorithm. In general, parallelization through spatial partitioning of streams, especially through density–based spatial partitioning, is believed to be an effective method to scale up computationally demanding spatial analysis tasks. Hence, implementing density–based spatial partitioning methods and testing their effectiveness is believed to be a interesting future research direction.

The LBA framework and database, proposed in Chapter 7, models implicit relevance through a simple scoring model which is based on a consumer segmentation that divides users into 29 different consumer groups. However, in real life, no two users' interests are *exactly* the same. Hence, altering the scoring model to include the *personal interests* of the individual mobile user, which are derived from his/her historical behavior, would allow targeting the individual user with even more *relevant* and *personalized* mobile ads. As outlined in the chapter, such personal interests can easily be captured by analyzing for example the type of businesses the user has previously visited or the user's reactions to previously received mobile ads. To evaluate the effects of personal interest scoring, the simulations of mobile user movements have to model personal interests and the possible influence of mobile ads on the future movements of mobile users. Hence, a full, bi–directional integration of ST–ACTS and the LBA framework is believed to be a fruitful research direction. Such an integration would allow to evaluate the accuracy of several different scoring models.

Chapter 8 proposed a grid–based framework to anonymize, collect, and analyze location data. The framework can be extended in a number of ways. First, additional grid–based anonymization policies can be devised. Second, the data mining component of the framework can be extended to include other, more complex data mining tasks. Finally, the multi–grid approach outlined in the chapter would have the following advantages. First, it would reduce or eliminate the boundary effects due to spatio–temporal generalization. Second, it would increase the spatio–temporal resolution of patterns while guaranteeing the same privacy to the users as the current single–grid based anonymization framework. Extending the framework in any of the above ways are believed to be interesting future research directions.

While Chapter 9 proposed a complete system for the privacy–preserving collection of *exact* trajectories, several future directions remain. First, different kinds of system architectures could be considered, for example a P2P network with hotspots. Second, the implementation and the large–scale real–world deployment of a full system could be considered. Third, devising effective methods for neighborhood detection in untrusted P2P network could be considered. Finally, the proposed system could be theoretically evaluated from the point of view of robustness and system performance in relation to privacy protection.

Based on the findings of and the topics investigated by the thesis, the following are high–level ideas for future research directions. Spatio–temporal generalization is an effective way to discretize the spatio–temporal domain, which in turn can be analyzed by common data mining techniques. A major drawback of the approach is the pattern support loss due to boundary effects. Addressing this shortcoming of the approach, through for example a multi–grid approach, or multi–level mining of dimensional hierarchies, are believed to be two promising future research directions.

The thesis mainly considered the extension of frequent itemset mining and rule mining methods to the spatio–temporal domain. Future research could consider extending clustering, another core data mining method, to the spatio–temporal domain. The clustering of trajectories, or, in relation to LBA, the clustering of activity sequences, could be considered. While spatio–temporal generalization could be applied prior to clustering, the so generalized trajectories / activity sequences have at least one unique characteristic: they are not of equal length. Two possible approaches to normalize such sequences are via Hidden Markov Models (HMM) and frequent itemset mining. In the HMM approach, an HMM could be constructed for every sequence, and using a meaningful distance metric between two HMMs, the HMMs could be clustered. In the frequent itemset mining approach, patterns could be mined among sequences. Then, using the patterns as positive / negative indicators, sequences could be mapped to a high–dimensional feature space for clustering. Using either one of the normalization approaches prior to clustering are believed to be interesting future directions.

While spatio–temporal activity sequences of mobile users contain mobility patterns that can be analyzed by some of the methods proposed in this thesis, they also contain *consumer* and *social* patterns. Methods to extract these latter patterns could consider the following. First, while a particular activity performed by a mobile user in Copenhagen, such as “bar–hopping” can be tied to a specific location in space, performing such an activity describes a consumer / social behavior that is independent of where the activity is performed, i.e., a person doing the same in Aalborg is similar to the person in Copenhagen. Hence, the analysis of *semantic* locations, i.e., places, are believed to be an interesting direction for future research. Second, while the things we do certainly define to some degree who we are, *whom* we do those things with are equally influential. Analyzing the interactions between mobile users in relation to the activities or places could not only have several interesting applications in social LBSes, but could provide valuable knowledge to social sciences, and hence it is believed to be an extremely interesting future research direction.

Bibliography

[1] R. Agrawal, T. Imilienski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proc. of the International Conference on Management of Data, SIGMOD*, pp. 207–216, SIGMOD Record 22(2), 1993.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th International Conference on Very Large Data Bases, VLDB*, pp. 487–499, Morgan Kaufmann, 1994.

[3] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proc. of the 11th International Conference on Data Engineering, ICDE*, pp. 3–14, IEEE Computer Society, 1995.

[4] R. Agrawal and R. Srikant. Privacy–Preserving Data Mining. In *Proc. of the International Conference on Management of Data, SIGMOD*, pp. 439–450, SIGMOD Record 29(2), 2000.

[5] S. J. Barnes and E. Scornavacca. Mobile Marketing: The Role of Permission and Acceptance. *International Journal of Mobile Communications, IJMC*, 2(2):128-139, Inderscience Publishers, 2004

[6] J. L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, ACM, 1975.

[7] J. L. Bentley and M. I. Shamos. Divide–and–Conquer in Multidimensional Space. In *Proc. of the 8th Annual ACM Symposium on Theory of Computing, ACM–STOC*, pp. 220–230, ACM, 1976.

[8] T. Brinkhoff. A Framework for Generating Network–Based Moving Objects. *Geoinformatica*, 6(2):153–180, Springer, 2002.

[9] D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proc. of the 17th International Conference on Data Engineering, ICDE*, pp. 443-452, IEEE Computer Society, 2001.

[10] CARLOS Ride–Sharing System: `http://www.carlos.ch/` (last accessed 20/11/2007)

[11] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, et al. Scalable Distributed Stream Processing. In *Proc. of the 1st Biennial Conference on Innovative Data Systems Research, CIDR*, Online Proceedings, 2003.

[12] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *Proc. of the 6th Workshop on Privacy Enhancing Technologies, PET*, volume 4258/2006 of Lecture Notes in Computer Science, Springer, 2006.

[13] C. -Y. Chow, M. F. Mokbel, and X. Liu. A Peer–to–Peer Spatial Cloaking Algorithm for Anonymous Location–based Services. In *Proc. of the 14th ACM International Symposium on Geographic Information Systems, ACM–GIS*, pp. 171–178, ACM, 2006.

[14] A. Civilis, C. S. Jensen, and S. Pakalnis. Techniques for Efficient Road–Network–Based Tracking of Moving Objects. *IEEE Transactions on Knowledge and Data Engineering, TKDE*, 17(5):698–712, IEEE Educational Activities Department, 2005.

[15] E. B. Cleff and G. Gidófalvi. Legal Aspects of a Location–Based Mobile Advertising Platform. To appear in the *International Journal of Intellectual Property Management, IJIPM*, 18 pages, 2008.

[16] T. H. Cormen, et. al. *Introduction to Algorithms.* (2nd Ed.). MIT Press and McGraw–Hill, 2001.

[17] T. G. Crainic, F. Malucelli, and M. Nonato. Flexible Many–to–Few + Few–to–Many = An Almost Personalized Transit System. In *Proc. of the 4th Triennial Symposium on Transportation Aanlysis, TRISTAN*, pp. 435–440, 2001.

[18] ESRI Business Analyst: `http://www.esri.com/businessanalyst/` (last accessed 20/11/2007)

[19] Experian: `http://www.experiangroup.com/` (last accessed 20/11/2007)

[20] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A Density–Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, KDD*, pp. 226–231, AAAI Press, 1996.

[21] M. Ester, H.-P. Kriegel, and J. Sander. Spatial Data Mining: A Database Approach. In *Proc. of the 5th International Symposium on Spatial Databases, SSD*, pp. 47–66, Springer, 1997.

[22] M. Ester, A. Frommelt, H.-P. Kriegel, and J. Sander. Algorithms for Characterization and Trend Detection in Spatial Databases. In *Proc. of the 4th International Conference on Knowledge Discovery and Data Mining, KDD*, pp. 44–50, AAAI Press, 1998.

[23] R. Finkel and J.L. Bentley. Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4(1):1–9, Springer, 1974.

[24] P. Fox, D. Rezania, J. Wareham, and E. Christiaanse. Will Mobiles Dream of Electric Sheep? Expectations of the New Generation of Mobile Users: Misfits with Practice and Research. In *Proc. of the 2006 International Conference on Mobile Business, ICMB*, p. 44, IEEE Computer Society, 2006.

[25] G. A. Frank and D. F. Stanat. Parallel Architecture for k–d Trees. Technical report, North Carolina University at Chapel Hill Dept. of Computer Science, May 1988.

[26] The Gallup Organization: `http://www.gallup.com/` (last accessed 20/11/2007)

[27] J. M. Gartenberg, C. Matiesanu, and N. Scevak. US Mobile Marketing Forecast, 2006 to 2011, Vision Report, JupiterResearch, USA, October 2006.

[28] Geomatic ApS – Center for Geoinformatik: `http://www.geomatic.dk/` (last accessed 20/11/2007)

[29] M. Gebski and R. K. Wong. A New Approach for Cluster Detection for Large Datasets with High Dimensionality. In *Proc. of the 7th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, volume 3589 of Lecture Notes in computer Science, pp. 498–508, Springer, 2005.

[30] G. Gidófalvi and T. B. Pedersen. Spatio–Temporal Rule Mining: Issues and Techniques. In *Proc. of the 7th International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, volume 3589 of Lecture Notes in computer Science, pp. 275–284, Springer, 2005.

[31] G. Gidófalvi and T. B. Pedersen. ST–ACTS: A Spatio–Temporal Activity Simulator. In *Proc. of the 14th ACM International Symposium on Geographic Information Systems, ACM–GIS*, pp. 155–162, ACM, 2006.

[32] G. Gidófalvi and T. B. Pedersen. Mining Long, Sharable Patterns in Trajectories of Moving Objects. In *Proc. of the 3rd Workshop on Spatio–Temporal Database Management, STDBM*, volume 174 of Online Proceedings of CEUR–WS, pp. 49–58, CEUR–WS, 2006.

[33] G. Gidófalvi and T. B. Pedersen. Cab–Sharing: An Effective, Door–To–Door, On–Demand Transportation Service. In *Proc. of the 6th European Congress and Exhibition on Intelligent Transport Systems and Services, ITS*, 2007.

[34] G. Gidófalvi, H. R. Larsen, and T. B. Pedersen. Estimating the Capacity of the Location–Based Advertising Channel. In *Proc. of the 2007 International Conference on Mobile Business, ICMB*, pp. 2, IEEE Computer Society, 2007.

[35] G. Gidófalvi, X. Huang, and T. B. Pedersen. Privacy–Preserving Data Mining on Moving Object Trajectories. In *Proc. of the 8th International Conference on Mobile Data Management, MDM*, 2007.

[36] G. Gidófalvi, X. Huang, and T. B. Pedersen. Privacy–Preserving Data Mining on Moving Object Trajectories. A DB Technical Report, 2007: `http://www.cs.aau.dk/DBTR/DBPublications/DBTR--19.pdf` (last accessed 20/11/2007)

[37] G. Gidófalvi, T. B. Pedersen, T. Risch, and E. Zeitler. Highly Scalable Trip Grouping for Large–Scale Collective Transportation Systems. To appear in *Proc. of the 11th International Conference on Extending Database Technology, EDBT*, 12 pages, 2008.

[38] G. Gidófalvi, H. R. Larsen, and T. B. Pedersen. Estimating the Capacity of the Location–Based Advertising Channel. To appear in *International Journal of Mobile Communications, IJMC*, 18 pages, Inderscience Publishers, 2008.

[39] G. Gidófalvi and T. B. Pedersen. Mining Long, Sharable Patterns in Trajectories of Moving Objects. To apear in *Geoinformatica*, 35 pages, 2008.

[40] B. Goethals. Survey on Frequent Pattern Mining. Online at: `http://citeseer.ist.psu.edu/goethals03survey.html` (last accessed 20/11/2007)

[41] M.L Gosseling and S. Doherty, eds., *Integrated Land–Use and Transportation Models – Behavioural Foundations*, Elsevier, 2005.

[42] K. Gouda and M. J. Zaki. GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets. *Data Mining and Knowledge Discovery*, 11(3):223–242, Springer, 2005.

[43] B. Gedik and L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *Proc. of the 25th International Conference on Distributed Computing Systems , ICDCS*, pp. 620–629, IEEE Computer Society, 2005.

[44] M. Gruteser and D. Grunwald. Anonymous Usage of Location–Based Services Through Spatial and Temporal Cloaking. In *Proc. of the 1st International Conference on Mobile Systems, Applications and Services, MobiSys*, pp. 31–42, ACM, 2003.

[45] A. Guttman. R–trees: A Dynamic Index Structure for Spatial Searching. In *Proc. of the International Conference on Management of Data, SIGMOD*, pp. 47–57, SIGMOD Record 14(2), 1984.

[46] T. Hägerstrand. "Space, Time and Human Conditions." In *Dynamic allocation of urban space*, eds. A. Karlqvist et al. Lexington: Saxon House Lexington Book, 1975.

[47] M. Hadjieleftheriou, G. Kollios, D. Gunopulos, and V. J. Tsotras. On–Line Discovery of Dense Areas in Spatio–Temporal Databases. In *Proc. of the 8th International Symposium on Spatial and Temporal Databases, SSTD*, volume 2750 of Lecture Notes in Computer Science, pp. 306–324, Springer, 2003.

[48] J. Han, K. Koperski, N. Stefanovic. GeoMiner: A System Prototype for Spatial Data Mining. In *Proc. of the International Conference on Management of Data, SIGMOD*, pp. 553–556, SIGMOD Record 26(2), 1997.

[49] J. Han and M. Kamber. *Data Mining: Concepts and Techniques* (2nd Ed.). Morgan Kaufmann, 2006.

[50] K. Heinonen and T. Strandvik. Consumer Responsiveness to Mobile Marketing. *International Journal of Mobile Communications, IJMC*, 5(6):603–617, Inderscience Publishers, 2007.

[51] H. Hu and D. L. Lee. GAMMA: A Framework for Moving Object Simulation. In *Proc. of the 9th International Symposium on Spatial and Temporal Databases, SSTD*, volume 3633 of Lecture Notes in Computer Science, pp. 37–54, Springer, 2005.

[52] Hitchsters: `http://www.hitchsters.com/` (last accessed 20/11/2007)

[53] INFATI. The INFATI Project Web Site: `http://www.infati.dk/uk` (last accessed 20/11/2007)

[54] M. Ivanove and T. Risch. Customizable Parallel Execution of Scientific Stream Queries. In *Proc. of the 31st International Conference on Very Large Data Bases, VLDB*, pp. 157–168, ACM, 2005.

[55] C. S. Jensen, D. Pfoser, and Y. Theodoridis. Novel Approaches to the Indexing of Moving Object Trajectories. In *Proc. of the 26th International Conference on Very Large Data Bases, VLDB*, pp. 395–406, ACM, 2000.

[56] C. S. Jensen. Research Challenges in Location–Enabled M–Services. In *Proc. of the 4th International Conference on Mobile Data Management, MDM*, volume 2574 of Lecture Notes in Computer Science, pp. 3–7, Springer, 2003.

[57] C. S. Jensen, A. Kligys, T. B. Pedersen, and I. Timko. Multidimensional Data Modeling for Location–Based Services. *International Journal on Very Large Data Bases, VLDB*, 13(1):1–21, Springer, 2004.

[58] C. S. Jensen, H. Lahrmann, S. Pakalnis, and S. Runge. The INFATI data. Time Center TR-79, 2004: `http://www.cs.aau.dk/TimeCenter/` (last accessed 20/11/2007)

[59] C. S. Jensen, D. Lin, B. C. Ooi, and R. Zhang. Effective Density Queries on Continuously Moving Objects. In *Proc. of the 22nd International Conference on Data Engineering, ICDE*, pp. 71–81, IEEE Computer Society, 2006.

[60] M. Kaplan. Mobile Advertising and Marketing: Market Analysis and Forecasts 2006–2011. Vision report, visiongain™, March 2006.

[61] H. Komulainen, A. Ristola, and J. Still. Mobile Advertising in the Eyes of Retailers and Consumers – Empirical Evidence from a Real–Life Experiment. In *Proc. of the 2006 International Conference on Mobile Business, ICMB*, p. 37, IEEE Computer Society, 2006.

[62] B. Kölmel. Location Based Advertising. In *Proc. of the 2002 International Conference on Mobile Business, ICMB*, IEEE Computer Society, 2002.

[63] K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Databases. In *Proc. of the 4th International Symposium on Spatial Databases, SSD*, volume 951 of Lecture Notes in Computer Science, pp. 47–66, Springer, 1995.

[64] R. Kuntschke, B. Stegmaier, A. Kemper, and A. Reiser. StreamGlobe: Processing and Sharing Data Streams in Grid–based P2P Infrastructures. In *Proc. of the 31st International Conference on Very Large Data Bases, VLDB*, pp. 1259–1262, ACM, 2005.

[65] T. Lee and J. Jun. The Role of Contextual Marketing Offer in Mobile Commerce Acceptance: Comparison between Mobile Commerce Users and Nonusers. *International Journal of Mobile Communications*, 5(3):339–356, Inderscience Publishers, 2007.

[66] M. Leppaniemi and H. Karjaluoto. Factors Influencing Consumers' Willingness to Accept Mobile Advertising: A Conceptual Model. *International Journal of Mobile Communications, IJMC*, 3(3):197–213, Inderscience Publishers, 2005.

[67] Y. Li, X. S. Wang, and S. Jajodia. Discovering Temporal Patterns in Multiple Granularities. In *Proc. of the 1st International Workshop on Temporal, Spatial, and Spatio–Temporal Data Mining, TSDM*, volume 2007 of Lecture Notes in computer Science, pp. 5–19, Springer, 2000.

[68] B. Liu, Y. Zhu, M. Jbantova, B. Momberger, and E. A. Rundensteiner. A Dynamically Adaptive Distributed System for Processing Complex Continuous Queries. In *Proc. of the 31st International Conference on Very Large Data Bases, VLDB*, pp. 1338–1341, ACM, 2005.

[69] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l–Diversity: Privacy Beyond k–Anonymity. In *Proc. of the 22nd International Conference on Data Engineering, ICDE*, p. 26, IEEE Computer Society, 2006.

[70] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung. Mining, Indexing, and Querying Historical Spatiotemporal Data. In *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, pp. 236–245, ACM, 2004.

[71] F. J. Massey. The Kolmogorov–Smirnov Test of Goodness of Fit. *Journal of the American Statistical Association, JASA*, 46(253):68–78, American Statistical Association, 1951.

[72] M. Merisavo, J. Vesanen, A. Arponen, Ss Kajalo, and M. Raulas. The Effectiveness of Targeted Mobile Advertising in Selling Mobile Services: An Empirical Study. *International Journal of Mobile Communications, IJMC*, 4(2):119–127, Inderscience Publishers, 2006.

[73] Mobile Marketing Association: `http://www.mmaglobal.com/` (last accessed 20/11/2007)

[74] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proc. of the*

32nd International Conference on Very Large Data Bases, VLDB, pp. 763–774, ACM, 2006.

[75] Oracle Spatial & Oracle Locator: Location Features for Oracle Database 10g: `http://www.oracle.com/technology/products/spatial/` (last accessed 20/11/2007)

[76] J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In *Proc. of the 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD*, pp. 11-20, ACM, 2000.

[77] D. Pfoser and Y. Theodoridis. Generating Semantics–Based Trajectories of Moving Objects. In *Proc. of International Workshop on Emerging Technologies for Geo–Based Applications*, pp. 59–76, 2000.

[78] K. Pousttchi, D. G. Wiedemann. A Contribution to Theory Building for Mobile Marketing: Categorizing Mobile Marketing Campaigns through Case Study Research. In *Proc. of the 2006 International Conference on Mobile Business, ICMB*, p. 1, IEEE Computer Society, 2006.

[79] M. Quddus, W. Ochieng, L. Zhao, and R. Noland. A General Map Matching Algorithm for Transport Telematics Applications. *GPS Solutions Journal*, 7(3):157-167, 2003.

[80] J. F. Roddick, M. J. Egenhofer, E. Hoel, D. Papadias, and B. Salzberg. Spatial, Temporal and Spatio–Temporal Databases–Hot Issues and Directions for PhD Research. In *Proc. of the 8th International Symposium on Spatial and Temporal Databases, SSTD*, volume 2750 of Lecture Notes in Computer Science, pp. 1–6, Springer, 2003.

[81] J.-M. Saglio and J. Moreira. Oporto: A Realistic Scenario Generator for Moving Objects. In *Proc. of the 10th Invernational Workshop on Database and Expert Systems Applications, DEXA*, volume 1677 of Lecture Notes In Computer Science, pp. 426–432, Springer, 1999.

[82] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[83] T. Sellis. Research Issues in Spatio–temporal Database Systems. In *Proc. of the 6th International Symposium on Spatial Databases, SSD*, pp. 5–11, Springer, 1999.

[84] X. Shang, K.-U. Sattler, and I. Geist. Efficient Frequent Pattern Mining in Relational Databases. In *Proc. of the Workshops on "Learning, Knowledge Discovery, and Adaptivity" (Lernen, Wissensentdeckung und Adaptivität), LWA*, pp. 84–91, 2004.

[85] Statistics Denmark: `http://www.dst.dk/` (last accessed 20/11/2007)

[86] STM. Space, Time Man Project Web Site: `http://www.plan.aau.dk/~hhh/` (last accessed 20/11/2007)

[87] Standard SQL 1999, ISO/IEC 9075:1999.

[88] W. Stallings. *Cryptography and Network Security Principles and Practices* (4th Ed.). Prentice Hall. 2005.

[89] L. Sweeney. K–Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge–Based Systems, IJUFKS*, 10(5):557–570, World Scientific Publishing, 2002.

[90] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias. Spatio–Temporal Aggregation Using Sketches. In *Proc. of the 19th International Conference on Data Engineering, ICDE*, pp. 214–226, IEEE Computer Society, 2004.

[91] Y. Theodoridis, J. R. O. Silva, and M. A. Nascimento. On the Generation of Spatiotemporal Datasets. In *Proc. of the 6th International Symposium on Spatial Databases, SSD*, pp. 147–164, Springer, 1999.

[92] Transportation Problems: `http://www.di.unipi.it/optimize/transpo.html` (last accessed 20/11/2007)

[93] Taxibus – Intelligent Group Transportation. `http://www.taxibus.org.uk/index.html` (last accessed 20/11/2007)

[94] texxi – Transit Exchange XXIst Century. `http://www.taxxi.com` (last accessed 20/11/2007)

[95] I. Tsoukatos and D. Gunopulos. Efficient Mining of Spatiotemporal Patterns. In *Proc. of the 7th International Symposium on Spatial and Temporal Databases, SSTD*, volume 2121 of Lecture Notes in Computer Science, pp. 425–442, Springer, 2001.

[96] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State–of–the–art in Privacy Preserving Data Mining. *SIGMOD Record*, 33(1):50–57, ACM, 2004.

[97] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering Similar Multidimensional Trajectories. In *Proc. of the 18th International Conference on Data Engineering, ICDE*, pp. 673–685, IEEE Computer Society, 2002.

[98] J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, pp. 236–245, ACM, 2003.

[99] X. Wang, Y. Yin, and H. Yu. Finding Collisions in the Full SHA–1. In *Proc. of the 25th Annual International Cryptology Conference, CRYPTO*, volume 3621 of Lecture Notes in Computer Science, pp. 17–36, Springer, 2005.

[100] O. Wolfson, B. Xu, and H. Yin. Dissemination of Spatial–Temporal Information in Mobile Networks with Hotspots. In *Proc. of the 2nd International Workshop on Databases, Information Systems, and Peer–to–Peer Computing, DBISP2P*, volume 3367 of Lecture Notes in Computer Science, pp. 185–199, Springer, 2004.

[101] D. Xin, J. Han, X. Yan, and H. Cheng. Mining Compressed Frequent–Pattern Sets. In *Proc. of the 31st International Conference on Very Large Data Bases, VLDB*, pp. 709–720, ACM, 2005.

[102] Y. Xing, S. B. Zdonik, and J.-H. Hwang. Dynamic Load Distribution in the Borealis Stream Processor. In *Proc. of the 21st International Conference on Data Engineering, ICDE*, pp. 791–802, IEEE Computer Society, 2005.

[103] X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. In *Proc. of the 3rd SIAM International Conference on Data Mining, SDM*, pp. 166–177, SIAM, 2003.

[104] M. J. Zaki and C.J. Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining. In *Proc. of the 2nd SIAM International Conference on Data Mining, SDM*, pp. 71–80, SIAM, 2002.

[105] E. Zeitler and T. Risch. Processing High–Volume Stream Queries on a Supercomputer. In *Proc. of the 22nd International Conference on Data Engineering Workshops, ICDEW*, pp. 147, IEEE Computer Society, 2006.

[106] E. Zeitler and T. Risch. Using Stream Queries to Measure Communication Performance of a Parallel Computing Environment. In *Proc. of the 27th IEEE International Conference on Distributed Computing Systems Workshop, ICDCSW*, pp. 65–74, IEEE, Computer Society, 2007.

Summary in Danish

Danish Title: Spatio–Temporal Vidensopdagelse i Lokationsbaserede Services

De såkaldte Lokationsbaserede Services (LBS) vinder frem overalt, hjulpet af massive landvindinger inden for kommunikations– og informationsteknologi, såsom den voksende udbredelse af og præcision i GPS–enheder og udviklingen af mindre og mindre enheder til trådløs kommunikation. Innovative LBS'er integrerer viden om brugerne i de udbudte services. En sådan viden kan udledes ved at analysere oplysninger om hvor brugerne befinder sig. De data som anvendes, har to dimensioner, *sted* og *tid*, som begge er genstand for analyse.

Målsætningen for denne afhandling er tredelt: For det første at overføre udbredte vidensopdagelse–metoder til det spatio–temporale domæne. For det andet at demonstrere anvendeligheden og nytteværdien af disse metoder og den deraf afledte viden for to lovende LBS–eksempler. For det tredje at eliminere frygten for at afsløre personfølsomme oplysninger via spatio–temporal vidensopdagelse – det kan undgås igennem systemer som netop sikrer og beskytter privatsfæren omkring brugerne i dataindsamling og vidensopdagelse–delen.

Kapitel 2 introducerer en general metode, pivottering, som overfører en bredt anerkendt og anvendt vidensopdagelse–metode, regelopdagelse, til det spatio–temporale domæne. Med afsæt i en karakteristik af en vifte af reelle, konkrete datakilder udleder kapitel 2 desuden en taksonomi for spatio–temporale data, og eksempler på nytteværdien og anvendeligheden af disse regler gives. I kapitel 4 anvendes den spatio–temporale variant med henblik på at finde mønstre af lange, fælles rejseruter for objekter i bevægelse. Evalueringer af empiriske data viser at metoderne implementeret via højniveau SQL udgør effektive værktøjer til en analyse af lange, fælles rejsemønstre.

Virklige rejsedatasæt for en større population af objekter som rejser inden for et afgrænset geografisk område, er vanskeligt tilgængelige. Til at kompensere herfor redegør kapitel 3 for udviklingen af en Spatio–Temporal ACTivity Simulator (ST–ACTS). ST–ACTS inddrager en mængde reelle geostatistiske datakilder og intuitive principper for rumlige sammenhænge for effektivt at generere realistiske spatio–temporale aktivitetsmønstre for mobile brugere.

Kapitel 5 foreslår en konkret LBS til transportsektoren, nemlig taxideling. En effektiv løsning bygger på en unik spatio–temporal grupperingsalgoritme som implementeres som en sekvens af SQL–sætninger. Kapitel 6 identificerer en skaleringsflaskehals for denne algoritme. For at eliminere denne flaskehals omformes grupperingsalgoritmen sådan at den udtrykkes som en kontinuert strøm af forespørgsler til et system som specifikt håndterer datastrømme. Enkle og dog effektive spatio–temporal partitioneringsmetoder angives, som parallelliserer de nødvendige strømme af beregninger. Eksperimentelle resultater viser at parallellisering igennem adaptiv partitionering fører til massive tidsbesparelser uden væsentlig påvirkning af kvaliteten af grupperingen (af de rejsende objekter). Spatio–temporal datastrøm–partitionering forventes at udgøre en effektiv platform til skalering af beregningsintensive spatiale forespørgsler og analyser på datastrømme.

Lokationsbaseret Reklamering (Location–Based Advertising, LBA), distribution af relevant kommerciel information til mobile forbrugere, betragtes som en af de mest lovende forretningsmuligheder blandt LBS'er. I såhenseende beskriver kapitel 7 en udviklingsramme for en LBA platform og en LBA–database som kan bygges til at støtte og drifte håndteringen af mobile (lokationsbaseret) reklamer. Igennem en simuleret, men realistisk population af mobile forbrugere og et univers af mobile reklamer bruges LBA–databasen til at estimere kapaciteten af og dermed potentialet i den mobile reklame som distributionskanal. Estimaterne er lovende for formuleringen af en stærk forretningsplan, men afdækker samtidig nødvendigheden af at kunne håndtere brugerspecifikke profiler, ønsker og hensyn.

Så snart data om brugere indsamles og analyseres, bliver beskyttelse af personfølsomme oplysninger en nødvendighed. En tilgang til at imødekomme dette præsenteres i kapitel 8 – en cellebaseret aggregering sikrer, at stedfæstelsen af brugerne anonymiseres via spatio–temporal generalisering; dernæst formuleres et system til indsamling af og vidensopdagelse på anonyme, stedbestemte data. Eksperimentelle resultater viser, at beskyttelsen af brugernes privatsfære ikke forhindrer, at nyttige mønstre kan identificeres der – selv om de hviler på sandsynlighedsmodeller – vil være tilstrækkelig præcise til mange LBS'er.

Med henblik på at eliminere enhver usikkerhed om de igennem vidensopdagelse udledte resultater opstilles i kapitel 9 en tilgang til indsamling af eksakte rejsemønstre for objekter i bevægelse der bevarer brugernes privatsfære og anonymitet. Den foreslåede tilgang inddrager ingen betroede komponenter, og anonymisering foretages af klienterne i et P2P–netværk via datageneralisering og dataombytning. Realistiske simuleringer viser, at inden for rimelige rammer og med en beskyttelse af privatsfæren og anonymitet viser systemet sig at være effektivt.